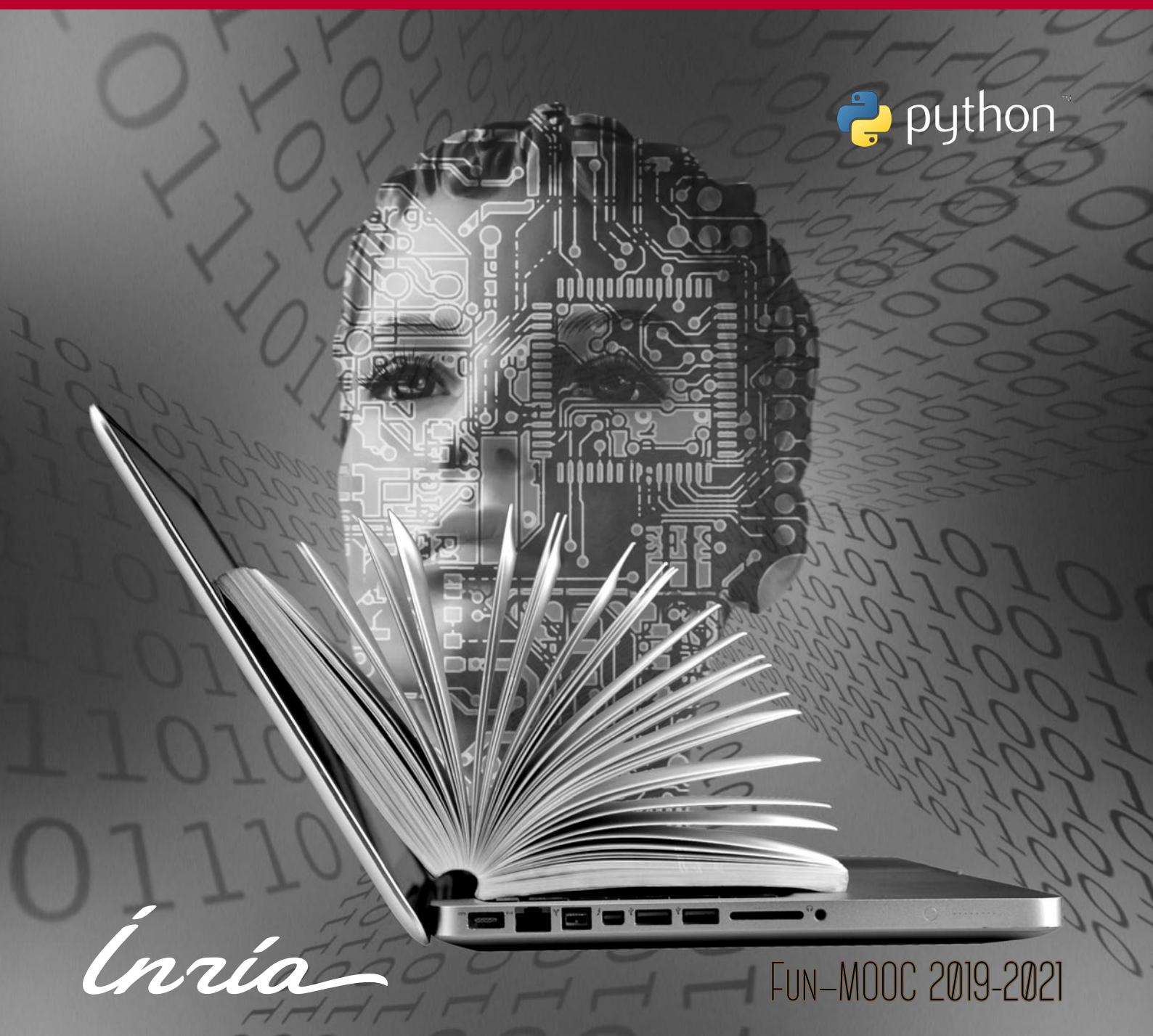


INFORMATIQUE ET CULTURE SCIENTIFIQUE DU NUMÉRIQUE

FORMATION ICN-SNT-PYTHON — MANUEL DES PARTICIPANTS



inria

FUN-MOOC 2019-2021



Ce document est réalisé au moyen de l'outil de composition de documents \LaTeX (\LaTeX Project Public License) — distribution $\text{\TeX} \text{Live}$, moteur \LaTeX — et de multiples extensions de style : The Comprehensive \TeX Archive Network (ressources) — \TeX-LaTeX Stack Exchange (site d'entraide de référence absolue).

Les images sont construites ou retouchées à l'aide des extensions graphiques PGF/TikZ ou des logiciels libres INKSCAPE et GIMP (GNU General Public License).

L'accès ouvert — voire libre — à l'ensemble de ces outils permet à tous l'élaboration de documents de qualité (agencement comme typographie). Chaque contribution en est ici chaleureusement remerciée.

Crédits photographiques et graphiques

Autant que faire se peut, les images et illustrations utilisées dans ce document sont du domaine public ou relèvent *a minima* de la licence Creative Commons CC0.

À cette fin, les infographistes ou détenteurs des droits à créditer sont : FLATICON, FREEPIK, FREESVG (nouvelle mouture du projet OPENCLIPARTS), PIXABAY, PUBLICDOMAINVECTORS, VEXELS, WIKIMEDIA COMMONS.

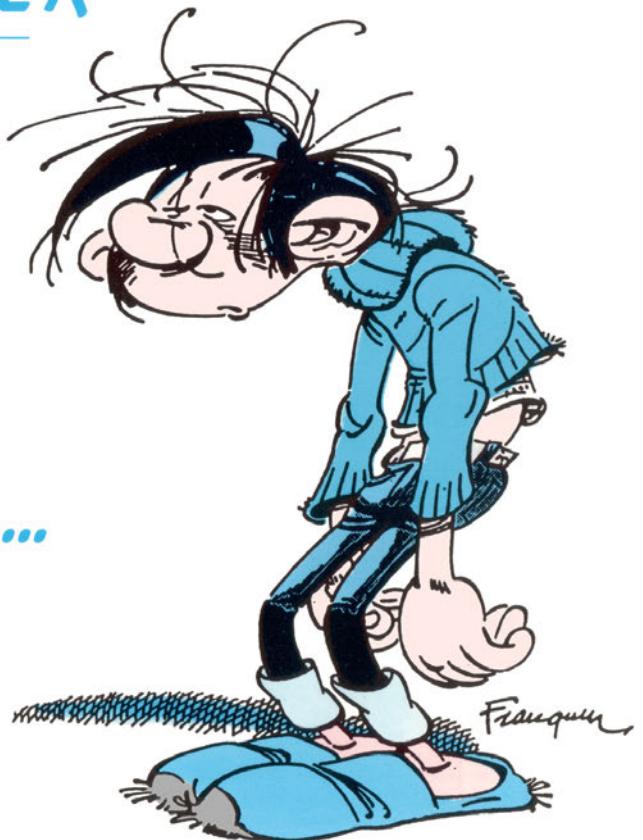
Pour ce qui concerne les éléments soumis à *copyright* — essentiellement issus de contenus extérieurs —, leur source et leurs droits associés sont mentionnés systématiquement aux abords des images et illustrations.

Au sujet des logotypes de projets et de marques déposées — officiels comme stylisés —, ils appartiennent à leurs éditeurs respectifs et sont affichés tels quels, considérant que cela contribue à leur notoriété.

Apport rédactionnel : Vincent DOUTAUT

NE PAS DÉRANGER

*UN GÉNIE
DÉFIE
L'ORDINATEUR ...*



AUTEURS

Ce document transcrit les Mooc réalisés par le LEARNING LAB INRIA, avec le soutien du Ministère de l'éducation nationale et de la jeunesse, en partenariat avec le projet CLASS'CODE et l'Université Côte d'Azur.

CONCEPTION ET RÉALISATION DES CONTENUS

- ▶ Frédéric ALEXANDRE — Chercheur en informatique, INRIA.
- ▶ Sylvie BOLDO — Chercheuse en informatique, INRIA.
- ▶ Isabelle CHRISMANT — Enseignante-Chercheuse en informatique, TÉLÉCOM Nancy, Université de Lorraine.
- ▶ Marie DUFLOT-KREMER — Enseignante-Chercheuse en informatique à l'Université de Lorraine.
- ▶ Catherine FARON ZUCKER — Enseignante-Chercheuse en informatique, Université de Nice.
- ▶ Brice GOGLIN — Chercheur en informatique, INRIA.
- ▶ Erwan KERRIEN — Chercheur en informatique, INRIA.
- ▶ Arnaud LEGOUT — Chercheur en informatique, INRIA.
- ▶ Dorian MAZAURIC — Chercheur en informatique, INRIA.
- ▶ Thierry PARMENTELAT — Ingénieur de recherche, INRIA.
- ▶ Charles POULMAIRE — Professeur de mathématiques au lycée Van Gogh d'AUBERGENVILLE.
- ▶ Vincent ROCA — Chercheur en informatique, INRIA.
- ▶ David ROCHE — Professeur de sciences physiques et d'informatique au lycée Guillaume-Fichet de BONNEVILLE.
- ▶ Thierry VIÉVILLE — Chercheur en informatique, INRIA.

INGÉNIERIE PÉDAGOGIQUE INRIA LEARNING LAB

- ▶ Laurence FARHI — Ingénierie pédagogique, Learning Lab, INRIA.
- ▶ Christelle MARIAIS — Ingénierie pédagogique, Learning Lab, INRIA.

CONTRIBUTIONS

- ▶ Mireille ARNAUD — Professeure de mathématiques, Académie de Nice.
- ▶ Pascal BARBIER — Chercheur en géomatique, ENSG.
- ▶ Sabrina BARNABÉ — Logisticienne, SNJazur.fr.
- ▶ Ikram CHRAIBI-KAADOUD — Docteure en informatique, Onepointcom.
- ▶ Xavier COLLILIEUX — Chercheur en géodésie, LAREG.
- ▶ Laurent COOPER — Professeur de mathématiques, Académie de Grenoble.
- ▶ Martine COURBIN — Ingénierie documentaliste et en médiation scientifique, INRIA.
- ▶ Vivien DA CRUZ — Professeur de technologie et formateur, DANE de Versailles.
- ▶ Maxime FOURNY — Professeur de mathématiques, Académie de Besançon.
- ▶ Pascal GUITTON — Enseignant-Chercheur en informatique, Université de Bordeaux.
- ▶ Loïc JOSSE — Professeur en sciences industrielles pour l'ingénieur, en informatique et sciences du numérique.
- ▶ Alexis LECOMTE — Professeur de mathématiques, Académie de Rouen.

RÉALISATION ET PRODUCTION DES VIDÉOS

- ▶ Philippe AIGOUY — Créeur transmédia, VSP Production.
- ▶ Jade BECKER — Conceptrice pédagogique et animatrice, MAGIC MAKERS.
- ▶ Jean-Jacques BIRGÉ — Designer sonore.
- ▶ Mikaël Cixous — Graphiste.
- ▶ Guillaume CLÉMENCIN — Comédien.
- ▶ Sonia CRUCHON — Conceptrice, cheffe de projet multimédia, réalisatrice de films et voix-off.
- ▶ Patrick DEMIRO — Réalisateur audiovisuel, CAP Production.
- ▶ Olivier GARDE — Réalisateur audiovisuel, Manhattan Studio Production.
- ▶ François GARDE — Réalisateur audiovisuel, Manhattan Studio Production.
- ▶ Nicolas LE DU — Monteur.
- ▶ Sophie DE QUATREBARBES — Coordinatrice du programme CLASS'CODE, conceptrice et réalisatrice multimédia, S24B.
- ▶ Julie STEIN — Chargée de projet numérique éducatif, Ligue de l'enseignement.



TABLE DES MATIÈRES

1 | VOUS AVEZ DIT NUMÉRIQUE ?

PREMIÈRE PARTIE — INFORMATIQUE, CRÉATION NUMÉRIQUE

11 | CHAPITRE 1 NUMÉRIQUE ET SCIENCES DU RÉEL

- 1 Entrailles d'un ordinateur
 - 1.1 Démontage et descriptif 11
 - 1.1.1 Périphériques — 1.1.2 Unité centrale
 - 1.2 Montage et fonctionnalités 15
 - 1.2.1 Projet RASPBERRY PI — 1.2.2 Architecture de VON NEUMANN
- 2 Système d'exploitation
 - 2.1 OS : trois idées-forces 23
 - 2.1.1 Gestion mémoire — 2.1.2 Gestion processeur — 2.1.3 Appels système
 - 2.2 À quoi sert un OS? 26
- 3 Interface humain-machine
 - 3.1 Interactivité 29
 - 3.1.1 Bref historique — 3.1.2 Concept d'utilisabilité — 3.1.3 Procédure de conception — 3.1.4 IHM et technologies — 3.1.5 Avenir de l'IHM
- 3.2 50 ans d'IHM : retour vers le futur 34
- 4 Loi, économie et éthique
 - 4.1 Numérique : loi et vie privée 47
 - 4.1.1 Données — 4.1.2 Contrôle et exploitation — 4.1.3 Automatisation — 4.1.4 Éducation et transparence
 - 4.2 Ville numérique et vie privée 51
 - 4.3 Comprendre les logiciels libres 53
 - 4.3.1 Fondements et principes — 4.3.2 Modèle économique
 - 4.4 Paradigme du logiciel libre 64
- 5 Que faire de ces ressources? Quiz

75 | CHAPITRE 2 WEB ET USAGES SOCIÉTAUX

- 1 Web et réseaux sociaux
 - 1.1 Avènement du Web social 75
 - 1.1.1 World Wide Web — 1.1.2 Réseaux sociaux — 1.1.3 Modélisation — 1.1.4 Enjeux
 - 1.2 Filtrage selon FACEBOOK 80
- 2 Calcul dans les nuages
 - 2.1 Quid du cloud? 81
 - 2.1.1 Infrastructure — 2.1.2 Problématique
 - 2.2 Cozy Cloud : nécessaire vertu 83

3	Monnaies cryptographiques	
3.1	Du <i>bitcoin</i> à la <i>blockchain</i>	86
3.1.1	Monnaie numérique	—
3.1.2	Impacts	
3.2	Startups bien de chez nous...	89
4	Numérique en question(s)	
4.1	Penser le numérique	91
4.1.1	Ontologie	—
4.1.2	Transformations	
4.2	Être et écran	94
5	Que faire de ces ressources? Quiz	

99

CHAPITRE 3

INTELLIGENCE ARTIFICIELLE ET ROBOTIQUE

1	Intelligence artificielle démythifiée	
1.1	Historique et enjeux	100
1.1.1	Origines	—
1.1.2	Enjeux actuels	
1.2	Intelligence artificielle débraillée	102
2	Découverte de la robotique	
2.1	Mécanismes des robots	104
2.1.1	Qu'est-ce qu'un robot?	—
2.1.2	Autonomie	
2.1.3	Adaptation et apprentissage	
2.2	Différences de finalités	106
2.2.1	Travail et exploration	—
2.2.2	Assistance à la personne	—
2.2.3	Modéliser le vivant	
2.3	Éveil des bébés robots	109
3	Impact sociétal de la robotique	
3.1	Robotique et éthique	116
3.1.1	Exemple du véhicule autonome	—
3.1.2	Collaboration homme-robot	—
3.1.3	Enjeux de la robotique médicale	
3.2	Informaticiens et éthique	119
4	<i>Deep learning</i>	
4.1	Comprendre le <i>deep learning</i>	123
4.2	Apprentissage automatique	124
5	Que faire de ces ressources? Quiz	

DEUXIÈME PARTIE – FONDEMENTS DE L'INFORMATIQUE

131

CHAPITRE 4

CODAGE BINAIRE

1	Représentation de l'information	
1.1	Électronique et logique	131
1.1.1	Système binaire	—
1.1.2	Opérateurs et composants logiques	
1.2	Nombre et texte	132
1.2.1	Assignation des nombres	—
1.2.2	Codification du texte	
2	Infographie et audionumérique	
2.1	Photographie et dessin	139
2.1.1	Images matricielles	—
2.1.2	Images vectorielles	
2.2	Son et musique	140
2.2.1	Caractérisation et analyse des sons	—
2.2.2	Quantification et échantillonnage	
3	Manipulation des données	
3.1	Compression	151
3.1.1	Problématique	—
3.1.2	Images	—
3.1.3	Vidéos	—
3.1.4	Sons	
3.2	Organisation et stockage	155
3.2.1	Type et codage	—
3.2.2	Structure de données	—
3.2.3	Représentation des données en mémoire	
3.3	Base de données	157
3.3.1	Modèle relationnel et requête SQL	—
3.3.2	Protection et partage	
4	Que faire de ces ressources? Quiz	

163

CHAPITRE 5

LANGAGES DE L'INFORMATIQUE

1	Algorithmique	
1.1	Notions fondamentales	163
1.1.1	Décomposition en tâches	—
1.1.2	Variables	—
1.1.3	Instructions élémentaires	
1.2	Culture algorithmique	167
1.2.1	Introduction aux graphes	—
1.2.2	Parcours de graphe	
2	Programmation	
2.1	Langages de programmation	171
2.1.1	Langage machine	—
2.1.2	Langage de haut niveau et compilation	
2.2	Paradigmes de programmation	173
2.2.1	Bogues	—
2.2.2	Humains et langages	
3	Que faire de ces ressources? Quiz	

TROISIÈME PARTIE – NUMÉRIQUE : CULTURE ET PRATIQUE

185

CHAPITRE 6 DONNÉES ET OBJETS

- 1 Données et traitements
 - 1.1 Découvrir la thématique 185
 - 1.1.1 Ancrage dans le réel — 1.1.2 Volet historique
 - 1.1.3 Explication des notions
 - 1.2 Réaliser des activités 190
- 2 Photographie numérique
 - 2.1 Découvrir la thématique 191
 - 2.1.1 Ancrage dans le réel — 2.1.2 Volet historique
 - 2.1.3 Explication des notions
 - 2.2 Réaliser des activités 196

- 3 Informatique embarquée et objets connectés
 - 3.1 Découvrir la thématique 197
 - 3.1.1 Ancrage dans le réel — 3.1.2 Volet historique
 - 3.1.3 Explication des notions
 - 3.2 Réaliser des activités 201
- 4 Localisation et cartographie
 - 4.1 Découvrir la thématique 202
 - 4.1.1 Ancrage dans le réel — 4.1.2 Volet historique
 - 4.1.3 Explication des notions
 - 4.2 Réaliser des activités 207

5 Que faire de ces ressources ? Quiz

215

CHAPITRE 7 RÉSEAUX ET ENVIRONNEMENT

- 1 Internet et les réseaux
 - 1.1 Découvrir la thématique 215
 - 1.1.1 Ancrage dans le réel — 1.1.2 Volet historique
 - 1.1.3 Explication des notions
 - 1.2 Réaliser des activités 219
- 2 Web et information
 - 2.1 Découvrir la thématique 220
 - 2.1.1 Ancrage dans le réel — 2.1.2 Volet historique
 - 2.1.3 Explication des notions
 - 2.2 Réaliser des activités 225
- 3 Réseaux sociaux
 - 3.1 Découvrir la thématique 226
 - 3.1.1 Ancrage dans le réel — 3.1.2 Volet historique
 - 3.1.3 Explication des notions

- 3.2 Réaliser des activités 231
- 4 Enjeux environnementaux
 - 4.1 Découvrir la thématique 232
 - 4.1.1 Ancrage dans le réel — 4.1.2 Numérique : menace ou espoir pour l'environnement?
 - 4.2 Impacts environnementaux 235
 - 4.2.1 Aspects de fabrication — 4.2.2 Aspects de consommation — 4.2.3 Aspects de recyclage — 4.2.4 Autre impacts — 4.2.5 Comment agir?

5 Que faire de ces ressources ? Quiz

255

CHAPITRE 8 NOTIONS FONDAMENTALES EN PYTHON

- 1 Écosystème et environnement
 - 1.1 Organisation du cours 256
 - 1.1.1 Présentation — 1.1.2 Pourquoi PYTHON ?
 - 1.2 Outils de développement 260
 - 1.2.1 Interpréteur PYTHON — 1.2.2 IDLE
 - 1.3 Documentation et code PYTHON 265
 - 1.3.1 « Notebooks » JUPYTER — 1.3.2 Extension LATEX PYTHONTEX
- 1.4 Synthèse et compléments 269
 - 1.4.1 Modes d'exécution PYTHON — 1.4.2 Exercices d'application

2 Variables, objets et typage

- 2.1 Nommage et mots-clefs 277
 - 2.1.1 Types, objets et méthodes — 2.1.2 Référencement et variables — 2.1.3 Typage dynamique et gestion de la mémoire

2.2	Compléments et application	281
2.2.1	Mot-clefs PYTHON — 2.2.2 Gestion de la mémoire — 2.2.3 Typages statique versus dynamique	

3 Types numériques

3.1	Généralités	285
3.1.1	Quatre types numériques — 3.1.2 Manipulation comme calculette — 3.1.3 Affectations et opérations	

3.2	Compléments et application	291
3.2.1	Précision des calculs flottants — 3.2.2 Opérations bit à bit (<i>bitwise</i>) — 3.2.3 Exercices	

4 Que faire de ces ressources? Quiz

299

CHAPITRE 9

CONCEPTS ÉLÉMENTAIRES

1 Chaînes de caractères

1.1	Codage, décodage, Unicode	299
1.1.1	Encodage et manipulation — 1.1.2 Outils et formatage	
1.2	Compléments et application	312
1.2.1	Expressions régulières — 1.2.2 Exercices	

2 Notions de séquences et de listes

2.1	Définition et manipulation	328
2.1.1	Séquences — 2.1.2 Listes	
2.2	Compléments et application	333
2.2.1	Séquences, listes et <i>slicing</i> — 2.2.2 Méthodes spécifiques aux listes — 2.2.3 Mutabilité et immutabilité des objets — 2.2.4 Tris de listes	

3 Introduction aux conditionnelles, itérations et fonctions

3.1	Test <code>if-else</code>	341
3.1.1	Utilisation — 3.1.2 Blocs d'instruction et syntaxe	

3.2 Factorisation de code

3.2.1	Boucle <code>for</code> — 3.2.2 Fonctions	
-------	---	--

3.3 Compléments et application

3.3.1	Bonnes pratiques — 3.3.2 Fonctions et valeur de retour — 3.3.3 Exercices	
-------	--	--

4 Introduction aux compréhensions de listes et aux modules

4.1	Compréhensions de liste	359
4.1.1	Syntaxe et utilisation — 4.1.2 Exercices	
4.2	Modules	362
4.2.1	Attributs — 4.2.2 Bibliothèque standard et autres	

5 Que faire de ces ressources? Quiz

CINQUIÈME PARTIE — ADDENDA

371

RÉFÉRENCES

391

GLOSSAIRE

PROPOS LIMINAIRES

PROPOSER UNE SYNTHÈSE INTRODUCTIVE AU NUMÉRIQUE pose l'ambition du présent manuel. Pour ce faire, il s'appuie sur la transcription de trois Mooc¹ introduits par l'INRIA — Institut national de recherche en informatique et automatique — et diffusés sur la plateforme FUN-MOOC — France université numérique.

Les deux premiers Mooc s'adressent plus particulièrement aux enseignants du secondaire qui se destinent à intervenir dans les cours d'informatique des nouveaux programmes : ICN — « Informatique et création numérique » (2017–2019) — et SNT — « Sciences numériques et technologie » (2019–2021). Le troisième Mooc, « PYTHON 3 : des fondamentaux aux concepts avancés du langage » (2017–2021), recouvre un public plus large : tous ceux qui désirent s'initier au langage de programmation PYTHON.

Bien que la qualité intrinsèque² de ces formations se suffise à elle-même, compte tenu qu'elles se complètent, il a semblé intéressant de les regrouper au sein d'une sorte de « document unique ».

Dans la mesure où leur licence de publication³ le permet, une seconde motivation tient au fait que ce document puisse être remanié par tout un chacun ; bien entendu à titre personnel, mais on pense surtout aux enseignants et formateurs.

Cette première version⁴ se résume essentiellement à reprendre les contenus des Mooc, notamment en transcrivant les propos oraux des différents intervenants⁵ issus des supports multimédias. En l'état, il manque encore un chapitre sur les implications et applications de l'informatique (bioinformatique, médecine, arts, etc.) et un autre sur l'architecture des ordinateurs et des réseaux. Cependant, le taux de recouvrement n'est pas nul car ces sujets sont également abordés dans d'autres sections du manuel.

Les apports supplémentaires concernent modestement la partie sur les logiciels libres et celle sur le son et la musique. Par ailleurs les chapitres sont peut-être parfois un peu déséquilibrés en termes de volume d'information. Un meilleur réagencement avec le déplacement de certaines parties en annexe est éventuellement à étudier. Toutefois, le manuel reste exploitable comme tel.

Espérant que le lecteur y trouve satisfaction,
Bonne consultation,
— La rédaction

¹. Massive Open Online Course.

². Remerciements chaleureux sont adressés aux auteurs et contributeurs de ces Mooc ; naturellement !

³. Creative Commons CC-BY.

⁴. Version v.0.1a.

⁵. Les vidéos de la partie SNT ne sont pas reproduites à l'écrit, car ce sont des réalisations didactiques scénarisées et indépendantes ; à visionner pour elles-mêmes.

GUIDE DE LECTURE

LE CHOIX EFFECTUÉ POUR LE RENDU DE CE DOCUMENT est celui du format PDF — *Portable Document Format*. En corrélation avec les outils de composition employés, cela offre une mise en page clairement structurée et la plus « propre » possible pour envisager tout autant une éventuelle impression papier¹ qu'une visualisation écran via ordinateur ou liseuse numérique.

La perspective est ainsi de proposer un document multifonctionnel, proche du canevas des Mooc proposés sur la plateforme France université numérique et accessible hors ligne pour servir de référence, soit comme support personnel d'information, soit à des fins de consultation au sein d'un centre de documentation.

Compte-tenu de sa licence² chacun peut faire évoluer le document selon ses besoins et en extraire³ pour son propre usage tout ou partie, par exemple dans un contexte de formation ou d'enseignement.

Orientations et cadre

La vocation d'un manuel rédigé n'est pas la même qu'une formation en ligne. En tant que transcription écrite de documents oraux (vidéos) et interactifs (Web), le style narratif employé est impersonnel mais le maximum d'interactivité est respecté : navigation, hyperliens et quiz.

Aspects linguistiques et principes rédactionnels

Le français est une langue relativement riche. Aussi, dans la mesure du possible, la rédaction évite les anglicismes ou propose une traduction littérale des termes anglais usités. Néanmoins, pour un manuel ayant pour sujet l'informatique et plus généralement en tant que document technoscientifique, il serait ridicule d'appliquer cette règle aux mots désormais passés dans le vocabulaire courant ou ne disposant pas de traduction satisfaisante — spécialement en programmation.

Ainsi, les anglicismes évitables sont modifiés comme par exemple le célèbre « baser sur⁴ » à la place de « fonder sur » ou « librairie » pour « bibliothèque », mais ceux communément admis sont conservés, notamment en expression technique, comme « implémenter » et « technologie » au lieu respectivement de « planter » et « technique ».

En tant que manuel écrit, sa rédaction observe dans l'ensemble les recommandations typographiques⁵ de la langue française (cf. « Petites

¹. Format actuel des imprimeurs, mais déconseillé si l'on a un tant soit peu de fibre écologique!

². Creative Commons CC-BY-NC.

³. Notamment pour un document papier, il est nécessaire de supprimer le visionnage des vidéos.

⁴. En français : « baser à ».

⁵. Pour les plus soucieux de précision, il est possible de consulter et télécharger le dictionnaire raisonné « Orthotypographie » de Jean-Pierre LACROUX.

6. En français écrit, il n'y a normalement pas d'espace entre les lignes de deux paragraphes consécutifs; mais cette règle est très souvent déroger pour les manuels techniques. Ce document n'est *a priori* ni un essai, ni un roman...

7. Métier ayant tendance à être dévalué.

8. *LATeX Project Public License*.

9. *Comprehensive TeX Archive Network*.

10. *GNU General Public License*.

11. Bien que toujours plus populaires et efficaces, les formats *Markdown*, *ePub* voire *Jupyter-Notebook* n'ont pas la même vocation, ni de présentation, ni même de publication. Certes, des outils de conversion existent, mais comportent encore des limitations importantes pour être utilisés facilement — ils sont réservés à des structures de document et des mises en forme relativement simples et, point d'achoppement majeur pour *ePub*, interdisent l'appel à des fontes *OPENTYPE*: un des intérêts fondamentaux des moteurs de composition *XeTeX* et *LuaTeX*. Pour aller plus loin :

- ▶ Calibre;
- ▶ Pandoc.

12. Pour des raisons de sécurité et certainement commerciales, *ACROBAT READER™* n'est plus disponible d'un point de vue fonctionnel et pratique (instabilité) pour *LINUX* au-delà de la version 9.4.1.

13. Conseil : quel que soit le système d'exploitation, installer le logiciel *VLC*; la lecture des formats vidéos même les moins orthodoxes est généralement acceptée.

leçons de typographie » par Jacques ANDRÉ et « Petit typographe rationnel » par Eddie SAUDRAIS) : lettres capitales accentuées, ponctuation double avec espaces fines, noms propres en petites capitales, pas d'article commençant un titre, mots d'origine étrangère en italique, retrait de la première ligne d'un paragraphe, pas de mot isolé sur une ligne, emphase en italique mais pas en gras, etc. Cependant, là encore, par souci pratique, certaines consignes ne sont pas honorées, que ce soit les césures — coupure d'un prénom et d'un nom, césure de certains mot à deux syllabes ou commençant par « con- » — ou qu'il s'agisse du rajout d'espaces⁶ inter-paragraphes pour aérer le texte.

Par ailleurs, si l'anglais n'a cure des répétitions, quant à lui le français les tolère peu car « ça sonne mal ». Un effort est fait pour les éviter mais il va sans dire que ce n'est pas toujours possible ou approprié — c'est en particulier fort délicat lors de la remise en forme de la contribution d'un auteur; sans conséquence ou changement de sens? C'est tout le travail d'un(e) secrétaire⁷ de rédaction...

Partis pris techniques

Un certain nombre de choix techniques sont conjoncturels ou arbitraires. Le premier d'entre eux est de ne s'appuyer que sur des solutions ouvertes voire libres — au sens *GNU des logiciels libres*.

Au-delà des aspects culturels, la motivation est que l'ensemble des outils employés et des exemples présentés soient accessibles gratuitement à tout un chacun. Aussi, à la fois pour la rédaction et la consultation du document mais encore pour l'apprentissage de *PYTHON*, la plateforme de préférence est donc un système d'exploitation *LINUX* et, plus spécifiquement ici, la distribution « grand public » *UBUNTU MATE 20.04 LTS*. Cette dernière est suffisamment « légère » pour être installée sur des machines relativement peu puissantes et relève d'un paradigme traditionnel de bureau qui ne peut que ravir les seniors...

La composition du document est effectuée à l'aide de l'outil multiplateforme *LATeX*⁸ — distribution *TeXLive*, moteur *LuaTeX* — et de multiples extensions⁹ de style. Quant aux illustrations et images, elles sont élaborées ou retouchées au moyen des extensions graphiques *PGF/TikZ* ou des logiciels libres¹⁰ *INKSCAPE* et *GIMP*.

Au-delà de la stricte mise en forme, l'intérêt majeur du format *PDF* en consultation numérique¹¹ est de pouvoir gérer les *hyperliens*, tout autant que d'inclure ou de lancer des fichiers multimédias. De ce fait, ces fonctionnalités sont abondamment exploitées dans ce document et nécessitent un lecteur *PDF*¹² tel qu'*ADOBE READER™* pour les plateformes propriétaires *WINDOWS* et *MACOS*.

Une des seules — si ce n'est la seule — lacunes des systèmes d'exploitation fondés sur un environnement *GNOME/LINUX* est, malheureusement encore à ce jour, de ne pas proposer une visionneuse disposant de toutes les fonctionnalités du format *PDF*, notamment multimédias, à savoir *Evince* et *Atril*. Néanmoins, depuis le printemps 2020, le lecteur de document par défaut de l'environnement *KDE/LINUX* *Okular* introduit ces fonctionnalités. Ainsi, la lecture de contenus multimédias dans le corps même du document est également possible sous *LINUX*.

En revanche, quelque soit le système d'exploitation et le lecteur *PDF* énoncé, les couches *Optional Content Groups* (*OCG*) et *Optional Content Membership Dictionaries* (*OCMD*) sont désormais implémentées. Ces fonctionnalités du format *PDF* sont au fondement de l'interactivité des quiz du document (cf. infra).

En outre, en situation conventionnelle d'utilisation, le lecteur multimédia¹³ installé par défaut est ouvert et permet, à partir du document,

de lancer le visionnage vidéo. Il en est de même d'autres applications comme la consultation de documents externes, PDF ou sites Web.

La navigation au sein même du document peut au choix s'appréhender à partir des signets de sectionnement du document affichés par le lecteur utilisé (ADOBE ACROBAT READER™ ou autres), de la table des matières, ainsi que des sommaires par chapitre¹⁴ ou partie, mais également au moyen des liens directs proposés dans le corps du document. À cette fin, l'affichage des signets représentant la structure du document est ouverte par défaut — ainsi nommé le « panneau latéral ».

D'un point de vue pratique, l'ensemble des liens cliquables est révélé par survol de la souris et décelable par `code couleur` (cf. infra).

Aspects formels et mise en page

La conception de ce manuel suit quelques règles simples de mise en forme. Pour une meilleure lisibilité à l'écran et une approche plus moderne à l'impression, la fonte — ou police de caractères — est choisie **sans empattement** (police dite « bâton ») — *sans serif* en anglais. La préférence s'est portée sur la famille de fontes *Fira Sans*, développé à l'initiative de la fondation MOZILLA dans le cadre du projet FIREFOX OS, semble-t-il avorté à ce jour.

La famille de fontes *Fira Sans* est une des polices bâton de licence ouverte¹⁵ les plus complètes en termes de graisses des caractères (*Hair*, *UltraLight*, *Light*, *Regular*, *Medium*, *SemiBold*, *Bold*, *ExtraBold* et *Heavy*) et de glyphs (petites capitales et symboles mathématiques).

Pour la police à chasse fixe — *monotype* en anglais —, utilisée dans les listings ou la transcription de commandes système, la fonte *Fira Mono* possède une chasse trop importante et le dévolu s'est jeté sur la fonte **Ubuntu mono**. Cela a l'avantage de correspondre aux interfaces graphiques du système d'exploitation UBUNTU MATE pris en exemple comme outil de base.

Dans les us et coutumes de typographie, il est communément admis que la lisibilité optimale d'un texte est obtenue pour des longueurs de ligne d'environ quatre-vingts caractères¹⁶ au maximum. Ce faisant et pour disposer d'un texte suffisamment aéré, le corps des caractères est pris pour avoir des lignes de l'ordre de soixante-quinze caractères.

Par conséquent, la mise en page offre une marge suffisamment importante pour accueillir la plupart des illustrations et des notes additionnelles sans pour autant perturber le fil du texte principal.

En complément, l'usage veut que l'appel aux couleurs se résume là aussi à l'essentiel pour éviter qu'un effet « sapin de Noël » vienne gêner la lecture. Les codes couleurs du document sont associés à deux couleurs principales et deux couleurs secondaires :

1. un **bleu électrique** pour les hyperliens et les éléments graphiques d'agencement du manuel — RGB = (83, 104, 120);
2. un **rouge-brun** complémentaire au bleu, mais aussi pour les avertissements et certains encarts — RGB = (153, 25, 25);
3. un **vert canard** (*teal* en anglais) pour les environnements de solution et encarts d'explication — RGB = (0, 128, 128);
4. une **palette de marrons** quasi exclusive aux illustrations.

Pour ce qui concerne la coloration syntaxique des listings et la simulation des interpréteurs PYTHON, les codes couleurs sont repris soit du thème « *Radiant-MATE* » de la distribution UBUNTU MATE, soit du module PYTHON PYGMENTS, soit des interpréteurs IDLE et IPYTHON tels que configurés par défaut sur la plateforme (cf. infra).

¹⁴. La première page des chapitres et parties proposent un sommaire partiel relatif à leur contenu.

¹⁵. SIL Open Font Licence.

¹⁶. Cela s'avère être la proposition par défaut de tous les éditeurs de texte.

Structure et contenus

Les possibilités de navigation au sein du document déjà évoquées sont issues de la structuration même des contenus rédactionnels. Un ensemble de conventions graphiques permet le repérage rapide des différents environnements associés.

Arborescence des supports

Ce manuel constitue le corps des transcriptions des trois Mooc « Informatique et création numérique », « Sciences numériques et technologie » et le début de « PYTHON : des fondamentaux aux concepts avancés du langage ».

En parallèle de ces contenus rédactionnels, les éléments multimédias (essentiellement les vidéos) et les documents externes associés (principalement les fiches de travaux pratiques ou propos complémentaires) forment un tout cohérent et sont repris comme liens internes. Cela suppose au préalable d'installer dans le même répertoire que le manuel les arborescences par chapitre sous les répertoires . ▶ **Videos** (sans accent) et . ▶ **Documents** — téléchargement puis décompression des archives respectives.

Les vidéos associées au sujet en cours de consultation sont repérées par un grand pictogramme dans la marge de la page (voir ci-contre). Si la visionneuse PDF employée le permet (ADOBE READER™ sous MACOSX ou WINDOWS et OKULAR sous Linux), la vidéo peut-être lue dans le document lui-même par simple clic sur l'iconographie. Pour d'autre lecteurs PDF (comme EVINCE ou ATRIL), il faut cliquer sur le petit pictogramme pour lancer le lecteur multimédia installé par défaut sur le système.

Agencement du document

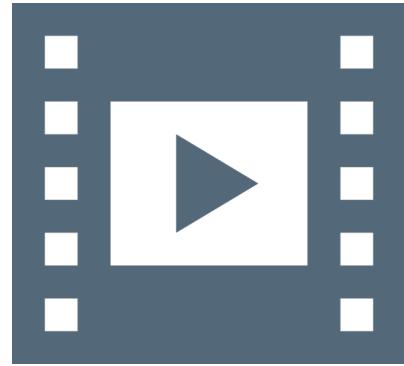
La teneur du discours se subdivise par parties et chapitres qui correspondent aux diverses thématiques des Mooc qu'ils transcrivent :

- I. **INFORMATIQUE, CRÉATION NUMÉRIQUE** — Comme son nom l'indique et en tant qu'introduction, cette partie recouvre les principaux chapitres du Mooc éponyme.
- II. **FONDEMENT DE L'INFORMATIQUE** — Les contenus communs aux Mooc ICN et SNT sont présentés dans cette partie.
- III. **NUMÉRIQUE : CULTURE ET PRATIQUE** — Le propos est d'exposer ici les contenus originaux du Mooc SNT, en explicitant quelques domaines d'application où le numérique est devenu incontournable et en offrant des pistes d'activités à réaliser avec les élèves.
- IV. **CODAGE EN LANGAGE PYTHON** — Pour clôturer le corps du document, cette partie est également explicite en apportant le début du tronc commun du Mooc consacré à PYTHON.

Le document se voulant évolutif, des annexes peuvent bien entendu être apportées sur certains sujets additionnels comme par exemple des données historiques sur les calculateurs et l'informatique, les portes logiques et des éléments d'électronique, des notions de traitement de signal en audionumérique ou en multimédia et des applications avec PYTHON/MATPLOTLIB, etc.

Signalétique, pictogrammes et visuels spécifiques

Pour faciliter la lecture et structurer la présentation des contenus, il est fait appel à des environnements repérés par des éléments graphiques et autres pictogrammes, soit disposés en marge, soit dans le corps du texte.



VIDÉO 0 — Éducation à l'informatique,
Gérard BERRY.

 Ceci est un encart pour une remarque additionnelle importante au regard du propos principal.

Ainsi, ces compléments sont indiquées par l'un des pictogrammes suivant :  pour une note indépendante,  pour une alerte ou  dans le cas d'une interrogation. De même, les notes de rédaction et certains avertissements sont exprimés en marge de manière explicite dans leur intitulé (voir ci-contre).

NOTE DE LA RÉDACTION

Ceci est un encart de marge, le plus souvent indiquant une note de la rédaction ou un avertissement spécifique.

1

Des notes numérotées et de pleine page sont également proposées pour faire état d'un conseil important ou d'une procédure à suivre particulièrement signifiante (cf. apprentissage de PYTHON).

Les documents internes (voir supra) et externes à ouvrir et/ou à télécharger sont désignés par un pictogramme qui représente leur nature de fichier :  texte ou LIBREOFFICE WRITER,  PDF,  page Web,  vidéo et  pour un lien externe explicite.

Les exercices et leurs solutions bénéficient d'environnement spécifiques (voir ci-dessous). Les compétences requises sont rapportées en marge par trois petits « voyants » qui distinguent leur niveau de réalisation en passant au vert : basique, intermédiaire et avancé.



EXERCICE DE STYLE

2 POINTS



Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.



SOLUTION NON MOINS DE STYLE

2 POINTS



Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

À l'instar des exercices, les questionnaires à choix multiples — ou « quiz » — jouissent d'un environnement particulier associé à une interface interactive. Leur présentation suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implantée dans l'extension de style L^AT_EX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

De manière traditionnelle en Interface humain-machine (IHM), lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (*radio button*) ; en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (*check box*). En outre, après validation des réponses (bouton « Vérifier »), leur explication s'affiche en marge ou en infobulle (bouton « Afficher la réponse »).

Qu'il s'agisse des exercices ou des quiz, un système de points peut — mais pas obligatoirement — leur être assignés. Les points sont affichés en haut à droite de leur environnement respectif.

☰
QUIZ A — AUTOÉVALUATION
1,5 POINTS

QUIZ A.1 — LANGAGE DE PROGRAMMATION

Quel langage de programmation est présenté dans ce Mooc?

JAVA

PERL

C++

PYTHON

FORTRAN

PASCAL

PHP

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER

QUIZ A.2 — MODÈLE DE COULEUR RGB/RVB

Quelles sont les composantes de couleur d'une image RGB/RVB?

Rouge

Bleu

Magenta

Noir

Jaune

Vert

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER

Comme déjà mentionné, les colorations syntaxiques des codes de programmation sont celles présentes par défaut dans la distribution UBUNTU MATE ou fournies par le module PYTHON PYGMENTS. Le langage de programmation est rappelé en marge des listings :  ,  ,  ,  et ainsi de suite.



```

1 public class HelloWorld
2 {
3     public static void main (string[] args)
4     {
5         System.out.println("Hello world!");
6     }
7 }
```



```

1 print("Hello world!")
```

À des fins didactiques et pour ne pas troubler inutilement le lecteur néophyte, les fenêtres de terminal LINUX (*shell*) et des interpréteurs PYTHON ainsi que l'environnement des notebooks (cf. partie PYTHON)

sont simulées pour avoir à l'écran peu ou prou le même affichage que dans le manuel. Cela produit les différentes configurations à suivre.

The image contains three screenshots of terminal windows, each showing a different Python environment:

- Terminal (Top):** Shows a standard terminal window with a dark background. It displays a command-line session where the user creates a directory 'programming', changes into it, and runs Python 3.8.2. The session ends with an exit command.
- Terminal (Middle):** Shows another terminal window with a dark background. It displays an IPython session starting with 'ipython'. The session includes a few commands (20*30, a=10, print(a)) and ends with an exit command.
- Python 3.8.2 Shell (Bottom):** Shows a Python shell window with a light gray background. It displays a standard Python 3.8.2 session with the same commands as the other terminals, ending with an exit command.

Below the shell window, there is a table showing the history of the session:

In[1]	20 * 30
Out[1]	600
In[2]	a=10
	print(a)
Out[2]	10

Il est néanmoins à noter qu'une fois présentés tous ces différents environnements de développement, c'est une console IDLE qui est privilégiée, car pleinement intégrée au document au moyen de l'extension L^AT_EX intitulée PYTHONT_EX (cf. chapitre 8 § 1.3.2).

VOUS AVEZ DIT NUMÉRIQUE ?

QUID DU NUMÉRIQUE ? Le « numérique » recouvre les champs disciplinaires de l'informatique qui, de fait, est à double entrée, à voir à la fois comme une science et une technologie.

Comme corollaires respectifs, l'informatique est également une culture et une industrie (voir une discussion détaillée sur le [terme numérique](#)), avec des usages et des conventions. Du reste, on distinguera les sciences *du* numérique — à savoir, informatique et mathématiques appliquées — des sciences numériques — autrement dit, les sciences transformées par le numérique.

Au-delà de la dénomination « SNT », *Sciences [du] Numérique[s] et Technologie*, il faut bien comprendre que ce sont ces quatre facettes qui sont à faire découvrir aux élèves, pour leur permettre de se construire une vraie vision sur ces sujets.

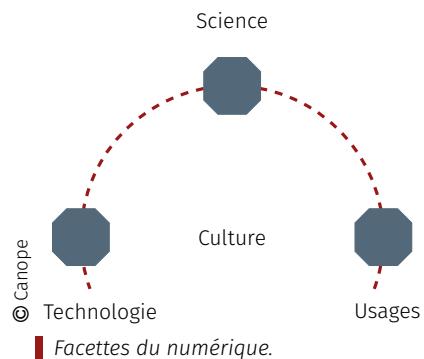
L'enseignement de sciences numériques et technologie en classe de seconde a pour objet d'appréhender les principaux concepts des sciences [du] numérique[s], mais également de permettre aux élèves, à partir d'un objet technologique, de comprendre le poids croissant du numérique et les enjeux qui en découlent.

Vivre à l'ère du numérique

La numérisation généralisée des données, les nouvelles modalités de traitement ou de stockage et le développement récent d'algorithmes permettant de traiter de très grands volumes de données numériques constituent une réelle rupture dans la diffusion des technologies de l'information et de la communication. Cette révolution multiplie les impacts majeurs sur les pratiques humaines.

Par exemple, l'actuel mobile multifonction est un objet technologique qui permet, comme le téléphone du XX^e siècle, de téléphoner, mais [qui est aussi] devenu une interface universelle d'accès à l'information et de commande d'autres objets.

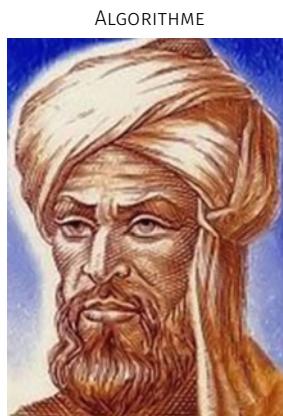
La convergence d'activités utilisant le numérique est un phénomène généralisé lié au développement de la science informatique et des technologies associées et notamment à leur intégration avec le domaine des télécommunications, à l'informatisation massive de champs économiques variés (communication, audiovisuel, transports, instrumentation scientifique médicale et technique, outillage numérique, objets connectés, etc.) et, bien sûr, à la création du réseau Internet.



pngtree.com
©



Concepts définis par la science informatique



© Wikimedia Commons



Malgré leur grande variété, ces avancées se fondent toutes sur l'universalité et la flexibilité d'un petit nombre de concepts en interaction :

- les *données*, qui représentent sous une forme numérique unifiée des informations très diverses : textes, images, sons, mesures physiques, sommes d'argent, etc. ;
- les *algorithmes*, qui spécifient de façon abstraite et précise des traitements à effectuer sur les données à partir d'opérations élémentaires ;
- les *langages*, qui permettent de traduire les algorithmes abstraits en programmes textuels ou graphiques de façon à ce qu'ils soient exécutables par les machines ;
- les *machines* et leurs systèmes d'exploitation, qui permettent d'exécuter des programmes, d'assurer le stockage des données et de gérer les communications, y compris les objets connectés et les réseaux.

À ces concepts s'ajoute un élément transversal : les interfaces qui permettent la communication avec les humains, la collecte des données et la commande des systèmes.

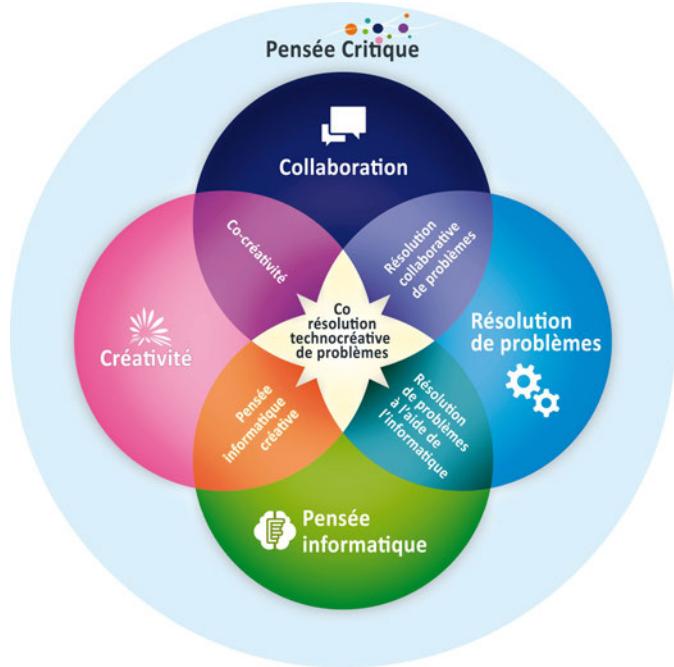
Motivation et attendus

COMPÉTENCES TRANSVERSALES

Cet enseignement inclut l'acquisition :

- de savoirs (connaissance scientifique et technique en sciences [du] numérique[s]),
- de savoir-faire (pratique de la programmation et de la création d'objets numériques),
- et de savoir-être et agir pour savoir-devenir par rapport au numérique (attitude ni technophobe ni technophile, mais technocritique, au-delà des idées reçues pour tirer le meilleur du numérique et positionner son projet d'avenir face à cette réalité).

Au-delà, cet enseignement vise à développer *deux grandes compétences transversales : faire preuve d'autonomie, d'initiative et de créativité et développer sa réflexion personnelle et son esprit critique*.



S'initier à la pensée informatique permet effectivement de développer les compétences du XXI^e siècle, notamment la pensée critique et créative afin que tous les citoyens possèdent les connaissances minimales nécessaires à leur participation politique, économique, sociale et culturelle à l'ère du numérique, par exemple face à ce qu'on nomme intelligence artificielle.

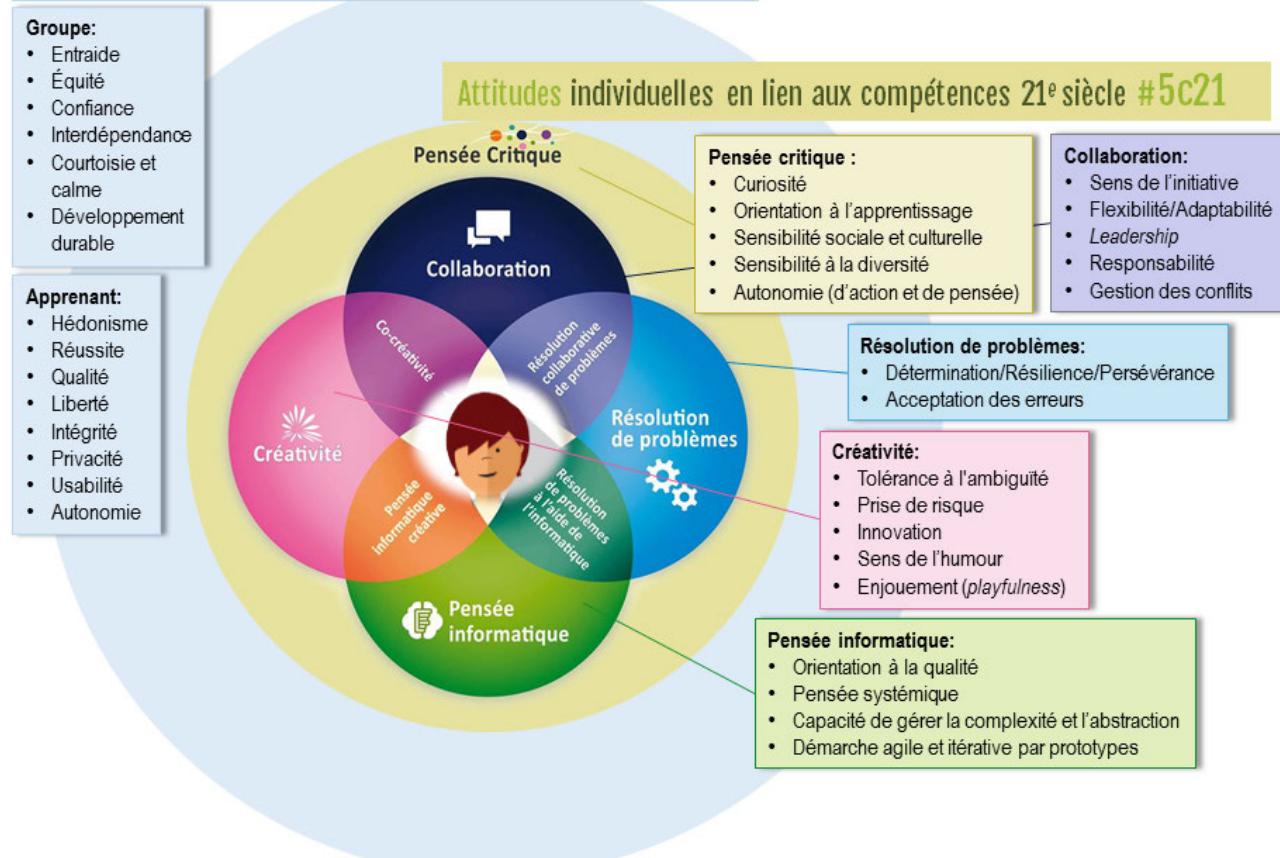
Ces compétences font l'objet d'études et sont bien précisées, même si elles recouvrent des champs très vastes du développement de la personne en passe de devenir adulte. Elles sont aussi associées à des valeurs et à une vision humaniste du monde.

Apprendre à une future garagiste ou à un futur fleuriste, à programmer et à s'initier à la pensée informatique plutôt que de le ou la convaincre qu'il suffit de savoir utiliser les produits commerciaux du numérique, relève d'un vrai choix de société : celui de ne pas laisser les grands systèmes numériques être conçus par d'autres et de ne pas uniquement les consommer, voire les subir, mais participer à leur création et se donner les moyens d'en user de manière critique.

COMPÉTENCES, ATTITUDES ET VALEURS

On liste ici les mots-clés qui permettent de relier les grands éléments de ces compétences transversales à des notions usuelles que nous connaissons bien et qui s'intègrent naturellement dans nos démarches pédagogiques.

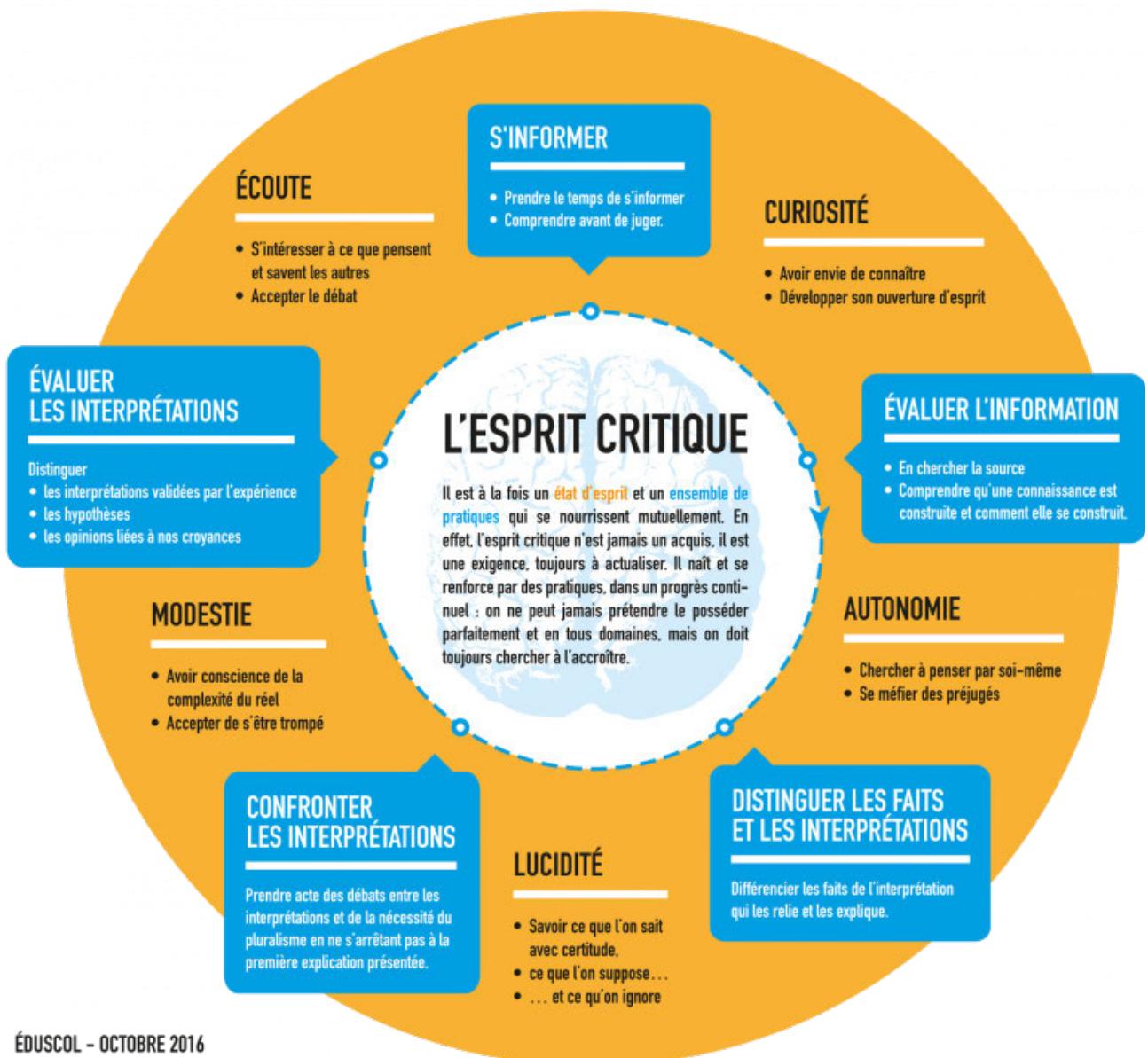
Valeurs pour la communauté d'apprentissage #5c21



DÉVELOPPER L'ESPRIT CRITIQUE

L'esprit critique est une démarche de questionnement des opinions ou des théories, mais aussi une posture intellectuelle humaniste : je

m'intéresse aux arguments utilisés par l'autre, à ce qui conduit à les exprimer (voir par exemple cette critique de l'esprit critique).



ÉDUSCOL - OCTOBRE 2016

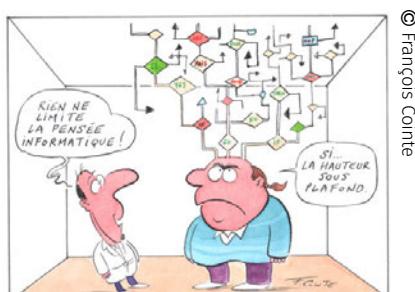
Savoirs informatiques

PENSÉE INFORMATIQUE

La pensée informatique a une définition, due à Jeannette WING. C'est un ensemble de compétences et de connaissances, utilisées en science et technologie informatique, mais applicables à d'autres domaines.

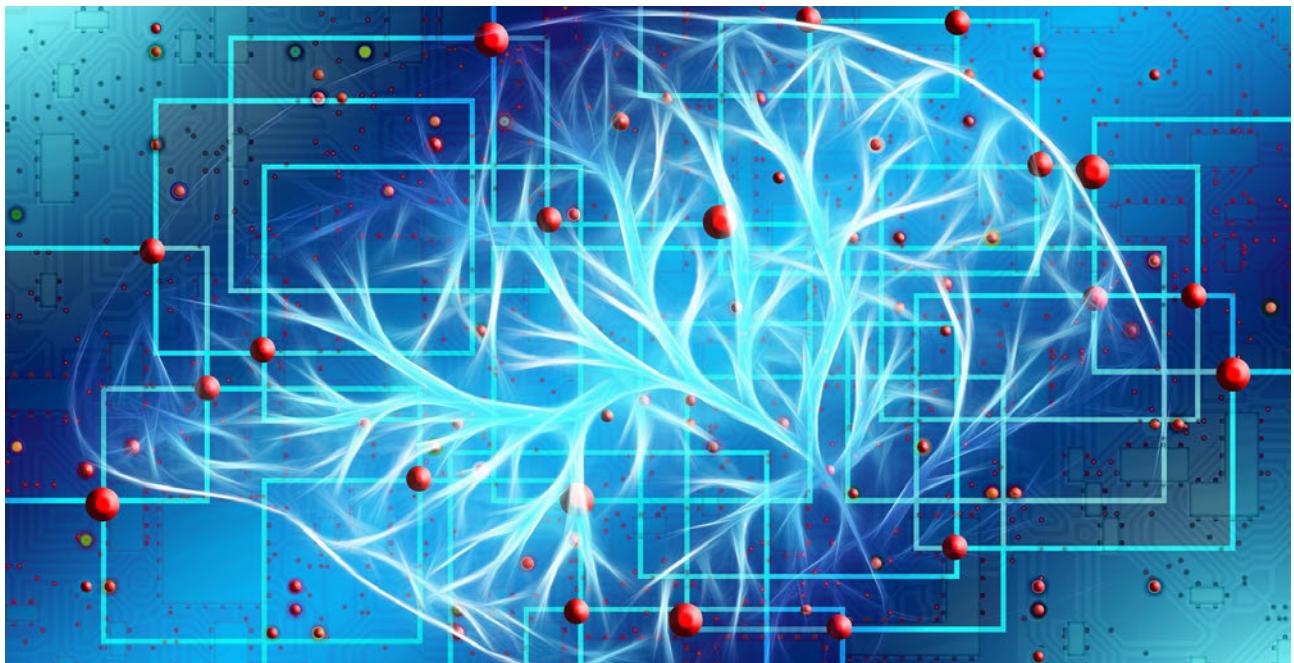
Cette panoplie d'outils intellectuels inclut, par exemple, la capacité à nommer de manière pertinente les objets et en expliciter leur type ou catégorie pour les manipuler correctement, à maîtriser la complexité d'un grand problème ou d'un système en le hiérarchisant, à pouvoir spécifier dans ses moindres détails un procédé pour qu'il puisse s'exécuter sans ambiguïté de manière mécanique, etc.

En résumé, l'informatique ne sert pas qu'en informatique comme discuté au niveau du projet CLASS'CODE qui a produit cette formation et



comme détaillé dans l'article original de Jeannette WING. Peut-on définir un mode de pensée spécifique à l'informatique ? La pensée informatique est présentée ici comme un ensemble d'attitudes et de connaissances universellement applicables, que nous gagnerions tous à apprendre et à maîtriser.

© Gerd Altmann via Pixabay



ENSEIGNEMENT DE L'INFORMATIQUE

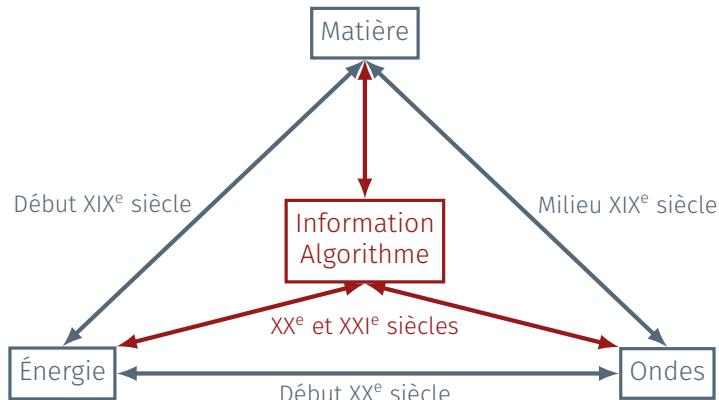
“L'information ne pèse pas, ne brûle pas, ne sent pas, ce n'est pas une quantité physique usuelle, et pourtant elle se stocke, se transporte et se duplique très facilement, tandis qu'elle se transforme avec des algorithmes. Cela conduit à une nouvelle façon de penser et d'agir avec des leviers d'une immense efficacité.

— Gérard BERRY

En février 2019, Gérard BERRY fait le point sur l'enseignement — et l'éducation — à l'informatique, lors d'un cours au Collège de France (à visionner, durée 1h15mn et/ou à consulter les planches de diaporama).

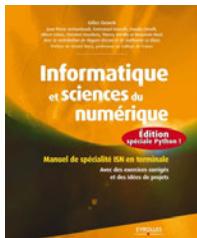


VIDÉO 1 — Éducation à l'informatique.

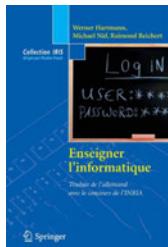


■ *Information et monde physique : évolution des sciences et techniques.*

Ouvrages généraux de référence



[DOWEK et al., 2013]



[HARTMANN et al., 2011]



[ABITEBOUL et DOWEK, 2017]



[LAZARD et MOUNIER-KUHN, 2019]



[TRICOT, 2007]



[TISSEURAND, 2019]

Informatique et sciences du numérique — Gilles DOWEK et al. (352 pp, Eyrolles éditeur, août 2013). Ce manuel librement accessible en ligne, est le manuel de spécialité ISN en terminale, avec des exercices corrigés et des idées de projets, il contient toutes les notions de base dont on a besoin en SNT pour se former, et beaucoup d'éléments pour préparer les cours, en faisant attention aux éléments hors programme.

Enseigner l'informatique — Werner HARTMANN WERNER, Michael NÄF, Raimond REICHERT avec Robert CABANE (176 pp, Springer Verlag, Collection Iris, 2011). Explique ce qu'un enseignement de l'informatique doit être, ce qu'il n'est pas et ce qu'il ne doit surtout pas devenir. Il explore méthodiquement certaines questions avec des situations vécues quotidiennement au nombre desquelles : existe-t-il une didactique de l'informatique ? Une pédagogie vaut-elle mieux qu'une autre ? Comment gérer la diversité au sein des groupes qui apprennent l'informatique ? Pourquoi et comment aborder l'abstraction ?

Le temps des algorithmes — Serge ABITEBOUL et Gilles DOWEK (192 pp, Éditions le Pommier, Collection Essais, janvier 2017). Cet ouvrage nous aide à nous poser des questions fondamentales sur la place des algorithmes dans notre société, où les relations sociales, la vie professionnelle, la propriété des biens immatériels, médecine, industrie, transport, sciences jusqu'à la démocratie même, suscitent de vrais bouleversements. Il permet de prendre du recul par rapport aux aspects sociétaux du programme et de les nourrir des liens avec la science et technologie informatique.

Histoire illustrée de l'informatique — Emmanuel LAZARD et Pierre-Éric MOUNIER-KUHN (240 pp, EDP Sciences, 2019). Une description de l'histoire de l'informatique et du numérique à travers des dates et des événements concrets (Antiquité, XVII^e au XIX^e siècle et surtout XX^e et XXI^e siècles). Une base de connaissances de référence pour préparer ces cours et fournir de repères au élèves.

Apprentissages et documents numériques — André TRICOT (277 pp, Éditions Belin, Sup psychologie, 2007). Chaque idée relativement à l'apprentissage avec des documents et outils numériques est passée au crible avec une méthode simple et rigoureuse : énoncer un mythe (par exemple le mythe de l'apprentissage ludique) et préciser en quoi il consiste. Étudier ce qu'en disent les travaux scientifiques. Donner des exemples (de jeux sérieux, dans le chapitre cité). Et en tirer une conclusion, sérieuse, modérée et sans prétention. Voir aussi deux ouvrages complémentaires plus récents : *Apprendre avec le numérique* et *L'innovation pédagogique*.

Au cœur des réseaux : des sciences aux citoyens — Fabien TARISSAN (168 pp, Édition le Pommier, Collection Essai, Mars 2019). Cet ouvrage traite de science, plus particulièrement des réseaux (informatiques mais pas seulement), des algorithmes qui y opèrent et de l'impact qu'ils ont sur notre manière de nous informer en ligne (*fake news*, publicité ciblée, données personnelles, etc.) et de l'économie sous-jacente. Il permet de travailler sur les thématiques réseaux sociaux, objets connectés et internet et réseaux.

L'Hyperpuissance de l'informatique — Gérard BERRY (506 pp, Éditeur Odile Jacob, Collection Sciences, octobre 2017). Ce livre montre de façon non technique comment la science et la technologie informatiques mettent l'information au cœur de l'action, qu'elle soit produite par les humains ou par les machines et conduisent à un

nouveau schéma mental différent de celui des siècles précédents, qui confère un pouvoir étonnant à ceux qui le comprennent et l'organisent. Il permet de justifier l'enseignement SNT lui-même et de nourrir le travail sur la pertinence de ces éléments, non sans nourrir les contenus.

Big Data — Xavier PERRET, Guy JACQUEMELLE (240 pp, Editions Kawa, Collection : Tout savoir sur, Septembre 2014). Cet ouvrage présente des exemples d'application des *big data*, leurs enjeux et l'importance dans notre société avec une entrée culturelle pluridisciplinaire pour les élèves : des films ; déjà utilisé avec des élèves (en ICN) pour faire réaliser des exposés. Il permet de voir ou revoir des films parfois anciens d'anticipation au travers du prisme de la collecte des données et de leur utilisation. Les auteurs ont créé un cours en ligne ouvert à partir de cet ouvrage.



[BERRY, 2017]



[PERRET et JACQUEMELLE, 2014]

INFORMATIQUE, CRÉATION NUMÉRIQUE

- | | |
|-----------|--|
| 11 | CHAPITRE 1
NUMÉRIQUE ET SCIENCES DU RÉEL |
| 75 | CHAPITRE 2
WEB ET USAGES SOCIÉTAUX |
| 99 | CHAPITRE 3
INTELLIGENCE ARTIFICIELLE ET ROBOTIQUE |

NUMÉRIQUE ET SCIENCES DU RÉEL

DE NOS JOURS, NOMBRE DE PERSONNES POSSÈDENT UN ORDINATEUR, que ce soit * une station de travail — isolée ou en réseau, autrement dit et hors accès Internet, souvent synonyme de « à la maison » ou « au travail » —, un ordinateur portable, une tablette, un *smartphone*, un téléviseur voire une domotique connectés. Tous ces objets ont été créés sur le même modèle et strictement la même architecture conceptuelle.

*Rappelons toutefois que certains n'ont toujours pas le bénéfice ou la maîtrise, même élémentaire, de tels outils : « fracture numérique » oblige, souvent équivalente de sociale.

Il est clair que cette présentation générale mérite d'être soutenue par d'autres apports comme, entre autres, les origines du calcul automatisé ou l'évolution de la technologie des transistors.

Pour certains, l'ordinateur reste encore « une boîte noire » mystérieuse — parfois au sens propre comme figuré — qui, lorsque tout fonctionne correctement, ne saurait souffrir d'une investigation plus poussée. Néanmoins, disposer de quelques éléments d'information, tant des points de vue matériels que logiciels, s'avère bénéfique; ne serait-ce que pour choisir en toute connaissance de cause son équipement, vis-à-vis de commerciaux en conseils, parfois, pour le moins orientés.

1 Entrailles d'un ordinateur

Sans chercher à entrer dans le détail du fonctionnement d'un ordinateur, il semble cependant utile de pouvoir observer ses différentes composantes et d'en décrire les fonctionnalités qui lui sont associées.

1.1 Démontage et descriptif d'un ordinateur

Dans la perspective de démythifier le sujet, il est d'abord proposé le démontage d'un ordinateur conventionnel — type PC, pour *Personal Computer* en anglais — afin de montrer les organes qui le compose (cf. vidéo 1.1). Il est donc question d'explorer les divers éléments génériques d'un ordinateur, depuis les aspects extérieurs connus de tous, jusqu'aux plus intimes, qui en assurent son — supposé — bon fonctionnement.

1.1.1 Périphériques

Un ordinateur traditionnel répond à un canon préétabli de configuration, qui comprend *a minima* une plus ou moins grosse boîte que

SOMMAIRE

1 Entrailles d'un ordinateur	
1.1 Démontage et descriptif	11
1.1.1 Périphériques	■ 1.1.2 Unité centrale
1.2 Montage et fonctionnalités	15
1.2.1 Projet RASPBERRY PI	■ 1.2.2 Architecture de von NEUMANN
2 Système d'exploitation	
2.1 OS : trois idées-forces	23
2.1.1 Gestion mémoire	■ 2.1.2 Gestion processeur
2.1.3 Appels système	
2.2 À quoi sert un OS ?	26
3 Interface humain-machine	
3.1 Interactivité	29
3.1.1 Bref historique	■ 3.1.2 Concept d'utilisabilité
3.1.3 Procédure de conception	■ 3.1.4 IHM et technologies
3.1.5 Avenir de l'IHM	
3.2 50 ans d'IHM : retour vers le futur	34
4 Loi, économie et éthique	
4.1 Numérique : loi et vie privée	47
4.1.1 Données	■ 4.1.2 Contrôle et exploitation
4.1.3 Automatisation	■ 4.1.4 Éducation et transparence
4.2 Ville numérique et vie privée	51
4.3 Comprendre les logiciels libres	53
4.3.1 Fondements et principes	■ 4.3.2 Modèle économique
4.3.3 Logiciels libres	
4.4 Paradigme du logiciel libre	64
5 Que faire de ces ressources ? Quiz	

À PROPOS DE L'INTERVENANT

Erwan KERRIEN est chercheur en imagerie médicale dans l'équipe INRIA MAGRIT. Ses travaux visent à enrichir l'environnement du chirurgien pendant l'opération : techniques de vision par ordinateur, réalité augmentée et simulation guidée par l'image. Chargé de mission en médiation scientifique au centre INRIA NANCY-GRAND EST, il anime de nombreuses initiatives. Il est un des concepteurs et auteurs du Mooc ICN.

1. Périphériques externes : sous-entendu externes à l'unité centrale.



VIDÉO 1.1 — Éléments d'un ordinateur.

2. Les écrans « tactiles » des équipements proposés depuis quelques années sur les téléphones mobiles, tablettes et autres ordinateurs portables sont aussi des périphériques d'entrée et de sortie : leur interface — dite IHM pour Interface Homme-Machine — permet dans un même temps, de lancer des instructions et d'en récupérer les résultats.

3. Dans le cas de la vidéo Vid. 1.1, ce type d'unité centrale est communément appelée une « tour ». Il existe désormais sur le marché une large variété de formats d'unité centrale : depuis les mini-PC de bureau — et *barebones* à compléter —, jusqu'aux stations audionumériques des studios d'enregistrement ou bien serveurs de 19 pouces prévus pour être compartimentés « en cascade » — *rackable* en anglais.

4. Universal Serial Bus, conçu par COMPAQ et MICROSOFT et diffusé à partir de 1996. Les normes ont évoluées des versions 1 à 3 de nos jours, dont la différence essentielle pour l'utilisateur est le débit du transfert des données. Source : Wikipedia [W](#).

5. Personal System/2, dont le nom provient des PC IBM de troisième génération introduits en 1987. Source : Wikipedia [W](#).

6. Digital Visual Interface, développé par la société Digital Display Working Group en 1999. Source : Wikipedia [W](#).

7. Le format HDMI — *High Definition Multi-media Interface* — est établi par un large consortium de compagnies en 2002. Il est devenu un standard de transfert de données vidéo en haute définition très répandu aujourd'hui. Source : Wikipedia [W](#).

8. On parle aussi aujourd'hui de « réalité augmentée » (voir § 3.1.4).

l'on appelle *unité centrale*, accompagnée d'un écran — également dénommé *moniteur* en lien avec l'industrie audiovisuelle ou comme anglicisme direct —, d'un clavier, d'une souris et, par exemple, d'une clef ou d'un *disque dur* de stockage nomade. Tous ces derniers éléments sont dénommés des *périphériques*¹ externes.

Les périphériques peuvent être disposés en *entrée*, comme le clavier, la souris, une caméra ou un scanner et permettre d'alimenter l'ordinateur en instructions et tâches à accomplir. Ils peuvent également être placés en *sortie*, à l'instar d'un moniteur ou d'une imprimante et restituer le travail effectué par la machine.

D'autres périphériques s'avèrent être à la fois² en entrée et en sortie, par exemple un disque dur externe où il est aussi bien possible d'y lire des informations en entrée et d'en inscrire en sortie.

L'ensemble constitué par les périphériques externes est ainsi branché à l'unité centrale³ — généralement à l'arrière —, sur des prises que l'on appelle des ports. En retournant l'ordinateur, on voit tous les branchements envisageables d'une unité centrale. Ceux-ci sont essentiellement assurés par des ports physiques avec, pour chacun d'entre eux, un type particulier de connectique.

On distingue ainsi des ports *USB*⁴ pour notamment les périphériques dits de *stockage de masse* comme les disques durs externes et autres clefs, mais également pour la souris, une caméra, etc.

Bien que ce format soit maintenant devenu obsolète, on peut aussi trouver sur des machines plus anciennes un ou plusieurs port(s) *PS/2*⁵ pour y brancher un clavier ou une souris.

Bien entendu, on dispose également de ports « vidéo » sur lesquels sont connectés les écrans, soit aux formats *VGA* ou *DVI*⁶ — là encore en désuétude —, soit au format *HDMI*⁷ pour les moniteurs les plus récents en haute définition, voire en très haute définition (4K).

On peut enfin avoir des ports *FireWire* — format plutôt abandonné au profit de l'*USB*, notamment depuis la version 3 —, mais surtout un port particulier, l'*Ethernet*, qui montre qu'Internet peut aussi être considéré comme un périphérique externe à l'ordinateur de par sa connexion à distance à d'autres équipements.

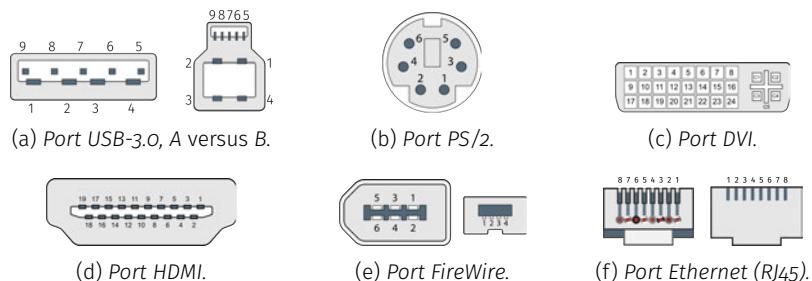


FIGURE 1.1 — Différents ports et connectiques associées d'un ordinateur.

Pour être complet, il faut encore mentionner que la plupart des ordinateurs possèdent des entrées/sorties audio au format mini-jack, respectivement pour un micro et un casque ou système d'enceintes. De ce point de vue, tout comme pour la vidéo, les composants dédiés au traitement de données audiovisuelles sont désormais très souvent intégrés à l'unité centrale (voir § 1.1.2 et figure 1.2 à suivre).

Bien que les technologies aient considérablement évolué pour intégrer ces fonctionnalités en configuration standard (carte mère et processeur), cela répond cependant à des besoins courants d'utilisation.

Pour des exploitations avancées ou professionnelles — simulation numérique et prototypage⁸ virtuel, jeux, infographie, musique assistée

par ordinateur (MAO), etc. —, il est nécessaire de compléter un équipement de base par des cartes graphique et/ou audio adéquates, destinées spécifiquement à ces types de données, qui s'avèrent⁹ gourmandes en termes de calculs à effectuer par le processeur.

1.1.2 Unité centrale

L'exploration de l'intérieur d'une unité centrale amène à constater beaucoup de choses. Pour commencer par sa relation avec l'environnement extérieur, il faut déjà noter la présence d'un bloc d'alimentation électrique. De cette alimentation sortent un ensemble de fils et de câbles de différentes couleurs qui servent à apporter l'électricité aux multiples composants internes de l'ordinateur.

En établissant le lien entre les divers ports et les éléments internes d'une unité centrale, on remarque que l'écran est branché sur la *carte graphique*, elle-même enfichée sur une grande plaque appelée la *carte mère*. Quant à eux, les clavier, souris et disques durs principaux — dits « maîtres » — sont directement connectés sur cette même *carte mère*.

Il est enfin à noter que les barrettes de mémoire dites « vives » et les cartes d'extension de tout ordre sont de même intégrées à la *carte mère* via leurs ports respectifs : DIMM versus PCI.

⁹. Il s'agit ainsi de déporter la charge de calcul d'opérations connues, donc anticipées, sur des éléments réservés à cet effet, audio comme vidéo.

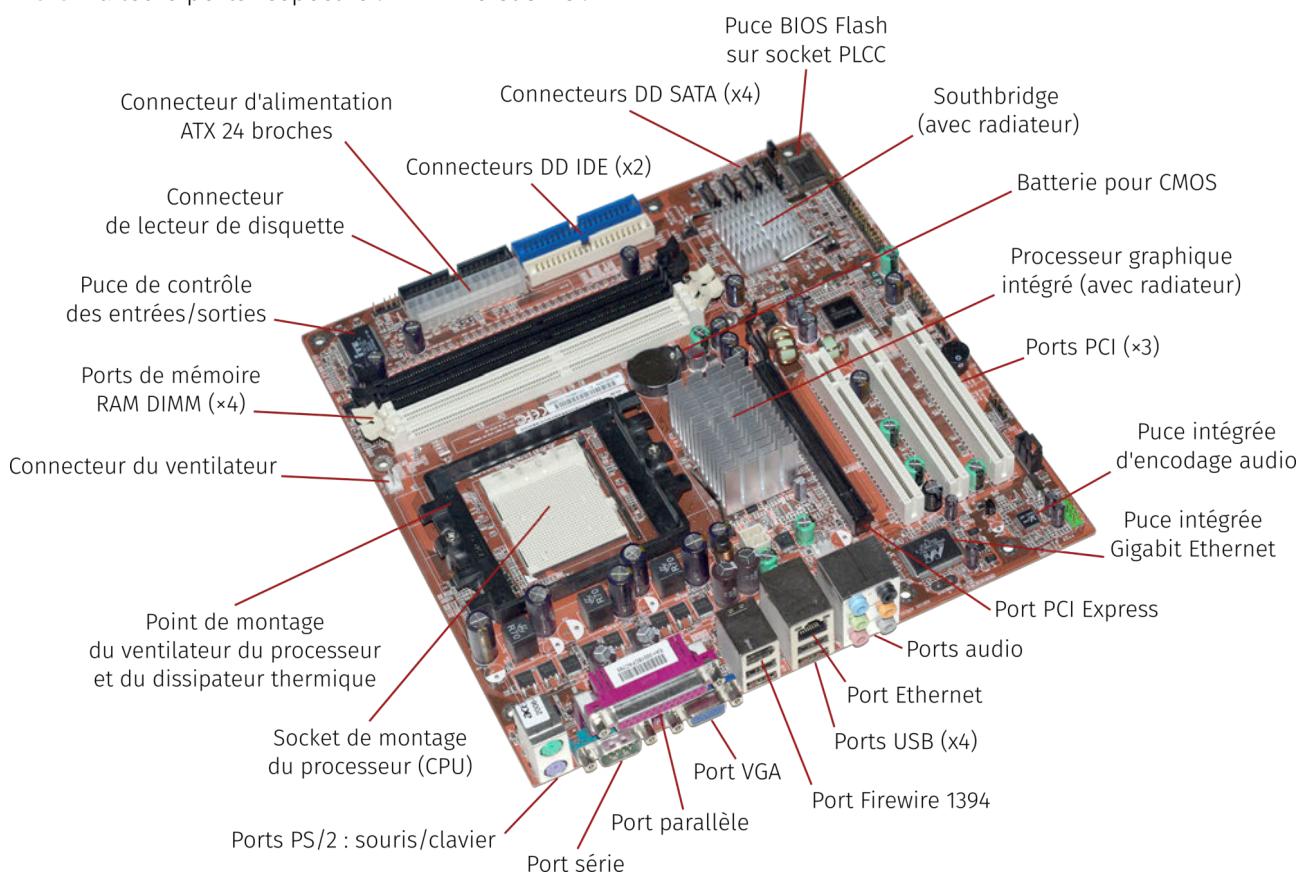


FIGURE 1.2 — Carte mère (assez ancienne, ≈ 2005) et ports des périphériques de l'unité centrale.

Sur certains ordinateurs devenus aujourd'hui obsolètes, mais toujours en service, on observe la présence d'autres composants — à considérer comme *périphériques internes* —, à savoir des lecteurs/graveurs de support numérique¹⁰ CD/DVD. En remontant le temps, d'autres matériaux antérieurs étaient présents, tels les lecteurs de disquettes...

En effet, la baisse spectaculaire du coût de la mémoire de stockage de masse (disques durs et clefs USB) constatée depuis la fin des années 2000 a modifié en profondeur les comportements et les modes

¹⁰. Les lecteurs/graveurs de type CD/DVD, considérés comme indispensables pour des raisons de puissance de calcul à la fin des années 1990-début 2000, sont désormais définitivement hors-jeu. Si besoin est, on peut s'en procurer à prix modique comme périphérique extérieur en connectique USB. Quant au lecteur de disquette, ils sont avantageusement remplacés par les clefs USB de toutes tailles de stockage.

11. Mise en exergue des mémoires plus rapides des technologies semi-conducteurs — SSD pour *Solid-State Drive* — au regard des mémoires à support magnétique des disques durs conventionnels.

12. *Peripheral Component Interconnect*, un bus initialement dû à INTEL dans les années 1990 qui n'a cessé d'évoluer ensuite. On peut citer un de ses dérivés assez largement utilisé, le *PCI-Express* qui a l'ambition de remplacer le PCI, mais également le bus graphique AGP — *Accelerated Graphics Port*. Source : Wikipedia [W](#).

13. Il s'agit de mémoire à accès très rapide que le processeur utilise comme tampon pour disposer des données en cours de traitement, en entrée comme en sortie.



FIGURE 1.3 — Éclaté d'un PC du milieu des années 2000 : ① écran, ② carte mère, ③ processeur, ④ connecteurs ATA, ⑤ mémoire vive, ⑥ carte d'extension, ⑦ alimentation, ⑧ lecteur CD, ⑨ disque dur, ⑩ clavier, ⑪ souris.

d'usage : qui de sa collection intégrale de photographies de vacances, voire de son système d'exploitation nomade.

Les offres du marché s'affranchissent donc désormais de proposer de telles fonctionnalités pour mettre en avant : puissances de calcul, performances du processeur, capacité et rapidité des périphériques de stockage¹¹ ou qualité de gestion graphique.

Par l'intermédiaire de câbles et de nappes — bus d'interconnexions en parallèle pour des raisons de performance —, on constate au global que tous les composants internes d'un ordinateur sont dûment connectés à la *carte mère* via des ports spécifiques, mais cette fois internes : dispositifs dédiés tels que cartes d'extension graphique, audio et autres (ports *PCI*¹²), ou bien barrettes de *mémoire vive*¹³ — RAM pour *Random Access Memory* en anglais.

Notamment par effet Joule, les divers composants sollicités d'un ordinateur produisent beaucoup de chaleur. Pour assurer leur bon fonctionnement, des systèmes de refroidissement sont introduits au sein de l'unité centrale.

Plusieurs solutions techniques existent, dont une des plus simples à mettre en œuvre est la ventilation de l'air ambiant, qu'il s'agisse de l'unité centrale elle-même, du *processeur* ou bien d'autres composants comme la carte graphique (voir figure 1.2). Les ventilateurs sont source de bruit, ainsi il est parfois fait appel à des conceptions avec des radiateurs en aluminium — bon diffuseur de chaleur —, en particulier dans les mini-PC : gain en bruit mais également en encombrement.

À l'observation d'une *carte mère* vierge, débarrassée de tous les éléments venant s'y greffer (cf. vidéo 1.1 et figure 1.2), on s'aperçoit que le point focal essentiel ou le cœur d'un ordinateur est le *processeur*. Tous les composants autour de celui-ci lui sont reliés par des bus optimisés selon les différentes délégations de tâches à accomplir : calculs purs, mémoire vive et zones tampons intermédiaires vis-à-vis des périphériques, fonctions graphiques, etc.

De par son architecture, le *processeur* comporte lui-même plusieurs parties complémentaires avec une *unité arithmétique et logique* (UAL) chargée en tant que telle des calculs, mais aussi de zones de mémoires extrêmement rapides dites des « *caches* » qui permettent de stocker les instructions en cours et les résultats intermédiaires. Là encore, le transfert des données se fait en parallèle pour des raisons de performances accrues.

Pour effectuer un récapitulatif en opérant un zoom arrière, le *processeur* constitue ainsi l'unité fondamentale et centrale de calcul : il est en lien avec tous les autres éléments de l'ordinateur via la *carte mère*. Sur cette dernière, toutes les fonctionnalités essentielles sont proposées au moyen de bus spécifiques : accès directs à la *mémoire vive* — barrettes DIMM —, au(x) disque(s) dur(s) — formats IDE ou SATA —, aux cartes d'extension comme la carte graphique — connecteur PCI, AGP, voire PCI Express —, lecteur-graveur de DVD et l'ensemble des ports de périphériques externes par leurs contrôleurs respectifs.

GLOSSAIRE CONTEXTUEL

CARTE MÈRE — Circuit imprimé qui rassemble un ensemble diversifié de connecteurs et de ports (connectique dédiée de certaines fonctionnalités) permettant d'y agréger les différents composants de l'ordinateur (processeur, cartes mémoire, carte graphique, disques durs et tout autre périphérique) et d'assurer leur liaison à travers des circuits imprimés dont le comporte-

ment est piloté par le **BIOS**, programme contenu dans la mémoire « morte » (ROM — *Read-Only Memory*) et lancé au démarrage pour notamment détecter et configurer tous les éléments connectés.

PÉRIPHÉRIQUE — Dispositif électronique connecté à un ordinateur, plus précisément sur un des ports dédiés de la carte mère.

PORT — Prise/connecteur permettant de brancher un périphérique à un ordinateur. Les ports sont placés sur la carte mère qui assure la liaison avec les autres périphériques et le processeur.

PROCESSEUR — Processeur ou CPU, pour *Central Processing Unit*. Dispositif électronique de calcul chargé d'exécuter les différentes instructions d'un programme. Ce faisant, les données numériques sont traitées, manipulées et échangées avec les autres composants de l'ordinateur — **mémoire vive** (RAM — *Random Access Memory*), disques durs et autres périphériques — via la **carte mère**. Quand le processeur est construit sur un unique circuit intégré, on parle de microprocesseur.

MÉMOIRE VIVE — La RAM, acronyme de *Random Access Memory* en anglais — littéralement « mémoire à accès aléatoire » nommée en français « mémoire vive » —, est un circuit imprimé sur une carte qui permet de stocker les données et les instructions employées dans un programme informatique. Contrairement à la mémoire dite « morte » en français — ROM, *Read-Only Memory*, littéralement « mémoire uniquement en lecture » —, le contenu de la mémoire vive peut être modifié et offre donc un espace temporaire de stockage dynamique pour le processeur.

1.2 Montage et fonctionnalités d'un ordinateur

Un ordinateur marque des signes de faiblesse, il est souffreteux, il rame et charger la moindre page Web prend plus de temps que la lire ? Seule solution : le remplacer... Certes, mais l'écran est encore valable, le clavier est comme neuf et la souris toujours aussi véloce.

En étant fortuné, aucune question à se poser, on se rue *illlico* chez son fournisseur de prédilection pour acquérir le dernier cri en la matière. En cas contraire où, au-delà des impôts, les factures s'accumulent, où la perspective est d'équiper son association préférée qui, par définition, n'a pas un rond... Existe-t-il une solution ?

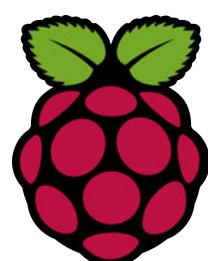
La question ne saurait se poser aussi directement s'il n'était pas envisageable d'y répondre par l'affirmative. En effet, pour environ 50 €, il est désormais possible depuis quelques années, avec un peu d'huile de coude et en consultant quelques documentations adéquates, de monter par soi-même un ordinateur fonctionnel, n'ayant de plus, rien à envier à une machine de cinq à dix ans révolus. En termes d'application, cela peut être une manière simple et peu onéreuse de maintenir un parc d'ordinateurs dans une salle informatique.

Cette solution remarquable fait appel au projet « **RASPBERRY PI** ». Initialement motivé par des objectifs pédagogiques et de transmission des savoirs, le succès du projet dépasse ces ambitions premières tant par ses qualités intrinsèques que par la variété des applications possibles : bureautique, domotique, mini-serveur, etc. L'engouement est tel que des revues lui sont partiellement ou entièrement consacrées.

Ayant la présence, on peut aussi citer le projet « **ARDUINO** » qui propose au moyen d'une carte électronique modulable, toutes sortes d'expérimentation en électronique numérique. Cependant, toutes choses étant comparables, le RASPBERRY PI s'avère un ordinateur complet dans

À PROPOS DE L'INTERVENANT

Yves PAPEGAY est chargé de recherche dans l'équipe HÉPHAÏSTOS INRIA SOPHIA-ANTIPOLIS. Il s'intéresse aux robots à câbles et à la robotique d'assistance. Il est spécialiste des outils de modélisation et de simulation, de calcul symbolique et d'analyse par intervalles.



ses fonctionnalités, bousculant un peu le petit monde de l'électronique numérique en créant tout un nouvel écosystème.

1.2.1 Projet RASPBERRY PI



VIDÉO 1.2 — Ordinateur Raspberry Pi.

14. *Errata* — La vidéo contient deux petites erreurs qui sont rectifiées ici :

- ▶ le port vidéo est un port vidéo RCA et non un port S-Vidéo comme annoncé, utile pour se servir d'un « vieux » téléviseur.
- ▶ l'alimentation des divers modèles 1, 2, 3 (modèle 4 prévu pour 2020) est en principe 5 V et non pas en 3 V comme décrit.

Le RASPBERRY PI est une idée un peu folle d'un concepteur anglais de jeux vidéo, DAVID BRABEN, qui a eu l'ambition de fabriquer un petit ordinateur pour une cinquantaine d'euros (voir¹⁴ vidéo 1.2).

En partenariat avec le laboratoire d'informatique de l'Université de Cambridge et Broadcom, une fondation a été créée en 2009 pour promouvoir l'informatique auprès des jeunes publics et plus largement auprès des personnes n'ayant pas les moyens de se procurer un ordinateur, mais pouvant récupérer de vieux matériels : téléviseur, clavier, etc. L'enjeu a ainsi été de concevoir et de faire fabriquer une carte mère et son processeur, disposant de toutes les fonctionnalités d'un ordinateur moderne, comme cœur essentiel d'une machine (cf. § 1.1.2). C'est chose faite depuis février 2012.

Le dimensionnement du RASPBERRY PI est délibérément réduit à sa plus simple expression, voulu pour correspondre au format d'une carte bancaire de crédit. Il s'agit d'une carte mère sur laquelle toutes les fonctionnalités d'un ordinateur conventionnel sont présentes, notamment les différents ports de communication, à savoir (en se fondant sur une description du modèle 3B) :

- ▶ processeur ARM Cortex A53 Quad-Core 1,2 GHz;
- ▶ mémoire vive 1 Go;
- ▶ un port HDMI, voire un port RCA sur les plus anciennes cartes, pour y brancher un téléviseur ou un écran;
- ▶ des ports USB 2 — quatre sur les cartes récentes — pour clavier, souris et autres extensions — selon les cas prévoir toutefois des extensions auto-alimentées pour ne pas perturber le bon fonctionnement du RASPBERRY PI;
- ▶ un port audio au format mini-jack pour un casque;
- ▶ un port Ethernet et des puces Bluetooth 4,1 et WiFi 802.11n pour les communications;
- ▶ un connecteur d'alimentation micro-USB.

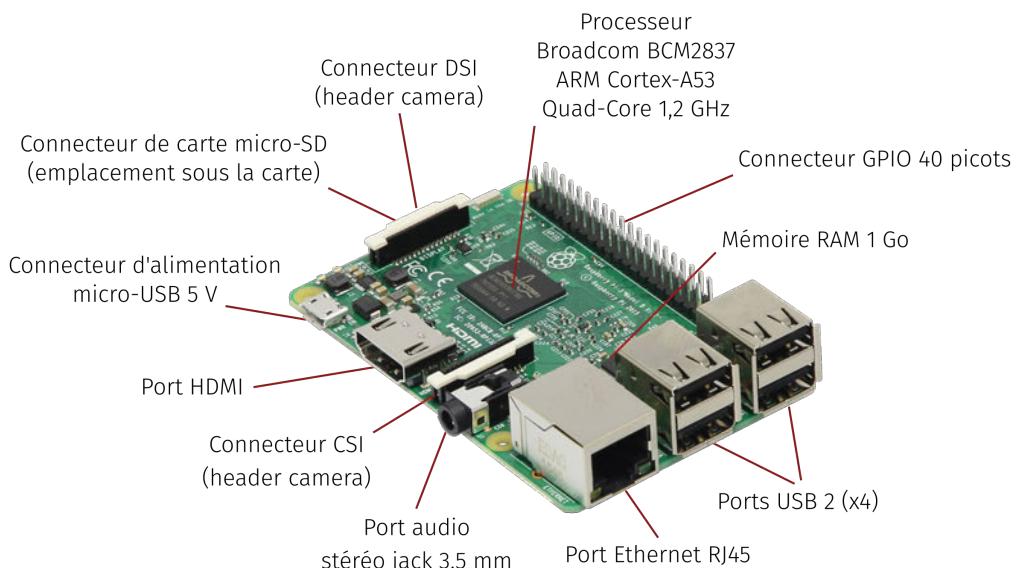


FIGURE 1.4 — Descriptif du nano-ordinateur RASPBERRY PI modèle 3B.

En l'état, ce dispositif ne contient pas tous les composants pour pouvoir être opérationnel. Outre un écran, un clavier et une souris avec leurs câbles associés, il manque en effet une alimentation extérieure et une carte mémoire dite flash — au format dit « *micro-SD* » pour faire

office régalien de « disque dur » : bien entendu, il est nécessaire de pouvoir enregistrer les données personnelles quelque part.

Enfin, pour les RASPBERRY PI les plus anciens, on peut également prévoir un *dongle* USB pour le WiFi; les plus récents disposent de puces Bluetooth 4.1/4.2 et WiFi n/ac.

Avant de pouvoir faire fonctionner cet ensemble, il faut encore au préalable installer un OS/système d'exploitation (cf. W) sur la carte micro-SD. Toujours pour des raisons de coût, de dimensionnement et de modularité, le système dûment conseillé est LINUX. La distribution « historique » du RASPBERRY PI, intitulée RASPIAN, s'est ainsi établie sur une base GNU/LINUX en version DEBIAN afin rester dans l'esprit d'ouverture pédagogique du projet initial, incluant en particulier des outils de programmation comme SCRATCH et PYTHON.

Compte-tenu du succès du projet, de multiples autres possibilités de systèmes d'exploitation libres sont envisageables selon la destination que l'on veut donner à sa machine : serveur réseau ou multimédia, bureautique, domotique, etc. L'essentiel est de choisir un système dont, en particulier l'interface graphique, ne soit pas trop gourmande pour ne pas « plomber » le RASPBERRY PI. Par exemple, une distribution généraliste comme UBUNTU MATE a été portée pour cette plateforme.

Passée sous silence jusqu'à présent, une dernière chose est à mentionner avec le RASPBERRY PI et qui s'avère non disponible avec un ordinateur conventionnel : le connecteur GPIO — General Purpose for Input/Output (cf. figure 1.4). Il s'agit d'une interface qui offre la possibilité de communiquer avec d'autres systèmes électroniques et permet au RASPBERRY PI d'être utilisé en tant que contrôleur pour, par exemple, piloter un robot, un système de capteurs ou un système embarqué.

GLOSSAIRE CONTEXTUEL

CONNECTEUR — Élément qui assure une connexion physique pour établir une liaison électrique ou une transmission de données entre deux entités matérielles distinctes. En informatique, ce terme est également employé pour désigner un port — même si le connecteur caractérise aussi la fiche mâle de la connectique qui lui est associée.

GPIO — GPIO, acronyme de *General Purpose Input/Output*, est un port d'entrée-sortie qui permet à une carte électronique — carte à micro-contrôleur ou carte-mère — de communiquer avec un périphérique quelconque. Ce type de port est programmable, permettant de s'adapter à une vaste diversité de périphériques, sous condition d'écrire un pilote adéquat.

DEBIAN — DEBIAN est le nom d'une distribution non commerciale qui repose exclusivement sur des logiciels libres. Elle est développée et maintenue par l'association DEBIAN, dont une particularité est qu'elle suit un mode de gouvernance communautaire. La version DEBIAN pour RASPBERRY PI peut se télécharger sur le site de la fondation Raspberry, soit via un installateur (NOOBS), soit via une image de la distribution (RASPIAN). Plus largement, la qualité et la stabilité reconnues de DEBIAN ont engendré de nombreuses distributions populaires la prenant comme fondement, par exemple UBUNTU ou LINUX MINT.

SYSTÈME D'EXPLOITATION — Un « système d'exploitation » — *Operating system* en anglais —, est un ensemble de programmes qui gèrent l'utilisation des ressources matérielles d'un ordinateur : stockage des mémoires à accès rapide et de masse des disques

durs, ordonnancement des calculs du processeur, communication avec les périphériques ou administration du réseau. Le système accepte ou refuse ces demandes, puis réserve les ressources en question pour éviter que leur utilisation n'interfère avec d'autres requêtes provenant d'autres logiciels.

NOTE DE LA RÉDACTION

Texte rédigé par Sacha KRAKOVIAK et publié sur [Interstices](#) — revue en ligne de culture scientifique du numérique — le 18 novembre 2011.

1.2.2 Architecture de von NEUMANN

Depuis plus de soixante ans, l'architecture des ordinateurs est conforme à un schéma qui a peu évolué depuis son origine : le modèle dit de « von NEUMANN ». La naissance de ce modèle, sa diffusion et ses premières mises en œuvre sont un moment-clé de l'histoire de l'informatique et des calculateurs.

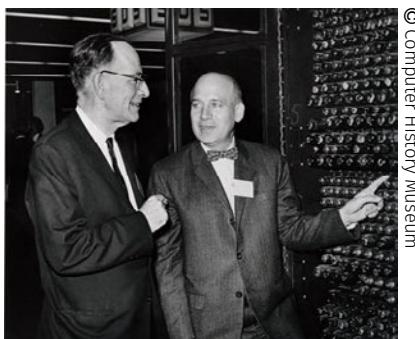
Le tableau de ce qu'en 1945 on n'appelait pas encore l'informatique présente un paysage contrasté. D'un côté, la notion de calcul effectif a trouvé un cadre rigoureux grâce aux avancées d'une discipline nouvelle née dans les années 1930, la méta-mathématique. Le lambda-calcul du mathématicien Alonzo CHURCH et la machine universelle (abstraite) d'Alan TURING, schémas dont TURING a montré l'équivalence, sont proposés en 1936 comme base de définition de l'algorithme, pièce maîtresse du processus calculatoire. D'un autre côté, plusieurs tentatives indépendantes visent à construire des machines électroniques ou électro-mécaniques capables d'exécuter des calculs complexes à grande vitesse. Les précurseurs en sont John ATANASOFF en 1938 aux États-Unis et Konrad ZUSE en 1941 en Allemagne.

Ces deux courants, celui des mathématiciens et logiciens d'une part et, celui des ingénieurs d'autre part, sont issus de deux mondes séparés et s'ignorent mutuellement. Les travaux de TURING ont sans doute eu une influence sur la conception en 1943-44 du [calculateur COLOSSUS](#) à Bletchley Park en Angleterre, mais il s'agit d'une machine spécialisée dont le seul objectif, qui sera d'ailleurs atteint, est le déchiffrement du code secret de la machine LORENZ, successeur de l'[ENIGMA](#), utilisée par l'armée allemande.

La guerre aura d'autres effets : les autorités allemandes ne soutiendront que modestement les travaux pionniers de ZUSE, alors que le département américain de la Défense financera un projet ambitieux lancé en 1943 à l'Université de Pennsylvanie par John Adam Presper ECKERT et John William MAUCHLY. Cet effort aboutira à la construction d'un grand calculateur électronique, l'[ENIAC](#), qui ne sera néanmoins pleinement opérationnel qu'en 1946. À cette même époque (1944), l'informaticien Howard AIKEN mène un autre grand projet à Harvard avec la collaboration d'IBM, mais la technique choisie est fondée sur l'électromécanique. Bien plus fiable que les tubes électroniques, cette voie ne sera toutefois pas poursuivie, mais l'expérience acquise sera exploitée plus tard par IBM dans la conception de ses premiers ordinateurs.

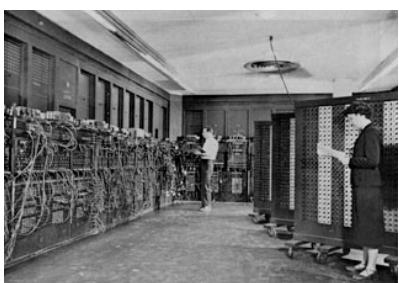
L'[ENIAC](#) — *Electronic Numerical Integrator and Computer* — fut le premier calculateur entièrement électronique possédant les mêmes capacités qu'une machine de TURING, aux limites physiques près. Il avait des dimensions imposantes : plus de 20 m de long, 2 m 50 de haut pour une masse de 30 tonnes. Comportant 18 000 tubes électroniques, il consommait 150 kilowatts.

Les données étaient lues sur cartes perforées, mais le programme était représenté sur un support externe, sous la forme d'un panneau de connexion analogue à celui d'un standard téléphonique. Pour programmer une application (initialement, le calcul de tables de tir pour l'artillerie), il fallait faire un plan des connexions nécessaires, puis procéder au câblage physique, un travail long, fastidieux et sujet aux erreurs.



Computer History Museum

À gauche, John William MAUCHLY (1907-1980) et à droite, John Adam Presper ECKERT (1919-1995).



© US Army

ENIAC, à gauche; panneau de connexion.

La détection et la correction des fautes étaient également laborieuses. Avec la fiabilité réduite des tubes électroniques, ce mode de programmation constituait le grand point faible du projet. Bien conscients de cette faiblesse, ses concepteurs ont commencé dès 1944 à réfléchir à l'étape suivante, avant même la mise en service de l'ENIAC.

En 1944, John von NEUMANN est introduit dans le projet ENIAC par Herman GOLDSTINE, qui assurait la liaison scientifique du projet avec le département de la Défense. VON NEUMANN était un esprit universel, dont les contributions allaient des mathématiques à la logique, la physique et l'économie. Il avait rencontré Alan TURING et connaissait ses travaux. Il participait alors au projet Manhattan et l'Histoire dit que GOLDSTINE lui parla du projet ENIAC lors d'une rencontre fortuite sur un quai de gare. Quoi qu'il en soit, von NEUMANN accepta un poste de consultant dans ce projet et prit une part active aux travaux menés par ECKERT et MAUCHLY sur la conception de l'*EDVAC* — *Electronic Discrete Variable Automatic Computer* —, le successeur de l'ENIAC. En juin 1945, une première version, incomplète, d'un rapport sur la conception de l'EDVAC fut diffusée par GOLDSTINE, sous la signature du seul VON NEUMANN qui l'avait rédigé comme document de travail. ECKERT et MAUCHLY en furent, à juste titre, profondément choqués; par ailleurs, ils entrèrent en conflit avec l'Université de Pennsylvanie pour des questions de brevet et ces deux circonstances provoquèrent leur départ du projet en mars 1946 pour fonder leur entreprise, ECKERT-MAUCHLY COMPUTER CORPORATION. VON NEUMANN lui-même quitta le projet à la même époque pour Princeton, où il travailla avec Julian BIGELOW sur le calculateur de l'IAS — Institut d'études avancées.

A. ARCHITECTURE NOVATRICE

Le *First Draft of a Report on EDVAC* est un document de cent-une pages qui décrit, d'une part un schéma d'architecture de calculateur, organisé en trois éléments (unité arithmétique, unité de commande et mémoire contenant programme et données) et, d'autre part, des principes de réalisation pour ces éléments, notamment les opérations arithmétiques. Si ce dernier aspect dépend partiellement de la technologie connue à l'époque et a donc nécessairement vieilli, le modèle d'architecture, qui marque une transition profonde avec les pratiques antérieures, reste d'une étonnante actualité. Ce modèle, auquel reste attaché le nom de VON NEUMANN, est exposé par le schéma en figure 1.5.

La première innovation est la séparation nette entre l'unité de commande, qui organise le flot de séquencement des instructions et l'unité arithmétique, chargée de l'exécution proprement dite de ces instructions. La seconde innovation, la plus fondamentale, est l'idée du programme enregistré : les instructions, au lieu d'être codées sur un support externe (ruban, cartes, tableau de connexions), sont enregistrées dans la mémoire selon un codage conventionnel. Un compteur ordinal contient l'adresse de l'instruction en cours d'exécution ; il est automatiquement incrémenté après exécution de l'instruction et explicitement modifié par les instructions de branchement.

Un emplacement de mémoire contient indifféremment des instructions et des données. Une conséquence majeure (dont toute la portée n'avait probablement pas été perçue à l'époque) est qu'un programme peut être traité comme une donnée par d'autres programmes. Cette idée, présente en germe dans la machine de TURING, trouve ici sa véritable concrétisation.

B. DIFFUSION ET MISE EN ŒUVRE DU MODÈLE

Le rapport Edvac circula largement et eut une influence notable sur bon nombre de travaux ultérieurs. En juillet et août 1946, le dépar-



© Union postale universelle
Timbre à l'effigie de von NEUMANN (1903-1957) émis à titre posthume par son pays natal, la Hongrie.

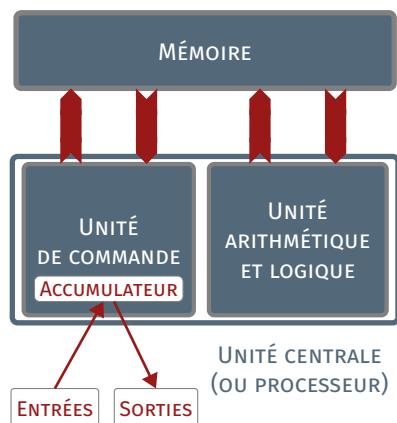


FIGURE 1.5 — Modèle originel d'architecture des ordinateurs de von NEUMANN.



© U.S. Army

EDVAC installé dans le bâtiment 328 du Ballistics Research Laboratory.

tement de la Défense finança une école d'été pour diffuser les connaissances acquises dans les différents projets de calculateurs. Organisées par la *Moore School* de l'Université de Pennsylvanie, qui abritait le projet ENIAC, ces *Moore School Lectures* furent un instrument de réflexion sur les concepts et techniques de base des calculateurs et surtout de diffusion des travaux menés dans les projets ENIAC et EDVAC. ECKERT et MAUCHLY, ainsi que GOLDSSTONE, furent les principaux intervenants, mais de nombreux autres conférenciers furent invités, dont Howard AIKEN et George STIBITZ, qui avaient construit des prototypes de calculateurs. Le seul conférencier non-américain était Douglas HARTREE, de l'Université de Manchester. Les auditeurs, au nombre d'une quarantaine, avaient tous été préalablement sélectionnés et invités.

Le projet EDVAC lui-même se trouva fortement perturbé par les départs d'ECKERT et de MAUCHLY, que suivirent plusieurs de leurs ingénieurs et ne fut achevé qu'en 1951. Son architecture était fondée sur le modèle présenté dans les *Moore School Lectures*, qui avait évolué par rapport au document originel. Entre temps, c'est en Europe que le modèle de VON NEUMANN trouva ses premières réalisations.

MACHINES DE MANCHESTER — En 1946, Frederic WILLIAMS, alors ingénieur au *Telecommunications Research Establishment* (TRE) en Angleterre — le siège de l'invention du radar —, travaillait sur une mémoire utilisant l'écran d'un tube cathodique pour le stockage de bits. Il commença une collaboration avec Thomas KILBURN, de l'Université de Manchester, et obtint un poste dans cette université. L'avantage de la mémoire à tube sur la ligne à retard (technologie dominante à l'époque) était de permettre un accès aléatoire plutôt que séquentiel.

En 1947, WILLIAMS et KILBURN décidèrent d'expérimenter leur mémoire connue sous le nom de « tube WILLIAMS » sur un mini-projet de calculateur qu'ils baptisèrent *Baby*. La taille de la mémoire était de 2 048 bits. En juin 1948, le premier programme tourna sur le *Baby*; il avait été inséré bit par bit sur l'écran cathodique.

WILLIAMS, KILBURN et leur équipe, que TURING rejoignit en 1948 mais sans y jouer de rôle majeur, construisirent ensuite le calculateur *Manchester Mark-1*. La mémoire à tube était complétée par une mémoire secondaire à tambour magnétique, ce qui était une innovation majeure. *Mark-1* fonctionnait mi-1949. La compagnie Ferranti qui était dès cette époque associée au projet, exploita le projet *Mark-1* pour servir de base à son premier produit commercial.

EDSAC DE CAMBRIDGE — Maurice WILKES, qui dirigeait le *Mathematical Laboratory* de l'Université de Cambridge et avait travaillé au TRE sur le radar, perçut très vite les possibilités ouvertes par le rapport EDVAC. Il assista aux *Moore School Lectures* en 1946 et lança dès son retour le projet d'un calculateur à programme enregistré appelé **EDSAC** — *Electronic Delay Storage Automatic Calculator*. Comme l'indique son nom, la mémoire utilisait des *lignes à retard*. Le premier programme fonctionna en mai 1949. On considère généralement l'*EDSAC* comme le premier calculateur à programme enregistré. Bien qu'un programme enregistré ait fonctionné auparavant sur le *Manchester Baby*, ce dernier n'était pas un calculateur complet mais essentiellement un outil de validation du tube WILLIAMS. Quoi qu'il en soit, Manchester et Cambridge ont été des pionniers en la matière.

L'*EDSAC* comportait 1 024 emplacements de mémoire de 18 bits et représentait les nombres entiers en *complément à 2* — mode de codage le plus courant à l'heure actuelle. Il n'avait pas à proprement parler de système d'exploitation, mais des « ordres initiaux » câblés, qui remplissaient les fonctions d'un chargeur. Une invention déterminante,



Source : Wikipedia

Tube de WILLIAMS-KILBURN d'un IBM 701 au Computer History Museum, Mountain View, Californie.



© Computer Laboratory, University of Cambridge

Maurice WILKES (1913-2010) face à la mémoire de l'*EDSAC*.

due à David WHEELER, fut celle des sous-programmes. La bibliothèque de sous-programmes — une centaine — était stockée sous forme de bandes de ruban perforé. Pour utiliser un sous-programme, on devait le copier physiquement sur le ruban contenant le programme utilisateur. La fonction d'éditeur de liens était donc réalisée à la main par des moyens mécaniques!

Le guide du programmeur¹⁵ de l'EDSAC, qui décrit notamment l'utilisation des sous-programmes, fut alors le premier ouvrage consacré à la programmation. WILKES inventa aussi en 1951 la micro-programmation, exploitée seulement dix ans plus tard dans la famille des IBM/360.

AUTRES PROJETS — Au début de 1946, Alan TURING lança au *National Physical Laboratory* (NPL) un projet de calculateur à programme enregistré, l'*Automatic Computing Engine* (ACE). TURING ne pouvait pas faire état de l'expérience du COLOSSUS, protégé par le secret militaire. Cette circonstance, ajoutée à des problèmes d'organisation, retarda l'avancement des travaux et TURING quitta le NPL en 1948 pour Manchester. Le projet n'aboutit qu'en 1950.

Le projet IAS fut lancé à Princeton par VON NEUMANN en 1946. Son principal ingénieur était BIGELOW. La machine commença à fonctionner à l'été 1951 et devint opérationnelle un an plus tard. Elle possédait 1 024 emplacements de mémoire de 40 bits, réalisés par des tubes WILLIAMS.

C. PREMIÈRES MACHINES COMMERCIALES

La compagnie FERRANTI, qui fabriquait des équipements électriques et électroniques, collaborait depuis 1948 avec l'équipe de Manchester. Son premier produit, le FERRANTI Mark-1 était directement inspiré du Manchester Mark-1. Sorti en février 1951, ce fut le premier calculateur diffusé commercialement.

La compagnie ECKERT-MAUCHLY (EMCC — Eckert-Mauchly Computer Corporation) a continué le travail commencé lors de la conception de l'EDVAC, en préparant un EDVAC-2, qui fut rebaptisé ensuite UNIVAC. En attendant l'aboutissement de ce projet de grande ampleur, EMCC construisit le calculateur BINAC pour la société NORTHROP en 1949. Confrontée à des difficultés financières, EMCC fut rachetée en 1950 par REMINGTON RAND (fabricant d'armes et de machines à écrire) et devint la division UNIVAC de REMINGTON RAND. Le premier UNIVAC-1 fut livré en juin 1951. La société UNIVAC resta longtemps une référence pour les ordinateurs de grande puissance.

Comparativement, le premier ordinateur d'IBM, le 701 (qui utilisait les tubes WILLIAMS), fut annoncé mi-1952.

D. QU'EN EST-IL AUJOURD'HUI ?

Plus de soixante ans après son invention, le modèle d'architecture de VON NEUMANN régit toujours l'architecture des ordinateurs. Par rapport au schéma initial, on peut noter deux évolutions essentielles :

- les entrées-sorties, initialement commandées par l'unité centrale, sont depuis le début des années 1960 sous le contrôle de processeurs autonomes (canaux d'entrée-sortie et mécanismes assimilés). Associée à la multiprogrammation (partage de la mémoire entre plusieurs programmes), cette organisation a notamment permis le développement des systèmes en temps partagé;
- les ordinateurs disposent maintenant de processeurs multiples, qu'il s'agisse d'unités séparées ou de « cœurs » multiples à l'intérieur d'une même puce. Cette organisation permet d'atteindre une puissance de calcul élevée sans accroître la vitesse des processeurs individuels, limitée par les capacités d'évacuation de la chaleur dans des circuits de plus en plus denses.

¹⁵. M. V. WILKES, D. J. WHEELER, S. GILL, *The preparation of programs for an electronic digital computer*. Addison-Wesley, 1951.



Source : Wikipedia W

Console de l'UNIVAC-I.

Ces deux évolutions ont pour conséquence de mettre la mémoire, plutôt que l'unité centrale, au centre de l'ordinateur et d'augmenter ainsi le degré de parallélisme dans le traitement et la circulation de l'information. En revanche, elles ne remettent pas en cause les principes de base que sont la séparation entre traitement et commande, ainsi que la notion de programme enregistré.

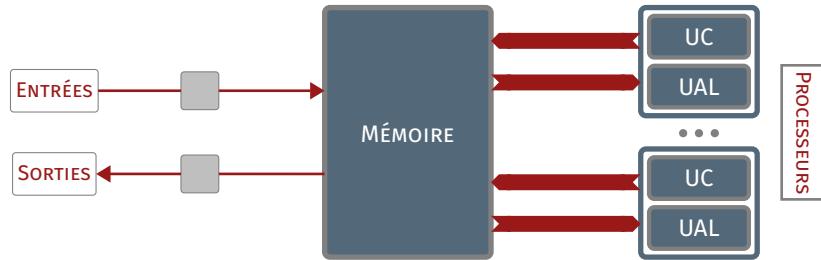


FIGURE 1.6 — Modèle actuel de von NEUMANN.

L'accès des processeurs à la mémoire se fait à travers un bus (non représenté sur la figure 1.6), voie d'échange assurant un transfert rapide de l'information. Mais au cours du temps, pour des raisons technologiques, le débit du bus a crû moins vite que le débit d'accès à la mémoire et surtout que la vitesse des processeurs. D'où un phénomène d'attente — le « goulot de von NEUMANN » — qui réduit les performances. Des palliatifs sont associés à l'usage généralisé de caches à plusieurs niveaux (mémoire d'accès rapide, voisine du processeur qui retiennent les données courantes) et au développement de machines à mémoire distribuée, mais se posent alors des problèmes de cohérence pour les données en copies multiples.

Quel modèle de machine après von NEUMANN ? La question, posée depuis longtemps, n'a pas de réponse claire aujourd'hui (voir le document « Calculer différemment », Interstices, 2011).

POUR ALLER PLUS LOIN... ▶

AUTOUR DU RASPBERRY PI

- ▶ Manuel d'utilisation du RASPBERRY PI — format PDF, *Adventures in Raspberry Pi & Raspberry Pi User Guide* —, à destination des adolescents pour découvrir les possibilités du RASPBERRY PI;
- ▶ *RASPBERRY PI : la petite histoire d'une grande idée*, Binaire, 28 décembre 2015;
- ▶ *Comment installer un RASPBERRY PI, ce nano-ordinateur à moins de 50 euros*, avec une présentation vidéo de RASPBERRY PI par Alan McCULLAGH, décembre 2016.

ARCHITECTURES AVANCÉES POUR PLUS DE PUISSANCE DE CALCUL

- ▶ *Le futur de l'ordinateur ? L'ordinateur quantique*, Philippe Jorrand, Interstices, 2005;
- ▶ *Architecture et performance : les super-ordinateurs*, Charlotte Truchet, binaire, 08 avril 2016;
- ▶ *Les machines d'aujourd'hui et de demain*, Albert Cohen, CanalU, collection Inria Science Info Lycée Prof; 86 mn. montrant les liens entre science informatique et architecture des ordinateurs;
- ▶ *Microcontrôleur : Comment ça marche ?* — SILIS Electronique, 25 février 2016 : 5 mn 34 s — Tout objet numérique n'a pas un processeur, une carte graphique, un disque dur... Alors pourquoi dit-on que tous ces objets fonctionnent sur le même schéma ? Parce qu'ils intègrent un *microcontrôleur* qui suit la

même architecture matérielle. Cette [vidéo](#) en donne une introduction claire.

2 Système d'exploitation

En usage courant d'un ordinateur — quel qu'il soit —, lors du lancement d'une application de lecture de vidéo, les images apparaissent à l'écran et du son est émis par les haut-parleurs. Dans le même temps, le gestionnaire de courrier électronique peut se manifester en prévenant de l'arrivée d'un nouveau message, auquel il est possible de répondre en saisissant du texte au clavier sans avoir à quitter le lecteur vidéo — et, en toute convenance pratique, sans obliger à reprendre la lecture depuis le début alors que le dénouement est proche !

Comment une application peut-elle ainsi interagir avec les périphériques matériels ? Comment deux applications peuvent-elles tourner en même temps sur un seul processeur ? Tout ceci est possible grâce au *système d'exploitation*, une couche logicielle intermédiaire entre la couche applicative et celle du matériel, dont les principes de fonctionnement sont à découvrir en trois concepts clefs.

2.1 Système d'exploitation en trois idées-forces

WINDOWS, LINUX, MACOS, ANDROID, IOS ou autre, sont des noms régulièrement entendus correspondant à une utilisation quotidienne mais, au final, à quoi cela fait-il vraiment référence ?

Ce sont en fait des logiciels spécialisés qui sont appelés *systèmes d'exploitation*. Un *OS/système d'exploitation* sert dans sa fonction première à gérer le lien entre tout ce qui appartient aux mondes applicatifs — autrement dit les logiciels utilisés — et le matériel à disposition tels que les processeurs, les disques durs, les écrans et ainsi de suite...

Il faut se souvenir qu'un ordinateur est essentiellement composé de trois éléments principaux : la *mémoire*, — qui sert à stocker de l'information et à y accéder quand on en a besoin —, le ou les *processeur(s)* — cerveau(x) de l'ordinateur, il(s) exécute(nt) les calculs et les tâches sur la base des informations qu'il(s) récupère(nt) en mémoire — et enfin, pour communiquer avec le monde extérieur, les *périphériques d'entrée/sortie* qui prennent en compte une commande ou restituent un résultat, qu'il s'agisse d'une imprimante, d'un écran, d'un clavier ou d'une souris, etc. Dans cette perspective, Internet peut également être considéré comme un périphérique d'entrée et de sortie.

Bien entendu, un ordinateur fait tourner plusieurs *applications* en même temps ; chacune d'entre-elle effectuant une tâche bien précise. Afin d'utiliser le matériel, on ne va pas les autoriser à accéder directement aux différents composants de l'ordinateur. C'est le rôle du système d'exploitation — *Operating System* en anglais — que d'être l'intermédiaire entre *application* et matériel et d'établir le lien entre les différents éléments de l'ordinateur. Ces derniers se déclinent en trois catégories principales : la mémoire, les processeurs et les entrées-sorties.

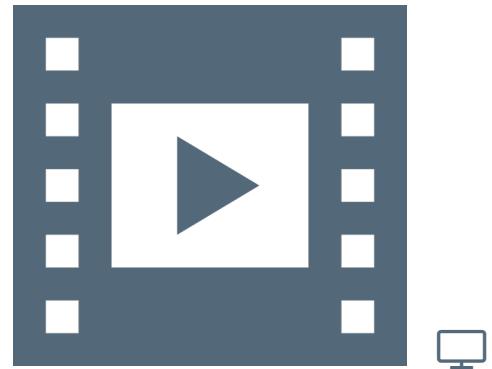
2.1.1 Gestion de la mémoire

Pour se faire une idée, la mémoire peut s'imaginer comme un grand tableau disposant de plusieurs entrées que l'on appelle des mots. Chaque mot correspond à une *adresse* de la mémoire physique.

Par exemple, en supposant que quatre mots sont à écrire en mémoire, ils vont se référer aux adresses numérotées de 0¹⁶ à 3. En pre-

À PROPOS DE L'INTERVENANT

Damien SAUCEZ est chercheur au sein de l'équipe INRIA-DIANA. Il consacre ses travaux aux réseaux information-centrés (*Information-centric networking*) — par exemple, les problèmes de routage et de contrôle de la congestion —, aux réseaux définis par logiciel (*Software Defined Networking*) — en particulier, les questions de résilience et de robustesse — et aux expérimentations à large échelle. Il est un contributeur actif de l'IETF et de l'IRTF.



VIDÉO 1.3 — Système d'exploitation.

¹⁶. Dans la réalité c'est un peu plus complexe, mais le principe est identique.



FIGURE 1.7 — Principe de fonctionnement de la mémoire virtuelle.

mière approche, on pourrait alors considérer qu'une application qui souhaite enregistrer le mot « toto » en mémoire physique, lui attribue par exemple l'adresse n°1. Une fois écrit en mémoire, l'application peut y accéder. En imaginant qu'une deuxième application veuille écrire le mot « titi » elle aussi à l'adresse n°1, cela va écraser le mot « toto » pour le remplacer par « titi ». Il y a clairement un problème, car la première application a perdu le mot dont elle a besoin et récupère à la place un mot inconnu. Cela ne fonctionne pas.

Le système d'exploitation permet d'éviter ce genre de déconvenues grâce au concept de *mémoire virtuelle*. L'idée sous-jacente est que le système d'exploitation va donner à chaque application l'illusion qu'elle dispose de toute la mémoire. La mémoire virtuelle s'apparente donc toujours à un tableau de mots avec leurs adresses respectives. En suivant l'exemple précédent, l'application va donc, de son point de vue, toujours écrire le mot « toto » à l'adresse n°1, mais en mémoire virtuelle. Quant à lui, le système d'exploitation est en charge de trouver un espace où écrire le mot dans la réelle mémoire physique et, bien entendu, de transcrire le lien entre mémoire virtuelle et mémoire physique. Dès lors, il n'y a plus de conflit de mémoire et de problème d'inconsistance entre les différentes applications lancées : ce qui va être l'adresse n°1 pour chaque application considérée, correspond en fait à plusieurs adresses physiques.

2.1.2 Gestion du processeur

Un ordinateur contient au moins un processeur et, s'il en possède plusieurs, leur nombre est toujours largement inférieur à celui des applications qui sont potentiellement à faire tourner conjointement.

Le système d'exploitation sert d'arbitre entre les diverses applications, de manière à partager « équitablement¹⁷ » l'accès au(x) processeur(s). En effet, un processeur est « stupide » ; il se contente uniquement d'effectuer des calculs¹⁸ et de ne réaliser qu'une tâche à la fois. C'est un peu comme dans un débat, la parole est concédée chacun son tour par le modérateur. Le système d'exploitation agit de manière similaire par l'intermédiaire de l'*ordonnanceur*. L'ordonnanceur établit ainsi qu'une application a assez effectué de calculs pour la mettre en pause et offrir l'accès au processeur aux autres applications.

Il existe néanmoins des règles d'optimisation. Par exemple, étant donné que les temps d'accès aux disques durs sont relativement plus importants que ceux des mémoires caches ou de la mémoire vive (RAM), au lieu d'attendre l'aller-retour avec le disque dur pour récupérer une information nécessaire à l'algorithme de calcul — et perdre ainsi un temps précieux —, le système d'exploitation place en réserve l'application en cours pour libérer le processeur et exécuter une autre application avant de reprendre le traitement une fois l'information obtenue.

2.1.3 Entrée-sortie : appels système

La typologie des ordinateurs est très disparate : un téléphone mobile, une carte à puces, un distributeur de billets, une station personnelle de bureautique, un *cluster* de calcul — simulation météo, automobile ou aéronautique —, un serveur de *Data center*, etc.

L'enjeu principal est de pouvoir écrire ou plus exactement programmer des applications qui, dans la mesure du possible, possèdent le même code de base pour chaque configuration matérielle sous-jacente, sans avoir besoin de la connaître. On parle de développement multi-plateforme. Bien que tout ne soit pas toujours possible aussi directement, les choses ont considérablement évolué ces dernières décennies. Cependant, dans nombre de cas d'utilisations courantes, cela

¹⁷. Il existe une hiérarchie dans les applications, que ce soit en termes de sécurité, de priorité ou autre critère prépondérant.

¹⁸. Certes, les processeurs sont suffisamment denses en circuits imprimés, donc véloces pour donner l'impression qu'il réagissent simultanément ou « en temps réel » aux instructions. Que *nenni* !

s'avère envisageable via le système d'exploitation, lequel fournit une couche d'abstraction entre matériel et logiciel : les *appels systèmes*.

Quid des appels systèmes ? À chaque fois qu'une application a besoin d'exécuter une tâche, comme déjà vu, elle ne va pas directement accéder au matériel, mais formuler et déléguer au système d'exploitation sa demande d'intervention. Par exemple, dans le cas d'un jeu, l'application veut afficher un rectangle¹⁹ à l'écran. Le système d'exploitation va d'abord vérifier que l'application est autorisée à le faire puis, dans l'affirmative, examiner comment opérer et enfin, réaliser la tâche en tant que telle en dessinant le rectangle voulu à l'écran.

Autre exemple, un microphone. Une application souhaite enregistrer le son fourni par un microphone connecté à l'ordinateur. Une fois la requête émise, le système d'exploitation s'attache à « ouvrir le canal microphone » et se met à l'écoute. Lors de l'apparition d'un événement sonore, le système d'exploitation génère une interruption, enregistre le signal d'entrée et le redirige sur l'application demanduse.

¹⁹. Dans la pratique et au delà des jeux, les systèmes d'exploitation ne sont pas forcément associés à une interface graphique. C'est par exemple le cas des serveurs : une interface graphique apportant potentiellement des failles de sécurité. Néanmoins, les autres configurations à vocation d'interaction avec un utilisateur incluent une interface graphique proche du système d'exploitation : on parle alors de bibliothèque graphique associée. À titre d'illustration, pour les systèmes GNU/LINUX, il existe deux interfaces majeures : GTK+/GNOME et Qt/KDE.

GLOSSAIRE CONTEXTUEL

OS/SYSTÈME D'EXPLOITATION — OS, acronyme de *Operating system*, est l'expression anglophone signifiant « *système d'exploitation* ». Il s'agit d'un ensemble de programmes qui gèrent l'utilisation des ressources matérielles d'un ordinateur : stockage des mémoires à accès rapide et de masse des disques durs, ordonnancement des calculs du processeur, communication avec les périphériques ou administration du réseau. Le système d'exploitation accepte ou refuse ces demandes, puis réserve les ressources en question pour éviter que leur utilisation n'interfère avec d'autres requêtes provenant d'autres logiciels.

APPLICATION — Une *application* ou un *applicatif* est, dans le domaine informatique, un *programme* — ou un ensemble *logiciel* — directement manipulé par l'utilisateur pour réaliser une tâche ou un ensemble de tâches élémentaires d'un même domaine formant un tout. Typiquement, un *éditeur de texte*, un *navigateur Web*, un *client de messagerie*, une *visionneuse d'images*, un *lecteur multimédia*, un *jeu vidéo*, sont des applications.

MÉMOIRE — Dispositif électronique qui sert à stocker des données (valeurs, instructions de programmes). Dans un ordinateur, il existe trois grands types de mémoire externe au processeur : la *mémoire vive (RAM)* (carte branchée sur la carte mère), la *mémoire morte (ROM)* (circuit intégré à la carte mère) et la *mémoire de masse* (disques durs, et autres supports de stockage périphérique). Le processeur intègre aussi plusieurs niveaux de mémoire, notamment de la *mémoire cache* (placée dans la puce qui contient le processeur) et les *registres* (internes au processeur).

MÉMOIRE VIRTUELLE — Mécanisme qui permet de simuler la présence d'un type de mémoire en utilisant un autre type (par exemple un disque dur). Il est utilisé par exemple pour simuler la présence de mémoire vive en utilisant de la mémoire de masse. Sous LINUX, ce type de mécanisme est dénommé « *swap* » où à l'installation du système d'exploitation, une partition du disque dur est réservée à cet effet (on conseille souvent une taille double de celle de la mémoire vive de l'ordinateur).

PROCESSUS — Forme que prend un programme quand il est exécuté au sein d'un système d'exploitation. Une instance de proces-

sus comprend en général : un ensemble d'instructions (souvent copiées dans la RAM depuis le disque dur), une place, appelée *espace d'adressage*, réservée en mémoire vive pour stocker les données qu'il manipule et les ressources matérielles et logicielles que le programme utilise.

ORDONNANCEUR — Composant logiciel du système d'exploitation qui est en charge d'allouer du temps processeur aux processus. Le choix du processus élu pour s'exécuter sur le processeur à un temps donné est fait par une procédure d'ordonnancement et implique de gérer les changements de contexte.

EN BREF...

Ainsi, le système d'exploitation gère la mémoire au moyen de la mémoire virtuelle, priorise l'accès au processeur entre les différentes applications concurrentes et abstrait les détails techniques de la machine pour avoir des codes d'application les plus transparents possibles vis-à-vis du matériel constituant l'ordinateur ; d'où, notamment, la notion et la présence de pilotes de périphérique — ou *drivers* en anglais —, proches du noyau du système d'exploitation et à considérer comme extension, qui s'avèrent nécessaires au fonctionnement des matériels non supportés nativement par le système d'exploitation.

2.2 À quoi sert un système d'exploitation ?

NOTE DE LA RÉDACTION

Texte rédigé par Claude KAISER et publié sur [Interstices](#) — revue en ligne de culture scientifique du numérique — le 23 juin 2015.

On se moque aisément de Monsieur JOURDAIN qui faisait de la prose sans le savoir. Mais en utilisant un ordinateur ou une tablette, se déclenche sans le savoir tout un flot de programmes souterrains, ceux du système d'exploitation, dont les rôles importants demeurent cachés.

L'informatique est une science jeune, comportant beaucoup d'aspects techniques et une forte compétition industrielle. Aujourd'hui, il existe un système d'exploitation dans chaque ordinateur : tablette, téléphone portable et plus généralement tout objet numérique, avec sur le marché des dizaines de systèmes d'exploitation différents. Dans leur nom, on trouve souvent le sigle OS pour *Operating System* en anglais. Même si les constructeurs utilisent des noms différents pour se démarquer de leurs concurrents, on retrouve dans tous les systèmes d'exploitation des aspects communs et des invariants.

D'un ordinateur ou d'une tablette, on distingue d'abord l'écran et le boîtier qui enferme le matériel — processeurs, mémoires et canaux d'entrées-sorties —, mais c'est le logiciel qu'il contient que l'on utilise.

La programmation directe de l'ordinateur est laborieuse. La machine, construite sur le modèle de VON NEUMANN, est universelle (voir sur [Interstices](#) « *Sous le signe du calcul* », par Jean-Louis GIavitto et François RECHENMANN et « *Alan TURING : du concept à la machine* », par Sacha KRAKOWIAK). Toutefois, le langage qu'elle reconnaît, composé d'instructions et de données, toutes codées en binaire, est très difficile d'emploi. Le système d'exploitation vient alors au secours de l'utilisateur, en lui permettant d'exploiter l'ordinateur mieux qu'en langage machine. Mais il fait plus que cela.

A. RÔLES DU SYSTÈME D'EXPLOITATION

Le but d'un système d'exploitation est de rendre aisée l'utilisation de l'ordinateur par chacun, comme s'il s'agissait d'une machine fictive, sa « machine virtuelle », qui aurait été construite pour lui. Le système fournit un accès commode et ergonomique, par exemple avec un écran comportant des fenêtres multiples et une interface graphique. De nos jours, en 2015, il peut aussi y avoir une interface tactile ou sonore.

Le système assure ainsi le stockage de programmes et de données de toutes sortes — textes, images, vidéos, films — dans des fichiers pré-



Source : Flickr CC BY-NC 2.0

Manchot : la mascotte de GNU/LINUX, système d'exploitation libre populaire.

parés par l'utilisateur ou chargés depuis des supports externes ou via le réseau. Idéalement, ce premier rôle de gestion de l'information et des fichiers doit permettre à l'utilisateur de n'avoir à préciser que les aspects logiques de son travail.

Le second rôle du système, le contrôle d'exécution, consiste à gérer au mieux les ressources — matérielles et logicielles — qui sont nécessaires pour lancer et suivre l'exécution des applications locales ou distantes. Le système s'occupe ainsi des aspects technologiques et des contraintes d'utilisation des ressources partagées, qu'elles soient attribuées à tour de rôle, comme le sont le processeur et l'imprimante ou divisées, comme le sont la mémoire et l'écran.

Ces deux rôles doivent être pérennes. Pour cela, le système d'exploitation assure — et c'est là son troisième rôle — la sécurité de fonctionnement en cas de panne d'origine interne (matérielle ou logicielle) ou d'agression provenant d'un environnement de plus en plus ouvert (le réseau). Il sauvegarde automatiquement les travaux en cours et veille à permettre le redémarrage après une panne. Il doit aussi rendre possible une évolution matérielle (changement dans la configuration matérielle) ou fonctionnelle (mise à jour et ajout de programmes).

Ces rôles sont remplis par des services décrits dans les programmes du système d'exploitation; l'exécution de ces programmes — et donc des services — est accomplie par ses processus.

Un programme est une entité passive, décrivant une suite d'instructions. Un processus est son pendant dynamique, une entité active qui représente l'exécution de cette suite d'instructions par l'ordinateur.

B. REPRÉSENTATIONS DU SYSTÈME D'EXPLOITATION

Dans sa mouture générique, un système d'exploitation peut s'appréhender selon deux aspects :

- une vision « statique » qui correspond à l'empilement hiérarchique de ses programmes;
- une perspective « dynamique » qui rend compte de l'exécution des programmes par les processus.

REPRÉSENTATION STATIQUE : LES PROGRAMMES — Le système d'exploitation d'un ordinateur est un fantastique regroupement de programmes et de données qui ont été élaborés pour fournir les services requis pour chacun des rôles cités précédemment. Ces informations sont empilées astucieusement en couches qui reflètent la forte structuration de la construction : au niveau le plus bas, on trouve les programmes du noyau qui matérialisent les principes sur lesquels la construction du système est fondée, puis les programmes qui pilotent l'accès au matériel; au-dessus, dans l'ordre ascendant, utilisant parfois les programmes des couches qui leur sont sous-jacentes, viennent les services propres du système d'exploitation, les services communautaires et, enfin, les programmes d'application destinés à l'utilisateur. La figure 1.8 schématisise la représentation habituelle de ces services.

Cet empilement structuré d'informations forme une grande base de données : les fichiers du système. Il faut y ajouter les fichiers des utilisateurs entreposés sur des supports gérés par le système d'exploitation ou accessibles à distance sur le réseau.

La rapidité des processeurs et la grande capacité de la mémoire permettent de construire des systèmes très complexes et de taille considérable qui peut atteindre une dizaine de gigaoctets. Par exemple, un système d'exploitation utilisé pour un ordinateur portable, le système d'exploitation MacOSX 10.6, occupe environ 5 Go en mémoire, auxquels s'ajoutent de 5 à 15 Go pour les bibliothèques partagées.



FIGURE 1.8 — Représentation statique du système d'exploitation : descriptif et approche en couches (cf. [Interstices](#)).

ATTENTION!

Lien mort : https://interstices.info/jcms/c_19937/octet.

REPRÉSENTATION DYNAMIQUE : LES PROCESSUS — De nombreux programmes du système qui ont leurs sources — codes et données — dans ces empilements de couches sont appelés à s'exécuter de façon concomitante. Ils répondent en temps réel aux besoins explicites ou implicites des utilisateurs et réagissent aux événements aléatoires issus de l'environnement. Quelquefois, c'est le même service qui est demandé simultanément par plusieurs événements ou utilisateurs. Le système d'exploitation peut ainsi être amené à gérer l'exécution concurrente de plusieurs centaines de programmes. Le mot concurrent, polysémique, évoque à la fois la simultanéité — les programmes courent ensemble — et la compétition dans l'attribution des ressources.

Afin de repérer ces différentes exécutions concomitantes, le terme de processus a été introduit. Ce mot a été utilisé dès 1960 dans le système MULTICS. Il sert à identifier le déroulement d'un programme séquentiel — parmi d'autres — et le distingue du texte du programme.

La figure 1.9 schématise cette vision dynamique, qui traduit l'activité des processus. On y distingue les processus du système, à savoir les processus chargés des infrastructures et des services communautaires et les processus créés plus spécialement pour fournir une « machine virtuelle » à chaque utilisateur. Chacun des processus est une abstraction du processeur physique.

C. PROCESSUS ET PROCESSEURS

La représentation dynamique fait apparaître une nuée de processus, alors qu'il n'y a pas assez de processeurs physiques pour en attribuer un à chacun. Or les processeurs physiques ont une vitesse environ un million de fois supérieure à la capacité de réaction de l'utilisateur. Cette énorme différence est mise à profit pour simuler le parallélisme de déroulement des programmes en partageant les processeurs entre les processus — appelé le pseudo-parallélisme. Le partage est rythmé par un top d'horloge qui déclenche l'interruption du processus en cours et l'attribution du processeur à un autre processus. Cette attribution est régie par un programme du noyau du système, l'ordonnanceur.

La figure figure 1.10 montre un exemple de partage d'un processeur (on suppose ici qu'il n'y en a qu'un) entre quatre processus déclenchés simultanément qui l'utilisent l'un après l'autre, par tranches séquentielles successives. L'attribution du processeur par le noyau consomme aussi du temps de processeur.

D. BILAN

La complexité des systèmes d'exploitation est grande, elle résulte tant du nombre et de la taille des programmes impliqués, que de la multiplicité des processus et de leurs interactions. Cette grande complexité rend illusoire le « zéro défaut » !

Aujourd'hui, on continue de développer de nouveaux systèmes, tout en recherchant l'augmentation de leur sûreté de fonctionnement. Pour cela, on commence à utiliser des techniques automatiques de certification de programmes.

POUR ALLER PLUS LOIN...

MÉTAPHORE BADINE

- ▶ De votre boulangerie à un système d'exploitation multiprocesseur, par Brice GOGLIN, Interstices, 15 février 2011.

QUELQUES ASPECTS HISTORIQUES

- ▶ Naissance des systèmes d'exploitation, par Sacha KRAKOWIAK et Jacques MOSSIÈRE, Interstices, 5 avril 2013;

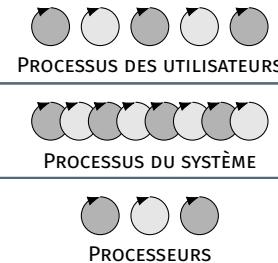


FIGURE 1.9 — Représentation dynamique du système d'exploitation : descriptif et approche en processus (cf. Interstices).

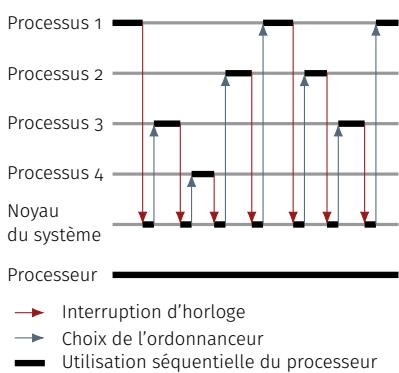


FIGURE 1.10 — Processeur partagé, prêté successivement à quatre processus (cf. Interstices).

COMPLÉMENT DE LECTURE

Pour comprendre dans le détail la dynamique d'un système d'exploitation et découvrir les processus qui se déclenchent lors du lancement d'un ordinateur, le lecteur est invité à consulter l'article « *Le ballet des processus dans un système d'exploitation* », Interstices, 2015.

- ▶ Débuts d'une approche scientifique des systèmes d'exploitation, par Sacha KRAKOWIAK, Interstices, 03 février 2014;
- ▶ Liste des systèmes d'exploitation maintenue sur Wikipédia. Pour les systèmes GNU/LINUX, se référer à DistroWatch.

APPRENDRE LINUX

- ▶ OPENCLASSROOMS propose un cours pour débuter en Linux.

COMMENT CRÉER SON PROPRE SYSTÈME D'EXPLOITATION ?

- ▶ Apprendre à programmer son propre noyau de système d'exploitation : une introduction avec Pépin, page en français qui date un peu, mais néanmoins relativement complète;
- ▶ OSDev.org, site dédié à la création de SE/OS;
- ▶ Suse Studio, pour créer une distribution LINUX (bien d'autres solutions existent sur des bases différentes, se fondant sur DEBIAN, UBUNTU ou autres, GENTOO et ARCHLINUX).

3 Interface humain-machine

Comment les humains interagissent-ils avec les machines (ordinateur, smartphone, etc.)? Comment concevoir des systèmes à la fois efficaces, efficaces et satisfaisants pour leurs utilisateurs?

Voici donc les questions que posent ce domaine particulier de recherche qu'est l'IHM — [Interface homme-machine](#), Interaction humain-machine —, qui s'intéresse à la conception de systèmes de relation, comme son nom l'indique, entre humains et machines.

De fait, l'IHM est une spécialité partie prenante de l'informatique, mais elle intègre également des disciplines comme, entre autres, les sciences cognitives, l'ergonomie, le design et l'électronique.

Les champs d'investigation comportent principalement deux axes :

1. le premier est de concevoir, d'observer, d'analyser et d'évaluer l'interaction et l'expérience des humains avec les technologies;
2. le second est plus technique : il s'agit d'envisager et d'élaborer les environnements existants et futurs, tout en appréciant leur évolution en fonction de l'interaction qu'ils ont avec les humains.

3.1 Interactivité

Interagir avec des technologies apporte des besoins nouveaux, lesquels stimulent à leur tour de nouvelles mutations. À titre d'illustration, la majorité des gens possède aujourd'hui un téléphone intelligent connecté à l'Internet. Les plus ou moins jeunes ont peut-être du mal à se départir des réseaux sociaux; cela s'avère quelque chose que la génération précédente n'aurait même pas pu imaginer.

Cette utilisation induit de nouvelles évolutions technologiques, qui conduisent alors à des solutions encore plus portables, plus petites, comme des bracelets électroniques ou des montres intelligentes, voire des puces électroniques greffées au corps.

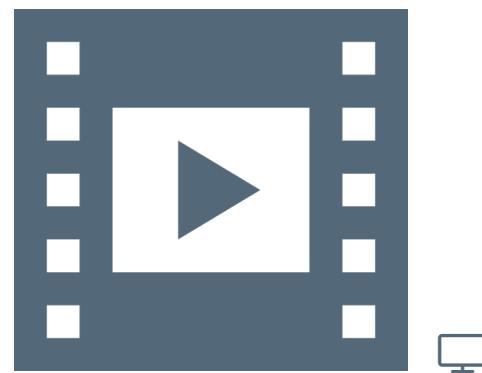
Néanmoins, au-delà des questions de philosophie et d'éthique posées, cela rend bien d'autres services et présente des intérêts utiles en termes de thématiques de recherche, comme l'accessibilité des terminaux aux personnes en situation de handicap ou l'interaction avec un ordinateur qui ne possède plus du tout d'écran, ni encore de clavier.

3.1.1 Bref historique

Les interfaces humain-machine apparaissent vraiment dans les années 1980, car l'informatique est auparavant réservée aux profession-

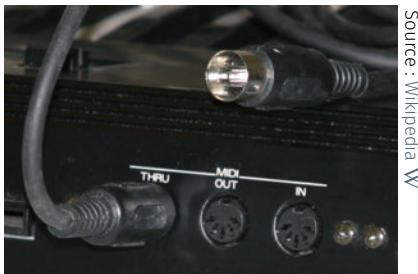
À PROPOS DE L'INTERVENANTE

Anke BROCK est chargée de recherche de l'équipe-projet POTIOC au centre de recherche INRIA BORDEAUX-SUD-OUEST. Née en Allemagne, elle a travaillé cinq ans dans un département de recherche et développement dans l'industrie automobile avant de rejoindre la France. À l'INRIA, elle œuvre dans le domaine de l'interaction Homme-Machine et se consacre désormais au développement de son propre projet de recherche : l'interaction avec les cartes géographiques.



VIDÉO 1.4 — Interface Humain-Machine I.

20. Notamment l'ATARI ST lancé en 1985.



Triplet classique des ports MIDI IN, OUT et THRU en connectique DIN.

21. À condition toutefois d'y consacrer un budget relativement conséquent pour le quidam moyen. La calculatrice programmable fut d'abord l'apanage des geeks en herbe des cours d'école au tournant des décennies 1970-1980, TEXAS INSTRUMENT se taillant alors la part belle.

22. Nouveau champ de recherche des mathématiques : l'analyse numérique.

23. Au-delà des méthodes de synthèse sonore purement mathématiques des années 1960, HILLER et RUIZ, dès 1970, utilisent l'ordinateur pour simuler la vibration d'une corde [RUIZ, 1970] — Pierre M. RUIZ, 1970 — "A technique for simulating the vibrations of strings with a digital computer", Master's thesis. Urbana, Illinois. [HILLER and RUIZ, 1971a] — Le-jaren A. HILLER and Pierre M. RUIZ, 1971a — "Synthesizing musical sounds by solving the wave equation for vibrating objects", *Journal of the Audio Engineering Society*, vol. 19, n°6/7, pp. 462–470 (Part I) and 542–551 (Part II).

24. Portail CNRTL, *ubiquitaire* [adj.] : « Qui manifeste la faculté qu'une personne [un objet] possède de se trouver en différents endroits à la fois ».

25. Méthodes itératives (cf. infra) — entre l'utilisateur et le programmeur —, de développement des logiciels avec, aujourd'hui, les méthodes dites « agiles ».

nels. C'est effectivement avec les ordinateurs personnels, alors orientés sur la bureautique, les jeux interactifs et la MAO²⁰ — Musique assistée par ordinateur —, qu'émergent les premières interfaces graphiques « tout public » et normalisations de format de données comme le MIDI — *Musical Instrument Digital Interface* (1983).

À cette époque, « tout²¹ le monde » devient donc un utilisateur potentiel de l'ordinateur. Cela conduit à révéler les problèmes et limites d'utilisabilité de ces machines à grande échelle. En parallèle et dans le même temps, se développent des disciplines comme les sciences cognitives, fondées sur la psychologie cognitive, l'intelligence artificielle ou encore la linguistique et la philosophie.

Nombre de domaines ont été impactés par l'avènement des ordinateurs. On a ainsi assisté aux premiers efforts de simulation numérique et de concepts mathématiques²² sous-jacents. Mais aussi et, posant derechef la question d'une interface adaptée, les premières tentatives de synthétiseurs musicaux²³ se sont faits jour. Avant de se généraliser à d'autres instruments électroniques, l'interface de prédilection fut d'abord un clavier... Cette fois musical, clone de celui d'un piano.

Avec la complexité croissante des logiciels et la nécessité de tenir compte du facteur humain dès la conception même d'un système d'information, les industries comme l'avionique et l'automobile — et plus largement la fabrication assistée par ordinateur —, mais encore des applications de la société civile comme l'accessibilité aux personnes âgées ou bien en situation de handicap, vont intégrer progressivement nombre de concepts d'interaction humain-machine comme la visualisation des données et des résultats, les systèmes collaboratifs directs ou via Internet (décennie 1990) et, plus récemment, les systèmes ubiquitaires²⁴ ou encore les procédures²⁵ de conception.

3.1.2 Concept d'utilisabilité

L'*utilisabilité* est un concept fondamental du champ sémantique de l'interaction homme-machine. Au début, on signifiait vaguement qu'un système soit utilisable quand il était facile à apprendre et à manipuler. Par la suite, ce concept a été élargi.

Plusieurs normes ISO définissent l'*utilisabilité*, dont par exemple la norme ISO 9241-2010, qui stipule que l'*utilisabilité* est « *le degré selon lequel un produit peut être utilisé par des utilisateurs identifiés pour atteindre des buts définis avec efficacité, efficience et satisfaction dans un contexte d'utilisation spécifié* ». Cette définition est assez complexe. Elle est décomposée pour appréhender séparément les concepts :

Efficacité — L'*efficacité* est la précision et le degré d'achèvement selon lesquels l'utilisateur atteint des objectifs spécifiques. Autrement dit, un système est efficace s'il permet à un utilisateur d'effectuer les tâches qu'il a envie ou besoin de faire.

Efficience — L'*efficience* est le rapport entre les ressources dépensées et la précision ou le degré d'achèvement selon lesquels l'utilisateur atteint les objectifs attendus. Plus précisément, un système rapide à manipuler et ne demandant pas beaucoup d'interactions va être plus efficace que celui où l'utilisateur a besoin de passer beaucoup de temps pour atteindre ces objectifs.

Satisfaction — La *satisfaction* caractérise les attitudes positives du maniement d'un produit; l'utilisateur prend-il plaisir à l'employer ?

Ainsi, pour qu'un produit soit utilisable, il faut à la fois qu'il soit efficace, efficient et satisfaisant. Cette assertion est fondamentale car, même un algorithme ultra performant ne sera pas adopté si son interface est conçue d'une manière qui ne respecte pas ces critères.

La norme ISO indique en outre que l'utilisabilité est définie en fonction des *utilisateurs* et des *contextes*. Pour concevoir des systèmes informatiques, il semble logique de les définir selon leur *cible*, à savoir les *utilisateurs finaux*. Cela peut concerner des attributs d'âge — enfants ou anciens —, devoir répondre à des besoins particuliers — malvoyants ou handicaps divers — ou bien encore, tenir compte des connaissances et compétences antérieures — néophytes versus aguerris.

On ne conçoit pas un système informatique de la même manière pour le grand public que pour des professionnels et des experts.

SYSTÈME D'AIDE À LA NAVIGATION AÉRIENNE — Un pilote d'avion ou encore un contrôleur aérien a besoin d'un système d'information qui ne fasse pas d'erreur : il doit être absolument fiable. Les critères d'efficacité et d'efficiency vont donc être essentiels. En revanche, la satisfaction est moins importante. Que le pilote prenne plaisir à utiliser le système est accessoire au regard de la sûreté du pilotage.

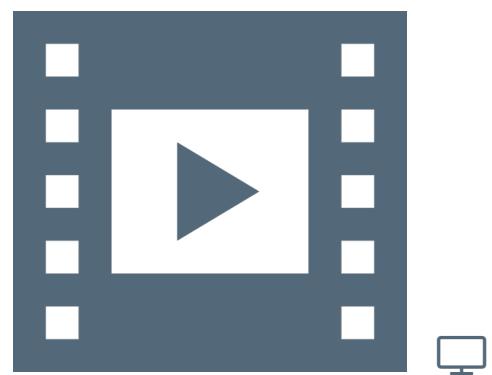
SYSTÈME INTERACTIF DE VISITE DE MUSÉE — L'objectif de la visite d'un musée est que l'utilisateur ait sa curiosité aiguisée, qu'il reste dans l'exposition et prenne le temps d'expérimenter les animations. Dans ce cas de figure, c'est la satisfaction qui est primordiale.

3.1.3 Procédure de conception d'un système utilisable

Pour créer des systèmes qui soient utilisables, trois approches de conception sont à distinguer :

- la *conception centrée utilisateur* prend en compte les besoins des utilisateurs sans pour autant les impliquer directement à la démarche. Des recommandations existent, par exemple certains critères ergonomiques sur lesquels il est possible de se fonder pour élaborer le système. Ces mêmes caractéristiques servent également à évaluer le système après conception, pour vérifier que le résultat obtenu répond bien au cahier des charges initial;
- la *conception participative* va plus loin en intégrant les utilisateurs au processus selon un déroulement en quatre étapes.
 - a. tout d'abord une *phase d'analyse* où les usagers sont observés dans les tâches effectuées. Des questionnaires ou des entretiens avec les usagers peuvent être proposés;
 - b. une deuxième phase concerne la *création de l'idée* où là aussi les utilisateurs sont impliqués. Par exemple, l'organisation de *brainstormings* les fait participer pour trouver des solutions mieux adaptées²⁶ à leurs besoins;
 - c. la suite est associée à une *phase de prototypage* du système d'information. Ce peut être des prototypes haute-fidélité vraiment codés (en « dur », c'est-à-dire opérationnels), comme basse-fidélité, par exemple juste des croquis sur papier qui permettent de représenter l'interaction avec le système;
 - d. vient ensuite une dernière *phase d'évaluation*, où on procède à des tests utilisateurs qui sont confrontés à ceux réalisés par les « vrais » utilisateurs finaux du système.

Ce processus est *itératif*. Après avoir effectué une évaluation, on peut recommencer un nouveau cycle en repartant d'une phase d'analyse afin d'apprecier si le premier prototype conçu répond vraiment — autrement dit avec suffisamment de précision —, aux besoins des utilisateurs. L'équipe de conception doit être mixte et pluridisciplinaire au sens où elle ne doit pas uniquement être composée de développeurs logiciels, mais par exemple faire intervenir des ergonomes ou des spécialistes en facteurs humains ;



VIDÉO 1.5 — Interface Humain-Machine II.

26. Appel aux connaissances et aux savoir-faire « métiers ».

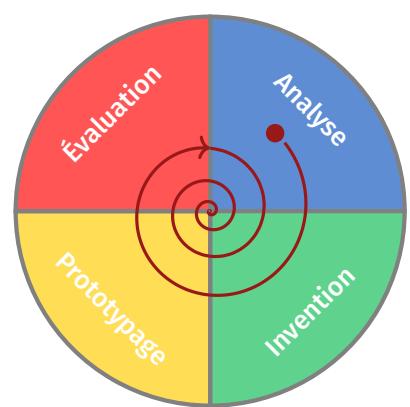


FIGURE 1.11 — Principe itératif associé à la conception participative.



Wikimedia commons © CC BY-SA

■ Prototype initial en bois de la souris inventée en 1963 par Douglas ENGELBART, présentée au public en 1968 et améliorée en 1979 par Jean-Daniel Nicoud.

²⁷ Le terme de « réalité virtuelle », bien que communément admis et usité, peut paraître antinomique (voir [discussion W](#)). En toute rigueur, il semble plus exact de faire mention de « prototypage virtuel » ou de « simulation numérique ». Cependant, ce vocabulaire est issu du monde scientifique et l'appellation VR – Virtual Reality – est plutôt dévolue aux applications multimédia : casques, jeux, etc.



■ Guitare acoustique « augmentée », aux effets paramétrables et autoamplifiée au moyen d'actuateurs situés sous la table d'harmonie au niveau du chevalet, lieu d'accroche des cordes et du transfert d'énergie vibratoire. Le comportement de la caisse d'une guitare est similaire à celui d'un haut-parleur, effet « bass-reflex » compris, dû à la rosace.

— enfin, avec la *co-conception*, les usagers sont toujours plus partie prenante du développement, ils deviennent eux-mêmes les concepteurs des systèmes. C'est actuellement envisageable en particulier avec le mouvement des FABLABS ; ils permettent à tout un chacun d'accéder à des imprimantes 3D, des découpeuses laser ou des dispositifs électroniques tels l'ARDUINO ou le RASPBERRY PI et ainsi autoriser la mise en œuvre de son propre système informatique, entièrement dédié à ses attentes.

3.1.4 Liens entre IHM et technologies numériques

Les dispositifs informatiques employés dans la vie de tous les jours sont issus des laboratoires... Parfois après une certaine latence !

À titre d'exemple, la souris d'ordinateur est introduite par Doug ENGELBART dans les années 1960 aux États-Unis, puis d'abord expérimentée dans les laboratoires de recherche dans les années 1970 avant de se généraliser dans la décennie 1980, notamment avec l'avènement des ordinateurs personnels.

Un cheminement identique est celui des écrans tactiles des smartphones et tablettes d'aujourd'hui. Ils sont aussi étudiés dans les laboratoires de recherche depuis les années 1960, alors que les premières diffusions d'exemplaires commerciaux sont réalisées à partir de 2007.

Les systèmes d'information dépassent désormais le cadre établi des ordinateurs *stricto sensu*. Les systèmes ubiquitaires — à savoir, omniprésents ou itinérants, au sens de la géolocalisation permanente —, que sont les téléphones mobiles et autres systèmes embarqués, s'invitent toujours plus dans notre quotidien : habitat, véhicule, sport, voire vêtements et peut-être à l'avenir de manière généralisée, certaines parties du corps biologique.

On peut encore observer et mentionner l'existence et la commercialisation de systèmes dits de *réalité virtuelle*²⁷ — « expérience sensorielle dans un environnement artificiel généré par des logiciels » (source W), par exemple les simulateurs de vol et de conduite automobile en contexte d'apprentissage — et de *réalité augmentée* — « superposition de la perception humaine et d'éléments (sons, images, vidéos, etc.) calculés par un système informatique en temps réel » (source W), comme certaines applications d'aide à l'orientation et au déplacement.

Le principe de la réalité augmentée est donc de combiner des informations issues de la perception du réel (vision, ouïe, toucher) avec celles de données complémentaires. À titre d'illustration, des lunettes « augmentées » permettent d'observer l'environnement, auquel on adjoint un certain nombre de paramètres supplémentaires : changement de colorimétrie ou rajout d'objets dans le champ visuel, etc.

Dans un autre domaine, celui des instruments de musique, la réalité augmentée ne s'attache pas à en donner une version uniquement numérique apportée par les instruments électroniques — à associer au vocable de « réalité virtuelle » —, mais bien de combiner le rendu acoustique naturel avec des traitements additionnels sur les signaux fournis par des capteurs et restitués par des actuateurs (anglicisme issu de « actuator ») intégrés à l'instrument acoustique.

3.1.5 Avenir de l'interface humain-machine

À quoi peut-on s'attendre à l'avenir concernant les interfaces entre humain et machine ? Il s'agit certainement et avant tout de permettre l'accès aux informations à un maximum de personnes. Cela passe inévitablement par une adaptation des interfaces à tous les publics : personnes âgées, en situation de handicap — visuel comme moteur —, etc.

En effet, la population des sociétés occidentales vieillissant et l'âge de vie s'accroissant, un des enjeux est de concevoir et de proposer des solutions au regard de problèmes cognitifs : rappel de prise de médicaments, détection des chutes, télémédecine pour les résidents périphériques et ruraux des métropoles, etc.

On peut également évoquer les populations des pays émergents, auxquelles des environnements informatiques sont envisageables pour l'accès aux informations et la participation à leur désenclavement.

Les développements en cours au sein des laboratoires sont multiples, desquels on peut citer les interfaces entre cerveau et ordinateur. Elles permettent une interaction avec un ordinateur exclusivement au moyen d'une mesure de l'activité cérébrale.

Par ailleurs, on peut faire état des interfaces humain-machine relatives aux domaines en pleine expansion comme le traitement massif de données (*Big Data*), l'intelligence artificielle ou encore la robotique.

GLOSSAIRE CONTEXTUEL

INTERFACE HOMME-MACHINE — Ensemble de moyens mis en œuvre pour qu'un humain puisse interagir avec une machine. Par extension, l'IHM est aussi le domaine de recherche qui conçoit des systèmes informatiques et étudie les interactions entre êtres humains et machines.

SCIENCES COGNITIVES — Sous un vocable commun, les *sciences cognitives* regroupent l'ensemble des disciplines qui étudient les mécanismes de la connaissance.

ERGONOMIE — Discipline qui étudie les interactions existantes entre les êtres humains et les composantes d'un système. On y prend en compte les facteurs humains pour concevoir et évaluer des tâches afin de les rendre compatibles avec les besoins et les capacités des utilisateurs.

DESIGN — Le *design* recouvre les notions de création et de conception d'un produit ou d'un système.

PSYCHOLOGIE COGNITIVE — Branche de la psychologie qui étudie spécifiquement le fonctionnement de l'acquisition des connaissances chez l'être humain : langage, mémoire, perception, concentration, etc. La cognition — du latin *cognito* pour action de connaître —, est ainsi l'ensemble des activités mentales et mécanismes qui se rapportent à la connaissance.

INTELLIGENCE ARTIFICIELLE — L'*intelligence artificielle* — IA — est, selon l'*Encyclopédie Larousse*, « l'ensemble des théories et des techniques mises en œuvre en perspective de réaliser des machines capables de simuler l'intelligence ». Cela recouvre un ensemble de concepts et de technologies plus qu'une discipline autonome constituée. Au regard d'une définition peu précise, certains, à l'instar de la *CNIL*, introduisent ce sujet comme « le grand mythe de notre temps » (source [W](#)).

LINGUISTIQUE — Discipline qui étudie le langage et le fonctionnement des langues par une approche descriptive.

ACCESSIBILITÉ — Concept initialement issu du monde du handicap et désormais étendu à l'ensemble des citoyens pour signifier la facilitation de l'accès dans différents domaines. En informatique, on parle d'accessibilité pour désigner l'adaptation des systèmes numériques et le développement d'outils spécifiques dans le cas de handicaps, comme par exemple la malvoyance.

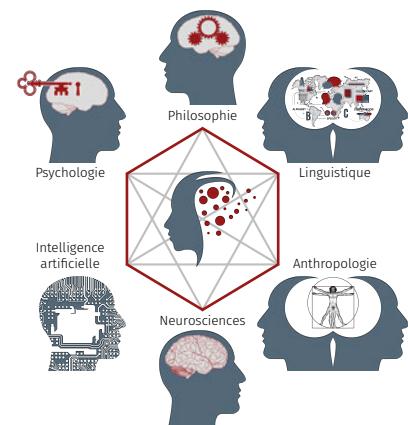


FIGURE 1.12 — Champs disciplinaires des sciences cognitives.

SYSTÈME UBIQUITAIRE — L'ubiquité ou l'omniprésence, est la capacité de se trouver en tout lieu ou à plusieurs endroits en même temps. En informatique, cela renvoie aux environnements dans lesquels ordinateurs et réseaux sont intégrés au monde réel et où l'utilisateur est connecté en permanence (cf. [W](#)).

UTILISABILITÉ — Conformément à la norme ISO 9241-11, il s'agit du « degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». L'utilisabilité possède plusieurs critères : l'*efficacité*, l'*efficience* et la *satisfaction*.

NORME ISO 9241-2010 — Cette norme ISO est directement liée aux IHM. « L'ISO 9241-210:2010 fournit des exigences et des recommandations relatives aux principes et aux activités de conception centrée sur l'opérateur humain, intervenant tout au long du cycle de vie des systèmes informatiques interactifs. Elle est destinée à être utilisée par les responsables de la gestion des processus de conception, et traite des manières dont les composants matériels et les logiciels des systèmes interactifs permettent d'améliorer l'interaction homme-système ».

EFFICACITÉ — Concept lié à l'atteinte du résultat prévu.

EFFICIENCE — Terme signifiant l'atteinte d'un résultat avec le moindre effort ou le temps minimal.

SATISFACTION — Évaluation subjective par l'utilisateur de l'interaction avec un système.

CONCEPTION CENTRÉE UTILISATEUR — Démarche de conception où les besoins des utilisateurs sont pris en compte à chaque étape.

CONCEPTION ITÉRATIVE — Approche de conception qui se réalise par bouclage de cycles répétés. Chaque cycle permet de tester, d'adapter et d'améliorer le produit.

CONCEPTION PARTICIPATIVE — Démarche de conception qui fait participer les utilisateurs. Il s'agit d'une démarche centrée utilisateur où est mis en avant le rôle actif de l'utilisateur.

Co-CONCEPTION — Approche également appelée *co-design*. Il s'agit de développer un système en collaboration avec l'utilisateur final, au-delà d'une simple validation d'étape de sa part.

RÉALITÉ VIRTUELLE — L'expression « réalité virtuelle » — multimédia immersif ou encore simulation — renvoie à une technologie numérique qui reproduit la présence d'un utilisateur en environnement artificiel généré au moyen de logiciels. Il s'agit donc d'un contexte dans lequel l'usager interagit pour vivre une expérience sensorielle qui peut inclure la vue, le toucher, l'ouïe et l'odorat : visuelle, haptique, sonore ou olfactive (cf. [W](#)).

RÉALITÉ AUGMENTÉE — La réalité augmentée se définit comme la superposition de l'observable physique avec différents éléments (sons, images, traitements de l'information, etc.) calculés par un système informatique en temps réel. Cela désigne souvent les méthodes qui incrustent de manière réaliste des objets virtuels dans une séquence d'images. Néanmoins, cela s'applique aussi bien à toutes les perceptions proprioceptives : visuelles, comme tactiles ou auditives — (voir [W](#)).

NOTE DE LA RÉDACTION

Texte de Michel BEAUDOUIN-LAFON, publié sur [Interstices](#) — revue en ligne de culture scientifique du numérique — le 1^{er} juillet 2016.

3.2 Cinquante ans d'IHM : retour vers le futur

Depuis qu'existent les ordinateurs, la question de l'interface avec les utilisateurs s'est posée. En cinquante ans [1960-2010], l'interaction entre homme et machine (IHM) a rendu l'informatique accessible au

plus grand nombre, d'une manière que personne n'avait anticipée. Mais, ne sommes-nous pas devenus prisonniers d'interfaces qui ont peu évolué depuis plusieurs décennies ?

L'interaction avec les ordinateurs s'avère aussi vieille que les ordinateurs eux-mêmes. Un ordinateur est une machine programmable, il faut donc pouvoir y entrer les données et les programmes puis en visualiser les résultats. Les premiers ordinateurs disposaient de lecteurs de cartes perforées et d'imprimantes. Toutefois, ces dispositifs ne permettaient pas une réelle interaction pendant l'exécution du programme.

Ainsi, l'histoire de l'IHM débute véritablement au début des années 1960 avec les travaux pionniers de Ivan SUTHERLAND sur SKETCHPAD ; ils ont montré comment un opérateur pouvait interagir en temps réel avec une machine exécutant un logiciel complexe.

Ce texte présente certains repères de l'épopée de l'IHM sans, bien entendu, prétendre²⁸ à l'exhaustivité. L'objectif est de mettre en regard les travaux pionniers avec les systèmes interactifs commerciaux actuels et d'attirer l'attention sur le décalage existant entre l'état de l'art et les standards du marché, entre les inventions et leur large diffusion.

A. IVAN SUTHERLAND : DE SKETCHPAD À LA RÉALITÉ VIRTUELLE

SKETCHPAD, dû à Ivan SUTHERLAND début des années 1960 et publié dans sa thèse de doctorat en 1963, est considéré comme la première interface graphique. Développé au MIT *Lincoln Laboratory*, c'est le premier système à utiliser un écran cathodique et un crayon optique pour l'édition graphique de dessins techniques.

Ce n'est que bien plus tard, en 1983, que Ben SHNEIDERMAN appelle « manipulation directe » ce type d'interaction avec des objets représentés à l'écran, par contraste avec l'utilisation systématique, jusqu'au début des années 1980, de langages de commandes obligeant à mémoriser leurs noms — forme d'interaction toujours pratiquée²⁹ dans ce que l'on appelle le « terminal ».

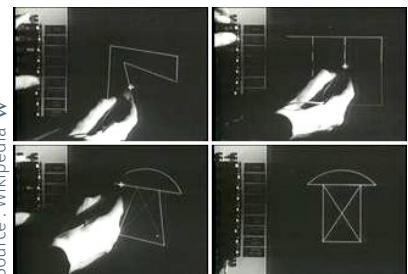
SKETCHPAD permet de créer interactivement des diagrammes et des plans. Il met en œuvre de nombreux concepts fondamentaux des interfaces graphiques modernes : désignation directe des objets à l'écran, retour d'information immédiat sous forme de lignes élastiques, zoom avant et arrière du dessin avec un facteur de 2 000 et ainsi de suite. Il inclut même des fonctions dont peu de logiciels modernes disposent. Par exemple, l'utilisateur peut spécifier des contraintes, comme le fait que des segments doivent être parallèles ou à angle droit, après la création du dessin. On voit alors l'objet s'animer pour satisfaire au mieux ces contraintes. Même au niveau de la mise en œuvre, les concepts sont étonnamment modernes : représentation des objets graphiques en mémoire, résolution de contraintes, système de rendu graphique.

SUTHERLAND développe SKETCHPAD sur le TX-2, un des rares ordinateurs de l'époque utilisable en ligne : jusqu'à la fin des années 1970, la grande majorité des ordinateurs sont utilisés de façon non interactive, en traitement par lots (« batch »).

Le TX-2 a 320 Ko de mémoire, deux fois plus que les plus gros ordinateurs commerciaux de l'époque, une unité de bande magnétique, la première imprimante de XEROX et l'entrée des programmes se fait par ruban perforé. Surtout, le TX-2 a un écran cathodique — en fait un oscilloscope — de 9 pouces (21 cm), un crayon optique et un panneau de boutons que SUTHERLAND utilise pour sélectionner des fonctions, comme la palette d'outils des interfaces modernes.

D'autres chercheurs utilisent le TX-2 à la même époque pour réaliser d'autres interfaces révolutionnaires, comme Ron BAECKER qui crée GENESYS, le premier système d'animation de l'Histoire.

²⁸. Pour une liste plus complète, voir par exemple un article de B. MYERS paru en 1992, ainsi que les sites Web mentionnés en annexe de ce texte.



Source : Wikipedia W

Écran de SKETCHPAD avec à gauche, la rangée de boutons permettant de choisir la commande et au centre l'opérateur utilisant le stylo optique pour créer des schémas. L'image en bas à droite résulte de l'application de contraintes d'orthogonalité entre les segments sur l'image en bas à gauche.

²⁹. Cet exercice est avant tout adopté sous les systèmes UNIX, notamment LINUX où, une fois les commandes clavier mémorisées, l'usage s'avère bien plus efficient que l'appel aux menus optionnels des interfaces graphiques.

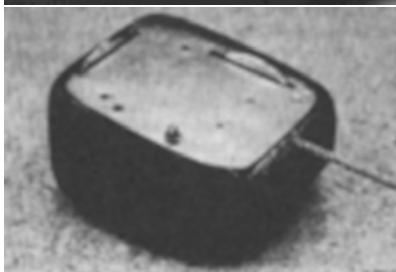


Source : Ohio State University

Premier casque de réalité virtuelle, réalisé par SUTHERLAND et SPROULL (Harvard, 1967).



Source : Engelbart 1968



Souris inventée par Doug ENGELBART et présentée au public en 1968, avec deux molettes orthogonales.



Source : Engelbart 1968

Clavier du système NLS, avec le « chord keyboard » sur sa partie gauche et la souris sur sa partie droite.

Peu après, Ivan SUTHERLAND devient un des pionniers de l'infographie et de la réalité virtuelle, avec notamment un algorithme d'élimination des parties cachées qui porte son nom. En 1967, alors professeur à Harvard, il imagine avec son étudiant Bob SPROULL le premier casque de réalité virtuelle affichant des images de synthèse. Par la suite, il s'intéresse à la robotique et crée l'entreprise EVANS & SUTHERLAND, célèbre dans les années 1980 pour ses systèmes graphiques haut de gamme.

B. DOUG ENGELBART : NLS/AUGMENT

Pendant qu'Ivan SUTHERLAND, sur la côte est des États-Unis, travaille sur SKETCHPAD, sur la côte ouest, Doug ENGELBART s'attache à ce qui deviendra NLS/Augment. ENGELBART est un visionnaire qui a anticipé une grande partie de l'évolution de l'informatique depuis les années soixante. Revenu de la guerre convaincu que seule la combinaison de la puissance de calcul des machines et de l'intelligence humaine peut résoudre les problèmes du monde, il publie en 1962 un article fondateur intitulé « *Augmenting Human Intellect : A Conceptual Framework* ». Il y présente sa vision du rôle des systèmes informatiques dans ce qu'il appelle l'augmentation de l'intellect humain, grâce à leur puissance de calcul mais aussi aux possibilités de collaboration qu'ils offrent.

D'une certaine façon, cet article fait écho à un autre article séminal, « *As We May Think* » de Vannevar BUSH, qui présentait en 1945 un système imaginaire appelé MEMEX, considéré aujourd'hui comme l'ancêtre de l'hypertexte. Lorsque Vannevar BUSH écrit son article, l'ordinateur existe à peine. Quinze ans plus tard, l'informatique s'est développée et ENGELBART peut commencer à mettre en œuvre sa vision, sur laquelle il travaillera le reste de sa vie, notamment au sein du BOOTSTRAP INSTITUTE qu'il crée en 1989 lorsque McDonnell DOUGLAS arrête le financement de son projet (pour un historique détaillé, voir l'article d'Interstices « Doug ENGELBART, inventeur et visionnaire »).

En 1964, Doug ENGELBART invente la souris, car il veut pouvoir facilement désigner des objets à l'écran ; il en brevete ce dispositif. Le brevet ne lui rapportera jamais rien, car les souris commercialisées plus tard utilisent une boule au lieu des deux roues de son système. Il crée également des claviers à accord (« *chord keyboards* ») qui permettent d'entrer des données en composant des accords avec les doigts d'une main, comme sur le clavier d'un piano. Peu de gens ont été capables de maîtriser ces claviers, qui sont cependant encore utilisés par les greffiers dans les tribunaux américains pour la saisie de texte. La souris, en revanche, s'est imposée comme le périphérique incontournable des interfaces graphiques.

Fin 1968, ENGELBART fait une démonstration publique devant mille personnes de son système NLS (oN-Line System), développé depuis plusieurs années au SRI, à Stanford. La démonstration est filmée et toujours disponible sur le site Web de Stanford.

NLS est un système hypertexte collaboratif couplé à un système de vidéoconférence. Des utilisateurs séparés de 45 km éditent collaborativement des données organisées hiérarchiquement, comme le plan d'un document découpé en chapitres, sections et sous-sections. Lorsqu'ils collaborent, ils peuvent se voir par vidéoconférence et utiliser des télé-pointeurs pour montrer des objets à l'écran.

L'interaction avec NLS est complexe, notamment à cause de l'utilisation du « *chord keyboard* ». Mais Doug ENGELBART a toujours été perplexe devant l'idée de systèmes conviviaux ou faciles d'utilisation. Pour lui, l'important est que le système permette aux utilisateurs de développer leurs compétences et de construire des organisations humaines plus évoluées. Il justifie cette approche avec l'exemple suivant : il est

plus facile de faire du tricycle, mais le vélo est plus rapide car on peut se pencher dans les virages; le temps passé à apprendre le vélo est donc largement rentabilisé. ENGELBART défend ainsi l'idée d'interfaces adaptées aux capacités des utilisateurs experts, quitte à nécessiter un apprentissage, plutôt que des interfaces simplistes devant être accessibles à tout le monde.

Certains aspects de *NLS* — renommé *Augment* lorsque ENGELBART quitte le SRI en 1978 — ont mis des décennies à s'imposer. Ainsi, ce n'est que tout récemment que la vidéoconférence et l'édition collaborative, par exemple avec *SKYPE* et *GOOGLE Docs*, sont devenues largement accessibles. L'explication en est peut-être que, avec le *XEROX Star* qui se profile et plus tard le *MACINTOSH*, l'informatique s'intéresse à des catégories d'utilisateurs différentes de celles que visait ENGELBART : les « *knowledge workers* » (travailleurs intellectuels) pour *NLS*, les secrétaires pour le *Star*, le grand public pour le *MACINTOSH*. L'avènement dans les années 1980 de l'informatique dite individuelle est d'ailleurs bien la preuve que la vision d'ENGELBART de systèmes spécifiques à la collaboration est restée très longtemps ignorée. Il aura fallu attendre l'explosion du Web 2.0, des réseaux sociaux, des services comme *SKYPE* et *GOOGLE Docs* pour que les outils informatiques commencent à fournir de réelles capacités de collaboration.

C. XEROX : LE STAR

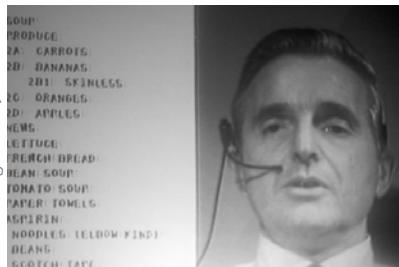
En 1970, XEROX crée son laboratoire de recherche à Palo Alto, le *PARC*. XEROX veut non seulement développer sa technologie de la photocopie, mais aussi se lancer sur le marché des systèmes bureautiques. Le *slogan* favori des chercheurs du *PARC* est dû à l'un de ses fondateurs, Alan KAY : « la meilleure façon de prédire le futur, c'est de l'inventer ». De fait, XEROX PARC est le théâtre d'un nombre spectaculaire d'inventions qui ont marqué l'informatique. Ainsi, au moment du lancement du projet *Star* en 1975, XEROX a déjà inventé l'imprimante à laser et le réseau local *Ethernet* et développe le langage objets *SMALLTALK*.

Dès 1968, Alan KAY, considéré comme le père de l'informatique individuelle, a sa propre vision de l'ordinateur, qu'il appelle le *DYNABOOK*. Elle consiste à fournir aux utilisateurs, non pas des applications pré-programmées, mais un ensemble d'outils pour construire chacun son propre environnement. Pour mettre en œuvre sa vision, Alan KAY développe dans les années 1970 le langage *SMALLTALK* et son environnement de programmation graphique, premier du genre. Il poursuit encore aujourd'hui ses travaux au sein du *Viewpoints Research Institute*.

La première station de travail personnelle munie d'un écran graphique développée à XEROX PARC est l'*Alto*. Elle sert de base à de nombreuses applications qui permettent d'affiner les principes de l'interaction graphique : édition de texte et de dessins (images ou formes graphiques), courrier électronique, outils bureautiques. C'est dans ce contexte qu'est lancé le projet *Star* en 1975, qui débouche en 1981 sur l'annonce du « *Xerox 8010 Information System* », nom commercial du *Star*. Cette machine est conçue pour les secrétaires de direction, cible logique pour XEROX qui s'affiche comme « *The Document Company* » et cherche à trouver des relais de croissance dans la perspective de l'échéance de ses brevets sur la photocopie. Même si sa commercialisation est un échec, le *Star* révolutionne l'informatique en préfigurant l'avènement des ordinateurs personnels et des interfaces graphiques.

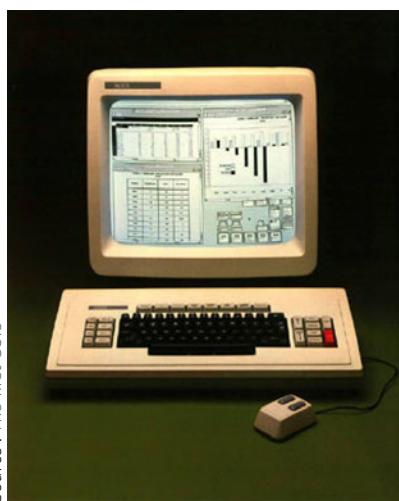
Les aspects matériels du *Star* sont conçus en fonction des besoins identifiés pour le logiciel. Un tel matériel est constitué d'un processeur micro-codé d'une puissance inférieure à un million d'instructions par seconde, muni d'opérations rapides pour accéder à l'écran (*BitBlt*),

Source : Engelbart 1968



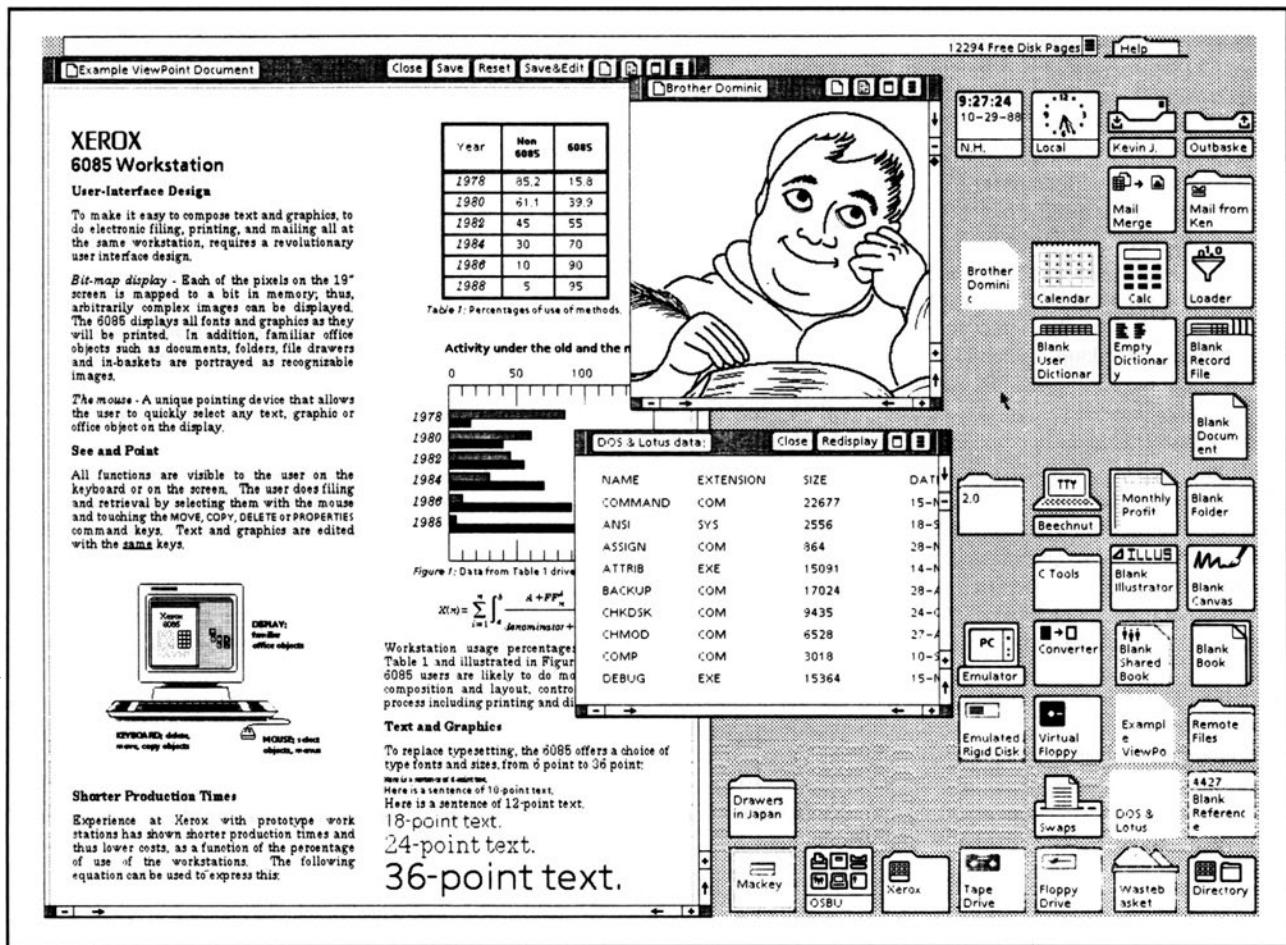
Édition collaborative de texte temps réel et vidéoconférence dans *NLS/Augment*.

Source : The first GUIs



Écran, clavier et souris du Xerox 8010, connu sous le nom de *Star* (1981).

de 385 Ko de mémoire, d'un disque de 10 à 40 Mo, d'un lecteur de disquettes 8 pouces et d'une connexion Ethernet. Les périphériques d'interaction sont un écran noir et blanc de 17 pouces, une souris à deux boutons et un clavier spécial, muni de deux pavés de touches de fonctions, à droite et à gauche de la partie alphabétique. Le logiciel est programmé en MESA, une variante évoluée de PASCAL développée au SRI. Le développement du *Star* représente à l'époque un effort de 30 hommes-années, pour une machine dont la capacité de calcul et de stockage est une fraction de celle de la moindre calculette d'aujourd'hui.



Source : The Xerox Star : A retrospective

Interface graphique du *Star*. À gauche, une fenêtre avec un document mélangeant texte, graphique et tableau. Deux autres fenêtres au milieu et, à droite, un ensemble d'icônes de documents et de ressources accessibles.

Le *Star* est, dès le départ, une machine destinée à être connectée à un réseau local Ethernet. L'interface permet de naviguer de manière totalement transparente parmi les ressources du réseau (imprimantes, serveurs de fichiers, etc.) et de créer son propre environnement en déplaçant les icônes de ces ressources sur le bureau. Le *Star* est la première machine à offrir des fenêtres qui se superposent et à utiliser la métaphore du bureau avec, notamment, des icônes représentant les documents, les dossiers et d'autres ressources. Larry TESLER invente le copier-coller, s'inspirant de la façon dont les maquettistes utilisent des ciseaux et de la colle dans l'édition pour littéralement copier et coller des morceaux de texte. Il met près de quinze ans à imposer cette méthode, qui paraît pourtant aujourd'hui évidente et indispensable.

Mais le plus frappant dans l'interface du *Star* vis-à-vis des systèmes actuels est que le système est centré sur la notion de document : un nouveau document est créé à partir d'un modèle existant et, tout docu-

ment peut contenir du texte, des dessins, des formules mathématiques, des tableaux, tous éditables sur place. Pour l'utilisateur, la notion d'application est inexistante.

L'interface est conçue pour utiliser un nombre minimal de commandes, dont les principales sont accessibles directement par des touches de fonctions du clavier : copier, déplacer, détruire, changer les propriétés. L'interface n'a pas de barre de menus, seulement un ou deux menus déroulants pour les fonctions les moins fréquentes. Il n'utilise pas de boîtes de dialogue dites modales, qui interrompent l'utilisateur, mais des boîtes de propriétés associées à la partie du document en cours d'édition. Grâce à la configuration du clavier, l'interaction consiste à manipuler la souris à la main droite pour désigner les objets d'intérêt et sélectionner les options dans les boîtes de propriétés, puis utiliser les touches de fonctions à gauche du clavier avec la main gauche pour spécifier les actions. En cela, le *Star* reprend le style d'interaction de *NLS/Augment* en le simplifiant.

Tous les concepts des interfaces modernes sont présents au sein du *Star*. À vrai dire, le *Star* est encore en avance par rapport aux interfaces actuelles : transparence du réseau, environnement centré sur les documents, utilisation d'un petit nombre de commandes qui s'appliquent à un grand nombre de contextes, interaction non-modale, autant de caractéristiques du *Star* qui ne sont toujours pas présentes dans les environnements actuels. Pourtant le *Star* est un échec commercial : système trop cher, cible *marketing* mal évaluée et surtout, incapacité de XEROX à sortir de son créneau historique des photocopieurs.

C'est le *MacIntosh* d'APPLE qui, trois ans plus tard, est le réel point de départ du marché de l'informatique personnelle. Certes, le *MacIntosh* s'est largement inspiré du *Star* — on cite fréquemment la visite de Steve JOBS et de son équipe au XEROX PARC en 1979. Mais le concept est, dès le départ, différent et propre à APPLE et une grande partie de l'interface du *MacIntosh* est dérivée du *Lisa*, créée avant la visite de JOBS au PARC. APPLE invente la barre de menus et les boîtes modales, laisse de côté l'aspect réseau et conserve le concept d'application qui était familier aux utilisateurs de l'APPLE.

Quinze ans plus tard, début des années 1990, APPLE tente d'introduire une approche centrée sur les documents avec *OpenDoc*, mais le projet est abandonné. Entre autres, il remet en cause le modèle commercial selon lequel les éditeurs de logiciels vendent des applications autonomes et indépendantes les unes des autres. Pourtant, le modèle de document, que l'on trouve notamment sur le Web, est conceptuellement plus adapté aux usages bureautiques que celui d'application.

D. APPLICATION EMBLÉMATIQUE : LE TABLEUR

Certaines applications interactives ont révolutionné l'usage des ordinateurs. En 1979, Dan BRICKLIN et Bob FRANKSTON commercialisent VISICALC, le premier tableur de l'histoire. BRICKLIN, étudiant à Harvard, en a l'idée en utilisant une calculette TEXAS INSTRUMENTS. Il imagine comment un système permettant la visualisation dite « tête haute » d'une feuille de calcul et piloté par un « trackball », lui faciliterait la résolution de ses exercices d'économie en lui permettant de tester rapidement plusieurs hypothèses.

La visualisation tête haute date des années 1950 et a d'abord été utilisée dans les avions de chasse, pour afficher les informations directement sur la vitre du cockpit plutôt que sur des écrans du tableau de bord, évitant ainsi aux pilotes d'avoir à baisser la tête pour lire les informations. Certaines voitures utilisent aujourd'hui ce système d'affichage. Quant au « trackball », c'est une boule dont seule la partie supérieure



Clavier du Star avec pavés de touches de fonctions disposés de part et d'autre du clavier alphabétique.

Version alpha de VISICALC en 1979. Capture d'écran obtenue sur un APPLE.

émerge de son boîtier et que l'on peut faire tourner sur elle-même. Elle est un peu moins précise qu'une souris, mais offre l'avantage de nécessiter peu de place.

BRICKLIN abandonne la visualisation tête haute pour se rabattre sur l'écran de son APPLE et décide d'utiliser les touches de positionnement du curseur du clavier, car le contrôleur de jeu de l'APPLE n'est pas assez précis pour permettre le pointage direct des cellules.

L'algorithme de calcul des cellules est dérivé de celui de SUSSMAN et STALLMAN du MIT — *Massachusetts Institute of Technology* — pour recalculer instantanément les valeurs de l'ensemble du tableau quand on change le contenu d'une cellule. L'interaction avec un tableur est donc d'une grande simplicité : déplacer le curseur sur une cellule, puis entrer sa valeur ou la formule qui la calcule.

Il n'existe peut-être pas d'autre application ayant eu un impact aussi important et dont le concept n'ait pas changé en trente-cinq ans. Grâce à VISICALC, les comptables peuvent faire en un quart d'heure ce qui leur prenait vingt heures par semaine auparavant. Mais surtout, comme BRICKLIN l'avait bien vu, le tableur devient un outil d'aide à la décision et non pas simplement un outil de calcul : il permet de tester et de comparer rapidement plusieurs hypothèses et d'échanger, non seulement des résultats, mais aussi la méthode de calcul.

Enfin, le tableur est fondé sur un modèle tellement simple et puissant qu'il est largement détourné, approprié de mille manières par ses utilisateurs. Certains l'utilisent pour littéralement dessiner (et simuler) des circuits électroniques, d'autres pour réaliser des œuvres d'art ou encore pour faire de la mise en page. Le tableur a ainsi la propriété, rare pour un programme informatique, de dépasser les usages attendus par ses concepteurs.

E. OCCASION MANQUÉE DE L'INTERACTION : LE WEB

Autre concept ayant eu un impact sur nos usages de l'ordinateur : celui de l'hypertexte. Comme évoqué plus haut, il remonte à l'article visionnaire de Vannevar BUSH en 1945.

Le terme hypertexte lui-même a été inventé en 1968 par Ted NELSON, qui publie en 1981 un ouvrage intitulé « *Literary Machines* » où il présente XANADU, une vision d'un système mondial en réseau pour la publication de documents.

XANADU est fondé sur un procédé de « transclusion³⁰ » : au lieu de copier le texte d'un document lorsqu'on le cite, on inclut une référence au document source, qui garde ainsi trace des citations et met en œuvre un système de micropaiement à l'acte.

NELSON a depuis tenté, sans réel succès, de réaliser XANADU. Par ailleurs, de nombreux systèmes hypertexte ont été développés dans divers laboratoires à partir de cette époque.

En 1980, Tim BERNERS-LEE crée au CERN — Centre européen en recherche nucléaire³¹ —, un système hypertexte qui sera le précurseur du Web. En 1989, il propose alors au CERN un projet de système hypertexte en réseau et réalise en 1990 NEXUS, un prototype qui est à la fois un navigateur — « *browser* » en anglais — et un éditeur de pages Web. Il invente le langage HTML, pour la description des pages et le protocole de communication HTTP entre navigateur et serveur Web. Cependant, le prototype est implanté sur la NEXT, une machine³² peu répandue, ce qui nuit à son déploiement.

En 1993, Marc ANDRESEN implémente le navigateur MOSAIC sous l'environnement³³ X-WINDOW, qui est largement utilisé dans le monde de la recherche, mais le temps lui manque pour y intégrer un éditeur de

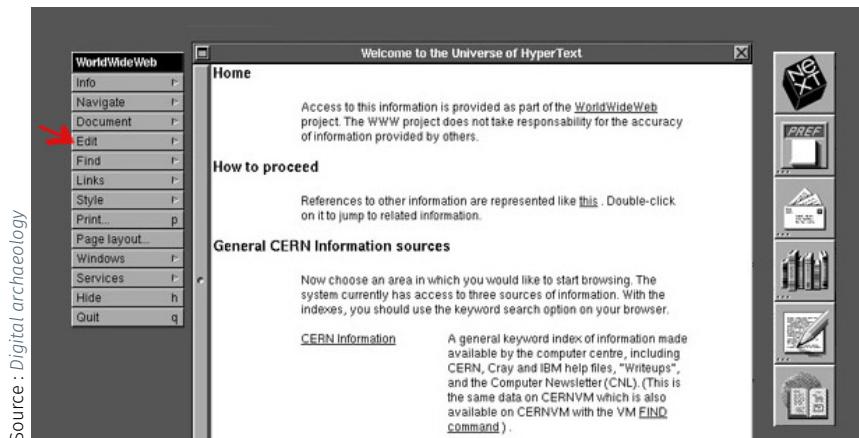
³⁰. Terme inventé par Ted Nelson qui signifie l'inclusion par référence d'un document dans un autre (cf. Wikipedia [W](#)).

³¹. Le CERN, installé sur la frontière franco-suisse dispose d'équipements uniques au monde, en particulier son complexe d'accélérateurs de particule comme le LHC — *Large Hadron Collider*.

³². Station très intéressante — développée par les équipes de Steve JOBS lors de sa période de retrait d'APPLE —, mais qui n'a pas eu le succès commercial mérité.

³³. X-WINDOW est un gestionnaire de fenêtre des stations de travail UNIX des années 1990, notamment des stations SPARC produites par SUN MICROSYSTEMS.

pages Web comme dans la version de BERNERS-LEE. La diffusion de MoSAIC marque le début du développement exponentiel du Web en dehors de son contexte d'origine vers le succès qu'on lui connaît.



Source : Digital archaeology

NEXUS : premier navigateur Web (implémenté sur station NeXT). La commande « Edit » que pointe la flèche rouge affiche le contenu de la page Web en cours.



En termes d'interaction, les navigateurs Web marquent un grand coup d'arrêt, sinon un retour en arrière. Jusqu'à récemment, l'interaction sur le Web était limitée à cliquer des liens et remplir des formulaires; pas beaucoup mieux que ce que permettait notre MINITEL national. Alors que BERNERS-LEE voulait que chacun puisse également être un auteur, l'édition de pages et la construction de sites nécessitent des outils complexes et des connaissances avancées. S'agissant d'un système destiné à partager des documents, le support à la collaboration de groupe, chère à ENGELBART, est pratiquement inexistant.

Certes, l'avènement du Web 2.0 et l'évolution des standards ont permis de rapprocher l'interaction de celle que l'on peut avoir avec des applications classiques. Mais à quel prix! Chaque page doit embarquer du code JAVASCRIPT qui implémente, souvent de façon incomplète ou maladroite, des interactions aussi courantes qu'un menu déroulant ou le glisser-déposer; l'édition de texte est d'une grande pauvreté; la création de pages et de sites est de plus en plus complexe, sauf à utiliser des outils comme les *blogs* qui restreignent dramatiquement la forme des contenus et les capacités d'interaction.

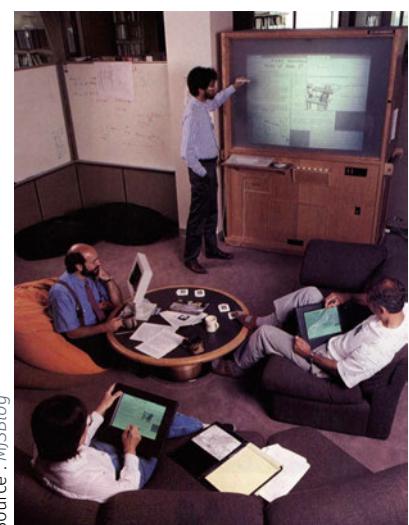
Le Web présentait une opportunité rare d'imaginer de nouvelles interactions collectives pour fabriquer et organiser des contenus riches et variés. Mais finalement, il est resté un système orienté essentiellement vers la diffusion de documents et, c'est en-dehors du Web, que se sont développées les inventions et innovations récentes en IHM.

F. VERS L'INTERACTION PHYSIQUE AVEC LE MONDE NUMÉRIQUE —

En 1991, Mark WEISER publie un article qui présente l'*Ubiquitous Computing* ou *Ubicomp*, sa vision de l'informatique du XXI^e siècle. Avec l'*Ubicomp*, la multiplication des ordinateurs et des écrans de toutes tailles permet l'accès à l'information en tout lieu et toute circonstance.

Dans son laboratoire du XEROX PARC³⁴ de Palo Alto, Mark WEISER développe des prototypes de systèmes *Ubicomp* avec des ordinateurs de trois tailles (*badge*, *bloc-note* et *tableau*) capables de communiquer entre eux pour créer un environnement interactif. Le *badge* permet de localiser son porteur et de mettre à disposition ses documents informatiques sur le *bloc-note* ou le *tableau* le plus proche; ce dernier pouvant être utilisé de manière collaborative.

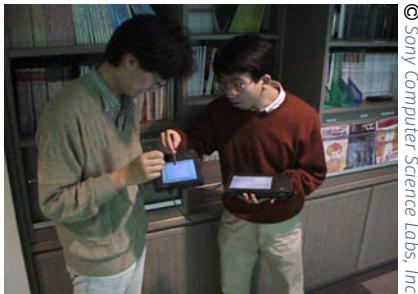
Cette vision préfigure clairement l'avènement des *smartphones*, tablettes tactiles, tableaux interactifs et Internet des objets. Néanmoins, elle reste loin d'être réalisée à ce jour, car là où WEISER imaginait tous



Source : MJSBlog

Prototype du système Ubicomp proposé par Mark WEISER.

³⁴ Voir l'article de Sacha KRAKOWIAK « XEROX PARC et la naissance de l'informatique contemporaine » publié dans Interstices le 27 avril 2012.



ces appareils fonctionnant de façon harmonieuse dans leur environnement, la situation actuelle est beaucoup plus chaotique.

Ainsi, transférer de l'information d'un appareil à l'autre est particulièrement fastidieux. Pourtant, Jun REKIMOTO a inventé en 1997 à SONY LABS le « *pick-and-drop* », une technique intuitive qui consiste simplement à « attraper » l'information sur un appareil avec un stylo pour la transporter vers un autre. Mais la prolifération des standards et les stratégies protectionnistes des constructeurs (leur propre « écosystème ») rendent difficile, voire impossible, l'interopérabilité entre appareils que nécessitent de telles interactions.

L'Ubicomp a été le premier d'une série de concepts d'interaction qui visent tous à mieux intégrer le monde numérique et le monde physique et abolir, ou tout au moins réduire, les frontières entre ces deux mondes. Ainsi, la *réalité augmentée* est inventée en 1993 en réaction à la *réalité virtuelle*, alors très en vogue. L'objectif est ici d'intégrer l'information numérique directement au sein des objets physiques plutôt que de la confiner dans le monde informatique de l'ordinateur.

Le *Digital Desk*, développé par Pierre WELLNER à RANK XEROX EURO-PARC, est le premier système de réalité augmentée et probablement le plus emblématique : grâce à un projecteur et une caméra montés au-dessus d'un bureau traditionnel, l'ordinateur peut suivre et interpréter les manipulations d'objets physiques posés sur le bureau, comme des feuilles de papier et, projeter des informations ou des applications, comme une calculette, que l'on peut manipuler à même le bureau.

Une autre approche de réalité augmentée initiée au même moment — notamment par Steven FEINER à Columbia —, utilise des dispositifs de réalité virtuelle pour superposer des images au monde physique.

À la même époque apparaissent les interfaces dites *tangibles*, produites notamment au *MIT Tangible Media Group* de Hiroshi ISHII au *Media Lab* du *Massachusetts Institute of Technology*. L'idée est de rendre l'information concrète, de lui donner une matérialité qui permet de tirer partie de notre facilité à manipuler des objets physiques.

L'exemple canonique d'une interface tangible est la « *Marble Answering Machine* » de Durrell BISHOP. Il s'agit d'un répondeur téléphonique dont chaque message est représenté par une bille, qui sort de la machine lorsque le message est reçu. On écoute le message en posant la bille sur un emplacement prévu à cet effet sur le répondeur, on l'efface en la remettant dans le répondeur, mais on peut aussi l'emporter avec soi comme pense-bête. De nombreuses interfaces tangibles ont été développées dans les laboratoires depuis vingt-cinq ans et continuent de l'être. Elles ont inspiré certains dispositifs de l'Internet des objets et on peut parier que ce mouvement va se poursuivre.

Un dernier exemple de la fusion des mondes physique et numérique est le papier interactif. Dès 1974, XEROX, encore lui, a inventé le *Gyricon*, un papier électronique dont une variante est utilisée aujourd'hui dans les liseuses électroniques. Mais c'est la technologie de l'entreprise suédoise ANOTO, au début des années 2000, qui permet réellement d'envisager des applications interactives. Grâce à un stylo muni d'une micro-caméra et d'un papier sur lequel est imprimée une trame de points quasiment invisible à l'œil nu, tous les tracés réalisés avec le stylo sont envoyés en temps réel à un ordinateur. On peut donc écrire sur du vrai papier, tandis que l'ordinateur analyse ces traces.

Wendy MACKAY, de l'INRIA, travaille depuis vingt-cinq ans sur le papier interactif et a créé de nombreux prototypes : des « *strips* » de papier utilisés par les contrôleurs du trafic aérien, des cahiers de laboratoire interactifs utilisés par des biologistes, des interfaces papier



Source : Pierre Wellner

Digital Desk, En haut, vue générale du prototype; en bas, illustration du détail de son utilisation.

pour la création musicale. Alors que l'écran de l'ordinateur devait nous débarrasser du papier, avec le mythe du bureau sans papier, c'est peut-être finalement celui-ci qui va nous débarrasser des écrans !



© Inria - Photo H. Raguët

Philippe LEROUX utilisant le papier interactif pour créer une composition musicale et contrôler son interprétation par l'ordinateur. « Quid sit Musicus » : Wendy MACKAY, Jérémie GARCIA et Philippe LEROUX.

G. INTERACTION GESTUELLE

Pour terminer ce tour d'horizon, un dernier pan de l'histoire de l'interaction est abordé. L'avènement des *smartphones* à écran tactile puis des tablettes d'une part, des jeux vidéos utilisant les mouvements du corps entier d'autre part, l'interaction gestuelle a eu un impact considérable cette dernière décennie. D'où proviennent ces technologies ?

C'est en 1964 que la première tablette graphique est créée par Tom ELLIS dans l'entreprise RAND. Cette tablette utilise un stylet et ELLIS développe le système GRAIL, le premier à reconnaître les marques tracées avec le stylet, comme les lettres de l'alphabet.

En 1972, le système PLATO IV de l'Université d'Illinois, destiné à l'enseignement assisté par ordinateur, est l'un des tous premiers à utiliser un écran tactile : écran à plasma de 512×512 pixels est muni d'une grille infrarouge tactile de 16×16 cases.

En 1985, Bill BUXTON, de l'Université de Toronto, développe la première tablette capable de détecter plusieurs points de contact simultanés avec leur pression. Il faudra plus de trente ans pour qu'un *smartphone*, l'*iPhone 6S*, intègre un écran multi-tactile sensible à la pression.

Dès 1969, Myron KRUEGER crée des installations qui permettent aux utilisateurs d'interagir avec l'ensemble de leur corps grâce à l'analyse en temps réel de leurs mouvements. Il invente le terme *Artificial Reality* pour décrire ces espaces d'interaction qui jouent avec les lois de la physique. La plus connue de ces installations est *VideoPlace*, présentée à partir de 1974. La caméra Eye Toy de SONY, commercialisée en 2003, et la Kinect de MICROSOFT, dévoilée en 2009, sont les descendants directs de *VideoPlace*. Il est intéressant de constater que ce sont les jeux vidéo qui sont les héritiers des installations de Myron KRUEGER. En fait, les jeux sont depuis longtemps un vecteur d'innovation en matière d'interaction, avec l'utilisation de périphériques d'entrée dédiés ou l'utilisation massive de la 3D.

En 1980, Rich BOLT du MIT présente le système *Put-That-There*, premier système dit multimodal qui combine la reconnaissance des gestes de la main dans l'espace, la désignation sur un grand écran grâce à un capteur à six degrés de liberté et la reconnaissance de la parole. Ce système vise des applications d'aide à la décision, notamment en situation



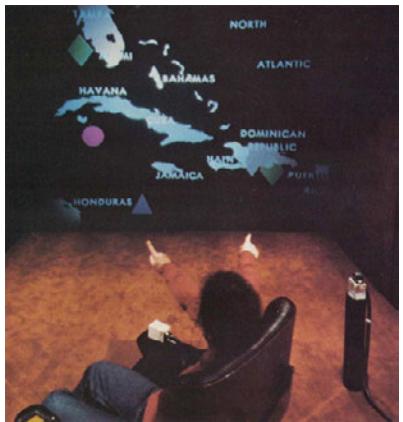
Source : Bill Buxton

Écran tactile du système Plato IV.



Source : Inventing Interactive

VideoPlace de Myron KRUEGER.



Source: Rich Bolt - MIT

Put-that-there de Rich Bolt : Architecture Machine Group Massachusetts Institute of Technology.

de crise comme le montre le scénario choisi de la crise de Cuba. Comme la plupart des applications multimodales qui lui ont succédé, *Put-That-There* met l'accent sur l'interaction vocale, les gestes servant à compléter de façon naturelle l'information transmise par la voix. Trente-cinq ans plus tard, bien que les assistants vocaux commencent à se généraliser, force est de constater que l'interaction multimodale n'est pas encore présente dans des produits commerciaux grand public.

Contrairement à certaines idées reçues, ce n'est donc pas APPLE qui a inventé l'interaction tactile avec l'*iPhone*, ni de même MICROSOFT qui a conçu l'interaction du corps entier avec la *Kinect*. Il ne faut pas pour autant sous-estimer le travail considérable réalisé par ces entreprises pour résoudre tous les problèmes techniques ayant conduit à l'industrialisation de ces produits et en assurer le succès.

De la même manière, les assistants vocaux d'AMAZON, de GOOGLE, d'APPLE et de MICROSOFT sont le résultat de longs et anciens travaux de recherche sur la reconnaissance de la parole, que nous n'avons pas la place de détailler ici. On peut cependant prédire que ces assistants seront tôt ou tard munis de capacités multimodales pour améliorer l'interaction.

H. BILAN : RETOUR VERS LE FUTUR

Cet historique de l'interaction humain-machine montre que la plupart des concepts des interfaces actuelles sont anciens. Cela ne doit pas surprendre : les technologies mettent généralement une trentaine d'années à passer des travaux initiaux au sein des laboratoires de recherche à une diffusion de masse ; l'informatique comme l'IHM n'échappent pas à cette règle.

Néanmoins, il s'avère que de nombreuses innovations significatives en IHM sont passées inaperçues. Par exemple, les menus circulaires — ou « *pie menus* » en anglais —, inventés en 1986 par Don HOPKINS puis améliorés en 1993 par Gordon KURTENBACH, peuvent diviser par trois le temps de sélection dans un menu. Pourtant, ils ne sont implantés dans aucune application commerciale de masse. C'est une autre loi de l'innovation, qui veut que de multiples inventions soient oubliées ou ignorées, jusqu'à ce qu'elles soient éventuellement redécouvertes.

Par ailleurs, il est plus préoccupant de constater que les visions de Doug ENGELBART et de Ted NELSON, qui datent de près de cinquante ans, sont loin d'être réalisées et que les interfaces graphiques actuelles sont une pâle copie de ce que permettait et promettait le *Star* il y a trente-cinq ans. En effet, la réalisation de ces visions risque fort d'être impossible avec les interfaces actuelles et leur cortège de standards incontournables comme les « *legacy applications* » — applications anciennes que l'on doit faire fonctionner dans un environnement moderne.

Comment alors imaginer que le modèle centré sur les applications, à ce jour dominant, puisse être remplacé sans rupture majeure par celui du *Star*, centré sur les documents ? Comment envisager que le Web puisse devenir un véritable média pour la collaboration distante, sans remettre en cause les protocoles existants ? Ces ruptures sont pourtant nécessaires, car les interfaces actuelles atteignent leurs limites : elles génèrent leur propre complexité d'usage et détournent l'utilisateur de l'objet de sa tâche ; elles ne tirent pas parti des capacités d'action, de perception et de communication des opérateurs humains ; elles ne sont pas adaptées à leurs contextes d'exploitation.

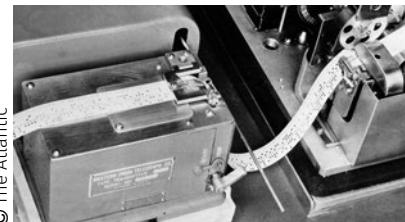
Entre les risques d'une telle régression et les promesses des interfaces à base d'agents intelligents, de langage naturel, d'*Ubicomp* et de réalité augmentée, l'IHM devra se frayer un chemin tandis que les chercheurs, entre évolution et révolution, continuent à inventer le futur.

NOTE DE LA RÉDACTION

L'informatique ubiquitaire (ou succinctement « *Ubicomp* ») est la troisième ère informatique. Elle succède à celles des ordinateurs centraux (*mainframe*) puis personnels (PC). L'usager dispose d'une gamme de petits appareils informatiques tels que le téléphone multifonction ou l'assistant personnel et leur emploi fait partie de sa vie quotidienne (source [W](#)).

RÉFÉRENCES BIBLIOGRAPHIQUES CIRCONSTANCIÉES

- [BERNERS-LEE et al., 1994] — Tim BERNERS-LEE, Robert CAILLIAU, Ari LUOTONEN, Henrik Frystyk NIELSEN, and Arthur SECRET, 1994 — “*The World-Wide Web*”, *Communications of the Association for the Computing Machinery*, vol. 37, n°8, pp. 76–82.
- [BOLT, 1980] — Richard A. BOLT, 1980 — “*Put-that-there: Voice and Gesture at the Graphics Interface*”, *Association for the Computing Machinery SIGGRAPH Computer Graphics*, vol. 14, n°3, pp. 262–270.
- [BUSH, 1945] — Vannevar BUSH, 1945 — “*As We May Think*”, *The Atlantic Monthly*, n°176, pp. 101–108. Reprinted and discussed in ACM Interactions, 3(2), March 1996, pp. 35–67.
- [CALLAHAN et al., 1988] — Jack CALLAHAN, Don HOPKINS, Mark WEISER, and Ben SHNEIDERMAN, 1988 — “*An Empirical Comparison of Pie vs. Linear Menus*”, CHI’88. Association for the Computing Machinery, Washington, D.C., USA, pp. 95–100.
- [ENGELBART, 1962] — Douglas C. ENGELBART, 1962 — “*Augmenting Human Intellect: A Conceptual Framework*”, Summary report, project n°3578. Contract AF 49(638)-1024. Stanford Research Institute: Menlo Park, California. 134 pages.
- [ENGELBART and ENGLISH, 1968] — Douglas C. ENGELBART and William K. ENGLISH, 1968 — “*A Research Center for Augmenting Human Intellect (AUGMENT 3954)*”, vol. 33. American Federation of Information Processing Societies, San Francisco, CA, pp. 395–410. Republished with articles n°4, 21, and 23 in *Computer Supported Cooperative Work: A Book of Readings*, Irene Greif [Ed.], Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988, pp. 81–105. Also republished with article n°3 in *New Media Reader*, Noah Wardrip-Fruin and Nick Montfort [Ed.], The MIT Press, 2003, Chapters 8 and 16.
- [ENGELBART, 1968] — Douglas C. ENGELBART, 1968 — *A Research Center for Augmenting Human Intellect (AUGMENT 3954)*, 90 minutes video recording of live online hypermedia demonstration/presentation at the Fall Joint Computer Conference, San Francisco, CA, December 9, 1968. Stanford Research Institute.
- [JOHNSON et al., 1989] — Jeff A. JOHNSON, Teresa L. ROBERT, William VERPLANK, David C. SMITH, Charles IRBY, Marian BEARD, and Kevin MACKY, 1989 — “*The Xerox ‘Star’: A Retrospective*”, *IEEE Computer*, vol. 22, n°9, pp. 11–26. Reprinted in Ron M. Baeker, Jonathan Grudin, William Buxton, Saul Greenberg (Eds.), *Human Computer Interaction: Toward the Year 2000*, Morgan Kaufman Publishers, 1995.
- [KURTENBACH and BUXTON, 1993] — Gordon KURTENBACH and William BUXTON, 24–Apr. 29, 1993 — “*The Limits of Expert Performance Using Hierarchic Marking Menus*”, INTERCHI’93. Association for the Computing Machinery, Amsterdam, The Netherlands, pp. 482–487.
- [KRUEGER and WILSON, 1985] — Myron W. KRUEGER and Stephen WILSON, 1985 — “*‘VIDEOPLACE’: A Report from the Artificial Reality Laboratory*”, *Leonardo*, vol. 18, n°3, pp. 145–151.
- [KRUEGER, 1991] — Myron W. KRUEGER, 1991 — “*Artificial Reality II*”, Addison-Wesley Professional, Boston, 304 pages.
- [MYERS, 1998] — Brad A. MYERS, 1998 — “*A Brief History of Human Computer Interaction Technology*”, *ACM Interactions*, vol. 5, n°2, pp. 44–54.
- [NELSON, 1993] — Theodor H. NELSON, 1993 — “*Literary Machines 93.1. The report on, and of, project Xanadu concerning word processing, electronic publishing, hypertext, thinkertoys, tomorrow’s intellectual revolution, and certain other topics including knowledge, education and freedom*”, Sausalito, California. 288 pages.



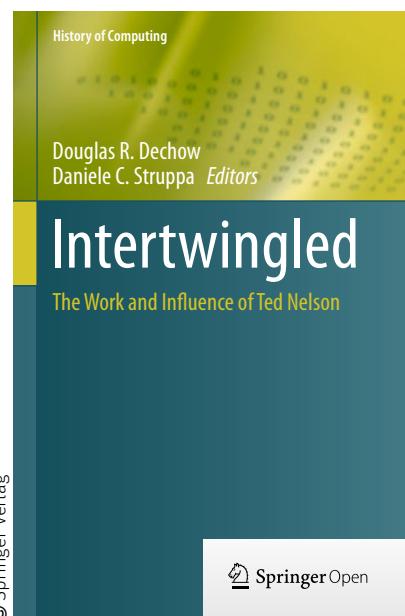
© The Atlantic

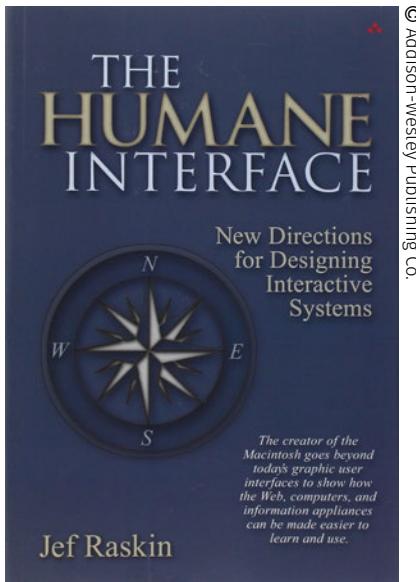
Selon Vannevar Bush, le télégraphe était une avancée technologique sans précédent qui pouvait évoluer de manière significative.



© Doug Engelbart Institute

Dispositif de la conférence en ligne conduite le 9 décembre 1968 (Fall Joint Computer Conference).





Aperçu de la banque mémoire et de la console du TX2.

[PERKINS et al., 1997] — Roderick PERKINS, Dan KELLER, and Frank LU-DOLPH, 1997 — “*Inventing the Lisa User Interface*”, *ACM Interactions*, vol. 4, n°1, pp. 40–53.

[RASKIN, 2000] — Jef RASKIN, 2000 — “*The Humane Interface. New Directions for Designing Interactive Systems*”, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 256 pages.

[SHNEIDERMAN, 1983] — Ben SHNEIDERMAN, 1983 — “*Direct Manipulation: A Step Beyond Programming Languages*”, *IEEE Computer*, vol. 16, n°8, pp. 57–69.

[David Canfield SMITH et al., 1982] — David Canfield SMITH, Charles IRBY, Ralph KIMBALL, and Eric HARSLEM, 1982 — “*The Star User Interface: an Overview*”, AFIPS’82. Association for the Computing Machinery, Houston, Texas, pp. 515–528.

[SUTHERLAND, 1963] — Ivan E. SUTHERLAND, 1963 — “*SketchPad: A Man-Machine Graphical Communication System*”, AFIPS’63. Association for the Computing Machinery, Detroit, Michigan, pp. 329–346. Published in September 2003 as an electronic technical report by the University of Cambridge, Computer Laboratory, UK, ISSN: 1476-2986.

[WEISER, 1991] — Mark WEISER, 1991 — “*The Computer for the Twenty-First Century*”, *Scientific American*, vol. 265, n°3, pp. 94–104.

[WELLNER et al., 1993] — Pierre WELLNER, Wendy MACKAY, and Rich GOLD, 1993 — “*Back to the Real World*”, *Communications of the ACM*, vol. 36, n°7, pp. 24–26. Special issue on Computer-Augmented Environments.

LIENS SUR L'HISTOIRE DE L'IHM

- ▶ Histoire de l'informatique : [The Machine That Changed the World](#) et [The History of Computing](#).
- ▶ Histoire de l'IHM : [A Brief History of Human Computer Interaction Technology](#); Howard Rheingold’s [Tools For Thought](#).
- ▶ Sketchpad : [Sketchpad : A man-machine graphical communication system](#) ; Early HCI Research by the Lincoln Lab TX-2 Group (MIT).
- ▶ NLS/Augment : [Doug Engelbart’s Biography](#); 1968 Demo.
- ▶ Xerox Star : [Xerox Star, a Retrospective](#); [Xerox Star Historical Documents](#) (courtesy Dave Curbow); Vidéos du Star sur YouTube; Bruce Damer’s Personal Histories of the Desktop User Interface; Biographie de Alan Kay; Alan Kay Viewpoints Research Institute.*
- ▶ Apple : [apple-history.com](#); [The Apple Museum](#); Jef Raskin.*
- ▶ VisiCalc : [History Introduction by Dan Bricklin](#)
- ▶ World Wide Web : [A Little History of the World Wide Web](#); [Histoire du Web](#) (WWW Foundation); [Histoire de l'hypertexte](#) (J. Nielsen).
- ▶ Pie Menus : [The Design and Implementation of Pie Menus](#).

* Lien mort.

POUR ALLER PLUS LOIN...

RESSOURCES COMPLÉMENTAIRES

- ▶ L'être humain au cœur de la recherche en IHM, entretien avec Wendy Mackay. Binaire (Le Monde) — L'informatique : science et technique au cœur du numérique —, 18/07/2014;
- ▶ Idée reçue : Grâce au numérique, on peut lire dans les pensées, par Fabien Lotte, Interstices, 13 septembre 2013. Pour démythifier les interfaces cerveau-machine;
- ▶ A propos de l'interaction homme-machine, entretien avec Nicolas Roussel, Interstices, 26 avril 2013. Une discussion critique sur les nouvelles interfaces;

- ▶ L'homme doit contrôler l'ordinateur pas l'inverse (lien mort), entretien avec Wendy Mackay, Futuremag, Arte, 16/11/2015. À propos des lunettes connectées;
- ▶ Favoriser l'interaction 3D sur des surfaces tactiles, projet INSTINCT (voir aussi le système TOUCHEO);
- ▶ En termes de recherche, on peut consulter les conférences annuelles de l'AFIHM — Association Francophone d'Interaction Homme-Machine, toutes accessibles sur une plateforme ouverte : [IHM'2018](#), [IHM'2017](#), [IHM'2016](#), [IHM'2015](#).

4 Aspects légaux, économiques et éthiques

La collecte massive des données modifie-t-elle la notion, chère à la société française, de « vie privée » ? Le droit doit-il s'adapter à une société désormais devenue numérique ? Pourquoi et comment expliquer les enjeux et les résultats d'un algorithme ?

L'articulation entre droit et techniques numériques pose autant de questions de droit, de société et d'éthique auxquelles tout citoyen doit normalement être sensible donc, par voie de conséquence, sensibilisé.

4.1 Numérique : loi et vie privée

En France — et de manière plus générale en Europe —, s'applique une des législations les plus protectrices en matière de données personnelles et de vie privée. Cependant, les nouvelles technologies, leur déploiement et leurs nouveaux usages mettent à mal, voire en péril, ces protections juridiques. Pour simplifier, on peut dire que plusieurs phénomènes concomitants se conjuguent.

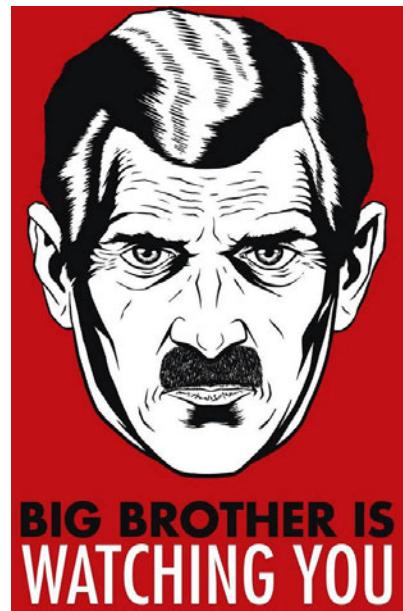
4.1.1 Collecte et utilisation des données

Un nombre toujours croissant de données est continuellement collecté de diverses manières. En naviguant sur Internet, en utilisant des téléphones mobiles et, de plus en plus à l'avenir avec l'avènement de l'Internet des objets (IoT), des quantités pharaoniques de données sont recueillies, souvent à l'insu des personnes concernées.

Ces données, agrégées et engrangées en masse, puis analysées dans la sphère de ce qu'on appelle aujourd'hui le *big data*, suscitent l'intérêt vorace de multiples acteurs et organisations pour les exploiter.

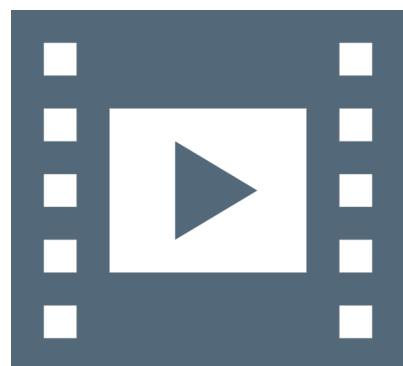
On voit là clairement poindre des tensions entre la protection de la vie privée telle qu'elle est envisagée par le droit et ce qu'il est convenu d'en faire en pratique. Typiquement, en prenant comme illustration le *big data*, la logique est la récupération du plus de données possibles pour ensuite en inférer de nouvelles connaissances cependant, sur des sujets, sur des thèmes, dans des directions qui ne sont pas forcément connus au moment de la collecte. Cette manœuvre s'avère complètement opposée à ce que prévoit la loi puisque, de nos jours, en France comme en Europe, quand des données personnelles sont rassemblées, cela doit être en regard de finalités précises, définies au moment de la collecte qui, une fois réalisées, impliquent l'effacement desdites données. Cette perspective est totalement inverse de celle des acteurs de l'analyse de données à grande échelle.

Pour autant, les technologies de l'information et de la communication (Tic) ne sont pas définitivement et indubitablement des menaces pour la vie privée. On peut effectivement aussi concevoir des nouvelles



À PROPOS DE L'INTERVENANT

Spécialiste en protection de la vie privée, Daniel LE MÉTAYER est un chercheur (DR) dans l'équipe [PRIVATICS](#) du centre INRIA GRENOBLE-RHÔNE-ALPES. Jusqu'en juin 2016, il a été responsable de l'INRIA PROJECT LABS CAPPRIS qui regroupait les équipes actives dans le domaine de la protection de la vie privée. Ses activités tournent autour des interactions entre le droit et les nouvelles technologies. Il lui arrive également d'alerter le public sur les limites et les dangers de certains projets comme ce fut le cas lors de la loi sur le renseignement en 2015 et du décret instituant le fichier TES en 2016.



VIDÉO 1.6 — Loi et vie privée.



techniques visant à protéger la vie privée et résoudre le genre de hiatus évoqué avec le *big data*.

Cette orientation positive représente des sources de sujets de recherche en matière d'informatique et de TIC. On peut notamment mentionner ici les techniques d'anonymisation. Il est connu — cela se trouve d'ailleurs prévu par la loi — que des données anonymes échappent aux lois de protection des données personnelles car, de fait et *par définition, une donnée anonyme n'est plus personnelle*.

Ainsi, afin d'exploiter des données à grande échelle, il est pratique de les avoir anonymisées auparavant. Toutefois, anonymiser des données est une tâche relativement complexe.

Avant tout, que signifie une donnée vraiment anonyme ? Hélas, cela se trouve tout bonnement très difficile à définir. C'est pourquoi, une des recherches majeures de ce champ d'application concerne actuellement les techniques d'anonymisation robustes.

4.1.2 Contrôle et exploitation des données



Un autre droit, prévu en France et en Europe, est celui du contrôle des données personnelles — et normalement garanti en France par la CNIL —, quelques fois dénommé par le terme d'*autodétermination informationnelle*. Comme d'autres, ce droit est mis à mal de différentes manières, en particulier par l'usage de plus en plus répandu d'algorithmes de prédiction ou d'algorithmes d'aide à la prise de décision.

Cela concerne de multiples champs d'application et impacte même nos activités quotidiennes. En prenant exemple de la thématique des médias informationnels, on constate qu'un nombre croissant de personnes se renseignent et obtiennent leurs informations sous le prisme des réseaux sociaux qu'elles consultent. En outre, ce dont on est pas nécessairement conscient, est que l'accès à ces informations est, en amont, dûment classé et hiérarchisé par le biais d'algorithmes dont on ne connaît pas forcément le fonctionnement précis.

Cet état de faits peut avoir des incidences majeures, notamment en termes de droit à l'information et éventuellement de censure. Cela s'est vu à l'occasion de l'élection présidentielle nord-américaine de 2016 où certains réseaux sociaux ont été accusés de classer les informations de manière biaisée, en favorisant un candidat plutôt qu'un autre; accusations renforcées par la méconnaissance des algorithmes utilisés.

Cela s'avère un exemple parmi tant d'autres. On peut ici également citer les algorithmes adoptés par certaines entreprises aux États-Unis pour classer les candidatures et recruter leurs employés, sans même avoir à conduire des entretiens d'embauche. Dans le domaine des assurances, il est fort vraisemblable de se voir proposer des primes d'assurance calculées en fonction d'algorithmes qui profitent les individus; même chose pour des prêts bancaires.

Peut-être encore plus grave, ce genre d'algorithmes commence à s'utiliser dans les institutions judiciaires et de police — sous le doux nom de *police prédictive*. Au visionnage du film « *Minority report* » réalisé en 2002 par Steven SPIELBERG, on peut envisager que ce genre de scénarios un peu dystopiques³⁵ se profilent à l'horizon, avec une police prédictive déjà à l'œuvre dans certains comtés aux États-Unis.

À l'aide de ce type d'algorithmes, peuvent se prévoir les endroits et jours de la semaine où la probabilité de crimes ou de méfaits divers est la plus importante. En fonction des résultats, les forces de l'ordre sont dépêchées en lieux et moments ainsi déterminés. À juste titre, on peut s'inquiéter des dérives d'un tel système, comme les risques de discrimination et de stigmatisation de certaines populations.

³⁵. Une dystopie est un récit de fiction anticipative dépeignant une société imaginaire organisée de telle manière qu'elle empêche ses membres d'atteindre le bonheur. Une dystopie peut également être considérée, entre autres, comme une utopie qui vire au cauchemar et conduit donc à une contre-utopie. (Source [W](#))

Ainsi, tous ces usages plus ou moins opaques des algorithmes sont de nature à porter atteinte aux fondements des sociétés de droit comme les démocraties, en apportant des traitements déloyaux, voire liberticides comme celui du droit à l'information mentionné plus haut.

4.1.3 Logiciels embarqués et automatisation

Outre les algorithmes d'aide à la prise de décision, il existe des situations où des véhicules — trains, métros, avions, fusées, satellites et, à coup sûr demain, voitures autonomes — ont du code logiciel embarqué. Pour pallier des défaillances potentielles ou des temps de réaction trop importants dus au facteur humain, ces codes prennent des décisions par eux-mêmes. Bien conçus, cela apporte une plus-value manifeste. Cependant, on peut s'interroger sur le bien fondé de cette délégation numérique dans certains cas critiques, lorsqu'il s'agit par exemple d'une question de vie ou de mort. Notamment pour une automobile autonome face à un piéton : quelle manœuvre est à réaliser ? Tourner quitte à aller dans le décor, freiner ou accélérer ? Qui privilégier, le piéton ou l'automobiliste ? Quelles sont les responsabilités et qui les assument en cas de dysfonctionnement ?

Face à des logiciels gigantesques de plusieurs millions de lignes de code, fournis par de multiples acteurs et sous-traitants, comment alors déterminer que la défaillance et les préjudices ont été causés par un composant logiciel spécifique, produit par un fournisseur particulier ?

À nouveau, cela soulève des questions vraiment complexes qui mettent en jeu évidemment des notions juridiques, de responsabilité et de dédommagement, mais encore des problématiques purement techniques, parce qu'il faut être capable d'analyser l'historique des événements — fichiers dits de *logs* — et de pointer l'endroit où les erreurs se sont produites, donc indirectement les composants et les fournisseurs qui doivent en supporter la responsabilité.

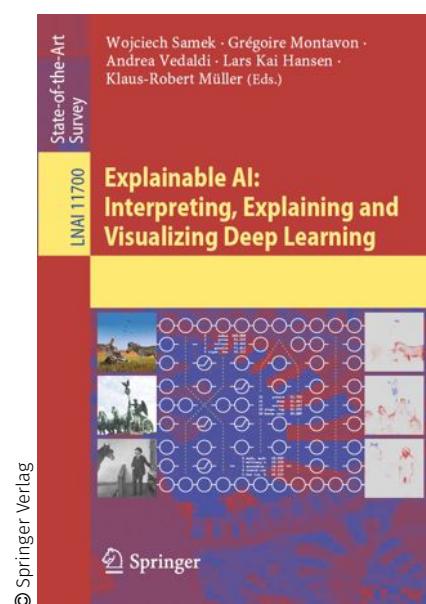
Tout ceci montre qu'à l'articulation du droit et de la technique, énormément d'incidences se cachent derrière la notion *a priori* banale de logiciel, désormais usuel au quotidien. Ce sont des questions de société, voire même d'éthique.

4.1.4 Éducation et transparence

La robotique s'invite toujours plus dans notre ordinaire : habitat, institutions, milieux sociaux, télémédecine... Et, bien entendu, applications militaires. Par conséquent, cela suppose que ces sujets ne soient vraiment pas laissés aux seuls techniciens, mais bien que l'entièreté de la société s'en préoccupe : décideurs de tout ordre dont, en « premiers de cordée », les représentants politiques, mais encore, voire surtout, les citoyens au sens le plus large possible.

Il est également crucial que les nouvelles générations, lycéens comme collégiens, s'emparent de ces questions et y soient initiés au travers de l'enseignement qui leur est dispensé. Au-delà des services apportés par le numérique, il est capital que la population en comprenne au mieux les contreparties juridiques, politiques et éthiques.

Outre le traitement des grandes masses de données par des solutions techniques d'anonymisation, un autre sujet majeur qui déclenche des travaux de recherche importants en ce moment concerne la transparence et, spécifiquement celle des algorithmes. Cela pose des questions extrêmement complexes. Par exemple, comment montrer qu'un algorithme utilisé pour filtrer des candidats à recruter pour un poste donné ne va pas induire des discriminations dans la procédure de sélection ? Pour cela il ne suffit pas de se contenter de publier le code du



logiciel en question — cf. logiciels libres et *open source* —, il faut être capable de l'analyser et de le comprendre.

Ainsi, sous l'appellation anglophone de *explicable Artificial Intelligence* — XAI, se profilent de nouveaux courants de recherche. La perspective annoncée est d'arriver à expliquer le résultat d'un algorithme, notamment les algorithmes utilisés en intelligence artificielle et qui reposent sur l'apprentissage. Ces derniers sont désormais couramment employés et à même de prédire ou d'établir des liens, des classifications, des relations très subtiles et quelques fois extrêmement précises, mais desquels on n'est pas en mesure d'expliquer le résultat. Cela engendre de grandes difficultés pour ceux qui ont à exploiter ces déductions et, dans certains cas, soulève des questions politiques et de société, à savoir : peut-on faire confiance à des algorithmes dans des circonstances où leurs conclusions amènent à prendre des décisions essentielles concernant des individus ? Bien que ces travaux soient encore émergents, ils vont probablement s'avérer prépondérants dans la décennie 2020-2030 à venir.

GLOSSAIRE CONTEXTUEL

INTERNET DES OBJECTS (IoT) — L'Internet des objets — en anglais *Internet of Things* —, représente l'extension d'Internet à des éléments et à des lieux du monde physique.

BIG DATA — Parfois qualifié comme *données massives*, le terme *big data* désigne des ensembles de données qui s'avèrent tellement volumineux qu'ils en deviennent difficiles à traiter avec les outils traditionnels de gestion de base de données ou de l'information. Le *big data* offre de nouvelles perspectives en matière d'analyse de données, notamment à des fins prédictives, avec des retombées espérées dans de nombreux domaines : santé, environnement, gestion des risques, etc.

TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION — Transcription de l'anglais ICT pour *Information and Communication Technologies*, TIC est un acronyme essentiellement utilisé dans les mondes universitaires et de l'administration pour désigner le domaine couvert par la **télématique** — informatique, audiovisuel, multimédias, Internet, télécommunications, etc. — qui permet aux utilisateurs d'accéder, produire, manipuler, communiquer des informations sous toute forme : texte, son, image, vidéo, etc. (source [W](#)).

TECHNIQUE D'ANONYMISATION — Les techniques d'anonymisation se rapportent aux moyens de transformer des jeux de données — effacement de certains attributs, généralisation, bruitage et autres manipulations —, afin de rendre très difficile voire impossible la « ré-identification » ou l'inférence de connaissances sur des personnes (physiques ou morales).

ALGORITHME DE PRÉDICTION — Algorithme qui permet de déterminer — pour de nouvelles données —, des caractéristiques à partir de la connaissance de données d'apprentissage préalablement agrégées. En quelque sorte, l'algorithme apprend des règles en se fondant sur les données d'apprentissage et les applique à de nouvelles données. Cela se réalise, entre autres, à l'aide de probabilités, de statistiques et de régressions.

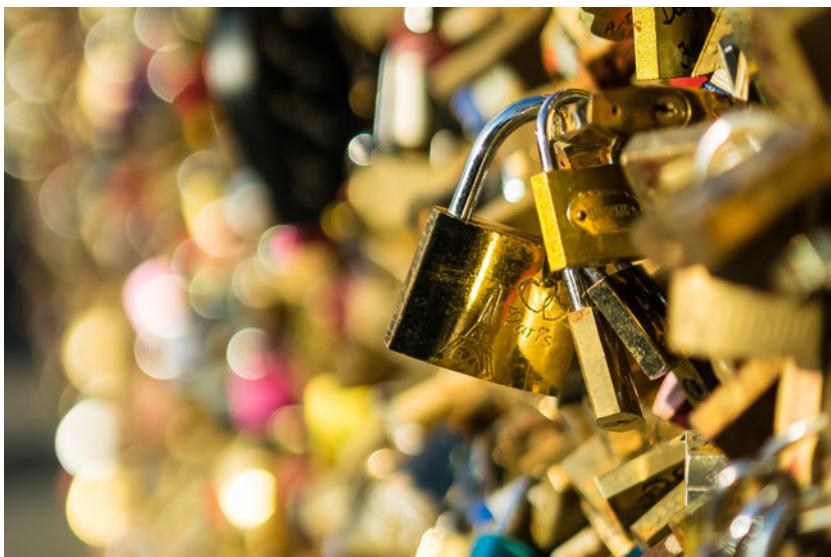
ALGORITHME D'AIDE À LA PRISE DE DÉCISION — Algorithme dont l'objet est d'apporter des informations utiles pour la prise de décision (par exemple en fournissant un classement pertinent, comme les algorithmes de recommandation et les moteurs de

recherche ou encore, en répertoriant des éléments dans des catégories). Dans certains cas, ces algorithmes peuvent même être utilisés dans des processus entièrement automatisés.

LOGICIEL EMBARQUÉ — Un *système embarqué* — ou logiciel embarqué — est défini comme un système électronique et informatique autonome, fonctionnant souvent en temps réel, spécialisé dans une tâche bien précise et intégré au sein d'une entité ou d'un objet : voiture, train, avion...

ROBOTIQUE — La *robotique* regroupe l'ensemble des techniques permettant la conception et la réalisation de machines automatiques — ou robots — à même de réaliser des tâches précises, quelque soit leur champ d'application : industrie, domotique, transport, médecine, sciences, etc. (source [W](#)).

4.2 Ville numérique : quels impacts sur la vie privée ?



Dans un article publié dans *Government Information Quarterly*, la sociologue Liesbet VAN ZOONEN propose un instrument de sensibilisation à destination des collectivités territoriales [VAN ZOONEN, 2016].

La ville numérique ou « *smart city* » suscite de nombreux débats parmi les chercheurs et les spécialistes. Ses promoteurs voient dans le *big data* une parfaite opportunité pour les villes de devenir plus riches, plus propres et plus efficaces; d'autres considèrent au contraire que les villes deviendront des espaces robotisés, ennuyeux, gérés par les seules *data – data driven* — où la créativité n'aura plus aucune place.

Le géographe Rob KITCHIN considère que la collection tous azimuts de données urbaines risque de produire des villes « panoptiques³⁶ », qui risquent de menacer le droit à la vie privée et à la liberté d'expression. Liesbet VAN ZOONEN considère que, si l'on ne prend pas en compte les aspects de vie privée, les projets de *smart cities* deviendront sujets à controverses, voire disparaîtront. Pour la CNIL, le sujet des *smart cities* fera l'objet d'une attention toute particulière de ses travaux d'innovation et de prospective en 2017...

A. « *PRIVACY FRAMEWORK** »

* Cadre de confidentialité

Quelles données? Personnelles ou non? Pour quels objectifs? C'est à partir de ces interrogations et en se fondant sur l'étude de la ville de

NOTE DE LA RÉDACTION

Texte publié en septembre 2016 sur le site du Laboratoire d'innovation numérique de la CNIL — Commission nationale de l'informatique et des libertés — (LINC), rédigé par Régis CHATELLIER, chargé des études prospectives.

³⁶. Selon le CNRTL, une structure « panoptique » désigne un bâtiment (prison, maison de correction) aménagé de telle sorte que d'un point de l'édifice on puisse voir tout l'intérieur.

ROTTERDAM — PAYS-BAS —, que Liesbet VAN ZONEN analyse des risques en termes de protection de la vie privée.

La sociologue emploie dans le premier graphique de son article (cf. [VAN ZONEN, 2016] et figure 1.13) le terme « *impersonal* » pour les données qui ne sont pas « à caractère personnel », auquel lui sera préféré en français le vocable de « non-personnelles ».

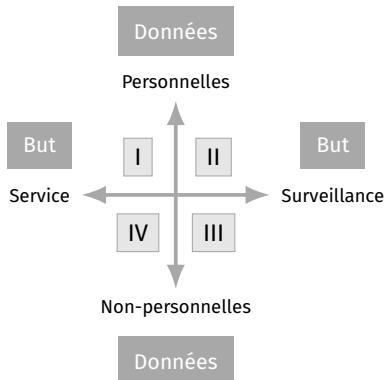


FIGURE 1.13 — Défis de vie privée des villes numériques.

On retrouve ainsi quatre quadrants aux propriétés différentes :

I. *Données personnelles à des fins de services.*

Il s'agit des données traditionnellement collectées par les collectivités — état civil, adresse, bureau de vote, profession, etc. —, des données liées à la situation économique — telles que chômage ou allocations sociales —, mais aussi des données relatives à la correspondance en ligne et aux réseaux — adresse électronique, téléphonie, etc. Les finalités sont liées aux études démographiques, à l'urbanisme, aux services municipaux ou sociaux : autant d'usages qui ne comportent pas, selon la sociologue, d'enjeux de vie privée pour les habitants, à condition de respecter certaines règles et que le marché reste clair : une donnée pour un service. La relativisation des enjeux de vie privée vient du fait que les individus sont actifs et donc conscients que des données les concernant sont utilisées pour rendre un service.

II. *Données personnelles à des fins de surveillance.*

Ce quadrant recouvre les données personnelles collectées par la police, les autorités organisatrices des transports publics, mais encore les caméras de surveillance, les bases de données relatives à la reconnaissance faciale, etc. Ce quadrant cristallise les contestations des citoyens et des militants du respect de la vie privée. L. VAN ZONEN notifie que le règlement européen apportera de nouvelles garanties aux citoyens pour faire valoir et protéger leurs droits.

III. *Données non-personnelles à des fins de surveillance.*

On retrouve ici les données liées à la surveillance de l'espace urbain, sans *a priori* de reconnaissance possible des individus : *monitoring* du trafic automobile ou gestion des foules, obtenus par exemple au moyen de caméras infrarouge ou de capteurs de chaleur. Ces données peuvent cependant permettre de reconnaître une personne, notamment lorsqu'on utilise un logiciel de reconnaissance faciale sur des vidéos.

IV. *Données « non-personnelles » à des fins de service.*

La dernière catégorie correspond aux données non-personnelles collectées et utilisées au bénéfice des citoyens, telles que les données environnementales, de gestion des déchets ou de l'énergie (hors données de chaque foyer). Ce quadrant ne présente de prime abord pas d'inquiétude quant au respect de la vie privée, mais l'auteur rappelle que là aussi, grâce à l'agrégation et au recouplement de données pourtant anonymisées, il reste parfois possible de reconstituer des données personnelles.

B. COMMENT ABORDER CE CADRE D'ANALYSE ?

Deux illustrations concrètes d'application qui s'insèrent dans un tel système d'analyse sont proposées.

- i— **POUBELLES « INTELLIGENTES »** — La gestion intelligente des déchets figure parmi les projets classiques des « *smart cities* », en particulier par le biais technique de capteurs installés sur les bennes. Un capteur de poids peut servir à alerter les services de ramassage lorsqu'il devient nécessaire de récolter les ordures ; le système permet alors de réguler au mieux la tournée des camions dans la ville. La donnée collectée ne concerne que la localisation de la benne et la charge de déchets

qu'elle supporte. Selon la matrice « *Privacy Concerns* » (cf. [VAN ZONEN, 2016] et figure 1.14), il s'agit de données appartenant au quadrant IV, à savoir « *non-personnelles* » et agrégées à des fins de service.

En ajoutant un système de carte individuelle associé à l'ouverture de la benne, la poubelle connectée effectue une translation vers le quadrant II « *Données personnelles à des fins de surveillance* ». La donnée recueillie permet de savoir *qui jette telle quantité de déchets*. Le premier cas ne pose aucun problème direct quant à la protection des données personnelles, contrairement au second.

- ii – POLICE PRÉDICTIVE** — Les systèmes prédictifs de la criminalité se sont développés depuis quelques années. La première catégorie d'algorithme se fonde sur les statistiques relatives à l'historique des lieux et aux types de crimes constatés dans une ville, afin de déterminer les zones dans lesquelles la police doit patrouiller en priorité. On est alors en présence de données « *non-personnelles* » utilisées à des fins de surveillance (quadrant III).

Si, par les statistiques et potentiellement les données des réseaux sociaux, l'algorithme cherche non plus à déterminer le lieu, mais les personnes susceptibles de commettre un délit, les enjeux de protection de la vie privée se déplacent alors vers le quadrant II : « *Données personnelles à des fins de surveillance* ». Dans ce cas, le risque cité par Liesbet VAN ZONEN, est alors que *tous les habitants d'une ville deviennent suspects*.

Dans les faits, une enquête éloquente menée par PROPUBLICA démontre que ce type d'algorithme tendrait à renforcer des préjugés existants — racisme entre autres — et stigmatise sévèrement certains segments de la population.

C. OUTIL PÉDAGOGIQUE ET DE COMMUNICATION

Pensé dans des perspectives pédagogiques ou de communication à destination des collectivités territoriales, ce cadre d'analyse des « *Privacy Concerns* » se révèle un outil intéressant pour quiconque souhaite comprendre ou faire comprendre les impacts en termes de protection de la vie privée à l'occasion de projets de ville numérique ou *smart city*.

La logique est ici comparable aux guides d'*Analyse d'impact relative à la protection des données* (AIPD) ou en anglais *Privacy Impact Assessment* (PIA)³⁷ de la CNIL qui, dans une version plus approfondie et dans un formalisme plus complet, apportent les outils aux responsables de traitement dans leur démarche de mise en conformité.

L'approche de gestion des risques est d'ailleurs au cœur du règlement européen et la sensibilisation en amont à l'impact de nouveaux services numériques sur la vie privée reste la meilleure manière de faire entrer dans les esprits la culture de la protection de la vie privée et des libertés individuelles.

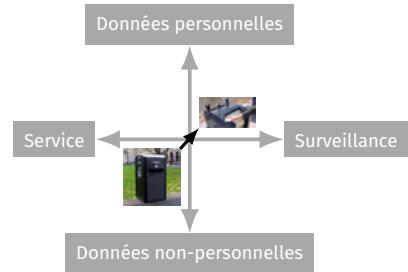


FIGURE 1.14 — Problèmes contrastés de confidentialité pour une gestion intelligente des déchets.

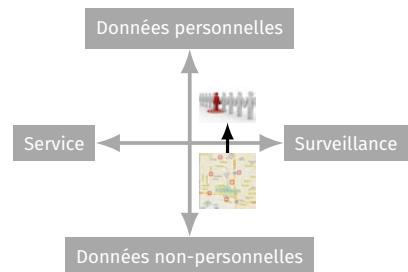


FIGURE 1.15 — Défis contrastés de gouvernance pour un maintien prédictif de l'ordre public.

³⁷. Une réactualisation de ces outils a été opérée par la CNIL début 2018 pour entrer en conformité avec le Règlement général sur la protection des données (RGPD). Une nouvelle version des outils logiciels subséquents a été réalisée, dont une mise à jour récente du 31 octobre 2019.

4.3 Comprendre les logiciels libres

Qu'est-ce qu'un logiciel libre ? Que peut-on réellement en faire et comment l'utiliser en pratique ?

Certes, ce sujet soulève des questions éthiques, philosophiques et, de fait politiques mais, dans le même temps, apporte aux modèles économiques en place des solutions qui se montrent originales, toujours en constante évolution et sources permanentes d'innovation.

4.3.1 Fondements et principes des logiciels libres

Même si l'appellation anglophone désignant les logiciels libres utilise le préfixe « *free* » — pour *free software* —, rien dans la définition du

À PROPOS DES INTERVENANTS

Philippe LHARDY est président de LINUX AZUR, association de promotion des logiciels libres et GNU/LINUX Côte d'Azur. Véronique FRITIÈRE, vice-pdte et secrétaire AFUL a assuré la présentation.

NOTE DE LA RÉDACTION

Les contenus initiaux sont ici fortement amendés et remaniés.



VIDÉO 1.7 — Logiciels libres I.

38. Les logiciels libres n'ont définitivement rien à voir avec les *freewares* et encore moins avec les *sharewares*, même si certains auteurs invitent parfois les usagers à les soutenir par une contribution pécuniaire : temps consacré, site Web et bande passante ont un coût d'investissement.

39. Sous l'influence du *lobby* des éditeurs, un débat encore récent a agité les institutions européennes concernant la « brevetabilité » des logiciels. C'est en sursis, mais déjà chose faite aux États-Unis et au Japon; d'où diverses clauses restrictives d'utilisation de certains logiciels en fonction du lieu géographique.

40. À également discerner d'un programme *open source*, même si nombre d'entre eux sont libres (cf. infra).

41. Dans la grande majorité des langages de programmation, les tableaux sont indexés à partir de la valeur zéro (0). D'où la numération des quatre libertés du logiciel libre : 0, 1, 2, 3. Une autre interprétation relevée tient au fait que le zéro n'a rien d'une nullité, mais s'avère un élément fondamental dans un système binaire.



APRIL : Association pour la promotion et la recherche en informatique libre, acronyme initial abandonné pour le slogan « April - Promouvoir et défendre le logiciel libre ». Association pionnière (1996), elle regroupe entreprises et professionnels, institutions et particuliers. L'APRIL se revendique de l'advocacy et non du lobbying, distinction anglo-saxonne importante ici, s'il en est.

logiciel libre ne définit, ni induit la gratuité. Il s'agit d'une imprécision voire d'une confusion sémantique appartenant à la langue anglaise.

En reprenant un *leitmotiv* traditionnel de la communauté du logiciel libre, *free* n'est pas à prendre au sens de *gratuité*, par exemple d'une bière, mais bien à considérer dans une perspective de *liberté*; libre expression ou libre échange. Cette ambiguïté n'apparaît pas en français.

Si la plupart des logiciels libres connus s'avèrent gratuits, c'est juste une incidence qui correspond à la manière dont leurs auteurs ont décidé³⁸ de les distribuer.

A. DROIT D'AUTEUR ET LIBERTÉ DES UTILISATEURS

Les programmes informatiques — qu'ils soient commerciaux, « propriétaires », « privatifs », « ouvert » ou bien encore « libres » — relèvent tous jusqu'à présent³⁹ du *droit d'auteur*, qui accorde la jouissance d'un monopole : celui d'interdire.

Assez logiquement, la *rédaction* d'un code informatique employant un *langage* de programmation constitue une *œuvre intellectuelle*; peu importe la manière dont il est diffusé, comme exécutable « verrouillé » ou « ouvert » au sens de la mise à disposition de son élaboration.

Ce faisant, le droit d'auteur garantit ainsi qu'il ne soit pas possible de copier un programme pour le donner ni encore moins le vendre, le modifier — ou essayer de le faire — ou l'employer en dehors des clauses stipulées par sa licence (cf. « Logiciels libres en partage », APRIL). Les licences déterminent les droits et les devoirs des utilisateurs :

- une licence « *propriétaire* » signifie la réservation du programme;
- quant à elle, une licence « *libre* » exprime la manière d'organiser la diffusion du programme.

En revanche, le droit d'auteur n'interdit pas de se lancer dans l'écriture d'un nouveau programme aux fonctionnalités similaires, compatible au niveau des formats de communication et de données et, inter-opérable avec le programme original.

Un logiciel libre⁴⁰ respecte la liberté des utilisateurs en répondant à quatre⁴¹ points fondamentaux :

1. liberté 0 — liberté de faire fonctionner ou exécuter le programme sans limitation, objectif particulier ou situation spécifique, pour n'importe quel usage;
2. liberté 1 — liberté d'étudier le fonctionnement intrinsèque du programme, de le modifier et de l'adapter à ses besoins propres;
3. liberté 2 — liberté de redistribuer des copies du logiciel à qui que ce soit, aussi bien en donnant qu'en vendant ces copies;
4. liberté 3 — liberté de publier et diffuser les changements apportés au programme initial.

L'accès au *code source* est une condition nécessaire directement issue des clauses 1 et 3 de la définition du logiciel libre. Le programme se doit d'être *a minima* disponible sous forme de sources éditable. En pratique, il est quasiment tout le temps fournit sous deux formats : code source et exécutable opérationnel — pas forcément pour tous les systèmes d'exploitation — (voir Vid. 1.7).

B. ILLUSTRATIONS ET INITIATIVES

Prenons pour exemple des logiciels largement utilisés, comme FIREFOX. Ce navigateur Web est une solution libre qui a l'avantage de fonctionner sur toutes les plateformes. Cela permet donc d'apprendre et de maîtriser un seul logiciel. Cela garantit en outre que, quel que soit le matériel acheté plus tard, l'investissement fourni dans l'apprentissage de l'outil pourra aisément se valoriser avec la nouvelle configuration.

À noter que cette orientation multiplateforme d'un logiciel transversal de référence relatif d'une fonction particulière n'est pas toujours appréciée de ceux qui souhaitent disposer d'un système exclusivement libre — c'est notamment le cas du « [Contrat social](#) » et des « [principes du logiciel libre](#) » selon DEBIAN, communauté à raison la plus « orthodoxe » de l'écosystème associé au logiciel libre (voir également [DEBIAN Free Software Guidelines](#)).

Si ce positionnement semble clairement celui à adopter et l'objectif à atteindre sur le long terme, il n'en reste pas moins vrai que les applications libres multiplateformes permettent une transition en douceur et « sans douleur » de ces perspectives pour les néophytes et les débutants « insuffisamment formés » (cf. note [45](#)).

À cet effet, on peut mentionner que les partisans de cette approche « *soft power* » se sont manifestés en 1998 par la proposition et la mise en œuvre du concept d'*Open Source*.

En fondant l'*Open Source Initiative*, les promoteurs principaux de cette orientation intermédiaire sont Bruce PERENS et Eric RAYMOND (voir [RAYMOND, 2001]). Cependant, force est de constater qu'un an après ce lancement, Bruce PERENS, ancien *leader* du projet DEBIAN après son fondateur Ian MURDOCK — le « ian » de DEBIAN —, prend ses distances et se retire de cette action à la suite de ce qu'il appelle un « échec de l'*Open Source Initiative* ». Alors, qu'en conclure ?

En outre, certains logiciels commerciaux, d'ordre professionnel spécifique ou par monopole établi des éditeurs, s'imposent aux utilisateurs et induisent de fait un environnement propriétaire. Dans l'attente d'une offre libre [42](#) équivalente, avoir la possibilité d'employer des outils libres hormis cette contrainte constitue indéniablement un « moindre mal ».

Pour d'autres tâches courantes et ne citer que leur association avec les « couteaux suisses » les plus répandus, il existe désormais une foultitude de programmes aux propriétés similaires, à la fois libres et multiplateformes — l'offre devient véritablement immense en se cantonnant aux plateformes libres comme [GNU/LINUX](#) ou [BSD](#) —, tels [43](#) que :

- THUNDERBIRD pour la messagerie électronique;
- APACHE pour les serveurs Web;
- AUDACITY pour la manipulation de l'audionumérique;
- VLC — *VideoLan Client* — pour le maniement des flux vidéo;
- GIMP et INKSCAPE pour l'édition, la création et la gestion des images respectivement matricielles et vectorielles;
- ou encore, non des moindres, pour la bureautique — traitement de texte, tableur, outil de présentation —, la suite logicielle LIBREOFFICE, *alter ego* de son équivalence privative trop connue.

D'ailleurs, en terme de diffusion, la gratuité constatée [44](#) de la plupart des logiciels libres (voir supra), autorise tout un chacun, professionnel autant que particulier, de tester, de choisir et de se former en toute indépendance à la solution qui lui convient le mieux ou bien celle adoptée par son employeur — lorsque la stratégie numérique de l'entreprise ou de l'institution s'est portée [45](#) sur un environnement de travail fondé sur les logiciels libres.

Par ailleurs, leur déploiement à partir d'une seule copie étant autorisée, cela induit une économie d'échelle conséquente car les adaptations et les modifications éventuellement apportées ne sont payées qu'une fois, font évoluer le logiciel et sont profitables pour tous (voir § 4.3.2, « *Modèle économique* »).

Pour les experts, ils ont la possibilité de façonnner, d'améliorer et de compléter à l'envie les programmes publiés sous une licence libre.



[42.](#) Indépendamment des applications expertes, l'exemple le plus marquant pour l'usager lambda est la gestion de fichiers PDF. Certes, la plupart des visionneurs libres offrent les fonctionnalités élémentaires attendues. En revanche, concernant la gestion multimédia ou les formulaires, il reste plus commode d'avoir une licence ACROBAT, laquelle n'est fournie que pour des systèmes d'exploitation privés (le lecteur acroread sous UNIX/LINUX est figé depuis 2013, pour des raisons de sécurité, mais surtout peut-être pour poursuivre la captation des infographistes professionnels avec les solutions d'ADOBÉ et ce, dès leurs études). D'autres exemples peuvent être cités, comme la suite MICROSOFT OFFICE ou l'outil de calcul numérique MATLAB.

[43.](#) Pour disposer d'une liste relativement fournie de logiciels libres, il est possible de se référer au projet Comptoir du libre, service offert par ADULLACT — Association des développeurs et utilisateurs de logiciels libres pour les administrations et les collectivités territoriales.

[44.](#) Pour enfacer le clou, *logiciel libre ne veut pas dire gratuit!* Néanmoins, un accès au code source permet adaptation et, le cas échéant, compilation pour dûment obtenir un binaire adéquat.

[45.](#) À titre d'illustration réussie et d'envergure, la Gendarmerie nationale a engagé sa migration vers les logiciels libres dès 2003. La bascule s'est opérée de manière progressive des applications vers un système d'exploitation s'appuyant sur la distribution UBUNTU : GENDBUNTU (voir retour d'expérience 2014). En 2018, 82% du parc informatique est entièrement libre.

C. HISTORIQUE ET MOUVEMENT SOCIO POLITIQUE

Initialement (décennies 1960-1970), le logiciel se trouve de fait plus ou moins libre. Les programmes sont à cette époque très fortement intriqués au matériel vendu et il est naturel et de coutume de les partager pour améliorer l'utilisation des systèmes informatiques, alors *mainframes* — à savoir, *ordinateur central* ou macro-ordinateur. Une culture de « *hacker*⁴⁶ » se construit petit à petit à partir de ces pratiques d'entraide et constitue le socle de l'informatique libre à venir (voir infra).

À la charnière des années 1970-1980 et l'avènement de l'ordinateur personnel, la frontière entre fournisseurs de matériels et éditeurs de logiciels se dessine distinctement et s'élargit quelque peu : le *no man's land* tacite alors établi est comblé par les acteurs de la programmation et du codage : l'industrie du logiciel est née.

C'est en effet à ce moment que se fait le choix délibéré de protéger le logiciel pour en assurer la vente en tant qu'entité distincte. Par voie de conséquence, les licences commerciales apparaissent, au motif que le logiciel ne peut pas s'utiliser sans conditions préalables, c'est-à-dire : que les sources ne sont pas disponibles, qu'il est interdit de modifier le programme et que seules les sociétés éditrices bénéficient de cette autorisation. Un monopole de quelques éditeurs de logiciels commerciaux encore aujourd'hui bien connus s'est alors installé durablement.

C'est dans un tel contexte et avec le recul, plutôt rapidement, que se dégage une réponse alternative et efficace à cette situation — prompte dans la réaction, mais plus laborieuse dans la mise en œuvre. L'initiative appartient à Richard M. STALLMAN, brillant chercheur américain à la personnalité charismatique ou irritante suivant le point de vue — selon sa « biographie autorisée⁴⁷ » [STALLMAN et al., 2010], possiblement à la limite du syndrome d'ASPERGER — et qui officie alors au sein du laboratoire d'intelligence artificielle AI LAB du MIT — Massachusetts Institute of Technology de Cambridge aux États-Unis.

Richard M. STALLMAN — aussi connu sous le sigle « *rms* » (en minuscules) — introduit ainsi la notion de logiciel libre et les quatre libertés qui lui correspondent énoncées plus haut. Par là-même, il définit également le cadre du mouvement politique et social du logiciel libre.

Chronologiquement — ou désormais historiquement —, STALLMAN lance donc le projet GNU en 1983, suite à une sombre anecdote de bourrage papier d'une imprimante *laser* offerte par XEROX à son laboratoire, dont il ne pouvait pas obtenir le code source⁴⁸ pour en corriger les défaillances (cf. [STALLMAN et al., 2010], pp. 1-15). Ce potin est devenu une légende incontournable dans la communauté du logiciel libre.

Considérant à juste titre que le système d'exploitation est l'élément stratégique crucial, le projet GNU, qui répond à l'acronyme récursif *GNU is Not UNIX*, a pour ambition d'offrir un équivalent libre aux divers types d'*UNIX*⁴⁹ propriétaires du moment.

De 1983 à 1990, les développeurs du projet ont sorti l'essentiel des composantes d'un tel échafaudage logiciel, à l'exception somme toute capitale du noyau. Pour cette raison, à partir de 1990, le défi est pour lors de fournir ce composé majeur sous la dénomination de HURD, relative à un double acronyme récursif exprimant « *Hird of Unix-Replacing Daemons* » puis HIRD, autrement dit, « *Hurd of Interfaces Representing Depth* » : jeu de mot avec « *herd* », qui signifie troupeau en anglais.

L'objectif est de disposer d'un noyau qui respecte les spécifications POSIX — les quatre premières lettres forment l'acronyme de *Portable Operating System Interface*; littéralement, *Interface portable de système d'exploitation*, agrémenté d'un « *X* » qui exprime l'héritage UNIX —, terme suggéré par STALLMAN (encore lui !), faisant à cette époque partie

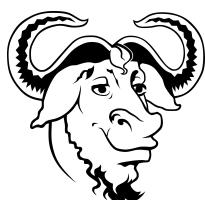
46. Expression à ne pas confondre avec le terme de « *cracker* », agissant sans éthique, se limitant à chercher l'intrusion par tout moyen à disposition (RFC-1983, 1996).



Ruben Rodriguez

Richard Matthew STALLMAN à la conférence annuelle LibrePlanet 2019 organisée par la Free Software Foundation.

47. [STALLMAN et al., 2010] — Richard M. STALLMAN, Sam WILLIAMS et Christophe MASSUTI, 2010 — « *Richard Stallman et la révolution du logiciel libre. Une biographie autorisée* », Accès Libre. FRAMABOOK-EYROLLES, Paris, 324 pages.



48. Pour information, Richard M. STALLMAN est, entre autres, le créateur et parmi les programmeurs principaux de l'éditeur de texte à tout faire *Emacs*, du compilateur *gcc* et du débogueur associé *gdb* (logiciels inclus au projet GNU); excusez du peu !

49. Les systèmes d'exploitation type UNIX sont indissociables du langage de programmation C. Ce dernier, conçu à cette fin par Ken THOMPSON et Dennis RITCHIE en 1972 au sein des laboratoires BELL, est concomitant à la réécriture de leur aïeul MULTICS de 1964, d'abord nommé UNICS.

du comité qui écrit la première version de la norme IEEE 1003 en 1988 — *Institute of Electrical and Electronics Engineers*.

En 1985, le cadre juridique des logiciels libres commence à se structurer conjointement à la création de la Free Software Foundation. Les principes afférents au logiciel libre sont sous-jacents dans les milieux universitaires, mais les contributeurs de GNU, dont STALLMAN lui-même, souhaitent s'assurer que leurs concours au projet restent libres.

Avec la participation d'Eben MOGLEN, alors jeune juriste, sont établies les différentes licences GNU, afin d'asseoir la pérennité des logiciels libres. Ainsi, suite à moult évolutions et reprises, c'est à partir de 1989 que les licences GNU sont érigées. Après trente ans écoulés, on y distingue à ce jour en version 3.0 :

- la GNU General Public License (GPL);
- la GNU Lesser General Public License (LGPL);
- et la GNU Free Documentation License (GFDL), parfois critiquée pour ses imprécisions, en particulier par la communauté DEBIAN.

Pour ce faire, STALLMAN s'appuie, notamment pour ce qui concerne la licence GPL — *General Public License* —, socle et pivot des logiciels libres, sur le concept de *copyleft* — ou gauche d'auteur⁵⁰ — inventé par Don HOPKINS et, l'histoire en témoigne, le popularise largement.

Au-delà des libertés qui caractérisent les logiciels libres, le *copyleft* garantit leur persistance ou leur hérédité. On évoque parfois à son sujet le vocable de licence *virale* — à bien entendu considérer dans une acception positive —, au sens où toute œuvre dérivée de l'œuvre originale doit être placée sous la même licence (cf. § 4.3.2). C'est un peu un principe cousin qui guide certaines licences CREATIVE COMMONS — mais de loin pas toutes! — édictées bien après.

Au sujet de la GNU GPL, Eben MOGLEN formule que c'est la « *création d'un pot commun auquel chacun peut ajouter, mais rien retirer* ».

Lors de ses interventions en France, Richard M. STALLMAN — qui, au passage, s'exprime dans un français impeccable — débute invariablement sa présentation des logiciels libres par : « liberté, égalité, fraternité ». Approche sensée résonner auprès des citoyens de l'hexagone :

- liberté → utiliser, copier, améliorer et diffuser;
- égalité → droits identiques pour tous;
- fraternité → coopérer avec tous et construire ensemble.

Plutôt que de paraphraser un discours, rien de mieux que de se faire une opinion propre en visionnant un de ses exposés, en version ramassée d'une entrevue ou dans le contexte plus étoffé d'une conférence (INTERTICE 2012 — CRDP⁵¹ de Versailles).

D. SYSTÈME D'EXPLOITATION GNU/LINUX

En 1991, un étudiant finlandais du nom de Linus Benedict TORVALDS, inspiré par les travaux — en particulier sur MINIX, système d'exploitation clone d'UNIX, créé à des fins pédagogiques — et ouvrages publiés par Andrew Stuart TANENBAUM⁵² [TANENBAUM, 2008], professeur à l'Université d'Amsterdam, se lance par plaisir et évènements circonstanciés dans l'écriture du noyau d'un système d'exploitation⁵³ pour son ordinateur doté d'un processeur Intel 80386 [TORVALDS et DIAMOND, 2001].

Peu importe les anecdotes⁵⁴ discursives intermédiaires, il se trouve que Linus TORVALDS poste le 25 août 1991 un message fondateur désormais célèbre sur USENET, informant de son travail et appelant à contribution. Ce n'est qu'un peu plus tard, vraiment assuré qu'il en conservera la paternité et la gratuité, qu'il publie son code sous licence GNU GPL.

Contrairement aux idées reçues, Linus B. TORVALDS n'est clairement pas un chantre du logiciel libre. Il voit plutôt avec bienveillance l'*Open*



⁵⁰. Cette appellation de « *copyleft* » se fonde sur un subtil double jeu de mots qui oppose deux sens bien distincts aux mots *right* et *left*, le premier invoque littéralement « droite » et « gauche », le second confronte « le droit » et « l'abandon », signifiant ainsi que l'auteur renonce aux droits sur son travail.

⁵¹. Centre régional de documentation pédagogique, composante réseau CANOPÉ.

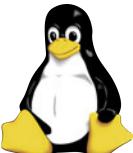
⁵². Ouvrages estimés comme référence en réseaux [TANENBAUM et WETHERALL, 2011] et en systèmes d'exploitation [TANENBAUM, 2008] — Andrew S. TANENBAUM, 2008 — « *Systèmes d'exploitation* », Informatique. PEARSON FRANCE, Paris, 1070 pages.

⁵³. [TORVALDS et DIAMOND, 2001] — Linus B. TORVALDS et David DIAMOND, 2001 — « *Il était une fois LINUX. L'extraordinaire histoire d'une révolution accidentelle* », OSMAN EYROLLES MULTIMÉDIA (OEM), Paris, 300 pages.

⁵⁴. De vifs échanges techniques sur USENET entre L. B. TORVALDS et A. S. TANENBAUM sont restés dans les annales (noyau monolithique versus micro-noyau) avant que ce dernier reconnaîsse sans animosité les qualités de LINUX.



« BEASTY », mascotte des systèmes *BSD*, représente un diablotin. Créeé en 1988 par Marshall K. MCKUSICK, sa popularité vient de la déclinaison de John LASSETER, directeur artistique des studios *PIXAR*.



« Tux », mascotte du noyau *LINUX*, figure un manchot. Rétro-acronyme entre *TorVALDS* et *UNiX*, il est issu d'un concours de 1996 remporté par Larry EWING.



Source. Ses motivations sont essentiellement tournées vers les performances techniques; il n'a cure des conséquences sociopolitiques.

Dans le même temps, faute de volontaires et, peut-être provenant d'ambitions trop importantes, la mise en œuvre du noyau HURD patine.

Par ailleurs, les batailles juridiques entre *AT&T* et *BSD — Berkeley Software Distribution*, premier système libre de type *UNIX* à compter de juin 1989 — freinent considérablement le développement de *BSD* et éloignent les utilisateurs par crainte du paiement de redevances si *AT&T* l'emporte. Le procès se clôture début 1994 et donne raison à *BSD*.

En parallèle, les « *hackers* » s'emparent de *LINUX* et apportent de multiples contributions, d'autant plus avec sa publication sous *GPL* en 1992. Associé aux outils *GNU*, c'est donc le noyau *LINUX* qui s'impose pour offrir un système d'exploitation entièrement libre. À l'été 1993, les premières distributions toujours maintenues aujourd'hui apparaissent, à l'instar de *SLACKWARE* et... De *DEBIAN*, aux fondements des distributions à présent parmi les plus populaires, telles *UBUNTU* ou *LINUX MINT*.

Il existe pléthore de distributions, plus ou moins différencierées ou spécialisées (voir *DISTROWATCH*). Hormis celles déjà citées et leurs dérivées de tout genre, on peut mentionner celles dont l'origine remonte à la décennie 1990, essentiellement *RED HAT LINUX* aux États-Unis et *SUSE* en Allemagne, toutes deux créées en 1994 et s'étant assez rapidement orientées — à partir de 2002 — en tant que distributions commerciales à destination des entreprises.

Initialement, au-delà des services vendus et outils spécifiques proposés, les différences entre distributions se caractérisent par leur gestionnaire de paquets logiciels et leur environnement de travail par défaut — principalement *GNOME/GTK+* versus *KDE/QT* —, bien que ce dernier puisse se changer relativement facilement.

Les gestionnaires de paquets sont d'une part, *Red Hat Package Manager* (*RPM*) pour la distribution éponyme et ses dérivées — dont sa version communautaire *FEDORA* ou, en France, *MANDRAKE*, puis *MANDRAVA/OPENMANDRIVA* et actuellement *MAGEIA* — ou *ZYpp/YaST* (fondé sur *RPM*) pour *SUSE* et désormais *OPENSUSE* et, d'autre part, *Debian Package Manager* (*DPM*) pour les distributions s'appuyant sur *DEBIAN*.

Enfin, au début du millénaire (2002), des distributions voient le jour dont l'installation ne s'opère pas au moyen de fichiers binaires sur support amovible (disquette, *Compact Disk* et autre clef USB) mais directement par Internet et compilation des sources sur la machine de l'utilisateur. Elles s'adressent aux usagers avancés qui savent ce qu'ils font. La méthode de mise à jour fait appel au *rolling release*, à savoir « publication continue ». *ARCH LINUX* et *GENTOO* fonctionnent sur ce principe.

De nos jours, *GNU/LINUX*, à l'exception des ordinateurs *APPLE* fondés sur un OS à base *BSD* (question de licence), surclasse et remplace quasiment tous les systèmes *UNIX*, sur toutes les plateformes et infrastructures — modèle de processeur, serveur Web, ferme ou *cluster* de calcul, système embarqué, station de travail, etc.

4.3.2 Modèle économique

Le logiciel libre ne définit pas de modèle économique en lui-même. Cela n'adresse donc pas directement une problématique économique mais, en amont, relève plutôt d'une discussion éthique.

Toutefois, le logiciel libre est désormais tellement développé, facile d'accès et pratique d'utilisation que de nombreuses sociétés commerciales et institutions y font appel en masse. Qu'en est-il alors ?

NOTE DE LA RÉDACTION

Les contenus initiaux sont ici fortement amendés et remaniés.

A. LICENCE LIBRE ET PRODUIT COMMERCIAL

Deux orientations essentielles liées aux logiciels libres coexistent ; elles établissent par essence une fracture importante en termes d'approche et de finalité, à la fois philosophique, politique et économique.

Une première catégorie d'applications s'appuie sur le concept du *copyleft* — typiquement représentée par la licence GNU GPL, *General Public License*. Comme évoqué précédemment (voir page 57), le *copyleft* induit que le logiciel conserve sa qualité de logiciel libre — suggéré par l'attribut de « licence virale », encore une fois à dûment apprécier dans une acception positive. En conséquence, sa redistribution impose alors sa nature libre aux codes qui en sont dérivés.

Par opposition, cela définit une seconde classe de programmes informatiques, pour laquelle le code source peut se voir exploiter à des fins commerciales sans être pour autant divulgué, notamment pour ce qui concerne les adaptations apportées.

En pratique, les licences BSD ou MIT sont bien plus permissives — au sens d'une exploitation privée — que les licences GNU de la FSF. Elles autorisent la reprise d'un code informatique pour se l'approprier, juste en le mentionnant lors de sa modification et de sa redistribution. La façon dont ce code est intégré dans une application propriétaire n'oblige en rien la diffusion des sources.

Ces codes sont très couramment agrégés aux applications commerciales. Pour illustration, les systèmes d'exploitation diffusés par APPLE se fondent sur des BSD déjà utilisés pour la station NEXT — NEXTSTEP — et modifiés pour obtenir le système MAC OS X, en toute opacité. Seules certaines parties du code sont libres sous licence *Apple Public Source License* (cf. DARWIN), pour essayer de « raccrocher » certains hackers au développement du système. Si ça fonctionne, pourquoi pas ?

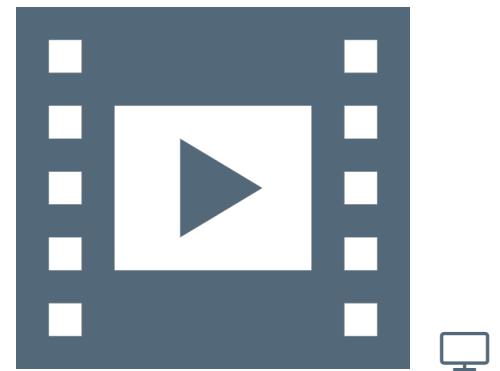
Ainsi, ces permissivités permettent à certaines sociétés de faire des économies d'échelle en recherche et développement sur la base d'un existant déjà performant et optimisé, pour se concentrer sur une plus-value lucrative — réelle ? Cela donne des plateformes générales de programmation à partir desquelles on poursuit le travail, sans aucun retour coopératif. Déontologiquement, chacun est juge de ces usages.

B. MODES D'APPLICATION ET D'UTILISATION

Le mouvement du « Libre » remet singulièrement en question la notion juridique traditionnelle de la *propriété intellectuelle* (droit d'auteur et dérivés) et de la *propriété industrielle*⁵⁵ (brevets, dessins, modèles). Il infère une contestation des canons établis depuis le XVIII^e siècle pour proposer des paradigmes alternatifs tenant compte du regard nouveau porté à la création et à l'innovation, issus des bouleversements de ces dernières décennies : révolution numérique, pluralité des modes d'élaboration ou mondialisation — pour un panorama complet on peut avec profit s'attacher à la lecture de l'essai⁵⁶ de Benjamin JEAN [JEAN, 2011].

Martelons une fois encore que : « *libre ne veut pas dire gratuit* ». Du point de vue du quidam moyen, cette confusion provient probablement du fait que, dans l'économie du numérique, le coût marginal de la diffusion d'un produit tend vers zéro ou s'avère négligeable. Néanmoins, en amont, un logiciel ou ouvrage d'autre nature est conçu et résulte de la mise en œuvre d'un ou désormais d'une foultitude d'acteurs humains. Pour des projets d'envergure, le travail fourni se comptabilise parfois en millions de jours-personnes et bien plus encore.

Quels schémas sous-tendent alors leur participation, voire leur rétribution ? Pourquoi, mais également comment, vis-à-vis des réticences des informaticiens « à l'ancienne⁵⁷ » conditionnés par les « majors »



VIDÉO 1.8 — Logiciels libres II.

55. Les licences libres s'appliquent aussi au matériel (*free hardware*), par diffusion de plans et autres spécifications. C'est par exemple le cas de la carte électronique ARDUINO déjà mentionnée. Cela s'accentue depuis quelques années avec l'essor des mouvements du « *Make* », du « *Do It Yourself* » et des FABLABS.

56. [JEAN, 2011] — Benjamin JEAN, 2011 — « *Option libre. Du bon usage des licences libres* », FramaSoft, Paris, 307 pages.

57. Indépendants, employés de TPE ou de grand compte qui, emportés malgré eux dans une ronde pernicieuse, accumulent les certifications WINDOWS et autres Cisco pour justifier de leur capacité à intervenir chez leurs clients et, de fait, les maintenir en « état de siège ».

^{58.} Coté certification de compétences, les logiciels libres sont loin d'être en reste. Il existe effectivement des validations d'acquis d'expérience et agréments associés : LPIC1 — Administrateur LINUX —, LPIC2 — Ingénieur LINUX —, LPIC3 — Environnement mixte. La préparation de ces certifications peut s'envisager au moyen des ouvrages [ROAUT, 2011] et [BOBILLIER, 2012].

^{59.} Les entreprises en informatique, initialement dénommées « Société de service et d'ingénierie en informatique » (SsII), à l'impulsion du SYNTEC NUMÉRIQUE ont été amenées, suite aux évolutions constatées, à devenir en 2013 « Entreprise de service en numérique » (ESN). Par répercussion, les acteurs privés du libre alors reconnus sous l'appellation « Société de service en logiciels libres » (SsLL), ont évolué comme « Entreprise du numérique libre » (ENL).

^{60.} [ÉLIE, 2009] — François ÉLIE, 2009 — « Économie du logiciel libre », Accès Libre. Eyrolles, Paris, 186 pages.

NOTE DE LA RÉDACTION

Au sujet des incidences potentielles de la révolution numérique, du « Libre » ou encore du « Faire/Make » sur le monde du travail, la rédaction invite à examiner les analyses du sociologue Michel LALLEMENT — « *L'âge du faire. Hacking, travail, anarchie* » [LALLEMENT, 2015] — et à explorer la pensée du philosophe Bernard STIEGLER — « *L'emploi est mort, vive le travail! Entretien avec Ariel Kyrou* » [STIEGLER, 2015a] — « *La Société automatique. L'avenir du travail* » [STIEGLER, 2015b]. Pour illustration, visionner :

- ▶ Économie participative, 2013;
- ▶ Société automatique, 2014;
- ▶ Révolution numérique, 2016.

de l'édition logicielle propriétaire, le « Libre » a-t-il ainsi pu se tailler une telle place au sein d'une économie de marché de prime abord étriquée face à ces concepts novateurs ? Quels modèles, ou plutôt quelles pratiques, ont permis ce changement⁵⁸ de paradigme ? Comment dûment appréhender et convenablement interpréter le phénomène assez remarquable des contributions apportées aux logiciels libres ? Celles-ci sont-elles prosaïquement désintéressées ? Quels en sont les fondements sous-jacents ? À l'évidence, plusieurs schémas non exclusifs sont bien entendu à considérer.

Si le particulier « éclairé » ou bien l'universitaire « averti » — l'un comme l'autre, parfois à l'origine d'une avancée significative — en bénéfice par contre-coup participatif, d'un point de vue macroéconomique, le maître-mot semble clairement être *le service*. Pour mémoire, les distributions RED HAT ENTERPRISE LINUX et SUSE se sont délibérément et officiellement orientées sur la cible des entreprises dès l'année 2002, en vendant explicitement des services au-delà de leurs offres logicielles.

Le service et l'accompagnement technique représentent la première marche sur laquelle les dites « Entreprises du numérique libre⁵⁹ » s'appuient. Conseil, documentation, formation, adaptation, mise en œuvre et déploiement sont sans équivoque leur fonds de commerce. Sur ce volet, on peut citer l'exemple des sociétés françaises LINAGORA et SMILE pour ce qui concerne les solutions « clefs en main » envers les entreprises, ainsi que, dans une bien moindre mesure, FREE pour les fournisseurs d'accès Internet (FAI) ou OVH pour l'hébergement de sites Web.

Qu'en est-il d'essayer de conceptualiser cette mosaïque des possibles ? Par son opuscule, François ÉLIE tente différentes approches de la modélisation économique⁶⁰ des logiciels libres [ÉLIE, 2009]. À tort ou à raison, à partir du concept de « métastabilité », son propos a le mérite de proposer cinq catégories de modèles, qualifiées de stables ou bien d'instables (repris tels quels en table 1.1), à savoir :

- modèle communautaire pur, à l'instar de DEBIAN ou MOZILLA;
- modèle hybride communauté-entreprise, à l'exemple de MySQL;
- modèle de mutualisation par l'offre, pour illustration, les fondations APACHE et ECLIPSE;
- modèle éditeur Open Source, à l'image d'entreprises ou institutions qui confient leurs adaptations et développements logiciels à des SsLL/ENL possédant de solides solutions libres — on peut citer en France LOGILAB, notamment pour l'expertise en PYTHON;
- modèle de mutualisation par la demande, liée à un collectif de clients comme certaines communes, institutions ou entreprises abondant chacune d'une brique supplémentaire à l'édifice.

Les trois archétypes étiquetés comme stables sont les modèles *communautaire*, de la mutualisation *par l'offre* (coopération d'industriels) et *par la demande* (coopération entre clients); chacun d'entre eux renvoie à des communautés. A contrario, les modèles *hybride* et *éditeur* sont des approches de transition, *a priori* instables.

Pour à la fois élargir le débat et concrètement bénéficier d'avis techniques de développeurs et de juristes sur ces questions — historique, culture et mouvement du « Libre », cadre législatif, usage des licences et production de logiciel —, on ne peut que vraiment conseiller la lecture des différents essais publiés par FRAMABOOK sur le sujet : « *Histoires et cultures du Libre. Des logiciels partagés aux licences échangées* » [PALOQUE-BERGES et al., 2013], « *Libres conseils. Ce que nous aurions aimé savoir avant de commencer* » [PINTSCHER et al., 2013], « *Option libre. Du bon usage des licences libres* » [JEAN, 2011], « *Produire du logiciel libre* » [FOGEL, 2012] (dernière version en anglais).

TABLE 1.1 — Modèles économiques associés aux logiciels libres — d'après [ÉLIE, 2009], p. 45.

Modèles économiques des logiciels libres				
Type	Motivations	Segments	Modèle	Avenir
Communautaire	Plaisir, intérêts indirects	Applications réseau, niches diverses	Bénévolat, R&D ouverte	Mathématiques
Hybride communauté-entreprise	Cesser de manger des pizzas la nuit	Applications réseau, et middleware	Vous travaillez gratuitement, je touche l'argent	Les versions Community avancent plus vite
Mutualisation par l'offre	Économies d'échelle	Middleware	Investissement et retour sur investissement	Solide si soutenu par les intégrateurs et constructeurs
Éditeur Open Source	Travailler avec un club d'utilisateurs	Middleware, applications métiers complexes	Investissement et retour sur investissement	Tôt ou tard rattrapé par la concurrence
Mutualisation par la demande	Séparation entre solution et prestation	Applications métiers	Investissement coopératif	En émergence partout chez les acteurs publics

Par suite et à titre d'illustration, on peut donc objectivement mentionner deux modèles économiques courants, stable puis instable selon la taxonomie⁶¹ établie par François ÉLIE :

1. un premier canon résulte d'un groupement d'intérêt commun autour d'un produit « phare », souvent sous la forme d'une fondation — APACHE, MOZILLA, LIBREOFFICE, etc. Dans cette mutualisation par l'offre, les entreprises — parmi les plus importantes du secteur, y compris les GAFAM⁶² —, soit financent directement le projet, soit mettent à disposition certains personnels de leur effectif pour apporter contributions et maintenance. Beaucoup de développeurs sont ainsi salariés pour travailler sur du logiciel libre. Cela profite à la communauté générale, voire à l'humanité, en co-opérant à un pot commun de connaissances et de logiciels ;
2. une autre approche largement usitée fait appel à ce qu'on nomme le *dual licensing* ou licence double — voire multiple. Il s'agit de publier le logiciel selon deux orientations : une version libre ou Open Source dite « community » et une seconde désignée comme « corporate ». Cette dernière est vendue avec du service « clefs en main » et profite des évolutions et innovations potentielles apportées par la communauté sur la première. Ce principe de fonctionnement peut, de fait, se montrer assez aléatoire. MySQL⁶³ est diffusé de cette manière. Depuis le rachat par SUN puis par ORACLE et afin de garantir son évolution sous statut de logiciel libre, cela a amené les concepteurs initiaux de MySQL à créer une fondation — que même MICROSOFT soutient ! — et à implémenter un « fork » intitulé MARIADB pour éditer tout le code en libre sous GNU GPL, GNU LGPL et BSD (cf. encart « Licence libre en pratique » page 62).

Enfin, une autre manière d'opérer, parfois insidieuse, est d'offrir des services prétendument gratuits au sein des « nuages ». Certes l'accès à ces services est sans rétribution péculiaire, mais l'intérêt marchand est en données personnelles collectées pour générer de la publicité ou se revendre à des acteurs tiers ; c'est, entre autres, le fonctionnement de GOOGLE. Les utilisateurs finaux utilisent ainsi un service fondé sur du logiciel libre, mais n'ont pas directement recours à celui-ci.

Ce type de services, hormis des GAFAM omnipotents, peut tout à fait se justifier comme payant si la plus-value mérite rétribution. Commer-

61. Taxonomie (déf. CNRTL) : A. Science des lois et principes de la classification des organismes vivants; par extension, science de la classification. B. Classification d'éléments; suite d'éléments formant des listes qui concernent un domaine, une science.

62. GAFA ou GAFAM : GOOGLE, APPLE, FACEBOOK, AMAZON et MICROSOFT.



63. À défaut de MySQL, on peut avantagéusement se tourner vers POSTGRESQL, sous licence cousine de BSD ou MIT. Cependant, MySQL est devenu au fil du temps un standard du Web, en particulier pour la plupart des CMS — Content Management System. Mais surtout, en aval, les offres d'hébergement mutualisé ne proposent que le moteur de bases de données MySQL (cf. OVH).

LICENCE LIBRE EN PRATIQUE

En amont ou à l'issue d'un projet libre, quelle(s) licence(s) choisir ? Trois principales possibilités sont à disposition :

- ▶ **Projet entièrement libre** — Afin de garantir que les travaux dérivés resteront libres, la diffusion est à réaliser sous GNU GPL, c'est limpide.
- ▶ **Contribution de « référence »** — Si le code a vocation d'être un standard dans un cadre libre comme propriétaire, la licence appropriée est la GNU LGPL. Cela s'applique principalement aux bibliothèques. Le but est donc d'offrir un code libre à la place de solutions privatives qui existent, mais ne sont pas accessibles. Il en est ainsi de la bibliothèque GNU C de même que nombre de bibliothèques graphiques comme : [PYGOBJECT](#) (anciennement PYGTK), [PySide](#) (Qt) ou [WxWIDGETS](#).
- ▶ **Application en « bien commun »** — Ce cas a trait aux [nouvelles](#) (après 1999) licences de type BSD ou MIT. Compatibles avec les licences GNU (l'inverse n'est pas vrai), elles sont libres, laxistes, sans copyleft. Elles s'emploient en situation libre ou propriétaire, pour tout ou partie de code, uniquement par mention de leur auteur/éditeur.

D'autres circonstances invitent à utiliser les licences [APACHE 2.0](#) ou [GNU AFFERO](#), mais le portrait général du choix des licences libres est brossé.

cial ou non, tout repose alors sur la confiance que l'utilisateur accorde à chaque prestataire auquel il fait appel.

Cette dernière approche ne se montre viable que si les services apportés sont référents dans leur domaine d'application, voire singuliers. D'où, indubitablement, à court ou moyen termes, une situation de monopole si l'avance constatée perdure. Par suite, on assiste bien souvent à une course effrénée à « l'innovation », uniquement motivée par le gain et le marché, mais non par l'amélioration des intérêts véritables de l'utilisateur ou de la connaissance technoscientifique sous-jacente.

Il existe certainement des contextes encore différents et, probablement, beaucoup d'autres modèles économiques restent à inventer. Cela se trouve néanmoins une affaire complexe à appréhender en pratique.

GLOSSAIRE CONTEXTUEL

CODE SOURCE — Le code source d'un programme est une suite d'instructions logiques écrite par un informaticien — aussi nommé programmeur ou développeur — dans un langage de programmation. Autrement dit, c'est un fichier éditable, aux instructions lisibles et compréhensibles par un être humain à condition qu'il en maîtrise la signification.

EXÉCUTABLE — Se dit de l'état dans lequel se trouve une tâche après son activation et avant qu'elle ne soit finie.

Par extension, cela désigne également le nom du fichier d'un programme — dits de *script* pour les langages interprétés (BASH, PERL, PYTHON, etc. : traduction « à la volée » en langage machine) ou comme résultat d'une *phase de compilation* (traduction du langage de programmation en langage machine) pour les langages dits *compilés* (FORTRAN, PASCAL, C/C++, etc.) — prêt à être lancé pour réaliser les tâches qui lui incombent.

BSD — BSD est l'acronyme de *Berkeley Software Distribution*, [suite logicielle](#) provenant de l'Université de Californie située à Berkeley. À l'*origine*, il s'agit d'extensions apportées au système d'exploitation UNIX AT&T. Plusieurs systèmes *Open Source* sont fondés sur cette base, en incluant des éléments issus d'autres projets, y compris du projet GNU : un noyau BSD, la bibliothèque C BSD — différente de celle du projet GNU —, des utilitaires — interpréteur de commandes, gestionnaire de fichiers, compilateurs et éditeurs de liens —, le système X Window pour l'affichage graphique — BSD ne définit pas de « bureau graphique » tels que [GNOME](#) ou [KDE](#) —, etc. D'abord principalement dévolus aux serveurs — les couches réseaux des BSD ont été portées dans bien d'autres systèmes —, les BSD s'utilisent de nos jours aussi bien comme station de travail. On distingue essentiellement les distributions [FREEBSD](#), [NETBSD](#) et [OPENBSD](#).

GNU/LINUX — GNU/LINUX ou bien, par raccourci innapproprié LINUX, est un système d'exploitation entièrement libre, produit d'un grand nombre d'entreprises et de bénévoles qui contribuent à son élaboration. Il est disponible sous la forme d'une myriade de *distributions* généralistes ou spécialisées (assemblage cohérent de multiples logiciels libres — noyau LINUX et outils GNU — pour fournir un système d'exploitation opérationnel), telles que : DEBIAN, UBUNTU, LINUX MINT, RED HAT, SUSE, RASPIAN, etc.

LINUX — LINUX est le noyau du système d'exploitation GNU/LINUX. Le noyau est le cœur du système, c'est lui qui s'occupe d'ordonner les tâches et de fournir aux logiciels applicatifs une interface de programmation pour utiliser le matériel au moyen

de modules d'extensions spécifiques nommés « pilotes » — ou « drivers » en anglais. LINUX, contraction entre Linus et UNIX, est initié par Linus TORVALDS en août 1991.

GNU — GNU est l'acronyme récursif de *GNU's Not UNIX*. Cela qualifie un ensemble complémentaire de programmes utilitaires libres rattachés aux noyaux des systèmes d'exploitation de type UNIX. L'ambition initiale du projet était de développer un système d'exploitation complet en y adjointant son propre noyau connu sous le nom de HURD, double acronyme récursif de *Hird of Unix-Replacing Daemons* (littéralement « *Multitude de démons^a remplaçant Unix* ») puis HIRD, autrement dit, « *Hurd of Interfaces Representing Depth* » : jeu de mot avec « *herd* », qui signifie *troupeau^b* en anglais. Cependant, la montée en charge remarquable de l'opérationnalité de LINUX de 1991 à 1996, associée à sa licence de publication GPL en 1992, l'a imposé de fait comme noyau du projet GNU ; d'où la double affiliation de dénomination ces systèmes en tant que GNU/LINUX.

UNIX — Système d'exploitation multitâche et multi-utilisateur, UNIX est créé en 1969 par Kenneth Thompson. Il repose sur un interpréteur ou superviseur (le shell) et de nombreux petits utilitaires, accomplissant chacun une action spécifique, commutables entre eux (mécanisme de « redirection ») et appelés depuis la ligne de commande. LINUX ou BSD — « *Berkeley Software Distribution* » — sont des dérivés libres et ouverts de type UNIX.

APACHE — APACHE est un serveur Web sous licence libre parmi les plus répandus sur Internet. Il s'agit d'une application fonctionnant initialement sur les systèmes d'exploitation de type UNIX, mais désormais porté sur tous les OS majeurs du marché.

FIREFOX — Firefox, développé par la Mozilla Foundation, est un des logiciels libres multiplateformes les plus répandus et des plus performants pour la navigation sur le Web.

LIBREOFFICE — LIBREOFFICE est une suite bureautique libre complète, dérivée du projet OpenOffice.org, créée et gérée par THE DOCUMENT FOUNDATION suite au rachat en 2010 de SUN MICROSYSTEMS — alors détenteur de la marque — par ORACLE.

MySQL — MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde.

VLC — VLC MEDIA PLAYER est un lecteur multimédia libre. Ce logiciel est multiplateforme et distribué sous licence GNU GPL. Il intègre les codecs nécessaires à la lecture de la plupart des formats audio et vidéo. De plus, le lecteur est capable de lire un grand nombre de flux réseaux.

PHP — PHP — Hypertext Preprocessor, acronyme auto-référentiel — est un langage de script avant tout utilisé pour produire des pages Web dynamiques via serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

RICHARD M. STALLMAN — Richard STALLMAN est à l'origine un brillant programmeur issu du MIT (entre autres, l'éditeur EMACS). Initiateur du mouvement du logiciel libre, il lance en 1983 le projet GNU, *GNU IS NOT UNIX* — acronyme récursif —, en fondant les bases juridiques du projet sur la Licence publique générale (*General public licence* — GPL) initialement prévue pour les codes de logiciels et la Licence publique générale restreinte (*Lesser general public licence* — LGPL) pour les bibliothèques

a. Les « daemons » sont des programmes qui tournent en tâche de fond et qui apportent des services de gestion spécifique au système d'exploitation.

b. Bien sûr, sous entendu de gnou/gnu.

associées. Bien que dévolues au domaine des logiciels, voire des matériels informatiques, ces licences peuvent tout aussi bien s'appliquer à toute œuvre relevant du droit d'auteur, en parallèle ou à l'instar des licences *Creative Commons*.

COPYLEFT — Le concept de *Copyleft* est une méthode générale pour rendre *libre* un programme — ou toute œuvre relevant du droit d'auteur —, imposant à toutes les versions modifiées ou étendues de l'œuvre d'être libres également : en quelque sorte c'est une, voire la première, « contagion virale positive », au sens du partage de connaissances. Son invention est celle de **Don HOPKINS**, largement popularisée par Richard STALLMAN dans le cadre du projet **GNU**.

LINUS B. TORVALDS — **Linus TORVALDS** est un informaticien d'origine finlandaise reconnu pour avoir créé en 1991 le *noyau LINUX*. Depuis, il continue d'en diriger le *développement* où, à l'instar de **Guido van Rossum** pour **PYTHON**, de **Larry Wall** pour **PERL** et de **Mark Shuttleworth** pour **UBUNTU**, il en est considéré comme le « *dictateur bienveillant* ».

4.4 Logiciel libre et ouvert : révolution ou évolution ?

NOTE DE LA RÉDACTION

Texte rédigé par Gérard GIRAUDON et publié sur *Interstices* — revue en ligne de culture scientifique du numérique — le 16 mai 2006.

Une question fondamentale de la communauté scientifique et, plus particulièrement, pour quelqu'un qui s'occupe de réussir le transfert de ses résultats, est de comprendre, au-delà des clivages idéologiques, en quoi la dynamique du logiciel libre et ouvert favorise la création d'un dialogue permanent entre recherche, industrie et société.

A. PROBLÉMATIQUE DU LOGICIEL LIBRE ET OUVERT

Quelques mots d'histoire permettront certainement de mieux comprendre l'origine de la problématique.

Au début, jusque dans les années 1960, le monde de l'informatique était peuplé de machines, des calculateurs lourds et peu rapides pour lesquels il fallait aménager des « cavernes » adaptées. L'informatique signifiait alors lampes, circuits, fils et transistors.

Bien sûr le logiciel existait, mais n'avait pas de personnalité propre ; d'ailleurs, les constructeurs ne facturaient pas le logiciel livré avec la machine. En ces temps-là, ils régnaient en maîtres sur la planète informatique, dans un écosystème peuplé de tribus quasiment incapables d'échanger des données ou du logiciel si elles n'étaient pas marquées du même totem vénéré. Échange, communication, interopérabilité en dehors de sa tribu étaient des mots vides de sens et la seule solution semblait le monopole du dominant.

Puis sont arrivés, durant la décennie des années 1970, deux projets révolutionnaires. Le premier est celui d'un réseau informatique longue distance, robuste et pouvant connecter tout type de machine ; le second est celui d'un système d'exploitation (abrégé OS pour « *Operating System* » en anglais) indépendant du matériel.

Le premier donna naissance à la révolution Internet, avec comme premières applications le courrier électronique ou « *mail* » (**sendmail**) et le transfert de fichier (**ftp**) ; le second fut l'aventure UNIX. Cette genèse est très bien décrite par Diane REVILLARD dans son livre blanc « *Organisations et logiciels libres* » [REVILLARD, 2005].

Ainsi, on s'aperçoit rétrospectivement que la réussite de tels projets de développement logiciel tenait au fait qu'ils étaient « *Open Source* ». Cette dénomination n'existe pas à l'époque et encore moins celle de logiciel libre. Comme Monsieur JOURDAIN, les chercheurs s'adonnaient à



l'*open source* sans le savoir, simplement parce que c'était le mode naturel pour partager un projet de développement de code dans la sphère académique. On remarquera bien entendu qu'il y en a eu d'autres, par exemple les débuts du projet MATLAB.

De ces aventures, restent l'exemplarité, la compréhension des mécanismes constitutifs de l'édition d'un programme informatique et une licence — BSD, la première du monde du logiciel libre et ouvert.

i- **FACE AU POUVOIR DES ÉDITEURS** — Au début des années 1980, la diffusion en masse de l'informatique auprès de cibles aux profils non techniques et la prise de conscience de la valeur intrinsèque du logiciel en tant que tel, se sont traduites par la montée en puissance d'une nouvelle espèce d'acteurs : les éditeurs de logiciel. Ceux-ci avaient l'ambition de devenir les acteurs dominants avec, en conséquence, une protection du savoir et du savoir-faire via la vieille technique du secret.

Il a fallu attendre 1984 pour qu'en réponse à la fermeture et à la réappropriation des codes sources, possibles avec la licence BSD, Richard STALLMAN, chercheur en informatique du MIT, lance avec la licence GNU GPL l'aventure du « *free software* » : logiciel libre en français.

Est-ce à dire que le code source d'un logiciel diffusé en licence BSD était moins libre ? Non, à l'évidence, mais on voit immédiatement qu'au-delà de l'accès au code source, ce sont les droits et les obligations que confère la licence avec laquelle le programme est publié qui font la différence d'appréciation de cette liberté. La conséquence directe a été que les informaticiens fervents partisans du logiciel libre et de l'*open source* se sont mis à mieux comprendre le droit de la propriété intellectuelle et, notamment en France, le droit de la propriété littéraire et artistique auquel est rattaché le logiciel... Et même, pour certains, jusqu'à en devenir des spécialistes.

ii- **QUELLES LIBERTÉS ?** — À partir de la fin des années 1990, bien que partageant un socle commun de principes, deux courants de pensée du libre vont coexister — « Libre » versus « Open Source ». Ces préceptes, transcrits dans la licence, garantissent un certain nombre de libertés aux utilisateurs du logiciel, dont la première est celle de pouvoir accéder au code source afin d'en étudier son fonctionnement et de l'adapter à leurs besoins. Viennent ensuite la liberté de l'utiliser et de l'exécuter pour quelque usage, la liberté d'en redistribuer des copies puis, enfin, la liberté de l'améliorer et de rendre publiques les modifications, de telle sorte que la communauté toute entière en bénéficie.

En restant très schématique, la différence entre les courants logiciel « libre » et « *open source* » tient essentiellement dans les acceptations du mot « liberté » qui, dans la GNU GPL se traduit parfois par obligation de redistribution. Pour les tenants de l'approche lancée par R. STALLMAN, il ne peut y avoir, en dehors des logiciels diffusés sous GNU GPL, d'autres « logiciels libres », traduction de « *free software* ».

En l'occurrence, l'expression française s'avère plus précise dans l'esprit que l'expression anglaise, puisque que *free* peut aussi se traduire par gratuit... Or l'utilisation de la GNU GPL n'interdit pas une diffusion rémunératrice du logiciel. Pour d'autres (voir le site de l'*Open Source Initiative*), la dynamique du logiciel libre ne peut se réduire à la seule utilisation de la licence GNU GPL... La preuve en étant que le mouvement a été lancé avec l'utilisation de la BSD. Ce point de vue est celui de ceux qui parlent de logiciel « *Open Source* », sans qu'une traduction se soit imposée en français, même si l'on pourrait parfaitement parler de *logiciel ouvert*.

Pour plus de détails, renvoi est fait à lecture du [livre blanc](#) de Diane REVILLARD ou aux définitions sur WIKIPÉDIA, logiciel libre comme *open*

LOGICIEL LIBRE



© François Cointe

source. Il semble toutefois que le plus simple serait de parler de « *logiciel libre et ouvert* » et rapporter les courants de pensée aux licences utilisées. Néanmoins, il est possible de schématiquement les ramener à l'analyse d'un tableau à quatre entrées : diffusion virale ou non, liberté héréditaire ou non (ce tableau a été établi par un groupe de travail interne à l'INRIA en 2002). Le choix d'une licence est cependant primordial, son évocation est reprise par la suite (cf. infra).

iii – CRÉER UNE DYNAMIQUE — Comment le logiciel libre et ouvert est-il une révolution ou une évolution pour favoriser le dialogue entre recherche, industrie et société; en quoi cela facilite-t-il la transformation d'une part, d'une expression des besoins socio-économiques en problématiques scientifiques et, d'autre part, de résultats technologiques en produits ou solutions pour les entreprises et les citoyens ?

Au XIX^e siècle et durant près d'un siècle et demi, la révolution industrielle s'est majoritairement construite sur des « secrets » de fabrication et des brevets donnant un droit de monopole. Jusqu'à présent, l'industrie de l'informatique s'est fondée sur les mêmes principes. Parce que l'informatique est tout à la fois une science, une technologie et une industrie, le passage à l'ère numérique et l'entrée de nos civilisations dans la société de l'information donne à l'informatique et, surtout au logiciel, une importance toute particulière.

Selon les tenants de la GNU GPL, l'enjeu réel du logiciel libre s'avère avant tout social, car le mouvement lancé par Richard STALLMAN prend racine dans un idéal qui postule comme prérequis la liberté et le caractère universel du savoir et de l'information. D'autres courants issus de l'*open source* sont plus pragmatiques, moins idéologiques; ils considèrent que le logiciel libre et ouvert est un nouveau mode de relations entre les entreprises, les clients, les citoyens et les chercheurs.

Mais au-delà de ces distinctions, il est essentiel de conserver à l'esprit qu'en informatique, les notions de communauté d'utilisateurs, de partage de l'information et d'ouverture des codes sources représentent une longue tradition.

B. LOGICIEL LIBRE : OBJET DE RECHERCHE ET DE TRANSFERT

En sciences de l'information et de la communication, le logiciel libre est d'abord un vecteur privilégié de collaboration et d'échange entre chercheurs. Il permet d'expérimenter facilement une idée. Il favorise la diffusion d'un résultat, au même titre qu'une publication. Sa réutilisation contribue à l'accumulation de savoir et de savoir-faire.

Par ailleurs, en facilitant l'incorporation immédiate de code issu de la recherche dans les solutions utilisées dans le monde socio-économique, le logiciel libre est également un moyen de raccourcir le transfert. C'est en effet une voie rapide — à défaut de directe — de transformation des connaissances en technologie, puis en produits ou solutions.

Qu'il s'agisse de recherche ou de transfert technologique, l'exigence de qualité concernant l'architecture et le code du logiciel est croissante. Les chercheurs sont conduits à professionnaliser leur approche du développement logiciel, avec une ambition forte de diffusion internationale et un souci constant de qualité favorisant la réutilisation. Il faut passer de la création d'un logiciel résultant d'un procédé « artisanal », consommateur de la matière grise d'un individu, à un processus pouvant impliquer plusieurs dizaines ou centaines d'individus. La solution est de maximiser la réutilisation des logiciels en s'assurant de leur interopérabilité grâce au respect des standards ouverts.

Rendre libre un logiciel permet le partage des efforts et facilite l'accès et la mutualisation de technologies. Obtenir un retour d'expérience

des utilisateurs doit permettre de valider le travail et de faire émerger de nouvelles problématiques de recherche.

i- **POSITIONNEMENT PRAGMATIQUE** — Avec le logiciel libre, appréhendé ici dans une perspective européenne, il est possible de prévoir l'élaboration d'un modèle économique intelligent et économique, créateur de valeur et d'innovation. L'avantage est aussi une réelle indépendance commerciale en regard de produits logiciels parfois en situation de monopole. On engendre ainsi un écosystème, à qualifier de vertueux, où les utilisateurs ne sont plus passifs, mais placés au centre du dispositif.

Développer du logiciel n'est pas gratuit. Dans un tel modèle, les frais sont partagés par une communauté d'intérêt qui bénéficie en retour du travail collectif. Le coût des tests et de mise au point se répartit entre les utilisateurs du système ; le prix de revient s'évalue surtout en temps.

Participer au développement, donc à la vie d'un logiciel libre et ouvert, ne signifie en rien travailler bénévolement. Bien au contraire, proposer des services multiples à l'utilisateur et des applicatifs fondés sur des modules libres existants est une manière très efficace d'organiser une activité économique.

Cette approche n'est pas idéologique. *L'open source* est un excellent outil de valorisation lorsqu'il s'agit de renforcer l'interopérabilité et de mutualiser des infrastructures pour élaborer des standards nouveaux. Néanmoins, cela ne signifie pas que tous les logiciels proposés par les chercheurs soient à publier sous licence *open source*. Il n'est pas question d'opposer à tout prix logiciel « libre » à « propriétaire ».

Dans de nombreux cas, une licence d'exploitation *ad hoc* reste préférable. La démarche est pragmatique, avec un seul objectif : maximiser l'impact des résultats des recherches. Il n'y a pas de chemin unique.

ii- **MUTUALISER LES EFFORTS** — Pour accomplir un projet de développement logiciel, un environnement de travail collaboratif est à agencer. Une telle « forge » logicielle met à la disposition des scientifiques et des développeurs une large panoplie d'outils : gestion de versions du logiciel, suivi des bogues et des améliorations, administration des tâches et autres facilités de communication — listes de diffusion, forums, pages de documents, etc. Ces dispositifs sont gérés de façon intégrée au moyen d'une interface Web, comprenant à la fois un espace public et un espace privé. L'environnement de travail évolue et s'adapte en fonction du projet. Cela encourage les bonnes pratiques de travail collaboratif et simplifie la diffusion du logiciel.

Parmi les nombreuses forges existantes, la plateforme SOURCEFORGE est sans doute la plus connue. Comme autre exemple, BUGZILLA est la forge des logiciels libres de la MOZILLA FOUNDATION. On peut encore citer LIBRE SOURCE, environnement collaboratif issu d'une plateforme financée par le RNTL — Réseau national des technologies logicielles.

Associer les utilisateurs à l'évolution d'un logiciel, c'est garantir que celui-ci soit véritablement adapté à leurs besoins et évolue en conséquence. Une solution est alors de créer un « consortium » auquel adhèrent les usagers. Ce modèle a été celui choisi pour échafauder la communauté *open source* du middleware avec OW2 — anciennement OBJECTWEB — ou pour édifier le logiciel de calcul numérique scientifique SCILAB. Cela facilite de fait la transition du logiciel libre d'un « objet de recherche » déconnecté des enjeux du marché, vers un « objet de transfert technologique » intégré aux logiques industrielles, sans pour autant renier la notion de recherche scientifique sous-jacente.

Il existe plusieurs modèles de gouvernance. Par exemple, pour SCILAB, une équipe dédiée au développement du logiciel s'appuie sur les compétences et le support de tous les membres du consortium et sur

© François Cointe



des ressources externes. L'équipe de développement offre un support et un vrai service. La sécurité juridique est également assurée. De quoi gagner la confiance de plus en plus d'utilisateurs !

iii – APPLICATIONS DE PLUS EN PLUS LARGES — Les entreprises ou l'administration publique commencent à déployer des solutions qui se fondent sur des architectures *open source*. Elles y voient essentiellement un moyen de permettre l'interopérabilité par la standardisation de l'infrastructure logicielle. Facilité de déploiement des innovations, rapidité de mise en œuvre des solutions client, jeu de la concurrence et maîtrise des coûts font partie des conséquences espérées.

Dans ce contexte, de nombreux acteurs industriels ou associatifs — tels l'**AFUL**, Association francophone des utilisateurs de **LINUX** et des logiciels libres — sollicitent les instituts publics pour travailler sur le sujet, car cette dynamique rencontre encore des réticences, à cause surtout des questions de pérennité, de qualité et de garanties juridiques.

Répondre à ces interrogations se montre une priorité, afin d'établir une confiance nécessaire au succès des modèles économiques relatifs aux logiciels libres et ouverts. Par leur engagement, les institutions publiques sont en mesure de contribuer à cette stabilité attendue. Aussi, il est avant tout impératif de définir un cadre juridique clair.

C. CECILL, UNE FAMILLE DE LICENCES DE DROIT FRANÇAIS

© François Contet



Pour amplifier le développement, la diffusion et l'utilisation des logiciels libres, les organismes publics et les entreprises ont donc besoin d'un cadre juridique intelligible et formel : c'est le rôle des licences qui définissent les droits et les devoirs des licenciés et des concédants.

Les licences logicielles de type libre et *open source* font pour la plupart référence au droit nord-américain. La conformité de ces licences au droit européen se trouve souvent sujette à débat, ce qui génère un manque de confiance, notamment sur les principes de la garantie, de la responsabilité et du droit d'auteur.

C'est pourquoi, en 2004, le CEA, le CNRS et l'INRIA ont élaboré **CECILL**, première licence française de logiciel libre. **CECILL** — acronyme pour Ce(a)C(nrs)I(nria)L(ogiciel)L(ibre) — est maintenant diffusée en version 2 et deux autres licences sont disponibles, **CECILL-B** et **CECILL-C**. Qu'en est-il en pratique comparativement aux autres licences ?

i – QUELLE LICENCE CHOISIR ? — C'est directement des droits et des obligations que lui confère la licence avec laquelle il est diffusé qu'un logiciel peut être qualifié de libre, *open source* ou non libre.

- **CECILL** reprend les principes de la **GNU GPL** et en est compatible. Celui qui distribue un logiciel réalisé à partir d'un logiciel sous **CECILL** est tenu de le distribuer sous **CECILL**;
- **CECILL-B** donne une grande liberté au licencié au même titre que la licence **BSD**. Le licencié peut modifier le logiciel et diffuser le résultat avec une licence de son choix. La contrepartie est l'obligation de citation par le licencié de l'usage du logiciel à travers une page Web publique, ce qui répond à l'objectif fort de mesure bibliométrique d'impact;
- **CECILL-C** a été conçue pour le cas des bibliothèques et plus généralement des composants logiciels. Dans l'esprit de la **GNU LGPL** (qui n'est plus vraiment adaptée à certaines situations), elle demande aux utilisateurs de rediffuser à la communauté les modifications qu'ils réalisent sur un logiciel sous **CECILL-C**.

Le choix d'une licence se révèle donc un élément clef pour la diffusion d'un logiciel et pour son succès. C'est une décision que les équipes

de recherche peuvent prendre avec un spécialiste de valorisation industrielle, afin de trouver l'adéquation entre les objectifs recherchés et les conséquences pour le chercheur, son équipe et son employeur, qui est le détenteur des droits patrimoniaux.

À noter qu'il existe des liens ténus entre ces licences logicielles et celles qui relèvent des *Creative Commons* — permettant de partager librement, de manière sûre, des documents ou des œuvres sur Internet — ou bien encore celles liées à la production artistique.

D. ALORS, RÉVOLUTION OU ÉVOLUTION ?

À partir de cette brève présentation des arguments liés à la diffusion sous licences libres et ouvertes des logiciels issus de la recherche, que peut-on conclure des changements induits dans les interactions de la recherche, d'une part, avec elle-même et, d'autre part, avec le reste de la société (entreprises, administrations, citoyens, etc.)? Constituent-ils une révolution à accomplir ou plutôt une évolution à réussir?

La première mission de la recherche est d'explorer l'inconnu, d'essayer d'apporter des réponses en développant de nouveaux savoirs, de nouvelles connaissances. Faciliter, grâce à des diffusions sous licences libres, la réutilisation de connaissances implicites qui se matérialisent dans des lignes de programme, s'inscrit naturellement dans la tradition de publication des résultats de la recherche depuis la fin du XVII^e siècle ; rien de révolutionnaire dans cette modalité. Il s'agit juste de s'adapter à cette évolution et, notamment d'ajuster un des mécanismes importants de la carrière des chercheurs : la mesure de notoriété et des citations.

Il y a cependant un changement particulier, une évolution majeure dans le rythme de création. En facilitant la réutilisation rapide de savoir et savoir-faire, la diffusion libre et ouverte de logiciel accélère sensiblement la création de connaissances disciplinaires, mais également et, c'est peut-être le plus important, dans les disciplines connexes, voire dans des communautés plus éloignées.

Parce qu'elle est immergée dans le monde réel et, par suite en fait partie, la recherche doit écouter les problèmes issus de ce monde, afin tout à la fois de ressourcer son questionnement et proposer des technologies aptes à participer à l'élaboration de solutions satisfaisant les besoins socio-économiques. La diffusion des savoir et des savoir-faire par les logiciels libres et ouverts apporte trois évolutions majeures :

1. elle accélère le transfert de technologie issue de la recherche vers le monde socio-économique à un coût de transaction quasi-nul;
2. elle contribue à la standardisation et l'interopérabilité, donc accélère la réutilisation et l'accumulation rapide de technologie;
3. et, de manière plus large, elle favorise le dialogue entre recherche et société.

Justement, du point de vue de la société, le logiciel devient un objet du quotidien ; c'est l'avènement de la société numérique. Les répercussions des choix opérés dans ce domaine ont un impact décisif sur des pans entiers de la société ; certains secteurs économiques devront donc s'adapter. Pour eux, c'est une vraie révolution.

Tout changement est difficile, surtout sans anticipation... Or, des forces conservatrices sont accrochées à des modèles d'affaires n'ayant pas évolué depuis plusieurs décennies. Un bon exemple en est le débat récent⁶⁴ autour du projet de loi DAVSI — Droits d'auteur et droits voisins dans la société de l'information —, où les fournisseurs de logiciels libres P2P — Peer to Peer ou pair-à-pair en français — seraient condamnables pour un usage illicite de leur logiciel : c'est le fabricant de l'outil qui est responsable et non celui qui l'utilise frauduleusement... Autrement dit,

⁶⁴. Ce texte date de 2006, néanmoins la thématique reste identique. À la loi DAVSI de 2006 a fait suite la loi HADopi de 2009. Plus récemment, le danger guette encore avec le débat sur l'acceptation par l'Europe de la brevetabilité des logiciels.

NOTE DE L'AUTEUR

Ce dernier paragraphe reprend la conclusion que j'ai formulée dans une tribune publiée dans *LES ÉCHOS* daté du 23 février 2006 et dont la publication témoigne que le monde économique est en train d'entendre ces arguments.

comme si le constructeur automobile était considéré responsable de l'accident commis par un chauffard alcoolique ou encore de même, le fabricant de marteaux lorsqu'un psychopathe tue sa victime avec un tel ustensile. Il y a donc danger.

Ouvert et partagé, mais cependant applicable aux mutations structurelles d'une économie de services complexes, telle est ma conception du logiciel libre et ouvert : mondialiser les savoirs, mutualiser les savoir-faire et encourager ici les nécessaires applications industrielles du « libre ». La culture du logiciel libre est légitime dans sa propension à partager des savoirs et à créer une dynamique vertueuse de mutualisation des technologies structurantes.

Je considère que c'est une chance à saisir pour la recherche et comme vecteur de transfert et d'innovation industrielle. Mais si la diffusion du logiciel libre et ouvert est une alternative crédible à celui dit propriétaire, ce n'est ni un renoncement à ce mode de publication, ni une opposition de fait aux logiques commerciales. C'est une occasion d'ouvrir des espaces nouveaux d'expression et de collaborations, de promouvoir d'autres modèles et, pourquoi pas, de favoriser la mutation et l'évolution du paysage économique.



POUR ALLER PLUS LOIN...

RESSOURCES COMPLÉMENTAIRES

- ▶ DATAK : jouez et apprenez à protéger vos données personnelles, PIXEES. ([lien mort](#));
- ▶ Dossier *Interstices* — Vie privée, surveillance, *Big Data*, analyse de données, sécurité; mieux comprendre les mots clés;
- ▶ Liste des organismes des pays européens de protection de la vie privée; CNIL — France —, CBPL — Belgique;
- ▶ Sciences du numérique et impact sur la société — Références pour réfléchir aux enjeux sociétaux des STIC.

LOGICIELS LIBRES

- ▶ FREE SOFTWARE FOUNDATION, FSF EUROPE, Open Source Initiative, GNU, APRIL — Promouvoir et défendre le logiciel libre.
- ▶ [ÉLIE, 2009] — François ÉLIE, 2009 — « *Économie du logiciel libre* », Accès Libre. EYROLLES, Paris, 186 pages.
- ▶ [JEAN, 2011] — Benjamin JEAN, 2011 — « *Option libre. Du bon usage des licences libres* », FramaSoft, Paris, 307 pages.
- ▶ [STALLMAN et al., 2010] — Richard M. STALLMAN, Sam WILLIAMS et Christophe MASSUTI, 2010 — « *Richard Stallman et la révolution du logiciel libre. Une biographie autorisée* », Accès Libre. FRAMABOOK-EYROLLES, Paris, 324 pages.
- ▶ [TORVALDS et DIAMOND, 2001] — Linus B. TORVALDS et David DIAMOND, 2001 — « *Il était une fois LINUX. L'extraordinaire histoire d'une révolution accidentelle* », OSMAN ÉYROLLES MULTIMÉDIA (OEM), Paris, 300 pages.
- ▶ [LESSIG, 2004] — Lawrence LESSIG, 2004 — “Free Culture”, THE PENGUIN PRESS, New York, 346 pages. (Traduction française).
- ▶ [BROCA, 2018] — Sébastien BROCA, 2018 — « *Utopie du logiciel libre* », Essais. LE PASSAGER CLANDESTIN, Paris, 380 pages.
- ▶ Le format *Open Document*, par Jean-Christophe Becquet, API-TUX, *Formats ouverts et interopérabilité #150*.

5 Que faire de ces ressources ? Autoévaluation

Le questionnaire à choix multiple* — QCM — à suivre clôture la chapitre « Numérique et sciences du réel ».

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

QUIZ 1 — ENTRAILLES ET CONTEXTE D'UN OBJET NUMÉRIQUE 7 POINTS

QUIZ 1.1 — ARCHITECTURE DE VON NEUMANN 1 POINT

L'architecture de « VON NEUMANN » a peu évolué depuis les premiers ordinateurs. Comment la caractériser ? Cocher les trois assertions exactes.

- L'architecture de VON NEUMANN repose sur trois éléments fondamentaux : unité de commande, unité arithmétique et mémoire.
- Cette architecture est complètement inspirée du cerveau humain.
- L'unité centrale est composée d'une unité de commande qui organise le déroulement des instructions du programme et d'une unité arithmétique qui exécute ces instructions.
- Comme un programme est stocké en mémoire, il peut être modifié par un autre programme qui le considère comme une donnée.
- Le modèle de VON NEUMAN n'a pas évolué depuis sa création.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style `LATeX` usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

QUIZ 1.2 — RASPBERRY-PI 1 POINT

Pourrions-nous équiper toutes les salles informatiques de l'ÉDUCTION NATIONALE de RASPBERRY-PI ? Il y a sûrement des obstacles à cette idée :

- cela nécessite une formation technique collective plus forte que des produits commerciaux clés en main;
- cela va à l'encontre des intérêts et des enjeux financiers des majors du domaine;
- et cela réclame plus de travail de la part des personnels qu'un produit clé en main.

Néanmoins, parmi les arguments ci-dessous en faveur de cette proposition quels sont les cinq qui sont exacts (un d'entre eux est faux).

- Cela diminue drastiquement les coûts de déploiement matériel.
- C'est un ordinateur suffisamment puissant pour faire tourner tous les logiciels actuels : édition vidéo, CAO, simulation, etc.
- La prise en main est réellement immédiate : moins d'une heure quand on a le matériel.
- On montre que l'informatique se développe bien plus à travers des objets connectés de ce type que de simples ordinateurs.
- Cela illustre la possibilité d'utiliser du matériel de récupération, ce qui est un geste d'éducation citoyenne.
- C'est un levier pour permettre de lever le voile de l'informatique et non uniquement d'en consommer les produits.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 1.3 — ACRONYME IHM 1 POINT

Que signifie l'acronyme IHM ? Plusieurs réponses sont possibles.

- Interfaces Homme-Machine.
- Interactions Homme-Machine.
- Interventions Homme-Machine.
- Internet Homme-Machine.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 1.4 — SYSTÈME D'EXPLOITATION 1 POINT**

En bref, à quoi sert un système d'exploitation ? Choisir la réponse que quelqu'un pourrait donner à ses garagiste ou fleuriste.

- C'est la couche logicielle qui permet — entre autres — de rendre transparent aux éléments applicatifs, les composants matériels ou micro-programmés de la machine pour n'avoir à spécifier que les aspects logiques du logiciel.
- C'est le truc qui exploite l'ordinateur.
- Dans le smartphone ou l'ordinateur, c'est tout simplement le logiciel qui fait le lien entre les applications et le matériel utilisé.
- C'est un logiciel spécifique aux ordinateurs, devenu inutile dans les smartphones qui sont basés sur des applications, donc pas besoin d'en parler à mes garagiste ou fleuriste.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 1.5 — TRANSPARENCE DES ALGORITHMES 1 POINT**

On parle beaucoup de transparence des algorithmes, par exemple pour les admissions post-bac ou pour un système de vote. Quel niveau de complexité pose cette exigence ? Choisir la bonne réponse.

- Rendre public le code source d'un programmeur est contraire aux règles de la protection intellectuelle, cela doit rester secret, comme les brevets.
- Il suffit de rendre public tous les codes sources des programmes.
- Il faut publier les codes sources, mais aussi en publier une analyse pour expliquer leur fonctionnement.
- Aucun. De toute manière un algorithme a un comportement plus ou moins imprévisible, donc cette notion de « code ouvert » est un leurre.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 1.6 — LIBERTÉS FONDAMENTALES DU LOGICIEL LIBRE 1 POINT**

Quatre « libertés fondamentales » définissent le logiciel libre, d'autres en découlent. Cocher les quatre éléments qui sont aux fondations de cette approche.

- La liberté d'exécuter le code du logiciel pour faire ce qu'on veut.

- La liberté d'obtenir toute l'aide et assistance dont on a besoin pour l'utiliser.
- La liberté d'étudier le fonctionnement du logiciel.
- La liberté de distribuer le logiciel, y compris de le vendre.
- La liberté de distribuer le logiciel, mais uniquement gratuitement.
- La liberté de distribuer le logiciel, mais uniquement sous la même licence.
- La liberté de modifier le logiciel, avant de le redistribuer.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 1.7 — PROTECTION DE LA VIE PRIVÉE 1 POINT**

Protéger la vie privée est un droit majeur (article 9 du code civil). Mais alors, pourquoi peut-on étaler sur la place publique certaines informations personnelles ? Avec les données numériques individuelles, ces questions prennent un grande ampleur et touchent plus de gens. Qu'en dit — en bref — la loi ?

- La loi ne dit rien, de toute façon liberté d'expression et droit à la vie privée sont inconciliables dans les faits.
- C'est la contribution de l'information à un débat d'intérêt général (procès, vie politique, ...) qui fonde principalement la jurisprudence sur ce sujet.
- C'est la notion de personne publique ou non, le levier de la loi, un personnage public ne rentre plus dans le cadre de l'article 9.
- En fait la loi s'applique complètement différemment dans l'espace numérique.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

WEB ET USAGES SOCIÉTAUX

LE WEB SOCIAL, AUSSI CONNU SOUS LE NOM DE WEB 2.0, a considérablement transformé les usages de l'Internet. Aujourd'hui, plus d'une dizaine de réseaux sociaux ont dépassé les cent millions d'utilisateurs; à l'exemple, WIKIPÉDIA est une gigantesque encyclopédie élaborée par ses propres utilisateurs.

Comment le Web a-t-il évolué d'un utilitaire documentaire vers une vocation à caractère social? Comment ont émergé les réseaux sociaux au cœur du Web 2.0? Des chercheurs tentent de répondre à ces questions et modélisent les usages sociaux du Web...

Parmi ces pratiques, devenues naturelles au quotidien pour la plupart* des individus, certaines, comme le « calcul dans les nuages » ou les monnaies cryptographiques, sont peu ou mal connues et méritent d'être mieux appréciées, en particulier dans leurs implications.

* De nouveau, comme au chapitre précédent, notons que les services et utilisations sociales du Web ont encore aujourd'hui leurs laissés-pour-compte : « fracture numérique » déjà évoquée ou « zones blanches » de certains territoires, entre autres ruraux.

1 Web et réseaux sociaux

Qu'est-ce que le Web ? Plutôt que le définir directement, il peut être plus aisément de répondre à son contraire : qu'est-ce que le Web n'est pas ? Cette approche se justifie notamment parce que les gens font souvent l'amalgame entre Web et Internet. Ainsi, le Web n'est pas l'Internet.

L'Internet est en effet un ensemble de standards et de technologies qui permettent de relier des ordinateurs et de raccorder des réseaux entre eux. On peut le concevoir comme un réseau routier sur lequel vont transiter différents types de services, de marchandises, de personnes ou encore de véhicules.

À l'instar du réseau routier offrant le passage de multiples éléments, l'Internet est le support de divers types d'application. Dans cette assertion, le Web en est une parmi d'autres.

1.1 Avènement du Web social

Le Web n'est clairement pas le seul « objet » de l'Internet. On peut par exemple également citer le courrier électronique, la téléphonie —

SOMMAIRE

1 Web et réseaux sociaux
1.1 Avènement du Web social 75
1.1.1 World Wide Web ■ 1.1.2 Réseaux sociaux ■ 1.1.3 Modélisation ■ 1.1.4 Enjeux
1.2 Filtrage selon FACEBOOK 80
2 Calcul dans les nuages
2.1 Quid du cloud ? 81
2.1.1 Infrastructure ■ 2.1.2 Problématique
2.2 Cozy Cloud : nécessaire vertu 83
3 Monnaies cryptographiques
3.1 Du bitcoin à la blockchain 86
3.1.1 Monnaie numérique ■ 3.1.2 Impacts
3.2 Startups bien de chez nous... 89
4 Numérique en question(s)
4.1 Penser le numérique 91
4.1.1 Ontologie ■ 4.1.2 Transformations
4.2 Être et écran 94
5 Que faire de ces ressources ? Quiz

À PROPOS DE L'INTERVENANT

Fabien GANDON, directeur de recherche INRIA et responsable de l'équipe WIMMICS (Université Côte d'Azur, CNRS I3S), étudie modèles et algorithmes afin de concilier Web social et Web sémantique. Il est aussi représentant de l'INRIA au consortium international de standardisation pour le Web (W3C).



VIDÉO 2.1 — Web sociétal I.

1. À titre de comparaison et en référence au chapitre précédent, on remarque que LINUX a vu son lancement en 1991, connu sa première phase de stabilisation de 1992 à 1996 pour fournir une solution viable au technophile moyen. Cette évolution s'est réalisée à travers le réseau téléphonique communiqué au moyen de modems permettant 56 bauds de vitesse de transmission théorique. Le prix des mémoires et transfert de données de cette époque imposait derechef une optimisation des codes.

VoIP, *Voice over Internet Protocol* — ou plus récemment la télévision et, plus largement, le multimédia. Il faut donc bien distinguer entre cette infrastructure de réseaux reliés entre eux que représente l'Internet et une de ses applications en surcouche qu'est le Web (protocole **http**).

1.1.1 World Wide Web

Initialement, le Web en lui-même est développé comme une application dévolue au partage d'une information statique, texte et image. Ces prémisses se situent au CERN — à la frontière franco-suisse — et datent du tournant des décennies 1980-1990. L'idée est alors d'apparier des documents entre eux pour permettre leur accès, même s'ils ne résident pas physiquement sur la même machine.

Au début, l'usage de cette fonctionnalité est réservée aux scientifiques, suite à l'ouverture progressive des connexions des universités à l'Internet — en France, avec RENATER pour Réseau National de Télécommunications pour la Technologie, l'Enseignement et la Recherche.

Au cours des années 1990, l'offre s'élargit assez rapidement au grand public. De ce point de vue, l'année 1997 marque en France un moment décisif : à la fois une meilleure accessibilité des matériels en termes de coût — ordinateur personnel (PENTIUM premières générations) et modem (modulateur-démodulateur) —, autant que les premières offres de service des fournisseurs¹ d'accès à l'Internet (FAI).

Le Web devient alors littéralement cette *toile* mondiale. En même temps, cela s'avère un peu ambigu puisque c'est à la fois les standards d'Internet qui permettent de créer cet objet et celui qui en émerge ; la toile mondiale sur laquelle on se balade tous les jours.

Dès l'origine de la « toile » dûment nommée, deux facettes se présentent. La première est d'ordre documentaire, afin d'associer des documents résidant potentiellement sur différentes machines. Mais il y a également l'idée que de multiples personnes puissent contribuer à différents documents demeurant sur plusieurs serveurs. Dès le départ, ces deux aspects — documentaire et social — coexistent, en perspective d'une interaction entre les divers contributeurs.

Au début, le Web est ouvert en écriture, mais très peu utilisent cette possibilité. La plupart des premiers serveurs Web sont accédés en lecture. Il faut attendre le milieu des années 1990 et la réouverture du Web en écriture notamment avec les *wikis* pour avoir la possibilité de voir tout le monde contribuer. C'est vraiment à ce moment-là que l'on bascule vers le Web social et toutes les applications sociales que l'on connaît désormais sur le Web.

1.1.2 Réseaux sociaux

Les réseaux sociaux prennent de multiples formes, mais on peut énoncer la définition d'un réseau social telle que celle proposée actuellement dans la presse : une application qui va utiliser les sciences et technologies de l'information, donc le Web, Internet et autres appareils mobiles pour mettre en *lien* des personnes. Cela constitue leur tronc commun depuis lequel elles se différencient, notamment en termes d'usage. S'agit de relier entre eux des *curriculum vitae* sur LINKEDIN ou d'échanger des messages plus ou moins sérieux sur FACEBOOK.

Ces applications diffèrent également en type de liens dans le réseau. Par exemple, est-ce un réseau d'amis, professionnel ou fondé sur des centres d'intérêt ? Une autre différenciation est aussi recouverte par les moyens d'accès : à travers le Web avec un navigateur sur ordinateur ou un mobile suffit-il. La distinction peut encore s'opérer selon les contenus partagés, à l'instar de messages courts comme sur TWIT-



TER ou de vidéos tel que sur YOUTUBE. Ce dernier exemple évoque une catégorie spécifique, le média social. Sa particularité est de produire et d'échanger des contenus multimédias (cf. chaînes YOUTUBE).

On peut aussi mentionner des réseaux sociaux explicites, autrement dit des réseaux sur lesquels les différents types de liens avec les autres personnes sont activement déclarés (cf. LINKEDIN), avec des réseaux sociaux qu'on peut qualifier d'implicites, c'est-à-dire dont la relation est calculée par rapport à l'activité des utilisateurs. En sciences, il existe des applications sociales comme RESEARCHGATE qui permettent aux scientifiques de poster et de maintenir leur bibliographie et les articles qu'ils écrivent. À travers cette activité et en regardant quelles sont leurs co-auteurs, on peut en déduire un réseau social implicite à cette activité. Enfin, une discrimination peut se conduire selon la plateforme : les applications sont-elles définitivement ancrées sur le Web comme TUMBLR où associées aux réseaux mobiles telles WHATSAPP ou SNAPCHAT ?

Par ailleurs, les réseaux sociaux n'ont pas attendu Internet ni même l'informatique pour exister. Depuis très longtemps la notion de structures sociales est étudiée en sociologie ; posséder et disposer d'un réseau est une expression bien connue. Même en informatique, cela prédate Internet. En fait, les *Bulletin Boards Systems* (BBS) ont connu leur apogée avant Internet dans les années 1970. Dans la foulée, des applications comme *Usenet* et les *Newsgroups* ont pris le relai sans attendre le déploiement complet de l'Internet. Dans la continuité, on va avoir le courriel avec les *mailing lists*, mais également les IRC — *Internet Relay Chat* — pour les conversations synchronisées. Ainsi, les forums ou *chat rooms* — salles de discussion — commencent très tôt en informatique.

Ensuite, avec le déploiement d'Internet, mais surtout du Web, cet espace n'est plus réservé aux technophiles. Une bascule s'opère en effet dans les années 1990 parce que le Web non seulement démocratise l'accès à Internet, mais aussi devient social. C'est-à-dire que ce n'est plus une bibliothèque, mais un endroit où tout le monde va pouvoir contribuer et écrire. Au début, c'est assez simple comme avec SIX DEGREES ou CLASSROOM pour simplement dire qui était son camarade de classe avant. Puis, petit à petit, apparaissent des applications comme les *wikis*, les *forums*, les *blogs* qui vont inviter à la contribution et donc aux médias sociaux.

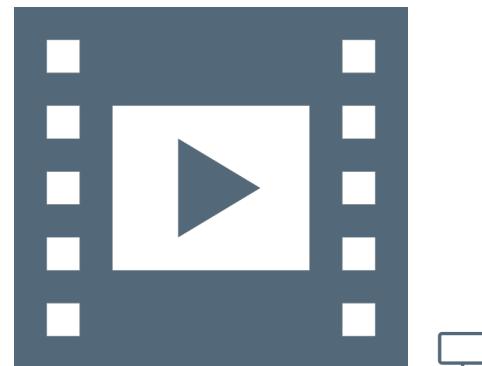
1.1.3 Modélisation des réseaux sociaux

Quand on a besoin de représenter ou de modéliser les réseaux en mathématiques, en informatique et plus généralement en sciences du numérique, le premier type de modèle employé se réfère à la *théorie des graphes*, pour capturer la structure du réseau social.

Dans un *graphe*, on trouve des *nœuds* qui représentent des entités comme les personnes et des *arcs* qui symbolisent les liens entre ces personnes. À partir de là, on va très vite voir que ce modèle peut s'enrichir et qu'émergent différents types de graphes pour décrire divers types de réseaux sociaux (cf. figure 2.1).

Si dans un réseau social, plusieurs liens sont possibles entre les personnes — par exemple un lien familial et un lien professionnel — le graphe va évoluer vers le multigraph, c'est-à-dire que plusieurs arcs vont relier deux mêmes nœuds.

On peut également faire appel à des *graphes orientés* où les relations entre individus sont dirigées dans un sens déterminé. Avec TWITTER, il est possible de suivre les posts d'une personne, mais rien n'oblige cette dernière à la réciproque ; les graphes sont alors orientés. Ce n'est pas nécessaire dans le cas de FACEBOOK où la relation « d'amitié » est convenue par défaut comme étant symétrique.



VIDÉO 2.2 — Web sociétal II.

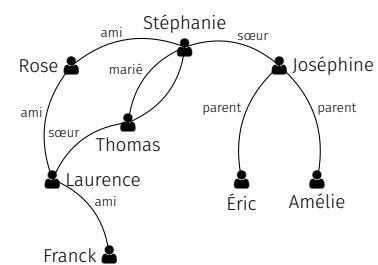


FIGURE 2.1 — Réseaux et graphes.

Plusieurs types de liens sont aussi envisageables dans un réseau. LINKEDIN propose un lien principal comme étant professionnel, mais il peut y en avoir d'autres. Auquel cas, il sont mentionnés dans un *graphe étiqueté* (voir figure 2.1), où les arcs comportent le type de lien entre les sujets. On peut encore s'intéresser à représenter des communautés ; on ne regarde alors plus vraiment la structure du réseau social, mais plutôt des groupes d'individus en son sein, par exemple selon des critères de mêmes centres d'intérêt.

D'un point de vue mathématique, là où on utilise des graphes, on peut aussi bien employer d'autres structures de modélisation comme les matrices. Une matrice peut s'élaborer à partir d'un graphe, notamment ce qu'on appelle une *matrice d'adjacence*, c'est-à-dire que pour chaque arc présent dans un graphe, pour chaque lien dans le réseau social, le coefficient de la matrice est mis à 1 pour signifier qu'il existe une relation. Les colonnes et les lignes d'une matrice d'adjacence représentent tous les nœuds du réseau et s'il y a un lien entre deux nœuds le coefficient à l'intersection de la colonne et de la ligne correspondante est mis à 1, sinon il est à zéro (cf. figure 2.2b).

Ce modèle matriciel peut être amené à évoluer. Si on doit évaluer la force d'une relation interpersonnelle ou tenir compte de sa certitude, en lieu et place d'un système binaire, on peut introduire des coefficients de pondération compris entre 0 et 1 : 0,25 pour une relation « faible » ou 0,75 pour un lien relativement fort (voir figure 2.2c).

Ainsi, selon le réseau, on est amené à employer diverses modélisations, mais celles-ci sont aussi guidées par les traitements à opérer.

1.1.4 Enjeux technoscientifiques

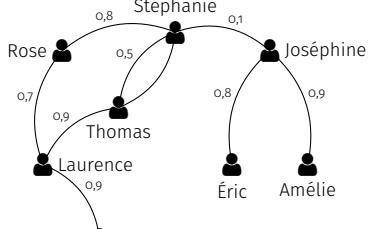
Les enjeux technoscientifiques sont nombreux avec les réseaux sociaux ; comme évoqué précédemment, celui du modèle en est déjà un. Viennent ensuite des questions pratiques : comment ce modèle peut être implémenté de façon efficiente en machine pour prendre le moins d'espace possible, pour être indexé et permettre un accès le plus efficace possible ? Mais aussi, comment peut-on le distribuer sur plusieurs machines lorsqu'il devient trop imposant et qu'il ne tient plus sur une seule machine ?

Donc, les approches de modèle, de représentation, de stockage et de gestion des données posent des interrogations technoscientifiques. En informatique, il y a toujours deux aspects, la structure de données et l'algorithme, émerge alors le problème des traitements à conduire sur les données, qui peuvent s'avérer extrêmement complexes.

La complexité des traitements tient d'une part, au volume de données à considérer — certains réseaux sociaux impliquent aujourd'hui des milliards de relations — et, d'autre part, à la nature même des traitements envisagés sur les données.

Par exemple, il est traditionnel de déterminer le plus court chemin entre deux personnes pour connaître comment elles sont reliées. Cette opération est déjà assez complexe : repérer tous les chemins qui existent, parmi ceux-ci regarder ceux qui concernent les deux nœuds d'intérêt et enfin trouver le plus petit dans cet ensemble.

De plus, dans un réseau social, on cherche les personnes les plus centrales, autrement dit les plus influentes. Pour cela, il existe une métrique qu'on appelle la *centralité d'intermédialité*. Elle consiste à évaluer le nombre de fois où une personne fait partie du plus court chemin entre deux autres. On calcule ainsi à quel point elle est intermédiaire et le nombre de fois où on passe par elle pour relier les autres. Il faut ainsi mesurer tous les plus courts chemins où elles apparaissent.



(a) Graphe pondéré.

	F	L	T	R	S	J	É	A
F	0	1	0	0	0	0	0	0
L	1	0	1	1	0	0	0	0
T	0	1	0	0	1	0	0	0
R	0	1	0	0	1	0	0	0
S	0	0	1	1	0	1	0	0
J	0	0	0	0	1	0	1	1
É	0	0	0	0	0	1	0	0
A	0	0	0	0	0	1	0	0

(b) Matrice d'adjacence non pondérée.

	F	L	T	R	S	J	É	A
F	0	0,9	0	0	0	0	0	0
L	0,9	0	0,9	0,7	0	0	0	0
T	0	0,9	0	0	0,5	0	0	0
R	0	0,7	0	0	0,8	0	0	0
S	0	0	0,5	0,8	0	0,1	0	0
J	0	0	0	0	0,1	0	0,8	0,9
É	0	0	0	0	0	0,8	0	0
A	0	0	0	0	0	0,9	0	0

(c) Matrice d'adjacence pondérée.

FIGURE 2.2 — Matrices d'adjacence.

On voit là l'augmentation de la complexité des calculs. En combinant celle-ci au volume de données à traiter, on aboutit rapidement à des problématiques d'optimisation des calculs, voire même d'approximation pour obtenir des résultats peut-être pas tout à fait exacts, mais suffisamment corrects et obtenus dans un délai raisonnable.

Plein d'autres questionnements sont à considérer, notamment ceux qui concernent les graphes. Ces derniers sont susceptibles de changer dans le temps en fonction de l'évolution des réseaux sociaux : création comme suppression de relations. Comment gérer, représenter et analyser ces modifications ? En référence aux médias sociaux, comment également traiter les changements de contenus : textes, images, sons, vidéos, etc.

Au global, la problématique est pluridisciplinaire, faisant appel au juridique, au politique et au sociétal. En résumé, on pourrait formuler que c'est justement parce qu'ils sont sociaux que ces réseaux interrogent toutes les disciplines.

GLOSSAIRE CONTEXTUEL

INTERNET — L'[Internet](#) est un ensemble de standards et de technologies qui permettent de relier les ordinateurs et les réseaux entre eux.

WEB — Le [World Wide Web](#), parfois aussi appelé la *Toile*, est une des applications d'Internet qui permet de lier et consulter à distance des documents (par ex. un journal), des interfaces d'applications (par ex. un service de réservation), des données (par ex. la météo d'une ville), etc.

RÉSEAU SOCIAL — Un [réseau social](#) associé au Web est relatif à une application qui utilise les sciences et technologies de l'information et de la communication pour mettre en relation des personnes. Hors monde numérique, un réseau social est un groupement de personnes qui a un sens.

WEB SOCIAL — Le [Web social](#) est une évolution particulière du Web caractérisée par l'interaction entre les utilisateurs et la production de contenus par ces derniers.

LIEN — Un lien est un élément qui établit des liaisons. Lorsqu'on parle de graphes, les liens sont ce qui relie les différents sommets ou nœuds.

GRAPHE — Un [graphe](#) est un type de structure de données composé de sommets (aussi appelés points, noeuds) reliés par des liens (aussi appelés arrêtes ou arcs).

GRAPHE ORIENTÉ — Si les liens d'un graphe sont orientés (l'orientation est matérialisée par des flèches), cela signifie que la relation va dans un seul sens, elle est asymétrique. Le graphe est donc orienté.

GRAPHE ÉTIQUETÉ — Il s'agit d'un graphe dont les liens sont étiquetés par un chiffre, un symbole, une lettre, etc.

MATRICE D'ADJACENCE — Une [matrice d'adjacence](#) est un outil mathématique qui permet de modéliser sous forme d'un tableau (ou matrice) les liens d'un graphe et donc, dans le contexte présent, les relations d'un réseau.

CENTRALITÉ D'INTERMÉDIARITÉ — La [centralité d'intermédiarité](#) est un moyen de mesure, parmi d'autres, de la centralité du sommet d'un graphe. Elle capture le nombre de fois où un noeud agit, dans un graphe, comme un point de passage le long du plus court chemin entre deux autres nœuds.

1.2 Algorithme Edge Rank ou le filtrage selon FACEBOOK

NOTE DE LA RÉDACTION

Texte rédigé par Rachid GUERRAOUI et publié sur Interstices — revue en ligne de culture scientifique du numérique — le 3 septembre 2014.

Si vous êtes adepte du réseau social FACEBOOK, vous aurez sans doute remarqué que toutes les activités concernant vos « amis » n'apparaissent pas forcément dans votre fil d'actualité.

Comment FACEBOOK fait-il le tri dans cette masse d'informations ? Pourquoi afficher telle actualité plutôt que telle autre ?



FACEBOOK connecte près d'un milliard de personnes. Chacune d'elles a un profil personnel, un compte qui comprend notamment un journal — précédemment appelé « mur » — sur lequel elle publie des informations la concernant et un fil d'actualité duquel elle voit défiler les activités de ses « amis » : leurs photos, les articles qu'ils partagent, leurs commentaires sur des articles postés par d'autres, etc.

Chaque utilisateur de FACEBOOK ayant en moyenne plusieurs centaines d'amis, tous les éléments que ceux-ci postent sur la plateforme, communément appelés *posts*, n'apparaissent pas dans son fil d'actualité, loin s'en faut. FACEBOOK opère en réalité une sélection radicale. D'une part, pour ne pas inonder les comptes d'informations qui disparaîtraient en une fraction de seconde à cause de leur trop grand nombre. D'autre part, pour faire en sorte que chaque personne trouve son fil d'actualité suffisamment intéressant pour rester connectée et active sur son compte. Car plus il y a de personnes connectées et plus FACEBOOK peut monnayer son support publicitaire.

Mais sur quoi FACEBOOK se fonde-t-il pour faire sa sélection ?

Pour décider ce qui va être affiché dans un fil d'actualité, FACEBOOK utilise un algorithme parfois appelé *Edge Rank*. Son principe de fonctionnement n'a rien de sorcier.

Pour chaque utilisateur « U », FACEBOOK calcule le score des informations (photo, note, article, commentaire, etc.) postées par ses amis. Le score d'un post « p » publié par l'un de ses amis « X », que l'on peut noter ici par « score(p) », correspond *grossost modo* au produit de trois variables, à savoir :

$$\text{score}(p) = A \times T \times F \quad (2.1)$$

- A désigne l'affinité de U par rapport à X : plus U a l'habitude de commenter des informations postées par X, plus A sera grand.
- T dépend du type de l'élément posté : ainsi, T sera plus élevé pour un élément de type photo que pour un petit texte par exemple.
- F représente la fraîcheur du post : plus un post est ancien, plus F diminue.

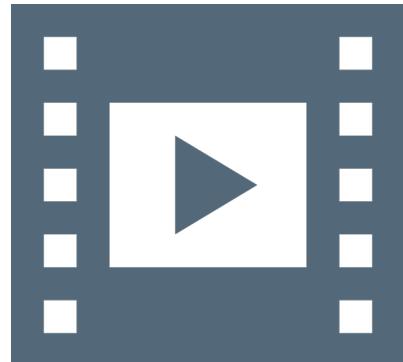
FACEBOOK calcule périodiquement les scores de toutes les informations postées par les amis de U et affiche dans le fil d'actualité de U celles qui ont le score le plus élevé.

Précédemment, il a été indiqué qu'approximativement le score correspond au produit des trois variables. En fait, il s'agit plus précisément d'une somme de produits, car chaque information peut avoir été commentée par plusieurs utilisateurs, et chaque commentaire contribue à augmenter son score.

Pour une explication plus complète du principe de l'algorithme *Edge Rank* de FACEBOOK, visionner la [vidéo 2.3](#) (durée : 10 min).

POUR ALLER PLUS LOIN...

- ▶ Autour du Web, Fabien GANDON, PIXEES;
- ▶ Les quatre facettes du Web, Fabien GANDON, PIXEES;
- ▶ Web sémantique et Web social, Fabien GANDON, 2009;
- ▶ Caractéristiques communes aux services de réseaux sociaux, CBPL Commission de la protection de la vie privée;
- ▶ Les réseaux sociaux, une vision par le sociologue Pierre MERCKLÉ. SES-ENS, 11/03/2012;
- ▶ 2025 Exmachina, Internet sans crainte, jeu sérieux d'éducation critique à Internet. Une production Tralalère avec la participation du CNC, le soutien de la Commission européenne et de la Délégation aux Usages de l'Internet dans le cadre d'Internet Sans Crainte;
- ▶ L'internet social (ou Web 2.0) : opportunités, impact et défi, Patrick VALDURIEZ, MSH-M TV, 9 février 2010;
- ▶ Les (r)évolutions de la planète Web, Agora Des Savoires 2015;
- ▶ Les algorithmes de classement utilisés dans les moteurs de recherche, Conférence de Michel HABIB, CANAL U, collection INRIA Science Info Lycée Profs, 63 mn, Juin 2009. Comment fonctionne GOOGLE.



VIDÉO 2.3 — Algorithme Edge rank, fil d'actualité FACEBOOK.



2 Calcul dans les nuages : *Cloud Computing*

La plupart des gens ont déjà entendu parler du « *cloud* ». Pourtant, peu de personnes savent exactement de quoi il retourne...

2.1 Quid du *cloud* ?

2.1.1 Infrastructure

DÉFINITION — Pour définir le « *cloud* » — nuage en anglais —, il faut peut-être revenir à ce qu'est l'informatique, soit d'une part, des données et d'autre part, des calculs effectués par des processeurs.

Chacun a l'habitude de son ordinateur personnel constitué de mémoires, d'un disque dur et d'un processeur. Cet ensemble calcule à domicile. Le concept de « *cloud* » est de déporter cette capacité de calcul hors de chez soi, dans « les nuages », c'est-à-dire probablement dans un *Data center* — ou centre de données en français —, qui peut parfois se situer de l'autre côté de la planète.

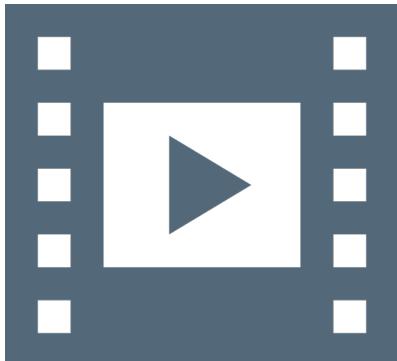
Le terme de *cloud* provient de la représentation d'Internet sous forme de nuage. À partir du moment où on va déposer ses calculs et ses données sur Internet, on va les mettre dans les nuages. Par extension, on va alors parler de calcul dans les nuages : *cloud computing*.

À PROPOS DE L'INTERVENANT

Spécialiste des données, de l'information et des connaissances, la recherche de Serge ABITEBOUL porte notamment sur la gestion d'informations sur le Web et la gestion de données personnelles. Ces sujets sont aujourd'hui essentiels face à l'accroissement et à la « massification » des données.

Directeur de recherche INRIA et professeur affilié à l'École normale supérieure de Cachan, Serge ABITEBOUL est diplômé de Télécom-Paris, a obtenu un Ph.D. de l'*University of Southern California* et une Thèse d'État de l'Université PARIS-SUD. Il est membre de l'Académie des Sciences et de l'Académie Europae, du Conseil scientifique de la SIF. Il a occupé la Chaire d'informatique au Collège de France (2011-2012) et la Chaire Francqui à l'Université de Namur (2012-2013). Il a été membre du Conseil national du numérique (2013-2016).

Il anime avec des amis le blog invité du Monde.fr, binary.blog.lemonde.fr, dont il est le fondateur.



VIDÉO 2.4 — Cloud computing.

En fait, il ne s'agit pas de quelque chose « d'évaporé », ce sont des éléments bien physiques; des ordinateurs et des disques qui ne résident pas au domicile de l'utilisateur, mais quelque part dans le monde... Dont on ne connaît même pas la localisation.

ASPECTS TECHNOLOGIQUES — Pour bien comprendre le *cloud*, il faut correctement appréhender ce qui a permis sa réalisation.

À l'origine de la création du *cloud*, il y a d'abord et principalement des réseaux hyper rapides qui connectent une entreprise ou la maison d'un particulier aux centres de données.

Par ailleurs, la seconde raison est liée à la baisse considérable des coûts des machines, des disques et du stockage qui sont devenus quasiment marginaux.

À partir de là, cela devient possible de concentrer dans un même lieu quantité de processeurs et de moyens de stockage puis de déporter l'ensemble de cette infrastructure.

La difficulté technique est de faire fonctionner ces gigantesques entrepôts de données. Deux problèmes majeurs se posent :

1. *la dissipation de la chaleur* — La concentration des ordinateurs génère une chaleur considérable qui demande de climatiser en permanence le *data center*. Des progrès remarquables en climatisation ont ainsi été réalisés;
2. *la gestion des pannes* — Statistiquement, un ordinateur va au bout d'un certain temps avoir une panne matérielle ou logicielle. Quand un million de processeurs sont placés dans un même entrepôt, la probabilité d'une panne devient très importante. Il faut donc avoir les moyens techniques de pallier cette problématique. Lorsqu'une panne se produit sur une application, on ne s'en aperçoit pas forcément car un autre ordinateur va automatiquement prendre le relais pour continuer son exécution. Beaucoup d'algorithmes ont été développés en ce sens.

2.1.2 Problématique des contraires

ATOOTS ET AVANTAGES — Lorsque des données sont placées dans le *cloud*, la première garantie est celle de la *sûreté*. Ces informations ne disparaîtront pas car elles sont probablement répliquées en plusieurs endroits, voire plusieurs centres de données.

Cette protection des données est également valable contre les intrusions. Ce n'est pas assuré à 100% mais, les compétences des personnels des *data center* sont certainement supérieures à celle d'un particulier ou d'une entreprise dont le métier n'est pas l'informatique.

Justement, un deuxième avantage du *cloud* pour les entreprises est de réaliser des économies de gestion. Ce n'est pas le cœur de métier, par exemple d'une entreprise de travaux publics, d'acheter et de gérer un parc informatique, ni d'embaucher et d'encadrer des *ingénieur système* et réseaux. Ainsi, le *cloud* permet d'*externaliser* ces services en perdant un peu le contrôle sur ses données.

INCONVÉNIENTS ET AMÉLIORATIONS — Beaucoup de points positifs du *cloud* ont jusqu'à présent été soulignés. Il existe néanmoins quelques aspects négatifs qui sont à exposer.

Tout d'abord les questions écologiques sont à préciser. En effet, quand les données sont distantes de plusieurs milliers de kilomètres cela génère une consommation d'énergie importante, voire des gaspillages, pour les rapatrier.

Un second point concerne un aspect sous-jacent relativ à une hyper-centralisation des calculs et des données. À l'arrivée, seules quelques entreprises possèdent toutes les données personnelles des individus

dans d'énormes centres de données. C'est un constat qui s'oppose à la philosophie originelle de l'Internet qui est plutôt de décentraliser et localiser les données et leur traitement. La décentralisation a beaucoup d'avantages; elle rend plus autonome et mène entre autres à des économies d'énergie.

Pour améliorer les choses, on pourrait justement imaginer de déconcentrer les *data center*. L'ensemble des équipements domestiques, de la télévision au téléphone mobile, sont des ordinateurs. Ce sont ce type d'ordinateurs qui pourraient localement conduire des calculs dans les nuages. En prenant l'exemple de la vidéo, plutôt que d'aller chercher un programme à l'autre bout du monde et dépenser inutilement de l'énergie, il serait possible de récupérer ce contenu au niveau régional ou local chez un voisin, lequel ne s'apercevrait même pas qu'il va servir un film. Ainsi, des économies substantielles d'énergie pourraient se réaliser, de même que retrouver l'esprit original de décentralisation de l'Internet et du Web.

GLOSSAIRE CONTEXTUEL

CLOUD — Le *cloud* signifie, qu'au lieu de stocker ses données et d'effectuer ses calculs sur son propre ordinateur, les deux sont confiés à un centre de données se trouvant sur Internet.

DATA CENTER — Un *Data center* ou centre de données en français est un site physique sur lequel se trouvent regroupés des équipements constituants des systèmes d'information : ordinateurs, baies de stockage, équipements réseaux et de télécommunications, etc.

SÛRETÉ — La sûreté de fonctionnement d'un système informatique est son aptitude à remplir ses fonctions en dépit de pannes matérielles ou logicielles.

INGÉNIEUR SYSTÈME — Métier de l'informatique qui prend en charge tout ce qui relève de l'exploitation des infrastructures informatiques matérielles et logicielles.

EXTERNALISER — Désigne le fait qu'une entreprise fasse appel à un prestataire extérieur pour certains pans de son fonctionnement plutôt que d'embaucher en interne un employé ayant la compétence requise (par exemple pour le ménage, la comptabilité ou l'accueil téléphonique). Avec le numérique, il devient plus facile et plus rentable d'externaliser beaucoup de services.

2.2 Cozy Cloud : vertueux par nécessité

Serge Abiteboul nous parle d'une startup, *Cozy Cloud*, qui développe un système de gestion d'informations personnelles. Il nous explique ce que sont ses systèmes, quels sont leurs buts. Avec les enjeux autour du contrôle des données personnelles, cette nouvelle approche prend tout son sens. Une startup qui mérite vraiment qu'on la suive de près.

NOTE DE LA RÉDACTION

Texte rédigé par Serge ABITEBOUL et publié sur Binaire — blog du numérique du journal LE MONDE — le 4 février 2016.

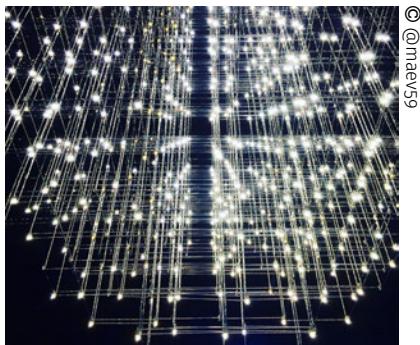
2 février 2016 — La startup *Cozy Cloud* et le bureau d'enregistrement GANDI sont lauréats de la 2^{ème} édition du Concours d'innovation numérique pour leur projet de *cloud* personnel grand public.

Nos données sont un peu partout, dans des services, dans de plus en plus de services différents. Nous finissons par ne plus très bien savoir où elles sont, ni même ce qu'elles sont ou ce qu'on fait avec. Donc, nous ne nous y retrouvons plus. Par exemple, nous nous rappelons que

“

Si vos données sont partout, c'est qu'elles ne sont nulle part.

— Benjamin ANDRÉ, PDG COZY CLOUD



nous avons l'adresse de ce copain, mais nous ne savons pas la trouver : dans nos contacts, dans nos mails, sur LINKEDIN, sur FACEBOOK, dans un SMS peut-être ou qui sait, sur WHATSAPP... Chacun de ces systèmes nous rend un service particulier, mais leur multiplication devient chaque jour un peu plus notre problème. Des systèmes se proposent de corriger cela, les systèmes de gestion de données personnelles, les *Pims* pour *Personal Information Management Systems*.

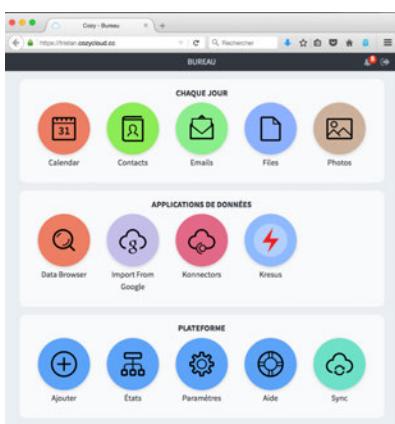
L'idée est simple : plutôt que de regrouper les données par services (les données sur les courriels de millions d'utilisateurs avec GMAIL, sur les films avec NETFLIX, sur les déplacements avec WAZE, etc.), on va regrouper les données par utilisateur. Donc nous aurons notre système à nous, pour nous, avec toutes les données des applications que nous utilisons. Ces données, nous voudrions qu'elles soient disponibles en permanence, de partout, on va dire que c'est « notre *cloud* personnel ».

Pourquoi promouvoir les *Pims*? Parce que la situation actuelle avec quelques sociétés — en caricaturant, les géants du Web —, qui s'approprient toutes les données du monde est fondamentalement malsaine. D'abord, à terme, nous y perdons notre liberté : nous sommes profilés par ceux qui savent tout de nous, qui choisissent pour nous ; et les services qu'ils nous offrent deviennent incontournables parce qu'eux seuls ont certaines informations et peuvent les fournir. Ensuite, ces grandes sociétés finissent par être à même d'étouffer la compétition. Internet et le Web qui ont servi véritablement de catalyse pour l'innovation, sont en train de devenir le royaume des oligopoles, les fossoyeurs des belles idées de liberté et de diffusion libre des connaissances des débuts. Bon, c'est un résumé un peu rapide, un peu brutal. Mais le lecteur intéressé pourra trouver un développement de ces idées [ABITEBOUL, ANDRÉ et al., 2015] dans CACM, la principale revue de l'ACM, une organisation internationale dédiée à l'informatique.

Donc, partons de l'hypothèse qu'il faille que chacun regroupe toutes ses données dans un système unique. Un *geek* saura installer un serveur et, en voiture LINUX! Mais la plupart des gens n'ont pas cette compétence et, même s'ils l'ont ou pourraient l'acquérir, ils ont probablement d'autres façons de dépenser leurs temps libre : le sport, les expos, le farniente...

Il y aurait bien une solution, ce serait de choisir les grands de l'Internet. Pourquoi pas tout mettre chez eux? Parce que nous aimerions avoir confiance dans le gardien de nos données. La confiance, le gros mot... Nous avons fait confiance aux fondateurs de GOOGLE, BRIN et PAGE, quand ils disaient « *Don't be evil!* ». Mais qui dirige GOOGLE aujourd'hui? Des actionnaires qui veulent maximiser leurs revenus? Pour protéger nos données personnelles, nous aimerions plus que de vagues promesses. Nous voulons des garanties! Nous allons donc plutôt choisir un tiers de confiance.

Un de ces tiers de confiance possibles, c'est la *startup* COZY CLOUD. Pour écrire cet article, j'ai rencontré son PDG Benjamin ANDRÉ. J'ai aussi côtoyé au Conseil national du numérique, son CPO — *Chief Product Officer* —, Tristan NITOT. Je suis fan des deux. Il faut rajouter que je suis un fervent supporteur des *Pims* et que ma recherche porte sur les *Pims*. Donc je ne suis pas toujours objectif quand j'en parle. Je pourrais parler objectivement de la recherche sur des *Pims*. Mais ce n'est pas le sujet de cet article. Ce qui m'intéresse ici c'est la gestion de données avec des *Pims* comme levier pour aller vers une société meilleure. Donc j'ai plus une casquette de militant que de scientifique. Cet article ne revendique donc aucune objectivité. Pourtant, je tiens quand même à souligner pour éviter les malentendus que je n'ai aucune participation financière dans COZY CLOUD ou d'ailleurs toute autre société de *Pims*.



Copie d'écran : bureau de Cozy Cloud

Un vrai argument des Pims (en tous cas, dans ma vision), c'est que leur logiciel est *open-source*. Bien sûr, nous n'avons pas le temps d'aller auditer leur code, mais d'autres peuvent le faire pour nous. Cette transparence sur la gestion des données est essentielle pour garantir que la plateforme ne va pas faire n'importe quoi avec nos données. Excusez du peu. Sans vouloir nous angoisser, toutes les données que nous avons à droite ou à gauche, des informations peut-être stratégiques pour nos entreprises, des informations intimes sûrement, les nôtres et celles de nos amis. Nous ne savons pas ce qu'on fait d'elles. Nous ne savons pas où elles atterrissent. Bon le mieux, c'est de ne pas trop y penser, ça va pourrir l'ambiance.

Le fait que le logiciel de la plate-forme soit en *open-source* et donc la transparence qui en résulte, est une qualité essentielle de ces systèmes. Cela facilite la vérification. Il faut aussi mentionner un autre aspect : la « portabilité ». N'ayez pas peur, c'est technique mais cela s'explique très bien.

Pour comprendre la portabilité, prenons un exemple de portabilité dans un autre domaine, l'automobile. Nous avons une PEUGEOT. Et puis, un jour, nous voulons changer de voiture. Nous sommes libres, d'acheter une RENAULT, même une VOLKSWAGEN, ce que nous voulons. Notre expérience de conducteur, nous la « portons » sous d'autres cieux. Nous n'avons pas à réapprendre. Dans les applications numériques, ça peut être différent. Nous avons choisi le KINDLE d'AMAZON. Et bien, c'est un super système, mais nous nous sommes fait avoir. Nous ne pouvons pas passer à un autre système sans perdre toute la librairie que nous avons achetée. Nous accumulons des années d'information, de données, dans un système et on nous dit « Restes avec nous ou perds tout ! » C'est l'emprisonnement par le vendeur, « *vendor lock-in* » en anglais. Nous aimerais pouvoir partir en « emportant » nos données dans le nouveau système – sans avoir à payer en argent, en temps, en quoi que ce soit. Le système doit nous garantir la portabilité, c'est-à-dire votre liberté de dire quand nous le souhaitons : « Ciao ! Sans rancune ».

Des systèmes comme COZY CLOUD nous permettent de partir quand nous le voulons, avec nos données. Nous restons si nous le voulons. C'est drôle de réaliser que le droit de partir peut devenir un argument pour choisir de rester. GOOGLE disait « *Don't be evil* » et il fallait croire sur parole qu'ils ne seront pas diaboliques. Dans un système qui garantit structurellement la portabilité, nous n'avons pas à les croire, ils n'ont d'autre choix que d'être angéliques s'ils veulent que nous restions. Cela pourrait être indiqué dans la loi. Des gens y travaillent.

Les députés ont validé le principe de récupération des données personnelles par les internautes. Il sera ainsi possible de transférer sa playlist ITUNES vers SPOTIFY ou ses photos INSTAGRAM vers une autre application. En revanche, cette obligation ne s'appliquerait qu'aux services grand public, excluant, devant la levée de boucliers des éditeurs de logiciels, les services inter-entreprises. Le Monde Économie, Sarah Belouzezzane et Sandrine Cassini, 19 janvier 2016.

Essayons de comprendre un peu mieux la techno. COZY CLOUD développe une plateforme pour gérer nos données personnelles. Nous pourrons un jour tout y mettre, nos contacts, nos courriels, nos déplacements GPS, nos documents, nos comptes bancaires, notre comptabilité... Ils nous proposent des applications qui réalisent certaines fonctionnalités (comme l'agenda) ou qui nous permettent juste de récupérer nos données d'autres services, par exemple nos mails de GMAIL. Cette plateforme, nous pouvons l'installer sur une machine personnelle, ou nous

“

La portabilité des données, c'est la possibilité pour un internaute de récupérer ses données depuis les grands services centralisés pour les mettre où il le veut. Pour lui, c'est une liberté fondamentale, celle de pouvoir « déplacer sa vie numérique » où bon lui semble, y compris chez lui.

— Tristan NITOT, CPO COZY CLOUD
Twitter : @nitot



pouvons demander à une société de l'héberger pour nous, par exemple OVH. Et à quoi sert Cozy CLOUD à part développer la plate-forme ? Ils peuvent gérer le système pour nous.

Nous n'avons pas dit grand-chose du *business model* de Cozy CLOUD. Bien sûr, c'est une *startup*, alors ils ont un *business model* qui montre qu'ils veulent se développer, ils cherchent des investisseurs, ils vont gagner plein d'argent. Mais nous pensons — nous espérons sans nous tromper — que Benjamin ANDRÉ, Tristan NITOT et les autres de Cozy CLOUD veulent d'abord changer le monde, en faire un endroit où il fait meilleur de vivre. Nous avons l'impression d'avoir entendu ça des tas de fois; ça peut prêter un peu à sourire; mais avec Cozy CLOUD, c'est tellement rafraîchissant.

Allez, un peu de fiction pour conclure, tout en restant conscient de la difficulté de prédire l'avenir. Nous aurons (bientôt) toutes nos données chez l'hébergeur de notre choix, elles seront gérées par un *cloud* personnel fonctionnant avec Cozy CLOUD (*Pimseur français*) qui procurera un point d'entrée unique à toutes nos données. Le système les rendra accessibles de partout, les synchronisera, les archivera, gérera nos Internet des objets, servira d'assistant personnel, suivra notre santé, notre vie sociale. Nous pourrons réaliser des analyses qui utilisent nos données mais qui, contrairement aux analyses *Big data* des autres, le fera pour notre bien et non pour maximiser le profit des autres. Et puis notre *Pims* pourra causer avec les *Pims* de nos amis... C'est dingue, nous étions totalement périphériques dans le monde des GAFAS, nous voilà transportés au centre du monde grâce aux *Pims*...

POUR ALLER PLUS LOIN...

- ▶ [Les données dans les nuages](#), Serge ABITEBOUL, Benjamin NGUYEN et Philippe RIGAUX, BINAIRE, 30 mai 2016;
- ▶ [Calculer dans les nuages](#), Patrick VALDURIEZ et Joanna JONG-WANE, INTERSTICES, 2011 (podcast 12 mn);
- ▶ [Managing your digital life](#), Serge ABITEBOUL, Benjamin ANDRÉ et Daniel KAPLAN, Communications of the ACM, 58:5, 2015;

3 Monnaies cryptographiques

3.1 Du *bitcoin* à la *blockchain*

À PROPOS DE L'INTERVENANT

Informaticien et mathématicien, Jean-Paul DELAHAYE est professeur émérite à l'Université de Lille et chercheur au CRISTAL (Centre de recherche en informatique, signal et automatique de Lille, UMR CNRS 9189).

Spécialiste en théorie de la complexité, il mène aussi des travaux dans le domaine de la modélisation et s'intéresse à l'utilisation et à la définition du hasard en informatique.

Jean-Paul DELAHAYE a publié de nombreux ouvrages scientifiques destinés à un large public. Il a reçu le Prix d'ALEMBERT 1998 de la Société Mathématique de France pour « *Le Fascinant nombre Pi* » et le Premier prix Auteur 1999 de la Culture scientifique du Ministère de l'éducation nationale, de la recherche et de la technologie.

Le *bitcoin* est une monnaie planétaire, cryptographique, fondée sur un système de transaction et de contrôle *peer-to-peer* — pair à pair en français —, la *blockchain* — littéralement « chaîne de blocs ». Comment cela fonctionne-t-il ? Quels en sont les enjeux ? Quels liens avec la sécurité de nos données, ici nos transactions bancaires ? C'est une véritable « révolution numérique ».

3.1.1 Monnaie numérique

L'argent peut-il exister sous forme numérique ? C'est une question à laquelle on peut répondre de plusieurs manières.

Déjà, chacun sait que lorsqu'on utilise une carte bancaire ou réalise un virement par Internet, c'est de l'argent numérique en fin de compte qui circule. Cependant, il existe depuis 2009 une autre forme de monnaie numérique plus intéressante qui, au niveau du développement de l'informatique et des idées va sans doute avoir un grand rôle; c'est ce que l'on nomme sous le terme de monnaies cryptographiques, dont le *bitcoin* est le premier exemple.

A. FONCTIONNEMENT

Le *bitcoin* est une monnaie créée *ex-nihilo* le 3 janvier 2009 en utilisant un réseau pair à pair. Son ou ses auteurs sont connus sous le pseudonyme de Satoshi NAKAMOTO, mais personne ne sait à qui il renvoie. Cette personne ou ce groupe a décrit le protocole, s'est arrangé pour écrire les programmes qui le font fonctionner et l'a mis en œuvre.

Ce protocole a aujourd'hui généré une monnaie qui s'appelle le *bitcoin*, dont la capitalisation totale vaut actuellement 10 milliards d'euros; ce qui est tout de même considérable.

Donc le *bitcoin* est une monnaie réelle, qui n'existe que sous forme numérique, il n'y a pas de billets, il n'y a pas de pièces *bitcoin*, c'est quelque chose qui circule uniquement sur les réseaux et dont le fonctionnement est fondé sur ce qu'on appelle un *fichier partagé* — on dit aussi un *registre partagé* — qu'on appelle la *blockchain*.

Pour comprendre le fonctionnement du *bitcoin*, il faut saisir le concept de *blockchain*. L'idée en elle-même est assez simple. Elle peut se formuler comme suit : « *si tout le monde est d'accord pour savoir qui possède des bitcoins, ceux-ci n'ont pas besoin d'être réels, c'est-à-dire matérialisés par des pièces métalliques, en or, en argent ou de quelque nature* ».

Cela semble un petit peu bizarre et, au début, on a du mal à y croire, mais cela fonctionne. Il existe ainsi un fichier² qui s'appelle la *blockchain*, qui indique précisément ce que détient chaque compte. Par le fait que ce fichier soit reproduit en de multiples exemplaires sur des ordinateurs différents — à peu près 5 000 dans le cas du *bitcoin* —, cela crée une sorte de confiance entre les utilisateurs car personne ne peut tricher en modifiant à sa guise le contenu du fichier, personne ne peut créer de *bitcoins* nouveaux sans que ce soit prévu par le fonctionnement général. L'origine de cette confiance est fournie par le partage de l'information. Le principe de la *blockchain* peut se généraliser à bien d'autres fonctions qu'un système monétaire.

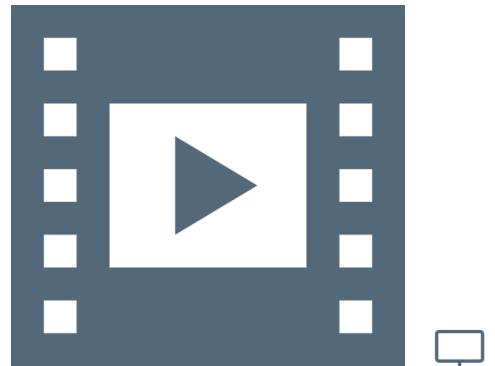
B. FIABILITÉ

Les *blockchains*, en particulier celle du *bitcoin*, se fondent sur les primitives de ce qu'on appelle la cryptographie mathématique. Ce sont elles qui créent la solidité des informations qui sont générées par la *blockchain*. Il en est de même de la fiabilité et de l'indestructibilité de ces informations. Tous les développements dérivés du *bitcoin* et de la *blockchain* sont issus des mathématiques, c'est une chose assez remarquable pour être soulignée.

C'est ainsi la maîtrise que l'on a aujourd'hui des protocoles de cryptographie, en particulier des systèmes de signatures numériques, des systèmes de hachage de fichiers, qui permet au protocole de fonctionner sans que personne ne puisse tricher. Car le cœur de la problématique est là, il ne faut pas que quelqu'un puisse manipuler le protocole pour s'attribuer de l'argent ou autre chose quand il s'agit d'une autre *blockchain*. Donc les mathématiques, et spécifiquement la cryptographie mathématique, grâce à la maturité de cette science informatique et numérique, sont à la base du *bitcoin* et de la *blockchain*.

C. TRANSACTIONS

Dans le protocole du *bitcoin*, il existe³ des signatures numériques ou cryptographiques. Quand quelqu'un détient des *bitcoins*, il dispose de deux informations. La première est son numéro de compte, comme un numéro de compte bancaire, que l'on va appeler la *clé publique*. La seconde est une *clé secrète*. Lors d'une transaction, le détenteur du



VIDÉO 2.5 — Monnaies cryptographiques.

². Techniquement, il s'agit d'une base de données distribuée, vérifiée par blocs et sécurisée par cryptographie (source [W](#)).

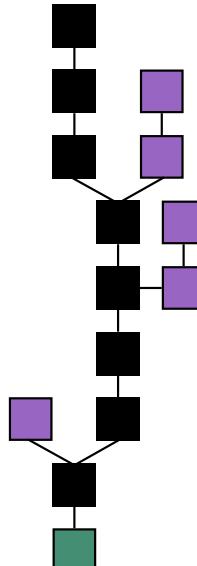


FIGURE 2.3 — Représentation d'une chaîne de blocs. La chaîne principale (noir) est composée de la plus longue suite de blocs après le bloc initial (vert). Les blocs orphelins sont violet (d'après [W](#)).

³. Le principe est équivalent pour d'autres *blockchains*.

compte va pouvoir la signer en utilisant sa clef privée au travers du réseau. Lui seul peut signer une transaction puisque lui seul possède cette clé privée. Dans le même temps, tous ceux qui connaissent son numéro de compte vont pouvoir vérifier que c'est bien lui qui a signé. C'est un petit miracle mathématique offert par les cryptologues : ces systèmes à double clefs permettent à tout le monde de vérifier qu'une signature est bonne.

Cette procédure est évidemment essentielle dans le protocole *bitcoin*; il ne s'agit pas simplement que certains comptes détiennent de l'argent, il faut aussi qu'il puisse circuler. Néanmoins, seul le détenteur d'un compte peut établir une transaction vers un autre.

3.1.2 Impacts

A. EXTENSIONS CRYPTOGRAPHIQUES

Dans un premier temps, on a pensé que la chaîne de blocs ne pouvait servir qu'à des applications de monnaie cryptographique — au passage, on peut noter qu'il en existe à ce jour environ sept cents établies sur le modèle du *bitcoin*, tellement l'idée a semblé intéressante, voire géniale. Depuis quelques années on s'est aperçu que la *blockchain* pouvait servir à bien d'autres choses, en s'appuyant sur le fait que ce fichier partagé est indestructible, par le nombre de copies et d'acteurs qui peuvent exercer un contrôle sur lui.

i – CADASTRE — Ce n'est pas encore en place, mais on a envisagé de faire un cadastre à base de *blockchains*. Les informations de propriété des terrains découverts seraient inscrites sur une *blockchain*, dont seules certaines autorités auraient le droit d'aller inscrire des informations.

Quand une vente serait opérée, on écrirait une information, non pas en effaçant la précédente, mais en venant la compléter en indiquant que tel terrain a été vendu à un nouveau propriétaire. Cette information serait accessible à tous à partir du moment où elle serait multipliée et rendue indestructible par l'utilisation de primitives cryptographiques.

Simplement en se connectant à l'Internet, on pourrait connaître les biens immobiliers des propriétaires avec bien plus de fiabilité qu'un système centralisé. Cela permettrait d'établir une sorte de confiance vis-à-vis des données, notamment dans des pays où la corruption sévit ou dans lesquels le cadastre n'existe pas.

ii – VÉRACITÉ DES DIPLOMÈS — Une autre application potentielle concerne les diplômes des écoles et des universités. Aujourd'hui, quand on veut savoir si quelqu'un possède un diplôme, c'est quasiment impossible ou alors il faut vérifier auprès de chaque université ou école qui a accordé ce diplôme.

On peut imaginer une blockchain sur laquelle les établissements d'enseignement et de formation indiquent les diplômes qu'elles délivrent, puis pour savoir si telle personne dispose effectivement de tel diplôme, chacun pourrait lire la blockchain. Là encore, la démultiplication du fichier garantirait la validité de l'information.

Régulièrement, des gens proposent de nouvelles applications des *blockchains*, donc d'une information partagée qui tient sa garantie de sûreté de par sa nature distribuée. Les développements en ce sens constituent une véritable révolution numérique, qui ne concerne pas uniquement les banques, mais également les organismes gouvernementaux et bien d'autres acteurs.

B. CONSÉQUENCES ÉCOLOGIQUES

À propos du *Bitcoin*, il est dit qu'il entraîne une dépense colossale d'énergie électrique et c'est vrai. La quantité d'électricité qui est dépen-



Diverses monnaies cryptographiques.



Blockchain et cadastre ? Ici le champs de Mars et la tour Eiffel.



Blockchain et diplômes délivrés ?

sée pour faire fonctionner le protocole est énorme. Néanmoins, il faut bien comprendre pour l'avenir du développement des *blockchains*, que cette particularité propre au *Bitcoin* n'est pas générale.

Dans leur ensemble, les *blockchains* n'ont pas obligation d'avoir un système — appelé système de minage du *Bitcoin* — qui entraîne une dépense importante d'électricité. C'est une erreur de croire que le développement des *blockchains* va être freiné par ces considérations.

GLOSSAIRE CONTEXTUEL

BITCOIN — Le *bitcoin* est une monnaie cryptographique associée à un système de transaction et de contrôle, nommé la *blockchain*.

BLOCKCHAIN — Technologie de stockage et de partage d'information avec un protocole de gestion de données numériques, qui est :

- ▶ transparente, car chacun peut consulter l'ensemble des échanges, présents et passés;
- ▶ sans organe de contrôle, car établie sur des échanges de *pair-à-pair* — *peer-to-peer* en anglais;
- ▶ infalsifiable et sécurisée, car différents exemplaires existent simultanément à de nombreux endroits ce qui empêche d'en modifier un ou quelques-uns.

MONNAIE CRYPTOGRAPHIQUE — Monnaie électronique fondée sur les principes de la cryptographie mathématique pour valider les transactions et émettre la monnaie elle-même.

3.2 Startups Blockchain bien de chez nous...

Le milieu des startups et des grandes entreprises d'informatique s'excite régulièrement sur un nouveau sujet. On voit défiler les modes à grande vitesse : big data, Internet des objets, machine learning... Et le petit dernier, la « blockchain ». Nous allons parler ici de startups françaises dans ce domaine. Nous allons supposer que le lecteur est vaguement familier avec la technologie blockchain, comme expliqué, par exemple, dans l'article de Jean-Paul DELAHAYE pour BINAIRE.

Une *blockchain* est un registre numérique public (sur le Web). Les données sont écrites dans un « grand livre » dont chaque ordinateur participant a une copie identique. On peut ajouter aux registres des transactions (au sens informatique comme au sens bancaire du terme) et ce, de manière totalement distribuée et sécurisée, assurant l'intégrité de ces transactions. Les transactions sont réalisées l'une après l'autre.

La *blockchain* permet donc de concevoir des architectures véritablement décentralisées; ce n'est pas simple parce que plusieurs transactions peuvent vouloir s'ajouter en même temps au registre et une seule doit y arriver. À chaque instant, la *blockchain* contient tout l'historique des transactions. La technologie *blockchain* est à l'origine d'une crypto-monnaie, le *Bitcoin*, qui agite beaucoup les milieux financiers. Les *Bitcoins* servent d'ailleurs eux-mêmes de « ressources » pour le fonctionnement des *blockchains*. Les *blockchains* semblent conduire à des échanges sécurisés distribués qui se libèrent du tiers de confiance (une banque centrale) ou d'une plateforme centralisatrice.

Cette techno n'est pas si vieille. L'an zéro du *Bitcoin*, c'est 2008. Mais pourquoi tant d'intérêt soudain : sans doute en partie à cause du succès de [chain.com](#) qui, autour du *Bitcoin*, propose une plateforme pour des échanges financiers entre entreprises. La technologie a mûri, ce que l'on observe également avec une plateforme *open-source* de *blockchain* très populaire, [ETHEREUM](#).

NOTE DE LA RÉDACTION

Texte de Serge ABTEBOUL et Pierre PARADINAS publié sur Binaire — blog du numérique du journal LE MONDE — le 20 juillet 2016.



© Laure CORNU

ATTENTION!

Lien mort au 31 juillet 2020.

On notera d'abord des startups autour des technologies financières comme PAYMIUM (NdR – associé à BLOCKCHAIN.IO depuis 2018) et LEDGER WALLET. La première permet d'acheter et vendre des Bitcoins, la seconde propose un portefeuille sécurisé sur smartcard ou clé USB. Mais la techno blockchain ouvre de nombreuses autres possibilités techniques, même si ses usages se cherchent encore.



Pour entrer dans ce nouveau domaine, il faut faire une distinction essentielle entre la blockchain public ou privé. Dans le public, utilisé pour les Bitcoins, chacun peut participer, par exemple en achetant ou en vendant des Bitcoins. Dans le privé, pour participer, il faut être « approuvé » par une autorité. Les vraies nouveautés semblent venir plutôt du côté des blockchains privées.

Une belle startup pour commencer : STRATUMN. La techno blockchain est compliquée. STRATUMN va la proposer « as a service ». On touche là à un des freins des blockchains : c'est une techno encore jeune et les outils sont encore compliqués à utiliser. STRATUMN essaie de les simplifier pour vous. Un beau programme. Vous venez avec votre idée d'application à coup de blockchain. Vous pouvez vous consacrer à cette application, STRATUMN gère la techno.

Mais pour développer quelle application ? Là vous avez le choix. Vous avez LEDGYS qui développe une place de marché sécurisée au-dessus de la technologie blockchain ou BELEM qui vise la gestion de données distribuées et la transmission d'informations sécurisée. Vous avez aussi KEEEX qui attaque le travail collaboratif avec encore de la gestion sécurisée de données décentralisées ou WOLEET, qui s'intéresse à la propriété intellectuelle et la certification entre autres choses.

Elles sont trop nombreuses ces startups. Il faudrait parler de AEDEUS, « la Première solution blockchain 100% française » (sic), ou de KAIKO « We organize Bitcoin's information by making the tools businesses need to succeed in the uncharted world of Bitcoin, and blockchain technology » (dans la version française de leur site).

Nous ne sommes pas certains d'avoir toujours compris ce que ces startups faisaient. Certaines nous ont paru encore très préliminaires. D'autres startups que nous avons regardées nous ont semblé bien trop fumeuses pour être mentionnées. Nous avons probablement raté de jolies perles. En tous cas, nous ressortons de cette balade dans les startups françaises du blockchain avec la sensation confuse qu'il est en train de se passer quelque chose, que si mille fleurs ne sont pas prêtes à surgir, quelques beaux bourgeons pourraient bien éclore.

POUR ALLER PLUS LOIN...

- ▶ Le bitcoin, une monnaie 100% numérique, Rémy CHRÉTIEN et Stéphanie DELAUNE, INTERSTICES, 8 septembre 2014;
- ▶ Le Bitcoin et la Blockchain, David LOUAPRE, Science étonnante, 24 juin 2016;
- ▶ Série d'articles publiés dans BINAIRE;
- ▶ Everything* You Always Wanted to Know About the Blockchain but Were Afraid to Ask, Arnaud SAHUGUET, The Foundry @ Cornell Tech, 17 octobre 2017;
- ▶ 3 start-up françaises qui innovent avec la blockchain, FRENCH-WEB.FR, 13 mai 2016;
- ▶ The Blockchain Might Be The Next Disruptive Technology, Florian GRAILLOT, TECHCRUNCH, 3 octobre 2015;
- ▶ BTC-Tech : Bitcoin and Cryptocurrency Technologies, Princeton University, Spring 2015.

4 Numérique en question(s)

Le numérique transforme en profondeur notre monde. Repérer ces transformations, comprendre les principes profonds du monde numérique et les changements de valeurs induits, pour mieux s'interroger sur notre devenir. Une invitation à revisiter les questions traditionnelles de la philosophie sous ce nouvel éclairage...

4.1 Penser le numérique

L'ingénierie des connaissances et le *Web sémantique* sont des disciplines, des outils technologiques qu'on utilise pour formaliser un certain nombre de domaines, lesquels n'appartiennent pas forcément aux champs scientifiques traditionnels, comme à titre d'exemple la banque. Il peut s'agir d'établir des *ontologies* précises ou larges concernant des objets, des processus de travail et tout ce qui peut se formaliser à des fins d'informatisation.

4.1.1 Notion d'ontologie

A. FORMALISATION

Cette formalisation en contexte informatique a d'abord été porté par l'intelligence artificielle, en particulier avec les *systèmes experts*. L'idée est de disposer d'une *modélisation des données* d'un domaine afin, une fois cette étape réalisée, de permettre à des machines de produire des *inférences*, des raisonnements à partir des entités du domaine et de générer ainsi de nouvelles connaissances.

Le mot *ontologie* est emprunté à la philosophie, introduit par John MCCARTHY le créateur de l'intelligence artificielle, à tout le moins l'initiateur de ce qu'on appelle l'*intelligence artificielle logique*. Il existe d'autres courants en IA, mais nous nous inscrivons ici dans celui-ci.

Le concept d'*ontologie* a une histoire complexe. Sans rentrer dans les détails, il désigne à la fois quand il apparaît, une théorie de l'objet et, sous sa forme popularisée, une théorie de ce qui existe. Ce mot a été adopté par John MCCARTHY pour indiquer ce qui existe à l'intérieur d'un domaine spécifique. On va donc procéder à la modélisation des données d'un domaine, de ses objets et de leurs relations entre eux comme mentionné plus haut.

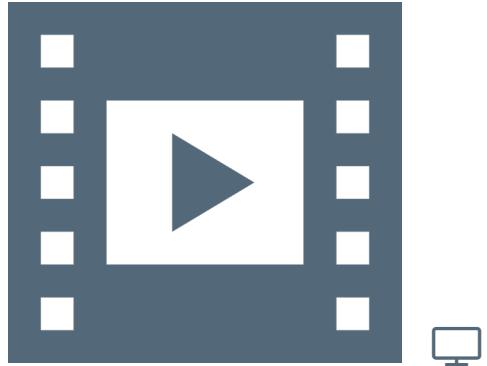
B. PHILOSOPHIE ET ONTOLOGIES INFORMATIQUES

Le concept d'*ontologie* en philosophe est extrêmement important, essentiellement employé au singulier. L'informatique s'est approprié le mot avec l'idée qu'il y a *des ontologies* puisque chacune est rattachée à un domaine différent.

Il existe donc deux acceptations selon la perspective que l'on adopte, considérant ainsi les *ontologies informatiques* comme des artefacts techniques. Néanmoins, même s'il y a une transformation importante du sens du concept d'une discipline à l'autre, des constantes demeurent, notamment le fait qu'en informatique on va aller chercher des outils de la philosophie pour modéliser les domaines et ce qui s'avère exister dans un espace limité et parfois, lorsqu'on essaye d'agrégner plusieurs domaines formalisés, faire appel à ce qu'on appelle des *ontologies de haut niveau* issues de la philosophie. Là encore, on à cette dernière emprunte des concepts permettant de décrire la réalité de manière très abstraite; c'est le travail des ingénieurs des connaissances, qui apportent de quoi peupler ces concepts et finalement raccorder ces *ontologies* les unes aux autres.

À PROPOS DE L'INTERVENANT

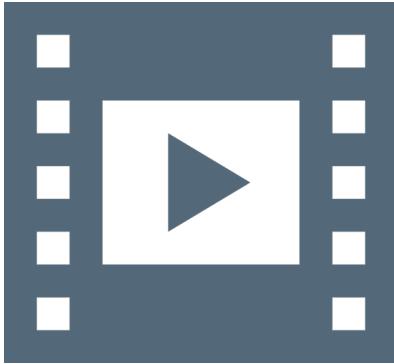
Alexandre MONNIN est docteur en philosophie de l'Université Paris 1 Panthéon-Sorbonne où sa thèse portait sur la philosophie du Web [MONNIN, 2013]. Il est chercheur dans l'équipe INRIA WIMMICS et expert *Open Data* auprès de la mission ETALAB sous la responsabilité du Premier Ministre. Il a initié plusieurs projets mobilisant les technologies du Web de données, à l'instar du DBPEDIA francophone et de RE-SOURCE, le système d'information de la Fondation des Galeries Lafayette pour l'art contemporain.



VIDÉO 2.6 – Informatique et philosophie.

D'ailleurs des philosophes travaillent désormais dans le champ des ontologies informatiques, comme par exemple Barry SMITH qui se qualifie « d'ontologue » et non plus de philosophe. Toutefois ces travaux ne sont pas conduits comme en ingénierie des connaissances, mais sous couvert d'une approche de philosophe. Il est donc intéressant d'avoir ce dialogue et ce regard sans forcément adhérer de manière intégrale à une vision plutôt qu'à une autre.

4.1.2 Transformations des valeurs dans le monde numérique



VIDÉO 2.7 – *Comment le monde devient numérique ? Gérard BERRY, 2008.*



A. CONFIANCE, DÉFIANCE, LÉGISLATION

Ainsi, le numérique s'empare d'un certain nombre de concepts, de pratiques, de valeurs, il les formalise, il les opérationnalise, il les numérise tout simplement, mais ce faisant, il les transforme.

Pour illustrer le propos, il est possible de prendre en exemple la problématique de la confiance numérique. La confiance telle qu'elle est définie par les sociologues consiste à ne pas savoir quelque chose, elle est de l'ordre du non-savoir tout en y portant crédit.

Si l'on confie son enfant à une nounou et que l'on met en place un dispositif avec des caméras qui la filment vingt-quatre heures sur vingt-quatre, la confiance envers la nounou est pour le moins absente... Donc on va avouer que le numérique ici, avec cette idée de surveillance, de générer des traces qui vont permettre de suivre tout ce qui se passe, n'est pas finalement un dispositif de confiance, mais en fait de défiance envers la nounou. Lui faire confiance serait : « je lui confie mon enfant, je ne sais pas ce qui se passe, mais j'accepte malgré tout de lui confier mon enfant ».

D'une certaine manière, en opérationnalisant la confiance, on aboutit au résultat inverse : la défiance. Le numérique transforme donc les valeurs ou les entités qu'il opérationnalise et, parfois, les transforme en sens opposé de ce qu'elles étaient précédemment.

La loi peut également fournir un bon exemple. Il existe des ontologies juridiques pour essayer de comprendre et établir des raisonnements à partir du droit. Toutefois, une des évolutions importantes de ces dernières années est de voir la justice rendue *a priori* en lieu et place d'un fondement normal du droit appliqué *a posteriori*, à l'issue d'un procès, d'un jugement, etc. En effet, comme pour les guerres préemptives – *preemptive wars* –, on assiste à une transformation du droit qui vise à prévenir un certain nombre de crimes ou de délits.

On va ainsi demander à des tiers, qui peuvent d'ailleurs être des entreprises, d'utiliser leurs algorithmes, leurs données, les données auxquelles elles ont accès concernant les personnes, pour justement permettre d'éviter qu'un certain nombre de crimes ou de conduites aient lieu. Donc, là aussi on a une transformation ; de l'*a posteriori* à l'*a priori*. L'*a priori* juridique n'aurait eu aucun sens précédemment, mais outiller techniquement par le numérique, il tend à devenir la nouvelle norme. C'est là où c'est quelque chose qu'il faut vraiment essayer de penser : la manière dont le numérique non seulement transforme le monde — le monde devient numérique —, mais encore comment ce devenir change un certain nombre de pratiques et de valeurs. Ce faisant, il n'est pas

certain que les valeurs nouvelles qui émergent soient celles dans lesquelles on se reconnaîsse toujours.

B. DEVENIR ET PÉRENNITÉ

À l'énoncé de Gérard BERRY, il faudrait rajouter une petite pastille. En effet, à y regarder de plus près, certes le monde devient numérique, mais il n'a pas les moyens de le rester.

C'est un point très important car le numérique n'est pas simplement de la science informatique, ce sont aussi des développements concrets, matériels, qui demandent des métaux, des ressources, de l'énergie, etc. Dans cette perspective, cela coûte finalement très cher de faire du numérique. À titre d'exemple, les outils de *deep learning* qu'utilisent les grandes entreprises et les algorithmes qui sont mobilisés consomment énormément d'énergie, donc ce ne sont pas forcément des choses généralisables dans toutes les situations, ni toutes les circonstances.

Ainsi, le monde devient numérique, il ne peut pas le rester et tout cela va en fait va très vite. Il ne faut pas simplement penser qu'on est dans une révolution et s'installer dans cette idée, d'une certaine manière qui peut être rassurante. Il y a une évolution, mais il faut se dire qu'on a affaire à une évolution qui est parallèle à d'autres évolutions, comme par exemple le changement climatique ou les perspectives d'affondrement dont on peut parler par ailleurs. Il va falloir penser tous ces éléments-là de concert, resynchroniser des futurs qui pour le moment ne le sont pas et qu'on ne sait pas exactement comment le faire. La question est de savoir que cette révolution numérique que nous vivons, ce devenir numérique du monde est lui-même temporaire. Il faut d'ors et déjà penser la fin du numérique en parallèle d'autres modèles. Ainsi, ces cultures numériques sont en partie temporaires et c'est ça qui est très difficile aujourd'hui à penser.

C. HÉRITAGE ET LIENS AU MONDE RÉEL

C'est dans l'expression « monde numérique » qu'il faut peut-être chercher des pistes de réflexion face aux problématiques mentionnées, notamment parce que le lien entre le monde et le numérique ne va pas du tout de soi. On peut considérer que justement le monde n'est pas *a priori* numérique, il n'est pas discret, il n'est pas *digital*. La production du monde numérique représente un investissement considérable ne serait-ce qu'en termes de matériel, donc dire que le monde devient numérique implique aussi de longs et coûteux processus.

Ainsi, se dessine ici une dialectique à penser entre un monde qui devient numérique⁴ et l'Ancien⁵ monde. Il y a donc une question d'héritage qui se pose au cœur de ce devenir numérique. De plus, si on considère que le monde ne peut pas rester numérique, nous devons penser l'héritage propre à ce monde numérique en deuxième surcouche. Là encore, que faut-il laisser de côté ? Ce sont là les véritables questions d'avenir et en particulier pour les jeunes générations.

GLOSSAIRE CONTEXTUEL

WEB SÉMANTIQUE — Extension du Web — à savoir syntaxique — dans laquelle l'information se voit associée à un sens bien défini, améliorant la capacité des logiciels à traiter l'information disponible sur le Web. Le Web sémantique repose sur des standards du W3C (organisme international de normalisation du Web) comme RDF (Resource Description Framework). L'expression a été inventée par Tim Berners-Lee, inventeur du Web et directeur du W3C.

⁴. Quel est-il, ce monde qui devient numérique ?

⁵. Qu'est-ce que nous gardons ? Qu'est-ce que nous abandonnons ?

ONTOLOGIE — Ensemble structuré de tous les termes et concepts de même que de leurs relations qui entrent dans la description d'un domaine de connaissance. Une ontologie a pour but de permettre l'automatisation de raisonnements à propos des objets du domaine concerné. « L'ontologie est aux données ce que la grammaire est au langage » ([W](#)).

SYSTÈME EXPERT — De manière générale, c'est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Sur ce sujet voir aussi la vidéo de Nicolas P. ROUGIER : [Enjeux et histoire de l'intelligence artificielle](#).

MODÉLISATION DES DONNÉES — Représentation abstraite, le modèle de données ne définit pas seulement la structure de données mais aussi ce que les données veulent vraiment signifier (sémantique).

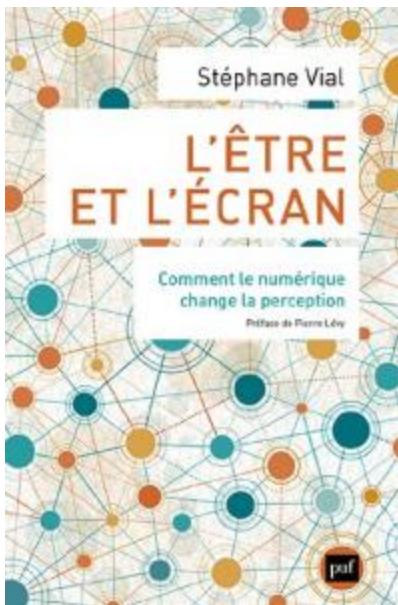
INTELLIGENCE ARTIFICIELLE LOGIQUE — Se référer aux explications sur ce courant de l'intelligence artificielle dans la vidéo de Nicolas P. ROUGIER : [Enjeux et histoire de l'intelligence artificielle](#).

DROIT PRÉEMPTIF — Sur ce sujet voir la vidéo de Daniel LE METAYER sur [le numérique, le droit et la vie privée](#).

DISCRET — Un objet est dit *discret* lorsque ses points sont écartés les uns des autres, le contraire étant dénommé un objet continu. L'opposition du continu et du discret est un thème philosophique important déjà interrogé par la philosophie grecque.

DIGITAL — Anglicisme souvent employé comme synonyme de numérique. Pour en savoir plus sur le débat digital vs. numérique : <http://www.blogdumoderateur.com/numerique-ou-digital>.

4.2 Être et écran : changement de la perception



Stéphane Vial, PUF, 2013 [VIAL, 2013]

« À tous ceux qui se demandent ce qu'il faut faire de la révolution numérique, il est facile de répondre d'un mot : il faut en faire le design ». Stéphane VIAL, ancien philosophe enseignant à l'école Boulle, ne prêche pas seulement pour sa paroisse — le « design numérique », prolongement du design classique adapté aux systèmes interactifs, cherchant à donner « vie » à des usages et « forme » à des contenus informatisés —, il s'inquiète plus fondamentalement des effets de la révolution en cours depuis trois décennies.

Il rappelle ainsi, par exemple, qu'entre 1981 et 2010 le nombre de terminaux interconnecté est passé de 213 à... 5 milliards. Et, pour figurer l'accélération fulgurante de ces évolutions, il souligne qu'en deux ans, entre 2010 et 2012, APPLE a vendu autant de tablettes IPAD que d'ordinateur MACINTOSH en vingt-quatre ans, soit 64 millions d'unités. Mieux, 120 millions étaient vendus en 2013 !

De quoi cette révolution est-elle le nom ? D'un bouleversement des conditions de perception du monde qui nous entoure, avec lequel on interagit, selon Stéphane VIAL. En clair, il faut désormais, quoi qu'on fasse, en passer par les « interfaces » (e-mail, FACEBOOK et écrans en tous genres).

« DÉFINIR À NOUVEAUX FRAIS LE CONCEPT DE VIRTUEL »

S'inspirant de BACHELARD et de sa « phénoménotechnique » — étude des effets de la technique sur la perception et les modalités d'apparition des phénomènes —, il conçoit le néologisme « ontophanique », pour rendre compte de ce qui se donne et apparaît à travers les interfaces numériques. Dénombrant les qualités de ce phénomène numérique, il définit du même coup à nouveaux frais le concept de « virtuel », trop marqué par un imaginaire métaphysique l'opposant au réel et

l'assimilant à celui de néant. Or le virtuel n'est pas le néant. Et, s'il en est un aspect, il ne résume pas le phénomène numérique.

Bref, la question ne se situe pas entre l'être et le néant, mais bien entre l'être et l'écran ou, avec l'être et l'écran. Cet effort de définition compte parmi les plus heureux de ce travail de recherche qui donne de la matière et du corps à une réalité numérique trop souvent et paresseusement conçue comme un monde irréel et vaporeux, sinon fumeux.

« LE MONDE NUMÉRIQUE PARTICIPE DE NOTRE MONDE »

Stéphane VIAL le martèle : le monde numérique participe de notre monde et ses effets sont tout aussi palpables. Comment ne pas voir que les réseaux sociaux, TWITTER, FACEBOOK, nos « ordinateurs mobiles » que sont désormais nos téléphones, les boîtes mail ou le développement des outils de réalité augmentée modifient notre environnement perceptif, participant à une « situation interactive généralisée » ?

Pour le chercheur, il convient donc d'exiger « du sujet contemporain un véritable travail phénoménologique en vue d'apprendre à percevoir cette nouvelle catégorie d'êtres, les êtres numériques [...]. Percevoir à l'ère numérique, c'est être contraint de renégocier l'acte de perception lui-même ». Comment le renégocier ? En acceptant, selon le vœu de Gilbert SIMONDON, réinvesti par Stéphane VIAL, d'accorder « aux objets techniques une place dans le monde des significations », en s'entendant sur le fait que « les objets font le monde ». Le design a de beaux jours devant lui.

POUR ALLER PLUS LOIN...

PHILOSOPHIE ET NUMÉRIQUE

- ▶ Penser le numérique : une question philosophique ?, Paul MATTHIAS, conférence du 1er décembre 2016;
- ▶ Pourquoi et comment le monde devient numérique, Gérard BERRY, conférence au Collège de France, 17 janvier 2008;
- ▶ Les nouvelles technologies : révolution culturelle et cognitive Michel SERRES, conférence sur les nouvelles technologies lors du 40^e anniversaire de l'INRIA , 2007;
- ▶ La société des calculs sous la loupe de la sociologie – A quoi rêvent les algorithmes, Dominique CARDON, Mais où va le web ? P(a)nser le numérique, article publié le 14 février 2014;
- ▶ Ontologies Informatiques, Fabien GANDON, INTERSTICES, article publié le 22/05/2006;

IMPACT ÉCOLOGIQUE DU NUMÉRIQUE

- ▶ Les effets rebond du numérique, Cédric GOSSART, ECOINFO, article publié le 23/12/2015;

WEB SÉMANTIQUE

- ▶ Mooc Web Sémantique et Web de données, Fabien GANDON, Olivier CORBY et Catherine FARON ZUCKER, INRIA 2015 Canal-U.

5 Que faire de ces ressources ? Autoévaluation

Les questions à choix multiple* — QCM — qui sont à suivre clôturent le présent chapitre « Web et usages sociétaux ». L'explication des réponses correctes s'affiche en marge : bouton « Afficher la réponse ».

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions, il s'agit de carrés (check box).

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style L^AT_EX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

QUIZ 2 — WEB ET USAGES 7 POINTS

QUIZ 2.1 — APPLICATION DE COURRIER ÉLECTRONIQUE 1 POINT
Le courrier électronique est une application du Web.

Vrai.
 Faux.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 2.2 — NOTION DE RÉSEAU SOCIAL 1 POINT
Depuis quand existent les réseaux sociaux?

Depuis la naissance d'Internet.
 Depuis la naissance de FACEBOOK.
 Depuis (presque) toujours. L'humain est un être social qui n'a pas attendu Internet pour construire son réseau.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 2.3 — INFLUENCE AU SEIN D'UN RÉSEAU 1 POINT
Qu'est-ce qui peut aider à déterminer la personne la plus influente au sein d'un réseau ?

Un graphe orienté.
 La centralité d'intermédiairité.
 Une matrice d'adjacence.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 2.4 — INFORMATIQUE DANS LES NUAGES 1 POINT
Quels sont les avantages principaux de l'utilisation du Cloud ?
Cocher les deux réponses exactes.

Plus de sûreté car mes données sont stockées de manière redondante dans des centres bien sécurisés.
 Moins d'impact environnemental puisque les machines ont été regroupées pour mutualiser leur gestion.
 La possibilité d'accéder à des services informatiques même pour une entreprise qui n'a pas la compétence en interne.
 C'est vraiment dans l'esprit d'Internet de décentraliser les données et les calculs.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 2.5 — MONNAIES CRYPTOGRAPHIQUES 1 POINT
À quoi pourrait servir la blockchain au delà de son usage pour les monnaies cryptographiques ?
Cocher les 2 réponses plausibles.

Enregistrer toute sorte de contrats (achats de biens, accords entre structures, etc.) de manière certifiée.

- Prouver l'identité d'une personne qui fait un acte.
- Prouver l'antériorité d'une création : œuvre artistique, invention technique, etc.
- Faire le café.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 2.6 — BLOCKCHAIN ET ÉCOLOGIE 1 POINT**

Quant au calcul quantique son seul point commun avec le nucléaire et d'en partager les notions de physiques théorique.> En quoi le mécanisme de blockchain pose-t-il un vrai problème écologique?

- Il n'en pose aucun bien au contraire puisque les calculs sont distribués, cela répartit la consommation.
- La méthode repose sur une consommation d'énergie de plus en plus importante pour sécuriser les échanges.
- C'est à cause de l'aspect quantique des calculs, cela génère une vraie pollution nucléaire.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 2.7 — NUMÉRIQUE ET PHILOSOPHIE 1 POINT**

Quelles notions philosophiques le numérique transforme-t-il?

Cocher les deux notions évoquées par Alexandre MONNIN.

- La loi.
- La beauté.
- La liberté.
- La confiance.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

INTELLIGENCE ARTIFICIELLE ET ROBOTIQUE

ENTRÉ MYTHE ET RÉALITÉ, l'intelligence artificielle pose question. Certaines personnalités (Bill GATES, Elon MUSK, Stephen HAWKING, etc.) ont lancé un appel afin de mettre en garde contre les dangers de l'intelligence artificielle. TERMINATOR et SKYNET vont-ils détruire l'humanité ? Visiblement, la question ne se pose pas comme cela...

Plus qu'être imbattable pour effectuer une tâche de calcul mental ou de jeu de go, l'intelligence se définit comme la capacité d'un système vivant à comprendre, interpréter, apprendre et s'adapter aux changements. *Être intelligent, c'est savoir trouver la réponse la plus adaptée à une problématique.* Pour cela, on s'appuie sur l'ensemble de nos facultés mentales et cognitives.

De plus, l'intelligence est incarnée, cela s'avère quelque chose d'in dissociable de notre corps. Si l'on reprend l'exemple du jeu de go, on fait appel à son intellect pour adapter sa stratégie de jeu à une situation, anticiper celle de son adversaire et trouver la meilleure combinaison pour emporter la partie. Mais alors, si en 2016 le programme ALPHA Go a réussi à battre le champion coréen du jeu de Go, cela veut-il dire que les programmes informatiques sont plus intelligents que l'intelligence humaine ? Bref, quelle différence entre l'intelligence d'un être humain et celle d'une machine ?

1 Intelligence artificielle démythifiée

Si l'intelligence artificielle est très impressionnante, elle se limite toujours à un domaine bien défini. Par ailleurs, elle n'est pas incarnée, contrairement à notre intelligence biologique. Elle se fonde sur l'apprentissage et pour cela a besoin de réaliser des statistiques sur beaucoup de données, à la différence de l'intelligence humaine qui peut faire des déductions pertinentes à partir de quelques exemples.

Bien entendu la machine est capable d'effectuer des calculs et de traiter des informations à un rythme affolant, mais cette machine ne comprend pas la tâche à exécuter. Par exemple, si on demande à un enfant de chercher l'image d'un chien dans un livre illustré, il lui suffira de visualiser une ou deux images de chien pour ensuite reconnaître l'animal, y compris dans une situation inhabituelle (de nuit, par exemple). L'enfant, au-delà de visualiser ce à quoi un chien peut ressembler saura par la suite, le définir, le décrire, voire même évoquer

SOMMAIRE

1	Intelligence artificielle démythifiée
1.1	Historique et enjeux 100
1.1.1	Origines ■ 1.1.2 Enjeux actuels
1.2	Intelligence artificielle débraillée 102
2	Découverte de la robotique
2.1	Mécanismes des robots 104
2.1.1	Qu'est-ce qu'un robot ? ■ 2.1.2 Autonomie ■ 2.1.3 Adaptation et apprentissage
2.2	Différences de finalités 106
2.2.1	Travail et exploration ■ 2.2.2 Assistance à la personne ■ 2.2.3 Modéliser le vivant
2.3	Éveil des bébés robots 109
3	Impact sociétal de la robotique
3.1	Robotique et éthique 116
3.1.1	Exemple du véhicule autonome ■ 3.1.2 Collaboration homme-robot ■ 3.1.3 Enjeux de la robotique médicale
3.2	Informaticiens et éthique 119
4	Deep learning
4.1	Comprendre le deep learning 123
4.2	Apprentissage automatique 124
5	Que faire de ces ressources ? Quiz

NOTE DE LA RÉDACTION

La partie introductive consacrée à l'intelligence artificielle est empruntée au Mooc « L'Intelligence artificielle... avec intelligence ! » également dispensé par l'INRIA sur la plateforme FUN-Mooc

de possibles liens affectifs qu'il aurait développés avec l'animal. Un algorithme, lui, aura besoin de centaines de milliers de photos avant de reconnaître un chien sans se tromper.

Par ailleurs, si une requête avec le mot « chien » est lancée sur Internet, le moteur de recherche sera capable d'afficher des centaines de millions d'images de chien, sans pour autant savoir ce qu'est un chien, ni le définir, ni le décrire, ni expliquer comment il a pu ressentir une quelconque émotion vis-à-vis de lui.

L'intelligence humaine ou animale — en somme biologique — se fonde sur des capacités cognitives et aussi émotionnelles, intimement reliée avec le corps. Une intelligence artificielle, dite « forte », qui serait capable d'être autonome et polyvalente dans des situations inattendues est un objectif scientifique. Cependant, actuellement, des résultats montrent que cette perspective idéale d'intelligence artificielle forte est techniquement impossible. Pour le moment, cela relève de la croyance, et non d'une future révolution scientifique.

1.1 Historique et enjeux

À PROPOS DE L'INTERVENANT

Nicolas ROUGIER est chercheur à l'INRIA au sein de l'équipe MNEMOSYNE et de l'Institut des maladies neurodégénératives à Bordeaux. Il travaille en neurosciences computationnelles et cherche à comprendre le fonctionnement du cerveau au travers de modèles informatiques.

Aujourd'hui l'intelligence artificielle est présente dans bon nombre de technologies utilisées quotidiennement : ordinateurs, téléphones portables, montres, enceintes... Elle est même présente dans les moteurs de recherche et sur les réseaux sociaux. C'est probablement ce qui explique le nouvel engouement qu'elle connaît aujourd'hui, elle nous passionne parce qu'elle a permis de *booster* les technologies et donc d'avoir de meilleurs usages de celles-ci.

1.1.1 Origines et cadre de développement

L'intelligence artificielle (IA) est une *discipline scientifique ancienne*, elle est d'ailleurs officiellement reconnue comme champ de recherche en 1956 et les avancées dans ce domaine sont en essor jusqu'en 1974.

Par la suite, les résultats espérés n'arrivant pas, les investisseurs se désintéressent de la discipline et les recherches en la matière connaissent un premier essoufflement qui dure jusqu'en 1980 : c'est le premier hiver de l'IA. C'est la *montée dans les années 1980 des systèmes experts*, qui permet de reproduire les capacités cognitives et d'être plus performant qu'un expert dans son domaine, qui relance la dynamique autour de la discipline. Mais là encore, l'enthousiasme des financeurs décroît face aux avancées plus lentes qu'attendues et l'IA connaît un second hiver d'une dizaine d'années.

Au milieu des années 1990, l'IA voit un nouveau *boom* propulsé — entre autres — par la victoire, en 1997 du programme DEEP BLUE d'IBM sur le champion du monde d'échecs Garry KASPAROV. De nouvelles performances, par exemple en reconnaissance d'images, sont à l'origine de l'enthousiasme actuel autour de la discipline. Les performances actuelles de l'intelligence artificielle, comme le développement des assistants vocaux, de l'analyse d'imagerie médicale ou celui des mécanismes intelligents à bord des voitures, résultent des avancées de la recherche de ces vingt dernières années. Ces *prouesses technologiques, parfaitables, impacteront sans doute à terme nos sociétés et seront à l'origine de transformations dans de nombreux domaines* comme notamment la santé, la justice, les médias ou les transports. Ces évolutions sont *cependant relatives, et nous devons adopter une attitude technocritique vis-à-vis de leurs applications*.

Une des toutes premières intelligences artificielles qui est été mise au point est le « logicien théorique », algorithme capable de démontrer



VIDÉO 3.1 — Intelligence artificielle.



trente-huit des théorèmes des *Principia mathematica*. Dès la naissance de l'IA émergent deux grands courants; l'un dit *symbolique* et l'autre *numérique*. Ils vont se distinguer par leurs perspectives, d'un côté on veut fabriquer un esprit et de l'autre modéliser le cerveau.

Fabriquer un esprit est parti d'une idée relativement simple de dire que le cerveau est une machine à traiter des symboles. Certains problèmes comme le jeu d'échec vont assez bien s'ancrer dans cette hypothèse. Énormément d'algorithmes ont été développés sur la puissance des symboles dans les années 1960 jusque début de la décennie 1970.

Quant au courant numérique, notamment mené par Franck ROSENBLATT, l'objectif de modéliser le cerveau a conduit au concept des réseaux de neurones artificiels et par exemple pour ROSENBLATT, à l'invention du *perceptron*.

De nos jours, si on sait répondre à certains problèmes particuliers comme le jeu de go, on ne parvient toujours pas à définir l'intelligence. Dans ces conditions, il est préférable de parler d'intelligences artificielles au pluriel voire même d'abandonner le terme au profit de dénominations différentes; c'est pourquoi les domaines de recherche actuels optent plutôt pour les appellations de concepts d'apprentissage machine, de réseau de neurones artificiels, de systèmes experts, etc.

1.1.2 Enjeux actuels

En 1956, Alan TURING propose un test pour détecter si une machine pouvait être qualifiée « d'intelligente ». L'intérêt de ce test est ce qu'on a appelé le jeux d'imitation ou le défi pour la machine est de se faire passer pour un humain en donnant des réponses convaincantes. Ce test fût et reste beaucoup critiqué — mais toujours cité — car il repose sur le langage et donc n'adresse la question que pour l'humain. Ni les bébés, ni les animaux ne pourraient passer le test. L'intelligence ne se réduit évidemment pas à la seule faculté de l'expression orale.

De nos jours, le défi posé par les chercheurs est d'élaborer un joueur de football et non plus d'échec ou de go. La raison provient du lien désormais tenu avec la robotique où un robot humanoïde doit pouvoir se mouvoir dans l'espace de manière autonome, mais aussi de manière collective en interaction avec ses partenaires de jeu. Il faut donc pouvoir détecter les émotions et les intentions de l'autre. Ces notions sont extrêmement complexes et font l'objet de beaucoup de recherches.

D'un point de vue plus général, cela rejoint le rejet de l'hypothèse symboliste en rapportant la thématique à ce qui est dénommé la *cognition incarnée*, dont un des textes fondateurs est un article de Rodney BROOKS, roboticien assez célèbre : « Les éléphants ne jouent pas aux échecs ». Ce titre un peu surprenant signifie que l'éléphant vit sa vie dans la savane ou la jungle sans avoir entendu parler du jeu d'échecs. On peut donc supposer qu'il n'a pas besoin de symbole mais possède toute son intelligence d'éléphant.

Aujourd'hui, les recherches en cognition incarnée suivent l'idée que l'intelligence se manifeste conjointement à un corps qui se développe et en interaction avec l'environnement. Si on veut par exemple tester la souplesse d'un objet, on a besoin d'un corps pour expérimenter la manipulation de cet objet. C'est un des grands champs de recherche en intelligence artificielle, mais aussi en robotique développementale.



Franck ROSENBLATT à 21 ans (1928-1971).



Rodney BROOKS en 2005 (1954-).

GLOSSAIRE CONTEXTUEL

PRINCPIA MATHEMATICA — Œuvre en trois volumes d'Alfred NORTH WHITEHEAD et Bertrand RUSSELL, publiés en 1910-1913, *Principia*

mathematica a pour sujet les fondements des mathématiques et est considérée comme un des livres les plus influents de l'histoire de la logique.

PERCEPTRON — Inventé en 1957 par Frank ROSENBLATT au laboratoire d'aéronautique de l'Université Cornell, le *perceptron* peut être vu comme le type de *réseau de neurones* le plus simple.

DEEP BLUE — *Deep Blue* est un superordinateur spécialisé dans le jeu d'échecs par adjonction de circuits spécifiques, développé par IBM au début des années 1990.

ALAN TURING — Alan TURING est un mathématicien et cryptologue britannique (1912-1954), auteur de travaux qui fondent scientifiquement l'informatique. Pour résoudre le problème fondamental de la décidabilité en arithmétique, il présente en 1936 une expérience de pensée que l'on nommera ensuite *machine de Turing* et des concepts de programmation et de programme, qui prendront tout leur sens avec la diffusion des ordinateurs dans la seconde moitié du XX^e siècle.

1.2 Intelligence artificielle débraillée

Note de la rédaction

Texte rédigé par Frédéric ALEXANDRE et publié sur Binaire — blog du numérique du journal LE MONDE — le 29 janvier 2016.

Si le nom de Marvin MINSKY (voir aussi en anglais), qui vient de décéder à Boston à l'âge de 88 ans, est indissociable du domaine de l'Intelligence Artificielle (Artificial Intelligence en anglais), dont il est un fondateur et reste un des chercheurs les plus influents, son impact a été encore plus large, aussi bien dans le domaine de l'informatique (prix Turing en 1969) que dans celui de la philosophie de l'esprit ou des sciences cognitives. Il a en effet aussi bien travaillé à décrire les processus de pensée des humains en termes mécaniques qu'à développer des modèles d'intelligence artificielle pour des machines.

Connu pour son charisme et la qualité de ses cours, il était professeur d'informatique au MIT à Boston, où il a créé dès 1959 le laboratoire d'IA avec John MAC CARTHY (autre prix Turing, inventeur du terme « Intelligence Artificielle »). Ce laboratoire et le plus récent MEDIA LAB auquel il a également appartenu, ont eu des impacts très importants dans de nombreux domaines de l'informatique.

Ce que l'on retient en général de Marvin MINSKY, c'est sa participation, avec MAC CARTHY, mais aussi NEWELL et SIMON, à la conférence de Dartmouth en 1956, généralement considérée comme fondatrice du domaine de l'intelligence artificielle. Il avait péché alors par excès d'optimisme en prédisant que le problème de la création d'une intelligence artificielle serait résolu d'ici une génération. La tradition veut qu'on retienne également sa participation, avec Seymour PAPERT, à un livre qui allait montrer les limitations des réseaux de neurones de type Perceptron et participer à ce que certains ont appelé l'hiver de l'intelligence artificielle, quand dans les années 70 les financeurs se sont détournés de ce domaine jugé trop irréaliste.

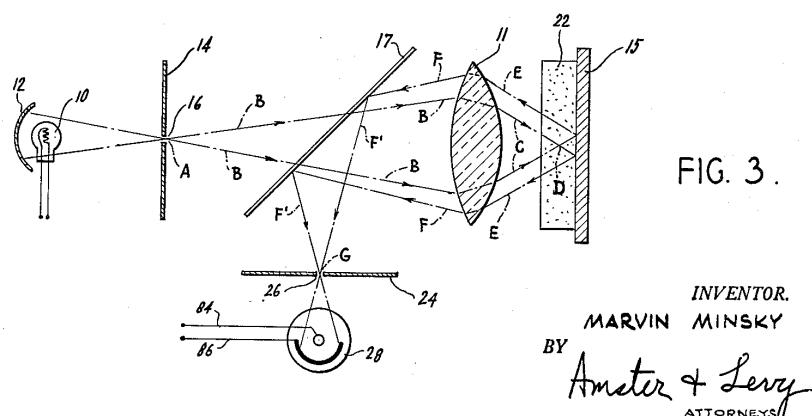
Ce que MINSKY a cherché à montrer tout au long de ses travaux, c'est que l'intelligence est un phénomène trop complexe pour être capturé par un seul modèle ou un seul mécanisme. Selon lui, l'intelligence n'est pas comme l'électromagnétisme : au lieu de chercher un principe unificateur, il vaut mieux la décrire comme la somme de composants divers, chacun avec sa justification. Il parlait ainsi d'intelligence artificielle débraillée ('scruffy' en anglais). Il insistait cependant sur le fait que chacun de ces composants pouvait être lui-même dépourvu d'intelligence.

Ce positionnement est très bien rendu dans son livre le plus connu, publié en 1985, "The society of mind", où il décompose l'intelligence en



Marvin MINSKY (1927-2016).

un grand nombre de modules ou d'agents, hétérogènes et parfois extrêmement simples, ce qui alimentait sa vision de l'esprit réductible à une machine. Il a poursuivi cette description dans un livre plus récent (*"The emotion machine"*, en 2006), avec d'autres processus plus abstraits, comme les sentiments. Avant ces écrits pour le grand public, il avait déjà proposé des contributions similaires pour le domaine de l'informatique, avec ses travaux sur le raisonnement de sens commun et la représentation de connaissances à l'aide de « *frames* » qui, dans les années 70, peuvent être vues comme précurseur de la programmation orientée-objet et qui lui ont en tout cas permis d'explorer de nombreux domaines de l'informatique relatifs à la perception visuelle et au langage naturel, ce qui l'a amené à être consulté par Stanley KUBRICK pour son film « *2001, Odyssee de l'espace* », pour savoir comment les ordinateurs pourraient parler en 2001...



- ▶ Vers une théorie de l'intelligence, Jean-Paul DELAHAYE, Interstices, 25 avril 2016;
- ▶ Pourquoi Stephen Hawking et Bill Gates ont peur de l'intelligence artificielle, France Culture, 2015.

2 Découverte de la robotique

À PROPOS DE L'INTERVENANT

Pierre-Yves OUDEYER est directeur de recherche INRIA et responsable scientifique de l'équipe Flowers. Il s'intéresse à la modélisation informatique et robotique de l'apprentissage et du développement sensori-moteur et social chez l'humain et les machines. Il étudie en particulier les mécanismes de curiosité, de maturation et le rôle du corps dans le développement cognitif et des interactions homme-robot permettant l'apprentissage par imitation. Il est aussi auteur du livre *Aux sources de la parole : auto-organisation et évolution* et participe activement à la diffusion des sciences vers le grand public au travers d'émissions, d'expositions et de documents et dispositifs de vulgarisation (cf. kits de robotiques pédagogiques pour le lycée).

Un élément paradoxal est que la robotique est en train de produire des objets programmables inouïs, mais tout à fait différents du fantasme usuel lié à la science-fiction du siècle dernier. Elle plonge ainsi le monde numérique au cœur même du monde réel et ce, au delà des idées reçues...

2.1 Mécanismes des robots

Les robots sont aujourd'hui partout autour de nous, ils ont une importance scientifique, sociétale, industrielle grandissante. Ils sont dans les champs, dans les usines, dans l'espace, au fond des mers, dans les jardins et même dans les salons. Un certain nombre de personnes pensent qu'au XXI^e siècle, ils seront un petit peu l'équivalent de la voiture au XX^e siècle. Ils n'ont pas simplement pénétré l'industrie, mais ils sont aussi en train de participer à la transformation de notre culture et même parfois, de contribuer à la manière dont nous comprenons nous-mêmes, les hommes, sous une facette renouvelée.

2.1.1 Qu'est-ce qu'un robot?

Un robot c'est avant tout une machine physique qui est dotée d'un certain nombre de moteurs pour produire des actions et des mouvements, qui est aussi dotée de capteurs — on peut les appeler aussi des senseurs — qui lui permettent de percevoir des choses dans son environnement, par exemple une caméra pour des images, des micros pour des sons, des capteurs de force pour détecter des obstacles ou qu'on les bouscule, des capteurs infrarouges pour parfois appréhender des choses que les humains ne sont pas capables de voir, mais que certains animaux sont à même de remarquer.

Ce qui est très important dans la définition d'un robot, c'est que les informations qui sont captées de l'environnement sont utilisées pour décider quels sont les mouvements, quelles sont les actions qui vont être produites par les robots. Il y a donc une interaction en permanence entre la perception et l'action; c'est cela qui définit les robots. C'est aussi ce qui les distingue des automates, par exemple de JAQUET-DROZ ou de VAUCANSON au XVIII^e siècle, qui étaient des machines extraordinaires, mais néanmoins dont l'action ne dépendait pas de ce qui se passait autour d'eux, ils faisaient toujours la même chose, quels que soient les événements qui se passaient autour d'eux.

Ainsi, un robot agit en fonction de ce qui se passe dans son environnement. Alors en pratique, cette définition recouvre une très grande diversité de formes de robots, qui peuvent correspondre à l'imaginaire que l'on en a, à savoir des robots humanoïdes ou animaloïdes, mais finalement qui aujourd'hui sont plutôt uniquement dans les laboratoires scientifiques. Mais cela concerne aussi beaucoup de robots de notre quotidien comme les avions qui sont aujourd'hui très largement robotisés, automatisés, les voitures quand elles sont en mode d'assistance à la conduite, des aspirateurs robotisés autonomes ou encore des



VIDÉO 3.2 – Robotique I.



Phantasme de robot araignée.

machines et des tondeuses à gazon. Cela inclut également les robots ludiques qu'on trouve dans les magasins de jouets, qui embarquent parfois des technologies très avancées.

À la diversité de formes des robots se conjugue aussi une relativement grande diversité de fond, de fonctionnement et de mécanismes. On peut alors parler de quelques grandes dimensions qui permettent de différencier les mécanismes.

2.1.2 Autonomie

Une première dimension à mentionner est l'autonomie. Certains robots ont un comportement qualifié d'autonome, parce que les actions sont prises pendant qu'il est en train de percevoir son environnement sans avoir recours à des instructions permanentes ou tout au moins partielles de la part de l'humain.

Par exemple, un robot qui est dans une usine automobile et qui assemble des voitures, parfois avec l'appui d'autres robots, est autonome si l'humain n'intervient pas dans l'assemblage et que les robots réalisent les tâches en toute indépendance. En revanche, un robot dans une centrale nucléaire télécommandé par un opérateur pour aller dans des pièces de confinement radioactives et définir quelles sont les actions qu'il doit faire à chaque moment n'est pas autonome.



2.1.3 Adaptation et apprentissage

Une autre dimension qui différencie plusieurs familles de robots est relative à l'adaptation et à l'apprentissage. Il existe certains robots dont le comportement est complètement figé à l'avance par un programme et, quelles que soient les expériences que le robot va faire dans son environnement, il se comportera toujours de la même manière. Par contre, d'autres robots sont dotés de familles différentes de mécanismes qu'on appelle des *algorithmes d'apprentissage*, qui leur permettent de mettre à jour leurs modes et leur stratégie d'action en fonction des événements qu'ils vont percevoir dans l'environnement, en fonction des essais et des erreurs qu'ils vont réaliser.

On a donc des robots qui sont à même d'explorer un environnement et progressivement en construire une carte qui va par la suite leur permettre d'y naviguer de manière relativement efficace alors qu'au début ils n'en étaient pas capables. Certains mécanismes vont leur permettre d'apprendre à reconnaître des objets nouveaux qu'un humain va leur nommer et des actions des humains ou apprendre à se déplacer en utilisant leurs jambes, autrement dit intégrer la locomotion.

Grace aux algorithmes d'apprentissage, ces machines détectent des régularités dans les expériences qu'ils peuvent réaliser. Une expérience, c'est essayer une action et en observer la conséquence, les mécanismes



Robot aspirateur.

peuvent se répéter un certain nombre de fois. En repérant les régularités, les robots peuvent comprendre un petit peu comment prédire les effets de leurs actions. Quand il s'agit de trouver une solution ou une stratégie de comportement, comme par exemple apprendre à marcher, les tactiques s'élaborent par essais-erreurs où, avec des algorithmes dits d'optimisation, les machines vont progressivement en raffiner les paramètres pour améliorer l'efficacité des solutions.

Dans un certain nombre de cas, les solutions qui sont trouvées par ces machines peuvent être assez surprenantes ou n'étaient pas forcément bien prédictes par l'ingénieur qui a conçu le robot ou l'algorithme d'apprentissage. Cela ne veut pas dire que ces robots ont inventer des choses complètement inimaginables, mais par exemple dans une stratégie de locomotion, cela peut correspondre à une manière d'utiliser son corps pour se déplacer qui peut paraître un petit peu saugrenue pour l'humain, mais qui au final s'avère très efficace.

On peut aussi avoir des algorithmes d'apprentissage un petit peu plus sophistiqués qui vont pousser les machines à explorer un environnement et à rechercher volontairement de la nouveauté ou de l'information afin d'augmenter leurs connaissances. Cela va permettre à ces machines de développer des répertoires de savoir-faire qui n'ont pas tous été programmés à l'avance. Cependant, il faut faire très attention quand on formule cette idée parce qu'on peut avoir l'impression que cela va être des machines relativement intelligentes et créatives, en fait elles sont encore extrêmement loin de la capacité d'adaptation, non seulement des humains, mais même d'animaux et de mammifères beaucoup plus simples. Il reste encore énormément de travail avant de construire des machines qui auraient la capacité d'adaptation, par exemple d'un enfant de trois ou quatre mois.



Dessin stylisé d'enfant cyborg.

2.2 Différences de finalités

Au-delà des différences entre robots en termes de mécanismes, il y a aussi des différences en termes de finalité, c'est-à-dire les raisons qui ont amené leurs concepteurs à les construire, à les tester ou à les disséminer. On peut distinguer plusieurs grandes orientations de finalités d'utilisation des robots dans notre société. On peut en mentionner essentiellement trois :

1. le travail et l'exploration;
2. l'assistance à la personne, à savoir accompagner l'humain dans ses tâches quotidiennes;
3. et l'emploi du robot comme outil de pensée, servant aux scientifiques pour essayer de modéliser et mieux comprendre certains mécanismes de cognition et d'apprentissage chez les animaux.

2.2.1 Travail et exploration

Parmi les robots en service dans le monde, la plupart sont des robots qui exécutent des tâches au sein d'usines manuelles. Il existe plus de neuf millions de robots industriels dans le monde, ce qui représente un nombre relativement élevé.

Très tôt, les entreprises se sont en effet intéressées à ces machines pour plusieurs raisons. D'abord, elles sont capables de réaliser des tâches qui sont pénibles, fatigantes et peu motivantes pour les humains, par exemple quand il s'agit de manipuler des pièces très lourdes ou dangereuses dans les usines. Ensuite, les entreprises s'y sont attachées parce que ces machines sont capables de souvent réaliser des



VIDÉO 3.3 – Robotique II.

tâches de manière très rapide, beaucoup plus efficacement que les humains. Parfois, c'est aujourd'hui le domaine de ce qu'on appelle la *co-robotique*, un certain nombre d'entreprises s'y investissent parce que la collaboration entre les humains et les machines permet de faire un travail de meilleure qualité — dans tous les sens du terme — que si on avait soit uniquement des humains ou soit seulement des robots.

Dans le domaine industriel et d'un point de vue historique, les premiers robots apparaissent dans les années 1960. En particulier en 1961, le robot Unimate est mis en service au sein d'une usine de GENERAL MOTORS. Depuis, les robots se sont beaucoup déployés dans le monde de l'automobile, mais pas seulement. Il y a bien d'autres domaines d'application, comme celui de l'agriculture qui aujourd'hui utilise beaucoup de robots pour cueillir des fruits dans les arbres, remplir des bouteilles, mettre des aliments dans des cartons, les trier et les ranger.

Dans cette catégorie du travail et de l'exploration, il n'y a parfois pas d'autre choix que d'employer des robots. C'est le cas des endroits et environnements trop dangereux pour l'homme ou, tout simplement, où l'homme ne peut pas aller. Un exemple de ce type d'environnement est celui des centrales nucléaires. Pour prévenir les incidents, on a impérativement besoin d'entretenir ces endroits et il faut être capable d'y aller. Les robots, téléguidés ou télé-opérés, sont souvent utilisés.

Un autre contexte dangereux est celui de l'inspection des coques des navires dans les ports, voire même directement en mer, pour essayer de devancer des accidents qui pourraient provoquer des désastres écologiques. Enfin, il y a un domaine inaccessible pour l'homme, c'est l'espace, un terrain sur lequel les robots ont beaucoup contribué à l'exploration du monde par l'humanité. Tout d'abord sur la lune, les premiers robots sont envoyés dès les années 1960. C'est en 1966 qu'ils arrivent sur la Lune avec la sonde Surveyor. Ensuite, plusieurs robots sont envoyés par les russes, comme le robot Lunokhod, et les américains comme les robots Mariner. Plus récemment, un certain nombre de robots sont déployés sur la planète Mars, comme Spirit et Opportunity. Grâce à ceux-ci, on a détecté des traces d'eau permettant des avancées scientifiques évidemment considérables.

2.2.2 Assistance à la personne

L'assistance à la personne est un autre champ d'application des robots, en particulier aujourd'hui dans une société où un certain nombre de personnes ont des handicaps physiques ou cognitifs. Des technologies robotiques ont été développées pour les aider à vivre de manière plus agréable, plus connectée à leur environnement. Pour les handicaps physiques, on a des robots qui leur permettent de les aider à se lever ou à s'asseoir, qui leur permettent de garder une autonomie physique. Pour les handicaps cognitifs, ça peut être des handicaps de mémoire. Il y a des robots qui vont stimuler les individus comme rappeler l'emploi du temps, les mettre en contact au fur à mesure de la journée avec leur famille, avec leur environnement médical.

Dans cette typologie d'applications, il existe également des robots qui assistent les gestes professionnels comme par exemple en milieu hospitalier pour accompagner les chirurgiens. Il y a encore des technologies robotiques, microscopiques, endoscopiques, qui permettent d'aller explorer l'intérieur du corps des humains. Un certain nombre de robots se développent dans le contexte des prothèses, par exemple quand des humains ont perdu un bras ou une jambe, il y a, aujourd'hui, des prothèses robotiques qui tentent de remplacer physiquement ces membres perdus et d'en restaurer le contrôle de manière à ce qu'ils soient le plus naturels possible.



Robot démineur Teodor.



Vue d'artiste de robot nomade sur Mars.

2.2.3 Modéliser le vivant

Un dernier secteur d'investigation, peut-être moins visible du grand public mais ayant son importance, concerne les laboratoires scientifiques qui, via les robots comme outil, tentent de modéliser le vivant.

Comment est-ce que les animaux et les humains en particulier, arrivent à acquérir des savoir-faire nouveaux, à s'adapter à un environnement qui lui-même évolue ? C'est un des grands mystères de la science aujourd'hui. C'est une énigme qui s'avère difficile à percer parce que les mécanismes de l'apprentissage impliquent l'interaction de processus nombreux, à plusieurs échelles d'espace et de temps, à l'intérieur du corps et du cerveau, mais aussi du cerveau avec le corps lui-même, et du corps avec son environnement. Cette interaction complexe est d'autant plus compliquée qu'à chaque fois que le corps et le cerveau interagissent avec l'environnement, le cerveau se modifie et donc, les modes d'interactions eux-mêmes se modifient.

Actuellement, on dispose de relativement peu d'outils pour comprendre cette complexité. En sciences physiques, il existe néanmoins une longue tradition d'étude des systèmes complexes comme ceux à la source de la formation des galaxies, des cristaux de glace ou de l'évolution du climat ou plutôt de la météo par exemple.

En physique, depuis très longtemps, on utilise les ordinateurs pour élaborer des modèles et lancer des simulations. En sciences du vivant et en sciences de la cognition en particulier, depuis plusieurs décennies les robots sont employés, un petit peu comme les ordinateurs des physiciens, pour modéliser certains aspects de cette complexité des interactions entre le cerveau, le corps et l'environnement.

Par exemple, cela peut servir à modéliser certains circuits neuronaux qui sont associés au contrôle moteur, la manière dont un enfant apprend à bouger sa main dans son champ visuel et à coordonner la perception et son action. Ce qui est très intéressant, c'est que comme c'est un cerveau et un modèle artificiel, on peut à volonté éteindre, entre guillemets, certaines parties du réseau neuronal et comprendre quel est l'impact que cela peut avoir sur le comportement, sur l'apprentissage. On peut aussi faire des expériences qui permettent de distinguer les contributions du corps et du cerveau pour la locomotion. On peut construire des robots qui ont la même forme, la même géométrie du corps que celle de l'humain et comprendre que même sans cerveau, quand on a la bonne géométrie, il peut générer spontanément un certain nombre de pas qui ressemblent beaucoup à ceux des humains.

Faire une expérience chez les animaux dans laquelle on séparerait le système nerveux du corps est impossible à la fois pour des raisons pratiques, mais surtout et évidemment pour des raisons éthiques. Avec des robots, on peut permettre un certain nombre d'expérimentations pour faire avancer nos intuitions, nos théories scientifiques.

Au-delà du contrôle moteur, il y a plusieurs laboratoires dans le monde qui s'intéressent à l'apprentissage du langage et à la manière dont un organisme peut, en interaction avec ses partenaires sociaux, acquérir des éléments de langage nouveaux. Ce travail met en œuvre des collaborations interdisciplinaires entre les sciences du numérique et la robotique d'une part, les sciences de la psychologie du développement et les neurosciences, d'autre part.

GLOSSAIRE CONTEXTUEL

ALGORITHME D'APPRENTISSAGE — Cela correspond à des algorithmes qui ajustent les paramètres de leurs calculs en fonction des



exemples qui leur sont donnés ou des retours (positifs ou négatifs, comme des récompenses ou punitions) issus de calculs précédents. Cela permet d'adapter leur fonctionnement aux données fournies.

ALGORITHME D'OPTIMISATION — Cela correspond à une famille d'algorithmes qui résolvent un problème par améliorations successives : on part d'une solution initiale par défaut, on la modifie un peu dans un sens ou dans un autre et si une de ses modifications améliore la solution, on réitère le procédé. Un critère de gain à maximiser ou de coût à minimiser est donc à la base de ces méthodes.

OPTIMISATION PAR ESSAIS/ERREURS — Cela correspond à un type d'algorithmes qui ajustent les paramètres par « renforcement », une solution est testée et en fonction du retour positif ou négatif, un ajustement se fait pour aller vers un meilleur comportement. Le fait que le retour se passe après l'action qui a pu en être la cause, parfois bien après, impose d'avoir une représentation interne de ce qui se passe au cours du temps.

MODÉLISATION — Représentation d'un objet réel ou abstrait, en éliminant les détails difficiles ou accessoires à reproduire, afin d'obtenir un résultat plus net à interpréter, un modèle est validé par son adéquation à des données (différentes de celles qui ont pu aider à le construire), donc à prédire des faits nouveaux; il peut-être mathématique ou informatique, mais aussi être un objet tangible (une maquette, un animal modèle pour certains fonctionnements biologiques).

CERVEAU — C'est l'organe principal du système nerveux des animaux, il régule les autres systèmes d'organes du corps, en agissant sur les muscles ou les glandes, et constitue le siège des fonctions cognitives (mémoire, apprentissage, planification, etc.); cette cognition n'existe que parce le cerveau est incarné dans un corps en interaction avec un environnement.

2.3 Éveil des bébés robots

Comment mieux comprendre le développement cognitif d'un enfant? En mettant dans les mêmes conditions d'apprentissage un robot curieux.

Dans son parc, un bambin empile consciencieusement un, deux, trois cubes, puis soudain, détruit l'édifice dans un grand éclat de rire. Et il recommence, avec à chaque fois un plaisir intact. Plus tard, près d'une rivière, il jettera quantité de cailloux pour les voir couler et autant de brindilles pour les suivre des yeux lorsqu'elles partent, flottant à la dérive. Ces gestes, mille fois répétés, ne sont pas anodins. Ils sont nécessaires aux êtres humains, dès leur plus jeune âge, pour comprendre le monde qui les entoure. De fait, les enfants expérimentent, construisent, manipulent, cassent... Pour tenter de comprendre les forces physiques. Jean PIAGET, l'un des pionniers de la psychologie développementale, a beaucoup étudié le rôle de l'action dans l'apprentissage et la découverte chez les enfants.

La science ne fonctionne pas autrement! Pour comprendre les vagues de l'océan, les chercheurs conçoivent des aquariums géants. Pour expliquer la formation des galaxies spirales, ils manipulent des simulations sur leurs ordinateurs. Ainsi, l'élaboration de modèles aide à construire un savoir. Qu'en est-il lorsque le sujet d'étude est l'être humain lui-même? Comment élucider les mécanismes de l'apprentissage humain, des émotions, de la curiosité? Étonnamment, nous pouvons

NOTE DE LA RÉDACTION

Texte rédigé par Pierre-Yves OUDEYER et publié sur [Interstices](#) — revue en ligne de culture scientifique du numérique — le 25 mars 2016. Une première version de cet article est parue dans le dossier n°87 Les robots en quête d'humanité de la revue [Pour la Science](#), numéro d'avril/juin 2015.

procéder de la même façon, grâce à des robots ! En effet, les chercheurs élaborent des « bébés robots » qui simulent certains aspects du corps et de l'esprit d'un enfant. Puis ils perturbent ces modèles de façon à déduire des comportements observés des informations sur les mécanismes internes. Les robots sont devenus des outils essentiels pour explorer la complexité du développement comportemental et psychologique d'un enfant.



© Inserm/P. Latron

FIGURE 3.2 – Les robots aident à étudier les interactions du cerveau, du corps et de l'environnement lors du développement cognitif.

Les sciences du développement ont invalidé la distinction entre la culture et la nature. On sait aujourd'hui que les gènes ne constituent pas un programme figé qui se déroulerait indépendamment de l'environnement. Plus important, de nombreux comportements ne peuvent s'expliquer par l'expression de quelques gènes, par le fonctionnement d'organes ou par quelques caractéristiques isolées de l'environnement. À l'inverse, ils traduisent les interactions de cellules, d'organes, de mécanismes d'apprentissage, de propriétés physiques et sociales de l'environnement à diverses échelles spatio-temporelles... En des termes plus techniques, le développement cognitif d'un enfant est un système dynamique complexe caractérisé par des phénomènes spontanés d'auto-organisation. De quoi s'agit-il ?

A. SYSTÈMES COMPLEXES

Les concepts de systèmes complexes et d'auto-organisation ont révolutionné la physique du XX^e siècle. Ils s'appliquent à des phénomènes aussi divers que la formation des cristaux de glace, des dunes de sable, des structures galactiques... Ces systèmes se caractérisent notamment par un grand nombre d'entités interagissant et par l'émergence de structures dont le plan global n'est pas présent au départ dans les différentes parties. Les simulations informatiques et les développements des mathématiques ont été un élément clef de l'avènement de ces concepts.

À la fin du XX^e siècle, les biologistes se sont emparés de ces idées, par exemple pour comprendre la formation des termitières. Grâce à des modèles, ils ont montré comment des interactions locales de milliers de petits termites, dépourvus de tout plan d'ensemble, peuvent conduire à l'érection de structures globales et fonctionnelles à grande échelle. D'autres simulations ont révélé les mécanismes auto-organisationnels qui expliquent les motifs sur la peau des zèbres et des girafes, les spirales des cornes et des coquilles, les battements du cœur...

Le développement d'un enfant met aussi en jeu les interactions de nombreux éléments, à un niveau sans commune mesure avec celui des phénomènes précédents. Ainsi, pour compléter l'arsenal d'ou-

tils conceptuels de la biologie et de la psychologie, les chercheurs ont commencé à construire des machines qui reproduisent certaines étapes du développement d'un enfant. La construction de machines qui se développent et apprennent comme des enfants n'est toutefois pas une idée entièrement nouvelle. Alan TURING, l'un des pionniers de l'informatique dans les années 1940, avait eu l'intuition de l'utilité des machines pour comprendre les processus psychologiques : « Plutôt que de créer un programme équivalent à l'esprit d'un adulte, pourquoi ne pas tenter d'en concevoir un qui reproduirait le cerveau d'un enfant ? »

Pour plusieurs raisons, les idées de TURING n'ont pas été traduites en un programme de recherche avant la toute fin du XX^e siècle. D'abord, les années 1950 ont été celles de l'émergence du cognitivisme et de l'intelligence artificielle. Selon les tenants de ces deux champs scientifiques, l'intelligence est uniquement fondée sur la manipulation de symboles abstraits. Dès lors, ils imaginaient, à tort, pouvoir reproduire directement l'état adulte d'un cerveau.

Ensuite, TURING a négligé deux éléments importants. Pour des organismes réels, l'apprentissage à partir d'une « feuille blanche », sans direction, est inopérant face à un flot gigantesque d'informations arrivant de toutes parts. Au contraire, l'épanouissement d'un enfant a besoin de balises, de mécanismes de guidage ! En outre, TURING n'a pas tenu compte du rôle du corps : le comportement et la cognition s'insèrent dans un « substrat », le corps, qui canalise le développement. Le rôle central du corps est la raison pour laquelle les robots, et non de simples simulations abstraites sur ordinateur, peuvent aider en sciences du développement.

B. MARCHE SPONTANÉE

Donnons quelques exemples, tel celui de la bipédie. Bien qu'elle nous soit familière, nous sommes loin de comprendre comment nous marchons sur deux jambes et comment les enfants apprennent à le faire. Ce mode de déplacement suppose une coordination en temps réel de nombreuses parties du corps. Chacun de nos muscles et de nos os est comme un musicien d'un orchestre symphonique jouant sa partition. De la cadence et de la coordination de l'ensemble naîtra une œuvre, en l'occurrence... Une marche cohérente et efficace.

Mais y a-t-il une partition qui précise le détail de la coordination ? Y a-t-il un chef d'orchestre ? Le cerveau a-t-il connaissance en permanence de l'état du corps et de l'environnement pour décider d'activer les bons muscles ? En termes plus techniques, la marche est-elle équivalente à un calcul ? La plupart des spécialistes de la marche humaine ont longtemps répondu oui. Dans ce cadre, la compréhension du développement de la marche nécessitait d'expliquer comment les enfants apprennent à effectuer ces calculs en temps réel et à faire des prédictions sur la dynamique du corps. Quelques roboticiens désireux de concevoir des machines bipèdes se sont fondés sur cette vision. Cependant, même si l'on compte quelques jolies performances, la plupart des robots obtenus tombaient facilement et avaient une démarche très peu naturelle. On doit donc se rendre à l'évidence, la marche serait un peu plus qu'un calcul. Ou un peu moins...

Dans les années 1990, le roboticien Tad MCGEER a conçu une expérience qui a bouleversé cette conception. Il a construit une paire de jambes mécaniques, calquée sur l'anatomie des membres humains et dépourvue de moteur ou de dispositif de calcul. Lancé sur une pente légère, le dispositif a... Marché ! À l'aide des interactions des composants mécaniques et de la gravité, les deux jambes ont automatique-



© S. Collins et al./Photo H. Morgan

En avant marche ! Cette structure adopte naturellement une marche bipède bien qu'elle ne possède ni moteur ni centre de contrôle. Cette marche est un comportement émergent résultant des interactions de l'anatomie (ici mécanique) avec l'environnement.

ment adopté une démarche similaire à celle d'un humain, le mouvement étant par ailleurs robuste et résistant aux perturbations.

D'autres laboratoires ont reproduit l'expérience et montré que la marche bipède pouvait durer « éternellement » sur un tapis roulant. La coordination des parties mécaniques du « robot », interagissant localement via les contacts physiques, est donc un phénomène auto-organisé à l'instar d'une termitière. La marche est un comportement dynamique émergent où la physique et l'anatomie ont un rôle essentiel, chacune fournissant à l'autre un support et un ensemble de contraintes.

Ces expérimentations révèlent comment des robots aident à élucider les rôles respectifs du corps et du système nerveux dans un modèle de la marche. Ce faisant, ils améliorent notre compréhension du développement humain. Ici, nous observons l'émergence spontanée d'un phénomène (la marche humaine) qui n'est ni inné (il n'est fondé sur aucun gène) ni acquis (l'apprentissage est absent). La distinction entre l'inné et l'acquis est parfois dépourvue de sens.

Une structure, ici la marche bipède, peut apparaître spontanément et émerger d'interactions biophysiques complexes. Une telle structure peut être mise à profit pour l'apprentissage et le développement. L'expérience précédente conduit ainsi à formuler l'hypothèse selon laquelle apprendre à marcher consiste à exploiter des mouvements qui sont inhérents à la dynamique du corps.



© Inria/Flowers

FIGURE 3.3 — Robots équipés d'un modèle de curiosité artificielle. Ainsi dotés, ils explorent un tapis d'éveil où ils apprennent à prédire les effets de leurs actions. Ce faisant, ils acquièrent spontanément des comportements et des connaissances dont la complexité augmente progressivement, par étapes.

Détaillons un second exemple, celui du rôle de la curiosité dans le développement des enfants. Ces derniers assimilent quantité de choses, le plus souvent progressivement, selon une chronologie spécifique. Ainsi, avant de marcher sur deux jambes sans soutien, ils apprennent à contrôler leur cou, puis à ramper sur le ventre, à s'asseoir, à se tenir debout, à marcher en se tenant aux murs... Comment alors expliquer cette progression ?

Plusieurs étapes d'une telle « trajectoire développementale » apparaissent dans le même ordre chez de nombreux enfants. Cependant, quelques-uns suivent des parcours différents. Comment d'une part, expliquer cette apparente universalité et, d'autre part, la variabilité indi-

viduelle ? Cette universalité est-elle le fruit d'un programme ? Chaque écart signifie-t-il que quelque chose s'est déréglé ?

L'environnement social est un élément central dans le guidage des processus développementaux. Il a été l'objet d'études de roboticiens qui se sont intéressés au rôle de l'imitation, de l'attention conjointe, du langage, de la synchronisation enseignant-élève. Mais une autre force tout aussi fondamentale est à l'œuvre en nous : la curiosité. Elle nous pousse à la découverte, à la création, à l'invention...

Des travaux en neurosciences et en psychologie ont montré que notre cerveau est naturellement enclin à tester de nouvelles activités pour le simple plaisir d'apprendre et de les pratiquer. Toutefois, nous ne savons que peu de choses sur la curiosité et son rôle dans le développement. Les neurobiologistes commencent à peine à identifier les circuits impliqués dans les comportements d'exploration spontanée.

Plusieurs équipes ont proposé, pour améliorer nos connaissances sur la curiosité et son rôle, de fabriquer des robots qui apprennent, découvrent et fixent leurs propres objectifs à partir de modèles d'apprentissage fondés sur des mécanismes d'exploration spontanée. Un exemple est fourni par l'expérience du tapis d'éveil, ou *Playground experiment* (voir figure 3.3).

Un robot apprend en faisant des expériences : il essaie, observe les effets de ses actes et détecte les régularités entre ses actions et les conséquences. Il peut ensuite faire des prédictions. La façon dont il choisit ses actions relève de la démarche scientifique : il sélectionne les expérimentations qui, selon lui, vont améliorer ses propres prédictions, lui apporter de nouvelles informations, augmenter sa capacité d'apprentissage. Ce faisant, il continue de consacrer une part de son temps à tester d'autres choses afin de découvrir de nouvelles pistes pour progresser. Le robot est aussi doté de mécanismes qui classent des expériences sensori-motrices en différentes catégories selon leur degré d'enrichissement et de contrôlabilité.

C. CURIOSITÉ ET AUTO-ORGANISATION

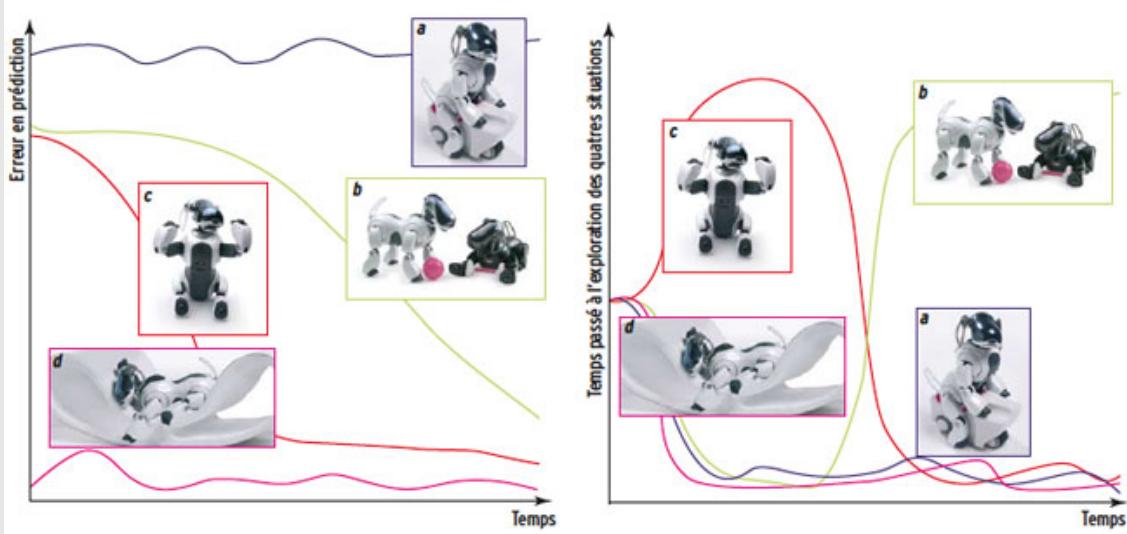
À tout moment de son développement, le robot se concentre sur les activités exploratoires qui améliorent l'apprentissage, en sélectionnant celles qui ne sont ni trop faciles ni trop compliquées (voir encadré ci-après). Ce comportement incarne l'hypothèse que le cerveau priviliege les expérimentations qui sont juste au-dessus de son niveau actuel de connaissances et de compétences. Un tel modèle permet de proposer une définition pratique de la curiosité : elle serait un mécanisme de motivation qui pousse l'organisme à explorer des activités dans le seul but d'obtenir des informations pour elles-mêmes (par opposition à la recherche d'informations dans le but de trouver de la nourriture ou un abri par exemple).

Dans l'expérience du tapis d'éveil, le robot apprend des tâches à partir de ses propres initiatives (par exemple, attraper un objet posé devant lui), mais il fait aussi évoluer son comportement, où apparaissent des phénomènes d'auto-organisation, en augmentant progressivement la complexité.

Des étapes cognitives apparaissent sans qu'elles aient été préprogrammées. Ainsi, après avoir commencé par des mouvements désordonnés, le robot se concentre souvent d'abord sur les mouvements de ses membres pour, éventuellement, atteindre des objets. Ensuite, il s'applique à attraper ces objets avec sa bouche, puis, finalement, explore les interactions vocales avec les autres robots. Pourtant, les ingénieurs n'ont programmé aucune de ces activités, ni *a fortiori* la chronologie de leur apparition.

COMMENT UN ROBOT PEUT-IL ÊTRE CURIEUX ?

Le modèle de curiosité du robot lui permet de trouver de nouvelles « niches de progrès ». Imaginons un environnement avec quatre types de contextes sensoriels et moteurs pour le robot : il peut dormir, bouger une patte, taper dans une balle sans bouger ou faire du scooter.



Si l'on forçait le robot à se concentrer sur chacune de ces activités séparément, on mesurerait l'évolution de son erreur en prédition dans chaque contexte (à gauche). Dans la situation a (faire du scooter), l'erreur en prédition est toujours élevée, peut-être parce que cette situation est trop compliquée pour le système d'apprentissage. Dans la situation d (dormir), l'erreur est toujours basse et ne change pas (cette situation est facile et donc peu intéressante pour le système d'apprentissage). Dans les situations b (taper dans une balle) et c (bouger une patte), l'erreur en prédition est importante au départ, mais diminue ensuite. En pratique, le robot placé dans cet environnement ignore les quatre types de situations et, *a fortiori*, les courbes d'apprentissage correspondantes. Au départ, il explore aléatoirement son environnement, découvrant qu'il existe des situations différentes et évalue l'intérêt de chacune en termes de réduction potentielle de ses erreurs en prédition.

Les courbes du temps passé à explorer chaque situation (à droite) montrent que le robot évite les situations a (trop compliquée) et d (trop simple), qui ne permettent pas de progrès en apprentissage. Il les explore cependant de temps en temps et par hasard pour vérifier qu'elles restent peu intéressantes. À l'inverse, il se consacre à la situation c pour laquelle ses prédictions s'améliorent le plus vite. Après un certain temps, la situation c est maîtrisée et, par conséquent, prédictible : il l'abandonne. Il consacre alors l'essentiel de son temps à la situation b qui, à ce stade de son développement, lui procure le plus de progrès en apprentissage.

Ces paliers, qui traduisent une auto-organisation, résultent des interactions dynamiques entre la curiosité, l'apprentissage et les propriétés du corps et de l'environnement. Quand on répète plusieurs fois la même expérience avec des paramètres identiques, ce sont approximativement les mêmes phases qui apparaissent le plus souvent. Mais, quelquefois, des individus intervertissent des étapes, voire acquièrent des savoir-faire différents.

C'est dû à de petites contingences, à des faibles variations de l'environnement physique et au fait que la dynamique du développement dispose de ce que les mathématiciens nomment des attracteurs. Ce sont des états particuliers vers lesquels un système évolue spontanément dès qu'il se trouve à proximité.

Ces expérimentations robotisées nous aident à comprendre le développement. Elles permettent de modéliser les mécanismes d'un apprentissage guidé par la curiosité. Elles montrent aussi comment, sur le long terme, des processus développementaux fondés sur la curio-

sité peuvent s'auto-organiser et gagner en complexité, sans pour autant avoir un plan prédefini. Dans ce cadre, les différences individuelles deviennent des phénomènes émergents. On comprend alors comment des processus développementaux peuvent varier selon les contextes, même avec une trame identique sous-jacente.

© Inria/Flowers



FIGURE 3.4 — *POPPY a été développé à l'INRIA. Ce robot, en open source (tous ses composants sont publics), aide à explorer les mécanismes du développement cognitif des enfants. Il est aussi utilisé pour l'éducation des écoliers et des collégiens.*

D. COMPRENDRE ET CRÉER

La compréhension du développement infantile est un des plus importants défis de la science. Une difficulté notable est que ce développement résulte d'interactions de très nombreux mécanismes à diverses échelles spatiales et temporelles. Une approche systémique et constructiviste semble nécessaire.

Les physiciens l'ont compris il y a longtemps quand ils se sont confrontés aux systèmes complexes. Pour les étudier, ils ont bâti des modèles formels avec lesquels ils simulent des aspects de la réalité. Le physicien prix NOBEL Richard FEYNMAN l'a ainsi exprimé : « On ne peut pas comprendre ce que l'on ne peut créer. » Une telle approche a désormais sa place dans les sciences du développement. Avec les robots, le corps devient une variable expérimentale, un élément que l'on peut systématiquement modifier de façon à étudier les effets de ces changements sur le développement. Ce fut longtemps un rêve, c'est aujourd'hui possible.

POUR ALLER PLUS LOIN...

- ▶ Comment se déplacent les robots ? (partie 1 et partie 2), Thierry FRAICHARD, CANALU (coll. INRIA Science Info Lycée Profs), 2013;
- ▶ Des ordinateurs aux robots : les machines en informatique, Pierre-Yves OUDEYER, 1,2,3... Codez, 2016 (ouvrage en accès libre après connection gratuite);
- ▶ Poppy Éducation, plateforme robotique open-source fondée sur l'impression 3D;
- ▶ Commander les robots, les maths et l'informatique qui sont sous le capot, par Bernard ESPIAU, Interstices, 26 mai 2008;
- ▶ 50 nuances de robots mous ! ou robots encore plus inouïs qu'en science-fiction, Christian DURIEZ, BINAIRE, 28 mai 2015.

3 Impact sociétal de la robotique

L'utilisation des robots ne se limite plus à des espaces industriels contrôlés. Ils sont intégrés dans nos environnements quotidiens et y interagissent de manière plus ou moins prédictible. Ce sont donc des objets auxquels les lois actuelles en matière de responsabilité, par exemple, ne sont plus complètement adaptées. Décryptage...

3.1 Robotique et éthique

À PROPOS DE L'INTERVENANT

Jean-Pierre MERLET est directeur de recherche INRIA, responsable scientifique de l'équipe *Hephaïstos* et membre Fellow de l'IEEE. Ses recherches actuelles en robotique portent sur le développement de plateformes robotisées destinées aux personnes âgées et handicapées, pour les aider à garder une certaine autonomie et autres services à la personne. Il a aussi apporté des contributions majeures en robotique de précision à hautes performances (robots parallèles) et de même qu'en prospective scientifique liée à la robotique. Il est aussi un chercheur très engagé, tout particulièrement en ce qui concerne la défense d'une recherche publique de qualité et les liens entre la recherche et les grands enjeux éthiques et environnementaux.

Dans l'imaginaire collectif, la robotique est liée aux films, aux véhicules autonomes et aux robots plus ou moins humanoïdes, compagnons capables de tenir un discours intelligent et réalisant des tâches relativement complexes. En fait, il n'en est rien et probablement dans un futur lointain, les robots, outils de recherche pour autant admirables, ne sont encore pas prêts pour endosser ces rôles; bon nombre de verrous scientifiques restent à lever auparavant, sans compter les problèmes de consommation énergétique.

Il est néanmoins vrai que depuis quelques années le robot a franchi la barrière du monde industriel où il existait à part pour se rapprocher de l'humain. C'est d'ailleurs déjà le cas au regard par exemple des robots aspirateurs. C'est la tendance de la robotique actuelle : le robot qui va à la rencontre de l'humain. Mais cette proximité va induire des risques et des problèmes légaux, de responsabilité et d'éthique.

3.1.1 Exemple du véhicule autonome

Pour l'instant, le véhicule autonome n'est pas prêt à répondre à toutes les situations et a encore des problèmes de prises décision, néanmoins dans un environnement bien structuré et avec une bonne cartographie des lieux, il y a désormais la possibilité d'autonomie du véhicule. Cependant, cela ne va pas sans poser des problèmes d'éthique.

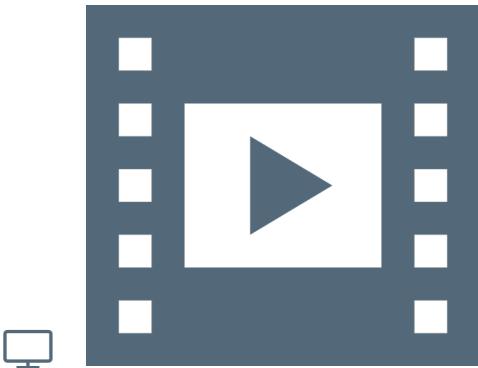
En effet, la problématique traditionnelle est de remarquer que fatalement, un jour, le véhicule se retrouvera dans une situation de choix cornélien entre deux options tragiques. D'un point de vue mécanique, il doit freiner en urgence, mais il est obligé de diriger soit vers un piéton ou un groupe de piétons, soit vers une barrière ou un mur. Chacune des options peuvent conduire à une issue tragique pour un ou plusieurs des protagonistes. Quelle est la « bonne » solution ?

Qui sauve-t-on ? Le(s) passager(s) du véhicule ou le(s) piéton(s) ? Certains prétendent que dans la mesure où le passager/conducteur a payé pour avoir une sécurité accrue dans son véhicule, il doit en bénéficier et être sauvé coûte que coûte. On peut déplacer le problème en considérant toujours un seul conducteur et un groupe de piétons. Que fait-on ? À partir de combien de piétons est-on en mesure de sacrifier le conducteur et non plus les piétons ?

Une telle illustration est donnée par les scientifiques pour imager les futurs possibles avec les avantages et les inconvénients. Leur rôle n'est alors pas de trancher mais c'est à la société toute entière qu'il appartient de faire des choix, de formuler les réactions à envisager face à des événements inadmissibles.

3.1.2 Collaboration homme-robot

Le robot se rapproche de l'homme dans une fonction d'aide et d'appui, mais aussi pour observer le comportement de l'humain dans certaines circonstances et gérer des situations d'urgence.



VIDÉO 3.4 – Éthique et robotique.



Véhicule autonome futuriste.

Ces robots d'assistance commencent à investir le monde industriel sous couvert de l'appellation de « cobots ». Il s'agit d'un système qui promeut la *collaboration* en l'homme et le robot. Le robot va se charger des tâches les plus dangereuses ou les plus pénibles comme soulever de lourdes charges. En revanche, l'homme possède une expertise du geste que le robot n'atteindra jamais, mais a des difficultés vis-à-vis de l'applicabilité de la tâche. Faire cohabiter ces deux approches semble relever du bon sens.

Dans le cas de la robotique industrielle, la législation est assez claire et bien balisée en terme de gestion des risques. Par contre, l'aspect observationnel pose des problèmes d'éthique car la machine qui aide l'homme va être capable de donner des informations sur la manière dont il effectue son travail. On peut arguer que c'est une bonne chose et que les indicateurs de performance chers à nos managers vont être utiles au développement de l'entreprise. En même temps, ces indicateurs sont très souvent établis avec une machine qui n'est pas complètement au courant de l'ensemble du contexte de la réalisation de la tâche. Il peut donc y avoir des raisons objectives à faire quelque chose moins vite que d'habitude parce qu'il y a eu un facteur de complexité supplémentaire qui ne sera pas mesuré par les indicateurs. Donc, ça pose un tout petit peu de soucis d'évaluer les personnes par des machines qui n'ont pas forcément toutes les capacités pour appréhender la situation à sa juste valeur.

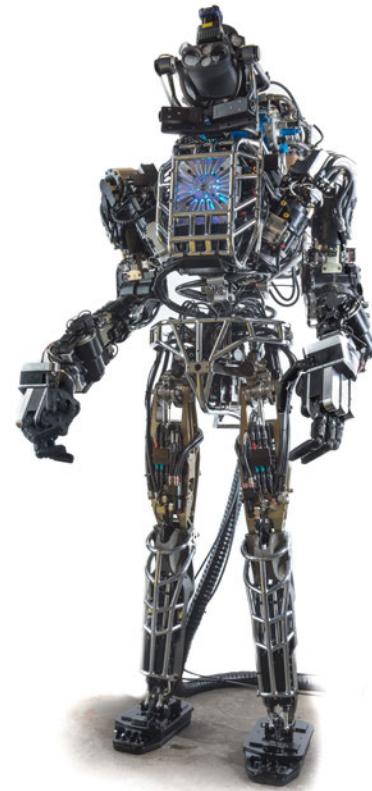
En parallèle, il y a bien sûr le problème de l'observation de l'homme par la machine. Est-ce que le technicien qui fait le geste va accepter d'être surveillé par cette machine ? Qu'est-ce qu'on lui dit à ce technicien de ce que fait la machine, de ce que mesure la machine ? Dans quel objectif a-t-il accès à ces données ? Sont-elles anonymisées ? Servent-elles simplement à faire des traitements statistiques ? Est-ce qu'elles vont sortir de l'entreprise ? Tout cela relève de problèmes d'éthique sur lesquels il faudra bien un jour se pencher.

3.1.3 Enjeux de la robotique médicale

On peut également considérer le domaine de la robotique médicale où la motivation première des machines est l'assistance à la personne. L'objectif est aussi de s'adresser ici à la communauté médicale pour évaluer l'état de santé d'un patient, être capable d'observer son comportement et de signaler des situations d'urgence, comme par exemple de détecter l'apparition d'une pathologie émergente parce que le dispositif d'assistance va relever et mesurer une anomalie qui va attirer l'attention d'un médecin.

Les avantages sont là encore nombreux... Tout autant que les problèmes d'éthique. D'abord, que ce passe-t-il lorsque le dispositif d'assistance blesse le patient, par exemple une machine puissante destinée à lever des personnes ? Le risque est donc accru et pose le problème de l'acceptation. Jusqu'où tolère-t-on qu'une machine prenne en charge la mobilité d'une personne sachant que cette machine fatalement un jour présentera un défaut ?

Là encore, de même qu'en cobotique, on retrouve aussi la problématique de l'observationnel de l'individu. Certes, on part d'un bon sentiment ; on mesure des paramètres d'état de santé. Mais évidemment, ces données peuvent avoir un intérêt économique très fort auprès de personnes extérieures qui vont par exemple profiter de l'état de faiblesse des personnes âgées pour essayer de leur vendre de l'assurance-vie, de la téléassistance ou tout ce qu'on peut imaginer comme solutions dont l'efficacité reste encore à prouver mais qui présentent pour les entreprises un intérêt économique considérable.



Atlas (2013), robot androïde de BOSTON DYNAMICS.

Donc, qu'est-il acceptable en termes d'observation ? Qu'est-ce qu'on va donner comme information au sujet qui est instrumenté ou qui utilise un objet de sa vie quotidienne pour évaluer son état ? Qu'est-ce qu'on lui dit de ce qu'on observe et en des termes suffisamment simples pour que cela soit compréhensible par tout utilisateur. Cela pose déjà un petit problème sur la sécurité des informations. Ces informations, on peut certes les anonymiser¹ et on peut essayer de les crypter. Malgré tout, à un moment donné, il faudra bien les transmettre. Transmettre à qui ? Dans un cadre privilégié d'une relation médecin-patient, donc juste vers le médecin ? Plus largement, ces données ont pourtant un intérêt pour la communauté médicale puisqu'on peut observer des tas de choses sur un grand nombre d'individus. Ce faisant, cela peut aider à faire progresser des traitements en se fondant justement sur ces analyses de grandes données. Donc on est dans une espèce d'antinomie où les données devraient être les plus confidentielles possible et, en même temps, devraient être utilisées par au moins un public très averti pour en tirer des bénéfices sur la médecine.

Après, il y a la sécurité des transmissions. Alors d'abord, là aussi, il faut démythifier un tout petit peu ce monde hyper-connecté de manière extrêmement fiable qu'on nous présente. C'est loin d'être la réalité et il n'y a pas besoin d'aller très loin autour de soi pour trouver une zone blanche. Imaginez une personne âgée qui chute, il faut absolument transmettre cette information pour la sauver. Mais là, on peut être dans un contexte où le réseau classique d'informations ne fonctionne pas. Cela veut dire que là aussi, il faut inventer des solutions alternatives aux réseaux Wifi ou hertzien qui permettront la transmission de cette information. Donc trouver des solutions alternatives de transmission de l'information tout en garantissant la sécurité, qu'un extérieur ne puisse pas intercepter ces informations pour les exploiter de manière frauduleuse.

De tous ces problèmes soulevés par la robotique, les experts scientifiques sont capables de voir et d'anticiper leurs avantages et leurs inconvénients, mais c'est à l'ensemble de la société, au législateur, aux citoyens de décider de ce qui sera admissible ou non.

GLOSSAIRE CONTEXTUEL

ROBOTIQUE — Appareil effectuant, grâce à un système de commandes automatiques, une tâche précise pour laquelle il a été conçu dans le domaine industriel, scientifique ou domestique. Le mot « robot » a été créé (dans une œuvre de fiction) à partir du mot tchèque « *robota* » qui signifie « travail, besogne, corvée ».

VERROU SCIENTIFIQUE — On parle de verrou scientifique pour un résultat pas encore obtenu mais dont dépendent d'autres résultats scientifiques ou techniques. Cela constitue une contrainte à lever pour qu'un domaine de recherche avance.

ÉTHIQUE — L'éthique est une réflexion de fond sur un sujet, qui permet ensuite de proposer des normes, limites et devoirs, dans le respect de certaines valeurs. La morale désigne l'ensemble de ces règles établies.

COBOT — S'emploie comme contraction de « robot collaboratif » et désigne un robot en interaction réelle, directe ou télé-opérée avec un opérateur ou un utilisateur humain.

ANONYMISER — Garantir qu'un jeu de données contenant des informations personnelles ne permet pas d'identifier des individus, donc évite d'accéder à des informations privées et confidentielles les concernants.

¹. Voir ??, « Numérique et sciences du réel ».

CRYPTER — Anglicisme employé pour le mot *chiffrer*, c'est-à-dire opération qui vise à rendre secret un document par *chiffrement*, pour que sa compréhension soit impossible à toute personne qui n'a pas la clé de (dé)chiffrement.

3.2 Informaticiens et éthique du numérique

Que deviendra la notion de vie privée dans notre société numérique ? L'hypermnésie et l'hyper-connectivité du net sont-elles des facteurs d'asservissement ou de libération de l'homme ? Quelle sera notre responsabilité vis-à-vis de robots commandés par la pensée, quelle sera notre cohabitation avec les robots ? Construire des robots ressemblant à l'homme est-il tabou ? Faut-il souhaiter ou redouter le transhumain, cet hypothétique homme augmenté de capacités intellectuelles et physiques jusqu'à prendre notre relai dans l'évolution ? Peut-on sans précautions utiliser des données personnelles, génétiques ou comportementales, à des fins de recherches ?

Cet échantillon d'interrogations — dont certaines relèvent encore de la fiction — illustre l'ampleur et la diversité des questions éthiques que pose l'explosion du numérique.

Il serait présomptueux de vouloir traiter ici ces questions, d'autant que l'on ne peut pas en débattre entre informaticiens seulement ; le propos est plutôt d'esquisser comment le monde scientifique les aborde actuellement. Au delà des seuls spécialistes, l'accent est mis sur la nécessaire approche décloisonnée des questions éthiques et la nécessaire inscription de celles-ci dans l'espace public.

ÉTHIQUE ET DÉONTOLOGIE

En gros l'éthique — qu'elle soit générale ou appliquée à un domaine — relève d'abord de la philosophie et l'humain est en son centre. L'éthique se définit classiquement comme la science de la morale.

La déontologie, qui n'est pas notre sujet ici, définit de manière plus opérationnelle les pratiques d'une profession, en accord avec l'éthique et le droit. La plus connue est la déontologie médicale. En informatique, le CIGREF (grandes entreprises utilisatrices) et le SYNTEC (SSI) ont défini leur code de déontologie. La déontologie engage comme le serment d'Hippocrate. L'ANR, agence nationale de projets scientifiques, affiche pour sa part une charte qui énumère les éléments suivants :

- Développer une recherche sérieuse et fiable ;
- Honnêteté dans la communication ;
- Objectivité ;
- Impartialité, indépendance ;
- Ouverture et accessibilité ;
- Devoir de précaution ;
- Équité dans la fourniture de références et de crédits ;
- Responsabilité vis-à-vis des scientifiques et chercheurs à venir.

Dans le monde anglo-saxon, on parle plus volontiers d'intégrité (*integrity*) qui met l'accent sur la responsabilité individuelle de comportement. Le comité d'éthique du CNRS (COMETS) vient dans cet esprit d'éditer un guide pour promouvoir une recherche intègre et responsable.

UNE APPROCHE SCIENTIFIQUE NÉCESSAIREMENT OUVERTE

Partout, les réflexions éthiques mobilisent le regard croisé des philosophes, historiens, sociologues, juristes voire économistes. On peut même dire « qu'élargir ses horizons » est inhérent à la démarche éthique, car on ne peut en général pas isoler les réflexions sur une science ou une technologie. Par exemple, en informatique, les considérations

Note de la rédaction

Texte rédigé par Max DAUCHET et publié sur *Binaire* — blog du numérique du journal *LE MONDE* — le 11 mars 2014.



sur le *Big Data* ou sur l'anonymat ne peuvent être considérées que dans le contexte sociétal.

DES SUJETS INCONTOURNABLES

Dans le but louable de financer équitablement les cultes, les Pays-Bas avaient mis en cartes perforées IBM les données confessionnelles de leur population dès les années trente, ce qui servit en 1940 les funestes desseins des envahisseurs nazis. Notons que ce pays intègre maintenant tout particulièrement la préoccupation éthique dans ses programmes scientifiques. Certes, on ne peut pas incriminer les seules technologies, la délation par la peur et l'oppression est arrivée ailleurs au même résultat avec du papier et des crayons seulement. Il reste qu'au-delà des controverses, on voit que l'on peut difficilement se laver les mains de tels sujets : *in fine*, il s'agit des rapports entre la démocratie et les totalitarismes et, de l'avenir de notre monde.

Depuis le procès des médecins de Nuremberg et maintenant avec les possibilités ouvertes par la biologie et la médecine, la bioéthique occupe le devant de la scène en éthique appliquée. Cependant les débats éthiques s'élargissent et la sécurité alimentaire, l'environnement et le numérique suscitent à leur tour des questionnements. Ainsi au niveau européen le réputé appel à projets individuels de l'*European Research Council* (ERC) compte 105 occurrences de "ethic(s)" ! De plus, les candidats doivent remplir un questionnaire éthique de vingt-six items, dont une douzaine est susceptible de concerner le numérique, notamment à travers l'usage de données personnelles (dont génétiques ou biométriques), les neurosciences, les technologies pour la santé, l'usage militaire ou encore la vie privée (la surveillance et maintenant la sous-veillance, qui consiste en la possibilité pour chacun de mettre instantanément sur le *Net* tout ce qu'il perçoit ou que son *smartphone* capte des personnes qu'il croise).

DES PERCEPTIONS VARIABLES DE PAR LE MONDE

Une vision de l'homme inspirée des religions révélées, celles des fils d'Abraham, peut percevoir le transhumanisme comme une transgression. Un asiatique influencé par le shintoïsme peut par contre concevoir l'homme comme participant à un tout dans un continuum entre la vie, la nature et l'artefact et ne manifester aucune appréhension à l'égard des humanoïdes. Dans nos sociétés, la confiance en la science comme vecteur de progrès s'effrite parfois face à l'ambivalence des technologies qui envahissent notre quotidien. Ainsi on peut voir dans les technologies numériques un facilitateur d'épanouissement et de démocratie ou, à l'opposé, un instrument mu par le profit qui accroît les inégalités entre les hommes. Les compromis entre respect de la vie privée et sécurité sont perçus différemment selon les continents ou les aspirations politiques... Plus généralement, une étude norvégienne de 2010² portant sur trente-quatre pays représentatifs de la diversité de la planète met en évidence de grandes diversités de perception des sciences et des technologies selon les continents, les cultures, la richesse par habitant ou encore le genre. Un maillage sans frontières des réflexions est donc nécessaire afin d'avoir pleinement conscience de la relativité des préconisations que l'on peut formuler à l'attention d'une nation ou d'une communauté — ainsi en Californie les journaux parlent le plus souvent de l'informatique pour souligner de nouveaux systèmes, des réussites, alors qu'en France on va insister sur les risques de pédophilie.

UN PAYSAGE EN CONSTRUCTION

Face à ces enjeux les initiatives se multiplient, principalement sous l'impulsion des États-Unis d'une part et de l'Europe d'autre part. Sur le

². ROSE, *Relevance Of Science Education*.

plan scientifique, la bioéthique naturellement, mais aussi l'environnement et la sécurité alimentaire provoquent des débats de société, davantage que le numérique, car ces secteurs sont perçus comme conditionnant de manière intrusive le devenir biologique de notre espèce.

LES COMITÉS D'ÉTHIQUE SCIENTIFIQUE — Ce sont généralement des instances consultatives, indépendantes, que l'on peut saisir ou qui s'auto-saisissent de sujets éthiques, et qui fournissent des préconisations. Ils ont un rôle de réflexion et sensibilisation amont.

En Europe, l'*European Group of Ethics* (EGE) joue ce rôle sur tout le spectre scientifique. La Commission Européenne est certes prolixe en tous domaines, mais on soulignera quand même l'abondance de la documentation sur l'éthique en général, et sur le numérique en particulier. Cette abondance semble viser davantage la sensibilisation des chercheurs que la construction d'une vision proprement européenne.

En France, le plus ancien et le plus en vue est le Ccne, créé en 1983 à l'initiative de François MITERRAND. Ce comité est placé auprès du Premier ministre, et sa composition garantit la représentation des grands courants philosophiques et religieux. Le CNRS dispose pour sa part depuis 1994 du COMETS. L'INRA et le CIRAD ont fusionné leurs comités.

Concernant l'informatique, la Cerna (Commission de réflexion sur l'éthique de la recherche en sciences et technologies du numérique d'Allistene) a été créée fin 2012 sous l'impulsion de l'INRIA et du CNRS par Allistene, l'alliance des sciences et technologies du numérique qui réunit le CEA, la CGE, le CNRS, la CPU, INRIA et l'Institut MINES-TÉLÉCOM. Le point de vue y est celui de la recherche et non des usages.

Il s'ajoute encore au paysage des groupes de travail qui émergent à l'initiative d'établissements ou de groupes d'établissements, comme par exemple le groupe Prométhos en éthique de l'innovation sur le plateau de Saclay.

LES COMITÉS OPÉRATIONNELS D'ÉTABLISSEMENT — Ils traitent les questions engageant leur responsabilité à travers des projets ou la déontologie des personnels.

LES INSTANCES DE VALIDATION ET CERTIFICATION (RESPECT DE NORMES ÉTIQUES) — De plus en plus d'institutions demandent une certification de conformité éthique des projets de recherche attestée par un *Institutional Review Board* (IRB), notamment pour les recherches impliquant l'homme.

LES ESPACES DE DÉBAT PUBLIC — La France dispose depuis 1995 d'une Commission nationale du débat public (CNDP), créée dans le cadre de la loi relative au renforcement de la protection de l'environnement, dite loi Barnier. Cette commission, aux consultations fort diverses, a notamment (mal)traité des nanotechnologies en 2010.

À un niveau intermédiaire entre le débat public et le cénacle de spécialistes, on peut citer les espaces éthiques régionaux qui se mettent en place à l'initiative du Ccne. Centrés sur la pratique hospitalière, ils associent des représentants des usagers.

Notons aussi que face au trouble suscité par la biologie de synthèse, le ministère de l'enseignement supérieur et de la recherche a confié en 2012 au Cnam la création d'un observatoire de la biologie de synthèse, chargé d'informer le grand public et d'échanger avec lui.

LES CONFÉRENCES ET SYMPOSIUMS SCIENTIFIQUES — Il est naturel que des manifestations scientifiques accompagnent la montée des préoccupations éthiques, dans le numérique comme ailleurs. Ainsi l'Ieee, maintenant association internationale des professionnels

du secteur numérique et dont les racines remontent à 1884 et l'avènement de la fée électricité, lance l'année prochaine un symposium sur l'éthique appelé à devenir annuel.

Deux conférences internationales de recherche sont consacrées depuis une vingtaine d'années aux différents aspects éthiques liés à l'informatique, CEPE et ETHICOMP. À l'initiative de la CERNA, elles se tiendront cette année conjointement et pour la première fois en France.

D'une manière générale, les Français sont peu présents dans ce genre de manifestation. Ainsi, lors de la *Third World Conference on Research Integrity* (WCRI) qui s'est tenue en 2013 à Montréal, il n'y avait que quatre français sur 500 participants. Ce manque d'appétence de notre monde académique est peut-être dû à la faible reconnaissance dans notre pays des investissements dans les questions éthiques et plus généralement les questions interdisciplinaires. Peut-être aussi que le regard fixé dans le rétroviseur sur notre rôle phare du temps des Lumières, nous percevons avec circonspections les nouveaux espaces éthiques qu'ouvrent les technosciences. Nuançant ce dernier propos, de nouveaux lieux de réflexions émergent dans notre pays, notamment au sein d'associations comme la FING (Fondation Internet nouvelle génération), Renaissance numérique, la Quadrature du net. Par ailleurs, l'AFIA (Association française d'intelligence artificielle) a consacré un récent bulletin à un « Dossier Éthique et IA », et la SIF (Société informatique de France) a reproduit une première version du présent propos dès le numéro deux de son bulletin intitulé 1024.



En conclusion, il importe que les scientifiques contribuent aux débats sur l'éthique des nouvelles technologies, et du numérique en particulier, faute de quoi les espaces laissés en friches pourraient être investis par des obscurantistes ou des aventuriers. Nous devons aussi veiller dans notre pays à dépasser nos vieilles habitudes de « s'affronter d'abord, débattre ensuite », car aborder les sujets éthiques en termes de pro- versus anti-technologies nous ferait passer tragiquement à côté du sujet. Et pour cela sensibilisons, informons, formons à commencer par la jeunesse.

POUR ALLER PLUS LOIN...

- ▶ Des robots et des humains, Jean-Pierre MERLET, INTERSTICES, 20 février 2015 et BINAIRE, 02 mars 2015;

- ▶ Informatique et éthique, Serge ABITEBOUL et Gilles DOWECK, Podcast Science #166, 11 juillet 2016 (160 minutes);
- ▶ Le dilemme macabre de la voiture automatique : tuer un piéton ou le passager? David LAROUSSE, LE MONDE SCIENCE ET TECHNO, 23 juin 2016;
- ▶ L'emphatie des robots, où comment démythifier cette idée de personnalisation des robots, tout en montrant comment simuler cela de manière très sophistiquée, Laurence DEVILLERS, Binaire, 16 septembre 2016;
- ▶ Quel impact de la révolution robotique sur le droit français? texte d'analyse, Alexandre MANDIL, GLORIEUSE FRANCE DROIT & INTELLIGENCE ÉCONOMIQUE, 21 février 2016.

4 Deep learning

L'apprentissage automatique (*machine learning*) correspond à des algorithmes qui ajustent les paramètres de leurs calculs en fonction des exemples qui leur sont donnés. Cela permet d'adapter leur fonctionnement aux données fournies. L'apprentissage profond (*deep learning*) est une architecture qui associe en cascade plusieurs couches d'algorithmes de ce type pour hiérarchiser le problème et obtenir des performances bien plus importantes. Ce n'est pas simple! Mais à expliquer, plus sûrement...

4.1 Comprendre le deep learning

Les concepts du *machine learning* et du *deep learning* sont directement abordés par la vidéo de « Science étonnante » proposée par David LOUPRE. Les contenus sont issus d'un livre de vulgarisation scientifique : « Mais qui a attrapé le bison de HIGGS ? aux éditions Flammarion. Un billet vient compléter la vidéo en abordant les concepts sous-jacents n'ayant pas été intégrés à la vidéo.

On peut suivre son partage de grains de « science étonnante » sur son [blog](#) et son canal vidéo.

GLOSSAIRE CONTEXTUEL

MACHINE LEARNING — L'apprentissage automatique ou *machine learning* correspond à des algorithmes qui ajustent les paramètres de leurs calculs en fonction des exemples qui leur sont donnés. Cela permet d'adapter leur fonctionnement aux données fournies en entrée d'algorithme.

DEEP LEARNING — L'apprentissage profond ou *deep learning* est une architecture qui associe en cascade plusieurs couches d'algorithmes de ce type pour hiérarchiser le problème et obtenir des performances bien plus importantes.

NEURONE ARTIFICIEL — Un neurone artificiel est une fonction qui combine des entrées pour calculer si la valeur de la sortie est plutôt élevée ou basse au regard d'un seuil défini, son calcul étant ajusté par des paramètres.

RÉSEAU DE NEURONES ARTIFICIELS — Un réseau de neurones fait référence à l'assemblage d'un grand nombre de neurones (sous-entendu artificiels) pour permettre de faire un calcul entrée/sortie très sophistiqué.

À PROPOS DE L'INTERVENANT

David LOUPRE, normalien et docteur en physique théorique, est chercheur dans l'industrie des matériaux. Passionné de culture scientifique, son parcours lui a permis de s'intéresser à de nombreux domaines de la science (physique fondamentale, mathématiques appliquées, physico-chimie des matériaux, thermique, mécanique, etc.). Il se qualifie de « responsable R&D (le jour), vulgarisateur scientifique (la nuit) ».



VIDÉO 3.5 — Deep learning.

GPU (GRAPHICAL PROCESSING UNIT) — Un GPU est un processeur supplémentaire dans nos ordinateurs dans lequel tous les calculs liés aux opérations graphiques d'affichage sont pré-cablés pour accélérer le traitement.

VISION PAR ORDINATEUR — Un ensemble d'algorithmes qui partent des valeurs des pixels des images pour en extraire des caractéristiques, localiser les objets qui y sont vus et les étiqueter.

4.2 Apprentissage automatique : pas à pas!

Note de la rédaction

Texte rédigé* par Colin DE LA HIGUERA et publié sur Binaire — blog du numérique du journal LE MONDE — le 23 juin 2015.

*Les éléments liés à une actualité ont été enlevés pour alléger le texte.

APPRENTISSAGE AUTOMATIQUE DITES-VOUS ?

Il est très probable qu'à l'heure où vous lisez ces lignes, vous aurez utilisé le résultat d'algorithme d'apprentissage automatique plusieurs fois aujourd'hui : votre réseau social favori vous a peut-être proposé de nouveaux amis et le moteur de recherche a jugé certaines pages pertinentes pour vous mais pas pour votre voisin. Vous avez dicté un message sur votre téléphone portable, utilisé un logiciel de reconnaissance optique de caractères, lu un article qui vous a été proposé spécifiquement en fonction de vos préférences et qui a peut-être été traduit automatiquement.

Et même sans avoir utilisé un ordinateur, vous avez été peut-être écouté les informations : or la météo entendue ce matin, la plupart des transactions et des décisions boursières qui font et défont une économie, et de plus en plus de diagnostics médicaux reposent bien plus sur les qualités de l'algorithme que sur celles d'un expert humain incapable de traiter la montagne d'informations nécessaire à une prise de décision pertinente.

De tels algorithmes ont appris à partir de données, ils font de l'apprentissage automatique. Ces algorithmes construisent un modèle à partir de données dans le but d'émettre des prédictions ou des décisions basées sur les données.

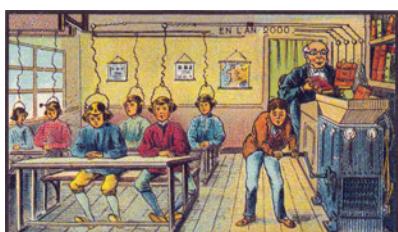
MAIS DEPUIS QUAND CONFIE T'ON CELA À DES ALGORITHMES ?

L'idée de faire apprendre la machine pour lui donner des moyens supplémentaires est presque aussi ancienne que l'informatique. C'est Alan TURING lui-même qui après avoir, en 1936, jeté les bases conceptuelles du calcul sur machine, donc de l'ordinateur, allait s'intéresser à cette possibilité. Il envisage en 1948 des « *learning machines* » susceptibles de construire elles-mêmes leurs propres codes.

Tout compte fait, c'est une suite logique de cette notion de machine de calcul dite universelle (c'est-à-dire qui peut exécuter tous les algorithmes, comme le sont nos ordinateurs ou nos smartphones). Puisque le code d'une machine, le programme, n'est qu'une donnée comme les autres, il est raisonnable d'envisager qu'un autre programme puisse le transformer. Donc pourquoi ne pas apprendre de nouveaux programmes à partir de données ?

Vue la quantité astronomique de donnée, on peut apprendre simplement en les analysant. Mais les chercheurs se sont rendus compte que dans ce contexte, un programme qui se comporte de manière déterministe n'est pas si intéressant. C'est le moment où on découvre les limites de ce qui est décidable ou indécidable avec des algorithmes. Il faut alors introduire une autre idée : celle d'exploration, de recours au hasard, pour que de telles machines soient capables de comportements non prévus par leur concepteur.

Il y a donc une rupture entre programmer, c'est-à-dire imaginer et implémenter un calcul sur la machine pour résoudre un problème, et



doter la machine de la capacité d'apprendre et de s'adapter aux données. De ce fait, on ne peut plus systématiquement prévoir un comportement donné, mais uniquement spécifier une classe de comportements possibles.

ALORS... ÇA Y EST ? L'INTELLIGENCE ARTIFICIELLE EST CRÉÉE ?

C'est une situation paradoxale, car le terme d'IA veut dire plusieurs choses. Quand on évoque l'intelligence artificielle, on pense à *apprendre, évoluer, s'adapter*. Ce sont des termes qui font référence à des activités cognitives qui nous paraissent un ordre de grandeur plus intelligentes que ce que peut produire un calcul programmé.

Et c'est vrai qu'historiquement, les premiers systèmes qui apprennent sont à mettre au crédit des chercheurs en IA. Il s'agissait de repousser les frontières de la machine, de tenter de reproduire le cerveau humain. Les systèmes proposés devaient permettre à un robot d'être autonome, à un agent de répondre à toute question, à un joueur de s'améliorer défait après défaite.

Mais, on s'accorde à dire que l'intelligence artificielle n'a pas tenu ses promesses. Pourtant ces algorithmes d'apprentissage automatique, une de ses principales composantes, sont bel et bien présents un peu partout aujourd'hui. Et ceci, au delà, du fait que ces idées sont à la source de nombreuses pages de science-fiction.



C'est peut-être notre vision de l'intelligence qui évolue avec la progression des sciences informatiques. Par exemple, pour gagner aux échecs il faut être bougrement intelligent. Mais quand un algorithme qui se contente de faire des statistiques sur un nombre colossal de parties défait le champion du monde, on se dit que finalement l'ordinateur a gagné « bêtement ». Ou encore : une machine qui rassemble toutes les connaissances humaines de manière structurée pour que chacun y accède à loisir sur simple demande, est forcément prodigieusement intelligente. Mais devant WIKIPÉDIA, qui incarne ce rêve, il est clair que non seulement une vision encyclopédique de l'intelligence est incomplète, mais que notre propre façon de profiter de notre intelligence humaine est amenée à évoluer, comme nous le rappelle Michel Serres.

PASSIONNANT ! MAIS [QUE SE PASSE-T-IL MAINTENANT] ?

L'apprentissage automatique est maintenant devenu une matière enseignée dans de nombreux cursus universitaires. Son champ d'application augmente de jour en jour : dès qu'un domaine dispose de données, la question de l'utilisation de celles-ci pour améliorer les algorithmes du domaine se pose systématiquement.

Mais c'est également un sujet de recherche très actif. Les chercheurs du monde entier, [étudient] parmi d'autres, les questions à suivre.

- ▶ Une famille d’algorithmes particulièrement efficace aujourd’hui permet d’effectuer un apprentissage profond (« *deep learning* »). De tels algorithmes simulent une architecture complexe, formées de couches de neurones artificiels, qui permettent d’implémenter des calculs distribués impossibles à programmer explicitement.
- ▶ L’explosion du phénomène du *big data* est un levier. Là où dans d’autres cas, la taille massive des données est un obstacle, ici, justement, c’est ce qui donne de la puissance au phénomène. Les algorithmes, comme ceux d’apprentissage profond, deviennent d’autant plus performants quand la quantité de données augmente.
- ▶ Nul doute que des modèles probabilistes de plus en plus sophistiqués seront discutés. L’enjeu aujourd’hui est d’apprendre en mettant à profit le hasard pour explorer des solutions impossibles à énumérer explicitement.
- ▶ la notion de prédition sera une problématique majeure pour ces chercheurs qui se demanderont comment utiliser l’apprentissage sur une tâche pour prévoir comment en résoudre une autre.
- ▶ Les applications continueront à être des moteurs de l’innovation dans le domaine et reposent sur des questions nouvelles venant de secteurs les plus variés : le traitement de la langue, le médical, les réseaux sociaux, les villes intelligentes, l’énergie, la robotique...

Il est possible — et souhaitable — que les chercheurs trouvent également un moment pour discuter des questions de fond, de société, soulevées par les résultats de leurs travaux. Les algorithmes apprennent aujourd’hui des modèles qui reconnaissent mieux un objet que l’œil humain, qui discernent mieux les motifs dans des images médicales que les spécialistes les mieux entraînés. La taille et la complexité des modèles en font cependant parfois des boîtes noires : la machine peut indiquer la présence d’une tumeur sans nécessairement pouvoir expliquer ce qui justifie son diagnostic : sa « décision » reposera peut-être sur une combinaison de milliers de paramètres, combinaison que l’humain ne connaît pas.

Or, quand votre médecin vous explique qu’il serait utile de traiter une pathologie, il vous explique pourquoi. Mais quand la machine nous proposera de subir une intervention chirurgicale, avec une erreur moindre que le meilleur médecin, sans nous fournir une explication compréhensible, que ferons-nous ? [...]

BIBLIOGRAPHIE CIRCONSTANCIÉE

- ▶ *Probabilistic machine learning and artificial intelligence*, Zoubin GHAHRAMANI. Nature 521, pp. 452–459.
- ▶ *Apprentissage artificiel*, Antoine CORNUÉJOLS et Laurent MICLET, EYROLLES, Collection Algorithmes, 2^e édition, juillet 2011.
- ▶ *Intelligence artificielle*, Stuart RUSSEL et Peter NORVIG. PEARSON EDUCATION, 2010.
- ▶ *Machine Learning*, Peter FLACH, CAMBRIDGE UNIVERSITY PRESS, November 2012.
- ▶ *Computing machinery and intelligence*, Alan M. TURING, MIND, 59, pp. 433–460, 1950.
- ▶ *L’apprentissage profond : une idée à creuser ?* Ikram CHRAIBI KAAOUD et Thierry VIÉVILLE, INTERSTICES, 29 avril 2016.
- ▶ *L’apprentissage automatique : le diable n’est pas dans l’algorithme*, Philippe PREUX, Marc TOMMASI, Thierry VIÉVILLE et Colin DE LA HIGUERA, Binaire, 29 juin 2015.
- ▶ *Comment donner un sens à l’image numérique ?* Jean PONCE, INTERSTICES, 9 décembre 2014.

POUR ALLER PLUS LOIN...

- ▶ Comment apprendre aux ordinateurs à comprendre des images, Fei-Fei Li, conférence TED — *Technology, Entertainment, Design* —, 2015;
- ▶ Conférence inaugurale et cours au Collège de France, Yann LE-CUN, 2015-2016.

5 Que faire de ces ressources ? Autoévaluation

Les questionnaires à choix multiple* — QCM — à suivre clôturent le présent chapitre « *Intelligence artificielle et robotique* » et correspondent à chaque sujet qui y est abordé.

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style `LATeX` usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

QUIZ 3 — ROBOTIQUE ET INTELLIGENCE ARTIFICIELLE 6 POINTS

QUIZ 3.1 — INTELLIGENCE(S) ARTIFICIELLE(S) 1 POINT

On parle aujourd'hui d'intelligences artificielles au pluriel, plutôt que « d'une » intelligence artificielle. Que signifie cette formulation ?

Le fait que les robots et les bots logiciels dotés d'une intelligence artificielle identique à l'intelligence humaine se multiplient.

Le fait qu'il y ait plusieurs niveaux dans l'intelligence artificielle, comme dans le règne animal.

Le fait que des méthodes radicalement différentes résolvent des problèmes très précis de manière cloisonnée.

C'est une simple typo qui a fini par faire le buzz.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 3.2 — INTELLIGENCE(S) ARTIFICIELLE(S) 1 POINT

Pourquoi l'arrivée de cobots change-t-il la donne juridiquement? Donner les deux réponses exactes.

De par leur interaction avec les humains au quotidien.

Car dotés d'une intelligence artificielle, ils deviennent des êtres à part entière.

De même que pour une « personne morale », le droit est plus clair en considérant les cobots comme sujet (et non objet) de droit.

Par définition, puisque le terme cobot veut dire robot conscient.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 3.3 — DEEP LEARNING 1 POINT

La puissance du *deep learning* est de pouvoir apprendre tous les paramètres des différentes couches de calcul qui permettent de calculer une sortie à partir des entrées (par exemple étiqueter une image). Mais qu'en est-il des données à fournir?

Ce qui est remarquable c'est qu'un tout petit ensemble de données suffit à cet apprentissage automatique.

Cela ne fonctionne que sur de très grands ensembles de données.

Si les données sont des images, une ou deux images suffisent, sinon il faut beaucoup de données.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 3.4 — ROBOT 1 POINT

L'origine du mot « robot » n'est pas anodine, elle montre un lien avec un vrai mythe humain. Quelles sont les trois réponses exactes ?

Cela vient du tchèque tchèque « *robota* » qui signifie « travail, besogne, corvée », esclave en fait.

C'est Isaac AZIMOV qui a créé le terme en même temps qu'il définissait les lois de la robotique.

Le mot a été employé pour la première fois dans une pièce de théâtre en 1920.

Le mythe initial fait référence à un être artificiel organique.

En fait, cela vient du français *rabet* et correspond au premier *rabet* automatisé, dans une scierie des Vosges en 1869.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 3.5 — MARCHE BIPÈDE 1 POINT

Est-il possible de simuler une forme de marche bipède sur un plan incliné avec un dispositif purement mécanique sans moteur, ni aucun contrôleur électronique ou informatique ? Cocher la réponse exacte.

Évidemment non, compte-tenu de la complexité de la tâche !

En fait oui, c'est un comportement émergent qui résulte des interactions de la mécanique avec l'environnement.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 3.6 — ENVIRONNEMENT ROBOTISÉ 1 POINT

Notre environnement va être de plus en plus robotisé. Quelle forme tangible va prendre cette robotisation à grande échelle ? Choisir la réponse la plus plausible dans l'état actuel de nos connaissances.

Des robots humanoïdes, qui seront à notre service.

Cela n'arrivera jamais, tout ça ne marche que dans les labos.

Ce sont les objets du quotidien eux-mêmes qui vont se doter de processus algorithmiques sophistiqués.

Ils sont déjà là, moi-même, n'en suis-je pas une ou un ?

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

FONDEMENTS DE L'INFORMATIQUE

131

CHAPITRE 4
CODAGE BINAIRE

163

CHAPITRE 5
LANGAGES DE L'INFORMATIQUE

CODAGE BINAIRE

ACE JOUR, L'INFORMATION NUMÉRIQUE S'AVÈRE BINAIRE. Il convient donc de déterminer ce que recouvre ce système pour ensuite détailler comment les sons, les images, les vidéos ou toute autre donnée sont traités par cet appareillage qui s'est invité dans le quotidien de la plupart des gens en une trentaine d'années : l'ordinateur.

Cela offre l'occasion d'aborder les différents formats de données et d'évoquer la manière de les stocker, de les classer, autant que d'y accéder une fois ces informations enregistrées en base de données.

1 Représentation de l'information

Notre environnement comporte une foultitude d'informations différentes, que ce soient des photographies, des bruits, la température ambiante, la quantité d'argent disponible sur notre compte bancaire, notre âge, des dates, etc.

Plus ou moins structurée, cette multitude de choses très diverses est, soit appréhendée directement par nos sens humains, soit captée au moyen d'instruments de mesure. Comment alors traduire ces informations pour qu'un ordinateur puisse les traiter ?

1.1 Électronique et logique

Afin de pouvoir « discuter » avec un ordinateur, on procède à une numérisation, c'est-à-dire un codage de l'information ou une interprétation en nombres que l'ordinateur est capable de manipuler.

Un ordinateur est un appareil électronique qui, de fait, fonctionne à l'électricité. Schématiquement, si le courant électrique ne passe pas dans un composant, cela définit le niveau 0 et, inversement, si le courant passe, cela détermine le niveau 1. Il en est de même de tous les composants constitutifs de l'ordinateur et celui-ci répond à un système binaire — état 0/Non/Faux versus état 1/Oui/Vrai — de l'information.

1.1.1 Système binaire

L'information binaire (0 ou 1), le plus souvent transmise par un courant électrique — parfois par des variations d'intensité lumineuse comme dans les fibres optiques — est représentée par une tension, proche

SOMMAIRE

1	Représentation de l'information
1.1	Électronique et logique 131
1.1.1	Système binaire ■ 1.1.2 Opérateurs et composants logiques
1.2	Nombre et texte 132
1.2.1	Attribution des nombres ■ 1.2.2 Codification du texte
2	Infographie et audionumérique
2.1	Photographie et dessin 139
2.1.1	Images matricielles ■ 2.1.2 Images vectorielles
2.2	Son et musique 140
2.2.1	Caractérisation et analyse des sons ■ 2.2.2 Quantification et échantillonnage
3	Manipulation des données
3.1	Compression 151
3.1.1	Problématique ■ 3.1.2 Images ■ 3.1.3 Vidéos ■ 3.1.4 Sons
3.2	Organisation et stockage 155
3.2.1	Type et codage ■ 3.2.2 Structure de données ■ 3.2.3 Représentation des données en mémoire
3.3	Base de données 157
3.3.1	Modèle relationnel et requête SQL ■ 3.3.2 Protection et partage
4	Que faire de ces ressources ? Quiz



VIDÉO 4.1 — Représentation binaire.



1. Bien que dans le même temps, un brevet européen ait été semble-t-il déposé (W).

2. Par extension, ce nom a été donné aux récepteurs radiophoniques portatifs fonctionnant à base de transistors.

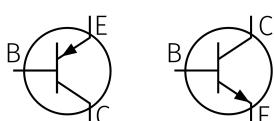


FIGURE 4.1 — Technologie des transistors, du type PNP (à gauche) et NPN (à droite). B : base – C : collecteur – E : émetteur.

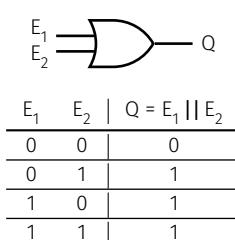


FIGURE 4.2 — Représentation et table de vérité de porte logique « OU » (notation étasunienne la plus usitée; voir comparatif avec la notation européenne).

de zéro volt pour l'état 0 ou bien voisine de cinq volts pour l'état 1. Les courants circulent surtout lors des changements d'état 0 → 1 et 1 → 0 (au-delà des courants de fuite), c'est la cause principale de consommation électrique; ils permettent de charger ou décharger les capacités des circuits électroniques.

C'est de là que provient le nom de « *bit* » pour « *Binary Digit* » en anglais. Donc, un bit vaut 1 ou 0 selon que le courant passe ou non. Les ordinateurs actuels, fondés sur l'électronique, répondent à ce canon. Peut-être qu'à l'avenir cela changera avec les perspectives à l'étude en photonique ou en mécanique quantique.

1.1.2 Opérateurs et composants logiques

À partir des bits l'enjeu est de conduire des opérations, autrement dit des calculs. L'opération la plus simple à réaliser est offerte par ce qu'on appelle un transistor. C'est un des composants élémentaires de l'électronique issu des laboratoires¹ BELL en 1948.

Les fonctions d'un transistor peuvent être l'amplification des courants électriques, la génération d'oscillations, la modulation ou la détection² de signaux. Un transistor est assimilable à un robinet qui laisse ou non passer le courant. Les transistors bipolaires à jonction — BJT —, longtemps dévolus aux systèmes analogiques sont disponibles depuis les années 1950. Ils sont obtenus en insérant un barreau semi-conducteur entre deux autres de signe opposé; ainsi, on distingue les transistors de type PNP — Positif–Négatif–Positif — et NPN — Négatif–Positif–Négatif. Ces derniers possédant de meilleures caractéristiques, ils sont plus répandus.

Actuellement, la microélectronique vise à réaliser des transistors CMOS — Complementary Metal Oxide Semi-conductor — qui consomment un courant le plus réduit possible quand on applique une tension sur leur entrée, justement pour limiter au maximum la puissance dissipée par ces transistors.

La technologie permet de réaliser des transistors très petits (de l'ordre de 5 nm). Des phénomènes complexes viennent alors se greffer y compris des courants non négligeables qu'il est difficile de contenir : la chasse au courant a toujours été un objectif et insister sur la notion de courant rend dès à présent explicite cet enjeu en matière d'énergie.

Beaucoup de choses sont dérivées des transistors, notamment des opérations sur les bits avec la notion de *porte logique*. Les portes logiques disposent en entrées de différentes arrivées de courant qui se combinent, pour offrir en sortie un résultat en fonction des arrivées. Elles sont appelées portes logiques car elles proposent l'ensemble des opérateurs logiques : « ET », « OU », « OU exclusif », « NON », etc.

Par exemple, une porte « OU » (cf. figure 4.2) reçoit deux entrées qui vont contenir du courant ou non. Elle va générer une sortie qui contient du courant si une des deux entrées ou si les deux entrées en possèdent (voir table de vérité).

Se fondant sur des combinaisons plus ou moins complexes de portes logiques et de composants électroniques associés, on peut élaborer des cellules ayant des fonctionnalités utiles pour le calcul : des additionneurs de bits, des bascules et registres à décalage, etc.

1.2 Données numériques et textuelles

En termes de représentation des nombres, se limiter au bit est bien entendu insuffisant; seules deux valeurs. Certes, si la numérisation constraint les possibilités, elle doit néanmoins permettre de conduire des

évaluations réalistes de l'environnement naturel. Deux aspects « productivistes » — à savoir hors perspective ludique — de l'utilité d'un ordinateur sont le calcul numérique — en substitution bien plus performante de la règle à calcul des écoliers des années 1950 — et le traitement de texte — en succession de la machine à écrire mécanique.

1.2.1 Assignation des nombres

Le principe du codage des nombres est simplement de considérer un assemblage de bits correspondant aux besoins de l'étendue et/ou de la précision de la représentation voulue.

Par conséquent, plutôt que le bit, l'unité de quantification courante est convenue à 8 bits, soit un *octet*³ — *byte* en anglais. Cela correspond à une puissance de deux facile à traiter par l'ordinateur et qui offre 256 valeurs, de 0 à 255 ($2^8 = 256$).

Comme application directe, on peut vouloir coder un âge qui, approximativement et pour le moment, se situe entre 0 et 127 ans. Pour ce faire il suffit⁴ de 7 bits ($2^7 = 128$, $2^6 = 64$ plus assez!).

Toutefois, pour coder une température, l'octet ne suffit plus. En effet, il faut pouvoir distinguer un nombre positif d'un nombre négatif, un nombre entier d'un nombre⁵ décimal... La représentation est plus subtile mais sur le principe reste la même : allouer suffisamment de bits et les agencer correctement pour réaliser l'opération.

Ces notions sont reprises par la suite, mais ce qu'il faut d'abord retenir est qu'un ordinateur est stupide : il sait manipuler les nombres, mais se fiche éperdument de leur signification. C'est au programmeur ou à l'utilisateur de fournir ces nombres de manière adaptée en entrée.

Par exemple, le calcul d'une vitesse doit se faire de manière adéquate en entrant les unités nécessaires au calcul : en m/s ou en km/h, ni l'une ou l'autre, ni l'une sans l'autre.

1.2.2 Codification du texte

Il existe diverses manières de coder du texte. Une première approche pourrait être d'associer bit à bit le dessin d'un glyphe en le décomposant sur une grille de bits (voir figure 4.3). On utilise un bit par case et on considère que si le bit est à zéro la case est noire, si le bit est à un elle est blanche. On décrit ainsi chaque point à l'écran.

Toutefois, la démarche précédente ne fonctionne pas, car en pratique il faut savoir, pour chaque lettre, la largeur et la hauteur à lui attribuer, autrement dit le nombre de cases horizontales et verticales à réserver. De plus, si on rajoute de la couleur, un seul bit par case ne suffit plus, cela devient rapidement ingérable de traiter facilement toutes les informations nécessaires. Ces questions d'affichage deviennent délicates et seront reprises par la suite.

Indépendamment de l'affichage, la représentation des lettres et autres caractères passe par un codage sous forme de nombres. Ainsi, à partir de spécifications normées, des tables de correspondances entre signe et code sont établies.

Originellement, le code qui a prévalu est le code ASCII, pour *American Standard Code for Information Interchange*. Comme son nom l'indique, il est bâtit pour l'anglais américain et codé initialement⁶ sur 7 bits, suffisant pour la ponctuation et l'alphabet latin sans accent (voir table 4.1). Avec l'essor des ordinateurs personnels (PC), le codage ASCII étendu, codé sur 8 bits, a été réalisé pour tenir compte des autres langues. De multiples moutures en sont déclinées et, pour les langues d'Europe de l'ouest comme le français, les normes ISO-8859-1 et ISO-8859-15 ont longtemps été employées (voir table 4.2).

³. Historiquement, ceci est aussi à mettre en relation avec le rapport entre capacité et coût de la mémoire.

⁴. Sur un octet, le huitième bit est mis à zéro.

⁵. Un nombre décimal est appelé nombre à virgule flottante ou par contraction un nombre « flottant ».

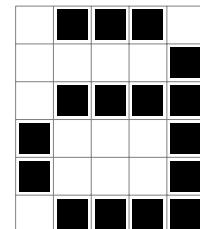


FIGURE 4.3 — Codage d'une lettre bit à bit.

⁶. En fait, le codage est sur 8 bits, le bit supplémentaire servant de paramètre de contrôle, mais les caractères en tant que tels sont codés sur 7 bits.

TABLE 4.1 — Table originelle de codage ASCII (7 bits).

TABLE DES CARACTÈRES ASCII CODÉS SUR 7 BITS							
DÉC.	BIN.	GLYPHE	DESCRIPTION	DÉC.	BIN.	GLYPHE	DESCRIPTION
0	0000000	NUL	Null	64	1000000	Ø	At Sign
1	0000001	SOH	Start of Header	65	1000001	A	Latin capital letter A
2	0000010	STX	Start of Text	66	1000010	B	Latin capital letter B
3	0000011	ETX	End of Text	67	1000011	C	Latin capital letter C
4	0000100	EOT	End of Transmission	68	1000100	D	Latin capital letter D
5	0000101	ENQ	Enquiry (End of Line)	69	1000101	E	Latin capital letter E
6	0000110	ACK	Acknowledge	70	1000110	F	Latin capital letter F
7	0000111	BEL	Bell	71	1000111	G	Latin capital letter G
8	0001000	BS	Backspace	72	1001000	H	Latin capital letter H
9	0001001	HT	Horizontal Tab	73	1001001	I	Latin capital letter I
10	0001010	LF	Line Feed	74	1001010	J	Latin capital letter J
11	0001011	VT	Vertical Tab	75	1001011	K	Latin capital letter K
12	0001100	FF	Form Feed	76	1001100	L	Latin capital letter L
13	0001101	CR	Carriage Return	77	1001101	M	Latin capital letter M
14	0001110	SO	Shift Out	78	1001110	N	Latin capital letter N
15	0001111	SI	Shift In	79	1001111	O	Latin capital letter O
16	0010000	DLE	Data Link Escape	80	1010000	P	Latin capital letter P
17	0010001	DC1	Device Control 1 to 4	81	1010001	Q	Latin capital letter Q
18	0010010	DC2	DC1 DC3 used to code	82	1010010	R	Latin capital letter R
19	0010011	DC3	XON	83	1010011	S	Latin capital letter S
20	0010100	DC4	and XOFF	84	1010100	T	Latin capital letter T
21	0010101	NAK	Negative Acknowledge	85	1010101	U	Latin capital letter U
22	0010110	SYN	Synchronous Idle	86	1010110	V	Latin capital letter V
23	0010111	ETB	End of Transmission Block	87	1010111	W	Latin capital letter W
24	0011000	CAN	Cancel	88	1011000	X	Latin capital letter X
25	0011001	EM	End of Medium	89	1011001	Y	Latin capital letter Y
26	0011010	SUB	Substitute	90	1011010	Z	Latin capital letter Z
27	0011011	ESC	Escape	91	1011011	[Opening bracket
28	0011100	FS	File Separator	92	1011100	\	Backslash
29	0011101	GS	Group Separator	93	1011101]	Closing bracket
30	0011110	RS	Record Separator	94	1011110	^	Caret
31	0011111	US	Unit Separator	95	1011111	_	Underscore
32	0100000	SP	Space	96	1100000	`	Grave accent
33	0100001	!	Exclamation point	97	1100001	a	Latin small letter a
34	0100010	"	Quotation mark	98	1100010	b	Latin small letter b
35	0100011	#	Hash	99	1100011	c	Latin small letter c
36	0100100	\$	Dollar sign	100	1100100	d	Latin small letter d
37	0100101	%	Percent sign	101	1100101	e	Latin small letter e
38	0100110	&	Ampersand	102	1100110	f	Latin small letter f
39	0100111	'	Apostrophe	103	1100111	g	Latin small letter g
40	0101000	(Opening parenthesis	104	1101000	h	Latin small letter h
41	0101001)	Closing parenthesis	105	1101001	i	Latin small letter i
42	0101010	*	Asterisk	106	1101010	j	Latin small letter j
43	0101011	+	Plus sign	107	1101011	k	Latin small letter k
44	0101100	,	Comma	108	1101100	l	Latin small letter l
45	0101101	-	Minus sign	109	1101101	m	Latin small letter m
46	0101110	.	Period	110	1101110	n	Latin small letter n
47	0101111	/	Slash	111	1101111	o	Latin small letter o
48	0110000	Ø	Digit zero	112	1110000	p	Latin small letter p
49	0110001	1	Digit one	113	1110001	q	Latin small letter q
50	0110010	2	Digit two	114	1110010	r	Latin small letter r
51	0110011	3	Digit three	115	1110011	s	Latin small letter s
52	0110100	4	Digit four	116	1110100	t	Latin small letter t
53	0110101	5	Digit five	117	1110101	u	Latin small letter u
54	0110110	6	Digit six	118	1110110	v	Latin small letter v
55	0110111	7	Digit seven	119	1110111	w	Latin small letter w
56	0111000	8	Digit eight	120	1111000	x	Latin small letter x
57	0111001	9	Digit nine	121	1111001	y	Latin small letter y
58	0111010	:	Colon	122	1111010	z	Latin small letter z
59	0111011	;	Semicolon	123	1111011	{	Left curly bracket
60	0111100	<	Less than sign	124	1111100		Vertical bar (pipe)
61	0111101	=	Equals sign	125	1111101	}	Right curly bracket
62	0111110	>	Greater than sign	126	1111110	~	Tilde
63	0111111	?	Question mark	127	1111111	DEL	Delete

TABLE 4.2 – Table de codage ASCII étendu ISO-Latin1/IBM-PC (8 bits).

TABLE 4.3 — Table de codage Unicode Latin1-supplement : U+0000–U+007F.

TABLE DES CARACTÈRES UNICODE LATIN1-SUPPLEMENT : O–127/U+0000–U+007F

DÉC.	CODE	GLYPHE	DESCRIPTION	DÉC.	CODE	GLYPHE	DESCRIPTION
0	U+0000	NUL	Null character	64	U+0040	ⓐ	At sign
1	U+0001	SOH	Start of Heading	65	U+0041	A	Latin Capital letter A
2	U+0002	STX	Start of Text	66	U+0042	B	Latin Capital letter B
3	U+0003	ETX	End-of-text character	67	U+0043	C	Latin Capital letter C
4	U+0004	EOT	End-of-transmission character	68	U+0044	D	Latin Capital letter D
5	U+0005	ENQ	Enquiry character	69	U+0045	E	Latin Capital letter E
6	U+0006	ACK	Acknowledge character	70	U+0046	F	Latin Capital letter F
7	U+0007	BEL	Bell character	71	U+0047	G	Latin Capital letter G
8	U+0008	BS	Backspace	72	U+0048	H	Latin Capital letter H
9	U+0009	HT	Horizontal tab	73	U+0049	I	Latin Capital letter I
10	U+000A	LF	Line feed	74	U+004A	J	Latin Capital letter J
11	U+000B	VT	Vertical tab	75	U+004B	K	Latin Capital letter K
12	U+000C	FF	Form feed	76	U+004C	L	Latin Capital letter L
13	U+000D	CR	Carriage return	77	U+004D	M	Latin Capital letter M
14	U+000E	SO	Shift Out	78	U+004E	N	Latin Capital letter N
15	U+000F	SI	Shift In	79	U+004F	O	Latin Capital letter O
16	U+0010	DLE	Data Link Escape	80	U+0050	P	Latin Capital letter P
17	U+0011	DC1	Device Control 1	81	U+0051	Q	Latin Capital letter Q
18	U+0012	DC2	Device Control 2	82	U+0052	R	Latin Capital letter R
19	U+0013	DC3	Device Control 3	83	U+0053	S	Latin Capital letter S
20	U+0014	DC4	Device Control 4	84	U+0054	T	Latin Capital letter T
21	U+0015	NAK	Negative-acknowledge character	85	U+0055	U	Latin Capital letter U
22	U+0016	SYN	Synchronous Idle	86	U+0056	V	Latin Capital letter V
23	U+0017	ETB	End of Transmission Block	87	U+0057	W	Latin Capital letter W
24	U+0018	CAN	Cancel character	88	U+0058	X	Latin Capital letter X
25	U+0019	EM	End of Medium	89	U+0059	Y	Latin Capital letter Y
26	U+001A	SUB	Substitute character	90	U+005A	Z	Latin Capital letter Z
27	U+001B	ESC	Escape character	91	U+005B	[Left Square Bracket
28	U+001C	FS	File Separator	92	U+005C	\	Backslash
29	U+001D	GS	Group Separator	93	U+005D]	Right Square Bracket
30	U+001E	RS	Record Separator	94	U+005E	^	Circumflex accent
31	U+001F	US	Unit Separator	95	U+005F	_	Low line
32	U+0020	Space		96	U+0060	`	Grave accent
33	U+0021	!	Exclamation mark	97	U+0061	a	Latin Small Letter A
34	U+0022	"	Quotation mark	98	U+0062	b	Latin Small Letter B
35	U+0023	#	Hashtag Octothorpe Sharp	99	U+0063	c	Latin Small Letter C
36	U+0024	\$	Dollar sign	100	U+0064	d	Latin Small Letter D
37	U+0025	%	Percent sign	101	U+0065	e	Latin Small Letter E
38	U+0026	&	Ampersand	102	U+0066	f	Latin Small Letter F
39	U+0027	'	Apostrophe	103	U+0067	g	Latin Small Letter G
40	U+0028	(Left parenthesis	104	U+0068	h	Latin Small Letter H
41	U+0029)	Right parenthesis	105	U+0069	i	Latin Small Letter I
42	U+002A	*	Asterisk	106	U+006A	j	Latin Small Letter J
43	U+002B	+	Plus sign	107	U+006B	k	Latin Small Letter K
44	U+002C	,	Comma	108	U+006C	l	Latin Small Letter L
45	U+002D	-	Hyphen-minus	109	U+006D	m	Latin Small Letter M
46	U+002E	.	Full stop	110	U+006E	n	Latin Small Letter N
47	U+002F	/	Slash (Solidus)	111	U+006F	o	Latin Small Letter O
48	U+0030	0	Digit Zero	112	U+0070	p	Latin Small Letter P
49	U+0031	1	Digit One	113	U+0071	q	Latin Small Letter Q
50	U+0032	2	Digit Two	114	U+0072	r	Latin Small Letter R
51	U+0033	3	Digit Three	115	U+0073	s	Latin Small Letter S
52	U+0034	4	Digit Four	116	U+0074	t	Latin Small Letter T
53	U+0035	5	Digit Five	117	U+0075	u	Latin Small Letter U
54	U+0036	6	Digit Six	118	U+0076	v	Latin Small Letter V
55	U+0037	7	Digit Seven	119	U+0077	w	Latin Small Letter W
56	U+0038	8	Digit Eight	120	U+0078	x	Latin Small Letter X
57	U+0039	9	Digit Nine	121	U+0079	y	Latin Small Letter Y
58	U+003A	:	Colon	122	U+007A	z	Latin Small Letter Z
59	U+003B	;	Semicolon	123	U+007B	{	Left Curly Bracket
60	U+003C	<	Less-than sign	124	U+007C		Vertical bar (pipe)
61	U+003D	=	Equal sign	125	U+007D	}	Right Curly Bracket
62	U+003E	>	Greater-than sign	126	U+007E	~	Tilde
63	U+003F	?	Question mark	127	U+007F	DEL	Delete

TABLE 4.4 — Table de codage Unicode Latin1-supplement : U+0080–U+00FF.

TABLE DES CARACTÈRES UNICODE LATIN1-SUPPLEMENT : 128–255/U+0080–U+00FF

DÉC.	CODE	GLYPHE	DESCRIPTION	DÉC.	CODE	GLYPHE	DESCRIPTION
128	U+0080	PAD	Padding Character	192	U+00C0	À	Latin Capital Letter A with grave
129	U+0081	HOP	High Octet Preset	193	U+00C1	Á	Latin Capital letter A with acute
130	U+0082	BPH	Break Permitted Here	194	U+00C2	Â	Latin Capital letter A with circumflex
131	U+0083	NBH	No Break Here	195	U+00C3	Ã	Latin Capital letter A with tilde
132	U+0084	IND	Index	196	U+00C4	Ä	Latin Capital letter A with diaeresis
133	U+0085	NEL	Next Line	197	U+00C5	Å	Latin Capital letter A with ring above
134	U+0086	SSA	Start of Selected Area	198	U+00C6	Æ	Latin Capital letter Æ
135	U+0087	ESA	End of Selected Area	199	U+00C7	Ҫ	Latin Capital letter C with cedilla
136	U+0088	HTS	Character Tabulation Set	200	U+00C8	È	Latin Capital letter E with grave
137	U+0089	HTJ	Character Tabulation with Justification	201	U+00C9	É	Latin Capital letter E with acute
138	U+008A	VTS	Line Tabulation Set	202	U+00CA	Ê	Latin Capital letter E with circumflex
139	U+008B	PLD	Partial Line Forward	203	U+00CB	Ë	Latin Capital letter E with diaeresis
140	U+008C	PLU	Partial Line Backward	204	U+00CC	Ì	Latin Capital letter I with grave
141	U+008D	RI	Reverse Line Feed	205	U+00CD	Í	Latin Capital letter I with acute
142	U+008E	SS2	Single-Shift Two	206	U+00CE	Î	Latin Capital letter I with circumflex
143	U+008F	SS3	Single-Shift Three	207	U+00CF	Ï	Latin Capital letter I with diaeresis
144	U+0090	DCS	Device Control String	208	U+00D0	ܹ	Latin Capital letter Eth
145	U+0091	PU1	Private Use 1	209	U+00D1	ܻ	Latin Capital letter N with tilde
146	U+0092	PU2	Private Use 2	210	U+00D2	ܻ	Latin Capital letter O with grave
147	U+0093	STS	Set Transmit State	211	U+00D3	ܻ	Latin Capital letter O with acute
148	U+0094	CCH	Cancel character	212	U+00D4	ܻ	Latin Capital letter O with circumflex
149	U+0095	MW	Message Waiting	213	U+00D5	ܻ	Latin Capital letter O with tilde
150	U+0096	SPA	Start of Protected Area	214	U+00D6	ܻ	Latin Capital letter O with diaeresis
151	U+0097	EPA	End of Protected Area	215	U+00D7	ܻ	Multiplication sign
152	U+0098	SOS	Start of String	216	U+00D8	ܻ	Latin Capital letter O with stroke
153	U+0099	SGCI	Single Graphic Character Introducer	217	U+00D9	ܻ	Latin Capital letter U with grave
154	U+009A	SCI	Single Character Intro Introducer	218	U+00DA	ܻ	Latin Capital letter U with acute
155	U+009B	CSI	Control Sequence Introducer	219	U+00DB	ܻ	Latin Capital Letter U with circumflex
156	U+009C	ST	String Terminator	220	U+00DC	ܻ	Latin Capital Letter U with diaeresis
157	U+009D	OSC	Operating System Command	221	U+00DD	ܻ	Latin Capital Letter Y with acute
158	U+009E	PM	Private Message	222	U+00DE	ܻ	Latin Capital Letter Thorn
159	U+009F	APC	Application Program Command	223	U+00DF	ܻ	Latin Small Letter sharp S
160	U+00A0		Non-breaking space	224	U+00E0	ܻ	Latin Small Letter A with grave
161	U+00A1	ܻ	Inverted Exclamation Mark	225	U+00E1	ܻ	Latin Small Letter A with acute
162	U+00A2	ܻ	Cent sign	226	U+00E2	ܻ	Latin Small Letter A with circumflex
163	U+00A3	ܻ	Pound sign	227	U+00E3	ܻ	Latin Small Letter A with tilde
164	U+00A4	ܻ	Currency sign	228	U+00E4	ܻ	Latin Small Letter A with diaeresis
165	U+00A5	ܻ	Yen sign	229	U+00E5	ܻ	Latin Small Letter A with ring above
166	U+00A6	ܻ	Broken bar	230	U+00E6	ܻ	Latin Small Letter Æ
167	U+00A7	ܻ	Section sign	231	U+00E7	ܻ	Latin Small Letter C with cedilla
168	U+00A8	ܻ	Diaeresis (Umlaut)	232	U+00E8	ܻ	Latin Small Letter E with grave
169	U+00A9	ܻ	Copyright sign	233	U+00E9	ܻ	Latin Small Letter E with acute
170	U+00AA	ܻ	Feminine Ordinal Indicator	234	U+00EA	ܻ	Latin Small Letter E with circumflex
171	U+00AB	ܻ	Left-pointing double angle quotation mark	235	U+00EB	ܻ	Latin Small Letter E with diaeresis
172	U+00AC	ܻ	Not sign	236	U+00EC	ܻ	Latin Small Letter I with grave
173	U+00AD	ܻ	Soft hyphen	237	U+00ED	ܻ	Latin Small Letter I with acute
174	U+00AE	ܻ	Registered sign	238	U+00EE	ܻ	Latin Small Letter I with circumflex
175	U+00AF	ܻ	macron	239	U+00EF	ܻ	Latin Small Letter I with diaeresis
176	U+00B0	ܻ	Degree symbol	240	U+00F0	ܻ	Latin Small Letter Eth
177	U+00B1	ܻ	Plus-minus sign	241	U+00F1	ܻ	Latin Small Letter N with tilde
178	U+00B2	ܻ	Superscript two	242	U+00F2	ܻ	Latin Small Letter O with grave
179	U+00B3	ܻ	Superscript three	243	U+00F3	ܻ	Latin Small Letter O with acute
180	U+00B4	ܻ	Acute accent	244	U+00F4	ܻ	Latin Small Letter O with circumflex
181	U+00B5	ܻ	Micro sign	245	U+00F5	ܻ	Latin Small Letter O with tilde
182	U+00B6	ܻ	Pilcrow sign	246	U+00F6	ܻ	Latin Small Letter O with diaeresis
183	U+00B7	ܻ	Middle dot	247	U+00F7	ܻ	Division sign
184	U+00B8	ܻ	Cedilla	248	U+00F8	ܻ	Latin Small Letter O with stroke
185	U+00B9	ܻ	Superscript one	249	U+00F9	ܻ	Latin Small Letter U with grave
186	U+00BA	ܻ	Masculine ordinal indicator	250	U+00FA	ܻ	Latin Small Letter U with acute
187	U+00BB	ܻ	Right-pointing double angle quotation mark	251	U+00FB	ܻ	Latin Small Letter U with circumflex
188	U+00BC	ܻ	Vulgar fraction one quarter	252	U+00FC	ܻ	Latin Small Letter U with diaeresis
189	U+00BD	ܻ	Vulgar fraction one half	253	U+00FD	ܻ	Latin Small Letter Y with acute
190	U+00BE	ܻ	Vulgar fraction three quarters	254	U+00FE	ܻ	Latin Small Letter Thorn
191	U+00BF	ܻ	Inverted Question Mark	255	U+00FF	ܻ	Latin Small Letter Y with diaeresis



Devant les difficultés d'interopérabilité des différents codages nationaux, à partir de 1991, un codage uniifié a commencé à être développé pour résoudre ces lourdeurs de gestion. Le principe reste le même — un numéro par caractère à afficher dans le texte — mais, au-delà d'exprimer une unification pour toutes les langues y compris les sinogrammes et kanjis asiatiques, les performances sont meilleures au sens où quelque soit le logiciel et le système d'exploitation, le codage est reconnu. Le coût est un peu plus de complexité et une consommation supérieure en ressources (voir pour comparatif des codes les tables 4.3 et 4.4).

La mise en place de l'Unicode a un peu traîné compte tenu du fait qu'il fallait gérer la migration de nombreuses applications vers ce nouveau standard, entre autres les sites Web; mais adieu aux signes cabalistiques dans les courriels en fonction de la configuration de l'expéditeur... On peut noter que tous les systèmes d'exploitation GNU/LINUX sont depuis une bonne décennie nativement au format [Unicode UTF-8](#).

Le dessin du caractère — le glyphe — à l'écran n'est important qu'au moment de l'affichage mais, pour stocker l'information, seuls sont nécessaires les différentes représentations codées d'un texte.

RESSOURCES COMPLÉMENTAIRES

Formation complémentaire

- ▶ Partie 1 du #2 [Module thématique : manipuler l'information, CLASS'CODE](#). Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme ➔

Articles

- ▶ [Codage de l'information, WIKIPÉDIA](#).
- ▶ [Nom de code : binaire, par Pascal GUITTON, INTERSTICES, 12 décembre 2008](#)
- ▶ [Les mots pour le dire : C comme codage, par Thierry VIÉVILLE, BINAIRE, 30 avril 2014.](#)

Vidéos

- ▶ [Codage de l'Information, Les capsules, 2014 \(lien mort\).](#)

Cours

- ▶ [Codages binaires et autres codages, par Fabien Torre, Université de Lille, 2013.](#)

Autres ressources

- ▶ [Codage de l'information, lycée Jules Ferry, Versailles, sect. SI.](#)
- ▶ [Systèmes de numération et codages, lycée Jospeh Desfontaines, Melle, section SI.](#)
- ▶ [Binaire et codage des informations, lycée Camille Corot, Mo- restel, option MPI.](#)



VIDÉO 4.2 — *Image et son.*

⁷. Plus précisément, la musique concrète et électroacoustique (cf. [Pierre SCHAEFFER](#)) puis électronique; non pas seulement celle des synthétiseurs, instruments électro- niques et autres DJ sur séquenceurs.

2 Infographie et audionumérique

Si le XX^e siècle a connu l'avènement puis le développement d'arts nouveaux plus ou moins bien classifiés comme la photographie, le cinéma, la bande dessinée, le jazz et le rock'n'roll (ben oui) ou encore la musique⁷ électronique, voire même selon certains le jeu vidéo, le XXI^e siècle ne manquera pas de compléter la liste avec gourmandise.

Au-delà de toute taxonomie, il y a fort à parier que le numérique au sens large en sera partie prenante comme un des vecteurs principaux, puisque c'est déjà le cas! À quand l'art reconnu de la programmation?

Toujours est-il que l'on constate depuis quarante ans énormément de bouleversements dans les industries audiovisuelles avec l'arrivée du numérique, qu'il s'agisse de l'infographie autant que l'audionumérique.

2.1 Photographie et dessin

Il n'y a qu'un pas du traitement de texte à la PAO — Publication assistée par ordinateur. Il a rapidement été franchi, avec comme corollaire le traitement des images : photographie et dessin.

Ces deux orientations donnent lieu à chacune leur manière de procéder avec des données visuelles : la segmentation pour ce qui est des images matricielles — dites aussi *bitmap*, littéralement « carte de bits » — et la modélisation mathématique pour les images vectorielles.

De nos jours, les images sont omniprésentes, qu'il s'agisse de télévision, de jeux vidéos ou de « réalité virtuelle » (voir chapitre 1). Comment les affiche-t-on concrètement ?

2.1.1 Images matricielles

Comme évoqué précédemment (voir § 1.2.2), cela n'est pas tant dans le codage d'une information visuelle que se pose le problème, mais dans son rendu : à l'écran en retouche ou à l'impression papier.

Pour présenter une image matricielle, on s'appuie sur la résolution de l'œil humain et une décomposition en *pixels* — pour *picture element* en anglais. Ces pixels sont l'assemblage de points de trois couleurs primaires que sont le rouge, le vert et le bleu — RGB pour *Red, Green, Blue* — sur la base desquelles on peut projeter n'importe quelle couleur. La taille des points est déterminée de manière à être inférieure à la discrimination visuelle humaine.

Chaque pixel est représenté par un nombre défini de bits. Tout dépend alors de la quantité de couleur nécessaire pour l'affichage. Trois cas de figure sont à considérer :

1. le noir et blanc, avec un seul bit par pixel;
2. les niveaux de gris, représentés par huit bits, soit 256 valeurs de 0 pour le noir à 255 pour le blanc avec toutes les valeurs intermédiaires de niveau de gris;
3. la couleur, avec huit bits par composante RGB, soit 24 bits au total pour chaque pixel, autrement dit 16 millions de couleurs.

La question qui se pose encore est de savoir si par exemple une suite de 24 bits correspond à un pixel en couleur ou 24 pixels noir et blanc. Un certain nombre d'information complémentaires sur l'image sont disposées en entête du fichier pour décrire complètement son contenu : dimensions de l'image et codage des pixels.

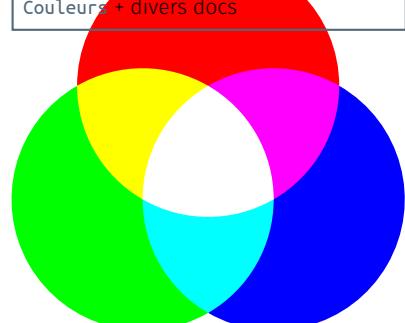
Cette dépendance aux dimensions de l'image peut être source de problème dits de « pixellisation », si d'aventure le nombre de points la décrivant est trop faible par rapport à la taille de l'affichage voulu (voir illustration en figure 4.5). Pour cette raison, en infographie et PAO, il est convenu un certain nombre de normes pratiques s'appuyant sur la résolution⁸ de l'image en points par pouce (ppp) — ou dpi pour « dot per inch » en anglais, à savoir :

- 72 ppp à 150 ppp suffisent pour une publication Web donc pour avoir une lecture à l'écran confortable (le critère est aussi de ne pas surcharger les réseaux et maintenir une navigation fluide);
- 150 ppp à 300 ppp pour une publication papier allant jusqu'au format A4, mais dont l'image ne couvre pas la totalité d'une page;
- de l'ordre 600 ppp pour une page A4 ou une photographie;
- au-delà de 1200 ppp en fonction des besoins pour de la photographie d'art et des affiches publicitaires 4 sur 3.

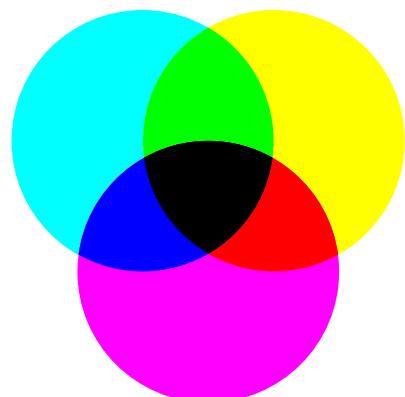
Selon la nature d'un document, il faut être vigilant aux images à y insérer, car si des interpolations et des retouches sont possibles, la marge manœuvre est relativement étroite. C'est parfois⁹ un casse-tête qui justifie le recours aux banques d'images de haute résolution.

NOTE DE LA RÉDACTION

Section à compléter pour rééquilibrer avec la partie son ? Par exemple : <https://images.math.cnrs.fr/Le-traitement-numérique-des-images.html>, <http://images.math.cnrs.fr/Compression-d-image.html>, <https://fr.wikipedia.org/wiki/Portail:Couleurs> + divers docs



(a) Synthèse additive — RGB.



(b) Synthèse soustractive — CMY.

FIGURE 4.4 — Couleurs primaires : rouge, vert, bleu. Couleurs secondaires : cyan, magenta, jaune (quadrichromie).

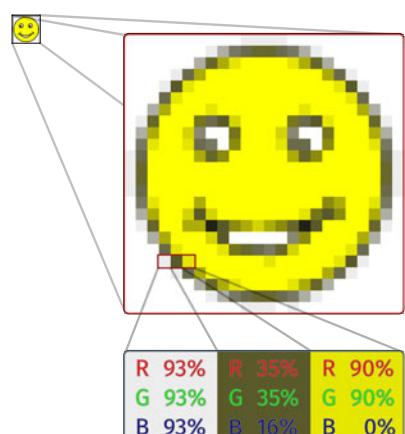


FIGURE 4.5 — Image matricielle : définition de couleur et pixellisation.

8. Cette résolution est également celle des imprimantes et des scanners.

9. Les éditeurs ont parfois du mal à expliquer cette contrainte à leurs auteurs.

2.1.2 Images vectorielles

Une autre approche pour décrire des images s'appuie sur une modélisation mathématique des formes. Cela s'applique particulièrement bien aux dessins techniques (architecture, DAO/CAO — Dessin/Conception assistés par ordinateur —, réalité virtuelle, etc.) et à l'élaboration de logotype, de signalétique et de police de caractères en infographie.

Au lieu de décrire une image pixel par pixel, on considère un assemblage de primitives géométriques telles que des segments de droite, arcs de cercle, polygones, etc. Le dessin vectoriel doit beaucoup aux courbes de Bézier, révélées par Pierre Bézier dans les années 1950.

À l'ensemble de ces courbes géométriques dites paramétriques (cf. W), on combine différents attributs : position, couleur, visibilité... Mais également diverses transformations : homothéties, interpolation, dégradés, extrusion, etc. Le principal format de fichier est de nos jours le SVG pour *Scalable Vector Graphics* et, pour les polices vectorielles compatibles Unicode l'OTF, pour *OpenType Font*.

Par contraste avec leurs analogues matricielles, les images vectorielles s'affranchissent ainsi de tout problème de pixellisation tout en étant plus précises et plus concises (fondations mathématiques), donc moins gourmandes en espace de stockage. En revanche, elles sont inopérantes pour la photographie.

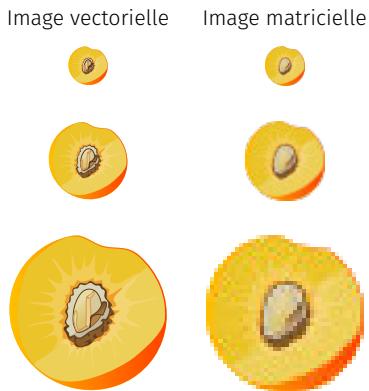


FIGURE 4.6 — Image vectorielle : mise à l'échelle sans perte.

RESSOURCES COMPLÉMENTAIRES

Articles

- ▶ Histoire du traitement des images, par Isabelle BELLIN, Interstices, 25 février 2004.
- ▶ Nom de code : binaire, par Pascal GUITTON, INTERSTICES, 12 décembre 2008.
- ▶ Traitement numérique des images, par Gabriel PEYRÉ, Images des mathématiques, 2011.
- ▶ Synthèse additive, WIKIPÉDIA.

Vidéos

- ▶ Qu'est-ce qu'une couleur ?, par David LOUAPRE, Science étonnante #25 (14' 44"), 2016.

2.2 Son et musique

NOTE DE LA RÉDACTION
Rejeter en annexe une bonne partie de cette section, notamment sur l'audition, compléter l'annexe par les courbes de masques pour la compression et du traitement de signal audio (FFT, etc.) pour programmes PYTHON/MATPLOTLIB. Faire la même chose pour la vision et le traitement d'images cf. § précédent.

En toute généralité, l'étude des phénomènes sonores relève tout autant de la physique ondulatoire — vibroacoustique — que de la théorie des signaux et de la communication. Elle en est même aux prémisses avec par exemple le téléphone. Ceci étant posé, qu'en est-il aujourd'hui de l'apport du numérique ?

2.2.1 Caractérisation et analyse des sons

Comme discipline scientifique, l'acoustique répond au triptyque traditionnel : production (émetteur), propagation (transmetteur) et réception (capteur). Chacun de ces volets fait appel à la fois à la physique et au traitement de signal — signal analogique *versus* numérique.

A. SIGNAL SONORE

Du point de vue de la physique, un son est une variation locale de pression atmosphérique produite par la mise en vibration d'une structure — corde, membrane, etc. — ou d'un fluide — tuyau d'orgue, bec de saxophone, etc. —, rayonnée dans l'air ambiant puis réceptionnée par l'appareil auditif. Formellement, on peut écrire :

$$P(r, t) = P_0(r, t) + p(r, t) \quad (4.1)$$

Les différentes notations de l'équation (4.1) représentent :

- $r \rightarrow$ localisation dans l'espace ($r = x + y + z = x.e_x + y.e_y + z.e_z$);
- $t \rightarrow$ dépendance dans le temps du phénomène constaté;
- $P(r, t) \rightarrow$ champ de pression totale existante dans l'air;
- $P_0(r, t) \rightarrow$ champ de pression atmosphérique ambiante;
- $p(r, t) \rightarrow$ pression acoustique, soit la variation — ou la perturbation — de pression atmosphérique due à la présence d'une onde.

ÉVOLUTION TEMPORELLE — En faisant fi des aspects¹⁰ physiques et supposant la pression atmosphérique constante, reste le « signal sonore » en tant que tel, localisé par exemple à l'oreille d'un auditeur. Par définition, c'est la variation de pression acoustique au cours du temps que l'on appelle « forme d'onde » (voir figure 4.7a).

L'unité d'évaluation de la pression est le *Pascal*, noté Pa, homogène au Newton par mètre carré (N/m^2). Les ordres de grandeur des pressions atmosphériques et acoustiques sont respectivement :

$$\begin{aligned} P_0 &\approx 10^5 \text{ Pa (ou } N/m^2\text{)} \approx 1 \text{ bar} \\ 10^{-5} \text{ Pa} &\quad p \quad 100 \text{ Pa, soit } 0,1 \text{ nbar} \quad p \quad 1 \text{ mbar} \end{aligned} \quad (4.2)$$

On peut immédiatement souligner la remarquable dynamique de réponse de l'oreille humaine. En effet, l'échelle de pression acoustique est de l'ordre de dix millions de valeurs possibles (10^7 , de 10^{-5} à 10^2 Pa) entre¹¹ la pression minimale juste audible et le seuil de douleur où le tympan s'engage dans une sollicitation déraisonnable.

Dans une perspective numérique, le chemin est à peu près le même : fichier temps-amplitude d'un signal de valeurs oscillantes variant autour de zéro au cours du temps — synthétiseur, ordinateur —, canal de transfert — table de mixage, séquenceur, réseau, etc. — puis sortie directe ou indirecte pour enregistrement ou restitution — sauvegarde ou transducteur quelconque, enceinte acoustique et/ou microphone.

Les paramètres temporels d'un signal sonore — *a fortiori* musical — sont de très grande importance perceptive. Pour s'en convaincre, il suffit de comparer les enveloppes temporelles caractéristiques de deux types d'instruments différents : un piano et une trompette (voir figure 4.8).

En acoustique musicale, plusieurs critères existent pour catégoriser les sons en fonction de la famille instrumentale. Une des indications les plus directes est de distinguer les instruments de musique selon leur *régime d'oscillation* :

1. *oscillations libres* → instruments à percussion, à cordes pincées et frappées (voir figure 4.8a);
2. *oscillations auto-entretenues* — ou *forcées* → instruments à vent et à cordes frottées — boucle de rétro-action — (cf. figure 4.8b).

Issue des méthodes de synthèse sonore analogique (1960-1970), notamment la synthèse par table d'onde — *wave-table synthesis* —, la description de l'évolution d'un son suit une courbe dite ADSR pour « *Attack-Decay-Sustain-Release* » (voir figures 4.9 et 4.10). Cela correspond aux différentes phases de l'évolution temporelle d'un son que l'on peut relier grossièrement à des paramètres psychoacoustiques :

1. transitoires d'attaque (A+D), effets temporels fondamentaux pour la reconnaissance perceptive;
2. régime quasi-stationnaire (S) lié à l'enveloppe spectrale (cf. infra) et donc associé aux paramètres de hauteur et de timbre du son;
3. transitoires d'extinction (R), de nouveau effets temporels plutôt relatifs aux composantes de perte ou d'amortissement du son.

SON ÉLÉMENTAIRE — Le signal sonore le plus simple est décrit par une fonction sinusoïdale. Dans le jargon des acousticiens, il est qualifié de son « *pur* », comme composante de base d'un signal sonore. Son

¹⁰. Il y aurait trop de choses à expliciter par rapport au contexte de ce document.

¹¹. Cela justifie l'utilisation d'une échelle logarithmique de quantification au moyen du décibel (dB).

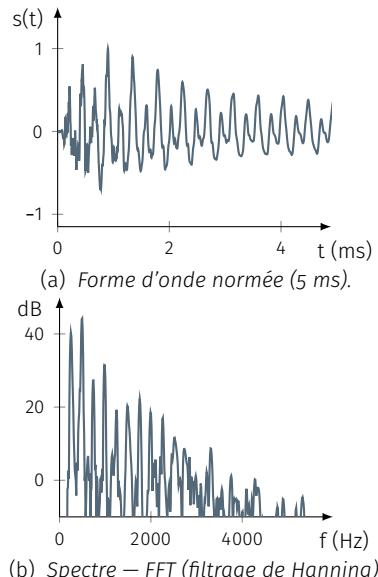


FIGURE 4.7 — Forme d'onde et spectre d'une note C_4 de violon jouée pizzicato (5 ms).

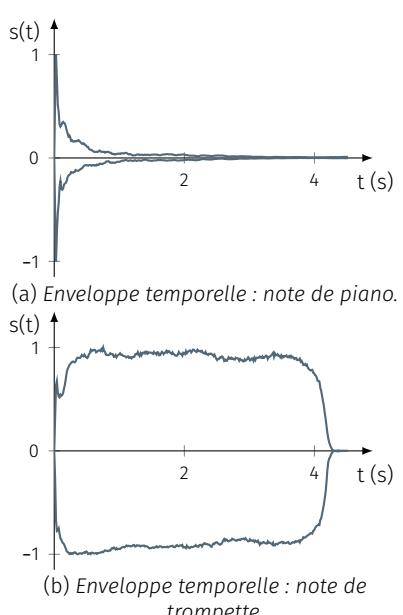


FIGURE 4.8 — Description d'un son musical (note A_4 , $t = 4,5\text{s}$) : transitoires d'attaque, régime quasi-stationnaire, relaxe et extinction.

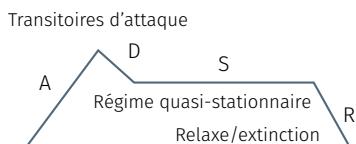


FIGURE 4.9 — Courbe ADSR : transitoires d'attaque, régime quasi-stationnaire et transitoires de relaxe et d'extinction.

expression mathématique permet d'introduire les concepts fondamentaux qui caractérisent un phénomène sonore (voir éq. 4.3).

$$p(t) = A(t) \sin(\omega t + \varphi(t)) = A(t) \sin(2\pi t/T + \varphi(t)) = A(t) \sin(2\pi ft + \varphi(t))$$

$t \rightarrow$ temps (s)

$A(t) \rightarrow$ amplitude (Pa, N.m⁻²)

$T \rightarrow$ période (s)

avec

$\varphi(t) \rightarrow$ phase (rad)

$f \rightarrow$ fréquence (Hz), $f = 1/T$

$\omega \rightarrow$ pulsation/fréquence angulaire (rad/s), $\omega = 2\pi f$

Les propriétés mathématiques des fonctions sinusoïdales sont bien adaptées à la description des signaux sonores et plus largement des phénomènes vibratoires et ondulatoires de toute nature. En effet, l'évolution de l'amplitude d'un sinus parcourt un cycle qui se répète à l'identique dans le temps (voir figure 4.11a), directement interprétable comme la variation d'un phénomène autour d'une position d'équilibre.

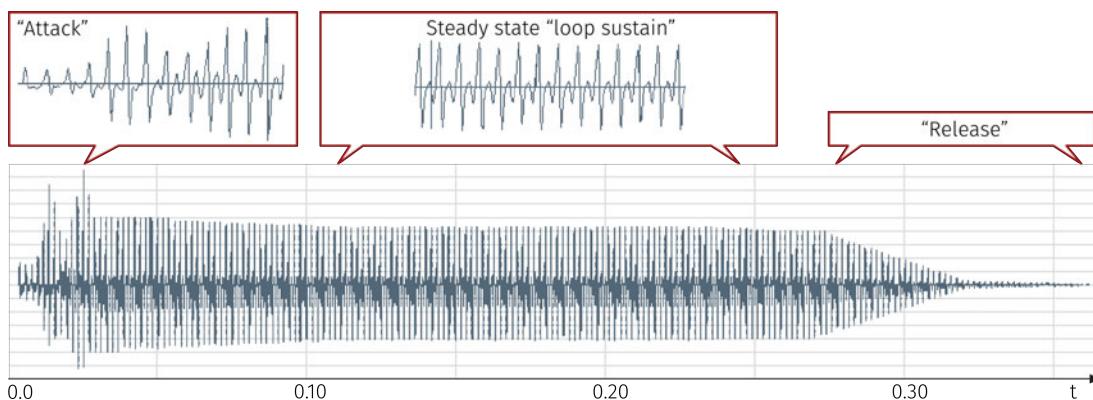
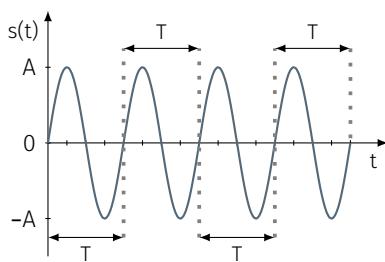
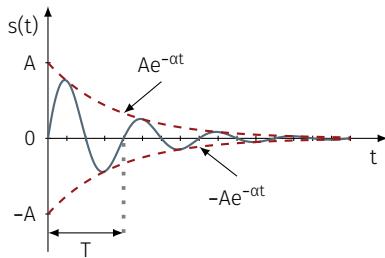


FIGURE 4.10 — Synthèse par table d'onde d'une note de trompette (d'après ROSSING, AKHATOV et al., 2007).



(a) Fonction sinusoïdale non amortie,
 $s(t) = A \sin(\omega t) = A \sin(2\pi f t)$.



(b) Fonction sinusoïdale amortie,
 $s(t) = A(t) \sin(\omega t)$, avec $A(t) = Ae^{-at}$.

FIGURE 4.11 — Signal sonore élémentaire : sinusoïde de fréquence f , de période $T = 1/f$, d'amplitude $A(t)$ et de phase à l'origine $\varphi = 0$.

Les fonctions sinusoïdales ont une amplitude de variation dans l'intervalle de valeurs comprises entre -1 et +1. Afin de décrire des signaux sonores d'amplitude quelconque, leur sont adjoints un facteur multiplicatif $A(t)$, variable au cours du temps — attribut qui permet de modéliser les phénomènes de pertes d'énergie des signaux réels.

Une sinusoïde d'amplitude constante au cours du temps permet effectivement une première description théorique des phénomènes oscillants, mais n'est le reflet que d'une modélisation partielle des phénomènes en jeu : un signal sonore ne perdure pas à l'infini après extinction de la source qui lui a donné naissance ! Un signal réel s'amortit progressivement ou, autrement dit, possède une amplitude décroissante en fonction du temps.

Cet amortissement est le reflet direct des pertes d'énergie dans le système considéré, lesquelles proviennent d'origines physiques très diverses suivant la nature du signal : frottements mécaniques, propriétés viscoélastiques et viscothermiques des matériaux solides ou des milieux fluides... Pour tenir compte des conséquences de ces phénomènes dans l'écriture du signal, l'amplitude est exprimée à l'aide de la fonction exponentielle sous la forme :

$$A(t) = A \exp(-\alpha t) = Ae^{-\alpha t}$$

$A \rightarrow$ amplitude maximale (Pa, N.m⁻²)

avec $t \rightarrow$ temps (s)

$\alpha \rightarrow$ facteur d'amortissement (1/s ou s⁻¹); $\alpha > 0$

Le facteur d'amortissement α est strictement positif et s'exprime en inverse de seconde. Sa valeur plus ou moins importante dépend de

l'origine des pertes qu'il traduit. Comme visualisable en figure 4.11b, il permet de contrôler l'évolution de la décroissance de la sinusoïde.

B. ANALYSE SPECTRALE

Un des grands résultats de l'acoustique¹² est de pouvoir décomposer les signaux sonores comme combinaison linéaire d'un ensemble de fonctions de base. Il s'agit en fait d'appliquer le « principe de FOURIER¹³ » qui, suivant la nature du signal sonore fait appel aux outils mathématiques adéquats : séries et transformées de FOURIER.

SIGNALS PÉRIODIQUES : SPECTRE DISCRET — Un signal sonore périodique ou quasi-périodique¹⁴ peut se décomposer en série de Fourier. Ce cas de figure est celui de nombreux signaux musicaux et permet une première approche des phénomènes.

Selon la théorie de FOURIER, un signal périodique se décompose sur la base des fonctions sinusoïdales comme la somme de signaux élémentaires de fréquences multiples les unes des autres.

$$p(t) = \sum_{n \geq 1} A_n(t) \sin(\omega_n t + \phi_n(t)) \quad (4.5)$$

avec par composante $A_n(t) \rightarrow$ amplitude
 $\omega_n \rightarrow$ pulsation (rad/s), $\omega_n = 2\pi n f_1$
 $\phi_n(t) \rightarrow$ phase

Le symbole \sum ¹⁵ signifie sommation dénombrable des composantes indiquées par n . Aussi, le spectre d'un tel son, à savoir l'ensemble des fréquences f_n construisant le signal, est appelé *spectre discret*. Ce type de spectre est encore désigné comme *spectre de raies* car composé d'un ensemble de raies spectrales, chacune relative à la présence d'une composante élémentaire (une fréquence) dans le signal¹⁶ résultant.

En musique, les fréquences f_n sont nommées les *harmoniques de rang n* . Un son comportant de nombreuses harmoniques est apprécié comme « riche » et inversement un son possédant peu d'harmoniques est qualifié de « pauvre ».

A contrario des sons de la vie quotidienne, les sons périodiques décrivent essentiellement ceux produits par les instruments à cordes frottées et pincées ou par les instruments à vent. La superposition de sons purs harmoniques peut s'envisager de différentes manières, comme :

1. la sommation de toutes les harmoniques $f_n = nf_1$. On obtient progressivement un signal « en dent de scie » qui approche de manière simpliste le son d'un violon;
2. la sommation des harmoniques impaires : $f_n = (2n-1)f_1$. On converge cette fois vers un signal « rectangulaire » qui, quant à lui, approche le son d'une clarinette simplissime.

DÉCOMPOSITION FRÉQUENTIELLE — Par dualité temps-fréquence et pour un signal continu, il est également possible de calculer la réponse en fréquence au moyen de la transformée de FOURIER (cf. figure 4.7b). La formulation mathématique est similaire au cas discret.

$$p(t) = \int_{-\infty}^{+\infty} p(f) \exp(2\pi f t) df = \int_{-\infty}^{+\infty} p(\omega) \exp(i\omega t) d\omega \quad (4.6)$$

La fonction $p(f)$, transformée de Fourier de $p(t)$, est fournie par :

$$p(f) = p(\omega) = \int_{-\infty}^{+\infty} p(t) \exp(-2\pi f t) dt = \int_{-\infty}^{+\infty} p(t) \exp(-i\omega t) dt \quad (4.7)$$

Pour chaque fréquence f , la valeur de $p(f)$ indique l'amplitude et la phase correspondantes du signal.

En toute généralité, les signaux sonores, musicaux ou non, possèdent des *spectres continus* — instruments de musique, voix parlée

12. Et de bien d'autres domaines d'étude.

13. Joseph FOURIER est un grand mathématicien du XVIII^e siècle. Les disciplines relevant de la vibroacoustique et du traitement de signal lui doivent beaucoup.

14. Un signal est quasi-périodique du fait de la présence de pertes. Ainsi, la décroissance ne donne pas une reproduction à l'identique toutes les périodes. En toute rigueur, l'amortissement introduit un léger décalage des cycles; négligeable en première approximation.

15. Lettre grecque sigma majuscule.

16. Cette technique de construction de signaux sonores est à la base de la méthode de synthèse de sons musicaux dite *additive*, initiée dans les années 1960 par Max MATTHEWS et Jean-Claude RISSET.

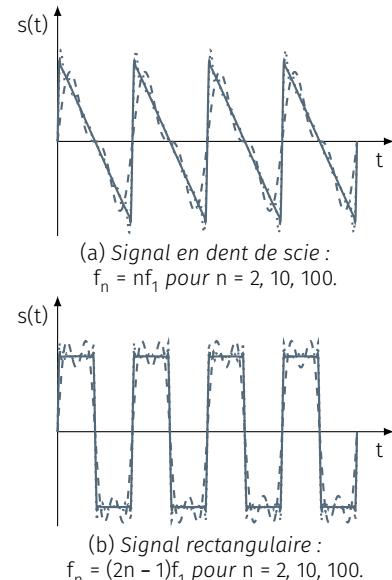


FIGURE 4.12 — Notion de périodicité par décomposition en harmoniques.

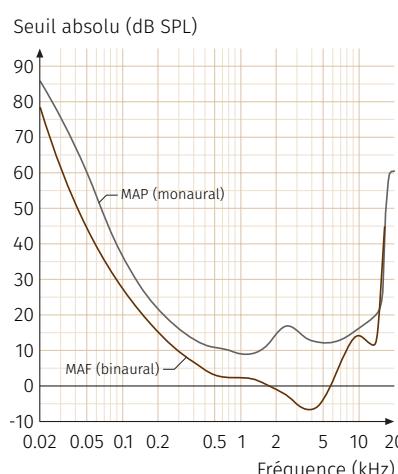


FIGURE 4.13 — Seuils d'audition en fonction de la fréquence : MAP (Minimum Audible Pressure), MAF (Minimum Audible Field) — Norme ISO 389-7.

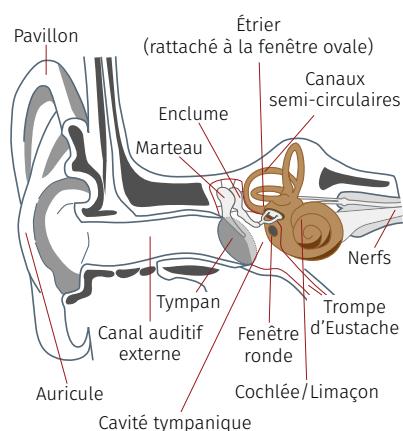


FIGURE 4.14 — Synopsis d'anatomie de l'appareil auditif humain.

¹⁷. Un transducteur est un dispositif transformant un type d'énergie en un autre. Par exemple, un microphone traduit une énergie acoustique en énergie électrique — inversement pour un haut-parleur.

¹⁸. Rien à voir avec un système soviétique!

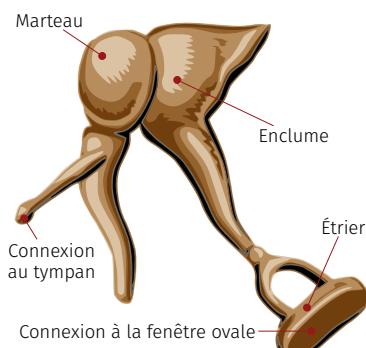


FIGURE 4.15 — Oreille moyenne : chaîne des osselets (marteau, enclume, étrier).

et chantée, bruits... L'agencement des fréquences ne conduit pas toujours à une décomposition simplement structurée.

L'analyse de FOURIER permet de décomposer les signaux comme le mélange d'un ensemble de *fréquences ponctuelles* ou *spectre de raies* avec un *spectre continu de bruit*. En théorie du signal et en synthèse sonore dite de FOURIER ou *additive*, on distingue une partie *déterministe* liée aux *raies spectrales* et une partie *aléatoire* associée au bruit.

D'un point de vue perceptif, les simples sommations de fréquences ponctuelles ne fournissent pas des signaux *réalistes*. La présence de bruits spécifiques à chaque instrument — bruit de choc des marteaux du piano sur les cordes, bruit de souffle des instruments à vent, etc. — intervient de manière remarquable dans la reconnaissance auditive d'un instrument particulier. Ces bruits sont associés aux premières millisecondes des transitoires d'attaque des signaux.

2.2.2 Quantification et échantillonnage

A. AUDITION, CLASSIFICATION ET PRÉSENTATION DES SONS

La courbe de réponse en fréquence de l'appareil auditif humain se situe sur une étendue d'environ 20 Hz à 20 kHz chez un jeune sujet (voir figure 4.13). En effet, pour une personne adulte, il y a rapidement une perte d'audition dans les aigus — phénomène de presbyacusie —, situant la limite supérieure plutôt autour de 16 kHz.

On peut constater que les seuils auditifs sont meilleurs en écoute binaire — deux oreilles. Cela provient du déphasage entre les signaux qui parviennent à l'oreille droite et gauche qui permet une meilleure détection. La forme du pavillon de l'oreille et du canal auditif interviennent également en opérant un filtrage (voir figure 4.14).

i—STRUCTURE DE L'OREILLE HUMAINE — L'oreille humaine — et de manière générale celle des mammifères — est structurée en trois parties et fonctions associées (voir figure 4.14), à savoir :

1. l'oreille externe, en charge d'optimiser la réception et la localisation des sons;
2. l'oreille moyenne, dévolue à la transmission et à l'adaptation en énergie des signaux sonores;
3. l'oreille interne, qui s'occupe du codage et de la « transduction ¹⁷ » de l'information — transformation d'énergie mécano-acoustique en énergie électrique (potentiel d'action des nerfs auditifs).

L'oreille externe est l'élément visible de l'ouïe. Elle est constituée du pavillon et du conduit — ou canal — auditif externe.

Les domaines d'intervention essentiels de l'oreille externe sont :

- l'amplification;
- le filtrage;
- la localisation des sons;
- la protection des dispositifs amonts de l'oreille aux agressions externes.

L'oreille moyenne est une cavité dont l'entrée est contrôlée par le tympan de section d'environ 55 mm². Cette cavité est appelée la *caisse du tympan* — incluse dans l'os du rocher —, au sein de laquelle se trouve la *chaîne des osselets* elle-même composée¹⁸ :

- du *marteau* (≈ 20 g), solidaire du tympan;
- de l'*enclume* (≈ 25 g);
- et de l'*étrier* (≈ 2 g), à son tour solidaire de la membrane de la *fenêtre ovale*, orifice d'entrée de l'oreille interne ($S_o \approx 3$ mm²).

Les osselets sont maintenus et articulés par des petits muscles et ligaments qui déterminent la raideur d'ensemble du système et permettent de contrôler la réponse de l'oreille selon le signal d'excitation arrivant au tympan. La chaîne des osselets, par effet levier, amplifie et

transforme le signal acoustique du tympan en vibration mécanique en entrée d'oreille interne.

REMARQUE — Le bas de la caisse du tympan dispose d'un orifice duquel un conduit — la *trompe d'Eustache* — débouche à son autre extrémité dans l'arrière-gorge. Cette connexion a pour fonction d'égaliser la pression de l'air entre les deux faces du tympan. Ce phénomène se produit lors d'une variation brusque de pression atmosphérique : prise d'altitude rapide en téléphérique ou avion non pressurisé, surpression au passage d'un TGV dans un tunnel, etc. Pour rééquilibrer les pressions, il est nécessaire de bailler ou de saliver.

Les fonctions principales de l'oreille moyenne sont :

- la protection de l'oreille aux niveaux sonores élevés (adaptation auditive). Aux fortes amplitudes, les muscles de la chaîne des osselets se contractent automatiquement¹⁹ (« réflexe stapédiens »).
- l'adaptation d'impédance via la chaîne des osselets. Par amplification du signal, les osselets opèrent une transformation d'énergie entre l'air — pression acoustique dans le conduit auditif au niveau du tympan — et le liquide physiologique au sein de l'oreille interne — au niveau de la fenêtre ovale. Ce phénomène mécanique possède deux origines explicatives :
 1. un rapport de surface relativement important entre tympan et fenêtre ovale : $S_t/S_o \approx 55/3 \approx 20 \quad P_o \approx 20 P_t$, soit +26 dB;
 2. un effet de levier de l'articulation de la chaîne des osselets : le rapport est à peu près de deux, soit +6 dB.

Au global, les effets cumulés représentent donc un gain de l'ordre de +32 dB. On peut également noter un effet de filtrage fréquentiel de l'oreille moyenne (par masse et raideur ajoutées).

Dans son ensemble, l'oreille moyenne est assimilable à un capteur de pression — à l'instar d'un microphone — via le tympan, lequel joue le rôle de la membrane du microphone. En poursuivant l'analogie, avec la transduction mais surtout l'amplification du signal, la chaîne des osselets peut se comparer au préamplificateur d'un microphone.

L'oreille interne est le dispositif le plus fragile et le plus complexe de l'appareil auditif. L'ensemble est ainsi protégé par une coque rigide appelée *labyrinthe osseux*, cavité fermée remplie de liquide physiologique — la périlymph et l'endolymphe —, qui comprend :

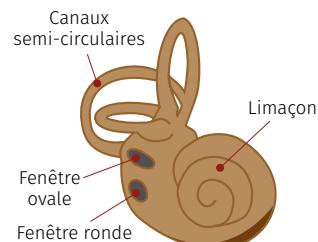
- le *vestibule* avec un orifice d'entrée, la *fenêtre ovale* et, un orifice de « sortie », la *fenêtre ronde* équipée d'une membrane pour permettre l'équilibre des forces de déformation;
- les *canaux semi-circulaires*;
- le *limaçon* ou la *cochlée* — ainsi intitulé à cause de sa géométrie, *cochlea* signifiant « escargot » en latin.

La partie supérieure formée des canaux semi-circulaires n'intervient pas dans les mécanismes de l'audition, mais comme moyen de contrôle de l'équilibre. En effet, chaque canal est disposé dans un plan quasi-perpendiculaire aux deux autres permettant de se repérer selon chacune des trois dimensions de l'espace physique du fait de la présence du liquide physiologique. Cela explique pourquoi des traumatismes de l'oreille peuvent engendrer des troubles de l'équilibre.

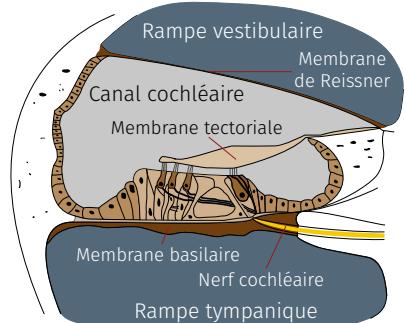
Le dispositif de décodage de l'information sonore est au sein de la cochlée, qui se compose principalement :

- de l'*organe de Corti*;
- des *membranes basilaire et tectoriale*;
- des *cellules ciliées internes et externes* (≈ 25000);
- du *nerf cochléaire*.

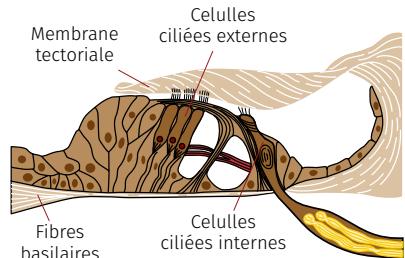
19. Un tel mécanisme d'adaptation réflexe met en valeur qu'il n'est pas possible de déterminer l'intensité réellement perçue à l'aide d'une grandeur physique. L'intensité perçue — sonie mesurée en phones — se voit donc être subjective et contextuelle.



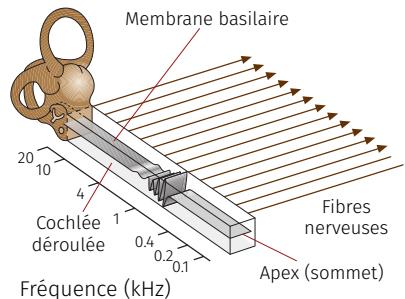
(a) *Labyrinthe osseux : canaux semi-circulaires, fenêtre ovale, fenêtre ronde et limaçon.*



(b) *Coupe de la cochlée : rampes tympanique et vestibulaire, canal cochléaire, membranes basilaire, tectoriale et de Reissner.*



(c) *Organe de Corti (focus) : cellules ciliées externes et internes, membrane tectoriale, fibres basilaires et nerfs auditifs.*



(d) *Mécanique de la cochlée : membrane basilaire et sélectivité en fréquence.*

FIGURE 4.16 — Oreille interne : labyrinthe osseux, coupe et déroulé du limaçon.

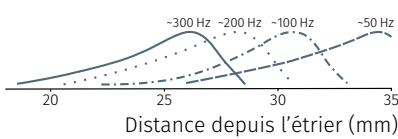


FIGURE 4.17 — Sélectivité en fréquence : profil d'excitation des cellules ciliées.

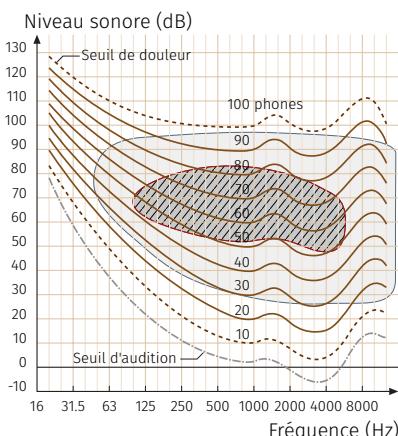


FIGURE 4.18 — Courbes d'isosonie binaurale(en phones) et champs auditibles de la parole (—) et de la musique (—).

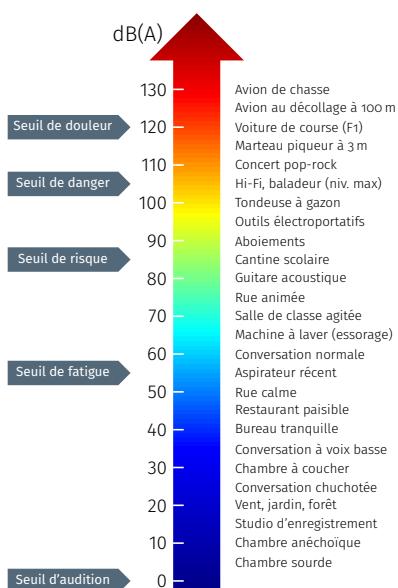


FIGURE 4.19 — Échelle de niveaux sonores rapportée aux décibels pondérés A : à partir de 40 dB(A) la nuit et 55 dB(A) le jour – gêne, stress, fatigue –, de 85 à 105 dB(A) – risques à moyens termes si exposition chronique, pertes auditives –, au-delà de 105 dB(A) – risques immédiats – acouphènes, surdité..

L'organe de Corti est constitué de l'ensemble du mécanisme actif de l'audition dont une coupe transversale est représentée en figure 4.16c. Il est disposé tout le long de la membrane basilaire, laquelle s'effile de plus en plus jusqu'à son extrémité ou « apex » (cf. figure 4.16d).

L'onde acoustique désormais transformée en vibration mécanique par l'oreille moyenne est communiquée à l'oreille interne par l'étrier au niveau de la fenêtre ovale. Le fait que la cochlée soit remplie de liquide physiologique, cette vibration va générer une onde au sein de la cochlée se traduisant par un déplacement transversal de la membrane basilaire à son passage. L'équilibre des contraintes se fait au niveau de la fenêtre ronde dont l'étanchéité est assurée par une membrane.

Le déplacement transversal d'une zone de la membrane basilaire va induire un mouvement de balancier de l'organe de Corti. Ainsi, les cils des cellules ciliées externes vont être en contact avec la membrane tectoriale qui les recouvre et déclencher une impulsion nerveuse.

Plus le son est intense, plus les cellules ciliées vont être sollicitées. À niveaux sonores élevés, les cellules ciliées internes entrent en action. L'ensemble des impulsions nerveuses est collecté par les fibres nerveuses pour déboucher sur « l'autoroute de l'information » que constitue le nerf auditif.

La sélectivité en fréquence est relative aux zones sollicitées le long de la membrane basilaire. Le phénomène est induit par le lien entre les fréquences du signal initial et leurs longueurs d'onde associées dans la vibration qui se propage le long de la membrane basilaire. La détection des hautes fréquences — petites longueurs d'onde — se produit près de la fenêtre ovale tandis que pour les très basses fréquences — grandes longueurs d'onde — cela se réalise vers l'apex (cf. figure 4.17). De ce point de vue, l'oreille interne et son mécanisme de détection des fréquences est assimilable à un « analyseur de spectre ».

ii - CHAMP AUDIBLE ET NIVEAU SONORE — En psychoacoustique, il n'est pas toujours possible de mesurer directement les phénomènes. Outre les difficultés techniques, il est en effet peu envisageable d'accéder à l'oreille moyenne ou la cochlée d'un être vivant. La démarche expérimentale doit donc s'opérer différemment à l'aide de tests effectués sur un panel d'individus. Les résultats sont ensuite analysés de manière statistique pour obtenir des conclusions moyennées.

Pour déterminer le champ audible par l'être humain, aussi bien en fréquence qu'en intensité, la procédure est similaire à celle de l'audiogramme ; elle consiste à présenter aux sujets des sons purs de niveaux croissants en intensité et de noter celui pour lequel le signal est détecté. En balayant l'ensemble des fréquences, on obtient la courbe dite du seuil d'audition (voir figures 4.13 et 4.18).

Pour établir la courbe dite du seuil de douleur, on procède par extrapolation en se gardant, bien évidemment, une marge de manœuvre afin de ne pas détruire les facultés auditives des sujets. Compte tenu de cette incertitude, on évalue communément le seuil de douleur pour des niveaux sonores autour de 120 dB.

L'établissement des courbes intermédiaires — entre 0 dB et 120 dB — se fait par comparaison des niveaux sonores du son pur présenté aux oreilles du sujet vis-à-vis de celui d'un signal de référence, en l'occurrence un son pur à 1 000 Hz de niveau sonore étalonné. Pour chaque niveau sonore du signal de référence, le son pur testé est présenté au sujet avec un niveau sonore croissant par paires successives jusqu'à ce qu'il soit jugé de même sensation d'intensité sonore.

Les courbes d'égal sensation d'intensité ainsi obtenues sont appelées courbes isosoniques ou d'isosonie (cf. figure 4.18). Chaque courbe

correspond à une évaluation en *phones*. En pratique, cette unité n'est pas utilisée, les sonomètres mesurent le niveau de bruit « physique » en décibels SPL — pour *Sound Pressure Level* —, auquel on applique un filtre en fréquence qui approche les courbes d'isosonie. On parle alors de décibels pondérés, dont le filtre le plus commun est nommé « A » et reflète les niveaux sonores de la vie courante.

La dynamique de l'audition humaine s'avère bien moins bonne en basses fréquences (voir figure 4.18). Par exemple, sa sensibilité à 100 Hz est environ mille fois inférieure de celle à 1000 Hz. Néanmoins, la nature a tout fait pour aider la préservation de l'espèce et l'inciter à la communication en la dotant de ses meilleures facultés auditives dans les intervalles de fréquences (100 à 5 000 Hz) et d'intensités (40 à 80 dB) correspondant au domaine de la parole : besoin initial d'alerte face à un danger, échange d'information, etc.

Pour notre plaisir, le domaine de la musique recouvre quant à lui l'essentiel du champ audible où seuls sont rejetés les niveaux sonores trop faibles — on entend rien ! — et trop forts — on ne s'entend plus !

iii – TYPOLOGIE OBJECTIVE DES SONS — Comme déjà évoqué, on distingue d'abord les sons entre eux en fonction de leurs contenus fréquentiels respectifs, puis selon la manière dont les fréquences qui les composent sont reliées entre elles.

a. **SON « PUR »** — Le son qualifié de *pur* ne contient qu'une seule composante fréquentielle. Cela constitue le signal sonore de base, soit une vibration sinusoïdale simple : $p(t) = A(t) \sin(\omega t + \phi(t))$.

b. **SON « COMPLEXE » OU « COMPOSÉ »** — Si un signal sonore comporte plus d'une composante en fréquence, celui-ci est qualifié de *composé* ou de *complex*. Cependant, il est nécessaire que l'on puisse dénombrer les différentes fréquences qui le composent — même si il y en a plusieurs centaines, voire plusieurs milliers ! Pour des signaux périodiques, on distingue deux catégories de sons complexes :

1. les sons « *harmoniques* » qui possèdent un spectre de raies (pics de fréquences) tel que : $f_n = \alpha(n)f_1$, avec $\alpha(n)$, c'est-à-dire que les fréquences (harmoniques) sont reliées à la fréquence fondamentale en tant que multiples entiers (par exemple, $f_2 = 2f_1$, $f_3 = 3f_1$, $f_4 = 4f_1$, etc.);
2. les sons « *inharmoniques* » caractérisés par un spectre de raies sans relation simple entre les fréquences (on parle de *partiels*).

c. **BRUITS** — Les bruits constituent un type particulier de sons complexes. Un bruit se définit comme un signal dont le spectre est composé de toutes les fréquences dans un intervalle donné de fréquences et distribuées de façon *aléatoire* — c'est à dire au hasard au cours du temps. Là encore, on distingue les bruits suivant deux grandes classes :

1. les bruits à « *faible bande de fréquences* », constitués de composantes dans un relativement faible intervalle de fréquences autour d'une composante particulière appelée *fréquence centrale* ou *caractéristique*;
2. les bruits à « *large bande de fréquences* » qui, à l'inverse, couvrent une importante gamme de fréquences (par exemple un bruit dit « *blanc* » comportant toutes les fréquences audibles²⁰).

Pour préciser plus avant de quoi il retourne, ces différents bruits se définissent comme suit :

- bruit « *blanc* » → distribution aléatoire sur toutes les fréquences audibles [20 Hz, 20 kHz] et toutes les phases $[0, 2\pi]$;
- bruit « *rose* » → distribution aléatoire sur toutes les octaves $[\log(20), \log(20000)]$ (énergie identique sur chaque octave);

²⁰. Le qualificatif de « *blanc* » est appliqué par analogie avec la vision, pour laquelle la couleur blanche est constituée de toutes les composantes du spectre visible.

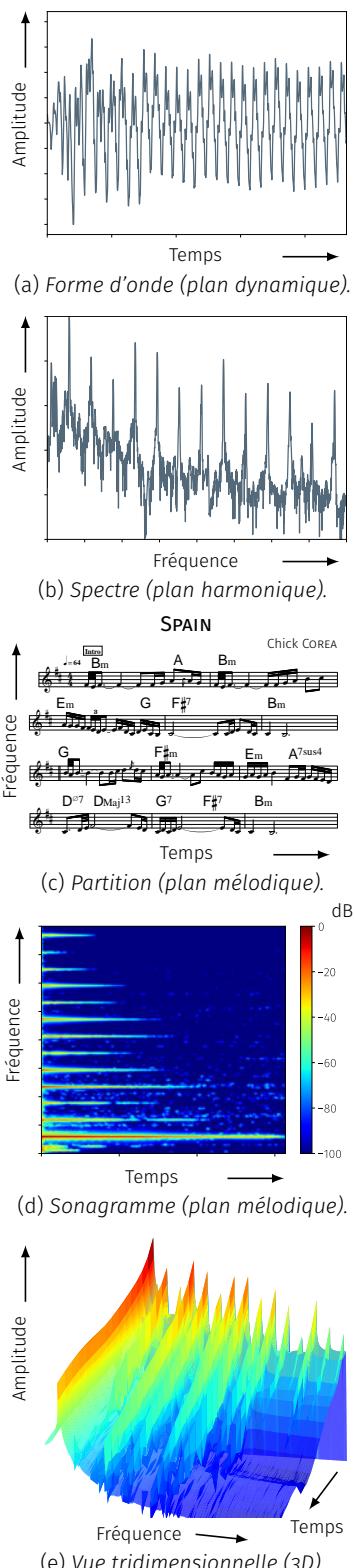


FIGURE 4.20 — Diverses représentations des sons : note de guitare et partition.

- bruit « faible bande » (par opposition à « large bande ») → distribution aléatoire autour d'une fréquence centrale $[f_c - \Delta f, f_c + \Delta f]$.

Les sons de la vie courante sont quasi-exclusivement complexes. En effet, en tant que signaux de référence sur le plan théorique, les sons purs sont en toute rigueur impossibles à obtenir naturellement. Leur fabrication se fait de manière artificielle à l'aide d'un appareillage électronique — générateur de fréquences — et aujourd'hui numérique via un processeur de calcul ou un ordinateur. Seuls quelques sons naturels peuvent être assimilés à des sons purs, comme ceux produits par un diapason, un sifflet ou un appeau de très bonne qualité.

Quant à eux, les signaux musicaux appartiennent souvent à la catégorie des sons complexes harmoniques pour tous les instruments qui demandent une justesse fine de tonalité comme les instruments à vent et dans une certaine mesure les instruments à cordes (inharmonicité des cordes). Celle des sons complexes inharmoniques rassemble la grande majorité des instruments à percussion.

IV — FIGURATION DES SONS — Il est d'usage de dire qu'un bon schéma vaut mieux qu'un long discours. Accéder aux informations contenues dans un signal sonore de manière à la fois globale et synthétique est un soutien pour l'analyse des sons. Un des atouts essentiels est d'autoriser des comparaisons rapides entre signaux d'origines différentes.

Pour visualiser les signaux sonores, trois catégories de schématisation sont à disposition (cf. figure 4.20) :

1. les représentations « temporelles », qui affichent l'évolution de l'amplitude — ou du niveau — du signal en fonction du temps : $L = \alpha(t)$. Cela transcrit de manière visuelle l'enregistrement du signal à l'état « brut ». Du fait de la dépendance en temps, on parle de représentation dans le *plan dynamique*. Dans le cas d'un signal vibratoire, elle est appelée « forme d'onde » ;
2. les représentations « fréquentielles » ou « spectrales », lesquelles correspondent à une analyse du contenu fréquentiel entre deux instants. Autrement dit, il s'agit ainsi d'une « photographie » du contenu spectral du signal à un moment donné de son évolution. On affiche donc des niveaux en fonction de la fréquence : $L = \alpha(f)$ et révèle les relations entre fréquences. On évoque ici le terme de représentation dans le *plan harmonique*. Ce type de figure est nommé « spectre » du signal ;
3. les représentations « temps-fréquences », qui permettent non seulement d'accéder au contenu fréquentiel du signal, mais aussi à l'évolution de ces composantes fréquentielles dans le temps. Parmi celles-ci, on en distingue deux genres :
 - les représentations dites dans le *plan mélodique* qui, de manière implicite sont à trois dimensions où le contenu fréquentiel est affiché en fonction du temps : $f = \alpha(t)$. Les partitions musicales en constituent l'illustration historique directe, avec des indications de niveau : *pianissimo* à *fortissimo*. Le *sonogramme* — ou *spectrogramme* — est un des moyens traditionnels d'analyse des sons en affichant directement des niveaux dépendant à la fois du temps et des fréquences : $L = \alpha(f, t)$. La détection de chaque composante du signal est indiquée par une échelle d'intensité de marquage plus ou moins fort — ou désormais à l'aide d'un code couleurs — qui permet d'évaluer les variations de niveau ;
 - les représentations « 3D » — sorte de sonogramme moderne. L'informatique permet d'afficher les données directement par

des graphes tridimensionnels qui offrent une visualisation instantanée des niveaux et de leur évolution : $L = \alpha(f, t)$.

B. CODAGE NUMÉRIQUE DES SONS

Quel que soit le champ de recherche ou d'application, l'acoustique actuelle fait abondamment appel aux signaux numériques ; les analyses décrites précédemment en sont l'illustration.

Convertir un signal analogique — donné en volt au cours du temps car enregistré au moyen d'un microphone — en un signal numérique, induit une séparation suivant deux dimensions : la représentation du voltage (amplitude discrétisée) et l'évolution temporelle (évaluation discrète du temps). La première opération est connue sous le nom de *quantification* et la seconde par le terme d'*échantillonnage*. Cette conversion n'est cependant pas directe et nécessite l'observation de quelques règles d'usage.

i- **QUANTIFICATION ET RAPPORT SIGNAL / BRUIT** — Les convertisseurs analogique à numérique — *Analog-to-Digital Converter* (ADC) — traduisent les valeurs d'entrée de volts en nombres entiers.

L'étendue des entiers possibles est fournie par le nombre de bits par échantillon relatifs aux capacités du convertisseur. Les progrès étant, on est passé de la norme à 16 bits du *Compact Disc*²¹ de 1982 aux cartes audio « professionnelles » actuelles à 24 voire 32 bits employées en studio d'enregistrement.

En écoute, l'intérêt réel d'un codage en 24 bits est relatif et se ressent parfois en musique classique et en bande son de cinéma (utilisation de plus de dynamique, cf. infra). Toujours est-il que pour conserver une capacité de stockage de 74 mn, les constructeurs sont restés à une quantification sur 16 bits.

Une conversion en échantillons sur M bits permet une quantification du voltage par 2^M valeurs. Par exemple, un convertisseur ADC sur 16 bits offre 2^{16} soit 65 536 valeurs différentes. Un tel convertisseur ayant à traduire des voltages entre -10 V et +10 V correspondrait à des valeurs entre -32 768 et 32 767 respectivement.

La conversion est linéaire. Ainsi, si le voltage est 0,3052 V cela donne 1000 ($32\,767 \times 0,3052 / 10$) et s'il est de 0,3055 V, on obtient 1 001 ($32\,767 \times 0,3055 / 10$). Néanmoins, un voltage de 0,3053 conduit également à une valeur de 1 000. Cet écart est appelé *erreur de quantification* ou *bruit de quantification* (voir figure 4.21).



²¹. De 1979 à 1982, PHILIPS et SONY se sont concertés pour définir la norme du *Compact Disc* (16 bits, 44,1 kHz) et commercialiser les équipements (voir W).

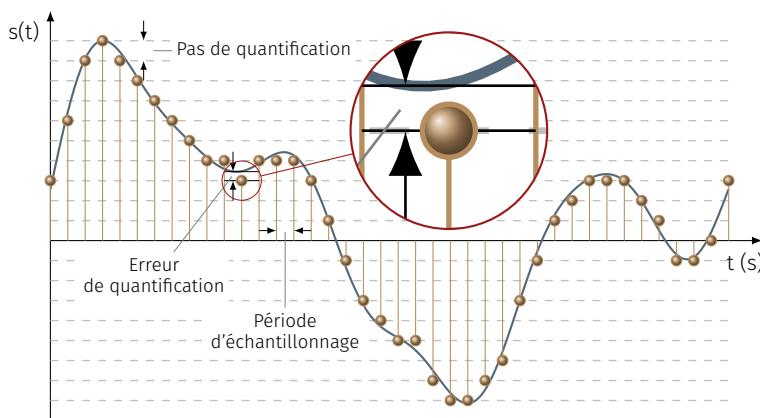


FIGURE 4.21 — Quantification et échantillonnage.

Le bruit de quantification fait référence au rapport signal sur bruit. La pratique veut que celui-ci soit le plus grand possible, notamment en correspondance avec la dynamique de réponse de l'audition (cf. figure 4.18), soit dans ses limites d'un ordre de grandeur de 90 dB. Les

22. Pour une oreille moyenne, 60 à 70 dB correspondent à une dynamique de rapport signal/bruit courante.

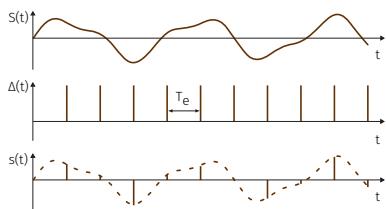


FIGURE 4.22 — Opération d'échantillonnage : multiplication du signal continu $g(t)$ par un peigne de Dirac $\Delta(t)$ donnant le signal discret $s(t)$.

calculs conduisent à avoir une dynamique de 6 dB par bit ($6 \times M$), soit pour 16 bits, 96 dB²² et, pour 24 bits, 144 dB. On constate donc que la norme *Compact Disc* remplit correctement sa fonction.

ii – OPÉRATION D’ÉCHANTILLONNAGE — Le processus d’échantillonnage remplace le signal analogique — fonction continue en temps — par une séquence de points dans la perspective de mémoriser, de transmettre ou de traiter le signal. Cette opération est illustrée en figure 4.22, où le signal continu est multiplié par un peigne de distributions de Dirac (« fonction » δ) équitabllement réparties d’une période T_e dans le temps pour obtenir une séquence de valeurs discrètes.

Pour numériser correctement un signal sonore, à savoir convertir le signal continu en signal discret et inversement, revenir à un signal analogique sans perdre d’information audible, il faut se conformer au *théorème de Shannon*.

THÉORÈME DE SHANNON

Soit un signal continu $S(t)$ à bande limitée, c'est-à-dire possédant une fréquence maximale f_{\max} , on considère un échantillonnage périodique défini par :

$$t_k = kT_e; s_k = u(t_k)$$

où k est entier naturel, T_e la période d’échantillonnage et $f_e = 1/T_e$ la fréquence d’échantillonnage.

Pour que le signal $S(t)$ puisse être entièrement reconstruit à partir des échantillons s_k , il est nécessaire et suffisant que :

$$f_e > 2f_{\max} \quad (4.8)$$

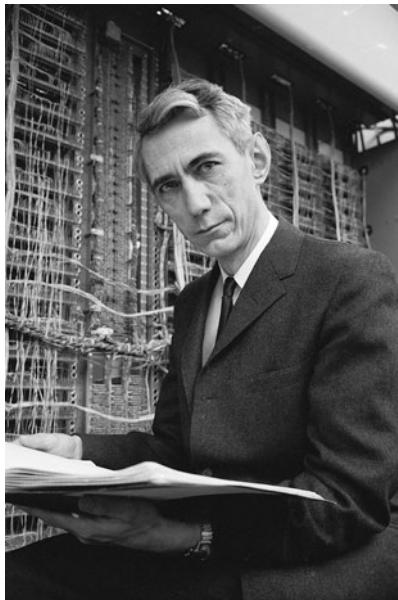
Ainsi, la fréquence d’échantillonnage d’un signal continu doit être strictement supérieure à deux fois la plus grande fréquence présente dans son spectre (condition de Nyquist-Shannon). Les signaux sonores se trouvant compris dans l’intervalle approximatif de [20 Hz ; 20 kHz], la fréquence d’échantillonnage doit *a minima* être supérieure à 20 kHz.

Compte-tenu que de jeunes et saines oreilles sont capables de percevoir au-delà de 20 kHz mais aussi, que les filtres appliqués aux signaux provenant des convertisseurs n’ont pas de fréquence de coupure stricte, la norme CD audio affiche une fréquence d’échantillonnage de 44,1 kHz. Pour le format DAT de SONY — *Digital Audio Tape* — des années 1990, la fréquence d’échantillonnage est de 48 kHz.

On comprend donc que plus la fréquence d’échantillonnage est élevée, plus le signal sonore sera conformément reproduit. Néanmoins, est-ce réellement nécessaire et pourquoi les cartes audio proposent des fréquences d’échantillonnage de 96 kHz ou 192 kHz ? Simplement car elles sont optimisées pour fonctionner à ces fréquences d’échantillonnage. De surcroît, cela constitue un argument commercial.

Mentionnons un autre exemple : la téléphonie. La bande passante de la parole se situant dans l’intervalle de fréquences [100 Hz ; 8 kHz], une fréquence d’échantillonnage de 16 kHz est suffisante pour ne pas nuire à l’intelligibilité des signaux.

Pour conclure, il est plus intéressant de disposer d’une profondeur de 24 bits — meilleure dynamique pour le classique, le jazz et toute musique à nuances fortes — que d’une fréquence d’échantillonnage élevée. Toutefois, qui peut le plus, peut le moins...



Claude Elwood Shannon (1916–2001).

3 Manipulation des données

Savoir coder l'information est une condition nécessaire, mais elle s'avère non suffisante dans la pratique. En effet, pour que les données soient exploitables, il faut également pouvoir les manipuler facilement à la fois en termes de volume d'information à traiter, d'organisation, de stockage et d'accès, le tout de manière sécurisée.

3.1 Compression des données

Au regard des codages des divers éléments multimédias présentés en section précédente, on constate que la quantité d'information à traiter peut devenir rapidement considérable. Pour se servir des données dans des conditions acceptables, il faut souvent avoir recours à des données compressées, au sens de volume d'information à traiter. On distingue alors deux catégories :

1. les compressions sans perte, où un retour aux données initiales est envisageable;
2. les compressions avec pertes, qui induisent une dégradation de l'information jugée « inutile » dans un contexte applicatif donné.

3.1.1 Problématique

Imaginons que nous ayons un film « Full HD²³ » de deux heures sans bande sonore. Deux heures équivalent 7200 secondes et, pour que le flux vidéo soit suffisamment fluide, on estime qu'il faut 24 images par seconde. Pour deux heures de film, cela donne donc 172 800 images.

Pour chaque image, la haute définition amène à avoir 1920 pixels en largeur et 1080 pixels en hauteur soit 2 073 600 pixels par image. Pour tenir compte des couleurs, chacun des pixels va être codé sur 24 bits (8 bits par composante RGB) donc, en tout, l'espace de stockage représente 8 599 633 920 000 bits, soit de l'ordre²⁴ du téraoctet, ce qui est énorme. À titre de comparaison, cela représente cent heures de téléchargement sur une ligne ADSL : un peu compliqué pour, par exemple, la vidéo à la demande...

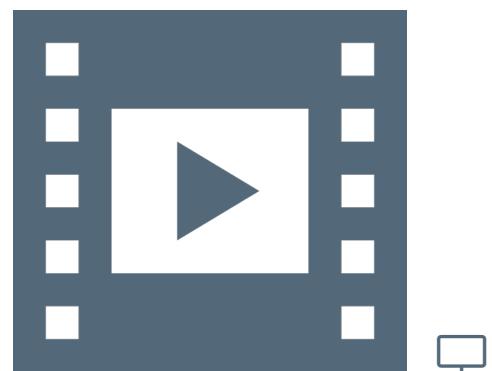
La problématique est ainsi de savoir comment faire pour qu'un film tienne sur un DVD ou pour gérer de manière fonctionnelle un service de vidéo à la demande.

3.1.2 Compression des images

Il s'avère que certaines images comme celles issues du dessin technique possèdent des zones répétitives (cf. figure 4.23) : couleurs uniformes, parties identiques, mêmes épaisseurs de trait, etc. Il devient donc inutile de stocker de l'information redondante.

Une telle approche se retrouve dans les formats d'images GIF — *Graphics Interchange Format* — et PNG — *Portable Network Graphics* —, mais se trouve uniquement adéquate pour des images relativement « simples », notamment destinées au Web. Les caractéristiques du PNG lui permettent d'enregistrer des photographies sans perte de données, au détriment de la taille du fichier (cf. W). Dans ce cas de figure, son intérêt essentiel est de permettre la transparence (par exemple pour des photographies détournées), ce que le format JPEG — *Joint Photographic Experts Group* — n'autorise pas (voir infra).

Très souvent, les photographies ne comportent pas de zones répétitives et la compression doit s'envisager autrement. En effet, l'acuité visuelle humaine (la résolution) ne perçoit pas tous les niveaux de détails. Cela conduit à les ignorer afin de ne pas les stocker pour réduire



VIDÉO 4.3 — Compression des données.

²³. Par souci de simplicité, est ici considéré un film Full HD même si en pratique, ce sont des films deux fois plus petits qui sont stockés sur DVD (en 720 px, soit 1280 × 720 pixels au lieu de 1080 px, soit 1920 × 1080 pixels).

²⁴. On confond souvent les puissances de 10 (par exemple 1 kilo correspond à $10^3 = 1000$) et les puissance de 2 (par exemple 1 kibi correspond à $2^{10} = 1024$) qui sont du même ordre de grandeur. Par exemple un disque dur de 1 téra = 10^{12} octets une fois formaté est exprimé en puissance de 2, en tébi et le nombre sera 91% inférieur environ : $10^{12} \div 2^{40} = 0.9094$ (W).

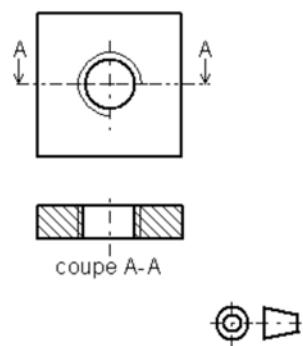


FIGURE 4.23 — Exemple de dessin technique (NdR : à reprendre).



(a) Photographie originale.



(b) Photographie (trop) compressée.

FIGURE 4.24 — Compression avec pertes.

en conséquence la taille du fichier. Ce faisant, cela induit logiquement une perte d'information.

La compression se réalise par approximation de certains blocs de pixels en les considérant comme identiques en fonction du taux de compression voulue. Le format le plus connu de ce type de compression est le JPEG. L'image résultante sera moins détaillée mais beaucoup plus petite, permettant d'adapter les images à leur utilisation : document A4, impression de photographie 24×36 , etc. Le taux de compression appliquée est donc un compromis entre qualité du rendu et taille du fichier résultant. La figure 4.24 en est l'illustration où la taille de fichier a été divisée par dix entre l'original et sa résultante. On constate que la compression est trop forte car les niveaux de détails perdus nuisent au visionnage de la photographie : dans le jargon des infographistes, on parle de pixellisation.

3.1.3 Compression des vidéos

Comme dans le cas d'une image, une vidéo contient des zones répétitives entre des images successives. La démarche est de stocker une image — typiquement au début de chaque nouveau plan — puis détecter les similitudes pour uniquement tenir compte des différences. Ces techniques se retrouvent dans les formats vidéos les plus connus de type MPEG — *Moving Picture Experts Group*.

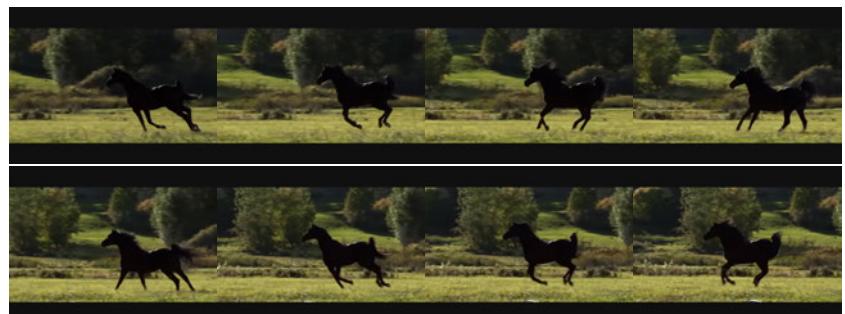


FIGURE 4.25 — Décomposition en image d'un mouvement.

À titre d'illustration, la figure 4.25 met en exergue les principes de la compression vidéo. On constate en effet que l'arrière plan et le corps du cheval varient relativement peu d'une image à l'autre. En revanche, ce sont les pattes du cheval qui expriment le mouvement et qui doivent être coder plus finement.

3.1.4 Compression des sons

À l'instar des images, la compression des données audionumériques est indispensable à leur exploitation : stockage, transmission, etc. Tout comme pour les images, on y distingue également deux grandes catégories de compression, sans et avec perte (voir table 4.5).

Les formats de compression audio sans perte sont relativement peu connus du grand public. On peut néanmoins citer le format FLAC — *Free Lossless Audio Codec*. Les codages généraux applicables à tout type de données (cf. [algorithme de Huffman](#)) sont peu efficaces en audionumérique, même en introduisant des techniques d'optimisation spécifiques [ROADS, 2007]. Cependant, les formats sans perte sont bien sûr utiles dans certains contextes comme le travail de studio.

Pour la compression avec perte, on s'appuie directement sur les aptitudes de la perception auditive humaine ; domaine d'étude de la *psychoacoustique*. Hormis la restriction à la bande de fréquences audibles

TABLE 4.5 — Formats de compression audio — d'après [ROADS, 2007].

Formats audionumériques		
NOM	TYPE	NOTE
AAC	avec perte	Métadonnées et schémas anticopie
ATRAC	sans/avec perte	MiniDisc, cinéma, baladeur, console de jeux
FLAC	sans perte	Open source : baladeur, ordinateur, jeux vidéos
MP3	avec perte	Métadonnées : nombreuses applications
Vorbis	avec perte	Open source : équivalent libre du MP3

[20 Hz; 20 kHz] voire moins en téléphonie, il s'agit de tenir compte des propriétés de transcodage mécano-électrique au sein de la cochlée (voir § 2.2.2) et notamment du *phénomène de masquage fréquentiel* (voir infra), pour ne coder que ce qui est *a priori* perceptible par l'ouïe. On parle alors de « *codage perceptuel* », comme dans le cas du format MP3 et de son successeur libre et plus performant, le format Ogg Vorbis.

En effet, un signal sonore précédé ou suivi par un autre son d'intensité plus forte est en partie ou totalement inaudible. Ce mécanisme se qualifie de *masquage temporel*. Le niveau de masquage dépend bien entendu du niveau du son masquant mais encore de sa fréquence et de l'intervalle en fréquence qui le sépare du son masqué. On parle alors de *masquage fréquentiel*. Il est normal que le contenu spectral intervienne dans l'*effet de masque* puisqu'il dépend directement de la sensibilité de l'oreille humaine. On peut faire l'analogie entre cette inertie temporelle de la perception auditive et la persistance rétinienne qui permet au mouvement cinématographique d'exister.

La sélectivité en fréquence se réfère à la capacité du système auditif de détecter les composantes sinusoïdales d'un son complexe. Elle intervient dans de nombreux aspects de la perception auditive, y compris de l'intensité, de la hauteur et du timbre. L'étude de la sélectivité en fréquence va souvent de paire avec celle des effets de masquage fréquentiel, qui se définissent donc comme le processus par lequel le seuil d'audition d'un son est atteint par la présence d'un autre son. Ce phénomène est d'autant marqué que les contenus fréquentiels des signaux sont proches voire identiques.

Ainsi, dans le principe, la cochlée répond au concept de *filtre auditif*, centré sur la fréquence d'excitation f_c de chaque composante sinusoïdale du signal. Tout se passe comme si le son était détecté le long de la membrane basilaire par des cellules ciliées « spécialisées » pour chaque fréquence; d'où l'analogie avec un analyseur spectral. Pour un même contenu spectral, plus le niveau d'intensité sonore est élevé, plus le nombre de cellules ciliées impliquées augmente. Cela semble intuitivement logique mais dans quelles proportions et comment ?

En fait, on met en évidence que la membrane basilaire et les cellules ciliées qui lui sont rattachées se comportent comme un banc de filtres subdivisant l'étendue des fréquences audibles en filtres dont les bandes passantes se chevauchent et qui sont définis par leur fréquence centrale et leur largeur de bande $[f_c - \Delta f; f_c + \Delta f]$.

Cette modélisation fondée sur les filtres auditifs découlent du phénomène de masquage et la question est de savoir, à partir des profils de masquage mesurés (voir figure 4.26) et des filtres auditifs, quels sont les profils d'excitation (voir figure 4.27) des cellules ciliées. Comme interprétation physiologique, on peut assimiler les profils d'excitation aux enveloppes des vibrations de la membrane basilaire. Plus le son est fort, plus le nombre de cellules ciliées en jeu augmente, en faisant intervenir progressivement les cellules ciliées externes (cf. § 2.2.2).

Il est à remarquer que la forme des courbes présente une pente plus raide vers les basses fréquences et plus étendue vers les hautes fréquences d'autant que le niveau sonore est élevé. Aussi, un signal donné sera plus facilement masqué par un son à composantes spectrales inférieures et inversement. On parle parfois de pré-masquage et de post-masquage.

La figure 4.28 détaille le principe des effets de masque où les zones ombrées sont relatives aux parties inaudibles, donc inutiles à coder dans une perspective de diminution des bits de quantification.

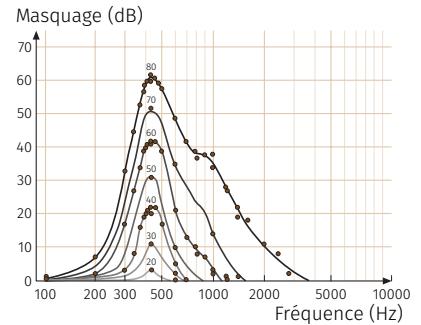


FIGURE 4.26 — Profil d'excitation calculé des cellules ciliées (sinusoïde à 1kHz de 20 à 90 dB SPL) — D'après [ROSSING, AKHATOV et al., 2007].

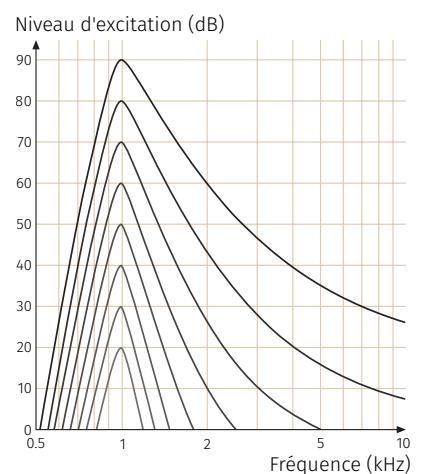


FIGURE 4.27 — Profil de masquage d'un bruit à bande étroite sur un son pur centré à 410 Hz — D'après [ROSSING, AKHATOV et al., 2007].

Au final, le codeur découpe le signal numérique en bandes — *frames* en anglais — de fréquence (filtre en peigne) et analyse le niveau de masquage pour chaque bande par le calcul de la courbe de seuil de masquage, à savoir l'enveloppe des effets de masque. Il en déduit le niveau de quantification de chaque bande, où dit autrement, le nombre d'échantillons non masqués qui doit être transmis et élimine les composantes masquées. Le résultat est ensuite traité par des techniques d'optimisation et de compression traditionnelles comme par exemple le codage de Huffman.

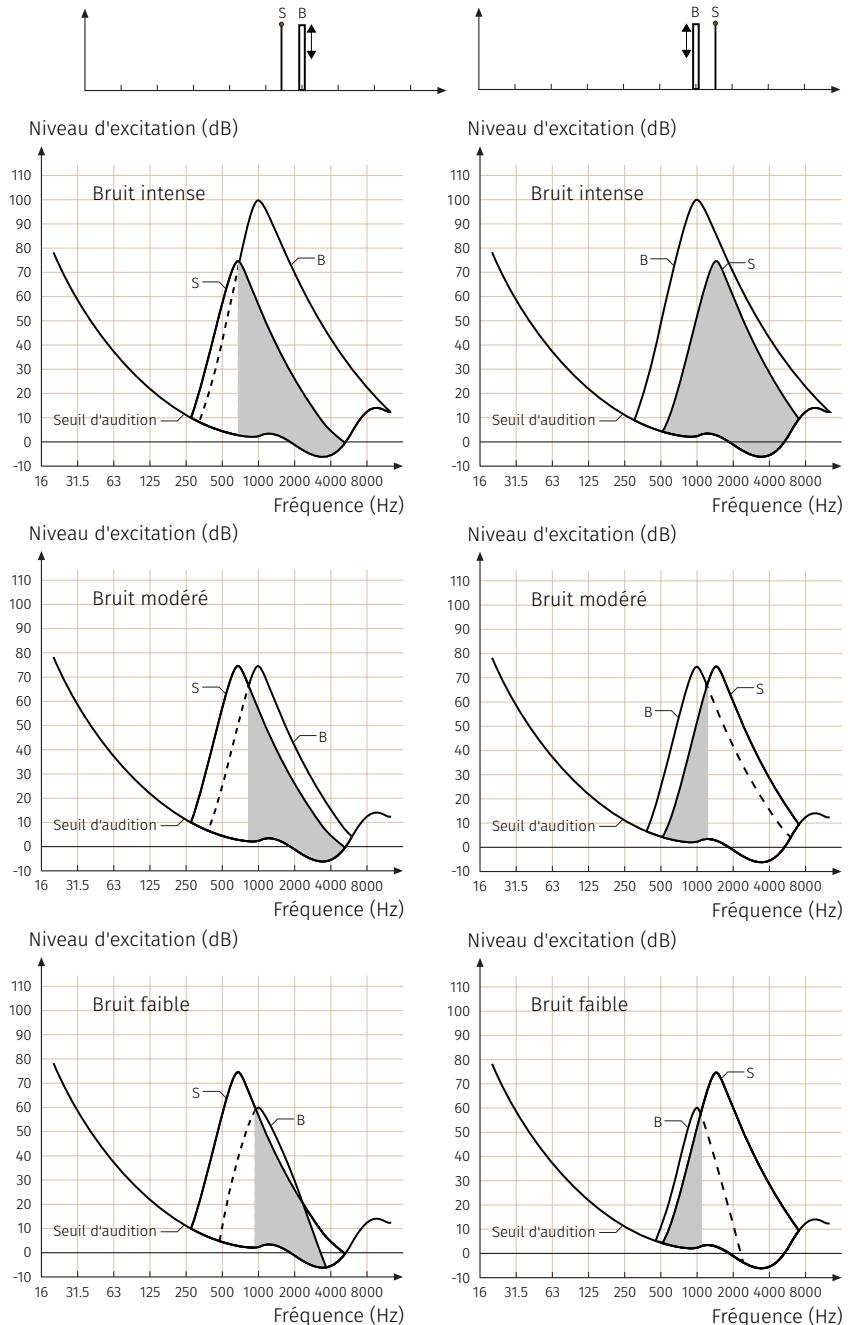


FIGURE 4.28 — Phénomène de masquage d'un son pur par un bruit à bande centré à 1 kHz. Interaction des profils d'excitation : la zone ombrée représente la partie éliminée due à la présence de bruit. D'après [BOTTE et al., 1989].

En pratique, les taux de compression peuvent aller jusqu'à 1/10 environ en fonction des exigences de l'application considérée. Au-delà, même une oreille « moyenne » perçoit très aisément la détérioration

du signal. Plutôt que de raisonner en taux de compression c'est en débit ou nombre de bits par seconde que l'on évalue les algorithmes, car hormis la taille d'un fichier, c'est la capacité de transmission des signaux pour la diffusion qui importe ; le flux d'information. On s'accorde à dire qu'un *bitrate* — ou *débit binaire*, nombre de bits autorisés en une seconde — de l'ordre de 128 kbps est un bon compromis pour un baladeur, mais s'avère limite voire insuffisant pour une chaîne Hi-Fi.

RESSOURCES COMPLÉMENTAIRES

Formation complémentaire

- ▶ Partie 2 du #2 Module thématique : manipuler l'information, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme ➔

Articles

- ▶ Compression vidéo, WIKIPÉDIA.

Vidéos

- ▶ Pourquoi la neige ou les confettis baissent la qualité des vidéos ?, Tom SCOTT, YouTube (4mn 20s), 2016;
- ▶ Les marmottes au sommeil léger (présentation ludique du codage de HUFFMANN de compression de données, sous forme d'activité), Marie DUFLOT-KREMER, YouTube (17mn 39s), 2017.

3.2 Organisation et stockage des données

Après avoir détaillé le codage des données selon leur origine, pour qu'elles soient exploitable par l'ordinateur, il est nécessaire de déterminer comment les stocker afin de faciliter leur accès ou leur manipulation, à savoir pouvoir ensuite correctement les décoder.

3.2.1 Type et codage

Peu importe leur contenu, les données sont stockées en mémoire, sur disque ou transitent au travers du réseau. En revanche, il faut précisément savoir comment elles sont codées et organisées pour que l'ordinateur puisse les lire, les générer ou bien les deux.

En pratique, un fichier de données comporte un entête qui indique leur taille, leur durée et leurs couleurs éventuelles, leur type de compression, etc. Cet entête spécifie donc le codage des données pour ensuite les exploiter en fonction des informations que l'entête a décrit.

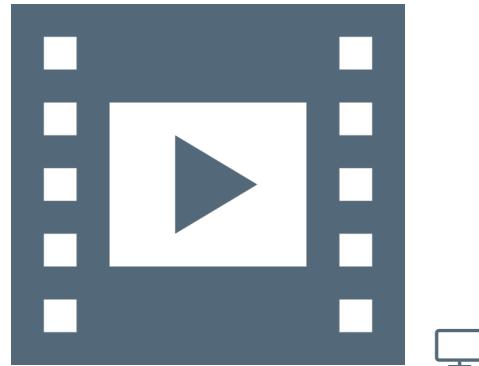
Il existe un certain nombre de données élémentaires normalisées :

- des entiers plus ou moins grands codés sur 8 bits — 1 octet, $2^8 = 256$ valeurs —, 16, 32 ou 64 bits — $2^{64} = 1,844\,674\,407 \times 10^{19}$ valeurs — et éventuellement négatifs;
- des nombres flottants qui s'exprime comme un nombre à virgule suivi d'une puissance de 10 ($1,456\,289\,63 \times 10^{23}$). Leur valeur va pouvoir être bien plus importante que les entiers, mais limitée par leur précision; une dizaine ou une vingtaine de chiffre significatifs après la virgule (cf. norme IEEE-754).

Le programmeur choisit le type de données en fonction de ce que celles-ci représentent : âge (entier positif sur un octet), mois (entier positif sur un octet), température (flottant avec peu de précision), niveau de gris (entier sur 8 bits), etc.

3.2.2 Structure de données

Parfois, il est nécessaire de faire appel à des données plus complexes que des nombres. C'est par exemple le cas d'une date qui se



VIDÉO 4.4 — Organisation des données.

compose d'un jour, d'un mois et d'une année ou bien d'une couleur qui se décrit par trois entiers, chacun associé à une composante rouge, verte et bleue. Il s'agit donc de stocker des combinaisons de nombres et non plus de simples chiffres.

En informatique, des données assemblées d'autres types élémentaires sont nommées des « *tuples* ». On trouve également les termes de *structure* ou d'*enregistrement*. Une date est ainsi composée de trois entiers (jour, mois, année), de même pour une image décrite par des pixels relatifs à trois entiers de composantes de couleur. Par extension, on peut établir des « *tuples de tuples* », comme par exemple une époque constituée de deux dates, une de début et l'autre de fin.

Parfois on ne sait pas à l'avance combien de données sont à manipuler : quels invités à une fête ou quelles étapes lors d'un voyage ? On utilise alors d'autres types de données comme les *tableaux*. Un tableau est un ensemble de tuples de structure similaire agencés ligne par ligne. En référence à la [table 4.6](#), chaque ligne ne peut comporter, dans l'ordre : qu'un numéro, qu'un nom, qu'une date de naissance et qu'une taille.

Un autre type de données utile à la gestion des tuples est une *liste*. Une liste est constituée d'un tuple de départ repéré par son adresse en mémoire et qui pointe vers son successeur, lui-même étant un tuple similaire en structure qui pointe vers le suivant et ainsi de suite (cf. [figure 4.29](#)). Cela ressemble à un tableau comme étant une succession d'éléments du même genre mais, en pratique, cela modifie l'efficacité au niveau algorithmique. Suivant les circonstances, une liste ou un tableau sera plus efficient.

Par exemple, l'insertion d'un nouvel élément dans un tableau est plus compliquée que dans une liste car il faut décaler toutes les lignes vers le bas, c'est-à-dire décaler en mémoire toutes les données stockées préalablement. Pour une liste, il suffit de définir un nouvel élément en mémoire et d'affecter son adresse comme étant celle du début ou de l'indication voulu de la liste (voir [figure 4.29](#)).

3.2.3 Représentation des données en mémoire

Comment sont réparties les données en mémoire ? Dans quel ordre et avec quelle taille ? L'organisation des différentes données doit être clairement spécifiée par le programmeur.

En prenant exemple sur la [table 4.6](#), cela signifie qu'il faut spécifier quelque part que les lignes du tableau seront ordonnancées d'abord avec le numéro d'occurrence puis le nom, la date de naissance et enfin la taille. Ceci peut-être établi différemment, mais doit être défini clairement en début de programme pour que le déroulé du programme puisse se référer correctement à ces données. On ne peut pas modifier la spécification au milieu du programme.

Pour l'enregistrement de données dans un fichier, la même problématique se pose. Il faut qu'un programme génère le fichier en respectant la spécification du format des données pour que d'autres programmes puissent la comprendre. C'est ce qu'on appelle un format ouvert, comme dans le cas d'une image GIF. Pour le décoder, on sait ainsi à l'avance que l'entête du fichier comporte une chaîne de caractères « *GIF89a* », puis 16 bits qui stockent la largeur, 16 bits pour la hauteur de l'image et ainsi de suite. N'importe quel programme ou personne voulant lire un fichier GIF saura que les données sont organisées de cette manière ; idem pour un enregistrement. Si le format de fichier est fermé, certains programmes pourront générer les fichiers, mais d'autres ne pourront pas les lire.

FIGURE 4.6 — Type de données « tableau ».

Type de données tableau			
N°	NOM	NAISSANCE	TAILLE
1	Joe	21/02/1869	1,6 m
2	Jack	15/06/1868	1,7 m
3	William	02/03/1867	1,8 m
4	Averell	09/11/1866	1,9 m

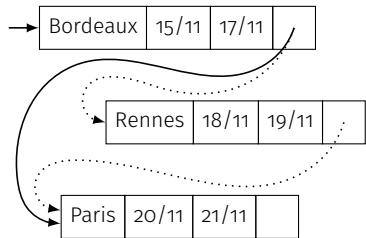


FIGURE 4.29 — Type de données « liste ». En pointillé : insertion d'un élément.

RESSOURCES COMPLÉMENTAIRES

Formation complémentaire

- ▶ Partie 3 du #2 Module thématique : manipuler l'information, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme ➤

Articles

- ▶ [Les données en question](#), Stéphane GRUMBACH et Patrick VALDURIEZ, INTERSTICES, 31 mars 2016;
- ▶ [Structures de données](#), WIKIPÉDIA.

Vidéos

- ▶ [Les bases de données à tricoter](#), (présentation ludique sous forme d'activité des bases de données), Marie DUFLOT-KREMER, YouTube collection SCIENCEPARTICIPATIVE), (18mn 37s), 2017.

3.3 Base de données

Le quotidien de tout un chacun amène à gérer des comptes bancaires, des stocks, des clients, des ventes ou répertorier des références bibliographiques, des sites Web, etc. L'objectif des bases de données est de permettre la manipulation de ces quantités énormes d'informations de manière compréhensible et suffisamment simple pour un humain et non telles qu'enregistrées par l'ordinateur.

Les logiciels dévolus à cette tâche s'appellent des *systèmes de gestion de base de données* (SGBD). Ce sont des médiateurs entre l'humain et les données. Les SGBD les plus connus sont ORACLE pour les logiciels propriétaires et POSTGRESQL, MYSQL ou son successeur MARIADB pour les solutions libres.

3.3.1 Modèle relationnel et requête SQL

Le modèle de base de données le plus répandu actuellement est le *modèle relationnel*. Comme son nom l'indique, le modèle relationnel se fonde sur des relations entre les données exprimées sous forme de tableaux comportant les données à manipuler.

Les tableaux 4.7 fournissent un exemple de schéma relationnel pour des séances de cinéma. Un premier tableau répertorie les films et un second les séances. Une autre possibilité aurait été de les fusionner pour présenter toutes les informations en même temps. Cependant, il y aurait beaucoup d'informations dupliquées et redondantes comme les réalisateurs et les acteurs pour chaque séance d'un même film.

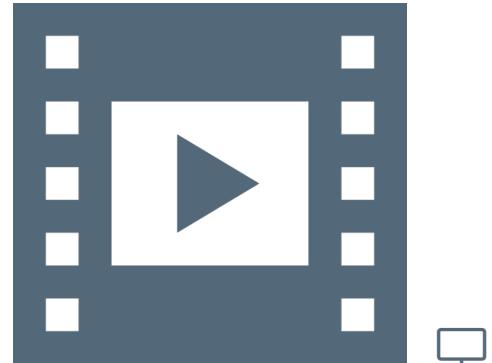
TABLE 4.7 — Modèle relationnel de séances de cinéma.

(a) *Films*.

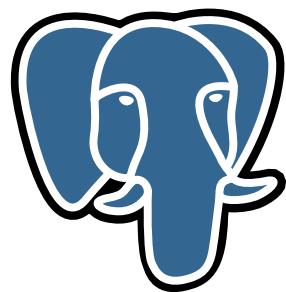
TITRE	RÉALISATEUR	ACTEUR
Imitation game	Morten Tyldum	Benedict Cumberbatch
Snowden	Oliver Stone	Joseph Gordon-Levitt

(b) *Séances*.

TITRE	CINÉMA	HEURE
Imitation game	CGR Le Français	19h45
Snowden	Les Tourelles	20h00
Imitation game	Les Tourelles	22h00



VIDÉO 4.5 — Base de données.



L'enjeu d'un système de base de données est de pouvoir formuler des requêtes comme par exemple : où peut-on voir un film avec un acteur particulier tel que Benedict CUMBERBATCH ? Un langage spécifique aux bases de données relationnelles, le SQL pour *Structured Query Language* — littéralement Langage d'interrogation structurée —, a été élaboré pour qu'un humain puisse construire puis lancer des requêtes facilement.

La requête précédente s'écrit comme suit en SQL :

```
SELECT Cinema FROM Films, Seances
WHERE Films.Titre = Seances.Titre
AND Acteur = "Benedict Cumberbatch"
```

En langage naturel, cela peut se traduire par : « Sélectionnes le(s) cinéma(s) à partir des tables donnant les films et les séances où les lignes pour lesquelles le titre du film est le même que celui de la séance et dont l'acteur correspond à Benedict CUMBERBATCH ».

3.3.2 Protection et partage des données

Un des autres grands intérêts des bases de données est de protéger les données. De nos jours les données se monnayent chèrement et sont souvent inestimables comme les comptes d'une entreprise ou, moins sérieusement mais affectivement, les photos de vacances.

Les systèmes informatiques ne sont pas infaillibles et sont sujets à des pannes matérielles, des bogues logiciels, des attaques, etc. Les bases de données mettent en œuvre des parades pour éviter ces problématiques par exemple par la réPLICATION des données, c'est-à-dire stocker les données à plusieurs endroits. Si un serveur tombe en panne, cela ne change rien pour l'utilisateur qui en toute transparence aura accès à ses données sur un autre serveur.

Les bases de données sont également utiles au partage des données. Très souvent, les données sont manipulées par plusieurs utilisateurs et en même temps. On peut prendre en exemple des données musicales en partage où Alice et Bob font leur choix de programmation. Imaginons les étapes suivantes :

- Alice lit un morceau de Leonard Cohen;
- Bob le lit à son tour;
- Alice ajoute un « *like* » à ce morceau;
- Bob fait de même un peu plus tard.

In fine, le constat sera que le « *like* » d'Alice a disparu et que seul celui de Bob est resté. Ce faisant, quand Bob a ouvert son morceau, il n'y avait pas encore le *like* d'Alice et quand il a terminé son écoute, son *like* a écrasé celui qui était présent.

Pour éviter de tels problèmes dits d'accès concurrents, les systèmes de gestion de base de données définissent des *transactions* qui permettent de partager les données et de faire des modifications sans conflit entre opérations de différents utilisateurs.

RESSOURCES COMPLÉMENTAIRES

Vidéos

- ▶ Sciences des données : de la logique du premier ordre à la Toile, Serge ABITEBOUL, Leçon inaugurale prononcée le jeudi 8 mars 2012 au Collège de France (57mn 57s), texte intégral;
- ▶ La séquence CLASS'CODE « Manipulez l'information » par Marie DUFLOT-KREMER avec juste la vidéo sur les bases de données et la séquence entière sur OPENCLASSROOMS ;

- ▶ Des activités débranchées, Marie DUFLOT-KREMER sur la manipulation des données;
- ▶ Bases de données relationnelles, CANALU.

Supports de cours

- ▶ Cours de bases de données - Modèles et langages, Philippe RIGAUX, 2002-2015;
- ▶ Cours de bases de données - Aspects système, Philippe RIGAUX, 2002-2014.

Mooc

- ▶ Bases de données relationnelles : comprendre pour maîtriser, Mooc rejoué régulièrement sur la plateforme Fun.

4 Que faire de ces ressources ? Autoévaluation

Le questionnaire à choix multiple* — QCM — à suivre clôture le présent chapitre « Codage binaire » et correspond à chaque sujet abordé.

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

QUIZ 4 — PRÉSENTATION DE L'INFORMATION 3 POINTS

QUIZ 4.1 — NOTION DE BIT 1 POINT

Pourquoi les informations sont-elles stockées avec des bits prenant seulement deux valeurs?

- Car la tension électrique ne peut prendre que 2 valeurs.
- Car c'est pratique à implémenter dans les circuits électroniques.
- Car les informations s'expriment naturellement sous cette forme.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implantée dans l'extension de style `LATEX` utilisée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

QUIZ 4.2 — ADDITION DE NOMBRES 1 POINT

Quand un circuit électronique additionne deux nombres...

- Il additionne bêtement sans réfléchir.
- Il sait à quelle grandeur et unité correspondent ces nombres.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 4.3 — CODAGE ASCII 1 POINT

Quels sont les avantages du codage ASCII des lettres?

- C'est facile à lire pour un humain.
- C'est facile à stocker et manipuler dans l'ordinateur.
- Cela code toutes les lettres de l'alphabet latin et la ponctuation
- Cela permet de représenter l'ensemble des caractères de toutes les langues du monde.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 5 — INFOGRAPHIE ET AUDIONUMÉRIQUE
3 POINTS

QUIZ 5.1 — COULEURS ET PIXELS 1 POINT

Comment se composent généralement les pixels couleurs ?

Trois composantes rouge, jaune et bleu.

Trois composantes rouge, vert et bleu.

Du nom standard de la couleur, en anglais.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 5.2 — CODAGE D'IMAGES 1 POINT

Comment peut-on coder une image carrée contenant un dégradé de couleur de rouge à gauche jusqu'à jaune à droite ?

100 lignes de 100 pixels, avec des couleurs plutôt rouge à gauche, jaune à droite, orange au milieu, etc.

Une formule mathématique donnant les dimensions du carré et décrivant le rouge à gauche et le jaune à droite, et la façon de calculer les nuances intermédiaires.

La meilleure solution dépend de ce qu'on veut faire de l'image.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 5.3 — DÉCODAGE D'UN SON 1 POINT

Quand l'ordinateur décide un son :

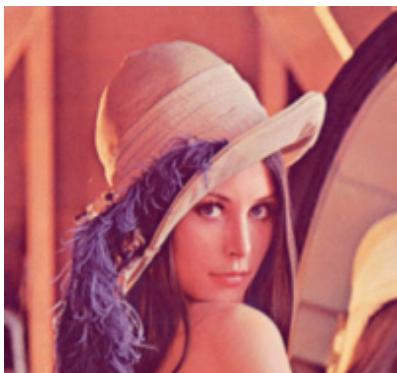
il sait que c'est un son.

il sait que c'est tel morceau de tel artiste.

il n'en sait rien, il applique juste bêtement un programme de décodage sans comprendre ce qui se passe.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 6 — COMPRESSION DE DONNÉES
3 POINTS



QUIZ 6.1 — TAILLE D'IMAGE 1 POINT

Quel espace occuperait une photo prise par votre smartphone avec un capteur à 10 millions de pixels en couleur si on ne la compressait pas ? Chaque couleur est codée sur 8 bits.

3 mégaoctets.

30 mégaoctets.

300 mégaoctets.

3 gigaoctets.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 6.2 — COMPRESSION D'IMAGE 1 POINT

Si on compresse* une image, quelles assertions se vérifient ?

*En exportant sous GIMP une photo au format JPEG. Un curseur permet de régler le taux de compression et ce faisant, le poids de l'image obtenue en cochant la case sous le curseur (prévisualisation).

- Elle peut ne peser qu'environ 1 kilo-octet après compression.
- Elle peut être de n'importe quelle taille inférieure à celle d'origine.
- Elle garde une bonne qualité quelle que soit la compression.
- Elle peut devenir moche.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 6.3 — DÉCOMPRESSION DE VIDÉO 1 POINT**

Comment l'ordinateur sait-il comment décompresser une vidéo ?

- Il lui suffit de regarder l'extension du fichier (.mov, .avi, .mpg, etc.), elle indique forcément le type de compression du fichier.
- Il regarde l'entête du fichier pour comprendre le type de compression et les caractéristiques de la vidéo.
- En fait, les vidéos compressées sont tout simplement toujours illisibles, c'est fait exprès pour les chiffrer.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 7 — ORGANISATION DES DONNÉES****3 POINTS****QUIZ 7.1 — CODAGE DE LA TAILLE D'UNE PERSONNE 1 POINT**

Comment choisir entre un entier 8 bits et un flottant 32 bits pour coder la taille d'une personne ?

- On regarde les valeurs possibles d'une taille humaine et la précision dont on a besoin.
- Qu'importe, l'ordinateur ne sait pas à quoi correspond ce nombre.
- On regarde l'espace nécessaire pour stocker ces types afin d'économiser de la mémoire.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 7.2 — FORMAT DES DONNÉES 1 POINT**

Comment est détecté le fait qu'une donnée soit un entier ou un tuple ?

- En mémoire, c'est le programmeur qui l'a spécifié clairement dans le programme.
- Peu importe, le programmeur n'a pas besoin de le savoir.
- Dans un fichier, l'entête des données, si elle existe, permet de détecter les données.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 7.3 — DÉCODAGE D'UN FICHIER IMAGE JPEG 1 POINT

Comment décoder une photo enregistrée dans un fichier image JPEG pour l'afficher? Un logiciel (spécialisé) ouvre le fichier image qui est au format JPEG et...

- pas besoin de décoder, il suffit de recopier la valeur des pixels pour les afficher à l'écran.
- lit le code informatique stocké dans l'entête indiquant comment décoder un fichier JPEG, puis l'exécute pour décoder les données et les afficher sur l'écran.
- a été programmé à partir de la spécification du format d'image JPEG et utilise la partie de son code correspondant pour décoder et afficher à l'écran.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 8 — BASES DE DONNÉES****1 POINT****QUIZ 8.1 — SYSTÈME DE GESTION MULTI-SERVEUR 1 POINT**

Pourquoi un système de gestion de base de données multi-serveur, avec réPLICATION DES DONNÉES, simplifie-t-il la consultation des données?

- On peut consulter dans un langage relativement facile à comprendre pour un humain.
- On peut consulter sans avoir besoin de savoir dans quel ordre les données sont stockées.
- On peut consulter même si un des serveurs tombe en panne.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

LANGAGES DE L'INFORMATIQUE

LA PROGRAMMATION EST UNE DES ACTIVITÉS LES PLUS VISIBLES de l'informatique. En amont, avant de se lancer tête baissée dans l'apprentissage d'un langage de programmation et le codage, il est souvent utile de détailler le mode d'emploi ou la recette permettant d'aboutir au résultat voulu indépendamment de toute programmation effective : cette étape est couverte par ce qu'on appelle l'**algorithmique**.

Ensuite, le terrain étant balisé, il devient plus aisé* de s'atteler à la programmation en tant que telle à l'aide d'un langage choisi en fonction de multiples critères contextuels et pragmatiques. Ce chapitre aborde ces deux facettes du traitement de l'information et de l'expression entre humain et machine.

**De nos jours, on utilise également des outils de modélisation comme UML — Unified Modeling Language — et autres processus de codage comme les méthodes dites agiles qui facilitent la conception et le développement des programmes informatiques, surtout pour les applications volumineuses impliquant de multiples contributeurs.*

1 Algorithmique

L'algorithmique est la science des algorithmes, mot dérivé du nom du mathématicien Muhammad AL-KHARIZMI ayant vécu au IX^e siècle.

Aujourd'hui, il s'agit d'établir, au moyen d'un langage quasiment naturel et de concepts élémentaires, une suite d'instructions que la machine est à même de « comprendre » et d'exécuter.

1.1 Notions fondamentales

Un algorithme répond au double objectif de conduire à un résultat, si possible correct, en un nombre fini d'opérations. Par définition, il prend un ensemble de données en entrée, exprime un traitement spécifique et fournit des données en sortie.

1.1.1 Décomposition en tâches

À l'instar des champs disciplinaires de la physique, il est souvent impossible de résoudre directement des problèmes complexes. La démarche est alors de décomposer la problématique en éléments simples auxquels on sait répondre et d'apporter petit-à-petit les briques man-

SOMMAIRE

1 Algorithmique

- 1.1 Notions fondamentales 163
 - 1.1.1 Décomposition en tâches ■
 - 1.1.2 Variables ■ 1.1.3 Instructions élémentaires

- 1.2 Culture algorithmique 167
 - 1.2.1 Introduction aux graphes ■
 - 1.2.2 Parcours de graphe

2 Programmation

- 2.1 Langages de programmation 171
 - 2.1.1 Langage machine ■ 2.1.2 Langage de haut niveau et compilation
- 2.2 Paradigmes de programmation 173
 - 2.2.1 Bogues ■ 2.2.2 Humains et langages

3 Que faire de ces ressources ? Quiz



VIDÉO 5.1 — Décomposition en tâches.



quantes menant à la solution. Cette approche est identique en algorithmique, où l'obtention d'une solution ne peut s'envisager que par étapes successives du fait que l'ordinateur ne sait manipuler qu'un nombre limité de concepts et d'instructions.

Pour expliciter le propos, on considère un algorithme conduisant à savoir si des élèves ont obtenu la moyenne à une épreuve corrigée. Autrement dit, étant donné un tableau de notes potentiellement grand, déterminer combien d'élèves ont la moyenne.

8	12	20	16,5	11	8,5	3	16	10,5
8	12	20	16,5	11	8,5	3	16	10,5

Pour le tableau ci dessus, c'est assez facile car on peut compter les notes au dessus de la moyenne « à la main », mais pour d'autres cas de figure avec plusieurs centaines de notes, l'ordinateur est un secours.

La question est alors de savoir ce que sait faire un ordinateur. Il peut réaliser des tâches simples et unitaires comme :

- faire un calcul;
- mettre le résultat du calcul dans une variable;
- exécuter séquentiellement (l'une après l'autre) des tâches;
- conduire un test pour choisir quelle tâche exécuter ensuite;
- faire une boucle (tant que quelque chose ne se passe pas)...

Reconsidérons le tableau de notes précédent et supposons qu'il ait un million de cases. Il est indubitablement nécessaire de réagir autrement qu'en comptant à la main.

En isolant une case, on la regarde et si son contenu est plus grand que 10 on renvoie 1, sinon 0. Si le tableau contient n cases avec $n > 1$, supposons que l'on sait que sur les $n - 1$ cases premières cases, on a s élèves qui ont la moyenne alors, en observant la n -ième case, si son contenu est plus grand que 10, il y a $s+1$ élèves ayant la moyenne, sinon il y en a s. Dès lors, on peut transcrire la problématique en algorithme exprimé en pseudocode.

```

ALGORITHME 5.1 — Notes supérieures à 10
1 Variable n entier > 0
2 Tableau t taille n (valeurs de t[0] à t[n-1])
3 Variable s entier > 0
4
5 s ← 0
6 Pour i allant_de 0 à n-1, Faire
7   Si (t[i] ≥ 10)
8     Alors s ← s + 1
9   FinSi
10 FinPour

```

POUR ALLER PLUS LOIN... ▶

Formation complémentaire

- ▶ Partie 1.1 et 1.4 du #1 Module fondamental : découvrir la programmation créative, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme ▶ — NB : SCRATCH n'est pas au programme SNT, mais les élèves ont maintenant été formés à SCRATCH en collège ou primaire; cette ressource permet de se mettre à niveau par rapport à cette compétence.

Articles

- ▶ Le langage assembleur, Wikipedia;



- ▶ [Les ingrédients des algorithmes](#), Gilles DOWEK, Thierry VIÉVILLE, Jean-Pierre ARCHAMBAULT, Emmanuel BACCELLI et Benjamin WACK, INTERSTICES, 21 avril 2010.

Vidéo

- ▶ [Class'Code Didacticiel - D-Clics numérique](#), initiation à la notion d'algorithme à travers la métaphore culinaire usuelle par Thierry Viéville, YouTube, (8mn 35s), 2016.

1.1.2 Variables

Qu'est-ce qu'une variable ? C'est un espace mémoire soit locale, soit distante, où le programme stocke une valeur ou une structure de donnée compliquée. Une variable est censée changer au cours de l'exécution du programme, sinon on parle plutôt de constante.

On peut citer quelques exemples de variables comme :

- le nombre de *followers*, de vues d'une vidéo sur YOUTUBE, de *like*, etc. (un entier);
- la moyenne d'un élève (un réel, nombre à virgule flottante);
- toutes les notes d'une classe (un tableau, un tableau de tableaux, une base de données);
- mais encore d'autres données plus complexes, à savoir une table de hachage, une liste doublement chaînée, etc.

Différentes manipulations sur les variables sont possibles :

- elle doit être créée (cela dépend en fait du langage de programmation, certains déclarant la variable lors de leur utilisation);
- elle peut être affectée d'une valeur particulière ($i \leftarrow 0$);
- elle peut être modifiée d'une valeur particulière ($i \leftarrow i+1$);
- elle peut être l'entrée du programme.

Les variables sont souvent typées, c'est-à-dire qu'en les déclarant on leur attribue une catégorie connue. On distingue à cet effet :

- | | |
|------------------------------|-------------------------------|
| – les entiers; | – les tableaux; |
| – les chaînes de caractères; | – les pointeurs; |
| – les nombres flottants; | – les structures composées... |

POUR ALLER PLUS LOIN...

Formation complémentaire

- ▶ Partie 3 du #1 Module fondamental : découvrir la programmation créative, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme .

Articles

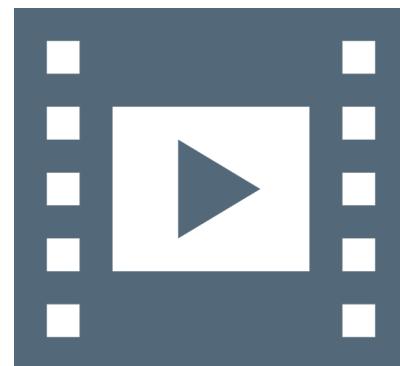
- ▶ [Les variables](#), Wikipedia;
- ▶ [Les pointeurs](#), Wikipedia;
- ▶ [Naissance des langages de programmation](#), Sacha KRAKOWIAK et Jacques MOSSIÈRE, INTERSTICES, 24 janvier 2012.

Vidéo

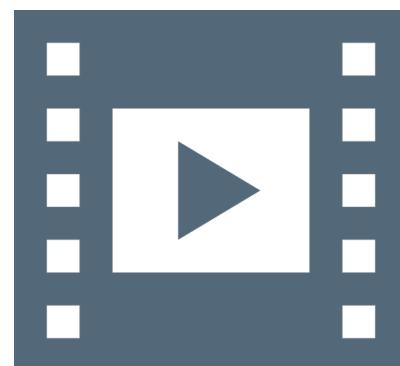
- ▶ [La notion de variable selon Gilles Dowek](#), (présentation très large public de la notion de variable, en 50 secondes par Thierry VIÉVILLE), YouTube.

1.1.3 Instructions élémentaires

Le cœur de l'algorithme est dérivé des instructions élémentaires qu'il est possible de réaliser avec un ordinateur. Comme pour l'ADN des cellules du monde vivant, elles sont au nombre de quatre à partir desquelles on peut élaborer *n'importe quel algorithme*.



VIDÉO 5.2 – Variables.



VIDÉO 5.3 – Instructions élémentaires.



La première instruction est l'affectation déjà évoquée précédemment; à une variable on associe une valeur donnée : $i \leftarrow 17$, $i \leftarrow i+1$.

La deuxième instruction est le test :

Si <booléen> **Alors** <instruction> **Sinon** <instruction>.

Si une valeur booléenne est vraie — une expression booléenne retourne soit vrai, soit faux —, alors on exécute une instruction, sinon on exécute une autre instruction.

Quelques exemples de tests sont donnée par :

- une valeur est-elle supérieure à 10 ?
- un calcul risque-t-il de dépasser les valeurs maximales/minimales du type de la variable ?
- est-il possible de déborder des limites licites d'un tableau ?

L'instruction suivante est la séquence. L'intérêt de l'informatique est d'exécuter un certain nombre d'opération les unes après les autres

<instruction> ; <instruction>.

On peut en fournir quelques exemples :

- affectation puis test;
- affectation, puis boucle, puis test;
- affectation, puis affectation, puis affectation, puis affectation.

Enfin, la dernière instruction est la boucle. On en distingue deux formulation : la boucle « **Pour** » et la boucle « **Tant que** ».

Pour variable de valeur à valeur **Faire** <instruction> **FinPour**.

La boucle « **Pour** » sert par exemple à parcourir un tableau pour en extraire des données ou mener des calculs : chercher un maximum, calculer une moyenne, remplir un tableau, etc.

TantQue booléen **Faire** <instruction> **FinTantQue**.

La boucle « **TantQue** » est plus générale et permet de réaliser plus de choses que la boucle « **Pour** », mais nécessite plus d'attention notamment au niveau de son instruction. Il faut en effet être sûr que le programme s'arrête et qu'une boucle infinie ne s'installe. On parle de problème de terminaison.

Une des utilisations traditionnelles de la boucle « **TantQue** » est une suite convergente qui va calculer par itération successives les valeurs, jusqu'à ce quelles soient suffisamment proches.

Soit comme exemple applicatif, un algorithme de recherche par dichotomie d'une valeur dans un tableau trié. En sortie, on obtient un booléen qui renvoie si la valeur est dans le tableau ou non.

ALGORITHME 5.2 — Recherche dichotomique

```

1 Variable n entier > 0
2 Tableau t taille n // tableau trié
3 Variable v, sup, inf entier
4
5 inf ← 0;
6 sup ← n-1;
7 TantQue (inf ≤ sup) Faire
8   i ← (inf+sup)/2;
9   Si (t[i] = v) Alors
10    Renvoyer Vrai;
11   Sinon
12    Si (t[i] > v) Alors
13      sup ← i-1;
14    Sinon
15      inf ← i+1;
16    FinSi
17  FinSi
```

¹⁸ **FinTantQue**
¹⁹ **Renvoyer** Faux

Le principe de la recherche dichotomique est de tester successivement la moitié du tableau, le quart, le huitième, ect. Cela ne peut fonctionner que si le tableau est trié, puisqu'on teste par relation d'ordre si la valeur cherchée peut potentiellement être dans une moitié successive du tableau, sinon dans l'autre. L'idée est alors de modifier **inf** et **sup** pour ne conserver que la partie du tableau où la valeur **v** est susceptible de se trouver. L'algorithme se termine lorsque l'on a trouvé la valeur dans le tableau ou, si elle n'y est pas lorsque les valeur **inf** et **sup** sont permutées.

Considérons une illustration simple où on cherche la valeur 5 dans le tableau qui suit. le déroulé de l'algorithme se présente ainsi :

-5	-1	2	3	4	4	6	16	25
↑			↑		↑			↑
inf			i					sup

-5	-1	2	3	4	4	6	16	25
↑			↑		↑			↑
inf			i					sup

-5	-1	2	3	4	4	6	16	25
↑								
inf,i,sup								

-5	-1	2	3	4	4	6	16	25
↑			↑		↑			
sup			inf					

Les valeurs de **inf** et **sup** étant inversées, la boucle s'arrête et on en conclut que la valeur 5 n'est pas dans le tableau.

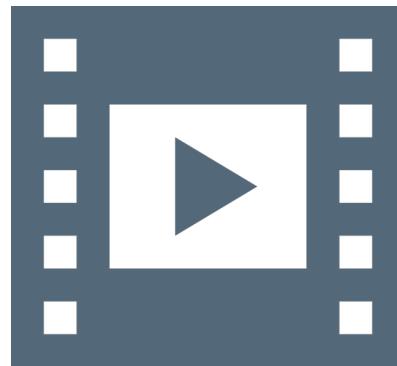
POUR ALLER PLUS LOIN... _____

Formation complémentaire

- Partie 2 du #1 Module fondamental : découvrir la programmation créative, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme .

Articles

- Qu'est-ce qu'un algorithme ?, Philippe FLAJOLET et Étienne PARIZOT, INTERSTICES, 24 février 2004;
- Algorithmes, mode d'emploi, Thierry VIÉVILLE, INTERSTICES, 16 janvier 2009;
- Genèse d'un algorithme, François RECHENMANN et Marie-Christine ROUSSET, INTERSTICES, 22 février 2011;
- Les ingrédients des algorithmes, Gilles DOWEK, Thierry VIÉVILLE, Jean-Pierre, ARCHAMBAULT , Emmanuel BACCELLI et Benjamin WACK, INTERSTICES, 21 avril 2010.



VIDÉO 5.4 – Algorithmique.

1.2 Culture algorithmique

Il existe de nombreuses structures de données différentes parfois complexes, parmi lesquelles se trouvent : les listes, les listes doublement chaînées, les piles et les files, les table de hachage, les arbres et les tas, les graphes (orientés ou non), les bases de données, etc.

Bien entendu, il ne s'agit pas ici de toutes les détailler — certaines ont été abordée précédemment —, mais de s'attacher à l'une d'entre elles, les graphes, afin d'illustrer le propos par un algorithme de parcours d'un graphe.

1.2.1 Introduction aux graphes

¹. Les graphes sont également présentés au chapitre 2, « Web et usages sociaux ».

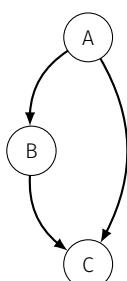


FIGURE 5.1 — Graphe orienté.

La question est alors de savoir comment représenter un graphe par une structure de données. Une première manière est d'utiliser un graphe dit d'adjacence qui, au moyen d'un booléen va dire si deux nœuds sont en relation ou non. Une autre façon de faire est de passer par une liste qui va répertorier les nœuds voisins.

1.2.2 Parcours de graphe

La figure 5.2 illustre le graphe de la présente partie « Fondements de l'informatique ». Les différents liens (les arêtes fléchées) indiquent les dépendances entre les séquences.

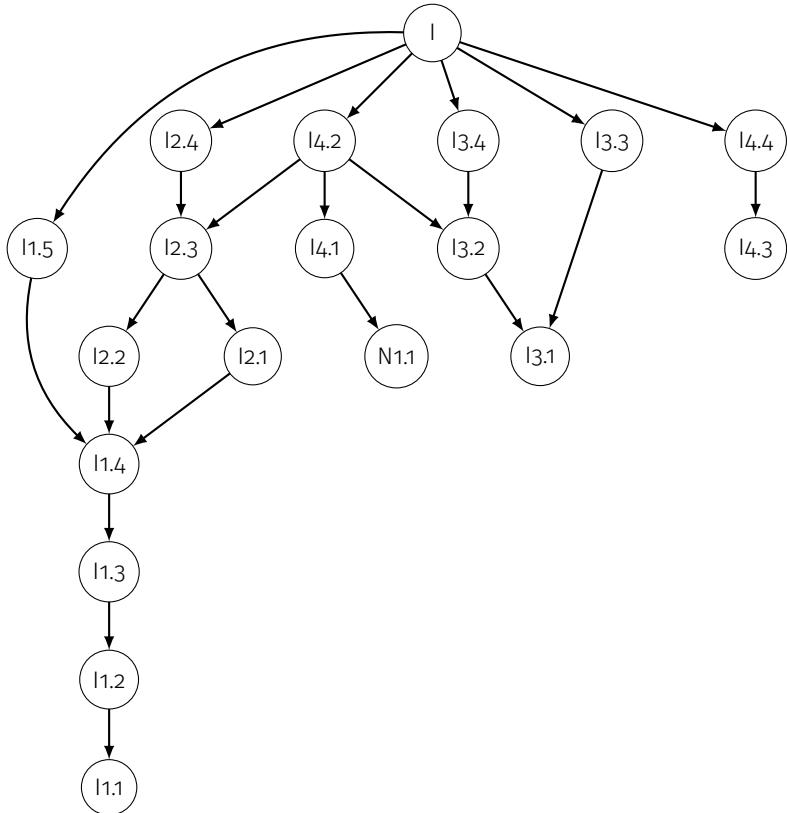


FIGURE 5.2 — Graphe des séquences de la partie « Fondements de l'informatique ».

Imaginons vouloir suivre une séquence donnée. Pour ce faire, on veut savoir quelles sont les séquences à suivre au préalable. On peut alors dérouler l'algorithme à suivre.

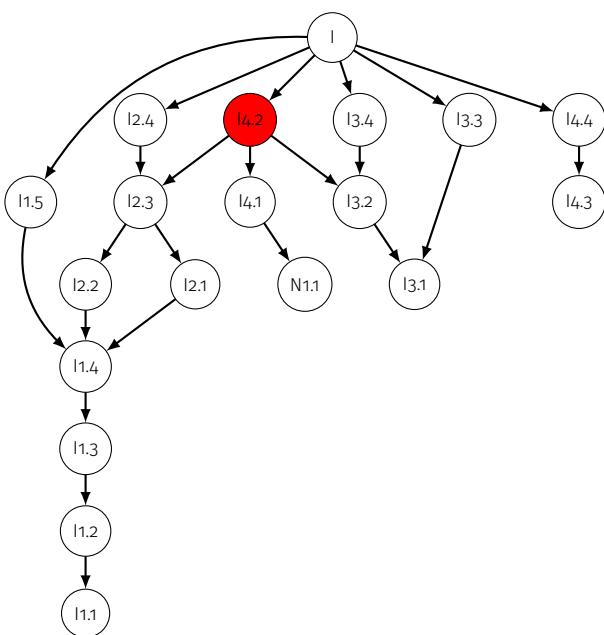
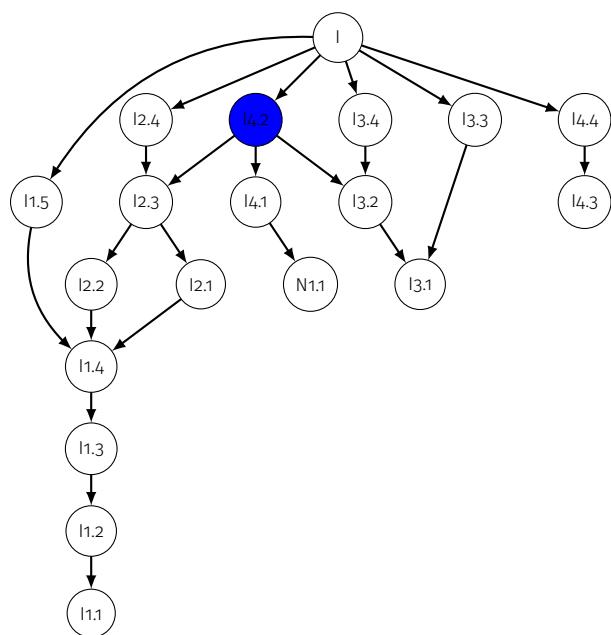
	ALGORITHME 5.3 — Parcours de graphe
¹ // Entrée : - noeud initial <i>n</i>	
² // - graphe de dépendance <i>g</i>	

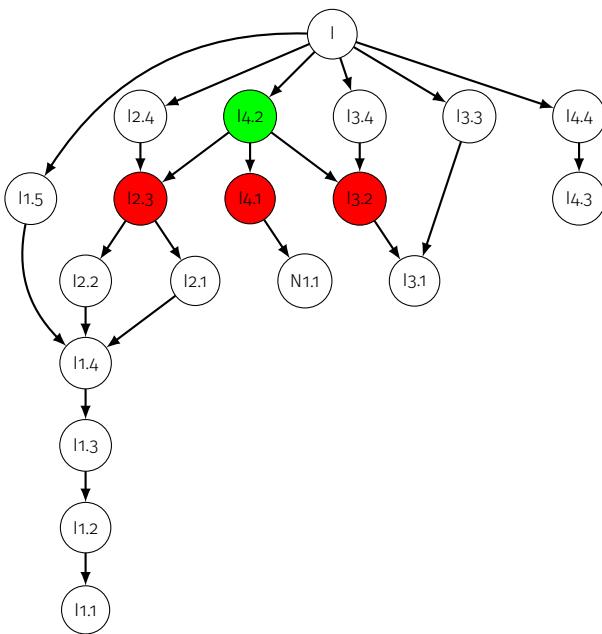
```

3 //           sous forme de liste de voisins
4 // Sortie : - liste des descendants de n (inclus)
5
6 res ← vide;
7 à_traiter ← vide;
8 à_traiter ← ajouter(n, à_traiter);
9
10 TantQue (à_traiter n'est pas vide) Faire
11   x ← tête(à_traiter);
12   à_traiter ← queue(à_traiter);
13   res ← ajouter(x, res);
14   l ← g[x]; // Liste des fils de x
15   à_traiter ← ajouter(l, à_traiter);
16 FinTantQue

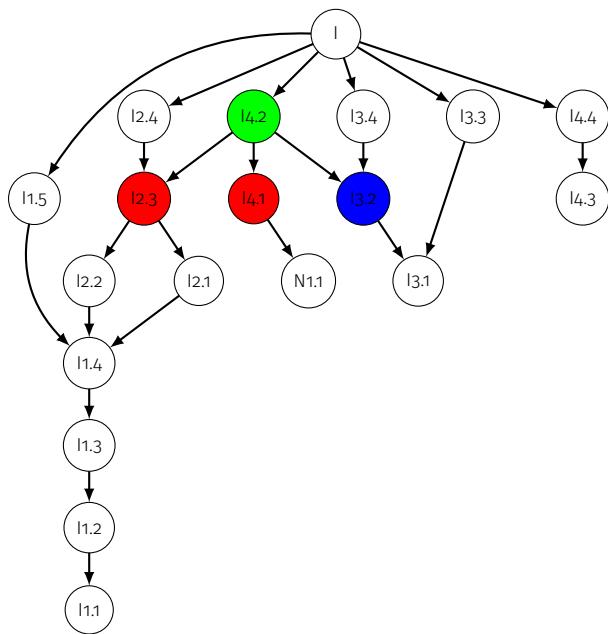
```

On montre en figures 5.3 et 5.4 le déroulé de l'algorithme en partant du nœud « I4.2 ». Les nœuds à traiter sont en rouge, la variable temporaire est en bleu et le résultat est en vert.

(a) *Initialisation.*(b) *Traitement de « I4.2 ».*



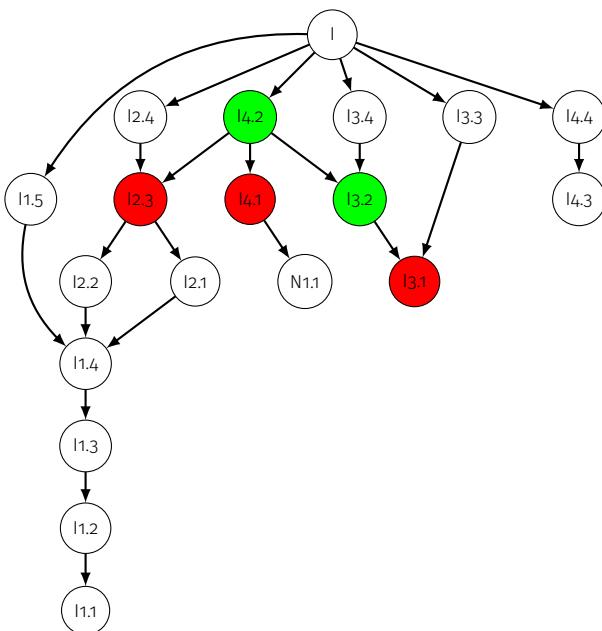
(c) Résultat de « I4.2 » avec trois fils à traiter.



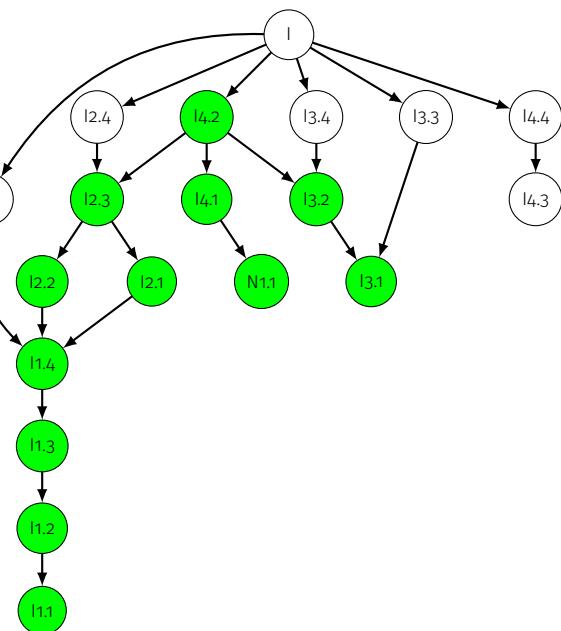
(d) Résultat une fois toutes les branches de « I4.2 » traitées.

FIGURE 5.3 — Parcours de graphe : déroulé de l'algorithme.

Partant de « I4.2 », on commence par le traiter, donc ce nœud passe en bleu, puis on s'attache à traiter tous ses fils. Le nœud « I4.2 » passe en vert et on doit traiter successivement les trois nœuds en rouge. Imaginons que le premier fils à traiter soit « I3.2 », il passe en bleu, puis en vert une fois le traitement effectué. Il reste dans cette branche à traiter le seul fils de « I3.2 ». Au final, pour suivre la séquence « I4.2 », il faut d'abord suivre l'arborescence des séquences donnée en figure 5.4b.



(a) Résultat de « I3.2 » avec un fils à traiter.



(b) Résultat final.

FIGURE 5.4 — Parcours de graphe : déroulé de l'algorithme (suite) et résultat final.

POUR ALLER PLUS LOIN... _____

Formation complémentaire

- ▶ Partie 4 du #1 Module fondamental : découvrir la programmation créative, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme ◎.

Articles

- ▶ [Les ingrédients des algorithmes](#), Gilles DOWEK, Thierry VIÉVILLE, Jean-Pierre, ARCHAMBAULT , Emmanuel BACCELLI et Benjamin WACK, INTERSTICES, 21 avril 2010.
- ▶ [Qu'est-ce qu'un algorithme?](#), Philippe FLAJOLET et Étienne PARIZOT, INTERSTICES, 24 février 2004;

Podcast et vidéo

- ▶ [À propos des algorithmes](#), Marie-Christine ROUSSET, Interstices, 15 juillet 2013.

Ouvrages

- ▶ *Apprendre à programmer avec Ocaml : Algorithmes, structures de données*, Sylvain CONCHON et Jean-Christophe FILLIÂTRE, EYROLLES, 2014;
- ▶ *Programmation efficace – 128 algorithmes qu'il faut avoir compris et codés en PYTHON au cours de sa vie*, Christoph DÜRR et Jill-Jenn VIE, ELLIPSES, 2016;
- ▶ *Algorithms*, Robert SEDGEWICK et Kevin WAYNE, 4^e éd., ADDISON-WESLEY PROFESSIONAL, 2011;
- ▶ *The Art of Computer Programming*, Donald E. KNUTH, ADDISON-WESLEY, 2011.

2 Programmation

Après avoir établi une solution algorithmique d'une problématique, vient le temps de son implémentation effective en tant que programme informatique. Pour cela, il faut pouvoir discuter sereinement avec la machine pour lui indiquer les consignes à respecter.

2.1 Langages de programmation

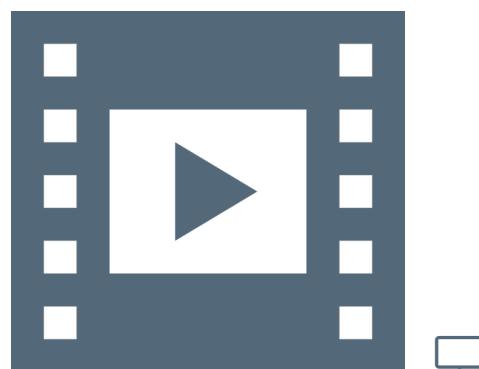
Comment le dialogue ou plutôt les ordres et impératifs se communiquent-ils à l'ordinateur ? Ce codage peut désormais se réaliser à l'aide d'une mosaïque de langages différents, mais alors lequel choisir ?

2.1.1 Langage machine

Gérard BERRY l'a exprimé dans une interview à Rue89 : « l'ordinateur est complètement con ». Que signifie ce constat fondateur et cette formulation ravageuse, mais claire comme de l'eau de roche ? Cela induit juste que l'ordinateur ne réalise que ce qu'on lui ordonne de faire pas plus, pas moins ; il faut tout lui dire, il n'a aucune initiative propre. Il ne sait pas faire quelque chose d'intelligent. La seule intelligence est dans l'esprit du programmeur.

Le langage machine, comme son nom l'indique, est celui que le processeur comprend. Pour chaque type de processeur un dialecte a été développé pour prendre en compte les particularités propres aux différents processeurs, mais la base est commune.

En fait, l'ordinateur, le smartphone ou la tablette ne sont pas des objets magiques. L'algorithme doit leur être pré-mâché dans ses plus infimes détails. Ce sont des dispositifs électroniques très avancés mais très simples dans leurs fonctionnalités. Leur avantage et leurs performances sont liés à leur rapidité d'exécution des tâches. Il ne savent



VIDÉO 5.5 – Langage machine.

obéir qu'à des ordres élémentaires qui correspondent aux instructions du langage machine, que l'on nomme également de l'*assembleur*.

Typiquement, le nombre d'instructions possibles est fini et relativement faible. Un processeur sait déplacer des éléments de la mémoire globale à la mémoire locale — dite vive ou dans les registres —, comparer deux valeurs ou les ajouter, multiplier, diviser. Il est aussi capable de sauter à un autre point du programme pour réaliser des boucles ou des tests. Un processeur est donc à même de faire des choses très simples et très unitaires.

```

1   movl    $0, -8(%rbp)
2   movl    $7, -4(%rbp)
3   cmpl    $0, -4(%rbp)
4   jle     .L2
5   movl    -4(%rbp), %eax
6   addl    %eax, -8(%rbp)
7   .L2:
8   movl    -8(%rbp), %eax

```

À titre d'illustration, le bout de code ci-dessus est en assembleur X86. L'instruction **movl** sert à mettre des valeurs dans des registres. Ici, on met la valeur **0** dans un certain registre, puis la valeur **7** dans un autre registre. Ensuite, on effectue une comparaison — instruction **cmpl** — entre la valeur **0** et le contenu du registre numéro **4**. Puis, à partir de cette comparaison, on fait un test et, en fonction du résultat, on va sauter à un autre point du programme qui est nommé **.L2**. Dans les deux cas de figure du test, on poursuit différemment les traitements.

On constate d'emblée que le langage machine est abscons, peu naturel et pas beau... Il est relativement difficile de comprendre la signification du code. Donc la seule conclusion à tirer est qu'on n'a pas du tout envie d'écrire du langage assembleur, c'est pourquoi on va utiliser des langages de plus haut niveau et de la compilation.

POUR ALLER PLUS LOIN...

Articles

- ▶ Gérard BERRY. L'ordinateur est complètement con, Xavier LA PORTE, L'OBS AVEC RUE 89, 21 novembre 2016.
- ▶ Le [langage assembleur](#), WIKIPÉDIA.

Vidéo

- ▶ Gérard Berry : l'homme qui orchestre les ordinateurs, CNRS, 2014 (6mn).

2.1.2 Langage de haut niveau et compilation

Quand on veut écrire un programme, on part d'une idée. Que veut-on faire et calculer? Après, on pense aux grandes lignes, c'est-à-dire à la mise en place d'une espèce de séquençage, comme précisé dans la partie algorithmique. On définit alors l'algorithme et on décide des structures de données. Ensuite, on choisit un langage de programmation en fonction d'un certain nombre de critères — ce choix est abordé en section suivante — et on se lance dans l'écriture du code.

On parle de langage de haut niveau car la sémantique et la syntaxe de ce type de langage se veulent proches de la compréhension de l'humain et non plus de la machine comme avec l'assembleur. Néanmoins, pour que le code que l'on écrit soit compréhensible par l'ordinateur, il faut le traduire en langage machine. C'est là qu'intervient la phase de compilation : obtenir un programme dit *exécutable* et donc exploi-



VIDÉO 5.6 — *Langages et compilation*.

table à partir d'un fichier texte constitué d'un ensemble d'instructions correspondant au langage de programmation choisi.

Le compilateur est un programme comme les autres qui agit comme traducteur en prenant en entrée un fichier texte écrit par exemple en C, C++ ou JAVA et livre un exécutable. Au passage, le compilateur vérifie un certain nombre de règles d'écriture comme la syntaxe ou le typage des variables, mais aussi propose des aides pour la détection des erreurs, voire dispose d'options pour optimiser le programme.

Il existe un second type de langage de haut niveau, les langages dits *interprétés*. Le code source — au format texte — n'est plus directement présenté au compilateur, mais confié à ce que l'on appelle un *interpréteur*, qui lui se charge de traduire les instructions en langage machine. Par exemple, PERL ou PYTHON sont des langages interprétés. Ici, c'est l'interpréteur qui a été compilé au préalable en fonction du processeur et du système d'exploitation.

Puisqu'il y a un intermédiaire, les langages interprétés sont généralement plus lents, mais de nos jours ce n'est plus un souci car d'une part, la puissance des machines apporte la transparence et, d'autre part, on peut combiner les langages entre eux sous forme de bibliothèques en fonction des tâches à accomplir. C'est à ce propos la grande force de PYTHON.

Le grand avantage des langages interprétés est, outre l'absence de compilation, d'être multiplate-forme, à savoir qu'un même fichier source « tournera » de manière identique quel que soit le système d'exploitation. En quelque sorte, ils sont « encore plus » de haut niveau. À ce titre, PYTHON est exemplaire car, comparativement, se trouve le plus proche du langage naturel.

POUR ALLER PLUS LOIN...

Formation complémentaire

- ▶ #1 Module fondamental : découvrir la programmation créative, CLASS'CODE, Partie 1, pour s'initier à la programmation avec SCRATCH (cela permet de s'adresser aux futurs élèves formés à SCRATCH au collège ou en périscolaire).
- ▶ Le langage assembleur, WIKIPÉDIA.

Articles

- ▶ Compilateur, WIKIPÉDIA.
- ▶ Grace Hopper, conceptrice du premier compilateur, WIKIPÉDIA.
- ▶ Comment faire confiance à un compilateur ? Xavier LEROY, INTERSTICES, 21 mai 2010.

Podcast

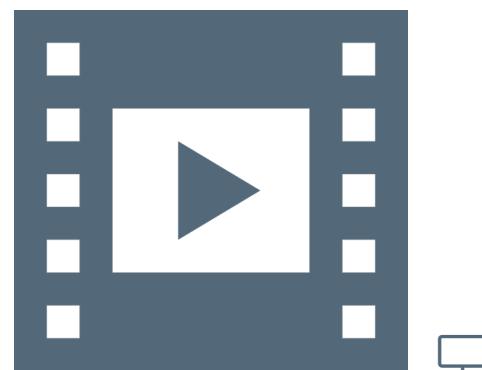
- ▶ À propos des compilateurs, Sandrine BLAZY et Joanna JONGWANE, INTERSTICES, 29 mars 2012, (09mn, 30s).

2.2 Paradigmes de programmation

2.2.1 Bogue

Les bugs jalonnent l'histoire de l'informatique. Pour le grand public, peut-être que le plus célèbre serait le « bug de l'an 2000 ». Il y en a d'autres, parfois dramatique, citons deux exemples.

La première anecdote est liée à l'USS Yorktown, un navire américain de guerre qui a été lancé en 1996 avec un programme Navy's Smart Ship qui signifie « bateau intelligent ». Lors d'essais, un membre d'équipage



VIDÉO 5.7 — Histoire de bugs.

s'est trompé sur ce qu'il rentrait comme valeur d'entrée dans la machine, il a rentré un zéro sur une valeur qui, en théorie, n'était pas censée être nulle. La conséquence, c'est qu'il y a eu dans le programme une division par zéro qui est remontée et a complètement planté le système d'exploitation. La conséquence est que plus rien ne fonctionnait sur le bateau, pas même les moteurs.



On peut trouver plus amples détails sur [WIKIPÉDIA](#) mais, ce qu'il faut retenir, c'est que les valeurs d'entrée doivent être testées parce qu'on n'est jamais sûr que l'utilisateur final est bien conscient qu'il y a des valeurs interdites. Par ailleurs, ce genre de valeur non attendue est également une porte d'entrée pour les attaques sur les programmes.

Un autre exemple concerne le lanceur de satellites Ariane 5, successeur d'Ariane 4, qui fonctionnait très bien avec un programme spécifique. Lors de la conception d'Ariane 5 ce programme a été repris à la lettre et implémenté sans autre vérification. Il se trouve simplement que le lanceur Ariane 5 est plus lourd que son prédecesseur et des valeurs maximales qui étaient correctes pour Ariane 4 ne l'étaient plus pour Ariane 5.

Au premier lancer d'Ariane 5, les valeurs maximales ont été dépassées et, en informatique, quand on dépasse les valeurs maximales autorisées d'un entier ou d'un flottant, il se passe des choses bizarres, en particulier pour les nombres à virgule flottante, on peut avoir des valeurs très grandes qui deviennent très petites ou des entiers positifs qui virent en négatif. Aussi, une valeur complètement absurde a fait que le lanceur de satellites a commencé à partir sur le côté et il a fallu le détruire pour ne pas qu'il tombe sur les habitations.

Ici, la conclusion est qu'il faut faire attention aux variables qu'on utilise et à leur type pour être sûr qu'on ne va pas dépasser les valeurs maximales et minimales autorisées (par exemple des tableaux).

Alors qu'est-ce que c'est qu'un *bug*? C'est *in fine* le résultat d'un programme que l'utilisateur ne désire pas. Cela peut être une erreur de programmation — autrement dit qu'effectivement la personne qui a programmé s'est trompée —, l'oubli d'un cas particulier, une typo ou une coquille, un dépassement de capacités comme dans l'exemple précédent, un accès illicite à la mémoire — c'est-à-dire accéder à un morceau de mémoire qui n'était pas prévu et donc qui n'était pas initialisé avec une valeur correcte —, etc.

Il existe aussi une catégorie de bug plus sournois, mais finalement au moins aussi fréquent, relatifs aux erreurs de communication, c'est-à-dire qu'effectivement, entre la personne qui a demandé le programme et la personne qui l'a programmé à la fin, il y a eu une mauvaise compréhension sur les valeurs d'entrée, sur ce que devait faire le programme, ou tout autre état de cause.

Il peut également y avoir une erreur de communication entre deux programmeurs. Les programmes actuels sont en effet écrits par des milliers de personnes qui doivent interagir et comprendre ce que fait le morceau de programme des autres développeurs. La figure 5.5 en apporte l'illustration en comparant les nombres de lignes de code pour certaines applications connues. On remarque que déjà un *pacemaker* et ses quelques 100 000 lignes de code est difficile à programmer par une seule personne, qu'en est-il alors de FACEBOOK et et ses soixante millions de lignes de code? Il y a fort à parier que des *bugs* se glissent dans les programmes et les fassent planter de temps à autres.

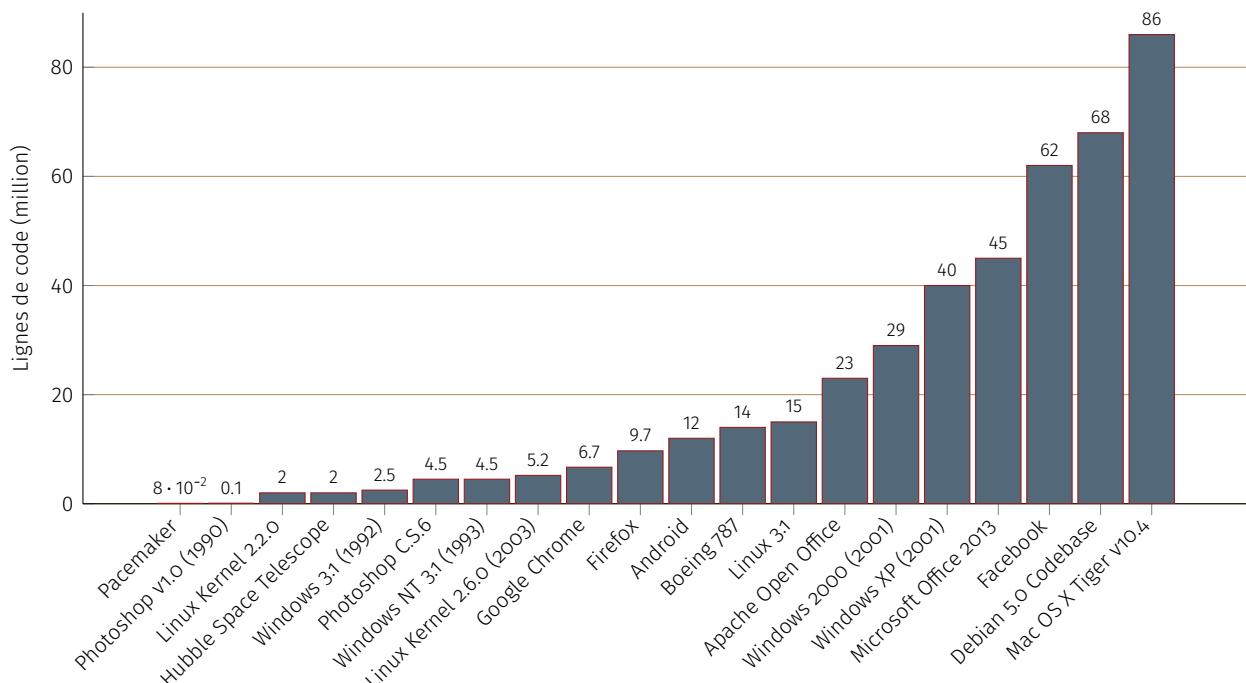


FIGURE 5.5 — Nombre de lignes de code de quelques applications connues (en million).

POUR ALLER PLUS LOIN...

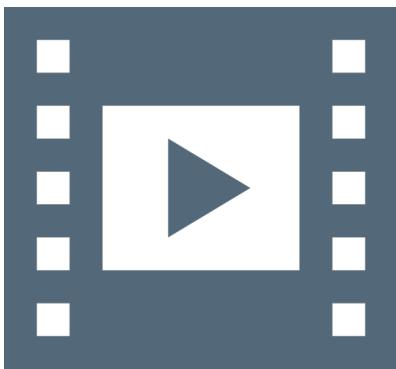
Formation complémentaire

- ▶ #1 Module fondamental : découvrir la programmation créative, CLASS'CODE, Partie 2.4.

Articles

- ▶ Dis papa (ou maman) comment arrivent les *bugs* dans le monde numérique, Aurélien ALVAREZ et Thierry VIÉVILLE, IMAGES DES MATHÉMATIQUES, 2014.
- ▶ Comment sont entrés mes images, textes, données dans la machine ? avec l'article ci-dessus, le film les Sépas sur les *bugs* et le jeu des pixels à travers le paravent en se passant bit à bit une image pour la reconstruire en aveugle.
- ▶ Idée reçue : c'est la faute à l'ordinateur ! Sylvie BOLDO, INTERSTICES, 19 février 2010.

2.2.2 Humains et langages



VIDÉO 5.8 — Choix d'un langage de programmation.

2. Le langage C a été conçu pour programmer les premiers systèmes d'exploitation UNIX au tout début de la décennie 1970.

Il existe de très nombreux langages de programmation aux caractéristiques différentes. Ce qu'il faut retenir, c'est qu'il est toujours possible de tout faire dans tous les langages de programmation. En revanche, cela ne sera pas toujours aussi facile, mais on peut utiliser n'importe quel langage de programmation pour écrire n'importe quel algorithme.

On peut classer les langages par paradigmes de programmation :

- *paradigme impératif*, le plus usuel et le plus proche du processeur ; on donne des ordres successivement, on fait des boucles, etc. Les exemples les plus traditionnels sont C² et FORTRAN ;
- *paradigme fonctionnel*, on y considère les fonctions comme des objets à part entière qu'on peut passer à d'autres fonctions. C'est une vision un peu plus mathématique de la programmation. Les exemples de tels langages sont HASKELL et OCAML ;
- *paradigme objet*, manière de faire de la programmation un peu plus générique, où effectivement on va pouvoir partager un certain nombre de fonctions entre des objets différents. Ce paradigme est très répandu actuellement et permet la réutilisation ou factorisation de code, gain de temps de développement. Les exemples sont JAVA et C++ ;
- *paradigme par contrainte* est plus une programmation logique dont un exemple est PROLOG. Imaginons un calcul d'emploi du temps avec un certain nombre de contraintes : tel professeur n'est là que lundi matin, telle salle ne peut faire que de l'informatique, etc. Toutes ces contraintes sont prises en compte et on obtient idéalement un emploi du temps correct pour tous les professeurs et toutes les salles.
- *paradigme événementiel*, le plus utilisé pour les interfaces graphiques ou la robotique. Le programme ne va pas se dérouler point par point, mais réagir à un certain nombre de *stimuli* de l'environnement : un clic de souris sur l'interface d'un programme ou le contact d'un robot avec un obstacle.

Dans les faits, beaucoup de langages de programmation sont multi-paradigmes ; c'est le cas de C++, JAVA ou PYTHON... Notons également qu'on peut interfaçer les langages entre eux de manière à profiter des atouts des deux langages ; PYTHON est typiquement riche de bibliothèques de ce type — écrites en C ou C++ — qui sont transparentes pour le programmeur.

Au final, choisir un langage de programmation est invariablement une question compliquée. Cela dépend de ce qu'on veut programmer, de ce qu'on a l'habitude de faire et dans quel cadre.

- ▶ Quel paradigme de programmation ?
- ▶ Quelles interfaces avec d'autres développements ? Souvent, le programme n'est qu'un morceau de projet plus conséquent et le « choix » s'avère imposé.

- ▶ Quelles bibliothèques de support? Pour une interface graphique, l'option adoptée peut être guidé par l'existence de bibliothèques graphiques ayant fait leur preuve.
- ▶ Quelle connaissance préalable du langage? A-t-on déjà de l'expérience dans un langage donné pour aller plus vite dans le développement?
- ▶ Quelle aide ou tutorat est mobilisable? Peut-on solliciter son entourage pour avancer plus rapidement?

POUR ALLER PLUS LOIN...

Articles

- ▶ Une définition des paradigmes de programmation, WIKIPÉDIA.
- ▶ Bjarne STROUSTRUP : le père de C++, un langage qui a de la classe, Pierre VANDEGISTE, INTERSTICES, 15 juin 2004.
- ▶ La programmation par contraintes, Étienne PARIZOT, Sylvain SOLIMAN et François FAGES, INTERSTICES, 24 février 2004.
- ▶ Programmation des échecs et d'autres jeux, Olivier TEYTAUD, Interstices, 26 novembre 2012.

Vidéo

- ▶ Premiers principes des langages de programmation, cours introductif par Gilles DOWEK à destination des professeurs des lycées ISN, CanalU (collection INRIA Science Info Lycée Profs, (90 mn), 2010).

Site de programmation Interactive

- ▶ <http://pythontutor.com/> est un outil en ligne qui n'est pas seulement dédié à PYTHON comme son nom l'indique mais aussi à JAVASCRIPT : il permet d'écrire du code et de visualiser le résultat, ce qui peut aider à comprendre la programmation.

3 Que faire de ces ressources ? Autoévaluation

Le questionnaire à choix multiple* — QCM — à suivre clôture le présent chapitre « Langages de l'informatique » et correspond à chaque sujet abordé.

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style `LATeX` usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

☰
QUIZ 9 — DÉCOMPOSITION EN TÂCHES
2 POINTS

QUIZ 9.1 — TÂCHES ÉLÉMENTAIRES 1 POINT

Pourquoi doit-on découper une tâche en tâches élémentaires?

Parce que ça nous embête.

Parce que l'ordinateur ne sait, par essence, exécuter que des tâches élémentaires.

Parce que c'est drôle.

Parce que l'ordinateur qui comprend tout veut vérifier qu'on sait le faire aussi.

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER

**QUIZ 9.2 — ARBRE GÉNÉALOGIQUE 1 POINT**

Prenons un graphe bien connu : l'arbre généalogique. Les nœuds sont les personnes et il y a une flèche d'un parent à un enfant. Que dois-je faire pour aller de mon grand-père à moi-même ?

- Rester sur le nœud.
- Descendre une flèche.
- Remonter une flèche.
- Descendre deux flèches.
- Remonter deux flèches.
- Descendre puis remonter une flèche.
- Remonter puis descendre une flèche.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 10 — NOTION DE VARIABLE****1 POINT****QUIZ 10.1 — UTILITÉ DES VARIABLES 1 POINT**

À quoi peut servir une variable ?

- À stocker la valeur d'un compteur.
- À stocker une valeur ou le résultat d'un calcul.
- À jouer de la musique.
- À donner accès à un espace mémoire.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 11 — INSTRUCTIONS ÉLÉMENTAIRES****3 POINTS****QUIZ 11.1 — ALGORITHME SIMPLE (I) 1 POINT**

Voici un algorithme simple à exécuter. Quelle est la valeur de *i* ?

```

1 i ← 2;
2 Si i < 0 Alors
3     i ← i + 1;
4 Sinon
5     i ← i - 1;
6 FinSi

```

- i* vaut 2.
- i* vaut -1.
- i* vaut 3.
- i* vaut 1.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 11.2 — ALGORITHME SIMPLE (II) 1 POINT

Voici un algorithme simple à exécuter. Quelles sont les valeurs respectives de x et de y ?

```

1 x ← 0; y ← 0;
2 Pour i de 0 à 5 inclus Faire
3   x ← x + 1;
4   y ← y + i;
5 FinPour

```

 x = 5 et y = 16.

 x = 6 et y = 12.

 x = 7 et y = 18.

 x = 6 et y = 15.

 x = 5 et y = 14.

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER
QUIZ 11.3 — BOUCLES ET ALGORITHMIQUE 1 POINT

Sélectionnez les affirmations vraies. Il peut y en avoir plusieurs!

 Une boucle « Tant que » se termine toujours.

 On peut avoir deux algorithmes différents pour la même chose.

 Un tableau, au niveau algorithmique, doit être composé de valeurs de même type (numérique par exemple).

 Le processeur (unité centrale) d'un ordinateur usuel peut traiter toutes les cases d'un tableau en même temps.

 La boucle « Pour » peut s'écrire avec une boucle « Tant que ».

 L'affectation peut permettre d'augmenter une valeur.

 L'affectation ne peut que permettre d'augmenter une valeur (et pas de la diminuer).

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER
QUIZ 12 — CULTURE ALGORITHMIQUE
1 POINT
QUIZ 12.1 — ALGORITHMES 1 POINT

Sélectionnez les affirmations vraies. Il peut y en avoir plusieurs!

 Je peux en inventer.

 C'est seulement sur Internet.

 Je peux en trouver dans des livres.

 Des gens en inventent des nouveaux.

 Ça ne sert à rien.

VÉRIFIER
AFFICHER LA RÉPONSE
RÉINITIALISER

QUIZ 13 — LANGAGE MACHINE 2 POINTS

QUIZ 13.1 — ÉCRITURE 1 POINT
On peut écrire l'assembleur directement à la main.

Vrai.
 Faux.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 13.2 — PRATICITÉ 1 POINT
C'est pratique et simple d'écrire l'assembleur directement à la main.

Vrai.
 Faux.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 14 — COMPILEATION 1 POINT

QUIZ 14.1 — COMPILATEURS 1 POINT
À propos des compilateurs. Que dire ?

Ils permettent de détecter des erreurs simples.
 Ils permettent de détecter toutes les erreurs.
 Ils transforment un texte compréhensible par un humain en quelque chose compréhensible par l'ordinateur.
 Ils sont des programmes qui prennent en entrée un programme.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 15 — BUGS ET BOUGES 5 POINTS

QUIZ 15.1 — OCCURRENCE DES BUGS 1 POINT
Les bugs arrivent parce que (bien sélectionner toutes les possibilités) :

les programmeurs sont bêtes.
 les programmeurs se sont trompés.
 les programmeurs n'ont pas prévu un cas particulier.
 l'ordinateur n'a pas obéi aux programmeurs.
 les programmeurs ont bien fait ce qu'on leur a demandé, mais pas ce que l'on voulait.
 les programmeurs n'ont pas communiqué entre eux.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

Les questions à suivre concernent les ordres de grandeurs de taille de code. Quels sont-ils ?

QUIZ 15.2 — PREMIER PROGRAMME 1 POINT

- Moins de 100 lignes.
- De 100 à 1 000 lignes.
- Quelques millions de lignes.
- Quelques dizaines de millions de lignes.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 15.3 — PREMIER PROJET OU JEU 1 POINT

- Moins de 100 lignes.
- De 100 à 1 000 lignes.
- Quelques millions de lignes.
- Quelques dizaines de millions de lignes.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 15.4 — FACEBOOK 1 POINT

- Moins de 100 lignes.
- De 100 à 1 000 lignes.
- Quelques millions de lignes.
- Quelques dizaines de millions de lignes.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 15.5 — BOEING 787 1 POINT

- Moins de 100 lignes.
- De 100 à 1 000 lignes.
- Quelques millions de lignes.
- Quelques dizaines de millions de lignes.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 16 — LANGAGE DE PROGRAMMATION

3 POINTS

QUIZ 16.1 — CHOIX DES LANGAGES 1 POINT

Pourquoi y-a-t-il plusieurs langages de programmation ?

- Parce que les programmeurs ont des préférences.

- Parce que cela crée plus d'emplois de programmeur.
- Parce que certains langages sont plus adaptés à résoudre certains problèmes.
- Pour que ce soit difficile d'échanger des bouts de programme.
- Parce que de nouveaux paradigmes (types de programmation) sont apparus.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 16.2 — PARADIGMES DE PROGRAMMATION 1 POINT**

Lesquels sont des paradigmes de programmation ?

- Objet.
- Cuisine.
- Fonctionnel.
- Événementiel.
- Clavier.
- Impératif.
- Utile.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 16.3 — PROGRAMME EN C 1 POINT**

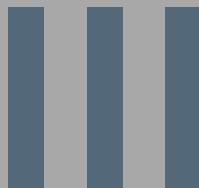
Complétez le programme suivant en C. Le but est de chercher la valeur 17 dans un tableau et de le remplacer par 0. On suppose avoir un tableau *t* de taille *N*. On peut alors écrire :

```
for (i=0;i < N;i++) {
    ????
    t[i] = 0; }
```

- while (*t[i]==17*)
- if (*t[i]==17*)
- t[i]=17*
- t[0]==17*

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

TROISIÈME PARTIE



NUMÉRIQUE : CULTURE ET PRATIQUE

- | | |
|------------|--|
| 185 | CHAPITRE 6
DONNÉES ET OBJETS |
| 215 | CHAPITRE 7
RÉSEAUX ET ENVIRONNEMENT |

DONNÉES ET OBJETS

LE PROGRAMME OFFICIEL SNT DÉFINIT SEPT CHAMPS D'INVESTIGATION. Les quatre premiers abordés dans ce chapitre concernent les données et les objets connectés, à savoir : les données structurées et leur traitement, la photographie numérique, l'informatique embarquée et les objets connectés, la localisation et la cartographie.

Chaque thématique possède un canevas commun : « Découvrir la thématique » et « Réaliser des activités ». Si la première de ces sections vise la présentation du sujet, la seconde est ouverte aux suggestions et amenée à intégrer les contributions des enseignants.

Il va sans dire que chaque thème énoncé bénéficie des contenus exposés dans les autres parties du document, notamment « Informatique, création numérique » et « Fondements de l'informatique ».

1 Données et traitements

1.1 Découvrir la thématique

1.1.1 Anchorage dans le réel

A. POINTS-CLÉS

- ▶ L'information humaine est transrite sous forme de données afin d'être manipulée numériquement.
- ▶ On distingue les données qui doivent être entrées dans la machine, des résultats de calculs, ou sortie des algorithmes.
- ▶ Les données en tant qu'objets numériques forment un bien non rival* dont la copie ne coûte quasiment rien, et que l'on peut dupliquer sans le consommer.
- ▶ La production gigantesque de données pose des problèmes planétaires en matière d'environnement (consommation énergétique, utilisation de ressources naturelles rares).
- ▶ La prolifération de données pose également un problème de pérennité à long terme (à l'échelle de plusieurs dizaines d'années) non résolu actuellement.

* *Bien non rival* : la plupart des objets matériels sont des biens rivaux c'est-à-dire que si on les consomme ou les utilise, ils ne sont plus disponibles pour les autres consommateurs, ce n'est pas le cas des objets informationnels comme par exemple une bonne histoire : si je la partage, elle reste intacte, voire elle s'enrichit.

SOMMAIRE

1	Données et traitements
1.1	Découvrir la thématique 185 1.1.1 Anchrage dans le réel ■ 1.1.2 Volet historique ■ 1.1.3 Explication des notions 1.2 Réaliser des activités 190
2	Photographie numérique
2.1	Découvrir la thématique 191 2.1.1 Anchrage dans le réel ■ 2.1.2 Volet historique ■ 2.1.3 Explication des notions 2.2 Réaliser des activités 196
3	Informatique embarquée et objets connectés
3.1	Découvrir la thématique 197 3.1.1 Anchrage dans le réel ■ 3.1.2 Volet historique ■ 3.1.3 Explication des notions 3.2 Réaliser des activités 201
4	Localisation et cartographie
4.1	Découvrir la thématique 202 4.1.1 Anchrage dans le réel ■ 4.1.2 Volet historique ■ 4.1.3 Explication des notions 4.2 Réaliser des activités 207
5	Que faire de ces ressources ? Quiz



VIDÉO 6.1 — Données et traitements.

B. MOTS-CLÉS

- ▶ **Mégadonnées** : (*Big Data*) on parle de mégadonnée quand le volume de données est tel qu'on peut faire des analyses statistiques qui permettent de prédire des informations, même si les données sont très diverses, sans information structurée, et approximatives.
- ▶ **Données ouvertes** : (*Open Data*) données numériques, d'origine publique ou privée, diffusées de manière structurée selon une méthode et une licence libre, garantissant leur libre accès et leur réutilisation par toutes et tous, sans restriction technique, juridique ou financière.
- ▶ **Licence libre** : licence s'appliquant à une œuvre de l'esprit (document, logiciel, etc.) par laquelle l'autrice ou l'auteur concède les droits que lui confère le droit d'auteur : usage de l'œuvre, étude de l'œuvre pour en comprendre le fonctionnement ou l'adapter à ses besoins, modification (amélioration, extension et transformation) ou incorporation de l'œuvre en une œuvre dérivée, redistribution de l'œuvre, c'est-à-dire sa diffusion à d'autres usagers, y compris commercialement.
- ▶ **Informatique durable** : (*green IT*) vise à réduire l'empreinte écologique, économique et sociale des technologies de l'information et de la communication.
- ▶ **Règlement général sur la protection des données (RGPD)** : renforce et unifie la protection des données pour les personnes au sein de l'Union Européenne.

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Les données constituent la matière première de toute activité numérique. Afin de permettre leur réutilisation, il est nécessaire de les conserver de manière persistante. Les structurer correctement garantit que l'on puisse les exploiter facilement pour produire de l'information. Cependant, les données non structurées peuvent aussi être exploitées, par exemple par les moteurs de recherche.

IMPACTS SUR LES PRATIQUES HUMAINES

L'évolution des capacités de stockage, de traitement et de diffusion des données fait qu'on assiste aujourd'hui à un phénomène de surabondance des données et au développement de nouveaux algorithmes capables de les exploiter.

L'exploitation de données massives (*Big Data*) est en plein essor dans des domaines aussi variés que les sciences, la santé ou encore l'économie. Les conséquences sociétales sont nombreuses tant en termes de démocratie, de surveillance de masse ou encore d'exploitation des données personnelles.

Certaines de ces données sont dites ouvertes (*Open Data*), leurs producteurs considérant qu'il s'agit d'un bien commun. Mais on assiste aussi au développement d'un marché de la donnée où des entreprises collectent et revendent des données sans transparence pour les usagers. D'où l'importance d'un cadre juridique permettant de protéger les usagers, préoccupation à laquelle répond le règlement général sur la protection des données (RGPD). Les centres de données (*Data Center*) stockent des serveurs mettant à disposition les données et des applications les exploitant. Leur fonctionnement nécessite des ressources (en eau pour le refroidissement des machines, en électricité pour leur fonction-

nement, en métaux rares pour leur fabrication) et génère de la pollution (manipulation de substances dangereuses lors de la fabrication, de la destruction ou du recyclage). De ce fait les usages numériques doivent être pensés de façon à limiter la transformation des écosystèmes (notamment le réchauffement climatique) et à protéger la santé humaine.

1.1.2 Volet historique

L'idée de pouvoir traiter mécaniquement de l'information est ancienne, dès le XVII^e siècle, par exemple, Gottfried Wilhelm LEIBNIZ va chercher à établir une langue dite caractéristique universelle, qui permettrait d'exprimer la totalité des pensées humaines et pourrait résoudre des problèmes par un calculateur (*calculus ratiocinator*), anticipant l'informatique de plus de trois siècles. Il faudra attendre le XX^e siècle pour comprendre qu'une telle machine est un objet impossible ne serait-ce qu'en mathématiques, d'après les théorèmes d'Alonzo CHURCH et Alan TURING : très simplement, certains calculs (par exemple savoir si un programme va boucler à l'infini, le problème de l'arrêt) nécessitent des temps... Infinis. On commençait à comprendre les limites de l'intelligence mécanique avant même de l'avoir fabriquée.

Ces mêmes personnes ont pourtant fondé, dans les années 1930, l'informatique, un domaine d'activité scientifique, technique et industriel concernant le traitement automatique de l'information par l'exécution de programmes informatiques par des machines. On peut attribuer à Ada LOVELACE, un siècle avant, d'avoir compris que l'on peut « calculer sur des nombres mais aussi sur des symboles », on parlerait de données numériques et symboliques aujourd'hui. Il est intéressant de noter que ces idées sont nées avant la technologie permettant de les mettre en œuvre.

Parallèlement, dans les années 1880, Herman HOLLERITH, futur fondateur d'IBM, fonde la mécanographie en inventant une machine électromécanique destinée à faciliter le recensement en stockant les informations sur une carte perforée. Ces premières cartes perforées ont fait leur apparition au XVIII^e siècle dans divers automates et en particulier les métiers à tisser, les orgues de Barbarie et les pianos mécaniques.

L'histoire de l'informatique débute véritablement au milieu du XX^e siècle avec l'architecture de John von NEUMANN, mise en application dans la machine universelle d'Alan TURING : les ordinateurs dépassent la simple faculté de calculer et peuvent désormais commencer à traiter des données.

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

- ▶ 1930 : utilisation des cartes perforées, premier support de stockage de données.
- ▶ 1956 : invention du disque dur permettant de stocker de plus grandes quantités de données, avec un accès de plus en plus rapide.
- ▶ 1970 : invention du modèle relationnel (E. L. CODD) pour la structuration et l'indexation des bases de données.
- ▶ 1979 : création du premier tableur, VisiCalc.
- ▶ 2009 : *Open Government Initiative* du président Obama.
- ▶ 2013 : charte du G8 pour l'ouverture des données publiques.



Ada LOVELACE (1815-1852).



John von NEUMANN (1903-1953).

1.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Toutes les informations humaines se codent en binaire; bien entendu ce n'est pas l'objet réel, ce n'est que son reflet numérique.
- ▶ Une donnée est spécifiée par des valeurs et chaque valeur a un type (par exemple : vrai ou faux, on dit booléen; ou bien numérique ou textuel; ou encore un type spécifique, comme une date); selon le type de la donnée on ne fait pas les mêmes opérations.
- ▶ Une donnée se décompose de manière atomique en données élémentaires, par exemple le nom d'une personne en prénom et patronyme. Bien structurer les données facilite leur traitement par des algorithmes.
- ▶ Une collection de données peut être ordonnée sous forme de liste, ou bien sans ordre sous forme d'un ensemble.
- ▶ La façon de structurer les données influe fortement sur les opérations de traitement : il est par exemple bien plus efficace de rechercher une donnée dans une collection toujours ordonnée, mais y insérer une information est plus coûteux.

B. MOTS-CLEFS

- ▶ Donnée : représentation d'une information au sein d'un système informatique.
- ▶ Métadonnée : donnée servant à définir ou décrire une autre donnée, pour permettre sa manipulation.
- ▶ Une base de données regroupe plusieurs collections de données reliées entre elles.
- ▶ Descripteur : mot ou un groupe de mots choisi pour caractériser les informations contenues dans un document et pour faciliter les recherches.

C. CE QUE DIT LE PROGRAMME

DONNÉES ET INFORMATION

Une donnée est une valeur décrivant un objet, une personne, un événement digne d'intérêt pour celui qui choisit de la conserver. Le numéro de téléphone d'un contact est une donnée. Plusieurs descripteurs peuvent être utiles pour décrire un même objet (par exemple des descripteurs permettant de caractériser un contact : nom, prénom, adresse et numéro de téléphone).

Une collection regroupe des objets qui partagent les mêmes descripteurs (la collection des contacts d'un carnet d'adresses par exemple). La structure de table permet de présenter une collection : les objets en ligne, les descripteurs en colonne et les données à l'intersection. Les données sont alors dites structurées. Pour assurer la persistance des données, ces dernières sont stockées dans des fichiers. Le format CSV (*Comma Separated Values*, les données avec des séparateurs) est un format de fichier simple permettant d'enregistrer une table. À tout fichier sont associées des métadonnées qui permettent d'en décrire le contenu. Ces métadonnées varient selon le type de fichier (date et coordonnées de géolocalisation d'une photographie, auteur et titre d'un fichier texte, etc.).

Les données comme les métadonnées peuvent être capturées et enregistrées par un dispositif matériel ou bien renseignées par un humain. Elles sont de différents types (numériques, textes,

dates) et peuvent se traiter de manières diverses (calcul, tri, affichage, etc.).

Certaines collections typiques sont utilisées dans des applications et des formats standardisés leur sont associés : par exemple le format ouvert vCard (extension **.vfc**) pour une collection de contacts.

Une *base de données* regroupe plusieurs collections de données reliées entre elles. Par exemple, la base de données d'une bibliothèque conserve les données sur les livres, les abonnés et les emprunts effectués.

ALGORITHMES ET PROGRAMMES

La recherche dans des *données structurées* a d'abord été effectuée selon une indexation préalable faite par l'homme. Des algorithmes ont ensuite permis d'automatiser l'indexation à partir de textes, d'images ou de sons.

Une table de données peut faire l'objet de différentes opérations : rechercher une information précise dans la collection, trier la collection sur une ou plusieurs propriétés, filtrer la collection selon un ou plusieurs tests sur les valeurs des descripteurs, effectuer des calculs, mettre en forme les informations produites pour une visualisation par les utilisateurs.

La recherche dans une base comportant plusieurs collections peut aussi croiser des collections différentes sur un descripteur commun ou comparable.

MACHINES

Les fichiers de données sont stockés sur des supports de stockage : internes (disque dur ou SSD) ou externes (disque, clef USB), locaux ou distants (*cloud*). Ces supports pouvant subir des dommages entraînant des altérations ou des destructions des données, il est nécessaire de réaliser des sauvegardes.

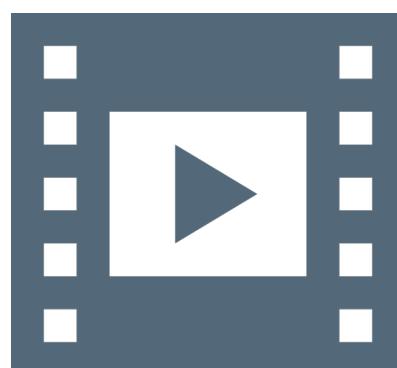
Des recherches dans les fichiers se font à l'intérieur même des ordinateurs, soit sur la base de leurs métadonnées, soit à partir d'une indexation (à la manière des moteurs de recherche disponibles sur le Web).

Les grandes bases de données sont souvent implémentées sur des serveurs dédiés (machines puissantes avec une importante capacité de stockage sur disques). Ces centres de données doivent être alimentés en électricité et maintenus à des températures suffisamment basses pour fonctionner correctement.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

- Une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte générale, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours.
- Des exemples concrets de ce qui est étudié ou produit aujourd'hui dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la



VIDÉO 6.2 — Données structurées et traitements, Élisa FROMONT.



parole à des personnes travaillant actuellement dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

Cette conférence sur les données et leur traitement a eu lieu le 28 mars 2019 dans le cadre du projet CLASS'CODE à l'Université de Nantes.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ Partie 1 du #2 Module thématique : manipuler l'information, CLASS'CODE. Cette formation pour enseignants du secondaire offre des vidéos accessibles en cliquant sur le pictogramme 
- ▶ Dans le manuel ISN ([librement accessible en ligne](#)), les chapitres de la deuxième partie, constituent une auto-formation alternative avec cours et exercices, notamment chapitre 11 page 150 structurer l'information.
- ▶ Pour les enseignants : la conférence de Françoise Tort sur les données et leur traitement, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Le module o « Données personnelles et législation » du Mooc [Protection de la vie privée dans le monde numérique](#), produit par l'INRIA sur la plateforme FUN.
- ▶ Une [sélection d'articles](#) du site INTERSTICES.

Créer son cours

- ▶ Un petit film sur l'[histoire de l'informatique](#) permet aux élèves de découvrir l'histoire des humains et des idées qui ont fondé l'informatique, avec un [livret](#), librement réutilisable.
- ▶ Un [document de réflexion sur les données personnelles](#), partageable avec les élèves.
- ▶ Un [document sur l'identité numérique](#), partageable avec les élèves, en lien avec la thématique Internet.

1.2 Réaliser des activités

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶  Découverte de quelques types de données structurées.
- ▶  Initiation au traitement de données avec Python.
- ▶  Qu'est-ce que le cloud ? Quelle est son empreinte écologique ?
- ▶  Des exposés et/ou débats proposés aux élèves sur la thématique des impacts environnementaux du numérique.
- ▶  Observer les métadonnées.

Fiche d'activité élève :

- ▶  Réfléchir à la cybercriminalité.

CE QUE PROPOSE LE PROGRAMME

- ▶ Consulter les métadonnées de fichiers correspondant à des informations différentes et repérer celles collectées par un dispositif et celles renseignées par l'utilisateur.
- ▶ Télécharger des données ouvertes (sous forme d'un fichier au format CSV avec les métadonnées associées), observer les différences de traitements possibles selon le logiciel choisi pour lire le fichier : programme PYTHON, tableur, éditeur de textes ou encore outils spécialisés en ligne.

- ▶ Explorer les données d'un fichier CSV à l'aide d'opérations de tri et de filtre, effectuer des calculs sur ces données, réaliser une visualisation graphique des données.
- ▶ À partir de deux tables de données ayant en commun un descripteur, montrer l'intérêt des deux tables pour éviter les redondances et les anomalies d'insertion et de suppression, réaliser un croisement des données permettant d'obtenir une nouvelle information.
- ▶ Illustrer, par des exemples simples, la consommation énergétique induite par le traitement et le stockage des données.

2 Photographie numérique

2.1 Découvrir la thématique

2.1.1 Anchage dans le réel

A. POINTS-CLEFS

- ▶ Selon une étude de KEYPOINT INTELLIGENCE/INFOTRENDS, environ 400 milliards de photos numériques ont été prises dans le monde en 2011, pour atteindre 1200 milliards de photos en 2017, dont environ 85% réalisées avec le smartphone.
- ▶ Le *selfie*, une photo de soi-même, est un élément-clé de son identité numérique, avec les autres photos de son profil elles constituent un reportage permanent sur ce que nous sommes, dans quel environnement nous vivons.
- ▶ Nous créons quotidiennement de nouveaux usages des photos numériques : nous pouvons par exemple utiliser les photos comme des textos pour transmettre une information instantanée de notre environnement immédiat.
- ▶ La présence permanente d'appareils photos ou de caméras autour de nous change complètement notre relation à l'espace public, où nous y sommes « vus en permanence ».
- ▶ Nous savons qu'une photographie, comme un son ou une vidéo, peut facilement être transformée, soit en étant utilisée hors contexte, soit en étant éditée produisant ainsi un document inexact.

B. MOTS-CLEFS

- ▶ Photographie numérique.
- ▶ Notion de selfie y compris reflet des émotions.

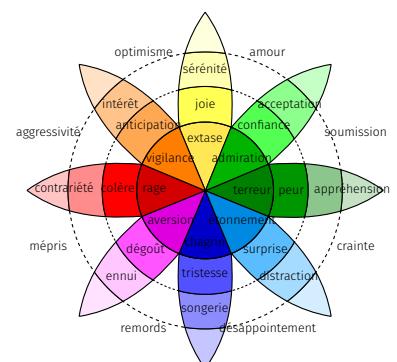
C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Les technologies de la photographie argentique ont eu une évolution très lente, liée aux progrès en optique, mécanique et chimie. Ce n'est plus du tout le cas de l'évolution actuelle, davantage due aux algorithmes qu'à la physique : algorithmes de développement et d'amélioration de l'image brute, algorithmes d'aide à la prise de vue. Cet exemple est caractéristique des façons de procéder de la révolution informatique par rapport aux approches traditionnelles.



VIDÉO 6.3 — Photographie numérique.



Roue des émotions de Robert PLUTCHIK.

La photographie numérique présente un coût marginal très faible et une diffusion par Internet facile et immédiate : chaque jour, des milliards de photos sont prises et partagées.

IMPACTS SUR LES PRATIQUES HUMAINES

La gratuité et l'immédiateté dans la réPLICATION DES IMAGES INTRODUISENT DE NOUVEAUX USAGES DE LA PHOTOGRAPHIE : À LA PHOTOGRAPHIE ARCHIVE (HISTOIRE DE FAMILLE) S'AJOUTENT LA PHOTOGRAPHIE À PARTAGER ET LA PHOTOGRAPHIE UTILITAIRE, PROTHÈSE DE LA MÉMOIRE (PHOTO D'UN TICKET DE CAISSE, D'UNE PRÉSENTATION LORS D'UNE RÉUNION DE TRAVAIL, D'UNE PLACE DE PARKING, ETC.). LES IMAGES S'INTEGRENt À TOUS LES DISPOSITIFS DE COMMUNICATION ET DE PARTAGE, TÉLÉPHONES, WEB ET RÉSEAUX SOCIAUX.

DE NOUVEAUX PROBLÈMES APPARAÎTENT, LIÉS À LA DIFFUSION DE PHOTOS QUI NE DISPARAÎTRONT JAMAIS (NOTION DE DROIT À L'OUBLI), AU TRUCAGE DIFFICILE À DÉTECTOR DES IMAGES, AU PISTAGE DES INDIVIDUS OU À L'OBSOLESCENCE DES SUPPORTS. EST AINSI POSÉE LA QUESTION DE L'ARCHIVAGE DE PHOTOGRAPHIES HISTORIQUES, SCIENTIFIQUES OU CULTURELLES.

2.1.2 Volet historique



Appareil argentique Leica I, 1925.

Bien que la photographie numérique soit relativement récente, la fin du XX^e siècle a été le théâtre de nombreux développements menant à sa création. La première image de Mars a été prise lorsque le Mariner 4 l'a survolé le 15 juillet 1965 avec un système de caméra conçu par la NASA/JPL. Il utilisait un tube de caméra vidéo, suivi d'un numériseur, plutôt que d'une mosaïque d'éléments capteurs à l'état solide. Cela a produit une image numérique qui a été enregistrée sur bande pour une transmission lente ultérieure vers la Terre.

La véritable histoire de la photographie numérique telle que nous la connaissons a commencé dans les années 1950. En 1951, les premiers signaux numériques ont été enregistrés sur bande magnétique via le premier magnétoscope. Six ans plus tard, la première image numérique a été produite par un ordinateur par Russell KIRSCH. C'était une image de son fils. À la fin des années 1960, Willard S. BOYLE et George E. SMITH, deux physiciens de BELL LABS Inc., ont inventé le dispositif à couplage de charge (CCD), un circuit à semi-conducteur utilisé par la suite dans les premiers caméscopes numériques pour la télédiffusion. Leur invention a été reconnue par un prix Nobel de physique en 2009.

La première photographie numérique couleur publiée a été produite en 1972 par Michael Francis THOMPSETT à l'aide de la technologie de capteur CCD. Elle a paru sur la couverture du magazine ELECTRONICS. C'était une photo de sa femme, Margaret THOMPSETT. Le Cromemco Cyclops, un appareil photo numérique développé comme produit commercial et interfacé avec un micro-ordinateur, a été présenté dans le numéro de février 1975 du magazine POPULAR ELECTRONICS. Il utilisait la technologie des semi-conducteurs à oxyde de métal (MOS) pour son capteur.

Le premier appareil photo numérique autonome (portable) a été créé plus tard en 1975 par Steven SASSON d'EASTMAN KODAK. La caméra de SASSON utilisait des puces de capteur d'image CCD développées par FAIRCHILD SEMICONDUCTOR en 1973. La caméra pesait 3,6 kg, enregistrait les images en noir et blanc sur une cassette, avait une définition de 0,01 mégapixels (10 000 pixels) et prenait 23 secondes pour capturer sa première image en décembre 1975.

Le premier appareil numérique largement disponible dans le commerce est le DYCAM Model 1 de 1990, également vendu comme LOGITECH FOTOMAN. Il utilisait un capteur d'image CCD, stockait des images sous forme numérique et était connecté directement à un ordinateur pour le téléchargement d'images. Proposés à un prix dérisoire, vers le milieu des années 1990, en raison des progrès technologiques, les appareils photo numériques sont rapidement devenus des produits grand public.

L'avènement de la photographie numérique a également fait place à des changements culturels. Contrairement à la photographie traditionnelle, les pièces sombres et les produits chimiques dangereux ne sont plus nécessaires pour la post-production d'une image : les images peuvent désormais être traitées et améliorées chez soi depuis un ordinateur. Cela a permis aux photographes d'être plus créatifs dans leurs techniques de traitement et d'édition. À mesure que le domaine gagnait en popularité, les types de photographie numérique et de photographes se diversifiaient. La photographie numérique a propulsé la photographie elle-même d'un petit cercle assez élitaire vers un cercle englobant de nombreuses personnes.

L'appareil photographique des téléphones a également contribué à populariser la photographie numérique, ainsi que l'Internet et les médias sociaux. Les premiers téléphones cellulaires dotés d'appareils photo numériques intégrés ont été fabriqués en 2000 par SHARP et SAMSUNG. Petits, pratiques et faciles à utiliser, les téléphones avec appareil photo ont rendu la photographie numérique omniprésente dans la vie quotidienne du grand public.

Sources : *Digital photography* W ; *History of the camera* W.

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

- ▶ 1826 : naissance de la photographie argentique.
- ▶ 1900 : photographie en couleurs. Après la dernière guerre mondiale, généralisation du format 24 x 36 et de la visée reflex.
- ▶ 1969 : arrivée des premiers capteurs CCD (Charge Coupled Device).
- ▶ 1975 : apparition des premiers appareils numériques.
- ▶ 2007 : arrivée du smartphone.



Appareil numérique SLR – Single-Lens Reflex – Minolta, 1995.

2.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Que ce soit à travers une caméra numérique ou dans notre cerveau, l'image que nous capturons est le résultat de transformations complexes qui permettent d'encoder l'information visuelle et de reconstruire l'image obtenue.
- ▶ Dans les appareils photographiques numériques, les aberrations optiques géométriques et chromatiques sont corrigées par interpolation pour reconstituer les couleurs, par filtrage pour augmenter le rapport signal sur bruit, par accentuation pour augmenter la netteté, par correction des couleurs (balance des blancs), etc.
- ▶ On dispose aussi de mécanismes qui combinent une rafale de plusieurs images pour augmenter le champ de vue, la profondeur de champ, la luminosité ou la clarté, etc.
- ▶ Le traitement d'image se fonde sur des calculs faits sur les valeurs des pixels (par exemple la soustraction entre deux images consécutives pour estimer un mouvement) et permet de transformer le capteur en outil de mesure.

- ▶ En imagerie médicale ou satellitaire, et dans d'autres domaines, on manipule des images multi-dimensionnelles : par exemple 3D + T (des vidéos volumiques) et les valeurs des pixels peuvent correspondre à d'autres mesures physiques que la lumière visible.
- ▶ Aujourd'hui et depuis le début du XX^e siècle, l'image, au-delà de la photographie, est devenue un support d'investigation visuelle de nombreux phénomènes physiques. C'est le cas en médecine, avec l'imagerie médicale mais également l'imagerie biologique ; en géologie, avec l'imagerie du sous-sol et l'imagerie satellitaire ; en physique particulière ou des plasmas. Une preuve en est le nombre important de prix Nobel décernés pour l'invention de nouveaux modes d'imagerie (le dernier datant de 2017 pour la *cryo-microscopie électronique*).
- ▶ Le traitement d'image est donc un outil d'analyse de données que tout scientifique peut être amené à employer dans son travail.
- ▶ On utilise par abus de langage le terme de résolution pour définition.

B. MOTS-CLEFS

- ▶ Traitement d'image.
- ▶ Vision par ordinateur.
- ▶ Données EXIF — *Exchangeable Image File Format*.

C. CE QUE DIT LE PROGRAMME

DONNÉES ET INFORMATION

En entrée, le capteur est formé de *photosites* en matrice de petits carrés de quatre photosites, deux verts, un bleu et un rouge, correspondant à la répartition des cônes de la rétine. La *définition du capteur* se mesure en millions de photosites.

En sortie, l'image est formée de *pixels* colorés homogènes, représentés par trois nombres RVB (rouge, vert, bleu). La *définition de l'image* se compte en mégapixels ; elle n'est pas forcément égale à celle du capteur. La *profondeur de couleur* est en général de 8 bits par pixel et par couleur pour l'image finale.

Des *métadonnées* sont stockées dans les fichiers images sous format EXIF (*Exchangeable Image File Format*) : modèle de l'appareil, objectif, vitesse, diaphragme, distance de mise au point, auteur, copyright, localisation, etc.

Les couleurs peuvent être représentées dans différents systèmes : RVB, TSL (teinte, saturation, lumière), avec des formules empiriques [rigoureuses issues de l'observation de la perception humaine] de passage d'un modèle à l'autre. On distingue différents formats des fichiers images, compressés ou non, avec ou sans perte : RAW, BMP, TIFF, JPEG.

ALGORITHMES ET PROGRAMMES

Des algorithmes permettent de traiter toutes les lumières, d'effectuer une retouche facile, avec une qualité maintenant bien supérieure à l'argentique. Avec l'arrivée du téléphone mobile, des algorithmes de fusion d'images permettent de concilier une excellente qualité avec un capteur et un objectif minuscules.

De nombreux algorithmes sophistiqués sont utilisés dans les appareils de photographie numérique :

- Lors de la prise de vue : calcul de l'exposition, mise au point, stabilisation par le capteur et/ou l'objectif, le tout en automatique ou manuel assisté, *focus-peaking* (scintille-

- ment des contours nets), prise en rafales rapides d'images multiples avant et après appui sur le déclencheur.
- Lors du développement de l'image issue du capteur en une image pixellisée : gestion de la lumière et du contraste, balance des blancs, netteté, débouchage des ombres, correction des distorsions ou des aberrations optiques.
 - Après le développement : compression du fichier (TIFF sans perte, JPEG avec perte).
 - En utilisant la fusion d'images : réduction du bruit et amélioration de la netteté, panoramas, HDR (High Dynamic Range), super-résolution par micro-décalages du capteur, *focus stacking* pour étendre la netteté avec plusieurs mises au point successives, réduction du bruit et amélioration de la netteté.

Certains appareils peuvent augmenter leurs fonctionnalités par téléchargement de nouveaux logiciels.

MACHINES

Comme les algorithmes de prise de vue et de développement demandent beaucoup de calcul, les appareils embarquent plusieurs processeurs, généraux ou spécialisés.

Les algorithmes prennent le relais des capteurs physiques en calculant les pixels de l'image finale : ils compensent par exemple les distorsions des lentilles. Des algorithmes permettent également de commander la mise au point et l'exposition automatique, ainsi que de compenser le bougé de l'utilisateur.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

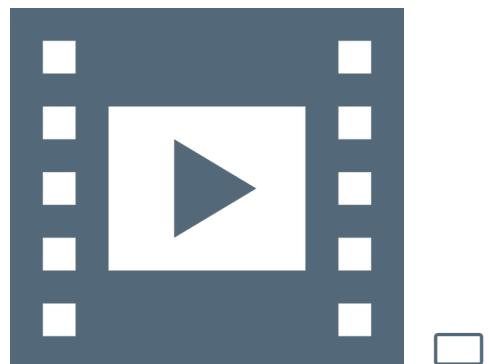
- Une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours.
- Des exemples concrets de ce qui est étudié ou produit aujourd'hui dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant actuellement dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur la *photographie numérique* a eu lieu le 7 mars 2019 à l'Université de Nantes, elle a été retransmise en ligne et est maintenant disponible sur YOUTUBE.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ Pour les enseignants : la conférence de Laurent CHENO sur la photographie numérique, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ *Histoire du traitement d'images* Isabelle BELIN, INTERSTICES, 25 février 2004.



VIDÉO 6.4 — Photographie numérique,
Jean-Gabriel AUBERT.

- ▶ Vision artificielle, Partie I et Partie II, Binaire, octobre/novembre 2017.
- ▶ Sur la psychologie de la vision, le site du Cerveau à tous les niveaux et quelques exemples d'illusions d'optiques expliquées, parmi d'autres.
- ▶ Sur la réalité augmentée un article de vulgarisation et un livre de référence, discuté et présenté.
- ▶ Une sélection d'articles du site INTERSTICES.

Créer son cours

- ▶ De la subjectivité d'une observation avec l'exemple des canaux martiens. Qu'est-ce qu'une forme ? Un système d'analyse d'image informatique doit-il toujours imiter le fonctionnement du cerveau ? Comment décrire une image ?
- ▶ Sur la chronophotographie, une des premières utilisations de la photographie à des fins scientifiques, avec une fiche d'activités et une proposition de séance.
- ▶ Sur l'imagerie médicale, un documentaire vidéo et un cours d'initiation sur le scanner tomodensitométrique, au-delà de l'article WIKIPEDIA sur la tomographie.
- ▶ Une vidéo rigolote sur la photo argentique et numérique qui fait partie de la série JACQUES a dit et reste très attractives, malgré quelques approximations scientifiques, intéressantes du reste à faire débusquer par les élèves.

2.2 Réaliser des activités

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ Principe des images numériques.
- ▶ Principe des capteurs photographiques.
- ▶ Écrire des programmes PYTHON (bibliothèque Pillow).
- ▶ Étude des métadonnées EXIF d'un appareil photo numérique.
- ▶ Différents traitements d'une image en PYTHON et en neuf fiches.
- ▶ Utiliser des photos pour mesurer une hauteur.
- ▶ Les images téléphonées (du codage des images).
- ▶ Mesurer la taille d'un objet distant.
- ▶ Prendre conscience du coût énergétique des données.
- ▶ Découverte du principe de la tomographie.
- ▶ Manipuler les pixels d'une image numérique en PYTHON.
- ▶ Reconnaissance faciale.

Fiche d'activité élève :

- ▶ Découvrir une image numérique..
- ▶ Travailler sur les métadonnées EXIF..

CE QUE PROPOSE LE PROGRAMME

- ▶ Programmer un algorithme de passage d'une image couleur à une image en niveaux de gris : par moyenne des pixels RVB ou par changement de modèle de représentation (du RVB au TSL, mise de la saturation à zéro, retour au RVB).
- ▶ Programmer un algorithme de passage au négatif d'une image.
- ▶ Programmer un algorithme d'extraction de contours par comparaison entre pixels voisins et utilisation d'un seuil.

- ▶ Utiliser un logiciel de retouche afin de modifier les courbes de luminosité, de contraste, de couleur d'une photographie.

3 Informatique embarquée et objets connectés

3.1 Découvrir la thématique

3.1.1 Ancrage dans le réel

A. POINTS-CLEFS

- ▶ Il y a aujourd'hui plus de processeurs dans des objets connectés ou enfouis dans les objets du quotidien que dans nos machines usuelles (ordinateur, tablette et smartphone).
- ▶ Pour utiliser ces objets connectés un enjeu majeur est celui des interfaces, interface humain-machine, de façon à ce qu'ils soient entièrement contrôlables, de manière compréhensible par les utilisateurs.
- ▶ La possibilité d'introduire des algorithmes au sein des objets de notre quotidien offre de nouvelles possibilités inédites et pose des problèmes de sûreté et de sécurité encore irrésolus.
- ▶ C'est un enjeu de société de savoir si cette nouvelle révolution technologique sera pilotée par la recherche de nouveaux marchés commerciaux ou par la couverture de vrais besoins de notre société. C'est par exemple le cas au niveau de ce qu'on appelle les *Smart Cities* (ou villes intelligentes).

B. MOTS-CLEFS

- ▶ Internet des objets..
- ▶ Sûreté..
- ▶ Sécurité..
- ▶ Interface humain-machine..

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Embarquer l'informatique dans les objets a beaucoup d'avantages : simplifier leur fonctionnement, leur donner plus de possibilités d'usage et de sûreté et, leur permettre d'intégrer de nouvelles possibilités à matériel constant par simple modification de leur logiciel.

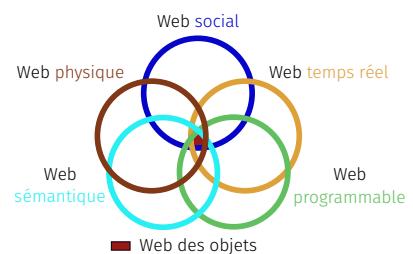
Après avoir transformé les chaînes de montage des automobiles et les avions dans les années quatre-vingt-dix, l'informatique intervient maintenant dans des domaines toujours plus nombreux : automobile, réseau ferroviaire et transports urbains, domotique, robotique, loisirs, etc., conduisant à un nouvel Internet des objets.

Pour les avions par exemple, l'informatique gère le vol en commandant finement des servomoteurs électriques, plus légers et plus fiables que les vérins hydrauliques, les réacteurs, la navigation et le pilotage automatique et, permet l'atterrissement automatique par temps de brouillard. Elle a eu un impact décisif sur l'amélioration de la sécurité aérienne.

Les objets informatisés avaient autrefois des *interfaces homme-machine* (IHM) dédiées, souvent dépendantes d'une liaison filaire directe. Mais les technologies du Web intégrées au télé-



VIDÉO 6.5 — Objets connectés.



Internet des objets.

phone portable autorisent maintenant d'y rassembler les interfaces des objets du quotidien, ce qui en simplifie et uniformise l'usage. Les objets informatisés deviennent ainsi connectés.

IMPACTS SUR LES PRATIQUES HUMAINES

L'impact de l'informatisation des objets devient considérable, surtout depuis que leurs interfaces s'unifient. Le but est de fabriquer des machines d'utilisation facile permettant des fonctionnalités améliorées, voire complètement nouvelles comme la voiture autonome. Celle-ci utilise à la fois des techniques de systèmes embarqués pour son fonctionnement et sa navigation et de l'intelligence artificielle pour l'analyse en temps-réel de l'environnement à l'aide de capteurs divers et variés (caméras, radars, lidars, etc.).

Comme l'informatique embarquée interagit avec le monde physique en exposant quelquefois des vies humaines ou des équipements critiques (réseaux électriques par exemple), elle est soumise à de fortes contraintes de sûreté (absence d'erreurs) et de sécurité (résistance aux attaques). En avionique, ferroviaire ou autres applications critiques, des processus lourds de certification externe sont utilisés. Cependant dans beaucoup de systèmes embarqués moins critiques, la sécurité reste souvent un point faible et les objets connectés sont de plus en plus utilisés comme robots pour lancer des attaques sur internet.

NOTE

Entre un robot et autre objet connecté, le principe est le même, les deux classes d'objets comprennent trois types d'éléments constitutifs : des capteurs, des actionneurs et un programme qui permet à l'objet d'interagir avec l'environnement. On définit par ailleurs la *domotique* comme de l'informatique associée à des objets connectés du quotidien associés aux bâtiments, « l'habitat intelligent ».

3.1.2 Volet historique

La notion d'Internet des objets résulte de la convergence de plusieurs technologies, de l'analyse en temps réel, de l'apprentissage automatique, des capteurs de produits de base et des systèmes intégrés. Les domaines traditionnels des systèmes embarqués, des réseaux de capteurs sans fil, des systèmes de contrôle, de l'automatisation (y compris l'automatisation de la maison et des bâtiments) et autres contribuent tous à l'activation de l'Internet des objets.

Le concept de réseau d'appareils intelligents a été discuté dès 1982 : un distributeur automatique de boisson modifié de l'Université Carnegie Mellon devenant le premier appareil connecté à Internet, capable de signaler ses stocks et de savoir si les boissons fraîchement chargées étaient froides ou non. L'article de 1991 de Mark WEISER sur l'informatique omniprésente, « L'ordinateur du XXI^e siècle », ainsi que des institutions académiques telles que UBICOMP et PERCOM ont produit la vision contemporaine de l'Internet des Objets. En 1994, Reza RAJI a décrit le concept dans IEEE Spectrum comme « un [déplacement] de petits paquets de données vers un grand ensemble de noeuds, afin d'intégrer et d'automatiser tout, des appareils ménagers aux usines entières ». Entre 1993 et 1997, plusieurs sociétés ont proposé des solutions. Le domaine a pris de l'ampleur lorsque Bill Joy a envisagé la communication D2D (*Device to Device*) dans le cadre de son cadre « Six Webs », présenté au Forum économique mondial de Davos en 1999.

Le terme « Internet des objets » a probablement été inventé par Kevin ASHTON de PROCTER & GAMBLE, futur centre d'identification automatique du MIT, en 1999, bien qu'il préfère l'expression « Internet pour les objets ». Un article de recherche mentionnant l'Internet des objets a été soumis à la conférence pour les chercheurs nordiques en Norvège, en juin 2002, précédée d'un article publié en finnois en janvier

2002. L'implémentation décrite ici a été développée par Kary FRÄMLING et son équipe de l'Université technologique de Helsinki et correspond plus étroitement à l'infrastructure moderne, c'est-à-dire une infrastructure de système d'information permettant la mise en œuvre d'objets connectés intelligents.

Définissant l'Internet des objets comme « simplement le moment où plus de choses ou d'objets étaient connectés à Internet que de personnes », CISCO SYSTEMS a estimé que l'Internet des objets était « né » entre 2008 et 2009, avec un ratio croissant d'objets par personnes de 0,08 en 2003 à 1,84 en 2010.

Sources : [Internet of things W](#); [Internet des objets W](#).

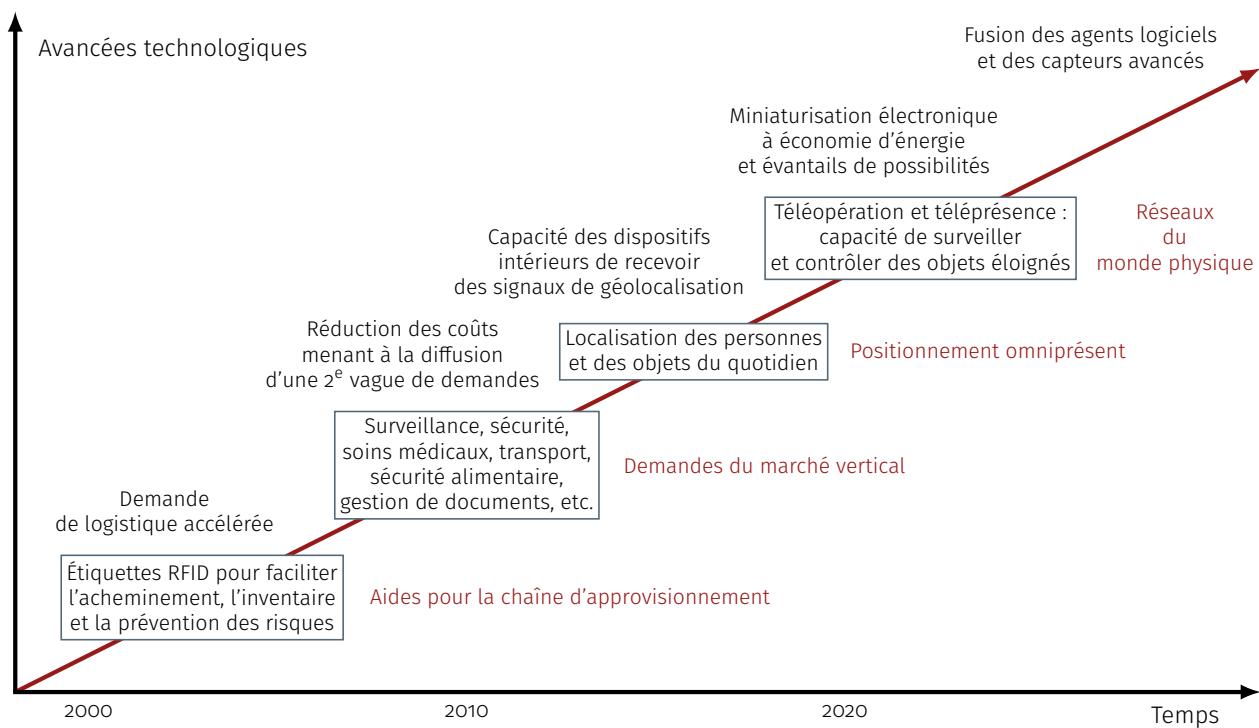


FIGURE 6.1 – Historique de la technologie : connectivité des choses (d'après SRI Consulting Business Intelligence).

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

- ▶ 1967 : premier système embarqué de guidage lors de la mission lunaire APOLLO.
 - ▶ 1971 : premier processeur produit par INTEL.
 - ▶ 1984 : sortie de l'AIRBUS 320, premier avion équipé de commandes électriques informatisées.
 - ▶ 1998 : mise en service du métro informatisé sans conducteur MÉTEOR (ligne 14 à Paris).
 - ▶ 1999 : introduction de l'expression « Internet des objets » par Kevin ASHTON.
 - ▶ 2007 : arrivée du smartphone.
- On estime à 50 milliards le nombre d'objets connectés en 2020.

3.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Pour programmer des objets connectés ou embarqués à l'intérieur d'un autre système, le paradigme de programmation de base est évé-

nementiel : ce sont les entrées des capteurs qui déclenchent des fonctions du code qui réagit et ajuste les sorties et son état interne.

- ▶ Pour assurer la sûreté informatique, on travaille à la fois au niveau de la conception et de la vérification : la conception avec des langages de modélisation du système utilisé (comme UML) et la vérification avec des méthodes formelles qui travaillent sur la sémantique du programme pour en prédire le fonctionnement.
- ▶ Mais ces méthodes prévisionnelles ne sont jamais complètes et c'est l'expérimentation qui permet de compléter la validation de tels systèmes.
- ▶ Pour assurer la sécurité des systèmes informatiques on utilise les mêmes méthodes que pour protéger un système classique : authentication des personnes ou des algorithmes qui accèdent à l'objet et chiffrage des données.

B. MOTS-CLEFS

- ▶ Programmation événementielle et programmation synchrone.
- ▶ Internet des objets.
- ▶ Sûreté.
- ▶ Sécurité.
- ▶ Interface humain-machine.

C. CE QUE DIT LE PROGRAMME

DONNÉES ET INFORMATION

Dans les systèmes *informatiques embarqués*, l'information provient soit des IHM soit des capteurs, pour contrôler automatiquement ou manuellement le fonctionnement physique par des actionneurs et transmettre des informations aux utilisateurs. Le flux d'informations à travers les IHM permet ainsi une interaction continue entre l'homme et la machine.

ALGORITHMES ET PROGRAMMES

Le développement des logiciels embarqués est délicat, car il pose souvent des questions de temps-réel, c'est-à-dire de respect de temps de réponse imposé. Ceci conduit à des méthodes de programmation spécifiques.

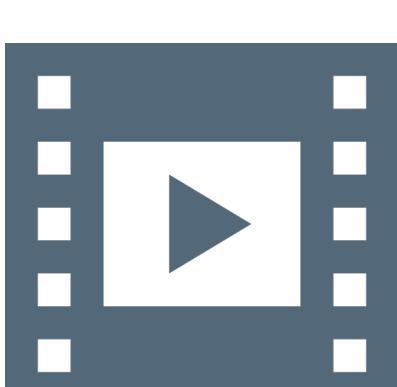
MACHINES

Les microprocesseurs sont beaucoup plus nombreux dans les objets que dans les ordinateurs et téléphones, mais ils sont souvent plus petits, moins chers et moins rapides. Les *capteurs* et *actionneurs* reposent sur des technologies physiques et électroniques variées, allant quelquefois vers l'électronique de puissance. Un problème essentiel est la réduction de la consommation électrique, surtout pour les appareils sur pile.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

- Une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général,



VIDÉO 6.6 – Objets connectés, Jean-Philippe ENEAU.

histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours.

- Des exemples concrets de ce qui est étudié ou produit aujourd’hui dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant actuellement dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur l’Informatique embarquée et les objets connectés a eu lieu le 21 mars 2019 dans le cadre du projet CLASS’CODE à l’Université de Nantes.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ Introduction aux interfaces humain-machine.
- ▶ Cours introductif à l’Internet des objets.
- ▶ Pour les enseignants : la conférence de Pascale COSTA l’informatique embarquée et objets connectés et aussi localisation, cartographie et mobilité, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Une sélection d’articles du site INTERSTICES.

Créer son cours

- ▶ Fiche de connaissance sur les IHM par Nicolas TOURREAU, validée sur Sciences numériques technologie.
- ▶ Qu'est-ce que la sécurité et la sûreté informatique ?
- ▶ L’Internet des objets, livre ouvert consultable en ligne.
- ▶ Revue en ligne avec beaucoup de pistes librement réutilisables concernant le RASPBERRY-PI.
- ▶ Démonter un ordinateur.
- ▶ Monter un ordinateur.
- ▶ Framework pour travailler sur ARDUINO en PYTHON. Possibilité d’utiliser le GUI KIVY pour créer une IHM qui pilote ARDUINO.

3.2 Réaliser des activités

EXEMPLE D’ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ Introduction aux interfaces « homme - machine ».
- ▶ Introduction aux principes de la voiture autonome.
- ▶ Découverte de la robotique et réflexion sur son impact au travail.
- ▶ Programmer un robot pédagogique (capteurs et actionneurs).

Fiche d’activité élève :

- ▶ Smartphone et APP INVENTOR.

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l’URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

CE QUE PROPOSE LE PROGRAMME

- ▶ Identifier les évolutions apportées par les algorithmes au contrôle des freins et du moteur d’une automobile, ou à celui de l’assistance au pédalage d’un vélo électrique.
- ▶ Réaliser une IHM pouvant piloter deux ou trois actionneurs et acquérir les données d’un ou deux capteurs.
- ▶ Gérer des entrées/sorties à travers les ports du système.
- ▶ Utiliser un tableau de correspondance entre caractères envoyés ou reçus et commandes physiques (exemple : le moteur

A est piloté à 50% de sa vitesse maximale lorsque le robot reçoit la chaîne de caractères « A50 »).

4 Localisation et cartographie

4.1 Découvrir la thématique

4.1.1 Ancre dans le réel

A. POINTS-CLEFS

- ▶ Une carte est un outil de représentation d'informations hiérarchisées liées à une localisation. Les cartes permettent de combiner différentes informations, pour visualiser des réalités aussi diverses que le développement d'un territoire ou la propagation d'une maladie.
- ▶ Le fait que nous soyons aujourd'hui localisé·e·s en permanence a des conséquences importantes sur notre vie quotidienne : cela permet d'être aidé en cas de situation critique, mais aussi d'être repéré à tout moment qu'on le souhaite ou non.
- ▶ La localisation est incontournable, et pas uniquement au niveau de notre *smartphone* : dès que nous interagissons avec un système numérique, nous sommes localisés, et le fait de ne plus se faire localiser à un moment donné est en soi une information.

B. MOTS-CLEFS

- ▶ Géolocalisation.
- ▶ Assistant de navigation.
- ▶ Géomatique.

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

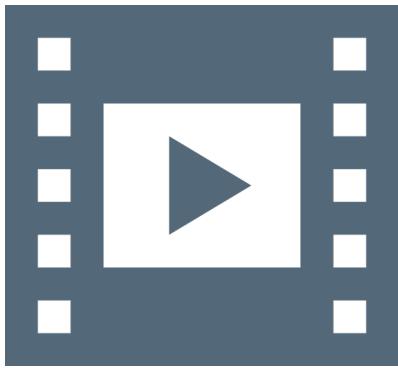
La cartographie est essentielle pour beaucoup d'activités : agriculture, urbanisme, transports, loisirs, etc. Elle a été révolutionnée par l'arrivée des cartes numériques accessibles depuis les ordinateurs, tablettes et téléphones, bien plus souples à l'usage que les cartes papier.

Les *cartes numériques* rassemblent toutes les échelles et permettent de montrer différents aspects de la région visualisée sur une seule carte. Les algorithmes de recherche permettent de retrouver sur la carte les endroits en donnant simplement leur nom, et de calculer des itinéraires entre points selon des modes de transports variés.

IMPACTS SUR LES PRATIQUES HUMAINES

Les cartes numériques, accessibles depuis un téléphone, remplacent progressivement les cartes sur papier. Leurs interfaces permettent d'accéder commodément à de nombreux types d'information. Couplé aux algorithmes de calculs d'itinéraires, le GPS est utilisé systématiquement pour les transports, l'agriculture, la randonnée, la navigation à voile, etc.

Le maintien à jour des cartes numériques est un problème difficile qui demande beaucoup de ressources au plan mondial. Les erreurs dans les cartes, inévitables à cause de l'énorme quantité



VIDÉO 6.7 — Localisation et cartographie.

d'informations à collecter et à traiter, peuvent avoir des conséquences dramatiques.

Par ailleurs, de nombreuses applications ont accès à la localisation dans un téléphone, ce qui leur permet d'envoyer des publicités non désirées, de suivre vos itinéraires, ou de localiser une personne. Enfin, le GPS n'est pas toujours sûr, car facile à brouiller à l'aide d'appareils simples.

4.1.2 Volet historique

Les premières cartes connues représentent les étoiles et non la terre. Des points datés de 16 500 BC, trouvés sur les murs de la grotte de Lascaux montrent une partie du ciel nocturne, incluant trois des étoiles les plus brillantes, Véga, Deneb, et Altaïr (le Triangle d'été), ainsi que l'amas d'étoiles les Pléïades. La grotte du Castillo en Espagne possède également une carte de la Couronne boréale datée de 12 000 BC.

Depuis l'Antiquité, jusqu'au milieu du XVI^e siècle, les relevés sont issus de témoignages. Les premières mises en forme « scientifiques » datent du II^e siècle après notre ère avec la cartographie de PTOLÉMÉE (150), où celui-ci énonce quelques précautions pour dessiner une carte sur un plan. Par la suite les relevés sont assemblés par des cartographes experts et alimentés par les premiers essais de statistiques rassemblés par les représentants de l'autorité (époque antique romaine, des moines savants du Moyen Âge puis des grandes découvertes). Les supports utilisés — notamment les cartes marines — sont grossières car elles ne respectent ni les angles, ni les distances réelles.

Le véritable développement intervient avec l'amélioration des outils de mesure mis au point par la géodésie et les géomètres, ainsi que l'amélioration des registres de tous types, devenant de larges sources statistiques. Aussi, les traits et les données s'affinent. Ainsi les premières mesures astronomiques (longitudes et latitudes) de localités de la France effectuées par Jean PICARD commencent en 1671, permettent à LA HIRE d'établir en 1682 une carte corrigée qui affine le contour du littoral et réduit considérablement les vraies proportions de la France.

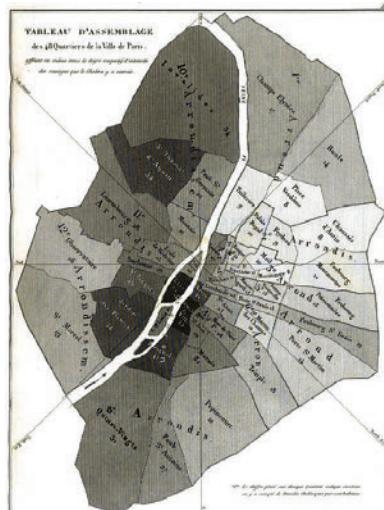
L'une des premières applications connues de l'analyse cartographique concernait le domaine de l'épidémiologie avec, en 1832, la publication du « Rapport sur la marche et les effets du choléra dans Paris et le département de la Seine », rédigé par le géographe français Charles PICQUET. Ce dernier a représenté les quarante-huit districts de la ville de Paris. Il a utilisé un système de coloris dégradé en fonction du pourcentage de décès par le choléra pour mille habitants.

L'utilisation des engins aéronautiques (dirigeables, avions, hélicoptères) à partir du début du XX^e siècle permet d'affiner et de mettre à jour plus rapidement la couverture cartographique, mais pour des espaces à chaque fois relativement limités et concernant presque uniquement les terres émergées. Dans la dernière partie du XX^e siècle, un pas technique majeur est franchi avec l'utilisation et le traitement numérique des ondes émises par des satellites : les contours terrestres sont alors pour la première fois photographiés depuis le ciel. Des cartographies du fond des océans ou des zones inaccessibles deviennent beaucoup plus précises. La cartographie complète de la Lune et de Mars est réalisée grâce aux satellites d'exploration ou sondes spatiales.

Grâce à des avancées mathématiques et informatiques, on obtient avec facilité toujours plus de projections planes innovantes, qui doivent toujours arbitrer entre conservation des parallèles, des aires, et des longueurs. Des cartes amorphes (cartogrammes) sont aussi apparues.



Carte/rocher de Bedolina.



Quartiers de Paris.



Carte topographique de l'île de Corfou, données satellitaires.

Le support digital permet la duplication, le transfert à bas coût, et le traitement automatisé (par exemple le projet [Corine Land Cover](#) pour l'aménagement du territoire).

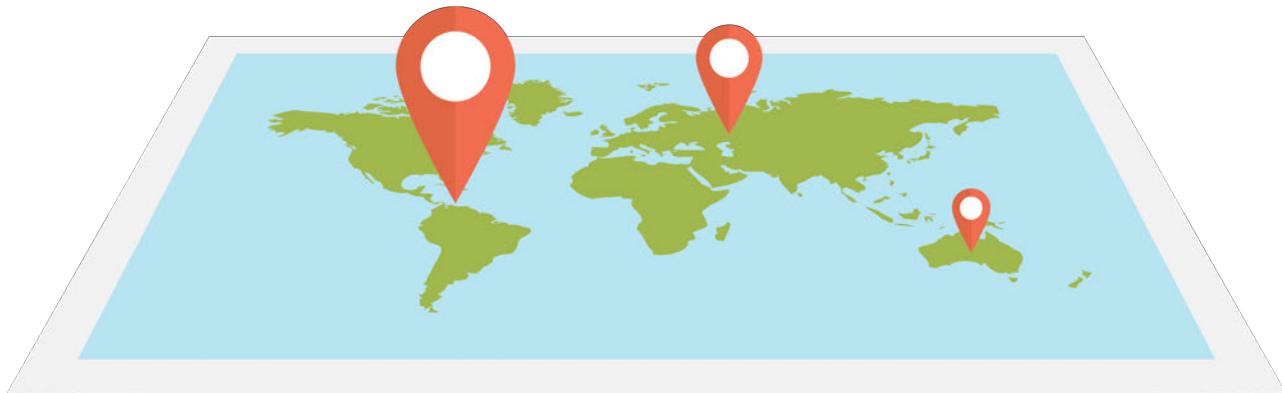
Un autre apport du numérique concerne la capacité à mettre en relation et diffuser des documents d'intérêt cartographique du monde entier et de toutes les époques, *via* Internet. En France, un Consortium intitulé « Cartes et photographies pour les géographes » vise à faciliter le tournant numérique de la recherche en sciences humaines et sociales, en développant le réseau de portails cartographiques et de plateformes de diffusion de données et métadonnées peu à peu mis en place, essentiellement par de grandes institutions, pour « généraliser l'accès à d'autres fonds pertinents et améliorer la diffusion des images géographiques » afin de « rendre accessibles, consultables et mobili-sables des données cartographiques et photographiques nombreuses et éparses, qui constituent des fonds de laboratoires de recherche, de bibliothèques remarquables ou des fonds de chercheurs... ». Un mouvement de libération des données ([Open data](#)) est également en cours qui, avec des organisations comme [OPENSTREETMAP](#), devrait permettre de largement développer la cartographie historique et collaborative.

Source : Cartographie [W](#); Système d'information géographique [W](#).

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

Les cartes ont été systématiquement numérisées à la fin du XX^e siècle. Le principal instrument de localisation, GPS (*Global Positioning System*), a été conçu par l'armée américaine dans les années soixante. Le premier satellite GPS fut lancé en 1978. Il y en a actuellement une trentaine, de sorte qu'à tout moment quatre à six satellites au moins sont visibles depuis tout point de la Terre. Couplé aux cartes numériques, le système GPS permet de se situer. Il n'est pas toujours efficace en ville, et peut être complété par d'autres moyens de localisation comme la détection de bornes Wi-Fi proches. D'autres systèmes plus précis, dont GALILEO, sont en cours de déploiement.



4.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ La localisation par satellite impose de recevoir quatre mesures, car il y a quatre inconnues : trois spatiales et une temporelle.

- ▶ Le calcul d'itinéraire correspond à un algorithme de calcul de plus court chemin (théorie des graphes), il est utilisé dans d'autres applications qui cherchent à trouver une suite d'étapes pour passer d'un état initial à un état final.
- ▶ Une carte numérique n'est pas une image matricielle, mais une image vectorielle ce qui permet de la représenter à n'importe quelle échelle, et d'y adosser toutes sortes d'informations.

B. MOTS-CLEFS

- ▶ Système d'information géographique (SIG).
- ▶ Système de positionnement par satellite.

C. CE QUE DIT LE PROGRAMME

DONNÉES ET INFORMATION

Les informations des cartes numériques proviennent de nombreuses sources : services géographiques des États, photos prises par des satellites, avions ou voitures, données fournies par les utilisateurs, etc. Ces informations sont de natures diverses : topographiques, géologiques, photographiques, liées aux transports, à l'activité industrielle ou touristique, etc. Des projets collaboratifs comme OPENSTREETMAP permettent à chaque utilisateur d'ajouter des informations à une carte en libre accès, qui deviennent alors visibles par tous les utilisateurs.

Un satellite GPS contient des horloges atomiques mesurant le temps à une très grande précision et envoyant régulièrement des messages contenant cette heure. Chaque message se propageant à la vitesse de la lumière, le récepteur peut calculer sa distance au satellite. On peut en déduire sa position en suivant plusieurs satellites, ce que fait automatiquement le récepteur GPS.

ALGORITHMES ET PROGRAMMES

Les algorithmes cartographiques concernent principalement l'affichage sélectif d'informations variées et le *calcul d'itinéraires*. L'affichage est paramétré par les informations à montrer, que l'on peut choisir par simples clics. Une difficulté est liée au mélange d'informations de types différents lors des changements d'échelle : les graphismes peuvent être très différents et beaucoup d'informations doivent être supprimées pour les grandes échelles, mais une route doit être représentée avec à peu près la même largeur, quelle que soit l'échelle.

Les récepteurs GPS fournissent la localisation sous une forme normalisée facilement décodable, par exemple selon le protocole NMEA 0183 (*National Marine Electronics Association*), ou directement dans les métadonnées EXIF d'une photo. La localisation et les cartes se couplent dans le suivi permanent de la position sur la carte ou sur un itinéraire précalculé.

MACHINES

Les machines utilisées pour la cartographie sont surtout les ordinateurs, tablettes et téléphones classiques équipés d'une application *ad hoc*. Les récepteurs GPS spécialisés restent importants pour la navigation maritime ou aérienne, mais ceux pour la randonnée pédestre sont en voie de disparition, supplantés par les téléphones.

L'heure fournie par le GPS sert aussi de base pour la synchronisation précise des horloges internes des ordinateurs connectés à internet, ce qui est très important pour tous les échanges d'informations.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

- Une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours.
- Des exemples concrets de ce qui est étudié ou produit aujourd'hui dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant actuellement dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur la *localisation, cartographie et mobilité* a eu lieu le 14 mars 2019 dans le cadre du projet CLASS'CODE à l'Université de Nantes. Elle a été retransmise en ligne et est maintenant disponible sur YOUTUBE.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ Une vidéo pour découvrir la géomatique (partageable avec les élèves).
- ▶ Une vidéo explicative du CNES sur le fonctionnement de Galileo (partageable avec les élèves).
- ▶ Pour les enseignants : la conférence de Pascale COSTA l'informatique embarquée et objets connectés et aussi localisation, cartographie et mobilité, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Une sélection d'articles du site INTERSTICES.

Créer son cours

- ▶ Fiche de connaissance sur le GPS par Nicolas TOURREAU, validée sur Sciences numériques technologie.
- ▶ Fiche de connaissance sur les cartes numériques par Nicolas TOURREAU, validée sur Sciences numériques technologie.
- ▶ La ressource librement utilisable « isoloir » propose une activité sur la localisation à faire en classe en semi-autonomie, ces documents sont librement réutilisables.
- ▶ Le portail de vidéos de Géodésie générale (Cours ENSG - Serge Botton – décembre 2013).
- ▶ Une approche calculatoire du positionnement d'un récepteur GNSS à partir des observations de code C/A et de phase de trois constellations, sous MATLAB ou OCTAVE, Clément FONTAINE et Jacques BEILIN.
- ▶ Cours sur l'utilisation de la Théorie des graphes en géomatique avec les contenus suivants : définitions de base, notions utiles, Stéphane PELLE.
- ▶ Une vidéo rigolote sur la géolocalisation qui fait partie de la série JACQUES a dit et reste très attractive, malgré quelques ap-



VIDÉO 6.8 — *Localisation cartographie et mobilité*, Daniel BOURREAU.



proximations scientifiques, intéressantes du reste à faire débusquer par les élèves.

- ▶ Articles de presse : [Le suivi par géolocalisation des sarcelles d'hiver : un défi prometteur!](#) et [La cartographie HD, élément-clé de la voiture autonome.](#)
- ▶ Il est possible de travailler sur [OSMAPI](#) ou aussi d'utiliser la bibliothèque [PYTHON FOLIUM](#) qui permet de travailler sur des cartes et d'obtenir le rendu dans un navigateur Web.

4.2 Réaliser des activités

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ [Localisation, cartographie et mobilité \(itinéraires\).](#)
- ▶ [Principe du GPS et de GALILEO.](#)
- ▶ [Présentation du projet OPENSTREETMAP et contribution.](#)
- ▶ [Carte personnalisée avec PYTHON et la bibliothèque FOLIUM.](#)
- ▶ [Principe de l'algorithme de DIJKSTRA; création d'un itinéraire à l'aide de la bibliothèque PYTHON PYROUTELIB3.](#)
- ▶ [Apprendre un élément du cours à travers la réalisation d'un quiz.](#)
- ▶ [Précision verticale des coordonnées GPS.](#)

Fiche d'activité élève :

- ▶ [Utilisation de GÉOPORTAIL..](#)

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

CE QUE PROPOSE LE PROGRAMME

- ▶ Expérimenter la sélection d'informations à afficher et l'impact sur le changement d'échelle de cartes (GÉOPORTAIL), ainsi que les ajouts d'informations par les utilisateurs OPENSTREETMAP.
- ▶ Mettre en évidence les problèmes liés à un changement d'échelle dans la représentation par exemple des routes ou de leur nom sur une carte numérique pour illustrer l'aspect discret du zoom.
- ▶ Calculer un itinéraire routier entre deux points à partir d'une carte numérique.
- ▶ Connecter un récepteur GPS sur un ordinateur afin de récupérer la trame NMEA, en extraire la localisation.
- ▶ Extraire la géolocalisation des métadonnées d'une photo.
- ▶ Situer sur une carte numérique la position récupérée.
- ▶ Consulter et gérer son historique de géolocalisation.

TABLE 6.1 – Géolocalisation : compétences attendues chez les élèves.

Géolocalisation	
CONTENUS	CAPACITÉS ATTENDUES
GPS, GALILEO	Décrire le principe de fonctionnement de la géolocalisation.
Cartes numériques	Identifier les différentes couches d'information de GÉOPORTAIL pour extraire différents types de données. Contribuer à OPENSTREETMAP de façon collaborative.
Protocole NMEA 0183	Décoder une trame NMEA pour trouver des coordonnées géographiques.
Calculs d'itinéraires	Utiliser un logiciel pour calculer un itinéraire. Représenter un calcul d'itinéraire comme un problème sur un graphe.
Confidentialité	Régler les paramètres de confidentialité d'un téléphone pour partager ou non sa position.

5 Que faire de ces ressources ? Autoévaluation

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style L^AT_EX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

Les questionnaires à choix multiple* — QCM — à suivre clôturent le présent chapitre « Données et objets » et correspondent à chaque sujet qui y est abordé.

** De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).*

QUIZ 17 — DONNÉES ET TRAITEMENTS
5 POINTS

QUIZ 17.1 — BIEN RIVAUX 1 POINT

Parmi ces exemples, le/lesquels est/sont des biens rivaux?

Un des disques vinyles (produit à plus d'un million d'exemplaires) de la chanson « Que je t'aime » de Johnny HALLIDAY.

La vidéo numérique du concert unique de Céline DION au Bourget.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 17.2 — CODAGE BINAIRE 1 POINT

Qu'est-ce qui peut être codé en binaire sur un ordinateur?

Un nombre.

Une mesure physique.

Un texte avec des caractères.

Une image ou une vidéo.

Un son de musique.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 17.3 — CARTES PERFORÉES 1 POINT

Qui a inventé le stockage de données sur des cartes perforées*?

** On fera une recherche Wikipedia pour trouver la réponse à cette question*

C'est Joseph Marie CHARLES dit JACQUARD l'inventeur du métier à tisser programmable au début du XIX^e siècle.

C'est au XVIII^e siècle qu'un autre inventeur créa les cartes perforées et JACQUARD améliora ensuite cette technique.

Cette histoire de métier à tisser programmable est une fausse nouvelle, il a fallu attendre la révolution industrielle pour qu'il fonctionne vraiment.

Rien de tout cela, il y avait déjà des cartes perforées en Chine dès le XI^e siècle.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 17.4 — FORMAT VCARD 1 POINT

Une VCard est un format standard ouvert d'échange de données personnelles. Ci-dessous un exemple :

```
BEGIN:VCARD
VERSION:2.1
FN: Jean Dupont
N:Dupont;Jean
ADR;WORK;PREF;QUOTED-PRINTABLE:;Bruxelles 1200=Belgique;6A Rue
    ↳ Th. Decuyper
TEL;CELL:+1234 56789
EMAIL;INTERNET:jean.dupont@example.com
END:VCARD
```

- Ce format permet à la fois de transmettre ses coordonnées entre agendas et est aussi utilisé par des logiciels de messagerie.
- Il faut surtout refuser tous les logiciels qui peuvent utiliser ce format car on se fait pirater ses données personnelles.
- Dans ce format la ponctuation et certains mots (comme par exemple TEL;CELL:) sont les éléments-clés du langage, les autres mots correspondent aux données.
- On peut y rentrer une information audio pour avoir une prononciation sonore correcte du nom de la personne.
- C'est un format vraiment très souple : on peut y ajouter n'importe quel mot-clé de son choix, sans se soucier des standards, les logiciels comprendront.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 17.5 — TYPAGE DES VARIABLES 1 POINT**

Pourquoi l'ordinateur a-t-il besoin de connaître le type des variables que l'on crée ?

- En fait il n'a pas besoin du type, c'est juste pour eniquer coûusement les programmeurs.
- Pour que l'ordinateur trie ces variables en les rangeant par type.
- Pour savoir combien de place réservé en mémoire pour stocker la variable, et comment traduire le code binaire en une valeur.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 18 — PHOTOGRAPHIE NUMÉRIQUE****7 POINTS****QUIZ 18.1 — CCD (CHARGE COUPLED DEVICES) 1 POINT**

En quelle année les capteurs CCD ont-ils été inventés ?

- 1969.
- 1975.
- 1985.
- 1989.
- 1995.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 18.2 — ENCODAGE DES IMAGES 1 POINT

Une image peut être encodée suivant différents modèles de couleurs, notamment RVB (Rouge, Vert, Bleu) et TSL (Teinte, Saturation, Lumière). Parmi les affirmations suivantes lesquelles sont vraies ?

- Le système de couleurs est lié aux photosites du capteur : certains sont en RVB, d'autres en TSL.
- Tous les photosites du capteur sont en RVB, le changement de modèle se fait ensuite par logiciel.
- Un pixel correspond le plus souvent à 4 photosites de 3 types : 1 en Rouge, 2 en Vert et 1 en Bleu.
- Un pixel correspond nécessairement à 3 photosites de chaque type : 1 par canal du modèle de couleurs.
- Chaque pixel est en fait un photosite, la décomposition en RVB se fait ensuite par analyse spectrale du signal.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 18.3 — MÉTADONNÉES EXIF 1 POINT**

Quelles informations a-t-on parmi les métadonnées EXIF d'une image ?

- La date et l'heure de prise de vue.
- Le temps d'exposition.
- Le nom des personnes photographiées.
- L'espace colorimétrique de l'image stockée.
- La liste des traitements appliqués pour passer de l'image captée par le capteur à l'image stockée en mémoire.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

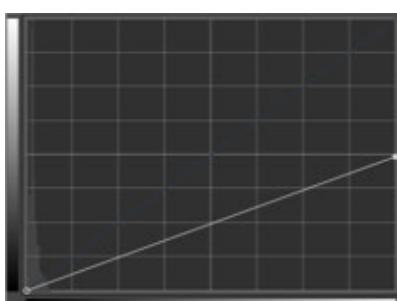
Dans un outil de traitement d'images comme GIMP, on peut ajuster la luminosité et le contraste d'une image en jouant sur des courbes (Menu « Couleurs », outil « Courbes... »). Une courbe représente graphiquement la manière dont chaque valeur de pixel est transformée : en abscisse, on trouve les valeurs de pixel d'origine et en ordonnée, les valeurs des pixels dans l'image ajustée, le lien entre les deux se faisant *via* la courbe.

Ci-dessous on expose quatre courbes de valeurs et quatre propositions d'effets sur une image. Associer la courbe à l'effet qu'elle produit sur l'image en sélectionnant la réponse adéquate. Chaque courbe correspond à un et un seul effet, et vice-versa.

QUIZ 18.4 — CONTRASTE VERSUS LUMINOSITÉ — COURBE A 1 POINT

Quel est l'effet de la courbe A ci-contre ?

- Aucun effet.
- Baisse de luminosité.
- Augmentation du contraste.
- Meilleur contraste sur les tons sombres.

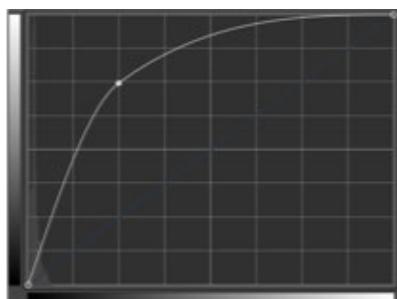
VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

Contraste vs luminosité, courbe A.

QUIZ 18.5 — CONTRASTE VERSUS LUMINOSITÉ — COURBE B 1 POINT

Quel est l'effet de la courbe B ci-contre ?

- Aucun effet.
- Baisse de luminosité.
- Augmentation du contraste.
- Meilleur contraste sur les tons sombres.

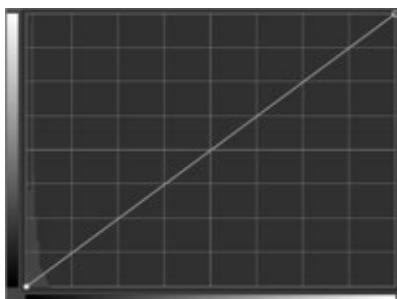
VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

Contraste vs luminosité, courbe B.

QUIZ 18.6 — CONTRASTE VERSUS LUMINOSITÉ — COURBE C 1 POINT

Quel est l'effet de la courbe C ci-contre ?

- Aucun effet.
- Baisse de luminosité.
- Augmentation du contraste.
- Meilleur contraste sur les tons sombres.

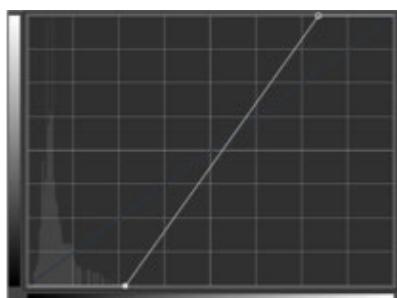
VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

Contraste vs luminosité, courbe C.

QUIZ 18.7 — CONTRASTE VERSUS LUMINOSITÉ — COURBE D 1 POINT

Quel est l'effet de la courbe D ci-contre ?

- Aucun effet.
- Baisse de luminosité.
- Augmentation du contraste.
- Meilleur contraste sur les tons sombres.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

Contraste vs luminosité, courbe D.

QUIZ 19 — INFORMATIQUE EMBARQUÉE ET OBJETS CONNECTÉS 5 POINTS**QUIZ 19.1 — INTERNET DES OBJETS — IoT/IoT 1 POINT**

L'Internet des objets est appelé ainsi car...

- ...les objets sont tous en mesure de communiquer entre eux.
- ...des objets sont connectés à Internet en tant que sous-ensemble.
- ...les humains sont vus comme des objets par les machines.
- ...on utilise un langage objet pour programmer Internet.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 19.2 — OBJETS CONNECTÉS ET DÉPORT DES DONNÉES 1 POINT**

Dans la vidéo de Guillaume sur la thématique « Objets connectés » il est dit : « Bien souvent, le programme associé à un objet n'est pas dans l'objet mais en ligne sur le serveur du fabricant qui récupère, traite et

possiblement conserve vos données [...] ». Quelles sont les motivations pour cela ?

- Le fabricant peut ainsi changer certains aspects du fonctionnement de l'objet et corriger les dysfonctionnements.
- Certains traitements sont trop complexes pour être exécutés directement sur l'objet, donc déporter le traitement vers des serveurs puissants permet de résoudre ce problème.
- En centralisant les données sur un seul serveur, on limite les risques de sécurité puisque l'on évite de disperser les informations sur une multitude de petits objets.
- Le fabricant peut observer le comportement de l'utilisateur de l'objet, comme savoir si une ampoule connectée est utilisée.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 19.3 — ROBOT VERSUS OBJET CONNECTÉ 1 POINT**

Peut-on assimiler un robot à un objet connecté ? Cela dépend des définitions mais voici quelques affirmations qui aident à faire la différence.

- Un système robotique comporte des capteurs en entrée et/ou des actuateurs en sortie et est piloté par un logiciel, comme tout objet connecté à l'Internet.
- Un robot est forcément un système humanoïde ou qui ressemble à un animal.
- Un objet connecté est forcément relié à un réseau, un robot ne l'est pas forcément.
- Robots connectés et objets connectés posent les mêmes problèmes de fiabilité et sécurité.
- Un robot doit interagir avec un humain, pas un objet connecté.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 19.4 — CONSOMMATION ÉNERGÉTIQUE 1 POINT**

L'utilisation d'objets connectés a-t-elle aujourd'hui tendance à optimiser ou augmenter la consommation électrique ?

- À ce jour, la consommation électrique est toujours diminuée par l'utilisation d'objets connectés, par exemple de domotique.
- Aujourd'hui, la consommation électrique n'est jamais réduite avec l'utilisation d'objets connectés, on génère surtout une consommation résiduelle.
- Actuellement, c'est un dilemme : l'utilisation génère une consommation résiduelle, et selon le produit et son usage on peut ou non optimiser sa consommation.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 19.5 — RESPECT DE LA VIE PRIVÉE 1 POINT**

L'accumulation d'informations par de multiples acteurs, depuis les fabricants d'objets (ampoules, prises, capteurs de température, etc.), les fabricants d'enceintes intelligentes, les développeurs d'applications pour smartphones dédiées au contrôle de ces objets, jusqu'aux fournisseurs

de services pour créer des automatismes à partir de ces objets, pose un problème fondamental de respect de la vie privée. Par exemple, la maison connectée contribue à permettre à des acteurs extérieurs de connaître la liste quasi exhaustive des *smartphones* et équipements informatiques de mon domicile. Vrai ou faux ?

- Faux, ces objets ne peuvent pas accéder à mes équipements informatiques individuels.
- Vrai dès l'instant où j'installe une application pour contrôler ces objets connectés.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 20 — LOCALISATION, CARTOGRAPHIE ET MOBILITÉ****5 POINTS****QUIZ 20.1 — HORLOGE GPS 1 POINT**

À la différence d'un satellite, un récepteur GPS ne possède pas d'horloge atomique : son heure va donc être un peu décalée par rapport à « l'heure officielle ». En quoi est-ce grave ?

- Le récepteur GPS ne sait pas calculer la distance exacte le séparant d'un satellite donné s'il ne reçoit pas d'autre information complémentaire.
- Le récepteur GPS ne sait pas si les messages reçus depuis les satellites sont les plus récents ou pas, c'est très gênant.
- Le récepteur GPS a besoin d'une mesure supplémentaire avec un 4ème satellite pour compenser ce décalage de temps possible.
- Le récepteur GPS ne sait pas à quel moment précis solliciter un message de localisation auprès des satellites GPS puisqu'il n'a pas la bonne heure.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 20.2 — SYNCHRONISATION GPS 1 POINT**

Vrai ou faux ? Grâce au GPS, on peut synchroniser très précisément l'horloge d'un équipement sur Terre, même en l'absence de connexion Internet (et d'horloge atomique !). Vrai ou faux ?

- Impossible, voyons !
- Ah oui, possible !
- Vous pouvez répéter la question ?

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 20.3 — SATELLITE GPS 1 POINT**

Dans la liste suivante, sélectionnez la proposition exacte.

- Chaque récepteur GPS calcule la position des satellites grâce à des éphémérides qui sont pré-calculés et stockés sur le récepteur.
- Chaque satellite GPS connaît avec une très grande précision sa localisation, à tout moment sur son orbite.

Les satellites GPS sont géostationnaires et par définition leur localisation est fixe dans le temps. C'est plus simple ainsi.

Chaque récepteur GPS communique avec les serveurs (américains) du système GPS pour déterminer la position de chaque satellite GPS qu'il a besoin de connaître.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 20.4 — ALGORITHME DE PLUS COURT CHEMIN 1 POINT**

Dans la vidéo de Guillaume qui présente la thématique Localisation, l'algorithme proposé pour choisir le plus court chemin entre deux gares sur la carte repose sur un certain nombre d'hypothèses. Lesquelles ?

La machine où se fait le calcul d'itinéraire a la connaissance complète a priori du graphe de connexion entre gares.

La machine où se fait le calcul d'itinéraire détermine au fur et à mesure le graphe de connexion entre gares.

La machine où se fait le calcul d'itinéraire a la connaissance complète des distances minimales entre la gare d'arrivée et chacune des gares intermédiaires.

La machine où se fait le calcul d'itinéraire détermine au fur et à mesure les distances minimales entre la gare d'arrivée et chacune des gares intermédiaires.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 20.5 — ATTRIBUTS DU PLUS COURT CHEMIN 1 POINT**

L'algorithme qui calcule un plus court chemin utilise des vecteurs portant des attributs. Quelles affirmations sont correctes ?

Si un attribut est la longueur du vecteur, on peut calculer le chemin le plus court.

Si un attribut est la longueur et la vitesse de circulation sur le vecteur, on peut trouver le chemin le plus rapide.

Si un attribut est le sens de circulation cela permet à l'algorithme de ne pas proposer de chemin en sens interdit.

Si un attribut est — entre autres — le volume du trafic en temps réel cela permet de détecter les bouchons.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

RÉSEAUX ET ENVIRONNEMENT

LE PROGRAMME OFFICIEL SNT FAIT MENTION DE TROIS THÈMES sur les réseaux qui viennent compléter les quatre champs d'investigation relatifs aux données et aux objets exposés dans le chapitre précédent. Il s'agit de l'Internet et ses réseaux adossés, du Web et de la diffusion d'information et des réseaux sociaux en tant que tels.

À ces trois sujets, vient se greffer ici celui de la problématique environnementale dérivée de l'utilisation du numérique et de tous ces réseaux. Bien que cela ne soit pas partie directement prenante du programme officiel, la consommation d'énergie et de matériaux rares pose en effet nombre de questions de transition écologique à résoudre.

Comme pour le chapitre précédent, chaque thématique possède le même canevas commun : « Découvrir la thématique » et « Réaliser des activités ». Si la première de ces sections vise la présentation du sujet, la seconde est ouverte aux suggestions et amenée à intégrer les contributions des enseignants.

Il est clair que chaque thème énoncé s'alimente des contenus exposés dans les autres parties du document, notamment « Informatique, création numérique » et « Fondements de l'informatique ».

1 Internet et les réseaux

1.1 Découvrir la thématique

1.1.1 Anchage dans le réel

A. POINTS-CLÉS

- ▶ Internet est un réseau de réseaux : il interconnecte près de 50 000 réseaux autonomes.
- ▶ Internet est possible parce que les pays du monde entier, même en conflit, respectent les standards de communication.
- ▶ La gouvernance internationale d'Internet est l'élaboration et l'application conjointes, par les États, le secteur privé, la société civile et les organisations internationales, des principes, normes, règles et procédures de fonctionnement d'Internet.
- ▶ Aujourd'hui Internet rassemble bien plus d'objets connectés que d'interfaces humaines (*smartphone*, ordinateur, tablette).

SOMMAIRE

1 Internet et les réseaux
1.1 Découvrir la thématique 215
1.1.1 Anchage dans le réel ■ 1.1.2 Volet historique ■ 1.1.3 Explication des notions
1.2 Réaliser des activités 219
2 Web et information
2.1 Découvrir la thématique 220
2.1.1 Anchage dans le réel ■ 2.1.2 Volet historique ■ 2.1.3 Explication des notions
2.2 Réaliser des activités 225
3 Réseaux sociaux
3.1 Découvrir la thématique 226
3.1.1 Anchage dans le réel ■ 3.1.2 Volet historique ■ 3.1.3 Explication des notions
3.2 Réaliser des activités 231
4 Enjeux environnementaux
4.1 Découvrir la thématique 232
4.1.1 Anchage dans le réel ■ 4.1.2 Numérique : menace ou espoir pour l'environnement ?
4.2 Impacts environnementaux 235
4.2.1 Aspects de fabrication ■ 4.2.2 Aspects de consommation ■ 4.2.3 Aspects de recyclage ■ 4.2.4 Autre impacts ■ 4.2.5 Comment agir ?
5 Que faire de ces ressources ? Quiz

B. MOTS-CLÉS

- ▶ Internet : le réseau des réseaux.
- ▶ La neutralité du Net.
- ▶ La gouvernance d'Internet.
- ▶ Statistiques sur Internet dans le monde et sur les chiffres en 2018.

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Grâce à sa souplesse et son universalité, Internet est devenu le moyen de communication principal entre hommes et machines.

IMPACTS SUR LES PRATIQUES HUMAINES

Internet a fait progressivement disparaître beaucoup des moyens de communication précédents : télégramme, télex, courrier postal pour une bonne partie et bientôt le téléphone fixe grâce à VoIP (voix sur IP). Son trafic prévu pour 2021 est de trois mille trois cents milliards de milliards d'octets ($3,3 \times 10^{21}$ octets).

Internet a aussi ses problèmes : absence de garantie temporelle sur l'arrivée des paquets et possibilité d'attaques par saturation en envoyant un très grand nombre de messages à un site donné, pour y provoquer un déni de service.

La neutralité du Net, présente dès l'origine du réseau, exprime l'idée que les routeurs doivent transmettre les paquets indépendamment du type de leur contenu : texte, vidéo, etc. Mais elle est constamment remise en cause par certains *lobbies* industriels.

1.1.2 Volet historique

L'histoire d'Internet remonte au développement des premiers réseaux de télécommunication. L'idée d'un réseau informatique, permettant aux utilisateurs de différents ordinateurs de communiquer, se développa par étapes successives, jusqu'à ce « réseau des réseaux » que nous connaissons aujourd'hui en tant qu'Internet. Il est le fruit à la fois de développements technologiques et du regroupement d'infrastructures réseau existantes et de systèmes de télécommunications.

Consulter la vidéo [vidéo 7.2](#) et le chapitre sur les réseaux d'[Open-Classrooms](#) pour disposer d'un historique de l'Internet.

Source : module 4, formation Class'Code.

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

Dès les années cinquante, les ordinateurs ont été mis en réseau pour échanger des informations, mais de façon très liée aux constructeurs d'ordinateurs ou aux opérateurs téléphoniques. Les réseaux généraux indépendants des constructeurs sont nés aux États-Unis avec ArpaNet (1970) et en France avec Cyclades (1971). Cet effort a culminé avec Internet, né en 1983.

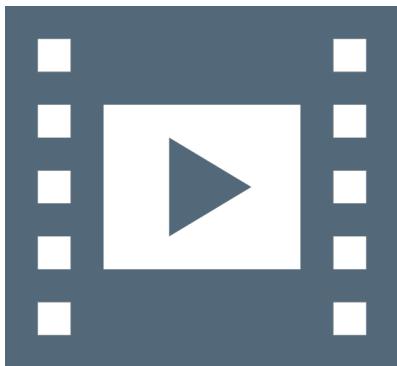
1.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Ce qui fait fonctionner Internet c'est la notion de protocole, qui est à la fois un standard de communication et un algorithme distribué sur plusieurs machines.



VIDÉO 7.1 – Internet et les réseaux.



VIDÉO 7.2 – Naissance de l'Internet.



- ▶ Les grands mécanismes sous-jacents à l'Internet sont l'identification et la connexion entre les machines et le transfert de paquets d'information entre elles.
- ▶ Les solutions qui fonctionnent à grande échelle sont basées sur des protocoles qui peuvent engendrer des erreurs qui sont ensuite corrigées par les logiciels qui les utilisent.

B. MOTS-CLEFS

- ▶ La notion de protocole TCP/IP, le service de DNS.
- ▶ La notion de réseau pair à pair.

C. CE QUE DIT LE PROGRAMME

PROTOCOLE TCP/IP

Internet est défini par le protocole IP (*Internet Protocol*), ensemble de normes qui permettent d'identifier et de nommer de façon uniforme tous les ordinateurs ou objets qui lui sont connectés. IP est accompagné de protocoles de transmission pour transférer l'information par paquets, le principal étant TCP/IP (*Transmission Control Protocol*). De nature logicielle, Internet s'appuie sur une grande variété de réseaux physiques où IP est implémenté. Il uniformise l'accès à tous les ordinateurs, les téléphones et les objets connectés.

DONNÉES ET INFORMATION

Internet manipule deux types d'information : les contenus envoyés et les adresses du destinataire et de l'émetteur. Ces deux types d'information sont regroupés dans des paquets de taille fixe, de façon uniforme et indépendante du type de données transportées : texte, images, sons, vidéos, etc. Les adresses sont numériques et hiérarchiques, comme **78.109.84.114**, mais l'utilisateur connaît surtout des adresses symboliques normalisées, comme wikipedia.fr. Le système DNS (*Domain Name System*) transforme une adresse symbolique en adresse numérique. Il est réalisé par un grand nombre d'ordinateurs répartis sur le réseau et constamment mis à jour.

ALGORITHMES ET PROGRAMMES

Lors du routage, un paquet peut ne pas arriver pour deux raisons : une panne matérielle d'une ligne ou d'un routeur, ou sa destruction. Chaque paquet contient l'information d'un nombre maximal de routeurs à traverser : pour ne pas encombrer le réseau, il est détruit si ce nombre est atteint. C'est le protocole TCP qui fiabilise la communication en redemandant les paquets manquants. Il garantit que tout paquet finira par arriver, sauf panne matérielle incontournable. TCP réordonne aussi les paquets arrivés dans le désordre et diminue la congestion du réseau en gérant au mieux les redemandes. Mais ni Internet ni TCP ne possèdent de garantie temporelle d'arrivée des paquets, ce qui nuit à la qualité de la diffusion du son ou des vidéos et de la téléconférence. En effet, dans une vidéo on peut perdre une image isolée, mais pas le fil du temps.

D'autres protocoles s'appuient sur ceux d'Internet, par exemple les protocoles du Web (HTTP et HTTPS) et le protocole NTP (*Net-*

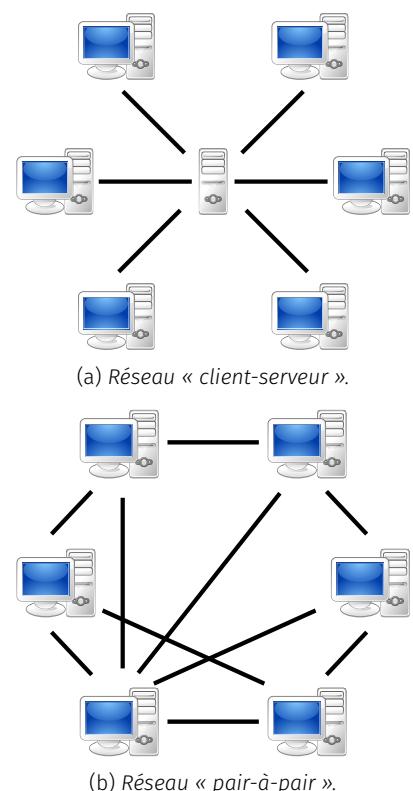


FIGURE 7.1 — Typologie des réseaux.

work Time Protocol) qui permet de synchroniser finement les heures des ordinateurs et objets connectés.

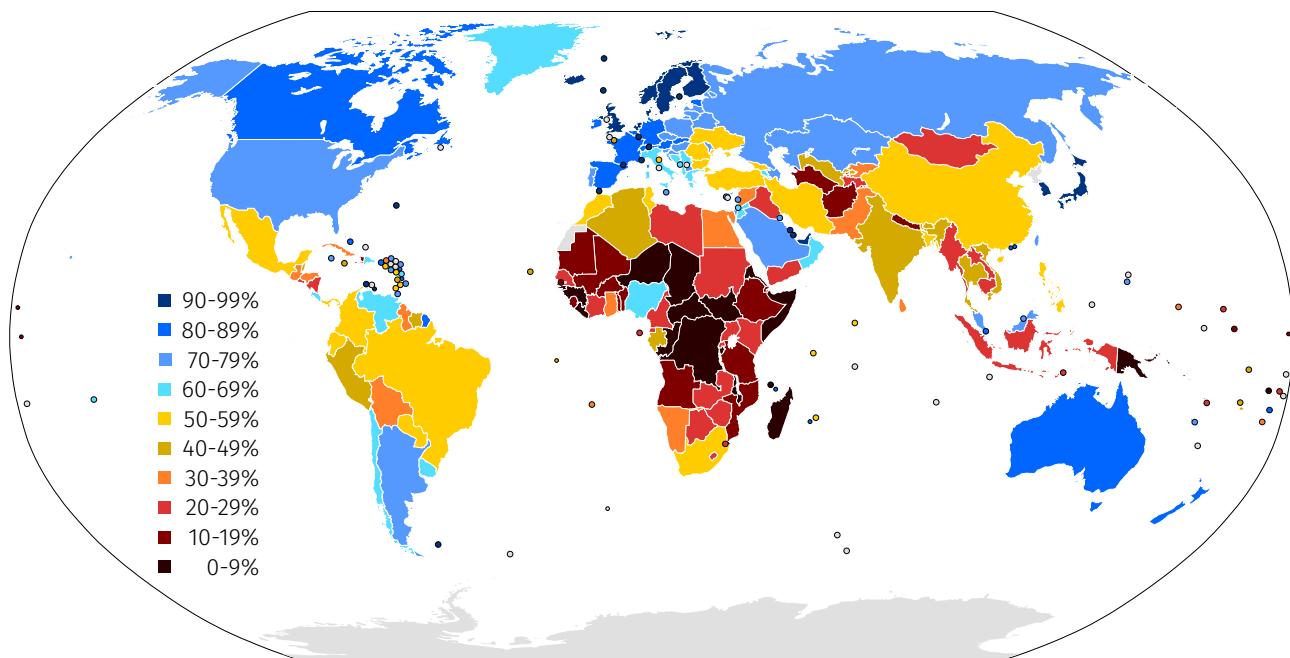


FIGURE 7.2 — Taux de pénétration de l'Internet par pays (mise à jour 2016).

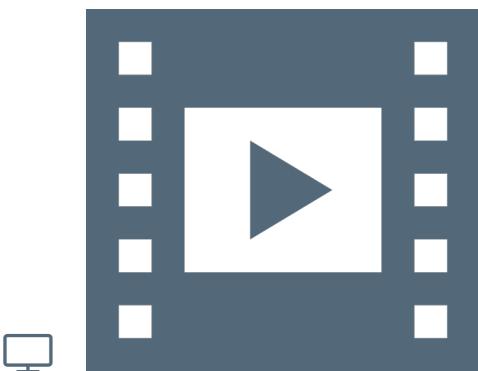
MACHINES

Réseau mondial, Internet fonctionne au moyen de routeurs, de lignes de transmissions à très hauts débits (fibres optiques) entre routeurs, de réseaux de téléphonie mobile, et de réseaux locaux. Ses protocoles étant logiciels, il peut s'appuyer sur *n'importe quel réseau physique* qui les implémente : 4G, Ethernet, ADSL, Wi-Fi, Bluetooth, etc. TCP/IP n'est pas implanté dans l'infrastructure, mais dans chacun des ordinateurs connectés et un serveur DNS est aussi un ordinateur connecté. Des mécanismes complexes assurent la continuité de la connexion, par exemple pour passer sans interruption de téléphonie 4G au Wi-Fi ou son ubiquité, par exemple pour passer de façon invisible d'antenne à antenne avec un téléphone portable quand on voyage. Dans les réseaux *pair-à-pair* s'appuyant sur Internet et souvent utilisé pour le transport de vidéos, chaque ordinateur sert à la fois d'émetteur et de récepteur.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

- une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours;
- des exemples concrets de ce qui est étudié ou produit dans ces domaines et une réflexion sur les besoins et enjeux actuels et



VIDÉO 7.3 — Internet et les réseaux, Benoît PARREIN.

à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur la thématique Internet a eu lieu le 12 mars 2019 dans le cadre du projet CLASS'CODE à l'Université de Nantes. Elle a été retransmise en ligne et est maintenant disponible sur YOUTUBE.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ La Partie 2 de la formation CLASS'CODE sur les réseaux permet de comprendre comment l'information circule sur le réseau, et de découvrir les acteurs du Net.
- ▶ La Partie 3 de la formation CLASS'CODE sur les réseaux permet de comprendre ce qu'est un protocole et d'appréhender l'évolution des réseaux.
- ▶ Pour les enseignants : la conférence de Christine GAUBERT-MACON sur le Web, Internet et les réseaux sociaux, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Une sélection d'articles du site INTERSTICES.

Ouvrages

- ▶ Fabien TARISSAN (2019) « Au cœur des réseaux : des sciences aux citoyens », Édition le Pommier. Il traite de science, plus particulièrement des réseaux (informatiques et autres), des algorithmes qui y opèrent et de l'impact qu'ils ont sur notre manière de nous informer en ligne (*fake news*, publicité ciblée, données personnelles, etc.), et de l'économie sous-jacente.
- ▶ Valérie SCHAFER (2018) « En construction : La fabrique française d'Internet et du Web dans les années 1990 », INA.
- ▶ Stéphane BORTZMEYER (2018) « Cyberstructure : L'Internet, un espace politique », C&F Édition.

Créer son cours

- ▶ La ressource « isoloir » propose une activité sur la régulation d'Internet, ces documents sont librement réutilisables.
- ▶ Comment relier la notion d'URL avec les couches réseaux d'Internet.
- ▶ Une vidéo explicative d'Internet pour les lycéen-ne-s.
- ▶ Le numérique, la loi et notre vie privée.

1.2 Réaliser des activités

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ Introduction à Internet : réseau de réseaux, adresse IP.
- ▶ Notion de protocole, TCP/IP, encapsulation des données.
- ▶ Notion de routage des paquets.
- ▶ Prise en main du logiciel de simulation réseau Filius.
- ▶ Introduction à la notion de DNS (*Domain Name Server*).
- ▶ Introduction aux réseaux pair-à-pair.
- ▶ Activité déconnectée/connectée, routage entre routeurs.

Fiche d'activité élève :

- ▶ Observer une communication en réseau Ethernet.

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

CE QUE PROPOSE LE PROGRAMME

- ▶ Illustrer le fonctionnement du routage et de TCP par des activités débranchées ou à l'aide de logiciels dédiés, en tenant compte de la destruction de paquets.
- ▶ Déterminer l'adresse IP d'un équipement et l'adresse du DNS sur un réseau.
- ▶ Analyser son réseau local pour observer ce qui y est connecté.
- ▶ Suivre le chemin d'un courriel en utilisant une commande du protocole IP.

TABLE 7.1 — Internet : compétences attendues chez les élèves.

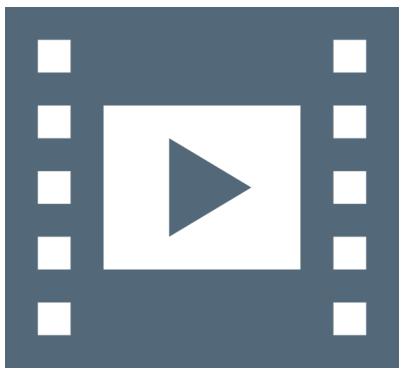
Internet	
CONTENUS	CAPACITÉS ATTENDUES
Protocole TCP/IP : paquets, routage des paquets	Distinguer le rôle des protocoles IP et TCP. Caractériser les principes du routage et ses limites. Distinguer la fiabilité de transmission et l'absence de garantie temporelle.
Adresses symboliques et serveurs DNS	Sur des exemples réels, retrouver une adresse IP à partir d'une adresse symbolique et inversement.
Réseaux pair-à-pair	Décrire l'intérêt des réseaux pair-à-pair ainsi que les usages illicites qu'on peut en faire.
Indépendance d'Internet par rapport au réseau physique	Caractériser quelques types de réseaux physiques : obsolètes ou actuels, rapides ou lents, filaires ou non. Caractériser l'ordre de grandeur du trafic de données sur internet et son évolution.

2 Web et information

2.1 Découvrir la thématique

2.1.1 Ancrage dans le réel

A. POINTS-CLÉS

- 
- VIDÉO 7.4 — Web et information.
- ▶ Le Web est à la fois un *espace documentaire* dans lequel on trouve de l'information (Web 1.0), un *espace social et participatif* dans lequel on crée soi-même de l'information pour la partager (Web 2.0), et un *espace applicatif* dans lequel on interagit avec des logiciels qui offrent un certain nombre de fonctionnalités.
 - ▶ Le *Web dit sémantique* (Web 3.0) est un *espace où la connaissance est structurée* de manière à être manipulée par des algorithmes qui interagissent à travers le Web pour proposer des services et un espace (Web 4.0) dans lequel les *objets connectés* eux-mêmes peuvent interagir entre eux.
 - ▶ Cette révolution génère de *nouveaux métiers* et de *nouvelles façons de travailler*, et conduit à la *dématerrialisation* des services (administratifs, etc...). Contrairement à une idée reçue elle génère plus qu'elle ne réduit les impacts environnementaux.
 - ▶ Tim Berners-Lee disait récemment que le Web, « *conçu comme un outil ouvert, collaboratif et émancipateur, a été détourné par des escrocs et des trolls, qui l'ont utilisé pour manipuler le reste des internautes à travers le monde* » et « *il n'est pas trop tard pour changer le Web* » conclut-il positivement.

- ▶ Il faut aussi comprendre le Web non référencé, Web profond à également distinguer du *Dark Web*. Ce dernier, sans fantasmer, est un phénomène important de société en lien avec une réalité socio-économique à connaître.

B. MOTS-CLÉS

- ▶ Le Web, une des applications d'Internet,
- ▶ L'encyclopédie Wikipedia.
- ▶ La dématérialisation d'une organisation.

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Le Web (toile ou réseau) désigne un système donnant accès à un ensemble de données (page, image, son, vidéo) reliées par des liens hypertextes et accessibles sur le réseau Internet.



WIKIPÉDIA

L'encyclopédie libre

© Gerd Altmann via Pixabay



IMPACTS SUR LES PRATIQUES HUMAINES

Dans l'histoire de la communication, le Web est une révolution : il a ouvert à tous la possibilité et le droit de publier; il permet une coopération d'une nature nouvelle entre individus et entre organisations : commerce en ligne, création et distribution de logiciels libres multi-auteurs, création d'encyclopédies mises à jour en permanence, etc.; il devient universel pour communiquer avec les objets connectés.

Le Web permet aussi de diffuser toutes sortes d'informations dont ni la qualité, ni la pertinence, ni la véracité ne sont garanties et dont la vérification des sources n'est pas toujours facile. Il conserve des informations, parfois personnelles, accessibles partout sur de longues durées sans qu'il soit facile de les effacer, ce qui pose la question du droit à l'oubli. Il permet une exploitation de ses données, dont les conséquences sociétales sont encore difficiles à estimer : recommandation à des fins commerciales, bulles informationnelles, etc. En particulier, des moteurs de recherche permettent à certains sites d'acquérir de la visibilité sur la première page des résultats de recherche en achetant de la publicité qui apparaîtra parmi les liens promotionnels.

2.1.2 Volet historique

C'est en 1990 que Tim BERNERS-LEE et Robert CAILLIAU, chercheurs au CERN (Organisation européenne pour la recherche nucléaire), conçoivent le premier site Internet en HTML (langage informatique utilisé pour développer les sites Web) et inventent le protocole HTTP. En bref, le Web est issu d'un croisement d'idées qui ont bien progressé depuis sa création : pour que le Web reste un lieu libre et ouvert, nous devons nous l'approprier!

Consulter la vidéo [vidéo 7.5](#) et le chapitre sur le Web d'OpenClassrooms pour disposer d'un historique du Web.

Source : module 4, formation Class'Code.

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

- ▶ 1965 : invention du concept d'hypertexte par Ted NELSON.
- ▶ 1989 : naissance au CERN par Tim BERNERS-LEE.
- ▶ 1993 : mise dans le domaine public, disponibilité du premier navigateur Mosaic.
- ▶ 1995 : mise à disposition de technologies pour le développement de site Web interactif (JavaScript) et dynamique (PHP).
- ▶ 2001 : pages standardisées grâce au *Document Object Model*.
- ▶ 2010 : mise à disposition de technologies pour le développement d'applications sur mobiles.

2.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Le langage HTML5 permet de définir une page Web, ses métadonnées qui sont dans l'en-tête (balise `<head>`) et son contenu dans le corps de la page (balise `<body>`), la mise en page est elle déléguée aux directives du CSS.
- ▶ Dans une page Web on peut mettre des éléments d'interaction, par exemple des champs de formulaire ou des éléments cliquables, du code JavaScript va alors permettre de rendre ces éléments actifs.
- ▶ Les adresses internet, ou URL (Localisation de Ressources Universelles), constituent un petit langage qui permet de localiser un contenu, mais aussi de faire une requête sur ce contenu. Chaque partie de l'URL correspond à un paramètre.
- ▶ Le protocole HTTP permet de faire une requête Web et d'obtenir la réponse, c'est le mécanisme standard qui permet d'utiliser le Web.

B. MOTS-CLEFS

- ▶ Trois langages principaux des pages Web : HTML5, CSS3, JavaScript.
- ▶ Protocole HTTP du Web et sa variante sécurisée HTTPS.
- ▶ Notion de moteur de recherche.
- ▶ Environnement client-serveur.

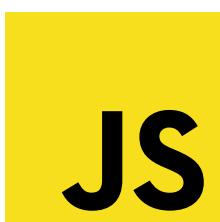
C. CE QUE DIT LE PROGRAMME

NORMALISATION DE LA PRÉSENTATION

Sur le Web, les textes, photos, vidéos, graphiques, sons, programmes sont exprimés et assemblés dans divers formats normalisés par un consortium mondial (W3C : *World Wide Web Consortium*), ce qui permet une circulation standardisée de toutes ces informations.



VIDÉO 7.5 — Naissance du Web.



Les pages Web sont écrites dans le langage de balises HTML (Hypertext Markup Language). Leur style graphique est défini dans le langage CSS (*Cascading Style Sheets*).

Les pages ont une adresse unique, nommée URL (*Uniform Resource Locator*). Elles sont accessibles via Internet en utilisant le protocole HTTP (Hypertext Transfer Protocol) ou sa version sécurisée HTTPS qui crypte les échanges. L'affichage des pages est réalisé chez l'utilisateur par un programme appelé navigateur.

Un hypertexte est un texte augmenté de renvois automatiques à des textes, des images ou des sons. Initialement, un hypertexte se restreignait à la mémoire d'un seul ordinateur. Dans une page Web, ce renvoi se fait sur n'importe quelle machine du réseau internet, par le truchement de l'adresse de la page Web du texte (URL) auquel il fait référence. La toile d'araignée construite par les liens peut être représentée sous forme d'un graphe qui matérialise la structure du Web.



© Gerd Altmann via Pixabay

MOTEURS DE RECHERCHE

Les moteurs de recherche permettent de trouver des informations dans des pages dont on ne connaît pas l'adresse, voire dont on ignore l'existence. La méthode de recherche appelée référencement naturel se décompose en trois grandes activités, réalisées par les moteurs de recherche : (1) le parcours automatique du Web pour collecter les pages visitées (aspiration des pages Web effectuée par des robots); (2) l'analyse du contenu des pages et leur indexation sur les mots qu'elles contiennent (constitution d'un annuaire inversé qui associe à chaque terme les URL des pages où il apparaît); (3) la troisième activité, réalisée à chaque fois qu'un internaute fait une requête, construit

une liste ordonnée des pages (classement) comportant les mots clés de la requête. Leur ordre dépend notamment de leur popularité (principe des liens), de leur pertinence (aux mots de la requête), et de l'ordre des termes de la requête.

Les concepteurs de site Web peuvent améliorer le référencement de leurs pages en choisissant bien les mots et en les plaçant à des endroits stratégiques dans les pages.

INTERACTION CLIENT/SERVEUR

Le Web s'appuie sur le dialogue entre clients et serveurs. L'interaction est à l'initiative des clients (les applications qui se connectent au Web, dont les navigateurs), qui envoient des requêtes HTTP aux serveurs. Ces derniers renvoient leur résultat : des pages qu'ils ont stockées ou qu'ils créent dynamiquement en fonction de la requête formulée. Les pages reçues par les clients peuvent contenir des codes exécutables (souvent en JavaScript) qui permettent aux clients d'effectuer des traitements en accédant aux ressources de son ordinateur et en interagissant avec les serveurs. Les applications peuvent être paramétrées pour autoriser ou interdire l'accès à des ressources locales aux programmes téléchargés par les pages.

SÉCURITÉ ET CONFIDENTIALITÉ

En formulant des requêtes sur des sites Web dynamiques et en laissant des programmes s'exécuter sur sa machine, l'utilisateur prend des risques : il peut communiquer des informations personnelles à son insu à des serveurs qui en gardent une trace, à distance ou localement par des *cookies*, ou encore charger des pages contenant des programmes malveillants, par exemple permettant d'espionner en continu les actions de l'utilisateur.

Par ailleurs, un navigateur peut garder un historique de toutes les interactions et le laisser accessible aux sites connectés. L'utilisateur peut utiliser des services qui s'engagent à ne pas garder de traces, comme certains moteurs de recherche. Il peut aussi paramétrier son navigateur de façon à ce qu'il n'enregistre pas d'historique des interactions. De fausses pages peuvent encore être utilisées pour l'hameçonnage des utilisateurs. Un nom de lien pouvant cacher une adresse Web malveillante, il faut examiner cette adresse avant de l'activer par un clic.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :

- une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours;
- des exemples concrets de ce qui est étudié ou produit dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant dans ces champs de spécialité



VIDÉO 7.6 – Web et information, Julien PIERRE.



(chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur *Le Web et son information* a eu lieu le 26 mars 2019 dans le cadre du projet CLASS'CODE à l'Université de Nantes. Elle a été retransmise en ligne et est maintenant disponible sur YouTube.

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ La Partie 1 de la formation CLASS'CODE sur les réseaux permet de se familiariser avec la création d'un site Web, au moyen d'un outil libre et facilement réutilisable.
- ▶ La conférence de Fabien GANDON *Les (r)évolutions de la planète Web*, 29 avril 2015.
- ▶ Pour les enseignants : la conférence de Christine GAUBERT-MACON sur le Web, Internet et les réseaux sociaux, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Le module 1 « Protéger ses données et son identité numérique » et le module 4 « Navigation et traçage sur le Web » du Mooc *Protection de la vie privée dans le monde numérique*, produit par l'INRIA sur la plateforme FUN.
- ▶ Une sélection d'articles du site INTERSTICES.

Créer son cours

- ▶ Des animations intéressantes sur les réseaux sociaux et le web en général (les *cookies*...) sur le site d'Arte.
- ▶ On dispose aussi sur pixees.fr d'une initiation à HTML (avec une présentation permettant de prendre du recul et un document sur les droits et devoirs à prendre en compte).
- ▶ Memento HTML et memento CSS pour les élèves.
- ▶ *Le Web devint sémantique*, article de popularisation, revue en ligne BINAIRE sur le site du Monde, 17 avril 2018.

2.2 Réaliser des activités

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ Introduction à la notion de Web.
- ▶ Introduction à la notion d'URL.
- ▶ Introduction au HTML et au CSS.
- ▶ Principe du modèle client-serveur.
- ▶ Le protocole HTTP, une première approche.
- ▶ Comprendre les cookies, paramétrier son navigateur web.
- ▶ Les moteurs de recherche.
- ▶ Fonctionnement d'un moteur de recherche.

Fiche d'activité élève :

- ▶ Découvrir HTML et CSS, avec mementos HTML et CSS pour les élèves.
- ▶ Le Web sécurité et confidentialité.
- ▶ Exemple de séquence sur le Web.

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

CE QUE PROPOSE LE PROGRAMME

- ▶ Construire une page Web simple contenant des liens hypertextes, la mettre en ligne.

- ▶ Modifier une page Web existante, changer la mise en forme en modifiant son CSS. Insérer un lien dans une page Web.
- ▶ Comparer les paramétrages de différents navigateurs.
- ▶ Utiliser plusieurs moteurs de recherche, comparer les résultats et s'interroger sur la pertinence des classements.
- ▶ Réaliser à la main l'indexation de quelques textes sur quelques mots puis choisir les textes correspondant à une requête.
- ▶ Calculer la popularité d'une page à l'aide d'un graphe simple puis programmer l'algorithme.
- ▶ Paramétrer un navigateur de manière qu'il interdise l'exécution d'un programme sur le client.
- ▶ Comparer les politiques des moteurs de recherche quant à la conservation des informations sur les utilisateurs.
- ▶ Effacer l'historique du navigateur, consulter les cookies, paramétrer le navigateur afin qu'il ne garde pas de traces.
- ▶ Utiliser un outil de visualisation tel que COOKIEVIZ pour mesurer l'impact des *cookies* et des traqueurs lors d'une navigation.

TABLE 7.2 — Web : compétences attendues chez les élèves.

Internet	
CONTENUS	CAPACITÉS ATTENDUES
Repères historiques	Définir les étapes du développement du Web.
Hypertexte	Maîtriser les renvois d'un texte à différents contenus.
Langages HTML et CSS	Distinguer clairement ce qui relève du contenu d'une page et de son style de présentation. Étudier et modifier une page HTML simple.
URL	Décomposer l'URL d'une page. Reconnaître les pages sécurisées.
Requête HTTP	Décomposer le contenu d'une requête HTTP et identifier les paramètres qui sont passés.
Modèle client/serveur	Inspecter le code d'une page hébergée par un serveur et distinguer ce qui est exécuté par le client et par le serveur.
Moteurs de recherche : principes et usages	Mener une analyse critique des résultats fournis par un moteur de recherche.
Paramètres de sécurité d'un navigateur	Maîtriser les réglages les plus importants concernant la gestion des cookies, la sécurité et la confidentialité d'un navigateur.

3 Réseaux sociaux

3.1 Découvrir la thématique

3.1.1 Ancrage dans le réel

A. POINTS-CLÉS

- ▶ La notion de réseau social est bien antérieure aux réseaux sociaux Internet, elle est étudiée dès la fin du XIX^e siècle, et est un champ d'étude humaine qui se fonde sur des modèles informatiques.
- ▶ On distingue la notion de communauté (personnes partageant des valeurs et des convictions communes) de la notion de société (personnes regroupées pour des raisons extérieures).



VIDÉO 7.7 — Réseaux sociaux.

- ▶ Le réseau social pour un individu a une taille maximale d'environ 150 personnes, au-delà, sa perception est celle d'une foule.
- ▶ Au sein d'un réseau, la chaîne des connaissances sociales liant une personne à n'importe quelle autre est généralement courte, toute personne est reliée à n'importe quelle autre par une chaîne de six maillons maximum; cela est vrai même pour les très grands réseaux.
- ▶ Il faut bien distinguer l'expérience de MILGRAM sur l'obéissance de celle sur les petits mondes et distinguer cette idée du concept mathématique éponyme.

B. MOTS-CLÉS

- ▶ Notion de réseau social.
- ▶ Routage dans les petits mondes.
- ▶ Notion de média social.

C. CE QUE DIT LE PROGRAMME

INTRODUCTION

Les réseaux sociaux sont des applications basées sur les technologies du Web qui offrent un service de mise en relation d'internautes pour ainsi développer des communautés d'intérêts.

IMPACTS SUR LES PRATIQUES HUMAINES

Le développement des réseaux sociaux introduit un nouveau type de liens sur le Web, qui ne relève pas de l'hypertexte : il s'agit de l'abonnement à des relations/des amis et de la possibilité de recommander de l'information en fonction du réseau ainsi constitué.

L'objectif annoncé des applications de réseautage social est de mettre les individus en relation les uns avec les autres. Quelle est la réalité ? L'*expérience de Milgram (1967)* semble indiquer la constitution de « petits mondes » où chacun est au plus à six liens de distance d'un autre. Peut-on éviter les phénomènes de communautés liés à des recommandations se renforçant les unes les autres pouvant aller jusqu'à un appauvrissement de la pensée critique ? Ces questions font référence au concept de *bonding* (renforcement de liens existants au sein d'un même groupe) versus *bridging* (construction de nouveaux liens non redondants).

Les affaires de fuite de données personnelles mettent en avant les questions liées aux modèles économiques des applications de réseautage social symbolisés par le slogan « quand c'est gratuit, c'est vous le produit ».

Les réseaux sociaux peuvent être le support d'un *harcèlement numérique*, par le biais de photographies partagées sans consentement ou impossibles à retirer, par la diffusion de fausses nouvelles, de dénonciations ou de calomnies. Des pratiques, des outils et des services permettent de se protéger, lutter et dénoncer de tels agissements.

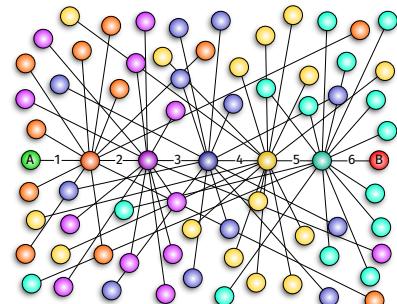


Illustration des six niveaux de séparation de la théorie des petits mondes.

3.1.2 Volet historique

À la fin des années 1890, Émile DURKHEIM et Ferdinand TÖNNIES ont laissé entrevoir l'idée de réseaux sociaux dans leurs théories et leurs recherches sur les groupes sociaux. TÖNNIES a fait valoir que les groupes sociaux peuvent exister en tant que liens sociaux personnels et directs



Visualisation d'un réseau social.

liant des individus partageant des valeurs et des convictions communes (« communauté»), ou des liens sociaux impersonnels, formels et instrumentaux («société»). DURKHEIM a donné une explication non individualiste des faits sociaux, affirmant que les phénomènes sociaux surviennent lorsque les individus en interaction constituent une réalité qui ne peut plus s'expliquer à l'aide des propriétés d'acteurs individuels.

Des développements majeurs dans le domaine sont observés dans les années 1930 par plusieurs groupes de psychologie, d'anthropologie et de mathématiques. En psychologie, dans les années 1930, Jacob L. MORENO a commencé l'enregistrement et l'analyse systématiques des interactions sociales en petits groupes, en particulier les salles de classe et les groupes de travail. En anthropologie, les travaux théoriques et ethnographiques sont à la base de la théorie des réseaux sociaux. En sociologie, les travaux de Talcott PARSONS (dans les années 1930) ont ouvert la voie à une approche relationnelle de la compréhension de la structure sociale.

Dans les années 1970, un nombre croissant d'érudits ont travaillé pour combiner les différentes pistes et traditions. Un des groupes était constitué du sociologue Harrison WHITE et de ses étudiants du département des relations sociales de l'Université de Harvard. Stanley MILGRAM, qui développa la thèse des « six degrés de séparation », était également au sein du département des relations sociales de Harvard à l'époque. Mark GRANOVETTER et Barry WELLMAN font partie des anciens élèves de WHITE qui ont élaboré et défendu l'analyse des réseaux sociaux.

À partir de la fin des années 1990, des sociologues, des politologues et des physiciens tels que Duncan J. WATTS, Albert-László BARABÁSI, Peter BEARMAN, Nicholas A. CHRISTAKIS et James H. FOWLER, ainsi que d'autres, ont analysé les réseaux sociaux. Les « traces numériques » et toutes les données émergentes disponibles sur les réseaux sociaux en ligne ont permis de créer de nouveaux modèles et méthodes d'analyse.

CE QUE DIT LE PROGRAMME

REPÈRES HISTORIQUES

- ▶ 1995 : CLASSMATES est l'un des premiers réseaux sociaux qui permettent aux étudiants de rester en relation.
- ▶ 2003 : apparition de MYSPACE, aujourd'hui en perte de vitesse et de LINKEDIN (racheté depuis par MICROSOFT), à vocation professionnelle.
- ▶ 2004 : apparition de FACEBOOK, d'abord réservé aux étudiants de l'université Harvard, puis ouvert au grand public en 2006.
- ▶ 2006 : apparition de TWITTER, qui permet l'échange de courts messages, limités au départ à 140 puis à 280 caractères (on parle de microblogage).
- ▶ 2009 : lancement de la messagerie instantanée WHATSAPP (rachetée depuis par FACEBOOK) qui se substitue à l'utilisation des SMS et MMS chez beaucoup d'utilisateurs.
- ▶ 2010 : arrivée d'INSTAGRAM (racheté depuis par FACEBOOK), qui permet le partage de photos et de vidéos.
- ▶ 2011 : début de SNAPCHAT qui permet, sur plateformes mobiles, le partage de photos et de vidéos, avec une limitation de durée.
- ▶ 2018 : on estime à 3,2 milliards le nombre d'utilisateurs actifs des réseaux sociaux.

En 2018, les réseaux sociaux utilisés en France sont états-uniens, toutefois il en existe bien d'autres : en Chine, par exemple, apparaît en 2009 l'application de microblogage WEIBO avec plus de

350 millions d'utilisateurs actifs en 2018; en 2012 naît l'application de messagerie WEIXIN (développée par TENCENT) qui compte en 2018 plus d'un milliard de comptes utilisateurs.

3.1.3 Explication des notions

A. IDÉES-FORCES

- ▶ Les réseaux sociaux s'étudient à l'aide de graphes, un modèle mathématique majeur (relations entre des personnes, entre des lieux, entre des actions, etc.).
- ▶ Beaucoup de problèmes concrets peuvent se ramener à des questions sur des graphes avec, selon les cas, des solutions connues ou non, calculables facilement ou pas.
- ▶ Ce qui nous est accessible sur les réseaux sociaux est présélectionné par des algorithmes de recommandation, par exemple en lien avec nos recherches précédentes. Cela biaise le résultat et tend à restreindre le champ de ce qui est accessible.

B. MOTS-CLEFS

- ▶ Notion de graphe.
- ▶ Systèmes de recommandation.

C. CE QUE DIT LE PROGRAMME

DONNÉES ET INFORMATION

Les différents réseaux sociaux permettent l'échange d'informations de natures différentes : textes, photos, vidéos. Certains limitent strictement la taille des informations, d'autres autorisent la publication, mais de façon limitée dans le temps. Certains permettent l'adjonction d'applications tierces (greffons ou *plug-ins*) qui peuvent ajouter des fonctionnalités supplémentaires.

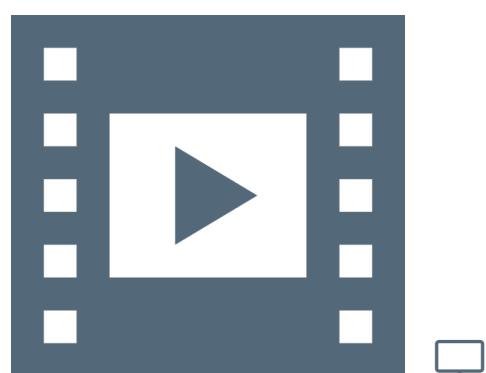
Toutes les applications de réseautage social utilisent d'importantes bases de données qui gèrent leurs utilisateurs, l'ensemble des données qu'ils sont amenés à partager, mais aussi celles qu'ils consentent à fournir (sans toujours le savoir), y compris sur leur vie personnelle.

ALGORITHMES ET PROGRAMMES

De très nombreux algorithmes sont mis en œuvre par les applications de réseautage social. Toutes les applications s'appuient sur la mise en relation avec des internautes membres du réseau, relations ou amis communs : des algorithmes opérant sur les *graphes* et les bases de données sont au cœur de ces services. À l'aide d'algorithmes de recommandation, les réseaux sociaux suggèrent aux utilisateurs des amis, des contenus, des annonces promotionnelles. Ils permettent aussi aux plateformes sociales d'étudier les comportements de leurs utilisateurs à des fins commerciales, politiques ou d'amélioration du service.

D. WEB-CONFÉRENCE

CLASS'CODE PAYS-DE-LOIRE a organisé des conférences qui abordent les sept thématiques du programme SNT. Leur objectif est de fournir une vue d'ensemble et de poser les bases nécessaires à l'appropriation de ces grands domaines sous deux angles définis par le programme :



VIDÉO 7.8 – Réseaux sociaux.

- une présentation historique (30 à 45 minutes) : grandes étapes de création/développement, acteurs majeurs, contexte général, histoire des idées et mise en contexte suffisamment fiable pour être réutilisée en cours;
- des exemples concrets de ce qui est étudié ou produit dans ces domaines et une réflexion sur les besoins et enjeux actuels et à venir à travers un débat (45 à 60 minutes) qui laisse la parole à des personnes travaillant dans ces champs de spécialité (chercheurs, enseignants, étudiants, entreprises...), que ce soit en recherche ou en entreprise.

La conférence sur les *réseaux sociaux* a eu lieu le 5 mars 2019 dans le cadre du projet CLASS'CODE à l'Université de Nantes. Elle a été retransmise en ligne et est maintenant disponible sur YOUTUBE.



© Gerd Altmann via Pixabay

RESSOURCES COMPLÉMENTAIRES

Se former

- ▶ Pour les enseignants : la conférence de Christine GAUBERT-MACON sur le Web, Internet et les réseaux sociaux, lors des formations nationales, accès par m@gistere (réservé aux enseignant-e-s).
- ▶ Présentation du Web au réseaux sociaux par Fabien GANDON.
- ▶ Une sélection d'articles du site INTERSTICES.

Ouvrages

- ▶ Fabien TARISSAN (2019) « Au cœur des réseaux : des sciences aux citoyens », Édition le Pommier. Il traite de science, plus particulièrement des réseaux (informatiques et autres), des

algorithmes qui y opèrent et de l'impact qu'ils ont sur notre manière de nous informer en ligne (*fake news*, publicité ciblée, données personnelles, etc.), et de l'économie sous-jacente.

Créer son cours

- ▶ Des animations intéressantes sur les réseaux sociaux et le web en général (les *cookies*...) sur le site d'Arte.
- ▶ La ressource librement utilisable « *isoloir* » propose une activité sur l'identité numérique et la liberté d'expression, à faire en classe en semi-autonomie, ces documents sont librement réutilisables.
- ▶ Dynamique des groupes de partage dans les réseaux sociaux : des recherches ont analysé comment les utilisateurs des réseaux sociaux font et défont les communautés au fil du temps, voir cet article en français : [De la difficulté de garder ses amis \(quand on a des ennemis\)!](#)
- ▶ Comment fonctionnent les systèmes de recommandation (article du Blog Binaire du Monde.fr) revue en ligne BINAIRE sur le site du Monde, 27 décembre 2016.
- ▶ [Les systèmes de recommandation : une catégorisation](#), Elsa Negre, INTERSTICES, 20 septembre 2018.
- ▶ [Dopamine](#), une websérie sur ARTE qui décortique notre addiction aux applis : TINDER, FACEBOOK, CANDY CRUSH, INSTAGRAM, YOUTUBE, SNAPCHAT, UBER, TWITTER, Léo Favier, Coproduction ARTE France, Les Bons Clients, Réseau Canopé, 8 vidéos.

3.2 Réaliser des activités

EXEMPLE D'ACTIVITÉS

Les activités proposées sont disponibles sous forme de fiches à télécharger (format ODT de la suite bureautique libre LIBREOFFICE).

- ▶ Faire des recherches sur différents réseaux sociaux.
- ▶ Possibilité de modéliser les réseaux sociaux par des graphes.
- ▶ Découverte de la notion de « petit monde ».

Fiche d'activité élève :

- ▶ Tracer un graphe de réseau social.

NOTE DE LA RÉDACTION

Toutes les fiches actuellement collectées sont disponibles à l'URL : <http://tinyurl.com/yx9qce8s> et on peut aussi proposer des activités.

CE QUE PROPOSE LE PROGRAMME

- ▶ Construire ou utiliser une représentation du graphe des relations d'un utilisateur. S'appuyer sur la densité des liens pour identifier des groupes, des communautés.
- ▶ Sur des exemples de graphes simples, en informatique débranchée, étudier les notions de rayon, diamètre et centre d'un graphe, de manière à illustrer la notion de « petit monde ».
- ▶ Comparer les interfaces et fonctionnalités de différents réseaux sociaux.
- ▶ Dresser un comparatif des formats de données, des possibilités d'échange ou d'approbation (bouton like), de la persistance des données entre différents réseaux sociaux.
- ▶ Analyser les paramètres d'utilisation d'un réseau social. Analyser les autorisations données aux applications tierces.
- ▶ Discuter des garanties d'authenticité des comptes utilisateurs ou des images.
- ▶ Lire et expliquer les conditions générales d'utilisation d'un réseau social.

► Consulter le site nonauharcelement.education.gouv.fr.

TABLE 7.3 — Réseaux sociaux : compétences attendues chez les élèves.

Internet	
CONTENUS	CAPACITÉS ATTENDUES
Réseaux sociaux existants	Distinguer plusieurs réseaux sociaux selon leurs caractéristiques, y compris un ordre de grandeur de leurs nombres d'abonnés. Paramétrier des abonnements pour assurer la confidentialité de données personnelles.
Modèle économique des réseaux sociaux	Identifier les sources de revenus des entreprises de réseautage social.
Rayon, diamètre et centre d'un graphe	Déterminer ces caractéristiques sur des graphes simples.
Notion de « petit monde » (MILGRAM)	Décrire comment l'information présentée par les réseaux sociaux est conditionnée par le choix préalable de ses amis.
Harcèlement numérique	Analyser l' article 222-33-2-2 du code pénal.

4 Enjeux environnementaux

L'enseignement des sciences numériques et technologie doit également permettre aux élèves de comprendre le poids croissant du numérique et les enjeux qui en découlent. Pour cette raison, ont été récoltées des ressources au sujet du numérique et de l'environnement afin d'offrir aux enseignants de la matière pour une prise de conscience par les élèves des impacts environnementaux du numérique, que ce soit pour la fabrication, la consommation et le recyclage.

Suite à la présentation de la thématique, une intervention de Françoise BERTHOUD est proposée issue des conférences *Comprendre et Agir*, Inria, juin 2018. Ensuite, des fiches de synthèse des ressources récoltées donnent quelques pistes de réflexion avec des exemples concrets et des schémas éloquents. Toutes les références à ces ressources sont données au fur et à mesure, ne pas hésiter pas à aller les consulter. Cette partie est en construction permanente au vu des impacts grandissants du numérique dans l'environnement.

4.1 Découvrir la thématique

AVERTISSEMENT

Attention, les données sur ce sujet sont plus ou moins disponibles, en général imprécises mais surtout très évolutives. Toutes les sources de cette séquence sont précisées.

Il est important de comprendre les impacts environnementaux du numérique, en pensant notamment aux trois facettes suivantes :

- fabrication (extraction des ressources, production, transport),
- utilisation (consommation électrique et de données),
- recyclage.

Qu'entend-on par numérique dans cette séquence ? On parle ici des équipements terminaux (PC, laptop, tablette, smartphone, mobile, etc.), des écrans (moniteurs), des serveurs et de leur environnement, des équipements réseaux passifs et actifs (filaire, Wi-Fi, GSM, xG, etc.) et de la télévision numérique ou connectée.

4.1.1 Ancrage dans le réel

A. POINTS-CLEFS

- Le secteur du numérique a un taux de croissance particulièrement important (+8 % par an). À l'échelle mondiale, il est responsable d'en-

viron 10 % de la consommation électrique, tout cela sans compter les objets connectés et les systèmes embarqués.

- ▶ Au niveau planétaire, les technologies du numérique sont responsables d'environ 4 % des émissions de gaz à effet de serre (GES), soit plus que celles engendrées par l'aviation civile !
- ▶ L'extraction et le raffinage des métaux puis la production des composants nécessaires à la réalisation des équipements électroniques ainsi que leur recyclage, lorsqu'il n'est pas fait dans les règles de l'art, sont aussi des sources considérables de pollution.
- ▶ Le numérique est à l'origine de gros problèmes sanitaires dans plusieurs pays africains et asiatiques où des sites de recyclages « informels » inondent les sols, air et eau de métaux lourds et de substances chimiques très polluantes.
- ▶ L'extraction des métaux est également responsable de conflits d'accès à l'eau et de conflits armés; par exemple en Afrique du Nord, en Amérique du Sud, en République Démocratique du Congo...
- ▶ Le développement des technologies numériques participe aussi de manière importante à l'épuisement de certaines ressources.
- ▶ Le recyclage représente un espoir pour amortir l'épuisement des ressources, mais actuellement les taux de récupération des métaux dans les appareils électroniques sont faibles : sur la cinquantaine de métaux présents dans un *smartphone*, dans le meilleur des cas seuls vingt sont recyclés.
- ▶ Sans remettre en cause les apports réels des technologies numériques dans la société, participant notamment à notre confort — à savoir communication, diffusion des savoirs, technologie GPS, etc. —, il est important de garder à l'esprit les coûts environnementaux associés (pollution, destruction des écosystèmes, réchauffement climatique, etc.) dont la croissance exponentielle s'ajuste sur notre propre consommation exponentielle du numérique.

Source : Plus d'ambitions climatiques ! Françoise BERTHOUD (CNRS), Jacques COMBAZ (CNRS) et Kevin MARQUET (Inria), Binaire, 17 juin 2019.

B. MOTS-CLEFS

- ▶ **Centre de données – Data center** — Un centre de données (en anglais *data center* ou *data centre*) est un lieu (et un service) regroupant des équipements constituants du système d'information d'une ou plusieurs entreprise(s) (*ordinateurs centraux*, serveurs, *baies de stockage*, équipements réseaux et de télécommunications, etc.) [W].
- ▶ **Effet rebond** — D'une manière très générale, l'effet rebond, encore appelé *paradoxe de Jevons*, peut être défini comme « l'augmentation de consommation liée à la réduction des limites à l'utilisation d'une technologie, ces limites pouvant être monétaires, temporelles, sociales, physiques, liées à l'effort, au danger, à l'organisation... ». Il en découle le corollaire suivant : les économies d'énergie ou de ressources initialement prévues par l'utilisation d'une *nouvelle technologie* [plus efficace ou performante] sont partiellement ou complètement compensées à la suite d'une adaptation du comportement de la société. Il a une grande importance pour l'établissement, l'évaluation et la mise à jour de stratégies et politiques énergétiques [d'après W]. Un exemple d'effet rebond : les batteries des *smartphones* gagnent en autonomie ce qui permet aux développeurs de proposer des applications plus gourmandes en énergie.
- ▶ **Streaming** — Très utilisé sur Internet et sur les réseaux de téléphonie mobile, le *streaming* permet la lecture d'un flux audio ou vidéo (cas de la vidéo à la demande) à mesure qu'il est diffusé. Il s'oppose ainsi à la diffusion par téléchargement de fichiers qui nécessite de récupérer

l'ensemble des données d'un morceau ou d'un extrait vidéo avant de pouvoir l'écouter ou le regarder [W].

- ▶ **Obésitel** — Néologisme désignant un logiciel qui utilise beaucoup de ressources physiques.
- ▶ **Obsolescence** — L'obsolescence est le fait pour un produit d'être dépassé et donc de perdre une partie de sa valeur d'usage en raison de la seule évolution technique (on parle alors d'*« obsolescence technique »*) ou de la mode (on utilise alors plutôt le mot « démodé »), même s'il est en parfait état de fonctionnement [W].



C. CE QUE DIT LE PROGRAMME

INTRODUCTION

L'enseignement de sciences numériques et technologie en classe de seconde a pour objet de permettre d'appréhender les principaux concepts des sciences numériques, mais également de permettre aux élèves, à partir d'un objet technologique, de *comprendre le poids croissant du numérique et les enjeux qui en découlent*. L'enseignement de sciences numériques et technologie aide à mieux comprendre les enjeux scientifiques et sociétaux de la science informatique et de ses applications, à *adopter un usage réfléchi et raisonné des technologies numériques* dans la vie quotidienne et à se préparer aux mutations présentes et à venir de tous les métiers. Cet enseignement a vocation à multiplier les occasions de mise en activité des élèves, sous des formes variées (exposés, travaux en groupe, mini-projets, productions individuelles ou collectives, etc.) qui permettent de développer des compétences transversales [...] comme faire un *usage responsable et critique* des sciences et technologies numériques.

IMPACTS SUR LES PRATIQUES HUMAINES

L'évolution des capacités de stockage, de traitement et de diffusion des données fait qu'on assiste aujourd'hui à un phénomène de surabondance des données et au développement de nouveaux algorithmes capables de les exploiter.

Les centres de données (*data center*) stockent des serveurs mettant à disposition les données et des applications les exploitant. *Leur fonctionnement nécessite des ressources* (en eau pour le refroidissement des machines, en électricité pour leur fonctionnement, en métaux et en eau pour leur fabrication) et *génère de la pollution* (manipulation de substances dangereuses lors de la fabrication, de la destruction ou du recyclage). De ce fait

les usages numériques doivent être pensés de façon à limiter la transformation des écosystèmes (notamment le réchauffement climatique) et à protéger la santé humaine.

4.1.2 Numérique : menace ou espoir pour l'environnement ?

Françoise Berthoud, conférences* Comprendre et Agir, INRIA, juin 2018.

* Le diaporama est téléchargeable : [Numérique versus environnement](#).

RÉSUMÉ DE L'INTERVENTION

Envisagé comme solution technologique à la transition énergétique et plus globalement aux questions environnementales, le numérique est largement promu depuis plus de dix ans, par les sphères politiques, industrielles, voire par les citoyens et les chercheurs eux-mêmes.

Quels effets aujourd'hui sur l'environnement pour quels impacts environnementaux ? Ne serait-il pas temps d'ouvrir les yeux sur une réalité qui a dépassé nos fantasmes collectifs ?

À PROPOS DE L'INTERVENANTE

Françoise Berthoud, ingénierie de recherche en informatique au CNRS et cofondatrice du groupe EcoInfo qu'elle dirige (créé en 2006), exerce une expertise reconnue sur la thématique des impacts environnementaux du numérique.

4.2 Impacts environnementaux du numérique

4.2.1 Aspects relatifs à la fabrication

A. DES MILLIARDS D'APPAREILS DANS LE MONDE

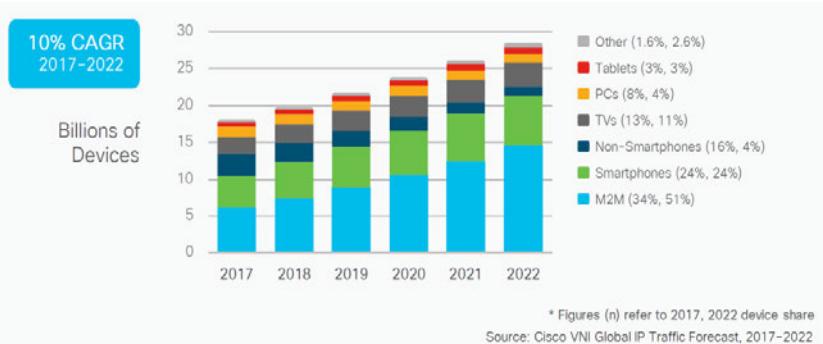
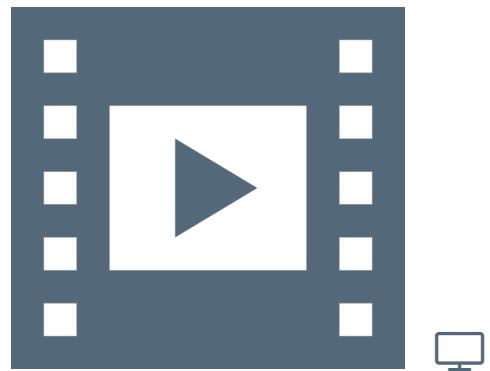


FIGURE 7.3 — Évolution et prospectives des équipements numériques dans le monde.

B. UTILISATION DE MÉTAUX RARES — OR, ARGENT, PALLADIUM...

- ▶ La fabrication¹ d'un smartphone fait intervenir une cinquantaine de métaux [Introduction aux impacts environnementaux du numérique, Kevin MARQUET, Françoise BERTHOUD, Jacques COMBAZ, 1024 numéro 13, Avril 2019] (et plus généralement 70 matériaux différents).
- ▶ « Au rythme actuel de production les réserves seraient de 15 ans pour l'étain, de 16 ans pour l'or, de 20 ans pour l'argent, ou encore de 39 ans pour le cuivre. Cependant, ces estimations doivent être prises avec précaution puisqu'elles ne tiennent compte que des réserves connues sans considérer les futures découvertes, les améliorations technologiques, ou les évolutions économiques ». [Introduction aux impacts environnementaux du numérique, Kevin MARQUET, Françoise BERTHOUD, Jacques COMBAZ, 1024 numéro 13, Avril 2019].
- ▶ « Jusqu'à récemment, l'amélioration technologique a permis une augmentation continue de l'efficacité énergétique d'extraction (d'environ 1 % par an), et ce malgré la diminution en concentration du minerai de



VIDÉO 7.9 — Numérique versus environnement.

1. L'article [Introduction aux impacts environnementaux du numérique](#) est disponible en consultation directe.

cuivre dans les exploitations. Du fait de limites physiques inhérentes aux processus d'extraction, cette tendance est en train de s'inverser et la quantité d'énergie nécessaire à l'extraction du cuivre devrait s'en-voler d'ici la fin du siècle selon certains spécialistes » [O. Vidal, conférence dans le cycle « Comprendre et Agir », INRIA ,2019]. « Ce schéma s'applique à plus ou moins long terme à toutes les ressources minières, ce qui limitera tôt ou tard nos capacités d'extraction ». [Plus d'ambitions climatiques! Françoise BERTHOUD, Jacques COMBAZ et Kevin MARQUET, BINAIRE, 17 juin 2019].

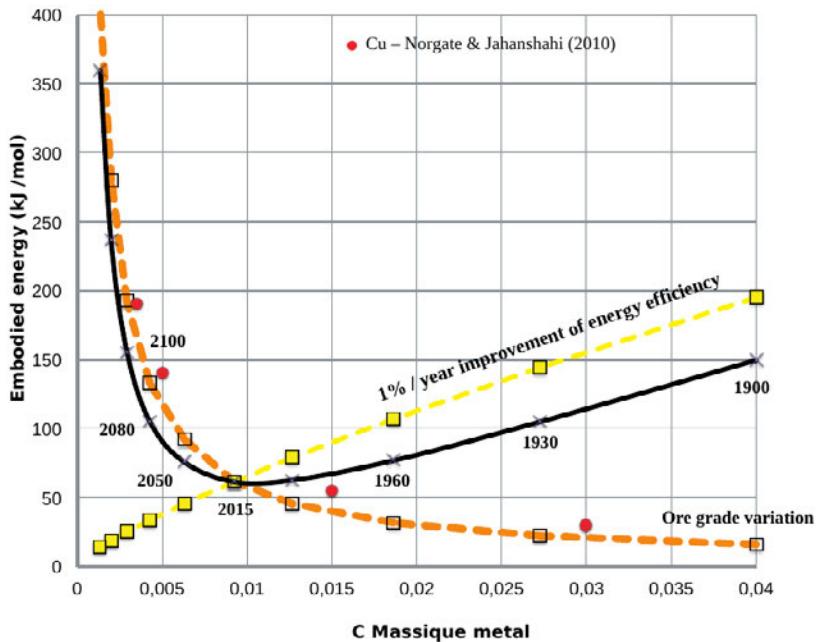


FIGURE 7.4 – Augmentation de l'énergie nécessaire à l'extraction du cuivre (d'après VIDAL, conférence du cycle « Comprendre et Agir », INRIA, 2019).

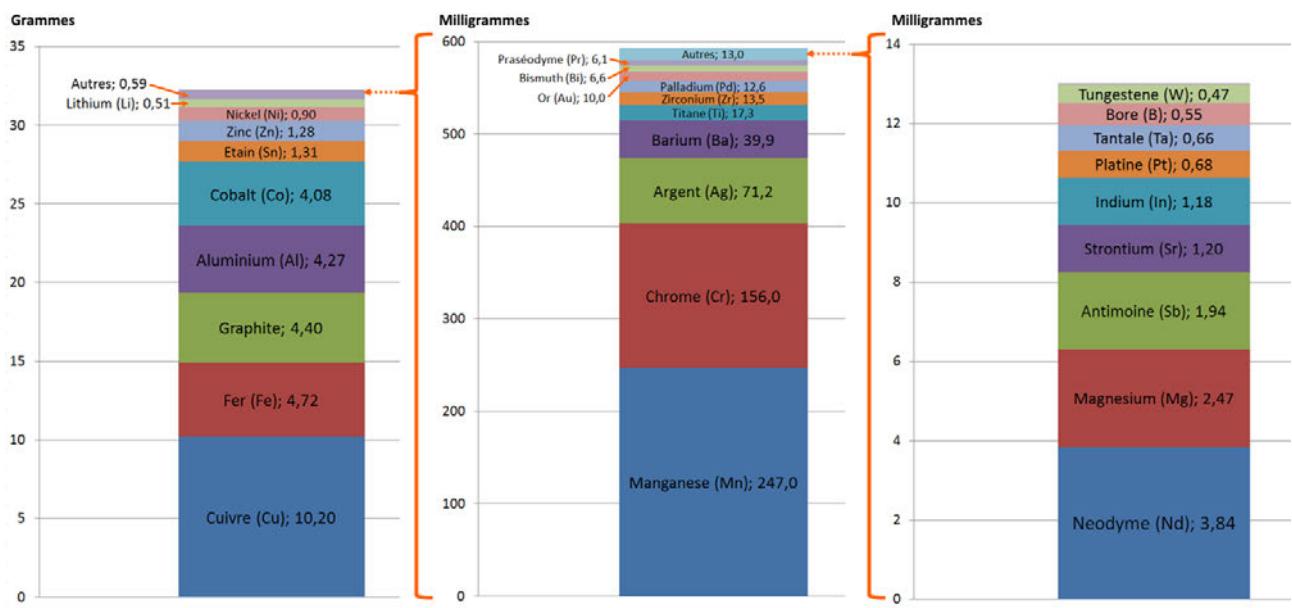


FIGURE 7.5 – Composition traditionnelle d'un smartphone (source ORANGE).

C. COÛTS ÉNERGÉTIQUES ÉLEVÉS DE PRODUCTION

« L'énergie est nécessaire non seulement pour le fonctionnement électrique des différents appareils électroniques (cf. figure 7.3 termi-

naux : ordinateur personnel fixe et portable, tablette, smartphone, téléphone portable traditionnel, box d'accès à internet, équipements audiovisuels connectés), mais aussi pour leur fabrication, leur transport et leur traitement en fin de vie » [MARQUET et al., 2019].

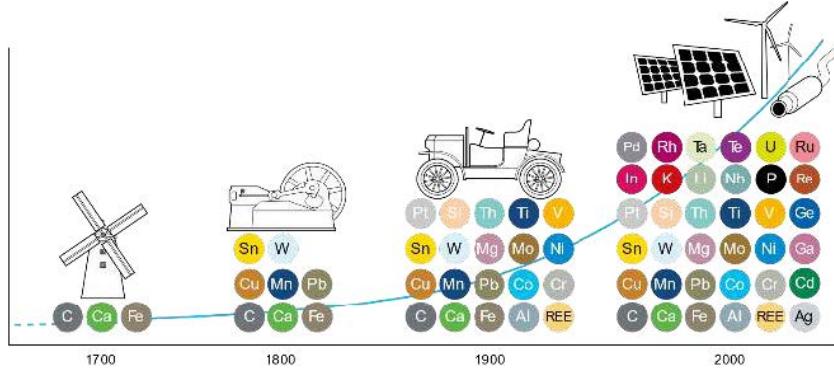


FIGURE 7.6 – Utilisation des métaux dans les TIC (d'après V. Zepf, 2014).

On voit en figure 7.7 que pour l'ensemble des appareils électroniques pris en compte, la phase de production correspond à 45 % de l'énergie totale utilisée durant le cycle de vie de ces appareils.

« Contrairement à une croyance bien installée, l'énergie consommée pendant la phase d'usage des équipements n'est pas prédominante. Le tableau [7.4] donne le ratio du coût énergétique de la phase d'utilisation par rapport à la phase de production de divers appareils ». À noter que l'usage d'Internet est absent de ces estimations. Par exemple, en France, la phase d'usage d'un smartphone n'est responsable que de 0,3 % des émissions de GES sur tout son cycle de vie (cf. figure 7.8) [MARQUET et al., 2019].

TABLE 7.4 – Énergie consommée et gaz à effet de serre (GES) induits par les phases de production et d'usages de quelques équipements.

Énergie consommée des équipements numériques					
RATIO USAGE/PRODUCTION	EN ÉNERGIE	EN ÉMISSIONS DE GES			
		FRANCE	EUROPE	USA	CHINE
Smartphone (2 ans)	6 %	0,3 %	2,6 %	4,5 %	6,2 %
Laptop (3 ans)	11 %	0,4 %	2,9 %	5,1 %	6,9 %
Serveur	—	50 %	—	—	—
TV connectée	—	1,1 %	8,9 %	15,0 %	19,5 %

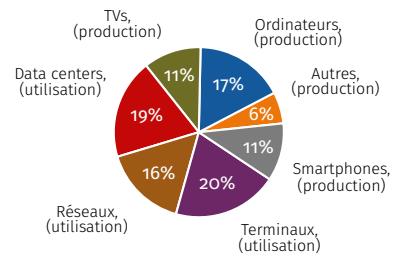


FIGURE 7.7 – Distribution de la consommation énergétique du numérique par poste en 2017 (d'après The Shift Project, 2018 – à partir de ANDRAE & EDLER 2015).

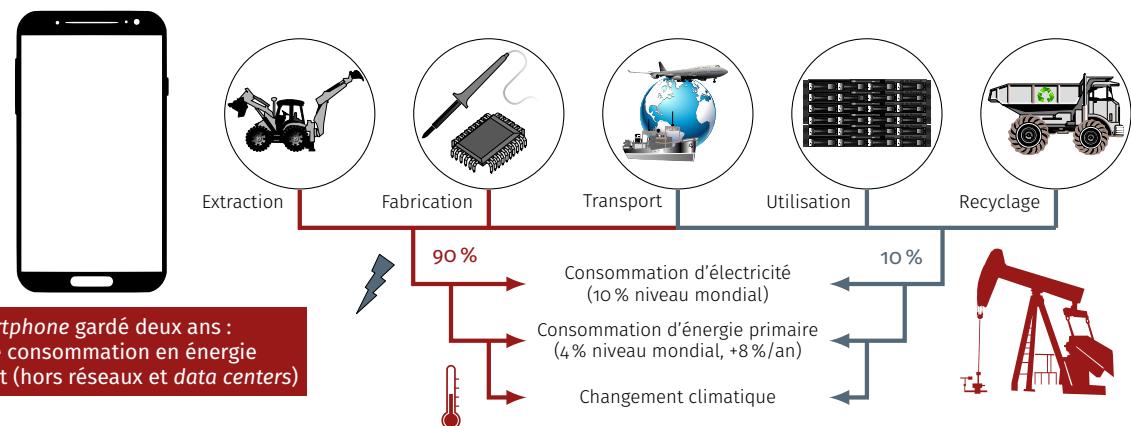


FIGURE 7.8 – Impact environnemental des TIC : énergie des smartphones (d'après Jacques COMBAZ et Françoise BERTHOUD, 2019).

4.2.2 Aspects relatifs à la consommation

A. CONSOMMATION ÉNERGÉTIQUE RELATIVE AU NUMÉRIQUE

« Aujourd'hui, les Technologies de l'information et de la communication (TIC) (équipements terminaux dont téléviseurs, équipements réseau, serveurs et leur environnement) sont responsables d'au moins 10 % de la consommation électrique mondiale, si l'on considère les phases de production et d'usage ». [Lean ict : Pour une sobriété numérique. Technical report, *The Shift Project*, 2018].

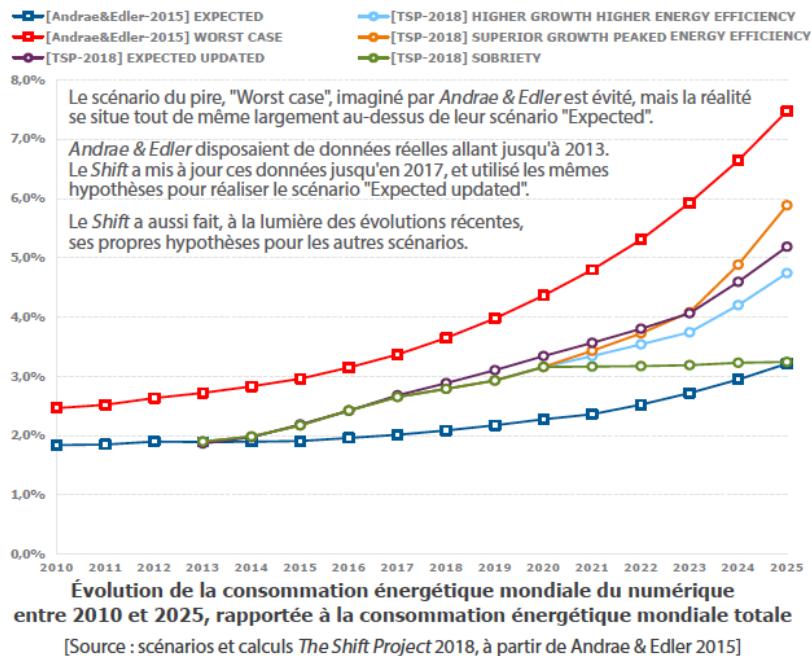


FIGURE 7.9 — Évolution de la consommation énergétique mondiale du numérique rapportée à la consommation énergétique totale (d'après *The Shift Project*, 2018 – à partir de ANDRAE & EDLER 2015).

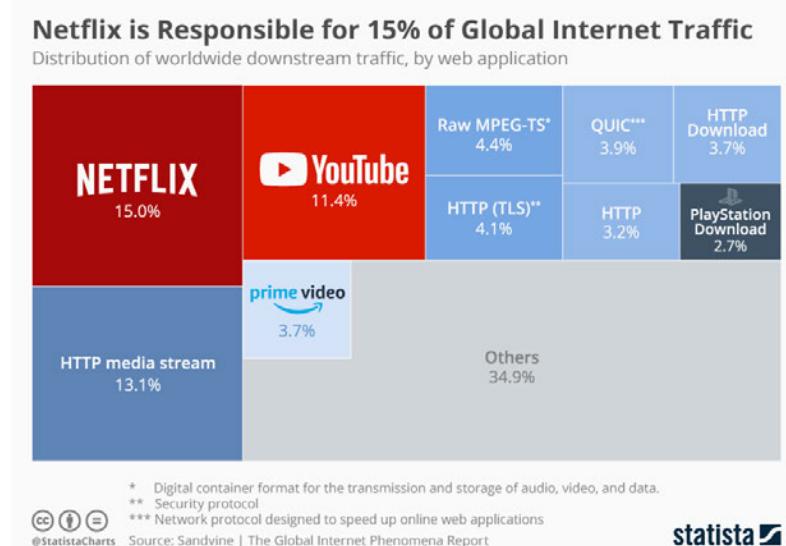


FIGURE 7.10 — Distribution au niveau mondial du trafic de téléchargement (chiffres octobre 2018).

B. STREAMING VIDÉO ET PRODUCTION DE CO₂

« L'association *The Shift Project* a fait ses calculs, rendus publics la semaine dernière : l'usage des NETFLIX et autres YOUTUBE a produit 306 millions de tonnes de CO₂ en 2018. Autant que l'Espagne. »

Dans le détail, la consommation de vidéos dites “online”, hébergées sur un serveur et transmises sur un terminal (*smartphone*, TV...) distant *via* un réseau, est à l'origine de 60 % du trafic de données numériques dans le monde l'an dernier. Inclus dans cette catégorie, les services à la NETFLIX et à la AMAZON PRIME comptent pour 34 % de ce total, suivis par les portails pornographiques (27 %), les plateformes telles que YOUTUBE et DAILYMOTION (21 %) et enfin les vidéos partagées sur les réseaux sociaux (18 %) ».

Source : Le streaming vidéo, une usine à CO₂, Frédéric MONFLIER, Informatique et Numérique, juillet 2019.

C. CONSOMMATION ÉNERGÉTIQUE DES CENTRES DE DONNÉES

En figure 7.11, une « ferme de serveurs » de FACEBOOK aux États-Unis. Un gros *data center* consomme 100 millions de watts (100 MW), soit un dixième de la production d'une centrale thermique. [Source : Le grand gâchis énergétique, 2018, Laure CAILLOCE, CNRS LE JOURNAL].

« Les *data centers*, ces usines de la donnée qui abritent des milliers de serveurs informatiques, sont-ils des gouffres énergétiques ? Le numérique, qui a pris une place inédite dans nos vies, a lui aussi une empreinte écologique. Dans son ensemble, le secteur du numérique engloutissait près de 10 % de la production électrique mondiale en 2015. Les *data centers* en accaparent 18 %, selon une synthèse publiée fin 2017 par l'association NÉGAWATT et reprise par le site GREENIT. À quoi servent ces centres de données et que penser de leur consommation d'énergie ? La révolution numérique est passée par là, entraînant dans son sillage de nouvelles pratiques, dont celle du *cloud computing* (ou “informatique dématérialisée”). D'ici 2021, la capacité de stockage des *data centers* devrait encore être multiplié par quatre, selon une étude de Cisco. En fait, le stockage des données est dans ce cas externalisé par des sociétés spécialisées, qui se chargent de ses aspects opérationnels et matériels. De quoi faciliter l'administration des serveurs, en les confiant à des professionnels... Et surtout réduire les durées d'interruption lorsque des pannes se produisent. Le numérique pesant désormais lourdement dans l'économie, on comprend qu'un site de e-commerce (par exemple) souhaite réduire au plus bas la durée où ses pages sont inaccessibles ». [Numérique et écologie : les *data centers*, des gouffres énergétiques ?, Sarah SERMONDADAZ, SCIENCES ET AVENIR, Mars 2018].



FIGURE 7.11 — Data center FACEBOOK.

i— RÉCUPÉRER LA CHALEUR DES DATA CENTERS — « En France, la consommation des *data centers* s'élevait à environ 3 TWh en 2015, soit davantage que la consommation électrique de la ville de Lyon, selon l'Union française de l'électricité (UFE). À quoi tient-elle ? Il faut bien entendu alimenter en électricité les nombreux appareils. Mais elle est principalement dissipée sous forme de chaleur lorsqu'elle passe dans un matériau conducteur, ce qu'on appelle "effet Joule". De ce fait, environ 50 % de la facture d'électricité d'un *data center*... Tient à la climatisation, comme l'expliquaient nos confrères de ACTU-ENVIRONNEMENT. Pour réduire leurs coûts, les *data centers* ont alors tout intérêt à maximiser le “*free cooling*”, ou refroidissement naturel en utilisant l'air frais extérieur. C'est pour cette raison que des géants comme FACEBOOK (par exemple) ont délocalisé leurs serveurs dans des pays nordiques comme la Suède ». [Numérique et écologie : les *data centers*, des gouffres énergétiques ?, Sarah SERMONDADAZ, SCIENCES ET AVENIR, Mars 2018].

ii— DES ÉQUIPEMENTS SURDIMENSIONNÉS — « Autre particularité du Web, son « hyperdisponibilité » : toutes les infrastructures sont dimensionnées pour absorber les afflux de données liés aux pics d'utilisation, soit quelques heures par jour à peine, et demeurent sous-utilisées le reste du temps... Cette « tyrannie » de l'utilisateur se re-

trouve jusque dans la conception des box Internet qui ne possèdent pas de bouton d'arrêt et fonctionnent jour et nuit. « Il faut une minute trente pour rallumer une box éteinte ; les fournisseurs d'accès estiment que c'est un temps beaucoup trop long pour les utilisateurs impatients que nous sommes devenus », explique Françoise BERTHOUD. Résultat : les box représentent à elles seules 1% de la consommation électrique française ». [Le grand gâchis énergétique, 2018, L. CAILLOCE, CNRS LE JOURNAL].

D. STOCKAGE ET ACHEMINEMENT DES DONNÉES

« Dans la phase d'usage, une grande partie de la dépense énergétique est le fait du stockage et de l'acheminement des données, c'est-à-dire des datacentres et des réseaux, sachant que le trafic réseau est en augmentation de 26 % par an ce qui est considérable. Il faut savoir qu'aujourd'hui 80 à 90 % du volume de données qui circule sur les réseaux est induit par la vidéo ».

« Dix minutes de streaming vidéo haute définition équivaut à utiliser à pleine puissance pendant cinq minutes un four électrique de 2000 W. L'ADEME estimait en 2011 que l'envoi d'un courriel contenant une pièce jointe de 1 Mo provoquait en moyenne l'émission d'environ 20 g de GES, soit l'équivalent libéré par la combustion du carburant nécessaire à la réalisation de 200 mètres en voiture » [MARQUET et al., 2019].

E. DES « OBÉSICIELS » TROP GOURMANDS

« La couche logicielle qui permet à tous ces équipements de fonctionner n'est guère plus optimisée. C'est le cas des applications pour smartphones développées à la va-vite pour pouvoir être mises rapidement sur le marché, qui consomment d'autant plus d'énergie qu'elles sont toujours ouvertes ». « La plupart des gens ne savent pas qu'en moyenne, trente-cinq applis tournent en permanence sur leur téléphone, qu'ils les utilisent ou pas », signale la chercheuse. Résultat, les batteries se vident en moins d'une journée, quand il suffirait de les éteindre en activant le mode économie d'énergie pour gagner jusqu'à plusieurs jours d'autonomie ». [Le grand gâchis énergétique, 2018, Laure CAILLOCE, CNRS LE JOURNAL].



« Cependant, toutes ces applications installées consomment des ressources de calcul et de stockage. Effectivement, même une application qui n'est jamais utilisée peut être programmée pour envoyer des notifications périodiques à des serveurs distants (des données de géolocalisation par exemple) et ainsi avoir une consommation énergétique non négligeable. Plusieurs techniques peuvent être mises en œuvre pour réduire l'impact des applications. Certaines relèvent de bonnes pratiques, notamment lorsqu'il s'agit d'être attentif et de personnaliser la configuration des applications. D'autres techniques s'inscrivent dans les domaines du génie logiciel et de l'éco-conception logicielle ». [Le syndrome de l'obésiciel : des applications énergivores, Françoise BERTHOUD, Éric DREZET, Laurent LEFÈVRE et Anne-Cécile ORGERIE, INTERSTICES, juillet 2015].

F. POLLUTIONS DIVERSES (EAU, SOL, AIR)

- ▶ Tarissement de l'eau.
- ▶ Érosion des sols.
- ▶ Fragmentation des territoires.

G. EFFICIENCE ÉNERGÉTIQUE

La loi de KOOMEY décrit une tendance à long terme dans l'histoire des ordinateurs. Selon cette loi, le nombre de calculs par joule d'énergie

dépensé double tous les dix-huit mois environ. Il s'avère que cette tendance a été remarquablement stable depuis les années 1950, le nombre de calculs par joule d'énergie dépensé ayant doublé environ tous les 1,57 ans. Cette loi énoncée par Jonathan KOOMEY aurait été énoncée comme suit : la quantité d'énergie dont une machine a besoin pour effectuer un nombre donné de calculs va diminuer d'un facteur deux chaque année et demi [W].

4.2.3 Aspects relatifs au recyclage

A. TRAITEMENT DES DÉCHETS (DEEE)

« Dans le monde, seulement 20 % (en poids) des déchets d'équipements électriques et électroniques (DEEE) sont traités par des filières de recyclage identifiées. En France, ce taux est porté à 50 %. Aujourd'hui, la difficulté à réparer des appareils de plus en plus complexes mais aussi les diverses sources d'obsolescence nous incitent à un renouvellement toujours plus rapide de notre matériel électronique. Ainsi, un smartphone est remplacé après deux à deux années et demi en moyenne (88 % sont changés alors qu'ils fonctionnent encore), trois à cinq ans pour un ordinateur portable, cinq ans pour un ordinateur fixe » [MARQUET et al., 2019].

B. RECYCLAGE : ESPORAI POUR L'ÉPUISEMENT DES RESSOURCES

« Dans les composants électroniques, les métaux sont très peu recyclés ». [Le grand gâchis énergétique, 2018, L. CAILLOCE, CNRS LE JOURNAL].

« Le recyclage représente un espoir pour amortir l'épuisement des ressources, mais actuellement les taux de récupération des métaux dans les appareils électroniques sont faibles. Porter le recyclage à 80 % (et même à 100 % !) permettrait de retarder cet épuisement dans le contexte de demandes en croissance exponentielle observé actuellement, mais pas de s'inscrire dans un processus d'économie circulaire ». [Plus d'ambitions climatiques ! Françoise BERTHOUD, Jacques COMBAZ et Kevin MARQUET, BINAIRE, 17 juin 2019].

C. OBSOLESCENCE

« La voracité en ressources matérielles des obésiciels contribue fortement à la chute de la durée moyenne d'utilisation des appareils électroniques (ordinateurs, smartphones, etc.) avant remplacement. En effet, ils ne sont pas capables de s'adapter à plusieurs générations successives de logiciels. C'est le phénomène de l'obsolescence : l'appareil est dépassé en raison de l'évolution des logiciels. »

- ▶ Au niveau du matériel : lorsque les pièces détachées nécessaires à la réparation d'un appareil ne sont plus disponibles, l'appareil doit être remplacé en totalité.
- ▶ Au niveau logiciel : incompatibilité de deux versions d'un logiciel, fin du support (par exemple, fin du support de WINDOWS XP depuis 2014 qui devient vulnérable aux failles de sécurité).

Source : Le syndrome de l'obésicel : des applications énergivores, Françoise BERTHOUD, Éric DREZET, Laurent LEFÈVRE et Anne-Cécile ORGERIE, INTERSTICES, Juillet 2015.

4.2.4 Autre impacts

A. IMPACTS SOCIO POLITIQUES

L'extraction des métaux nécessaires à la fabrication des équipements numériques est à l'origine de :

- conflits armés en République Démocratique du Congo ;

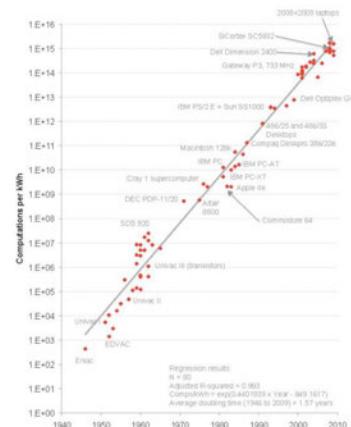


FIGURE 7.12 — Loi de Komey : nombre de calculs par kWh, de 1946 à 2009 (W).

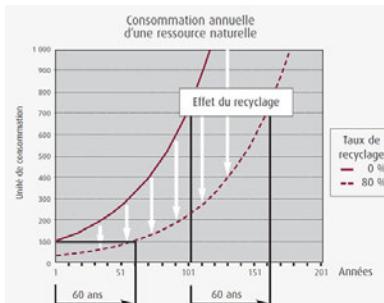


FIGURE 7.13 — Gain de temps permis par un fort taux de recyclage.

- conflits sociaux liés à l'accès à l'eau et à la pollution en Amérique du Sud;
- conflits miniers en Amérique latine.

B. GRANDES INÉGALITÉS D'ACCÈS AUX SERVICES NUMÉRIQUES

Il faut également avoir à l'esprit que l'accès aux services numériques à travers le monde est très inégal. Les cartes en figure 7.14 présentent les proportions de population ayant accès à l'Internet dans les différentes parties du globe en 2017 (voir également figure 7.2 pour 2016). Les écarts sont criants pour l'Afrique et une partie de l'Asie.

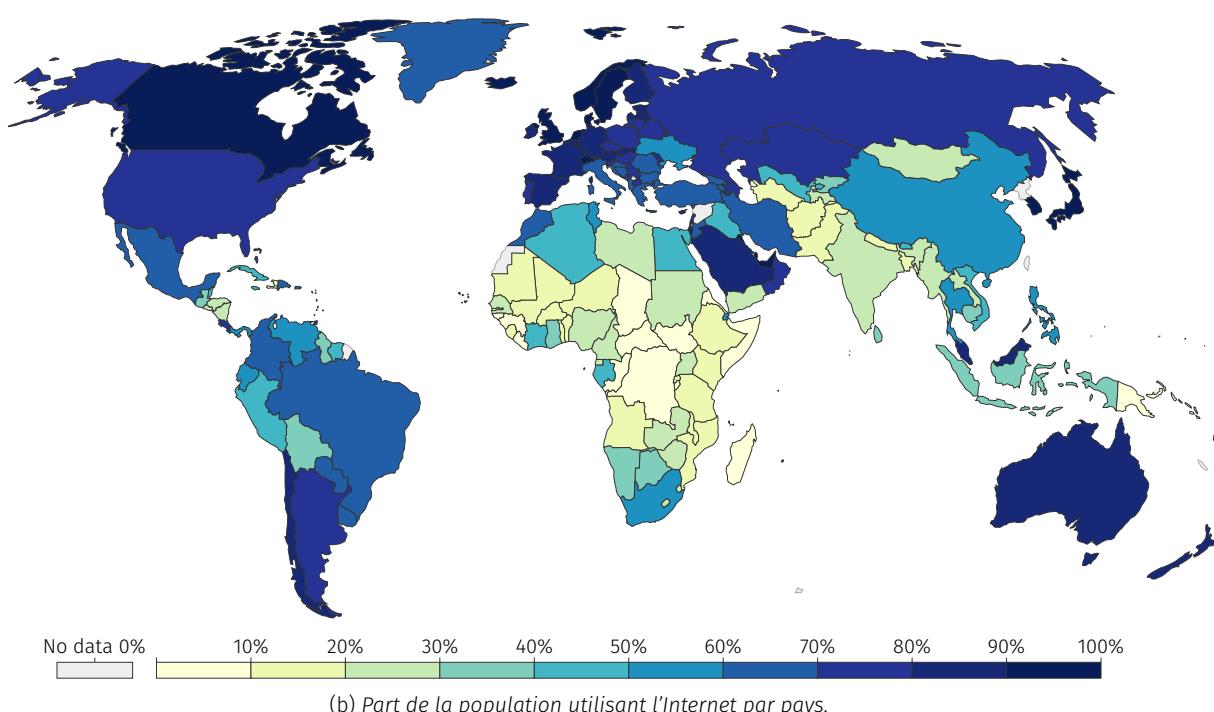
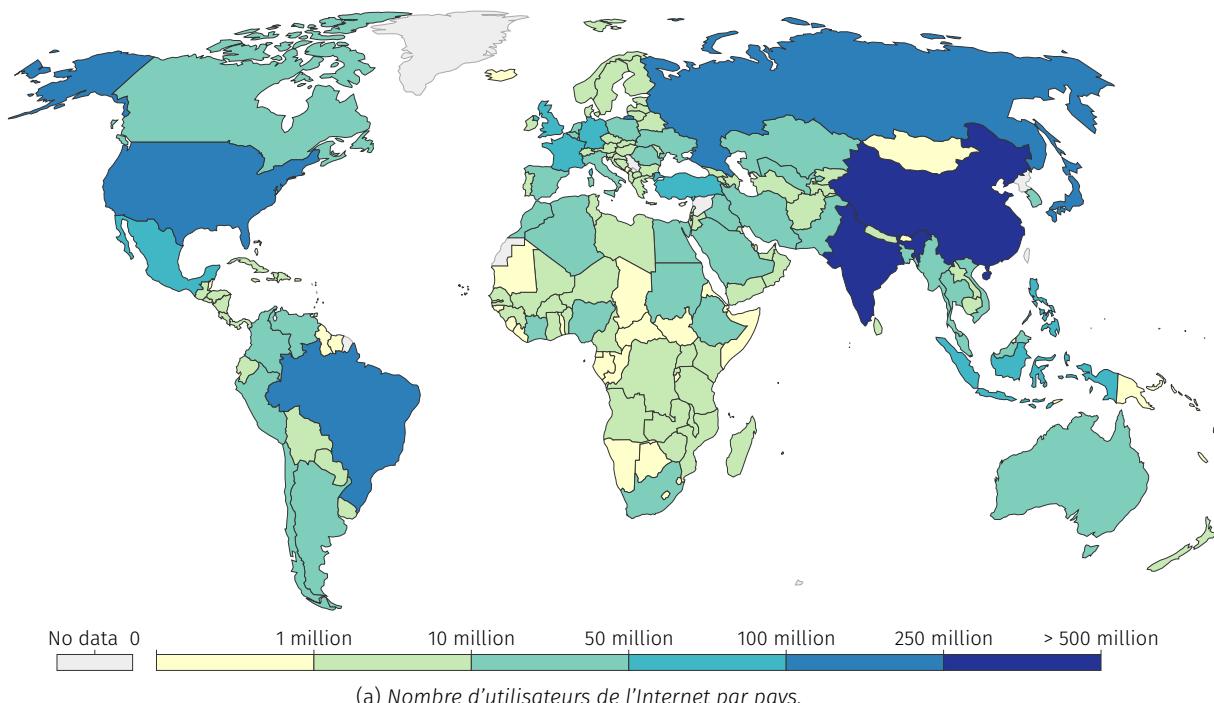


FIGURE 7.14 — Évaluation par pays de l'accès à Internet des populations en 2017 — les utilisateurs d'Internet sont les individus y ayant eu accès dans les trois derniers mois, tout équipement confondu (source ITU et Our World in Data).

C. IMPACTS SUR LA SANTÉ

Il faut noter également que l'usage du numérique a des impacts sur la santé, par exemple :

- addiction aux usages;
- troubles musculo-squelettiques...

4.2.5 Comment individuellement agir ?

« Choisir entre l'*utile* et le moins *utile* : l'auto-régulation des fournisseurs de services et le volontarisme des usagers ne suffiront pas : l'association "The Shift Project" plaide pour la mise en œuvre d'une régulation, à l'issue d'un débat sociétal, qui permet d'arbitrer entre ce qui est utile et ce qui l'est moins.

« Il ne s'agit pas d'être pour ou contre la pornographie, la télémédecine, NETFLIX ou les mails : il s'agit d'éviter qu'un usage précieux ne pâtit de la surconsommation d'un autre jugé moins essentiel » expliquent les auteurs de l'étude. Cette démarche n'irait pas à l'encontre du principe de neutralité du net, pour lequel la nature des contenus prime sur leur volume ». [Le streaming vidéo, une usine à CO₂, F. MONFLIER, Informatique et Numérique, juillet 2019].

« L'allongement de la durée de vie des équipements est une des principales préconisations pour diminuer les impacts environnementaux des TIC » [MARQUET et al., 2019].

On peut ainsi se tourner vers les « *low tech* » à l'opposé de « *high tech* ». « Le *low-tech* ou basse technologie est un ensemble de techniques simples, pratiques, économiques et populaires. Elles peuvent faire appel au recyclage de machines récemment tombées en désuétude. Ce sont des solutions techniques qui cherchent à être simples, bien pensées, dimensionnées et réparables. Elles sont issues d'une fabrication locale, favorisant l'emploi, plus proche de l'artisanat que de la production industrielle, voire de la prosommation » [W] (cf. LOW TECH MAGAZINE pour plus d'informations sur le sujet).



« Afin d'éviter de saturer les serveurs distants : supprimer ses vieux courriels (et surtout ceux contenant de volumineuses pièces jointes), ou encore limiter son utilisation des services de streaming en ligne (YOUTUBE, DEEZER, NETFLIX, etc.). Une vidéo comme Gangnam Style, visionnée 2,7 milliards de fois sur la planète, a consommé l'équivalent de la production annuelle d'une petite centrale, expliquait en 2017 Gary Cook, analyste pour l'ONG GREENPEACE, dans LE PARISIEN » [Numérique et écologie : les *data centers*, des gouffres énergétiques ?, Sarah SERMONDADAZ, SCIENCES ET AVENIR, Mars 2018].

« Éteindre les équipements non utilisés : ainsi, un serveur allumé mais inactif va consommer 100 W, contre 200 W au maximum s'il est en



NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style L^AT_EX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

plein calcul. La différence entre ces deux états pour le routeur sera de quelques pourcents à peine... Pourtant, personne ne songe à éteindre — au moins en partie — ces équipements aux heures creuses » [Le grand gâchis énergétique, 2018, Laure CAILLOCE, CNRS LE JOURNAL].

« L'impact environnemental de la transition numérique devient gérable si elle est plus sobre » [Lean ict : Pour une sobriété numérique. Technical report, The Shift Project, octobre 2018].

Bannir les courriels avec pièces jointes, limiter l'usage du *streaming* et réduire la qualité des vidéos regardées en *streaming*, et pourquoi pas instaurer un « jour sans numérique » par mois ou une soirée par semaine, de nombreuses actions simples peuvent être menées à l'échelle individuelle pour réduire l'impact du numérique sur l'environnement !

5 Que faire de ces ressources ? Autoévaluation

Les questionnaires à choix multiple* — QCM — à suivre clôturent le présent chapitre « Réseaux et environnement » et correspondent à chaque sujet qui y est abordé.

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

QUIZ 21 — INTERNET		5 POINTS
QUIZ 21.1 — ANALOGIE DU COURRIER POSTAL 1 POINT Quelle serait la meilleure analogie entre le courrier postal et le fonctionnement d'Internet ?		
<input type="radio"/> Il y a un facteur ou une factrice pour chaque lettre, qui prend ma lettre et l'amène directement à mon destinataire. <input type="radio"/> C'est comme le courrier postal, les lettres sont rassemblées dans des sacs, stockées, puis acheminées groupées d'une destination à l'autre, à heures fixes. <input type="radio"/> Il y a plein de factrices ou facteurs répartis sur le territoire qui font circuler les lettres de proche en proche selon leurs destinations. <input type="radio"/> Il y a un super-facteur central à qui on apporte toutes les lettres et qui les redistribue.		
VÉRIFIER	AFFICHER LA RÉPONSE	RÉINITIALISER
QUIZ 21.2 — NEUTRALITÉ DU NET 1 POINT Comment définir la « neutralité du Net » ?		
<input type="radio"/> C'est une notion technique : les opérateurs du Net doivent être de simples transmetteurs d'informations sans jamais trier, ou transmettre plus ou moins vite certains paquets d'informations. <input type="radio"/> C'est une notion sociétale qui impose aux personnes qui s'expriment de rester neutre, donc d'éviter d'exprimer leur opinion. <input type="radio"/> C'est une notion liée à la transmission électromagnétique à travers les câbles, qui ne doivent pas rayonner, donc rester neutres.		
VÉRIFIER	AFFICHER LA RÉPONSE	RÉINITIALISER

QUIZ 21.3 — DNS 1 POINT

À quoi sert un DNS ?

- Il permet d'associer un nom de domaine ou le nom d'une machine à son adresse, dite IP.
- C'est le mécanisme de routage, il permet de calculer les routes pour aller d'une machine à l'autre.
- C'est un mécanisme obsolète, il ne sert plus à rien.
- C'est un mécanisme de sécurité des échanges de données (DNS signifiant « *Data Secure Network* »).

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 21.4 — PROTOCOLE ET LANGAGE 1 POINT**

Quelle est la différence entre un langage de programmation informatique et un protocole informatique ?

- Aucune : dans les deux cas ce sont des instructions exécutées par une machine.
- Dans les deux cas, ce sont des instructions exécutées par une machine, mais il y a une différence importante : dans un protocole informatique le langage n'a pas besoin d'être complètement formalisé.
- Un protocole spécifie un ensemble de messages qu'il est possible d'envoyer ou de recevoir, et des actions à exécuter selon le message reçu
- La différence est qu'un protocole spécifie un algorithme qui doit rester secret pour sécuriser les communications.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 21.5 — PROTOCOLE ET TRANSMISSION 1 POINT**

Entre humains aussi (surtout quand la transmission est mauvaise) il y a des protocoles de transmission. Supposons qu'Ali doive transmettre cette série de chiffres à Barbara : 141 592 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105 820 974 944 592 307 816 406

Barbara et Ali communiquent de manière verbale, sur une ligne téléphonique. Disons que sur cette ligne, et dans un échange vocal entre Barbara et Ali, on a une erreur de transmission tous les quatre ou cinq paquets de chiffres environ (et donc en cas d'erreur Barbara notera un paquet de chiffres qui n'est pas celui prononcé par Ali).

Ali et Barbara ne parlent pas la même langue mais ils ont un vocabulaire commun. Ils ne peuvent prononcer que les mots « Allo », « ok », « non », « répète », « Terminé », « Merci », « Bye », ainsi que les chiffres.

Barbara et Ali cherchent avant tout un protocole sûr (donc qui permet des transmissions fiables). C'est leur contrainte prioritaire. Mais ils souhaitent aussi que ce protocole permette des échanges rapides (c'est leur deuxième contrainte, la fiabilité étant la priorité absolue).

Voici plusieurs protocoles possibles où ce que dit Ali est en italique et ce que répond Barbara en gras.

Protocole 1. Ici, Barbara valide ou non la réponse (en disant « ok » ou « répète »), paquet par paquet. En cas de doute ou de mauvaise réception, elle dit « répète » pour demander à Ali la retransmission du chiffre : *Allo ? Allo!* **ok** 592 **répète** 592 **ok** ... [etc.] ... **Terminé** **Merci** **Bye** **Bye**.

Protocole 2. Ici, Barbara renvoie systématiquement le paquet qu'elle vient de recevoir et Ali valide ou pas (en disant « ok » ou « non ») paquet par paquet : *Allo ? Allo! 141 141 ok 592 529 non 592 592 ok ... [etc.] ... Terminé Merci Bye Bye.*

Protocole 3. Barbara reçoit directement le message entier et la transmission est terminée : *Allo ? Allo! 141 592 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105 820 974 944 592 307 816 406 Terminé Merci Bye Bye.*

Protocole 4. Ici, Barbara reçoit directement le message d'Ali en entier puis elle le renvoie à Ali pour qu'il valide ou pas. Tant que le message entier n'est pas validé en retour par Ali, il faut recommencer : *Allo ? Allo! 141 592 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105 820 974 944 592 307 816 406 Terminé 141 529 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105 820 974 944 592 307 816 406 non ... [etc.] ... Terminé Merci Bye Bye.*

Parmi les protocoles décrits, lequel assure une transmission fiable et, en plus, ralentit le moins possible la transmission d'informations ?

Le protocole 1.

Le protocole 2.

Le protocole 3.

Le protocole 4.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER



QUIZ 22 — WEB ET INFORMATION

5 POINTS

QUIZ 22.1 — PARTIES D'UNE PAGE WEB 1 POINT

Quelle est la différence entre les parties `<head>` et `<body>` dans une page Web écrite en HTML5 ?

Le contenu de la partie `<head>` s'affiche au-dessus du contenu de la partie `<body>`.

Dans la partie `<head>` il y a les métadonnées du document et dans la partie `<body>` son contenu.

Cela n'existe plus dans les pages Web modernes.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 22.2 — WEB 2.0 VERSUS WEB 1.0 1 POINT

Que représente le Web 2.0 par rapport au Web 1.0 ?

2.0 signifie qu'on accède deux fois plus vite aux informations.

C'est une abréviation pour Web 2000 (Web à partir de l'an 2000).

Web 2.0 désigne l'idée de pouvoir lire les contenus et d'en créer.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 22.3 — ORDINATEUR VERSUS SMARTPHONE 1 POINT

On voudrait qu'une page Web s'affiche complètement différemment sur un ordinateur ou un smartphone. Quelle est la meilleure façon de faire ?

- Deux pages Web avec des balises HTML différentes, c'est plus sûr.
- C'est très facile avec un peu de programmation en JavaScript qui remodèle la page selon le support.
- C'est exactement le rôle de la feuille de style CSS.
- Il n'y a rien à faire, tout est géré par les navigateurs.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 22.4 — HAMEÇONNAGE — PHISHING 1 POINT**

Comment éviter le hameçonnage (*phishing*) qui consiste à me faire entrer mon mot de passe sur un site pirate pour me le voler et accéder frauduleusement à l'un de mes comptes ?

- Très facile, il faut regarder attentivement que la page affichée corresponde dans ses moindres détails à la page usuelle.
- Impossible, le mieux est de ne jamais se connecter sur un site.
- Il faut s'assurer que l'URL, surtout le nom de machine qui suit **http://** ou **https://**, soit exactement le même que le site usuel.
- Il y a une technique, entrer d'abord un mot de passe erroné pour voir la réponse du site Web, avant d'entrer le bon mot de passe.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 22.5 — URL — UNIFORM RESSOURCE LOCATOR 1 POINT**

Si on écrit **http://machine/chemin?paramètre=valeur** où... :

- « **http** » désigne le protocole pour accéder à cette ressource du Web;
- « **machine** » le nom ou adresse du serveur auquel on se connecte;
- « **chemin** » le chemin sur le serveur pour accéder à la ressource;
- « **paramètre=valeur** » un ou des éventuels paramètres...

Y a-t-il quelque chose de faux ?

- C'est faux : « **http** » ne désigne pas un protocole mais le simple fait qu'on accède à Internet.
- C'est faux : après « **http://** », ce n'est pas le nom ou l'adresse de la machine, mais une autre information.
- Les quatre items ci-dessus sont exacts.
- C'est ridicule : si on publiait le chemin vers la ressource sur le serveur, n'importe qui pourrait y accéder.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 23 — RÉSEAUX SOCIAUX****5 POINTS****QUIZ 23.1 — PETITS MONDES 1 POINT**

À quelle vitesse se propage une rumeur à travers un réseau humain ? Supposons que j'ai un ragot, top secret, que je ne raconte qu'à mes deux meilleurs camarades, qui eux aussi gardent le secret et ne le par-

tagent qu'avec deux autres camarades (différente-s), et ainsi de suite. En combien d'étapes la France entière sera-t-elle au courant ?

- Moins de 26 étapes.
- Moins de 128 étapes.
- Moins de 1 024 étapes.
- Moins de 65 536 étapes.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 23.2 — INFOX 1 POINT**

Comment vérifier une information vue sur Internet ?

- La question est sans fondement, si c'est sur Internet, c'est forcément vrai.
- Il suffit de voir si l'information est reprise à plusieurs endroits, si plusieurs sites la reprennent, elle est crédible.
- Il faut regarder la source de cette information, les sites crédibles citent toujours leurs sources.
- Il y a des sites de confiance qui répertorient les rumeurs pour aider à les démasquer.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 23.3 — RÉSILIENCE NUMÉRIQUE 1 POINT**

Que signifie la notion de résilience numérique, par rapport à une information négative qui nous touche ?

- Le fait qu'au bout d'un moment, les anciennes informations qui sont sur Internet s'estompent, enfouies sous les nouvelles.
- Le fait que si on est poursuivi par son passé ou une rumeur il suffit de changer d'identité numérique pour s'en libérer.
- Le fait d'intégrer cette information dans notre identité numérique pour en tirer le meilleur.
- Le fait de réussir à faire disparaître cette information d'Internet définitivement.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 23.4 — AMI OU ENNEMI ? 1 POINT**

Dans la classe, un élève est soit ami, soit ennemi, d'un autre, le lien est symétrique. Autrement dit : le diamètre du réseau est au plus deux. C'est à dire que deux personnes sont soit amis, soit ont au moins un ami commun. Mais quel est le plus petit nombre d'élèves tel qu'il est certain d'en trouver trois qui soient tou-te-s ami-e-s ou tou-te-s ennemi-e-s ? On pourra procéder par essais successifs (Difficile).

- Dès 5 élèves, il y aura au moins un trio d'ami-e-s ou ennemi-e-s.
- À partir de 6, quand on essaye toutes les possibilités cela marche.
- Non, cela marche uniquement à partir de 7.

Encore un petit effort, il en faut en fait 8.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 23.5 — RÉSEAUTAGE 1 POINT

Considérons un réseau social de type FACEBOOK dans lequel deux personnes peuvent être amies.

- ▶ Le diamètre du réseau est au plus 2, c'est-à-dire qu'il y a des personnes amies, et des amies d'amies, mais pas plus loin.
- ▶ Le degré du réseau est au plus 3, c'est-à-dire que chaque personne a au plus 3 amies.

Quel est le plus grand réseau (en termes de nombre de personnes) que nous pouvons construire ? On pourra consulter ce document pour asséoir sa réflexion.

Aucune limite, on peut toujours ajouter des personnes avec le bon nombre de liens.

Exactement 2 puissance 3, donc 8 personnes.

De 1 à 10 personnes, maximum, sauf pour le cas de 9 personnes qui est impossible.

Inutile de se poser la question, c'est un problème qui est résolu depuis fort longtemps.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 24 — NUMÉRIQUE ET ENVIRONNEMENT

7 POINTS

Les questionnaires suivants ont vocation à mettre en évidence des chiffres ou des faits afin de sensibiliser aux problématiques soulevées. Les contenus de ces quiz pourront, dans la même perspective, être utilisés avec des élèves.

Nota bene : Les sources des données utilisées et leur date sont indiquées dans les explications des questions qui sont affichées après avoir répondu.

QUIZ 24.1 — IMPACT ENVIRONNEMENTAL DES ÉQUIPEMENTS 1 POINT

Lorsqu'on parle de l'impact environnemental des équipements numériques (smartphones, ordinateurs, etc.), quelles sont les sources de pollution à considérer ?

L'extraction et le raffinage des métaux.

La production des composants nécessaires à la réalisation des équipements électroniques.

La consommation énergétique liée au café consommé quand on utilise tout ça.

La consommation énergétique liée au fonctionnement des équipements numériques.

La consommation énergétique liée à la circulation des données sur le réseau Internet.

La consommation énergétique des data centers (centres de stockage des données).

- Le recyclage des équipements numériques s'il n'est pas fait correctement.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 24.2 — CONSOMMATION D'ÉLECTRICITÉ 1 POINT**

Quelle part de la consommation mondiale d'électricité correspond au secteur du numérique ?

 0,1%. 1%. 10%. 40%.**VÉRIFIER****AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 24.3 — POLLUTION NUMÉRIQUE 1 POINT**

La pollution numérique, un fléau invisible...

Parmi les affirmations suivantes, sélectionnez toutes celles qui sont vraies — même si elles manquent de précisions.

- Si Internet était un pays, il serait le troisième consommateur mondial d'électricité, derrière la Chine et les États-Unis.

- Internet émet aujourd'hui 4% des gaz à effet de serre du monde, soit 1,5 fois plus que le transport aérien civil.

- Envoyer un mail avec une pièce jointe qui pèse environ 1Mo équivaut à laisser une ampoule de 60 watts allumée pendant 25min.

- Les mails professionnels d'une centaine de salariés pendant 1 an sont l'équivalent de 13 allers-retours Paris/New-York en avion.

- Nos propres cerveaux sont tellement sollicités par le numérique que leur consommation énergétique contribue au réchauffement climatique.

- Les vidéos en ligne équivalent à 1% des émissions mondiales de gaz à effet de serre.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 24.4 — CONSÉQUENCES ÉCOLOGIQUES ET SOCIÉTALES 1 POINT**

Au-delà de l'épuisement des ressources, quelles autres conséquences sont liées à l'extraction des métaux utilisés dans la fabrication des appareils électroniques ?

- Guerres civiles.

- Conflits d'usage de l'eau.

- Pollution de l'eau.

- Érosion des sols.

- Perte de biodiversité.

- Problèmes de santé.

Modification du pôle magnétique terrestre.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 24.5 — EMPLOI DES MÉTAUX 1 POINT

Combien de métaux différents sont utilisés dans un smartphone ?

Une dizaine.

Une vingtaine.

Pollution de l'eau.

Une cinquantaine.

Une centaine.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 24.6 — VIDÉOS EN STREAMING 1 POINT

Les vidéos en streaming correspondent à ...% du trafic des données numériques dans le monde (chiffres 2018).
Complétez la phrase avec l'un des chiffres suivants.

20%.

40%.

60%.

80%.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 24.7 — ACTIONS ET BONNES PRATIQUES 1 POINT

Une fois que l'on a pris conscience de l'impact du numérique sur l'environnement, comment peut-on agir au niveau individuel ?

Parmi les propositions suivantes, sélectionnez les actions qui permettent de limiter l'impact de l'usage des outils numériques.

Bannir les mails avec pièces jointes.

Supprimer ses vieux courriels.

Limiter son utilisation des services de streaming en ligne : YOUTUBE, DEEZER, NETFLIX...

Ne pas toujours opter pour un visionnage des vidéos en HD.

Éteindre les équipements non utilisés.

Réfléchir avant de renouveler un équipement qui marche encore.

Mettre en place un « jour sans numérique par mois ».

Se tourner vers les « low tech ».

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

IV

CODAGE EN LANGAGE PYTHON

255 | CHAPITRE 8
NOTIONS FONDAMENTALES EN PYTHON

299 | CHAPITRE 9
CONCEPTS ÉLÉMENTAIRES

NOTIONS FONDAMENTALES

DEPUIS L'AUBE DU MILLÉNAIRE, PYTHON S'EST IMPOSÉ comme un langage privilégié de programmation — notamment avec la version 2.0 datant de 2000 —, tout d'abord et faisant suite* au domaine de l'informatique, en contexte universitaire et de recherche puis, au fur et à mesure de la notoriété grandissante de ses qualités, en tant qu'outil d'apprentissage et de formation à la programmation informatique : universités, écoles d'ingénieurs et désormais enseignement secondaire.

* Par exemple, la distribution UBUNTU adopte PYTHON pour ses codes de configuration et « applets » (petites applications) dès sa première version d'avril 2004, mais également, la plupart des logiciels libres majeurs (GIMP, INKSCAPE, etc.) proposent rapidement la possibilité de développer des greffons — plug-ins en anglais — complémentaires à leur noyau principal, voire sont dans leur entièreté directement codés en PYTHON.

Les atouts intrinsèques de PYTHON qui peuvent expliquer ce constat sont relatifs aux faits que c'est un langage interprété — dit de script, autrement dit sans étape de compilation — que, par conception, sa syntaxe est dirigée par la lisibilité comme la facilité d'accès pour l'utilisateur, que c'est un langage de programmation orientée objet (Poo), qu'il est disponible pour tout système d'exploitation — WINDOWS, MACOS et LINUX — et enfin, non des moindres, que sa licence de diffusion est libre, y compris pour des applications à vocation commerciale.

Si dans ce Mooc et par conséquent ce document, l'acceptation originelle de PYTHON par la communauté scientifique est abordée au moyen d'illustrations concrètes de son utilisation (voir Partie IV, « Codage en langage Python »), son volet pédagogique en est bien entendu la motivation essentielle et le propos principal : ne pas mettre la charrue avant ses fameux bœufs...

1 Écosystème et environnement PYTHON

Cette approche consacrée à PYTHON 3 — à savoir que la version 2 du langage¹ a été plusieurs fois prorogée en coexistence avec la version 3 pour atteindre un basculement arrêté en 2020 — débute par les présentation et mise en œuvre d'un environnement de travail fonctionnel.

Sont ainsi évoqués en premiers lieux les généralités, l'organisation du matériel pédagogique puis les outils qui sont nécessaires de configurer pour la programmation avec PYTHON. La référence adoptée pour ce document, les exemples et exercices, est la version 3.6 de PYTHON.

SOMMAIRE

1 Écosystème et environnement

- 1.1 Organisation du cours 256
 - 1.1.1 Présentation ■ 1.1.2 Pourquoi PYTHON ?
- 1.2 Outils de développement 260
 - 1.2.1 Interpréteur PYTHON ■ 1.2.2 IDLE
- 1.3 Documentation et code PYTHON 265
 - 1.3.1 « Notebooks » JUPYTER ■ 1.3.2 Extension LATEX PYTHONTEX
- 1.4 Synthèse et compléments 269
 - 1.4.1 Modes d'exécution PYTHON ■ 1.4.2 Exercices d'application

2 Variables, objets et typage

- 2.1 Nommage et mots-clefs 277
 - 2.1.1 Types, objets et méthodes ■
 - 2.1.2 Référencement et variables ■
 - 2.1.3 Typage dynamique et gestion de la mémoire
- 2.2 Compléments et application 281
 - 2.2.1 Mot-clefs PYTHON ■ 2.2.2 Gestion de la mémoire ■ 2.2.3 Typages statique versus dynamique

3 Types numériques

- 3.1 Généralités 285
 - 3.1.1 Quatre types numériques ■
 - 3.1.2 Manipulation comme calculette ■
 - 3.1.3 Affections et opérations
- 3.2 Compléments et application 291
 - 3.2.1 Précision des calculs flottants ■
 - 3.2.2 Opérations bit à bit (*bitwise*) ■
 - 3.2.3 Exercices

4 Que faire de ces ressources ? Quiz

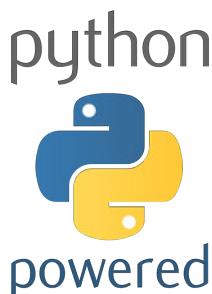
1. Actuellement en version 2.7 et définitivement figée sans autre révision.

1.1 Organisation du Mooc



VIDÉO 8.1 — *Organisation du cours.*

2. La partie sur PYTHON du manuel est extraite du Mooc « PYTHON :des fondamentaux aux concepts avancés du langage » et ne reprend que les deux premières semaines du tronc commun.



En termes de contenus et d’organisation, ce Mooc se présente en deuxième édition. La mouture actuelle fait en effet suite à l’apprentissage de PYTHON proposé entre 2014 et 2016, qui se fondait alors sur la version 2.7 du langage. Entre-temps, l’écosystème PYTHON a largement versé en version 3 — noyau comme multiples bibliothèques applicatives associées ou, selon le terme consacré, modules.

1.1.1 Présentation

L’exposé² est articulé autour de deux grandes parties : un tronc commun réparti sur six semaines de travail et couvrant les fondamentaux d’apprentissage de PYTHON puis, de manière optionnelle, quelques approfondissements correspondant à des utilisations spécifiques de PYTHON et certains modules particuliers.

En terme de tronc commun, l’objectif est d’aborder les fondamentaux du langage, c'est-à-dire les types de base, les fonctions, les structures de contrôle, les modules, les espaces de nommage et la programmation objet. On insistera sur les traits marquants de PYTHON, à savoir :

- un langage très lisible;
- pour lequel tout est objet;
- qui, pour permettre une bonne compréhension du code, est associé à la notion de liaison lexicale mais également à celle de typage dynamique;
- et dont l’approche serait incomplète sans parler des itérations et des espaces de nommage.

En ce qui concerne la partie optionnelle, trois directions sont proposées pour approfondir le cours de base. Une première semaine consacrée aux outils dits de « *data science* », une deuxième pour aborder la « programmation asynchrone » — sujet très innovant et très intéressant que vous êtes convié à consulter —, puis enfin, en dernière semaine, un complément sur les « *sujets avancés* », à savoir l’approche des métaclasses, des décorateurs et des techniques de programmation qui n’ont pas été jugés utiles d’inclure au tronc commun.

Il est nécessaire de mentionner que des deux premières semaines additionnelles proposées, il serait possible sans aucun problème de conduire un MOOC à part entière. Il s’agit donc d’une toute petite introduction pour savoir que cela existe et susciter suffisamment d’intérêt pour gratter au-delà de la surface et fournir les moyens d’approfondir tous ces sujets en autonomie.

Que les modules optionnels soient suivis ou non, l’enjeu est bien entendu de posséder le bagage nécessaire pour écrire du code propre, mais également de lire celui d’autres programmeurs. C’est un peu comme une langue étrangère, sans la lecture de code, on n’apprend pas les différents paradigmes et idiomès. C’est pour cette raison que, parfois, les informations données vont au-delà du strict nécessaire.

Par ailleurs, la perspective est également de pouvoir choisir et utiliser à bon escient les bibliothèques tierces de l’écosystème PYTHON. Un des atouts majeurs de PYTHON est sa polyvalence.

En termes de linguistique, évidemment le propos est en français, ainsi que les exercices et les supports pédagogiques. Néanmoins, il est fait appel de nombreuses fois à des documents en anglais, pour la raison que le langage et, plus généralement, les documentations et la quasi intégralité des codes existants sont en anglais.

En outre, certains termes sont parfois impossibles à traduire. Il a été convenu de conserver leur expression anglophone, permettant aussi, dès le départ, une recherche sur les moteurs de recherche plus aisée.

1.1.2 Pourquoi PYTHON ?

Avant de rentrer à proprement parler dans la technique, quelques mots sur les raisons qui expliquent que PYTHON soit un langage très agréable à utiliser et qui peut servir à de nombreuses d'applications.

PYTHON possède une foultitude de bonnes propriétés dont à l'usage la principale : la lisibilité. Pour s'en convaincre, on peut conduire un comparatif avec un certain nombre d'autres langages généralistes. À titre uniquement indicatif, ce parallèle est conduit avec les langages de programmation C++, JAVA et JAVASCRIPT.

A. COMPARAISON ENTRE LANGAGES DE PROGRAMMATION

Si l'on compare la manière de construire une boucle sur une liste en C++ et en PYTHON, cela peut ressembler aux codes 8.1.

En PYTHON, le code est déjà beaucoup plus court, moins touffu et il n'y a pratiquement aucun sucre syntaxique : pas de `begin`, de `end`, d'accolades et autres choses du même acabit.

En fait, le langage a pris le parti d'avoir une syntaxe orientée sur la présentation. Les deux phrases `print` et `manage` font partie du même bloc. L'œil humain le capte immédiatement, il n'y a pas besoin de rajouter des accolades ou des séparateurs du même genre.

La première chose importante à noter est donc que PYTHON possède une syntaxe articulée sur la présentation.

CODE 8.1 – Comparaison d'une boucle réalisée en C++ et en PYTHON.

C++ <pre>// parcourir une liste for (Cell *cell = cells.start; cell != NULL; cell = cell.next) { print(cell); manage(cell); }</pre>	 # parcourir une liste <pre>for cell in cells: print(cell) manage(cell)</pre>
--	---

La deuxième comparaison est conduite avec le langage JAVA. L'exemple pris ici est totalement pathologique ; un programme simple qui affiche « *Hello, world!* » que l'on trouve dans toutes les introductions des livres de programmation.

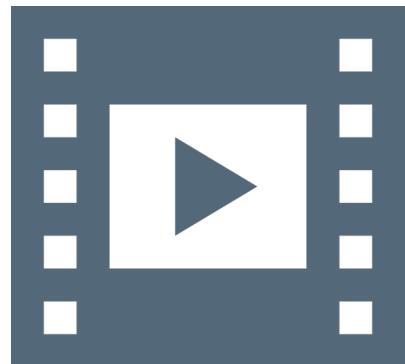
CODE 8.2 – Comparaison d'un programme « Hello world! » en JAVA et en PYTHON.

 <pre>public class HelloWorld { public static void main (string[] args) { System.out.println("Hello world!"); } }</pre>	 print("Hello world!")
--	--

En JAVA, c'est un peu abscons et assez long à saisir, il y a beaucoup de bavardage, de texte standard et passe-partout — *boilerplate* en anglais.

En PYTHON, on se retrouve avec un programme bien plus concis qui ne fait qu'une seule ligne. Évidemment, c'est un exemple pathologique, mais essentiel parce qu'il illustre le fait que certains langages comme JAVA ont une approche dogmatique.

En JAVA, il a été décidé qu'il fallait absolument que tout soit dans une classe, fort bien mais du coup, dans certaines situations, on se retrouve à devoir coder des parties de cette manière alors que cela ne se justifie pas. *A contrario*, PYTHON est un langage qui est totalement pragmatique ; cet aspect sera abordé ultérieurement.



VIDÉO 8.2 – Choix de PYTHON.



Une troisième comparaison rapide est menée avec JAVASCRIPT. Cette fois le code calcule la liste des carrés d'un certain ensemble donné. Ici encore, le code Python est plus ramassé et plus lisible.

CODE 8.3 — Codage du carré des éléments d'une liste en JAVASCRIPT et en PYTHON.

JS

```

1 // Carrés d'une liste
2
3 function squares(l) {
4     return l.map(function(x) {return x*x;})
5 }
```



```

1 # Carrés d'une liste
2
3 def squares(l):
4     return [x*x for x in l]
```

Il est volontiers admis que les exemples présentés sont totalement arbitraires, néanmoins, il en ressort deux aspects essentiels à retenir concernant PYTHON :

1. la facilité d'accès est volontairement considérée comme première perspective du langage : encore une fois, PYTHON est très facile à lire. Il est apparu fondamental au concepteur de PYTHON — Guido van Rossum (voir infra) — de proposer un langage avec lequel on peut facilement échanger, soit pour lire le code de quelqu'un d'autre, soit pour travailler avec quelqu'un d'autre. La lisibilité n'est donc pas le fruit du hasard, mais vraiment un choix délibéré dès la conception originelle du langage ;
2. les partis pris sont également guidés par le *pragmatisme*, par opposition à l'illustration donnée pour JAVA. À aucun moment dans la conception de PYTHON, des orientations ont été décidées en imposant une propriété. En revanche, face à un problème, le maître-mot est : quelle est la meilleure manière de régler ce problème ? Donnons-nous les moyens de trouver la solution la plus élégante pour que l'utilisateur ait le langage le plus agréable et le plus à même de résoudre ses problèmes.

B. HISTORIQUE SUCCINCT ET STABILITÉ



© Michael Cavataio

Guido VAN ROSSUM en 2019.

Cela peut paraître étonnant, mais PYTHON est relativement ancien, dont la première version publiée remonte à février 1991. Le langage est entré en version 1.0 en janvier 1994 puis en version 2.0 en octobre 2000. À partir de cette date, son essor n'a fait que s'amplifier.

Inutile de dire qu'à ces débuts le langage ne ressemblait pas exactement à ce qu'on peut utiliser aujourd'hui mais, si le gage de l'ancienneté est synonyme de qualité, PYTHON en bénéficie en ayant pas mal d'heures de vol derrière lui...

Une importante rupture est apparue entre la version 2 et la version 3 (décembre 2008). Tout le monde s'accorde à dire que ce fut relativement douloureux, mais nécessaire compte-tenu de, principalement, la manière dont étaient gérées les chaînes de caractères et les encodages. Ces questions seront abordées dans le cours. Depuis la version 3, évidemment, les choses sont redevenues totalement compatibles.

Ainsi, le langage a une histoire conséquente, qui n'a connu qu'une seule rupture de compatibilité en quelques trente ans, ce qui, parallèle à d'autres langages, est tout à fait raisonnable.

Cette évocation permet de mentionner une qualité indéniable de PYTHON : sa stabilité. En effet, la dernière version 2.7 de Python a été maintenue jusqu'en 2020, alors qu'elle était initialement censée devenir obsolète en 2010, lors de la sortie de la version 3.

À l'instar de tous les langages « mainstream », PYTHON tient à cœur de rester compatible pendant une très longue durée.

C'est l'occasion d'aussi noter la pérennité de la bibliothèque standard. Cette bibliothèque — library en anglais — comporte l'ensemble

des utilitaires qui sont empaquetés avec PYTHON. Cela fait partie du langage au sens où cela s'avère intégré à l'installation du langage, mais surtout, que la maintenance est concomitante à celle du langage. En utilisant un morceau de la bibliothèque standard, chacun est assuré de pouvoir l'utiliser dans la durée.

C. PORTABILITÉ ET PERFORMANCES

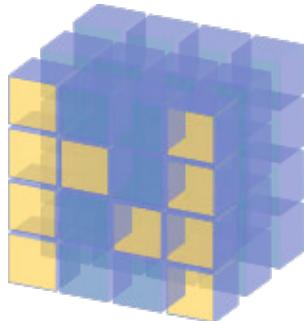
Un atout supplémentaire de PYTHON est sa *portabilité*. Autrement dit, un même code va pouvoir fonctionner indifféremment sur une plateforme WINDOWS, MACOS ou LINUX. Il s'érite même comme fondation logicielle de certains matériels comme le RASPBERRY PI.

PYTHON est un langage multi-facettes qui dispose d'une énorme base de données de code³ existant. Au-delà des grands domaines déjà signalés, il est par exemple possible d'écrire un site Web (cf. [Django](#)) ou, si besoin se fait sentir, de causer à sa porte de garage; certainement quelque chose est écrit quelque part pour le faire.

Une critique assez fréquemment formulée à propos de PYTHON est relative à ses performances. En fait, la plupart du temps, les problèmes qui ont besoin de grosses quantités de cycles⁴ ont été réglés en emballant — *wrapping* en anglais — du code compilé, typiquement écrit en C/C++, avec une interface PYTHON. Cela s'avère par exemple le cas avec la bibliothèque NUMPY, dévolue à la manipulation de tableaux.

En fin de compte, on obtient avec PYTHON un langage où les types de bases sont très puissants, dont les dictionnaires — abordés plus loin —, les ensembles ou les tableaux avec NUMPY. Par ailleurs, la gestion de la mémoire est automatique ; il n'est pas nécessaire de s'attacher à libérer la mémoire. À l'instar de JAVA, un *garbage collector* — ramasse-miettes en informatique et littéralement un éboueur — s'en occupe.

Tous ces éléments et le fait que PYTHON soit un langage interprété amènent à des développements extrêmement rapides.



3. Calcul scientifique :

- ▶ NUMPY, MATPLOTLIB, SCIPY, SYMPY.

Traitement de données :

- ▶ PANDAS, SCIKIT-LEARN.

Internet/Web :

- ▶ DJANGO, notebooks JUPYTER.



4. Cette critique peut se généraliser à tous les langages interprétés — même commerciaux comme MATLAB — lorsque, par exemple, il s'agit de réaliser de nombreuses boucles, comme en simulation numérique. Cet inconvénient reçoit le même traitement : en coulisses il est fait appel à des codes compilés pour obtenir des performances comparables avec PYTHON.



D. LICENCE ET GOUVERNANCE

Les droits sont détenus par la *PYTHON Software Foundation*, organisation à but non lucratif. La licence de PYTHON est très permissive, plutôt à rattacher aux licences BSD. Il possible de faire à peu près tout ce que l'on veut, y compris à des fins commerciales.

En terme d'évolutions, parce qu'il y a encore des évolutions aujourd'hui — comme la programmation asynchrone évoquée dans le cours —, PYTHON est un langage très vivant.

La manière de procéder se fonde sur un débat de nature démocratique autour des différentes propositions formulées, puis la décision revient *in fine* au créateur du langage, Guido VAN ROSSUM qui, jusqu'en 2018 s'est autoproclamé « dictateur bienveillant à vie » — *BDFL Benevolent Dictator For Life* — (comme Linus TORVALDS pour le noyau LINUX, Larry WALL pour PERL ou Mark SHUTTLEWORTH pour UBUNTU). C'est certainement pour cette raison que la cohérence du langage a pu être préservée pendant une durée aussi longue.

Depuis l'été 2018, cette organisation est modifiée avec le retrait partiel de Guido⁵ VAN ROSSUM. La nouvelle gouvernance s'articule⁶ autour d'un « *Steering Council* » à traduire en « Conseil de pilotage ».

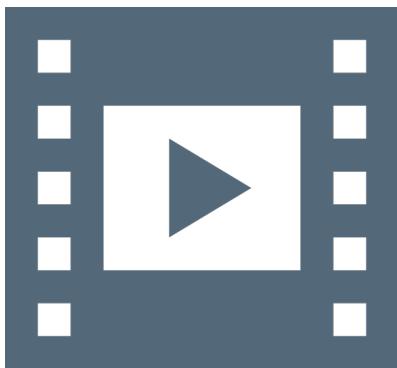
POUR ALLER PLUS LOIN...

ZEN DE PYTHON

- ▶ Le « Zen de Python » résume la philosophie du langage, ce document est disponible en important le module **this**.

5. Né en 1956, VAN ROSSUM passe progressivement le flambeau et anticipe une retraite bien méritée...

6. Le *Steering Council* est défini dans la [PEP-8016 — Python Enhancement Proposal](#) —, dont les membres élus sont : Brett Cannon, Nick Coghlan, Guido VAN ROSSUM, Barry WARSAW et Carol WILLING.



VIDÉO 8.3 — Environnements de développement : IDLE et interpréteur PYTHON.

DOCUMENTATION

- ▶ Article [WIKIPEDIA sur PYTHON](#).
- ▶ Page en [français](#) du wiki [PYTHON](#).
- ▶ Documentation originale de [PYTHON v.3](#) en anglais. C'est un très bon point d'entrée lorsqu'on cherche un sujet particulier, mais (beaucoup) trop abondant pour être lu d'un seul trait. Pour chercher de la documentation sur un module particulier, le plus simple est encore d'utiliser un moteur de recherche qui redirigera, dans la grande majorité des cas, vers la page qui va bien de la documentation de [PYTHON](#). À titre d'exercice, obtenir la documentation du module [pathlib](#) en saisissant les mots-clés « [python module pathlib](#) ».
- ▶ Traduction [française](#) de la documentation officielle.

SURVOL HISTORIQUE

- ▶ Foire aux questions (FAQ) officielle de [PYTHON](#) (en anglais) sur [les choix de conception et l'historique du langage](#).
- ▶ Article [WIKIPÉDIA](#) (en anglais) sur l'historique du langage.
- ▶ Article [WIKIPÉDIA](#) (en anglais) sur [la syntaxe et la sémantique de Python](#).

LICENCE ET DROITS

- ▶ Licence d'exploitation.
- ▶ Présentation de la [PYTHON Software Foundation](#).

DÉVELOPPEMENT ET GOUVERNANCE

- ▶ Choix et discussions des évolutions au moyen des PEP, soit [PYTHON Enhancement Proposals](#).
- ▶ Description du cycle de vie des PEP : <https://legacy.python.org/dev/peps/pep-0001/>.
- ▶ Préconisations du style de présentation (style guide) : <https://legacy.python.org/dev/peps/pep-0008/>.
- ▶ Index de l'ensemble des PEP : <https://legacy.python.org/dev/peps/>.

1.2 Outils de développement

INSTALLATION DE PYTHON

L'installation de [PYTHON](#) et de ses outils principaux de développement est détaillée en ?? « ?? ». Les procédures sont exposées en fonction du système d'exploitation.

Dans ce document, la position prise est de n'employer que des logiciels libres ou *Open Source*. Par conséquent, l'ensemble des exemples correspond à un environnement de travail fondé sur une base [DEBIAN/UBUNTU](#), en l'occurrence la distribution [UBUNTU MATE 20.04 LTS](#). Sa relative légèreté permet d'envisager l'installer sur des matériels anciens ou des [RASPBERRY PI](#) (non testé à ce jour).

Une fois l'installation du langage réalisée, il existe essentiellement deux familles d'outils pour communiquer avec [PYTHON](#), soit directement au moyen d'un interpréteur en ligne de commande ou via un environnement intégré de développement — IDE en anglais pour *Integrated Development Environment*.

1.2.1 Interpréteur PYTHON

L'interpréteur [PYTHON](#) est assimilable à un moteur qui fait tourner les programmes. En contexte standard, il n'y a besoin de rien d'autre.

Pour accéder à l'interpréteur Python, il faut d'abord lancer un terminal — « *shell* » en anglais pour « coque/coquille ». Le terminal est dévolu aux opérations en ligne de commande, à savoir non graphiques, sans IHM (cf. chapitre 1, « [Numérique et sciences du réel](#) »).

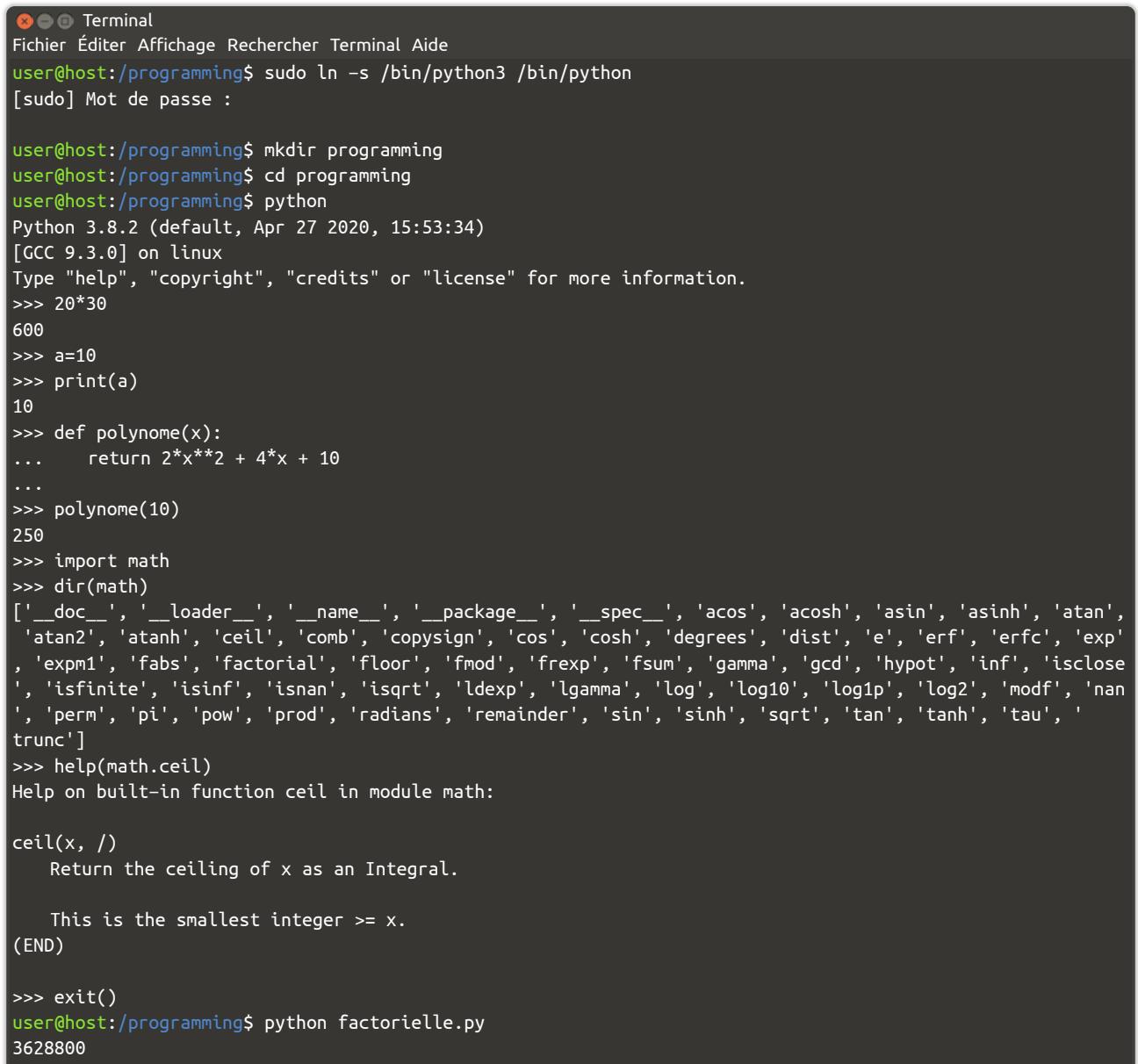
Faire appel à un terminal est assez naturel aux utilisateurs de systèmes de type [UNIX](#) — par exemple, saisir [`Ctrl + Alt + T`](#) sous [LINUX DEBIAN/UBUNTU](#). Sous [WINDOWS](#), l'accès au terminal se réalise au moyen de la commande [`cmd`](#).

Il ne faut pas confondre le terminal, dans lequel des commandes système ou des programmes écrits en langage de script « *shell* » — typiquement en *Bash* pour *Bourne-Again Shell*⁷ — peuvent être lancées, avec l'interpréteur [PYTHON](#), même si ce dernier va occuper la même fenêtre de terminal.

⁷. *Bash* tient son nom du créateur initial de l'interpréteur de commande [UNIX](#) « *sh* » pour *shell*, Stephen BOURNE. Sa programmation est à l'origine l'œuvre de Brian Fox dans le cadre du projet [GNU](#) (1989). Cette implémentation est désormais celle par défaut des systèmes [LINUX](#) et [MACOSX](#).

Ainsi, à l'invite de commande du terminal, il faut saisir la commande « **python** » pour lancer l'interpréteur PYTHON. Au préalable, on peut se placer dans un répertoire de travail⁸ dévolu aux programmes PYTHON, mettons : **home>user>programming**.

Sur certains systèmes⁹ où coexistent les deux versions 2.7 et 3.x, l'appel à PYTHON en version 3 se fait par la commande « **python3** ». Ce n'est pas commode, car il faut continuellement se souvenir de quelle version on a besoin, au cas où on aurait velléité d'intervenir sur d'anciens programmes. Cette situation n'a pas lieu d'être ici. Une première chose à faire est donc alors de créer un *alias* de commande dans un des fichiers cachés « **.bashrc** » ou « **.bash_aliases** » de la racine personnelle (cf. procédure : <https://doc.ubuntu-fr.org/alias>).



```

Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ sudo ln -s /bin/python3 /bin/python
[sudo] Mot de passe :

user@host:/programming$ mkdir programming
user@host:/programming$ cd programming
user@host:/programming$ python
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 20*30
600
>>> a=10
>>> print(a)
10
>>> def polynome(x):
...     return 2*x**2 + 4*x + 10
...
>>> polynome(10)
250
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
>>> help(math.ceil)
Help on built-in function ceil in module math:

ceil(x, /)
    Return the ceiling of x as an Integral.

    This is the smallest integer >= x.
(END)

>>> exit()
user@host:/programming$ python factorielle.py
3628800

```

Néanmoins, si un *alias* s'avère pratique pour l'utilisateur, il ne suffit pas au niveau du système. En effet, si un programme souhaite invoquer PYTHON au moyen de la commande « **python** », il ne la trouvera pas. La solution est alors de créer un lien symbolique — en quelque sorte un genre d'*alias* système — entre les commandes « **python** » et « **python3** » dans le répertoire **bin** où se trouve l'ensemble des exécutables du

⁸. Remplacer le mot « **user** » par le nom du compte utilisateur courant.

⁹. C'est cas d'UBUNTU 20.4 LTS dont la version 3.8 est installée par défaut, mais où l'appel à PYTHON se réalise toujours par la commande « **python3** ».

10. Dans la simulation de terminal :
- ▶ le compte utilisateur est « `user` »;
 - ▶ la machine hôte est « `host` ».

système. Si d'aventure cela ne suffisait pas à l'avenir, penser à réitérer la démarche dans le répertoire `usr > bin`. Les étapes à suivre sont synthétisées dans la simulation¹⁰ de terminal ci-dessous.

La fenêtre de terminal fait désormais place à l'interpréteur PYTHON. Au passage, on peut remarquer que la version qui est employée est bien en version 3.x et que l'interpréteur PYTHON est bien actif par la présence de son invite de commande spécifique formée de trois chevrons pointant vers la droite. On sort de l'interpréteur pour retourner au shell au moyen des commandes « `quit()` », « `exit()` » ou en saisissant les touches `Ctrl + D`.

À partir de là, l'interpréteur PYTHON peut s'utiliser, soit comme une grosse calculette, soit pour définir des variables, leur assigner des valeurs et les afficher ou, bien entendu, pour établir des fonctions à la volée en respectant les conventions d'indentation de PYTHON.

Il est également possible d'importer des bibliothèques externes (cf. l'outil de gestion « `pip` ») ou, dans le jargon PYTHON des modules avec la commande « `import <nom-module>` », d'en lister les fonctions disponibles avec « `dir(<nom-module>)` » et d'obtenir de l'aide sur une fonction par « `help(<nom-fonction>)` » (sortie avec la touche « `Q` »).

Employer directement l'interpréteur PYTHON est rapidement fastidieux, aussi, on préfère avantageusement construire les développements dans des fichiers séparés — d'où la création préalable d'un répertoire idoine —, puis les charger directement depuis le terminal.

À cet effet et pour illustration par l'exemple, au sein dudit répertoire dédié `home > user > programming`, on définit un programme dans le fichier¹¹ « `factorielle.py` » comportant le code ci-dessous. Il est ensuite extrêmement simple d'y faire appel avec la commande (voir la simulation de terminal) : « `python factorielle.py` ».



```
1 def factorielle(n):
2     """Calcul de factorielle"""
3     if n<=1:
4         return 1
5     else:
6         return n*factorielle(n-1)
7
8 # Commentaire : factorielle(10) en sortie
9 print(factorielle(10))
```

11. Sous UBUNTU, l'éditeur installé par défaut est `gedit`. Pour le bureau MATE, c'est un fork de `gedit` nommé `pluma`. Bref, peu importe, faire le choix d'un éditeur de préférence : `geany`, `emacs`, `vim`, etc.

AGRÉMENT POUR IPYTHON

Après l'installation de IPYTHON (voir détails en ?? « ?? »), la même création de lien symbolique est à conduire pour s'assurer de la version correcte du programme. Ainsi, depuis un terminal, il faut saisir : `sudo ln -s /bin/ipython3 /bin/ipython`

Dans ce cours, une version améliorée de l'interpréteur PYTHON standard est utilisée, à savoir IPYTHON (à installer en tant que tel).

À l'usage, cet interpréteur bénéficie de fonctionnalités voisines mais plus agréables à manipuler. Il en est ainsi de l'aide en ligne générale, accessible par un point d'interrogation.

De plus, la complétion (touche `Tab`) est bien plus efficace et pratique, de même que l'indentation qui, cette fois, est automatique et correctement agencée. Par ailleurs, la coloration syntaxique est appliquée par défaut, ce qui facilite rétrospectivement la lecture des instructions PYTHON saisies à la volée.

À titre d'illustration du propos, on peut d'abord importer comme précédemment le module mathématique : « `import math` ». Avec la complétion, il suffit de saisir le nom du module « `math.` » et au moyen de la touche `Tab` obtenir la liste des fonctions disponibles. Mais surtout, chose nouvelle, en saisissant de nouveau la touche `Tab`, on peut naviguer sur les fonctions proposées pour s'attacher à l'une d'entre-elles puis, en rajoutant un point d'interrogation, avoir l'aide en ligne qui lui est associée.

En ce qui concerne la coloration syntaxique de IPYTHON, une nouvelle fonction « `fibonacci` » est définie comme exemple concret d'utilisation (voir simulation de console ci-après). On y constate la notification des mots-clés du langage.

```

Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:~/programming$ ipython
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
Type `copyright`, `credits` or `license` for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type `?` for help.

In [1]: def fibonacci(n):
...:     return 1 if n <= 1 else fibonacci(n-1) + fibonacci(n-2)
...:

In [2]: fibonacci(10)
Out[2]: 89

In [3]: import math

In [4]: math.
    acos() atan() comb() degrees() erfc() factorial() fsum() inf isnan() log()
    acosh() atan2() copysign() dist() exp() floor() gamma() isclose() isqrt() log10()
    asin() atanh() cos() e expm1() fmod() gcd() isfinite() ldexp() log1p()
    asinh() ceil() cosh() erf() fabs() frexp() hypot() isinf() lgamma() log2()

In [4]: math.ceil?
Signature: math.ceil(x, /)
Docstring:
Return the ceiling of x as an Integral.

This is the smallest integer >= x.
Type:      builtin_function_or_method

In [5]: math.ceil(9.5)
Out[5]: 10

```

1.2.2 Integrated Development Environment — IDLE

Comme bon nombre d'autres langages informatiques, PYTHON dispose d'outils de développement dédiés — *Integrated Development Environment* ou Environnement de développement intégré (EDI) en français — facilitant la tâche des programmeurs. Toutes les fonctionnalités nécessaires au développement d'une application sont regroupées au sein d'une même interface de travail.

Pour ce qui concerne PYTHON et au-delà des EDI généralistes comme ECLIPSE/PYDEV ou les éditeurs multi-facettes comme EMACS, on peut citer les plus connus : PyCHARM (licence propriétaire ou communautaire APACHE), SPYDER¹² (sous licence libre MIT, donc sans copyleft) et, bien entendu, faisant partie de la distribution¹³ standard, IDLE.

Selon son auteur, Guido VAN ROSSUM, l'acronyme officiel de IDLE signifie *Integrated Development Environment*. Selon d'autres sources, on trouve également l'appellation de : *Integrated Development and Learning Environment*.

IDLE se présente sous la forme d'un *shell* interactif PYTHON (là aussi avec une invite de commande de trois chevrons) qui, nativement, dispose de la coloration syntaxique du code, du surlignage des messages d'erreur, de l'indentation et de l'autocomplétion automatiques des instructions, sans compter d'autres attributs de recherche et de débogage.

¹². SPYDER est intéressant à plus d'un titre. Conçu par et pour la communauté PYTHON scientifique, ses fonctionnalités le positionne comme une alternative libre à MATLAB : interactivité, analyse et visualisation de données (MATPLOTLIB), etc. Il peut être étendu par un système de greffons (plugins) et, construit sur la bibliothèque graphique PyQt, incorporé dans d'autres programmes comme une de ses extensions.

¹³. Pour LINUX, IDLE est disponible comme paquet indépendant pour avoir le choix d'installation d'un EDI de prédilection.

¹⁴ Pour LINUX, installer au préalable le paquet « `idle` » ou « `idle3` » avec le gestionnaire de logiciels ou au moyen de la ligne de commande.

Comme pour toute application, pour l'exploiter, il faut la lancer; quel que soit le système d'exploitation, IDLE est accessible quelque part à partir des menus¹⁴ du bureau. L'interface obtenue est identique à celle de l'interpréteur PYTHON rencontré précédemment (voir figure 8.1).

On peut alors reprendre toutes les opérations effectuées avec l'interpréteur PYTHON (voir § 1.2.1) : de l'utilisation comme calculette à la définition d'instructions et de fonctions à la volée, en passant par la levée d'exception d'erreur et l'aide en ligne. La différence essentielle est que pour saisir une fonction ou un code plus élaboré dans un fichier séparé, il n'y a pas nécessité de sortir de l'environnement mais il suffit simplement d'appeler l'éditeur de texte associé à IDLE : `File > New File`. Une nouvelle fenêtre s'ouvre dans laquelle le code est à entrer.

The screenshot shows the Python IDLE interface. On the left, there is a terminal-like window titled "Python 3.8.2 (default, Apr 27 2020, 15:53:34) [GCC 9.3.0] on linux". It contains the following Python code:

```

Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 20*30
600
>>> a = 10
>>> print(a)
10
>>> def polynome(x):
    return 2*x**2 + 4*x + 10

>>> polynome(10)
250
>>> polynom(10)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    polynom(10)
NameError: name 'polynom' is not defined
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copy',
'sign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm',
'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldex',
'p', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi',
'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'tau', 'trunc']
>>> help(math.ceil)
Help on built-in function ceil in module math:

ceil(x, /)
    Return the ceiling of x as an Integral.

    This is the smallest integer >= x.

>>> |

```

On the right, there is an editor window with the following Python code:

```

# Un commentaire
def idlefactorielle(n):
    """Calcul de factorielle (demo IDLE)"""
    if n <= 1:
        return 1
    else:
        return n * idlefactorielle(n-1)

```

FIGURE 8.1 – IDLE : console et éditeur de fichier PYTHON.

Une fois sauvegardé le fichier nouvellement créé en accédant à `File > Save`, ici enregistré sous « `idlefactorielle.py` », là encore, il n'y a pas besoin de sortir de l'environnement pour exécuter le fichier; il faut taper la touche de raccourci `F5` ou bien suivre le menu `Run > Run Module`.

La console de l'interpréteur se réinitialise — toutes les instructions et variables précédemment introduites sont effacées — et charge le module. Ce dernier est dès lors fonctionnel et peut s'utiliser par exemple en évaluant `idlefactorielle(10)`, fonction du module chargé.

Cette boucle interactive est le principe de travail avec IDLE et tout autre environnement de développement intégré : on introduit le nouveau code dans un fichier, on valide en sauvegardant puis on passe la main à l'interpréteur pour vérifier le résultat; et ainsi de suite...

D'un EDI à l'autre et au-delà de leurs fonctionnalités propres — visualisation de données, développement Web avec DJANGO, etc. —, seule

l'interface de présentation change. Juste à titre de comparaison, la figure 8.2 montre l'interface utilisateur de SPYDER, qui intègre dans le même espace de travail un éditeur de fichier, une console IPYTHON et un volet d'exploration de fichiers ou d'affichage de l'aide en ligne.

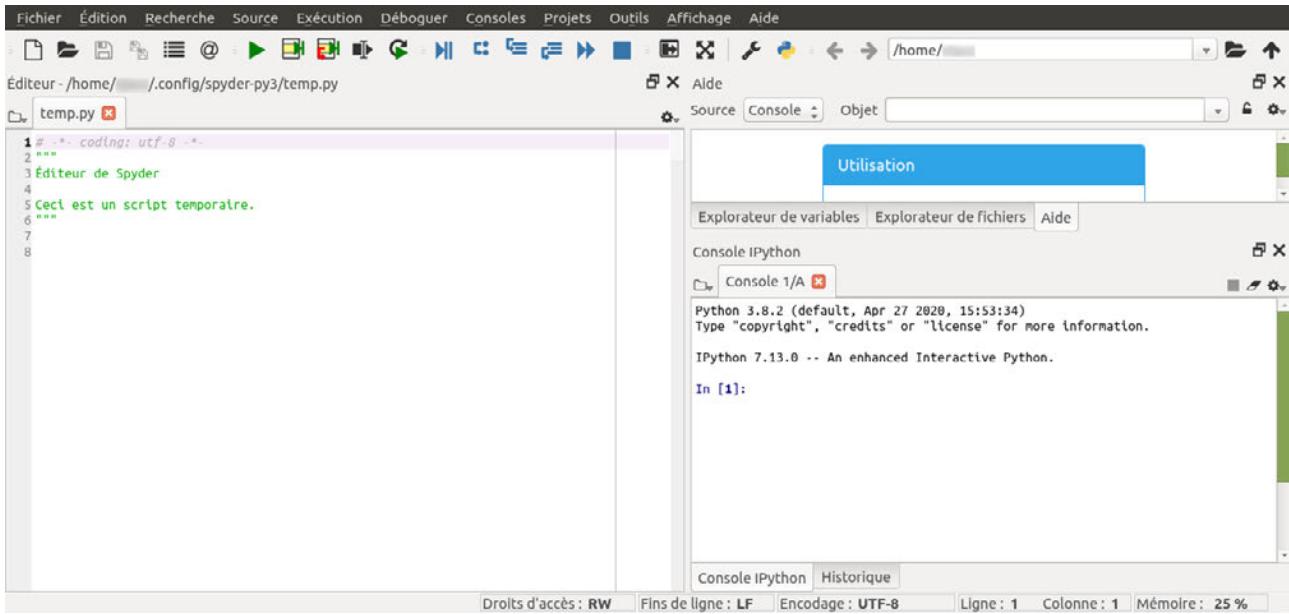


FIGURE 8.2 – Interface utilisateur de SPYDER.

1.3 Documentation et code PYTHON

Au fur et à mesure des années et, tout particulièrement dans la décennie 2010–2020, les outils de documentation et d'apprentissage de l'informatique se sont considérablement améliorés. L'essor de l'Internet y a largement contribué, à l'exemple de certains langages de balisage léger permettant la mise en exergue des codes.

Pour les documentations écrites, les outils de composition TeX (1983) puis LATEX (1994) restent maîtres en la manœuvre, à l'image de ce document. Néanmoins, un minimum d'interactivité est apportée par l'extension PYTHONTEX (depuis 2012), succinctement présentée en § 1.3.2.

En revanche, les *notebooks* JUPYTER (2015) apportent une interactivité complète entre présentation de données textuelles — cours et explications — et de codes modifiables à volonté — manipulations et tests. Cela offre un outil pédagogique efficient pour l'apprentissage d'un langage de programmation.

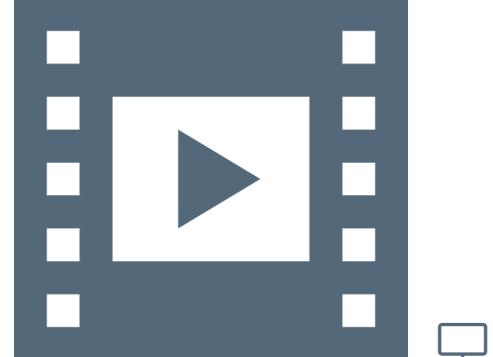
1.3.1 « Notebooks » JUPYTER

Que recouvre donc la terminologie de *notebooks* JUPYTER ?

En premier lieu et par contraste avec les outils auparavant présentés, le système de *notebooks* n'a pas besoin d'être installé en local¹⁵ sur la machine de l'utilisateur; il est hébergé sur une plateforme distante pour permettre le travail en ligne.

Un *notebook* est ainsi un mélange de texte et de code. On navigue au sein de ses différentes cellules au moyen de la suite de touches [Maj] + [Entrée]. Tant que l'on est sur du texte cela importe peu. En revanche, sur une cellule de code, la saisie des touches [Maj] + [Entrée] active le code contenu et le communique à l'interpréteur qui renvoie le résultat.

De cette manière, on peut lire linéairement le document du début à la fin en évaluant le code au fur et à mesure où, à chaque fois qu'une



VIDÉO 8.4 – Notebooks JUPYTER.



¹⁵. Sauf pour travailler sans connexion Internet ou développer soi-même une documentation sur ce principe.

ATTENTION!

Pour le bon fonctionnement des notebooks, il faut avoir autorisé le navigateur Web à accepter les cookies en provenance du site <https://nbhosting.inria.fr>, qui héberge l'infrastructure du Mooc.



FIGURE 8.3 – Barres de menu et de raccourcis d'un notebook JUPYTER.

cellule de code est rencontrée, son label entre crochets est incrémenté d'une unité. Cela représente le degré zéro de l'utilisation d'un *notebook*. On peut également mentionner que l'ensemble du *notebook* est évalué en une seule opération avec [Cell] → Run All.

Comme pour tous les programmes à interface graphique, les notebooks disposent de barres de menu et de raccourcis (cf. figure 8.3). Il est néanmoins conseillé, par souci d'efficacité, de retenir les quelques raccourcis clavier qui rendre la consultation plus fluide.

Lorsqu'une cellule de code est évaluée, JUPYTER ajoute sous la cellule In une cellule Out qui donne le résultat du fragment PYTHON considéré, soit 600 dans le premier exemple de la figure 8.4.

En seconde instance, on peut revenir sur une cellule de code pour en modifier les paramètres et de nouveau en tapant [Maj] + [Entrée], réévaluer le résultat retourné par la cellule.

Ces premières étapes sont reprises par la figure 8.4 où sont évaluées deux celulles de codes telles que présentées initialement puis après modification par l'utilisateur.

```
In[1] 20 * 30
Out[1] 600
In[1] 20 * 40
Out[1] 800
In[2] # math.sqrt (pour square root) calcule la racine carrée
      import math
      math.sqrt(2)
Out[2] 1.4142135623730951
In[2] # math.sqrt (pour square root) calcule la racine carrée
      import math
      math.sqrt(25)
Out[2] 5
```

FIGURE 8.4 – Évaluation et modification de cellules de code notebook JUPYTER.

L'intérêt du numéro entre crochets des cellules contenant du code est de suivre l'ordre dans lequel celles-ci sont évaluées : on peut tout à fait sauter certaines cellules dans le fil de lecture pour y revenir ensuite. Toutefois, cela devient rapidement confus et s'avère fortement déconseillé car pouvant mener à des résultats inattendus selon la progression pédagogique voulue du *notebook*.

En effet, les cellules de code peuvent être des fragments d'un même programme PYTHON ; les exécuter dans le désordre produit un ensemble différent, voire incohérent. La figure 8.5 expose une telle situation.

```
In[1] message = "Faites attention à l'ordre dans lequel vous évaluez les
      ↵ notebooks"
In[3] print(message)
-----
NameError                                 Traceback (most recent call last)
<ipython-input-3-83cfdf559502> in <module>
      1 print(message)

NameError: name 'message' is not defined
In[2] # ceci a pour effet d'effacer la variable 'message'
      del message
```

FIGURE 8.5 – Évaluation désordonnée de cellules de code notebook JUPYTER.

On peut avoir besoin de réinitialiser le *notebook* après avoir réalisé trop de modifications ou perdu le fil de ce qui a été évalué. Pour ce faire, il faut aller dans le menu du *notebook* pour faire un [Kernel] > [Restart] ou mieux, [Kernel] > [Restart & Clear Output], autrement dit, non seulement redémarrer le noyau, mais aussi remettre à zéro tous les affichages.

```
In[1] # Il faut exécuter cette cellule pour charger l'exercice
from corrections.w1s3_polyname import exo_polyname

In[2] # Une cellule de ce genre est destinée à montrer le résultat attendu
exo_polyname.example()

Out[2] Appel      Attendu
polyname(-2)    4
polyname(0)     -4

In[] # Le code proposé est à entrer ici
def polyname(x):
    "votre code"

In[3] # Le code proposé est à entrer ici (oubli de l'instruction 'return')
def polyname(x):
    2*x**2-4

In[4] # Une fois exécuté la cellule précédente avec votre code,
# il faut évaluer celle-ci pour vérifier si cela fonctionne
exo_polyname.correction(polyname)

Out[4] Appel      Attendu      Obtenu
polyname(-2)    4           None       NO
polyname(0)     -4          None       NO
polyname(2)     4           None       NO

In[3] # Le code proposé est à entrer ici
def polyname(x):
    return 2*x**2-4

In[4] # Une fois exécuté la cellule précédente avec votre code,
# il faut évaluer celle-ci pour vérifier si cela fonctionne
exo_polyname.correction(polyname)

Out[4] Appel      Attendu      Obtenu
polyname(-2)    4           4          OK
polyname(0)     -4          -4         OK
polyname(2)     4           4          OK
```

FIGURE 8.6 — Exercice auto-évalué.

Un point important à connaître est que l'utilisateur travaille sur une copie de la proposition initiale. Toutes les modifications qui sont faites par l'utilisateur lui appartiennent et peuvent être sauvegardées. Elles le sont automatiquement, mais si on souhaite disposer d'une progression plus fine, suivre le menu [File] > [Save] ou [File] > [Save as...].

À tout moment, la version initiale du *notebook* peut être rechargée en suivant [File] > [Reset to Original]; faire attention au code saisi auparavant, car il va bien entendu être écrasé.

Une autre fonctionnalité intéressante est de pouvoir télécharger le contenu du *notebook* dans de multiples formats et notamment en PYTHON : [File] > [Download as] > [Python (.py)]. Pour ce dernier format, les cellules de texte sont préservées sous forme de commentaires PYTHON, seules les cellules de code sont actives.

En contexte de partage d'expérience, comme par exemple en formation à distance, il est encore possible de publier une version HTML statique du fruit du travail d'un étudiant donné : [File] > [Share Static Version].

Une URL est obtenue afin d'être publiée dans un forum d'aide ou envoyée par courriel. Ainsi, les autres participants accèdent en lecture seule au code. Cette manipulation peut se réaliser plusieurs fois : c'est toujours la dernière version qui est partagée — l'URL reste la même pour un étudiant et un *notebook* donnés.

Enfin, des cellules supplémentaires peuvent être rajoutées : `Insert > Insert Cell Above` ou `Insert > Insert Cell Below`. Arrivé à la fin d'un document, chaque fois que la dernière cellule est évaluée, cela crée une nouvelle cellule. De cette manière, on dispose d'un brouillon pour effectuer des tests personnels concernant les sujets abordés par le *notebook* en cours de consultation.

La table 8.1 reprend les différentes actions possibles sur un *notebook* JUPYTER avec leur commandes associées.

EXERCICES AUTO-ÉVALUÉS — Le Mooc propose un certain nombre d'exercices à réaliser en autonomie, dont la correction est guidée par les réponses apportées, d'où le terme d'*auto-évalué*.

Au sein d'un *notebook*, un énoncé est formulé. À titre d'exemple, le propos est d'écrire une fonction qui implémente le polynôme $2x^2 - 4$ (cf. figure 8.6). La manière de procéder est toujours la même, à savoir évaluer les cellules à l'aide des touches `Maj + Entrée`.

L'évaluation de la première cellule d'exercice est fondamentale, car c'est elle qui charge son contenu. Les résultats attendus sont ensuite exposés dans la cellule suivante de manière à orienter les sorties du code à produire (voir figure 8.6).

La cellule encore suivante est ouverte au sens où elle est le réceptacle du code à proposer pour résoudre le problème énoncé. Une fois remplie, on passe à la cellule subséquente qui fournit la correction de la réponse donnée.

En cas d'échec, une sortie explicite (fond rouge) est renvoyée entre les attendus et les résultats actuels. En cas de succès, il en est de même, mais la comparaison (fond vert) entre attendus et résultats est bien entendu positive. On peut évaluer ces deux dernières cellules autant de fois qu'il est nécessaire pour atteindre une correction correcte (cf. figure 8.6 où les deux situations sont illustrées : réponse fausse et résultat correct).

1.3.2 Extension L^AT_EX PYTHONT_EX

Les quelques paragraphes à suivre n'ont aucun rapport direct avec le Mooc, ils offrent juste quelques éléments d'information supplémentaire aux personnes qui, maîtrisant correctement L^AT_EX, souhaiteraient reprendre ce document ou en rédiger un nouveau.

L'installation et l'emploi des outils de composition L^AT_EX utilisés pour ce document sont exposées en annexe. Il s'agit uniquement de mentionner une des extensions utilisées, PYTHONT_EX, qui apporte un minimum d'interactivité entre un document L^AT_EX et des syntaxes ou des programmes PYTHON qui sont à présenter et à expliquer.

L^AT_EX est un langage de balisage qui permet la mise en forme de documents de qualité, mais qui, contrairement à par exemple HTML, nécessite une phase de compilation pour passer d'un fichier texte source à un document PDF (pour simplifier). La procédure habituelle est donc de lancer un programme — en pratique `pdflatex` ou `lualatex` — pour gérer cette conversion : `pdflatex fichier.tex` pour avoir `fichier.pdf`.

Dans la conversion vers le format PDF, un certain nombre de fichiers auxiliaires sont également générés pour correctement afficher les références croisées (numéros de figure, table, équation, etc.). Plusieurs

TABLE 8.1 — Fonctionnalités d'un notebook JUPYTER.

Commandes et raccourcis	
ACTION	CLAVIER
Déplacement/ Évaluation	<code>Maj + Entrée</code>
ACTION	MENU
Évaluation globale	<code>Cell > Run All</code>
Réinitialisation	<code>Kernel > Restart</code>
Réinitialisation et remise à zéro	<code>Kernel > Restart Clear Out.</code>
Enregistrement	<code>File > Save</code>
Version originale	<code>File > Reset to Original</code>
Téléchargement	<code>File > Download as</code>
Lien partagé (URL/HTML)	<code>File > Share Stat. Vers.</code>
Insertion cellule	<code>Insert</code>
ACTION	RACCOURCI
Évaluation	<code>▶</code>
Insertion après	<code>+</code>
Interruption	<code>■</code>
Redémarrage avec confirm.	<code>⟳</code>
Redémarrage sans confirm.	<code>▶</code>

compilations sont alors nécessaires pour que soit stabilisé le fichier final en PDF. C'est le prix à payer de la qualité typographique et de la précision en regard d'autres traitements de texte traditionnels.

Dans ce processus, l'extension PYTHONTEX s'insère en tant qu'étape intermédiaire entre le fichier source et le résultat PDF, au même titre que la nécessité de compilations multiples (en pratique deux ou trois). C'est-à-dire que la procédure devient désormais : `pdflatex fichier ▶ pythontex fichier ▶ pdflatex fichier` où `pythontex` est un script PYTHON qui s'occupe d'interpréter du code PYTHON, en l'occurrence.

Certes, fort bien, mais quelle¹⁶ est la plus-value ? L'intérêt est que, moyennant une syntaxe adéquate, il est possible depuis un document LATEX d'envoyer du code PYTHON pour interprétation et d'en récupérer le résultat pour sa mise en forme au sein d'un document. Tout comme pour les *notebooks*, la bibliothèque d'affichage des codes est `Pygments`.

À l'évidence, cette solution est moins interactive qu'un *notebook JUPYTER*, mais s'avère très performante dans le cadre de la rédaction de manuels ou d'ouvrages de programmation informatique (PYTHON n'est pas le seul langage supporté, cf. documentation CTAN PYTHONTEX).

¹⁶ Au passage, cela montre la polyvalence et la puissance de PYTHON dans de multiples champs d'application.

1.4 Synthèse et compléments

Après l'abord des généralités à propos des contextes d'utilisation du langage PYTHON, une petite synthèse est menée et quelques exercices simples sont proposés pour avancer un peu plus.

1.4.1 Mode d'exécution PYTHON

Les différents modes d'exécution d'un programme PYTHON sont résumés en [table 8.2](#). Il s'agit maintenant de les étudier en détail. Pour cela, le comportement d'un tout petit programme va être regardé lorsqu'on l'exécute sous ces différents environnements.

Essentiellement, lorsqu'on utilise l'interpréteur en mode interactif — ou sous IDLE —, à chaque fois que l'on tape une ligne, le résultat est calculé (on dit aussi évalué) puis affiché.

En revanche, lorsqu'on écrit tout un programme, on ne peut plus imprimer le résultat de toutes les lignes, cela produirait un flot de données beaucoup trop important. Par conséquent, si on ne déclenche pas un affichage explicite avec, par exemple, la fonction `print`, rien ne se se produit en sortie.

En ce qui concerne le *notebook*, le comportement est un peu hybride entre les deux, en ce sens que seul le *dernier résultat* de la cellule est affiché.

Le programme qui sert ici d'illustration est simplissime.

```
1 10 * 10
2 20 * 20
3 30 * 30
```

Lorsque ces instructions sont soumises à l'interpréteur interactif (ouvrir un terminal, puis appeler `python`), on obtient :

```
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ python
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
```

TABLE 8.2 – Modes d'exécution PYTHON.

Exécution d'un programme PYTHON	
QUOI?	COMMENT?
Fichier entier	<code>python <fichier>.py</code>
Ligne à ligne	IDLE, <code>ipython</code> , <code>python</code> mode interactif
Par fragment	<code>notebook</code>

```
>>> 10 * 10
100
>>> 20 * 20
400
>>> 30 * 30
900
>>> exit()
user@host:/programming$
```

À noter que pour sortir de l'interpréteur, il faut saisir `exit()` — ou sous LINUX et MACOSX `Ctrl + Entrée`.

Que donne à présent cette même séquence de calculs dans un programme complet ? Pour cela, il faut tout d'abord fabriquer un fichier ayant pour suffixe `.py`, au moyen par exemple d'un éditeur de fichier, puis exécuter le programme ainsi obtenu. Le résultat doit ressembler à ceci (la fonction `cat` est une instruction UNIX qui permet de concaténer et/ou définir un fichier en ligne de commande) :

```
Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ cat > foo.py << EOF
> 10 * 10
> 20 * 20
> 30 * 30
> EOF
user@host:/programming$ cat foo.py
10 * 10
20 * 20
30 * 30
user@host:/programming$ python foo.py
user@host:/programming$
```

Tout au moins en apparence, ce programme semble *ne rien faire* !

En réalité, les trois valeurs **100**, **400** et **900** sont bien calculées, mais comme aucune instruction `print` n'est présente, rien ne s'affiche et le programme se termine sans signe apparent d'avoir fonctionné.

Ce comportement peut paraître un peu déroutant au début, mais comme déjà mentionné c'est tout à fait délibéré. Avec un programme fonctionnel faisant facilement plusieurs milliers de lignes, voire beaucoup plus, il ne serait pas du tout réaliste que chaque ligne produise une impression, comme c'est le cas en mode interactif.

En exécutant cette fois la même série d'instructions dans un *notebook* (par sélection de la cellule, puis saisie de `Maj + Entrée`), on a :

In[1] `10 * 10
20 * 20
30 * 30`

Out[1] `900`

Une seule valeur est obtenue en sortie — rubrique `Out[]` —, en l'occurrence **900**, qui correspond bien au *résultat de la dernière ligne*.

UTILISATION DE L'INSTRUCTION D'AFFICHAGE — Ainsi, pour afficher un résultat intermédiaire, on emploie l'instruction `print`. Son étude est abordée en détail plus loin dans le Mooc, mais en guise d'introduction, disons seulement que c'est une fonction comme les autres en PYTHON V.3.X.

In[1] `a = 10
b = 20
print (a, b)`

Out[1] `10 20`

On peut naturellement mélanger des objets de plusieurs types et donc combiner des chaînes de caractères et des nombres pour obtenir un résultat un peu plus lisible. En effet, lorsque le programme devient volumineux, il est important de savoir à quoi correspond une ligne dans le flot de tous les affichages. Aussi on préférera quelque chose comme :

In[2] `print("a =", a, "et b =", b)`

Out[2] `a = 10 et b = 20`

In[3] `# ou encore, de manière équivalente mais avec un f-string
print(f"a = {a} et b = {b}")`

Out[3] `a = 10 et b = 20`

Une pratique courante consiste d'ailleurs à utiliser les commentaires pour laisser dans le code les instructions `print` qui correspondent à du débogage (c'est-à-dire qui ont pu être utiles lors de la mise au point et qu'on veut pouvoir réactiver rapidement).

Remarquons enfin que l'affectation à une variable ne retourne aucun résultat. Autrement dit, en pratique, que si on écrit :

In[4] `a = 100`

même une fois l'expression évaluée par l'interpréteur, aucune ligne de sortie Out[] n'est ajoutée.

C'est pourquoi, il arrive parfois d'écrire, notamment lorsque l'expression est complexe et pour rendre explicite la valeur qui vient d'être affectée :

In[5] `a = 100; print(a)`

Out[5] `100`

Bien noter que cette technique est uniquement pédagogique et n'a absolument aucun autre intérêt dans la pratique; il n'est *pas recommandé* de l'utiliser en dehors de ce contexte.

LIGNE D'INVITE SHEBANG — Pour finir sur les modes d'exécution de PYTHON, une fonctionnalité pratique est à connaître, mais uniquement valable pour les systèmes Unix (LINUX et MACOSX).

Le lancement d'un programme Python s'effectue depuis le terminal par la commande : `python mon_module.py`. Lorsqu'il s'agit d'un programme que l'on utilise fréquemment, on n'est pas forcément dans le répertoire où se trouve le programme PYTHON. Aussi, on peut alors utiliser un chemin « absolu », c'est-à-dire à partir de la racine des noms de fichiers, comme : `python /le/chemin/de/mon_module.py`. Cependant, c'est assez malcommode et cela devient vite pénible à la longue.

Sur LINUX et MACOSX, il existe une astuce utile pour simplifier cela. En prenant par exemple le programme `fibonacci.py` de l'exercice d'application à suivre (cf. § 1.4.2), on l'édite avec un éditeur de texte pour lui rajouter une toute première ligne qui commence par un caractère de commentaire « # », seul caractère qui, du à sa position dans le fichier, n'a pas la signification de début de commentaire mais, par la présence d'un point d'exclamation immédiatement après le caractère dièse rend active la suite de la ligne comme chemin d'accès à l'interpréteur PYTHON (voir ci-dessous). Ceci s'appelle un shebang dans le jargon UNIX, rendu possible grâce aux variables d'environnement de ce type de systèmes.



```
1 #!/usr/bin/env python3
2
3 ## La suite de Fibonacci (Suite)
4 ...etc...
```

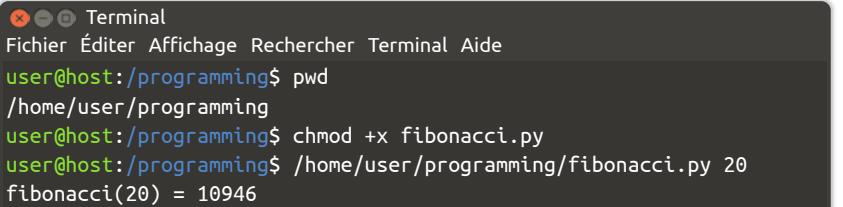
CODE 8.5 – Ligne d'appel PYTHON : « shebang »

AIDE EN LIGNE UNIX

Les systèmes d'exploitation type UNIX bénéficient d'une aide en ligne en saisisant dans un terminal, la commande `man` suivie du nom de la fonction sur laquelle on cherche de la documentation. La commande `man` est une contraction de *manual* en anglais. Par exemple, `man chmod` et `man pwd`.

Pour que cela puisse fonctionner, le fichier PYTHON doit être *exécutable*. Pour ce faire, on utilise les commandes UNIX « `pwd` », pour « *print working directory* » en anglais, qui affiche le chemin d'accès vers le répertoire où se situe l'utilisateur — et `chmod` — abréviation de « *change mode* » en anglais, commande qui permet de modifier les permissions et les modes d'accès des fichiers ou répertoires.

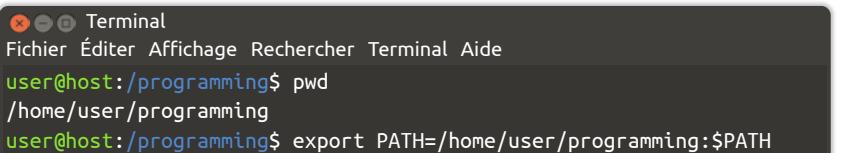
Dès lors, le fichier `fibonacci.py` peut s'utiliser comme une commande, sans avoir besoin de mentionner l'interpréteur `python3`, qui est désormais invoqué au travers du *shebang*.



```
Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ pwd
/home/user/programming
user@host:/programming$ chmod +x fibonacci.py
user@host:/programming$ ./fibonacci.py 20
fibonacci(20) = 10946
```

Par souci pratique, on peut encore aller plus avant et indiquer à la variable d'environnement PATH — pour chemin d'accès — du compte utilisateur, le répertoire où sont stockés tous les fichiers exécutables qui sont programmés (on constate ici tout l'intérêt d'avoir créé un répertoire à cet effet : `home > user > programming`). De cette manière, ils sont ainsi rendus accessibles à partir n'importe quel répertoire.

La manipulation a conduite peut se réaliser pour un usage temporaire, tant qu'on ne change pas de session, c'est-à-dire tant qu'on reste dans le même terminal. Il suffit de saisir dans le terminal la ligne de commande qui suit.

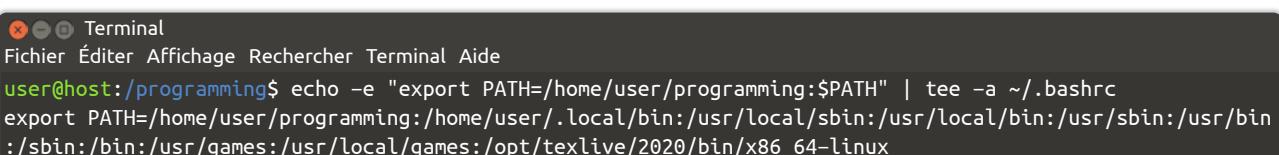


```
Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ pwd
/home/user/programming
user@host:/programming$ export PATH=/home/user/programming:$PATH
```

Pour une configuration pérenne, il faut rajouter la même ligne au fichier `.bashrc` qui gère les sessions utilisateur : `home > user > .bashrc`. Pour cela il faut l'édition ou employer la ligne de commande UNIX (voir la simulation de terminal ci-dessous; attention à bien stipuler l'option « `-a` » à la commande `tee`, sinon, cela écrase tout le fichier! Faire une sauvegarde du fichier avant toute manipulation!).

FICHIERS CACHÉS UNIX

Sous UNIX, les fichiers cachés ont leur nom précédé d'un point « `.` ». Pour les révéler, il faut saisir la commande `ls -a` pour *list all* ou, avec un bureau GNOME, taper les touches `[Ctrl]+[H]` dans le gestionnaire de fichiers (« `H` » pour *hide/hidden* en anglais).

Terminal


```
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:/programming$ echo -e "export PATH=/home/user/programming:$PATH" | tee -a ~/.bashrc
export PATH=/home/user/programming:/home/user/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
:/sbin:/bin:/usr/games:/usr/local/games:/opt/texlive/2020/bin/x86_64-linux
```

Le retour du terminal permet de vérifier tous les répertoires où le système va aller automatiquement chercher les programmes exécutables, par exemple les binaires pour `LATEX` (si `TeX Live` est installé et configuré). Surtout, on note un répertoire d'exécutables configuré par défaut `home > user > .local > bin` qui, une fois les programmes développés et stabilisés, peut accueillir pour exploitation, les programmes binaires de l'utilisateur.

Nota bene : Ce mécanisme fonctionne très bien tant que le point d'entrée — ici `fibonacci.py` — n'utilise que des modules standards. Au cas où il vient avec au moins un module externe, il est également nécessaire d'installer ces modules et d'indiquer comment les trouver. Cela est repris plus loin lors de la présentation des modules (cf. § 5).

1.4.2 Exercices d'application

A. SUITE DE FIBONACCI — PARTIE I

Fondons nous sur un premier exemple concret qui tourne : la¹⁷ suite de Fibonacci. L'enjeu est d'abord d'en détailler les étapes d'élaboration comme dans le *notebook* du Mooc, puis de l'exécuter en local sur une machine comme programme indépendant.

La suite de FIBONACCI se définit comme suit :

$$\begin{aligned} u_0 &= 1 \\ u_1 &= 1 \\ n \geq 2, u_n &= u_{n-1} + u_{n-2} \end{aligned} \quad (8.1)$$

Les premières valeurs numériques de la suite de FIBONACCI sont listées en *table 8.3*.

Pour programmer la suite de FIBONACCI, on commence par définir la fonction `fibonacci` comme il suit. Naturellement tout le bagage pour lire ce code n'est pas encore acquis, le but est uniquement de montrer un fonctionnement de l'interpréteur PYTHON et de IDLE.

In[1]

```
def fibonacci(n):
    "retourne le nombre de fibonacci pour l'entier n"
    # pour les petites valeurs de n il n'y a rien à calculer
    if n <= 1:
        return 1
    # sinon on initialise f1 pour n-1 et f2 pour n-2
    f2, f1 = 1, 1
    # et on itère n-1 fois pour additionner
    for i in range(2, n + 1):
        f2, f1 = f1, f1 + f2
    # print(i, f2, f1)
    # le résultat est dans f1
    return f1
```

Pour que cela soit un programme fonctionnel, on demande à l'utilisateur de rentrer un nombre ; il faut le convertir en entier car l'instruction `input` renvoie une chaîne de caractères :

In[2]

```
entier = int(input("Entrer un entier "))
```

Out[2]

Entrer un entier : 8

Le résultat est alors affiché une fois la cellule validée (`[Maj]+[Entrée]`).

In[3]

```
print(f"fibonacci({entier}) = {fibonacci(entier)}")
```

Out[3]

`fibonacci(8) = 34`

¹⁷. En mathématiques, une suite de FIBONACCI est un ensemble d'entiers dans lequel chaque élément est la somme des deux termes qui le précèdent (source W).

TABLE 8.3 — FIBONACCI : premières valeurs numériques.

Suite de FIBONACCI	
RANG n	FIBONACCI(n)
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89



EXERCICE 1 — SUITE DE FIBONACCI I



Le corps du programme étant défini, on peut à présent conduire les actions suivantes :

- exécuter le code dans le *notebook* du Mooc;
- télécharger ce code comme fichier exécutable `fibonacci_prompt.py` — `[File] > Download as > Python` — et au besoin le renommer;
- lancer le code sous IDLE;
- modifier le fichier source, par exemple pour afficher des résultats intermédiaires (la fonction `print` a été laissée en commentaire de manière à pouvoir être réactivée simplement);
- enfin, faire tourner le programme avec l'interpréteur PYTHON en ligne de commande : `$ python3 fibonacci_prompt.py`.

Ce code est volontairement simple et peu robuste pour ne pas l'alourdir. En effet et à titre d'illustration, ce programme se comporte mal si un entier négatif est en entrée d'algorithme.

AVERTISSEMENT

Cette version du programme ne fonctionne pas dans le *notebook*, justement car on n'a pas de moyen de l'invoquer en lui passant des arguments de cette manière. Le *notebook* est rédigé pour s'entraîner avec le téléchargement au format PYTHON et faire tourner le code en local sur un ordinateur.

B. SUITE DE FIBONACCI — PARTIE II

Toujours en reprenant comme guide illustratif la suite de FIBONACCI, une version légèrement différente est proposée pour permettre la saisie d'une valeur d'entrée en ligne de commande et non plus en répondant à une question.

MODULE argparse — Pour interpréter les arguments en ligne de commande, il est nécessaire d'importer le module particulier **argparse**.

In[]

```
from argparse import ArgumentParser
```

Il faut reprendre ensuite le code¹⁸ de la fonction de FIBONACCI.

In[]

```
def fibonacci(n):
    "retourne le nombre de fibonacci pour l'entier n"
    # pour les petites valeurs de n il n'y a rien à calculer
    if n <= 1:
        return 1
    # sinon on initialise f1 pour n-1 et f2 pour n-2
    f2, f1 = 1, 1
    # et on itère n-1 fois pour additionner
    for i in range(2, n + 1):
        f2, f1 = f1, f1 + f2
    #     print(i, f2, f1)
    # le résultat est dans f1
    return f1
```

À présent, pour dire au module **argparse** qu'on attend exactement un argument sur la ligne de commande et que celui-ci doit être un entier, on utilise un objet « **parser** ». Ici encore, ne pas s'inquiéter si le code est mal compris, l'objectif est de donner un morceau de code exploitable tout de suite, pour jouer avec l'interpréteur PYTHON.

In[]

```
# à nouveau : ceci n'est pas conçu pour être exécuté dans le
#             ↵ notebook !
parser = ArgumentParser()
parser.add_argument(dest="entier", type=int,
                    help="entier d'entrée")
input_args = parser.parse_args()
entier = input_args.entier
```

Le résultat est enfin affiché par une fonction **print**.

In[]

```
print(f"fibonacci({entier}) = {fibonacci(entier)}")
```

**EXERCICE 2 — SUITE DE FIBONACCI II**

Le corps du programme étant déterminé, on peut à présent conduire les actions suivantes :

- télécharger le code sur disque comme un fichier PYTHON exécutable **fibonacci.py** — [File](#) ➤ [Download as](#) ➤ [Python](#) — et au besoin le renommer;
- exécuter le programme avec simplement l'interpréteur PYTHON en ligne de commande : `$ python3 fibonacci.py 56`.

ATTENTION !

Être bien vigilant à sauvegarder le programme téléchargé sous un autre nom que **turtle.py**, car sinon cela empêche PYTHON de trouver le module standard **turtle**; le nommer en **turtle_basic.py** par exemple.

**EXERCICE 3 — DESSIN DE CARRÉ**

Il est ici proposé le dessin d'un carré. Pour ce faire le module **turtle**, conçu précisément à des fins pédagogiques est employé.

Pour des raisons techniques, le module **turtle** n'est pas disponible au travers de la plateforme FUN-Mooc. Il s'avère donc inutile d'essayer d'exécuter ce programme depuis le *notebook*. L'objectif de cet exercice est toujours de s'entraîner à télécharger le programme en utilisant le menu [File](#) ➤ [Download as](#) ➤ [Python](#) puis le charger en local pour l'exécuter avec IDLE.

```
>>> import turtle
```

On commence par définir une fonction qui dessine un carré de côté « `length` » — l'anglais reste la norme des codes informatiques :

```
>>> def square(length):
...     "Have the turtle draw a square of side <length>"
...     for side in range(4):
...         turtle.forward(length)
...         turtle.left(90)
... 
```

On initialise ensuite la tortue.

```
>>> turtle.reset()
```

On peut alors dessiner un carré :

```
>>> square(200)
```

Pour finir, on attend que l'utilisateur clique dans la fenêtre de la tortue et ainsi clôturer le programme :

```
>>> turtle.exitonclick()
```



EXERCICE 4 — DESSIN DE CARRÉ



Naturellement, il est possible de modifier le code de l'exercice précédent pour dessiner des choses plus amusantes. Pour cela, il faut d'abord chercher « `module python turtle` » dans un moteur de recherche pour en localiser la documentation.

Pour débuter, des exemples sont disponibles aux liens ci-après :

- ▶ `turtle_multi_squares.py` pour dessiner des carrés à l'emplacement de la souris en utilisant plusieurs tortues;
- ▶ `turtle_fractal.py` pour dessiner une fractale simple;
- ▶ `turtle_fractal_adjustable.py` variation de fractale, plus paramétrable.

[Solution page 275](#)



SOLUTION 4 — DESSIN DE CARRÉ (PAGE 275)

Les sources des codes présentés en lien dans l'exercice sont reproduites ci-dessous pour lecture hors ligne.

```
1 import turtle # Turtle module                                     CODE 8.6 — turtle_multi_squares.py
2
3 # Avoiding to call range for each square
4 sides = ['east', 'north', 'west', 'south']
5
6 def square(the_turtle, length):
7     "have the turtle draw a square of side <length>"
8     for side in sides:
9         the_turtle.forward(length)
10        the_turtle.left(90)
11
12 # Initializing
13 window = turtle.Screen()
14 window.title("Caroline et Chloé")
15
16 # Creating first turtle
17 caroline = turtle.Turtle()
18 caroline.reset()
```

```

19 caroline.color("hotpink")
20
21 # Creating second turtle
22 chloe = turtle.Turtle()
23 chloe.reset()
24 chloe.color("lightgreen")
25
26 # Alternate : turtle, twist and square size
27 contexts = ((caroline, 15, 100, ),
28             (chloe, 60, 30),
29             )
30 # Initializing alternate contexts
31 cycle = len(contexts)
32 counter = -1
33
34 # The callback triggered when a user clicks in x,y
35 def clicked(x, y):
36     global counter
37     counter += 1
38     # Alternate between the various contexts
39     (turtle, twist, size) = contexts[counter % cycle]
40     turtle.penup()
41     turtle.goto(x, y)
42     turtle.pendown()
43     turtle.left(twist)
44     square(turtle, size)
45
46 turtle.onscreenclick(clicked) # Arm callback
47
48 turtle.onkey(turtle.bye, 'q') # User can quit by typing 'q'
49 turtle.listen()
50
51 turtle.mainloop() # Reading and dispatching events

```

CODE 8.7 – turtle_fractal.py

```

1 import turtle
2
3 def left_triangle(length):
4     for i in range(3):
5         turtle.forward(length)
6         turtle.left(120)
7
8 def fractal_side(length, fractal):
9     if fractal == 0:
10         turtle.forward(length)
11     else:
12         length3 = length / 3.
13         fractal_side(length3, fractal-1)
14         turtle.right(60)
15         fractal_side(length3, fractal-1)
16         turtle.left(120)
17         fractal_side(length3, fractal-1)
18         turtle.right(60)
19         fractal_side(length3, fractal-1)
20
21 def fractal_triangle(length, fractal):
22     for i in range(3):
23         fractal_side(length, fractal)
24         turtle.left(120)
25
26 turtle.reset()

```

```

27 turtle.speed('fastest')
28 fractal_triangle(300, 3)
29 left_triangle(300)
30 turtle.exitonclick()

```

CODE 8.8 — turtle_fractal_adjustable.py

```

1 import turtle
2
3 def left_triangle(length):
4     for i in range(3):
5         turtle.forward(length)
6         turtle.left(120)
7
8 def fractal_side(length, fractal, proportions):
9     if fractal == 0:
10         turtle.forward(length)
11     else:
12         [l1, l2, l3] = [p * length for p in proportions]
13         fractal_side(l1, fractal - 1, proportions)
14         turtle.right(60)
15         fractal_side(l2, fractal - 1, proportions)
16         turtle.left(120)
17         fractal_side(l3, fractal - 1, proportions)
18         turtle.right(60)
19         fractal_side(l3, fractal - 1, proportions)
20
21 def fractal_triangle(length, fractal, proportions):
22     for i in range(3):
23         fractal_side(length, fractal, proportions)
24         turtle.left(120)
25
26 turtle.reset()
27 turtle.speed('fastest')
28 fractal_triangle(300, 4, (.1, .5, .4,))
29 left_triangle(300)
30 turtle.exitonclick()

```

2 Notions de variable, d'objet et de typage

Pour aborder PYTHON, il faut situer le décor en présentant les notions d'*objet*, de *variable* et de *typage dynamique*. Pourquoi ces notions sont-elles essentielles en PYTHON ? Parce qu'en PYTHON, tout est un objet – au sens du paradigme de la « programmation orientée objet ».

2.1 Nommage et mots-clefs

Conséquence de la programmation orientée objet, seuls des *objets* sont à manipuler dans les programmes. Le moyen de faire est de leur donner un nom par l'intermédiaire de *variables*. On dit que les variables *référencent* les objets.

2.1.1 Types, objets et méthodes

Focalisons à présent l'attention sur la *notion d'objet*. Dans un programme informatique, un objet est un morceau de code qui va contenir des données. Mais il va également rassembler un ensemble de méca-



VIDÉO 8.5 — Variables, objets et typage dynamique.



nismes permettant de manipuler ces données et que l'on appelle des *méthodes*. Cette formulation est commune aux langages orientés objet.

Les objets ont tous un type. Le type est le comportement par défaut qui va être déterminé pour ces objets. Par suite, le type va permettre de définir les données et les méthodes qui vont être associées à cet objet.

Prenons l'analogie d'une chaîne de montage de voitures. Dans une usine, des voitures sont construites selon des processus qui vont déterminer un ensemble de comportements que toutes les voitures qui sortent de la chaîne de montage vont posséder. Donc par exemple, la puissance du moteur, le fait que la voiture va avoir des clignotants, un accélérateur, vont être engendrés par la chaîne de montage. On peut alors dire que la chaîne de montage définit le type de voitures qui en sort et que la voiture en est l'objet.

Pour revenir aux programmes informatiques présents dans la mémoire de l'ordinateur, créons un premier objet PYTHON de type chaîne de caractères (abordé plus en détail par la suite) supposée associée au mot « *spam* ». Pour ce faire, la chaîne de caractères doit être positionnée entre deux apostrophes, soit : '*spam*'. Après validation par la touche **[Entrée]** ou **[CR]¹⁹** l'interpréteur PYTHON va créer cet objet.

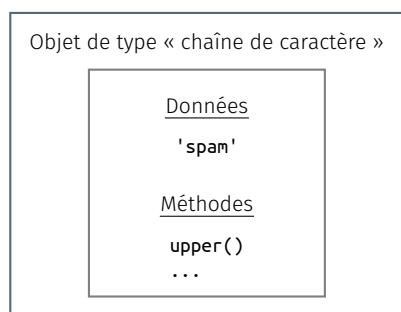
Cette chaîne de caractères a également un ensemble de méthodes comme par exemple **upper()**, qui correspond à la mise en capitales des lettres de la chaîne de caractères, soit « SPAM » pour l'exemple donné.

D'où viennent les méthodes sur les chaînes de caractères puisque rien n'a été défini ? En fait, toutes les méthodes viennent du type « chaîne de caractères ». Un type est donc un objet associé à toutes les méthodes qui lui sont applicables.

Pour appeler une méthode sur un objet, il suffit de le relier à la méthode appelée par la notation point « . » : '*spam*'.**upper()**. Il faut bien de mettre les parenthèses ouvrante et fermante car elles déclenchent l'exécution de la méthode sur l'objet, ici **upper()** sur '*spam*'.

Ainsi, chaque objet possède un type. On peut très simplement accéder au type d'un objet en appelant une fonction « *built-in* », c'est-à-dire prédéfinie dans PYTHON, qui s'appelle... **type** !

¹⁹. L'expression « Retour chariot » ou *Carriage return* est un vestige des claviers de machine à écrire mécanique.



Appel de méthode : `>>> 'spam'.upper()`
Affectation : `>>> note = 1`

Type d'objet, données, méthodes.

In[1]	<code>type(1)</code>
Out[1]	<code>int</code>
In[2]	<code>type('spam')</code>
Out[2]	<code>str</code>

Cette fonction est assez peu employée par les programmeurs expérimentés, mais elle est utile à bien comprendre le langage, notamment pour manipuler les valeurs numériques.

FONCTION `isinstance` — Une autre fonction prédéfinie, voisine de **type** mais plus utile en pratique, est la fonction **isinstance**. Elle permet de savoir si un objet est d'un type donné. Par exemple :

In[3]	<code>isinstance(23, int)</code>
Out[3]	<code>True</code>

À la vue de ce seul exemple, on pourrait penser que **isinstance** est presque identique à **type**; en réalité elle est un peu plus élaborée, notamment pour la programmation objet et l'héritage (abordés par la suite). On remarque en passant que la variable **int** est connue de PYTHON alors qu'elle n'a pas été définie. Il s'agit d'une variable prédéfinie, qui désigne le type des entiers (voir § 3).

En anticipant un peu sur la suite (cf. § 2.1.3), la fonction **isinstance** est utile car PYTHON est à typage dynamique. Aussi, il est souvent perti-

nant de s'assurer qu'une variable passée à une fonction est du (ou des) type(s) attendu(s), puisque contrairement à un langage typé statiquement comme C++, on n'a aucune garantie de ce genre à l'exécution.

2.1.2 Référencement et variables

Après la notion d'objet, comment les distinguer entre eux ? Cette différenciation s'opère en leur attribuant un nom. Plus exactement, le vocabulaire consacré est de parler de *référencement* des objets.

Pour référencer un objet, on le met en correspondance avec un *nom de variable* au moyen de la notation d'affectation représentée par le signe « égale » : « = ». Pour par exemple référencer l'objet « 1 » de type entier, on peut écrire : **note** « égale » 1 — **note** = 1. Désormais, on peut donc manipuler cet objet par l'intermédiaire de son nom de variable.

Un nom de variable en PYTHON se définit par n'importe quelle lettre en minuscule ou en majuscule, associées aux chiffres allant de 0 à 9 et également au caractère « tiret bas » ou « *underscore* » en anglais.

Un nom de variable peut indifféremment commencer par une lettre ou l'*underscore* mais pas par un chiffre. Un nom de variable est sensible à la casse, c'est-à-dire qu'il y a distinction entre lettres capitales²⁰ et minuscules ; deux variables sont distinctes si elles diffèrent d'une lettre en capitale et en bas-de-casse.

Par ailleurs, il est très important en PYTHON d'attribuer des noms de variable qui soient explicites. Par exemple, **moyenne_age_français** est un bon nom de variable, meilleur que **moy_age_f** et encore bien meilleur que simplement la variable **x**. En effet, PYTHON offre de nombreux mécanismes pour faciliter le nommage explicite des objets que l'on manipule, notamment pour la documentation automatique des codes (cf. concept de *docstring* en vidéo 8.3 rediscuté par la suite).

Les noms les plus simples sont constitués de lettres minuscules et/ou capitales. Par exemple :

```
In[] factoriel = 1
Factoriel = 100
factoriel == Factoriel
```

Le signe « == » permet de tester si deux variables ont la même valeur. Si tel est le cas, le test retourne le booléen **True** et, **False** sinon.

CONVENTIONS HABITUELLES

En règle générale, on utilise uniquement des minuscules pour désigner les variables simples et pour les noms de fonctions. Quant à elles, les capitales sont réservées pour d'autres sortes de variables, comme les noms de classe (à voir ultérieurement). Notons qu'il ne s'agit que d'une convention, ceci n'est pas imposé par le langage lui-même.

TIRET-BAS — Pour des raisons de lisibilité, il est également possible d'utiliser le tiret-bas « _ » dans les noms de variables. On préfère ainsi la première solution ci-dessous à la seconde :

```
In[] age_moyen = 75 # oui
AgeMoyen = 75 # autorisé, mais non
```

On peut également utiliser des chiffres dans les noms de variables comme par exemple :

```
In[] age_moyen_dept75 = 80
```

avec la restriction toutefois que le premier caractère ne peut pas être un chiffre, cette affectation est donc refusée :

```
In[] 75_age_moyen = 80 # erreur de syntaxe
```

²⁰. *Stricto sensu*, le terme de *majuscule* désigne la première lettre en *capitale* d'un début de phrase, d'où l'amalgame courant entre lettres capitales et majuscules.



En revanche, s'il est possible de faire commencer un nom de variable par un tiret bas comme premier caractère, à cette étape d'apprentissage, il est déconseillé de le faire; cette pratique est réservée à des conventions de nommage bien spécifiques.

```
In[] _autorise_mais_deconseille = 'Voir le PEP 008'
```

En tout état de cause, il est fortement contre-indiqué d'utiliser des noms de la forme `_variable` qui sont réservés au langage. Ce point sera évoqué par la suite, mais par exemple la variable qui suit n'a été définie nulle part mais qui elle existe bel et bien.

```
In[] __name__ # ne définissez pas vous-même de variables de ce genre
```

PONCTUATION — Dans l'étendue des caractères ASCII, il n'est pas possible d'utiliser d'autres caractères que les caractères alphanumériques et le tiret-bas. Notamment le tiret-haut « - » est interprété comme l'opération de soustraction. Attention à cette erreur fréquente :

```
In[] # erreur : en fait Python l'interprète comme 'age - moyen = 75'
age-moyen = 75
```

CARACTÈRES EXOTIQUES — En PYTHON3, il est maintenant possible d'employer des caractères Unicode dans les identificateurs :

```
In[] # caractères latins étendus/accents permis
nom_élève = "Jules Maigret"
# alphabet grec permis : α β γ δ ε ζ η θ κ λ μ ν ξ π σ τ υ φ χ ψ ω
from math import cos, pi as π
θ = π / 4
cos(θ)
```

Tous les caractères Unicode ne sont cependant pas permis — fort heureusement car cela serait source de confusion. Les documents qui précisent l'ensemble des caractères autorisés sont proposés en référence (cf. infra).

```
In[] # caractère interdit car estimé comme signe mathématique (produit)
Π = 10
# caractère encore différent, aussi un pi grec, cette fois-ci
# → acceptable comme nom de variable mais non défini
π
```

Il est très vivement recommandé :

- ▶ tout d'abord de coder *en anglais*;
- ▶ ensuite de *ne pas définir* des identificateurs avec des caractères non ASCII, voir par exemple la confusion que peut créer le fait de nommer un identificateur `π`, `Π` ou `π`;
- ▶ enfin si l'encodage employé n'est pas UTF-8 — plus rare désormais —, *dûment spécifier l'encodage dans le fichier source* (sujet abordé par la suite).

POUR ALLER PLUS LOIN...

Pour les esprits curieux, Guido VAN ROSSUM, le fondateur de PYTHON, est le co-auteur d'un document qui décrit les conventions de codage à utiliser dans la bibliothèque standard PYTHON. Ces règles sont plus restrictives que ce que le langage permet de faire, mais constituent une lecture intéressante pour un projet qui nécessiterait d'écrire beaucoup de code.

- ▶ PEP 008, section consacrée aux règles de nommage.

Au sujet des caractères exotiques dans les identificateurs :

- ▶ PEP 3131 qui définit les caractères exotiques autorisés;
- ▶ et qui repose à son tour sur (très technique!) :

<http://www.unicode.org/reports/tr31/>.

2.1.3 Typage dynamique et gestion de la mémoire

La dernière notion importante à appréhender ici est le *typage dynamique*. Comme précédemment évoqué, on dispose d'une part, de l'*espace des objets* et, d'autre part, de l'*espace des variables* — ce dernier sera revu par la suite sous le nom *d'espace de nommage*.

Quels liens existe-t-il entre ces différents espaces ? Supposons que l'on saisisse « `a` égale 3 » (`a = 3`), ce qui signifie que la variable « `a` » va référencer l'entier « 3 ». Lors du retour chariot, PYTHON va exécuter trois opérations (voir figure 8.7) :

1. la première va consister à créer l'entier « 3 » ; c'est donc un objet engendré dans l'espace des objets ;
2. puis, il va produire la variable « `a` » dans l'espace des variables ;
3. et, pour finir, il va établir une référence entre cette variable « `a` » et l'entier « 3 ».

Maintenant, supposons que l'on change d'objet et que l'on entre « `a` égale 'spam' » (`a = 'spam'`). PYTHON va alors suivre le même parcours en élaborant un objet de type chaîne de caractères qui s'appelle « '`spam`' » et une variable « `a` ». Cependant, la variable « `a` » existe déjà. La dernière opération va donc simplement consister à supprimer (« déréférer ») le lien avec l'objet « 3 » pour que la variable « `a` » référence désormais l'objet « '`spam`' » (cf. figure 8.7).

En fait, que signifie le typage dynamique ? Cela exprime que le type n'est pas lié à la variable qui référence l'objet mais est associé à l'objet. Pour aller plus loin, PYTHON est un langage que l'on appelle à *typage fort*, autrement dit, le typage est relatif aux objets et l'objet va garder le même type durant toute l'exécution du programme. En revanche, quant à la variable, elle peut référencer des objets qui vont être de types différents en cours d'exécution.

Pour terminer provisoirement, en exécutant « `del a` », cela supprime la variable « `a` » de l'espace des variables. Si l'objet n'a plus de référence, un mécanisme qui s'appelle « *garbage collector* » en PYTHON — et d'autres langages comme JAVA — va libérer automatiquement²¹ la mémoire de l'ordinateur une fois que les objets ne sont plus référencés.

2.2 Compléments et application

2.2.1 Mot-clefs PYTHON

Il existe en PYTHON certains mots spéciaux, qu'on appelle des mots-clés, ou *keywords* en anglais, qui sont réservés et ne peuvent pas être employés comme identifiants, c'est-à-dire comme nom de variable.

C'est le cas par exemple pour l'instruction `if`, qui permet bien entendu d'exécuter un code selon le résultat d'un test.

```
In[] variable = 15
      if variable <= 10:
          print("en dessous de la moyenne")
      else:
          print("au dessus de la moyenne")
```

Sa présence dans le langage interdit d'appeler une variable `if`.

```
In[] if = 1 # interdit, if est un mot-clé
```

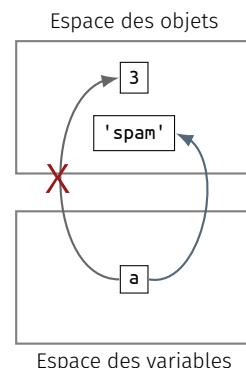


FIGURE 8.7 — Typage dynamique.

²¹ Ce n'est pas le cas en C, où pour optimiser les programmes, il faut explicitement libérer la mémoire (cf. infra).

22. Pour plus amples renseignements : https://docs.python.org/3/reference/lexical_analysis.html#keywords.

TABLE 8.4 — Mots-clés PYTHON.

Mots réservés en PYTHON		
False	del	lambda
True	elif	nonlocal
None	else	not
and	except	or
as	finally	pass
assert	for	raise
async	from	return
await	global	try
break	if	while
class	import	with
continue	in	yield
def	is	

23. Si ce cours est suivi au premier niveau d'approche, point besoin de ce complément : il suffit juste de retenir que Python se charge de tout !

La liste complète des mots-clés réservés est fournie par la table 8.4. Y sont indiqués²² en gras les nouveautés par rapport à PYTHON 2 — sachant que réciproquement **exec** et **print** ont perdu leur statut de mot-clé depuis PYTHON 2, ce sont maintenant des fonctions. Il faut donc y prêter attention, surtout au début, mais avec un tout petit peu d'habitude on arrive rapidement à les éviter.

Notons également que tous les bons éditeurs de texte supportant du code PYTHON vont colorer les mots-clés différemment des variables. Par exemple, IDLE colorie les mots-clés en orange, on peut très facilement se rendre compte d'éventuelles erreurs de nom de variable.

Cette fonctionnalité, dite de coloration syntaxique, permet d'identifier d'un coup d'œil, grâce à une convention de couleur, le rôle des différents éléments d'un code : variables, mots-clés, etc. D'une manière générale, il est fortement déconseillé d'utiliser un éditeur de texte qui n'offre pas cette fonctionnalité essentielle.

2.2.2 Gestion de la mémoire

L'objet de ce complément est de montrer qu'avec PYTHON il n'y a pas à se préoccuper de la mémoire. Pour expliquer la notion de gestion de la mémoire, il faut donner un certain nombre de détails sur d'autres langages²³ comme C et C++.

i – LANGAGES DE BAS NIVEAU — Dans un langage traditionnel de bas niveau comme C ou C++, le programmeur est en charge de l'allocation — et donc de la libération — de la mémoire.

Cela signifie que, sauf pour les valeurs stockées dans la pile, le programmeur est amené :

- à réclamer de la mémoire au système d'exploitation en appelant explicitement **malloc** (C) ou **new** (C++);
- et, réciproquement, à rendre cette mémoire au système d'exploitation lorsqu'elle n'est plus utilisée, en appelant **free** (C) ou **delete** (C++).

Avec ce genre de langage, la gestion de la mémoire est un aspect important de la programmation. Ce modèle offre une grande flexibilité, mais au prix d'un coût élevé en matière de vitesse de développement.

En effet, il est assez facile d'oublier de libérer la mémoire après usage, ce qui peut conduire à épuiser les ressources disponibles. À l'inverse, utiliser une zone mémoire non allouée peut conduire à des bugs très difficiles à localiser et à des problèmes de sécurité majeurs. Notons qu'une grande partie des attaques en informatique reposent sur l'exploitation d'erreurs de gestion de la mémoire.

ii – LANGUAGE DE HAUT NIVEAU — Pour toutes ces raisons, avec un langage de plus haut niveau comme PYTHON, le programmeur va s'affranchir de cet aspect de la programmation.

Pour anticiper un peu sur les parties du Mooc à venir, voici ce qui est à garder en tête s'agissant de la gestion mémoire en PYTHON :

- les objets sont créés au fur et à mesure des besoins;
- il n'y a pas à les libérer explicitement, le « *Garbage Collector* » s'en charge pour recycler la mémoire lorsque c'est possible;
- PYTHON a tendance à être assez gourmand en mémoire, comparé à un langage de bas niveau, car tout est objet et chaque objet est assorti de métainformations qui occupent une place non négligeable. Par exemple, chaque objet possède au minimum :
 - a. une référence vers son type — prix du typage dynamique;
 - b. un compteur de références — le nombre d'autres valeurs (variables ou objets) qui pointent vers l'objet. Cette information

- sert notamment au « *Garbage Collector* » pour déterminer si la mémoire empruntée par un objet peut être libérée ou non.
- un certain nombre de types prédéfinis et non mutables sont implementés en PYTHON, comme des singletons²⁴ c'est-à-dire qu'un seul objet est créé et partagé, c'est le cas par exemple pour les petits entiers et les chaînes de caractères;
 - lorsqu'on implémente une classe, il est possible de lui conférer cette caractéristique de singleton, de manière à optimiser la mémoire nécessaire pour exécuter un programme.

²⁴. En mathématiques, le singleton est un ensemble composé d'un seul élément. En génie logiciel, c'est un patron de conception (*design pattern*) dont la perspective est de restreindre à un seul objet l'instantiation d'une classe (source [W](#)).

2.2.3 Typages statique versus dynamique

Parmi les langages typés, on distingue les langages à typage statique de ceux à typage dynamique. On va ici s'attacher à tenter d'éclaircir ces notions pour ceux qui n'en sont pas familiers.

- i – **TYPAGE STATIQUE** — À une extrémité du spectre, on trouve les langages compilés, dits à typage statique, comme par exemple C ou C++.

En C on écrit, par exemple, une version simpliste de la fonction factorielle comme il suit.

C

```
int factorielle(int n) {
    int result = 1;
    for (int loop = 1; loop <= n; loop++)
        result *= loop;
    return result;
}
```

Calcul de factorielle en C

Comme on peut le voir — ou le deviner —, toutes les variables employées ici (comme par exemple **n**, **result** et **loop**) sont typées :

- on doit appeler **factorielle** avec un argument **n** qui doit être un entier (**int** est le nom du type entier);
- les variables internes **result** et **loop** sont de type entier;
- **factorielle** retourne une valeur de type entier.

Ces informations de type ont essentiellement trois fonctions :

1. en premier lieu, elles sont nécessaires au compilateur. En C si le programmeur ne précise pas que **result** est de type entier, le compilateur n'a pas suffisamment d'éléments pour générer le code assembleur correspondant;
2. en contrepartie, le programmeur a un contrôle très fin de l'usage qu'il fait de la mémoire et du matériel. Il peut choisir un entier sur 32 ou 64 bits, signé ou pas ou encore, construire avec **struct** et **union** un arrangement de ses données;
3. enfin et surtout, ces informations de type permettent de faire un contrôle *a priori* de la validité du programme, par exemple, si à un autre endroit dans le code on trouve :

C

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    /* premier argument de la ligne de commande : argv[1] */
    char *input = argv[1];
    /* calculer sa factorielle et afficher le résultat */
    printf("Factorielle (%s) = %d\n", input, factoriel(input));
    /*
     * ici on appelle factorielle avec une entrée de type
     * 'chaîne de caractères'
    }
```

alors le compilateur détecte l'appel de **factorielle** avec comme argument **input** qui, pour faire simple, est une chaîne de carac-

tères et, comme `factorielle` s'attend à recevoir un entier, ce programme n'a aucune chance de compiler.

C'est ce qu'on appelle le *contrôle de type*, ou *type-checking* en anglais. Si on ignore le point sur le contrôle fin de la mémoire, qui n'est pas crucial à notre sujet, ce modèle de contrôle de type présente :

- l'inconvénient de demander davantage au programmeur (faisant abstraction, à ce stade et pour encore simplifier, de langages à inférence de types comme ML et HASKELL);
- l'avantage d'avoir un contrôle étendu et surtout précoce (avant même de l'exécuter), de la bonne correction du programme.

Cela étant dit, le typage statique en C n'empêche pas le programmeur débutant d'essayer d'écrire dans la mémoire à partir d'un pointeur `NULL` — et le programme de s'interrompre brutalement. Il faut être conscient des limites du typage statique.

ii – **TYPAGE DYNAMIQUE** — À l'autre extrémité du spectre, on trouve des langages qui font appel au typage dynamique, comme... PYTHON. Pour comprendre cette notion, regardons la fonction `somme` suivante.

```
In[1] def somme(*largs):
    "retourne la somme de tous ses arguments"
    if not largs:
        return 0
    result = largs[0]
    for i in range(1, len(largs)):
        result += largs[i]
    return result
```

Naturellement, en état d'avancement du Mooc, on est pas en mesure de comprendre le fonctionnement intime de la fonction. Mais elle peut s'employer :

```
In[2] somme(12, 14, 300)
```

```
Out[2] 326
```

```
In[3] liste1 = ['a', 'b', 'c']
liste2 = [0, 20, 30]
liste3 = ['spam', 'eggs']
somme(liste1, liste2, liste3)
```

```
Out[3] ['a', 'b', 'c', 0, 20, 30, 'spam', 'eggs']
```

On peut donc constater que `somme` peut fonctionner avec des objets de types différents. En fait, telle que la fonction est écrite, elle va fonctionner s'il est possible de faire une addition « + » entre ses arguments. Ainsi, par exemple, on pourrait même faire :

```
In[4] # Python sait faire + entre deux chaînes de caractères
somme('abc', 'def')
```

```
Out[4] 'abcdef'
```

En revanche, on ne peut pas faire :

```
In[5] # ceci va déclencher une exception à l'exécution
somme(12, [1, 2, 3])
```

```
Out[5] -----
```

```
TypeError                                     Traceback (most recent call last)
<ipython-input-5-1b5269e9e129> in <module>
      1 # ceci va déclencher une exception à l'exécution
----> 2 somme(12, [1, 2, 3])
<ipython-input-1-29005c25d5bb> in somme(*largs)
      5     result = largs[0]
      6     for i in range(1, len(largs)):
----> 7         result += largs[i]
```

```
Out[5]      8     return result
TypeError: unsupported operand type(s) for +=: 'int' and 'list'
```

Il s'avère pertinent de remarquer que le typage de PYTHON, qui existe bel et bien, est qualifié de dynamique parce que le type est attaché à un objet — méthodes et attributs — et non à la variable qui le référence.

En PYTHON, on fait souvent référence au typage dynamique sous l'appellation imagée de *duck typing* (cf. citation ci-contre) :

On voit qu'on se trouve dans une situation très différente de celle du programmeur C/C++, en ce sens que :

- à l'écriture du programme, il n'y a aucun surcoût en matière de définition de type;
- aucun contrôle de type n'est effectué *a priori* par le langage au moment de la définition de la fonction `somme`;
- par contre, au moment de l'exécution, s'il s'avère qu'on tente de faire une somme entre deux types qui ne peuvent pas être additionnés, comme ci-dessus avec un entier et une liste, le programme ne peut pas se dérouler correctement.

Au regard de la question du typage, deux points de vue coexistent :

1. les personnes habitués au typage statique se plaignent du typage dynamique en disant qu'on peut écrire des programmes faux et qu'on s'en rend compte trop tard, à l'exécution;
2. à l'inverse les partisans du typage dynamique font valoir que le typage statique est très partiel, par exemple, en C si on essaie d'écrire dans un pointeur `NULL`, le système d'exploitation ne le permet pas et le programme sort tout aussi brutalement.

Selon l'approche, le typage dynamique est donc vécu comme inconvenient (pas assez de bonnes propriétés détectées par le langage) ou comme avantage (pas besoin de passer du temps à déclarer le type des variables, ni à faire des conversions pour satisfaire le compilateur). Cependant, à l'usage, chacun peut constater qu'en matière de vitesse de développement, les inconvénients du typage dynamique sont très largement compensés par ses avantages.

iii – **TYPE hints** — Signalons enfin que depuis PYTHON 3.5, il est possible d'ajouter des annotations de type, pour expliciter les suppositions qui sont faites par le programmeur pour le bon fonctionnement du code.

Ces annotations sont totalement optionnelles et, même présentes, ne sont pas utilisées²⁵ à l'exécution par l'interpréteur. L'idée est plutôt de permettre à des outils externes, à l'instar de `mypy`, d'effectuer des contrôles plus poussés concernant la correction du programme.

3 Types numériques

PYTHON offre quatre types d'objets numériques : les nombres entiers (`int`), décimaux (`float`), complexes (`complex`) et booléens (`bool`). Après leur présentation, certains aspects du calcul numérique sont évoqués.

3.1 Généralités

3.1.1 Quatre types numériques

Ouvrons un interpréteur PYTHON et listons une à une les propriétés des différents types numériques.

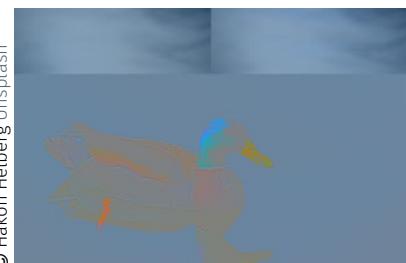
A. NOMBRE ENTIER

Pour créer un entier en PYTHON, il suffit de le saisir dans l'interpréteur. Cependant, une fois la saisie validée, il n'y a aucun moyen de le



If it looks like a duck and quacks like a duck, then it must be a duck.

— Attribué à James W. RILEY



© Håkon Helberg Unsplash

²⁵. Ce point sera repris par la suite.

NOTE DE LA RÉDACTION

Jusqu'à présent, de manière à ce qu'un nouvel utilisateur puisse y trouver ses marques, les environnements des différents interpréteurs PYTHON ont été simulés tels que disponibles sur une plateforme UBUNTU MATE. Pour des raisons pratiques — utilisation de l'extension PYTHONTEX — et de lisibilité — la coloration syntaxique sur fond noir est délicate —, et contrairement aux illustrations du Mooc, ce n'est pas l'interpréteur PYTHON mais IDLE qui est dorénavant pris en exemple. Les différences sont minimales.

manipuler car aucun nom lui a été attribué. Comme évoqué précédemment (cf. § 2.1.2), il faut passer par le mécanisme de référencement et par exemple stipuler : `i = 1`. L'objet « 1 » est désormais référencé par la variable « `i` ». On peut s'assurer de son type au moyen de la fonction prédéfinie — *built-in* — `type` (voir § 2.1.1).

```

Terminal
Fichier Éditer Affichage Rechercher Terminal Aide
user@host:~/programming$ ipython
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: 1
Out[1]: 1
In [2]: i = 1
In [3]: type(i)
Out[3]: int

```

Les entiers se manipulent comme dans une calculatrice. Ainsi, on peut leur appliquer les quatre opérations de base que sont l'addition — `1 + 4 = 5` —, la soustraction — `1 - 4 = -4` —, la multiplication — `3 * 5 = 15` — et la division naturelle — `3 / 6 = 0.5` —, mais encore des fonctions plus élaborées comme les puissances — `3 ** 3 = 27` — ou la valeur absolue — `abs(-4) = 4` —.

```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.

>>> 1
1
>>> i = 1
>>> type(i)
<class 'int'>
>>> 1 + 4
5
>>> 1 - 4
-3
>>> 3 * 5
15
>>> 3 / 6
0.5
>>> 3 ** 3
27
>>> abs(-4)
4
>>> 3 // 6
0
>>> 3 % 6
3
>>> i = i + 5
>>> print(i)
6

```

²⁶ À remarquer que l'on emploie soit la fonction `print()` ou le retour chariot. Ces deux fonctionnalités sont très proches et ne diffèrent de comportement que pour les chaînes de caractères. Cette différence est détaillée par la suite.

À noter qu'il existe aussi la division entière — `3 // 6 = 0` — dont on obtient le reste avec l'opérateur modulo « % » — `3 % 6 = 3`.

On peut bien entendu également réaffecter une variable avec le résultat d'une opération : `i = i + 5`, l'entier `i` vaut 1 donc `i + 5 = 6`. On peut le vérifier à l'aide de la fonction *built-in* `print` qui permet d'afficher le contenu²⁶ de la variable : `print(i) -> 6`.

Les entiers ont la caractéristique remarquable d'être de précision illimitée. En affectant à la variable `i` un entier extrêmement grand, on peut le multiplier par lui même ou l'élever à la puissance 3 voire plus; on obtient toujours un entier sans aucune troncation de précision.

```
>>> i = 39457983745972375892347
>>> i * i
1556932481297420209334461106675047624533168409
>>> i ** 3
61433416540610046741179271479879368323512136887733673946000605265923
```

B. NOMBRE DÉCIMAL

Le type `float` est le deuxième type d'objet numérique en PYTHON. Il est dévolu aux nombres décimaux. Primeur à l'anglais, le séparateur de décimales est le point, en lieu et place de la virgule en français.

Les décimaux ont une précision contrainte, généralement codés sur 15 chiffres significatifs et encodés sur 53 bits. Cela peut dépendre, évidemment, de la plateforme sur laquelle on fait tourner l'interpréteur.

```
>>> f = 4.3
```

C. NOMBRE COMPLEXE

Viennent ensuite les nombres complexes. Leur représentation est traditionnelle, en juxtaposant deux nombres décimaux, dont le second, pour la partie imaginaire, est suffixé par la lettre « `j` », ainsi : `4 + 5j`.

```
>>> c = 4 + 5j
```

D. CONVERSION DE TYPE

Stricto sensu, PYTHON se charge de toutes les conversions possibles. Bien entendu, il peut y avoir alors perte de précision.

L'addition d'un entier et d'un décimal donne un nombre décimal, avec perte de précision sur le nombre entier le cas échéant. En additionnant un entier, un décimal et un complexe, PYTHON va fournir un complexe, toujours avec perte de précision si cela se présente.

La conversion d'un décimal vers un entier se réalise avec la fonction prédéfinie `int()` et va induire une troncation. À l'inverse la conversion d'un entier vers un décimal s'opère avec la fonction prédéfinie `float()`. Il en est de même pour les complexes avec `complex()`.

```
>>> i = 39457983745972375892347597349285789347597349579
>>> f = 4.3
>>> i + f
3.9457983745972377e+46
>>> i + f + c
(3.9457983745972377e+46+5j)
>>> int(4.3)
4
>>> float(4)
4.0
>>> complex(4.0)
(4+0j)
```

E. VALEUR BOOLÉENNE

Le dernier type numérique à présenter est celui des valeurs booléennes. En tant que tel, ce type est particulier car *a priori* non censé faire partie des types numériques : valeur `True/False` pour vrai/faux. Bien noter que ces valeurs s'écrivent avec une première lettre capitale.

²⁷. D'autres exemples d'application plus élaborés sont fournis par la suite.

En PYTHON comme quelques autres langages, le type booléen est codé en tant que sous-ensemble des entiers — valeur **0** pour **False** et valeur **1** pour **True**. À titre d'illustration²⁷ est-ce que **1** est plus petit que **2**? C'est vrai. Est-ce que **1** est plus grand que **5**? c'est faux.

```
>>> 1 < 2
True
>>> 1 > 5
False
```

3.1.2 Manipulation comme calculette

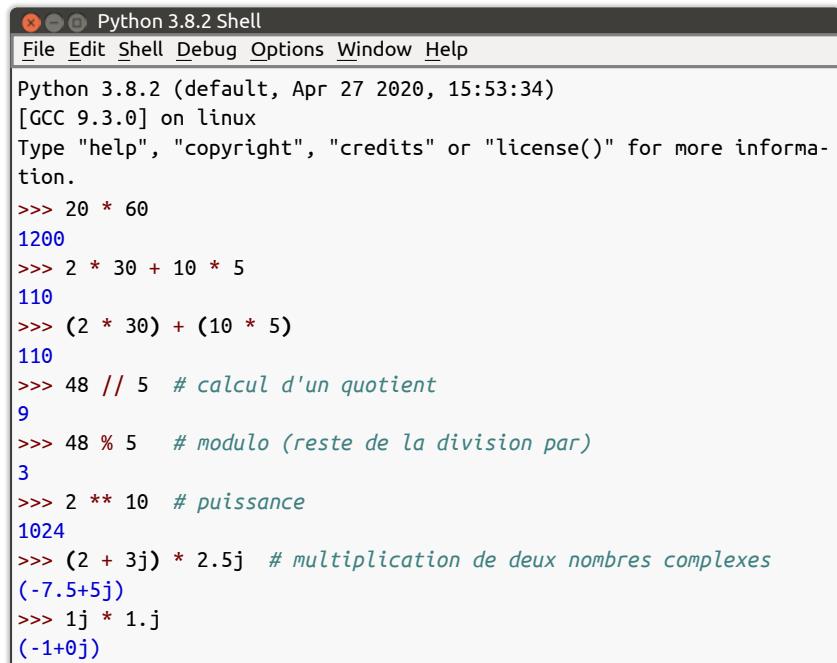
Au démarrage de l'interpréteur Python, on dispose en fait d'une calculette. Les *règles de priorité* entre les différents opérateurs sont habituelles; les produits et les divisions sont évalués en premier, ensuite les sommes et les soustractions.

2

De manière générale, il est recommandé de saisir des expressions correctement parenthésées. De plus, les parenthèses facilitent la lecture d'expressions complexes (cf. infra, la deuxième expression est équivalente de la troisième mais bien plus lisible).

TABLE 8.5 — Opérateurs PYTHON.

Opérateurs en PYTHON	
CODE	OPÉRATION
+	Addition
-	Soustraction
*	Multiplication
/	Division
//	Quotient
%	Modulo
**	Puissance



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.

>>> 20 * 60
1200
>>> 2 * 30 + 10 * 5
110
>>> (2 * 30) + (10 * 5)
110
>>> 48 // 5 # calcul d'un quotient
9
>>> 48 % 5 # modulo (reste de la division par)
3
>>> 2 ** 10 # puissance
1024
>>> (2 + 3j) * 2.5j # multiplication de deux nombres complexes
(-7.5+5j)
>>> 1j * 1.j
(-1+0j)
```

On peut aussi facilement conduire des calculs sur les complexes. Se souvenir que la constante complexe marquée **i** en français se note **j** en PYTHON, ce choix provient du BDfL²⁸ pour des raisons de lisibilité.

Aussi, pour entrer ce nombre complexe **j**, il faut toujours le faire précéder d'un nombre, donc ne pas saisir simplement **j** (qui serait compris comme un nom de variable) mais plutôt **1j** ou encore **1.j**.

i-VARIABLES — Il peut être utile de stocker un résultat qui sera utilisé plus tard ou de définir une valeur constante. Pour cela on utilise tout simplement une affectation.

```
>>> largeur = 5 # variable définie par assignation de valeur
>>> largeur * 20 # utilisation, ici comme un nombre
100
>>> largeur * 10 # après quoi bien sûr la variable reste inchangée
50
```

²⁸. Alias Guido VAN Rossum.

Pour les symboles mathématiques, on peut utiliser la même technique. Pour les valeurs spéciales comme π , on peut utiliser les valeurs prédéfinies par la bibliothèque mathématique de PYTHON. En anticipant un peu sur la notion d'importation, on peut la charger et ainsi imprimer les racines troisièmes de l'unité²⁹ par la formule :

$$r_n = e^{2i\pi \frac{n}{3}}, \text{ pour } n = 0, 1, 2 \quad (8.2)$$

29. Bien entendu, il sera possible de faire ceci plus simplement lorsque les boucles `for` seront vues.

```
>>> pi = 3.14159 # réels : le point remplace la virgule du français
>>> 2 * pi * 10
62.8318
>>> from math import e, pi
>>> n = 0
>>> print("n=", n, "racine =", e**((2.j*pi*n)/3))
n= 0 racine = (1+0j)
>>> n = 1
>>> print("n=", n, "racine =", e**((2.j*pi*n)/3))
n= 1 racine = (-0.49999999999998+0.8660254037844387j)
>>> n = 2
>>> print("n=", n, "racine =", e**((2.j*pi*n)/3))
n= 2 racine = (-0.500000000000004-0.8660254037844384j)
```

ii – TYPES — Le changement par rapport à une calculatrice standard provient du fait que les valeurs soient typées. Illustration pour les trois types de nombres établis jusqu'ici.

```
>>> type(3) # type entier : 'int'
<class 'int'>
>>> type(3.5) # type à virgule flottante : 'float'
<class 'float'>
>>> type(1j) # type complexe : 'complex'
<class 'complex'>
```

iii – CHAÎNES DE CARACTÈRES — On a également rapidement besoin de chaînes de caractères, étudiées par la suite en détail, mais en guise de mise en bouche, on peut écrire ce qui suit.

```
>>> chaine = "Bonjour le monde !"
>>> print(chaine)
Bonjour le monde !
```

iv – CONVERSIONS DE TYPE — Il est parfois nécessaire de convertir une donnée d'un type à un autre. Par exemple on peut demander à l'utilisateur d'entrer son âge au clavier grâce à la fonction `input`.

Pour faire des calculs — disons multiplier l'âge par deux —, en s'y prenant naïvement, le résultat obtenu est surprenant; il s'agit d'une concaténation de chaînes de caractères. En effet, la saisie à la volée s'opère assez logiquement sous le type chaîne de caractères. C'est pourquoi il faut ici d'abord convertir la (valeur de la) variable `reponse` en un entier, avant qu'il puisse ensuite être doublé (s'assurer au préalable de bien avoir entré une donnée qui corresponde à un nombre entier).

```
>>> reponse = input("quel est votre âge ? ")
quel est votre âge ? 35
>>> print(reponse) # chaîne de caractère saisie
35
>>> type(reponse) # variable de type chaîne de caractères
<class 'str'>
>>> 2 * reponse    # multiplier une chaîne donne une concaténation
'3535'
>>> age = int(reponse) # conversion de la chaîne en entier
```

```
>>> type(age)
<class 'int'>
>>> 2 * age # désormais, on peut correctement multiplier par 2
70
>>> # ou le tout en une seule passe :
>>> print("le double de votre age est", 2*int(reponse))
le double de votre age est 70
```

TABLE 8.6 — Fonction de typage PYTHON.

Fonction de typage en PYTHON	
TYPE	FONCTION
Entier	<code>int</code>
Flottant	<code>float</code>
Complexe	<code>complex</code>
Chaîne	<code>str</code>

De manière plus générale, pour convertir un objet en entier, en flottant ou en chaîne de caractères, on peut simplement appeler une fonction *built-in* qui porte le même nom que le type cible (voir table 8.6). Ainsi dans l'exemple précédent, `int(reponse)` représente la conversion de `reponse` en entier.

```
>>> a = 2345 # à l'inverse, si la donnée de départ est un entier
>>> str(2345) # conversion en chaîne de caractères
'2345'
>>> complex(2345) # conversion en complexe
(2345+0j)
```

Ceci se généralise à tous les types qui sont reconnus par PYTHON, afin de convertir un objet `x` en un type `bijule`, on appelle `bijule(x)`.

v – GRANDS NOMBRES — Comme les entiers sont de précision illimitée, on peut améliorer leur lisibilité en insérant des caractères *underscore* « `_` », lesquels sont simplement ignorés à l'exécution.

```
>>> tres_grand_nombre = 23_456_789_012_345
>>> tres_grand_nombre
23456789012345
>>> 123_456.789_012 # ça fonctionne aussi avec les flottants
123456.789012
```

vi – ENTIERS ET BASES — Les calculettes scientifiques permettent habituellement d'entrer les entiers dans d'autres bases que la base 10. En PYTHON, on peut également saisir un entier sous forme binaire, octale (base 8) ou encore hexadécimale (base 16).

Pour d'autres bases, on peut utiliser la fonction de conversion `int` en lui passant un argument supplémentaire.

```
>>> deux_cents = 0b11001000      # binaire
>>> deux_cents = 0o310          # octale
>>> deux_cents = 0xc8           # hexadécimale
>>> print(deux_cents)
200
>>> deux_cents = int('3020', 4) # quaternaire
>>> deux_cents = int('1300', 5) # quinaire
>>> print(deux_cents)
200
```

vii – FONCTIONS MATHÉMATIQUES — Naturellement, PYTHON fournit un ensemble très complet d'opérateurs mathématiques pour les fonctions exponentielles, trigonométriques et autres. Cependant, à cette étape de la discussion, leur étude concrète est reportée à plus tard.

3.1.3 Affectations et opérations (« à la += »)

Il existe en PYTHON toute une famille d'opérateurs dérivés de l'affectation qui permettent de faire en une fois une opération et une affectation. En voici quelques morceaux choisis.

i – INCrémentation et autres — On peut facilement augmenter la valeur d'une variable numérique en employant l'opérateur « `+=` ». Cette forme de directive, qui combine opération sur une variable et

réaffection du résultat à la même variable, est disponible avec tous les opérateurs courants (cf. table 8.5).

```
>>> entier = 10
>>> entier += 2
>>> print('entier', entier)
entier 12
>>> entier = 10 # équivalent aux lignes suivantes
>>> entier = entier + 2
>>> print('entier', entier)
entier 12
>>> entier -= 4 # même syntaxe pour la soustraction
>>> print('après décrément', entier)
après décrément 8
>>> entier *= 2 # même syntaxe pour la multiplication
>>> print('après doublement', entier)
après doublement 16
>>> entier /= 2 # même syntaxe pour la division naturelle
>>> print('mis à moitié', entier)
mis à moitié 8.0
>>> entier = 8
>>> entier /= 3 # même syntaxe pour la division entière
>>> print('mis à moitié', entier)
mis à moitié 2.6666666666666665
>>> entier = 2
>>> print('entier:', entier)
entier: 2
>>> entier **= 10
>>> print('à la puissance dix:', entier)
à la puissance dix: 1024
>>> entier %= 5
>>> print('modulo 5:', entier)
modulo 5: 4
```

ii – TYPES NON NUMÉRIQUES ET OPÉRATEURS ABSCONS – En réalité cette construction est disponible pour tous les types qui supportent l'opérateur employé. Par exemple, les listes (vues prochainement) peuvent être additionnées entre elles.

Beaucoup de types supportent l'opérateur d'addition, qui est sans doute et de loin celui le plus utilisé avec cette construction. Signalons enfin que l'on trouve aussi cette construction avec d'autres opérateurs moins fréquents. Et pour ceux qui connaissent déjà un peu PYTHON, on peut même l'appliquer avec des opérateurs de décalage.

```
>>> liste = [0, 3, 5]
>>> print('liste', liste)
liste [0, 3, 5]
>>> liste += ['a', 'b']
>>> print('après ajout', liste)
après ajout [0, 3, 5, 'a', 'b']
>>> entier <=> 2
>>> print('double décalage gauche:', entier)
double décalage gauche: 16
```

3.2 Compléments et application

3.2.1 Précision des calculs flottants

Comme pour les entiers, les calculs sur les nombres flottants sont réalisés par le processeur. Cependant, contrairement au cas des entiers où les calculs sont toujours exacts, les flottants posent un problème

de précision. Cela n'est pas propre au langage PYTHON, mais dû à la technique de codage des nombres flottants sous forme binaire.

Il faut retenir que lorsqu'on écrit un nombre flottant sous forme décimale, la valeur utilisée en mémoire pour représenter ce nombre, parce que cette valeur est codée en binaire, ne représente *pas toujours exactement* le nombre entré.

Aussi, les différentes erreurs d'arrondi qui se produisent à chaque étape du calcul s'accumulent et produisent un résultat parfois surprenant. De nouveau, ce problème n'est pas spécifique à PYTHON, il existe pour tous les langages et il est bien connu³⁰ des numériciens.

Dans une grande majorité des cas, ces erreurs d'arrondi ne sont pas pénalisantes. Il faut toutefois en être conscient car cela peut expliquer des comportements curieux.

```
>>> 0.2 + 0.4
0.6000000000000001
>>> 0.3 - 0.1 == 0.2 # expression fausse : erreur d'arrondi
False
```

A. SOLUTION ? PENSER EN TERMES DE NOMBRES RATIONNELS

Si le problème se pose bien en termes de nombres rationnels, il est alors tout à fait possible de le résoudre avec exactitude.

Alors qu'il n'est pas possible d'écrire exactement $3/10$ en base 2, ni $1/3$ en base 10, on peut représenter exactement ces nombres dès lors qu'on les considère comme des fractions et qu'on les encode avec deux nombres entiers.

PYTHON fournit en standard le module **fractions** qui permet de résoudre le problème. Voici comment on pourrait l'employer pour vérifier — cette fois avec succès —, que $0.3 - 0.1$ vaut bien 0.2 . Ce code anticipe sur l'utilisation des modules et des classes en PYTHON, ici des objets de type **Fraction** sont créés.

```
>>> from fractions import Fraction # import du module qui définit le symbole Fraction
>>> Fraction(3, 10) - Fraction(1, 10) == Fraction(2, 10) # cette fois, calculs exacts
True
>>> Fraction('0.3') - Fraction('0.1') == Fraction('2/10') # équivalent encore plus lisible
True
```

B. AUTRE SOLUTION ? MODULE « decimal »

Si par ailleurs, pour quelque raison, les nombres rationnels ne sont pas manipulés, du coup la représentation sous forme de fractions ne peut pas convenir.

Indiquons alors l'existence du module standard **decimal** qui offre des fonctionnalités très voisines du type **float**, tout en éliminant la plupart des inconvénients, mais au prix, bien entendu, d'une consommation mémoire supérieure.

```
>>> from decimal import Decimal
>>> Decimal('0.3') - Decimal('0.1') == Decimal('0.2')
True
```

POUR ALLER PLUS LOIN...

Tous les documents qui suivent sont en anglais :

- ▶ tutoriel sur les nombres flottants;
- ▶ documentation sur la classe **Fraction**;
- ▶ documentation sur la classe **Decimal**;

3.2.2 Opérations bit à bit (*bitwise*)

Il s'agit dans cette section d'expliquer des fonctions évoluées³¹ sur les entiers. Outre les opérations déjà commentées, il est également possible de faire des opérations bit à bit sur les nombres entiers.

Le plus simple est de penser à l'écriture du nombre en base 2. Considérons par exemple deux entiers constants et leur transcription en binaire, comme par exemple :

```
>>> x49 = 49
>>> y81 = 81
```

$$\begin{aligned}x49 &= 49 = 32 + 16 + 1 \rightarrow (0, 1, 1, 0, 0, 0, 1) \\x81 &= 81 = 64 + 16 + 1 \rightarrow (1, 0, 1, 0, 0, 0, 1)\end{aligned}\quad (8.3)$$

Pour comprendre comment passer de $32 + 16 + 1$ à $(0, 1, 1, 0, 0, 0, 1)$ et $64 + 16 + 1$ à $(1, 0, 1, 0, 0, 0, 1)$, il suffit d'observer que :

$$\begin{aligned}32 + 16 + 1 &= \mathbf{0} \quad 2^6 + \mathbf{1} \quad 2^5 + \mathbf{1} \quad 2^4 + \mathbf{0} \quad 2^3 + \mathbf{0} \quad 2^2 + \mathbf{0} \quad 2^1 + \mathbf{1} \quad 2^0 \\64 + 16 + 1 &= \mathbf{1} \quad 2^6 + \mathbf{0} \quad 2^5 + \mathbf{1} \quad 2^4 + \mathbf{0} \quad 2^3 + \mathbf{0} \quad 2^2 + \mathbf{0} \quad 2^1 + \mathbf{1} \quad 2^0\end{aligned}\quad (8.4)$$

A. OPÉRATEURS LOGIQUES

L'opération logique « `&` » va faire un *et* logique bit à bit entre les opérandes, ainsi :

```
>>> x49 & y81
17
```

En effet, on s'y retrouve car on a :

$$\begin{aligned}x49 &\rightarrow (0, 1, 1, 0, 0, 0, 1) \\x81 &\rightarrow (1, 0, 1, 0, 0, 0, 1) \\x49 \& y81 &\rightarrow (0, 0, 1, 0, 0, 0, 1) \rightarrow 16 + 1 \rightarrow 17\end{aligned}\quad (8.5)$$

Ici, l'opérateur logique « `|` » fait simplement un *ou* logique :

```
>>> x49 | y81
113
```

Cette fois, parce que :

$$\begin{aligned}x49 &\rightarrow (0, 1, 1, 0, 0, 0, 1) \\x81 &\rightarrow (1, 0, 1, 0, 0, 0, 1) \\x49 | y81 &\rightarrow (1, 1, 1, 0, 0, 0, 1) \rightarrow 64 + 32 + 16 + 1 \rightarrow 113\end{aligned}\quad (8.6)$$

Enfin, on peut également conduire la même opération à base de *ou exclusif* avec l'opérateur « `^` » :

```
>>> x49 ^ y81
96
```

Le *ou exclusif* de deux bits est vrai si et seulement si exactement une des deux entrées est vraie.

$$\begin{aligned}x49 &\rightarrow (0, 1, 1, 0, 0, 0, 1) \\x81 &\rightarrow (1, 0, 1, 0, 0, 0, 1) \\x49 ^ y81 &\rightarrow (1, 1, 0, 0, 0, 0, 0) \rightarrow 64 + 32 \rightarrow 96\end{aligned}\quad (8.7)$$

B. DÉCALAGES

Un décalage à gauche de, par exemple quatre positions, revient à décaler tout le champ de bits de quatre cases à gauche (les quatre nouveaux bits insérés sont toujours des 0). C'est donc équivalent à une multiplication par $2^4 = 16$.

```
>>> x49 << 4
784
```

³¹ Sans souci, les débutants en programmation peuvent sauter cette partie en cas de difficulté.

$$\begin{aligned} x49 &\rightarrow (0, 1, 1, 0, 0, 0, 1) \\ x49 \ll 4 &\rightarrow (0, 1, 1, 0, 0, 1, 0, 0, 0, 0) \rightarrow 512 + 256 + 16 \rightarrow 784 \end{aligned} \quad (8.8)$$

De la même manière, le décalage à droite de n revient à une division par 2^n (plus précisément, le quotient de la division).

```
>>> x49 >> 4
3
```

$$\begin{aligned} x49 &\rightarrow (0, 1, 1, 0, 0, 0, 1) \\ x49 \gg 4 &\rightarrow (0, 0, 0, 0, 0, 1, 1) \rightarrow 2 + 1 \rightarrow 3 \end{aligned} \quad (8.9)$$

C. ASTUCE

³². Pour poursuivre l'exploration, consulter la documentation de PYTHON : <https://docs.python.org/3/library/stdtypes.html#bitwise-operations-on-integer-types>

On peut utiliser³² la fonction `built-in bin` pour calculer la représentation binaire d'un entier. Attention, la valeur de retour est une chaîne de caractères de type `str`. Dans l'autre sens, on peut aussi saisir un entier directement en base 2.

```
>>> bin(x49)
'0b110001'
>>> x49bis = 0b110001
>>> x49bis == x49
True
```

Ici, `x49bis` est bien un entier.

3.2.3 Exercices

En corollaire de la discussion sur la précision des flottants, il faut savoir que le système de codage en mémoire impose aussi une limite. Les réels très petits ou très grands, ne peuvent plus être représentés de cette manière.

C'est notamment très gênant si on implémente un logiciel probabiliste — comme des graphes de MARKOV — où les probabilités d'occurrence de séquences très longues tendent très rapidement vers des valeurs extrêmement petites.



EXERCICE 5 — PLUS PETIT FLOTANT



Le but de cet exercice est d'estimer la valeur du plus petit flottant qui peut être représenté comme un flottant. Pour aider, voici deux valeurs :

```
>>> 10**-320
1e-320
>>> 10**-330
0.0
```

Comme on le constate, 10^{-320} est correctement imprimé, alors que 10^{-330} est, de manière erronée, rapporté comme étant nul.

Remarques :

- ▶ À ce stade du cours, pour estimer le plus petit flottant, procéder simplement par approximations successives;
- ▶ Sans utiliser de boucle, la précision obtenue n'est que fonction de la patience accordée à l'exercice par l'expérimentateur, ne dépasser pas 4 à 5 itérations successives:);
- ▶ Il est par contre pertinent d'utiliser une approche rationnelle pour déterminer l'itération suivante (par opposition à une approche « au petit bonheur »). Pour ceux qui ne connaissent pas, il est recommandé de se documenter sur l'algorithme de dichotomie;
- ▶ L'intérêt du *notebook* par rapport à un document écrit est de pouvoir expérimenter directement dans le fil de l'énoncé (pour créer de nouvelles cellules saisir `Alt + Enter`) ou, sinon, ouvrir un interpréteur comme IDLE.



EXERCICE 6 — PLUS GRAND FLOTTANT



La même limitation s'applique aux grands nombres. Toutefois, cela est un peu moins évident, car comme toujours il faut faire attention aux types.

En utilisant un exposant entier, les choses se passent bien. Dans ce premier cas `PYTHON` calcule le résultat comme un `int`, qui est un type qui n'a pas de limitation de précision (`PYTHON` utilise intelligemment autant de bits que nécessaire pour ce genre de calculs).

En revanche, si on essaie de conduire le même calcul avec un exposant flottant, on obtient une erreur.

On peut d'ailleurs remarquer que le comportement n'est pas vraiment cohérent, car avec les petits nombres PYTHON a silencieusement transformé 10^{-330} en 0, alors que pour les grands nombres, il lève une exception.

Quoi qu'il en soit, la limite pour les grands nombres se situe entre les deux valeurs 10^300 et 10^310 . On demande à nouveau d'estimer comme ci-dessus une valeur approchée du plus grand nombre possible de représenter comme un flottant.

```
>>> 10**300.  
1e+300  
>>> 10**310.  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
OverflowError: (34, 'Numerical result out of range')
```

COMPLÉMENTS AVANCÉS

En fait, on peut accéder aux valeurs minimales et maximales des flottants au moyen du module **sys**, pour « *system* »; en particulier avec la méthode **float_info**.

Sauf que si le maximum observé expérimentalement est voisin de cette valeur, le minimum ne correspond pas bien à celui donné par le module **sys**. L'explication de cette apparence contradiction réside dans l'utilisation de « nombres dénormaux ».

```
>>> import sys  
>>> print(sys.float_info)  
sys.float_info(max=1.7976931348623157e+308, max_exp=1024,  
               max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021,  
               min_10_exp=-307, dig=15, mant_dig=53,  
               epsilon=2.220446049250313e-16, radix=2, rounds=1)  
>>> print("Flottant minimum", sys.float_info.min)  
Flottant minimum 2.2250738585072014e-308  
>>> print("Flottant maximum", sys.float_info.max)  
Flottant maximum 1.7976931348623157e+308
```

4 Que faire de ces ressources ? Autoévaluation

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-MOOC. La fonctionnalité manquante — pas encore implémentée dans l'extension de style L^AT_EX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

Le questionnaire à choix multiple* — QCM — à suivre porte sur le présent chapitre « Notions fondamentales en PYTHON ».

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

QUIZ 25 — VARIABLE, OBJET ET TYPAGE DYNAMIQUE 3 POINTS

QUIZ 25.1 — OBJETS, AFFECTATION, TYPAGE DYNAMIQUE 1 POINT

Quelles affirmations sont vraies ?

Chaque variable possède un type.

Chaque objet possède un type.

Le typage dynamique permet de changer le type des objets.

L'opération d'affectation donne un nom à un objet.

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 25.2 — NOMS DE VARIABLE 1 POINT

Quels sont les noms de variables suivants autorisés* en PYTHON ?

* Pour être très clair, on parle bien de ce qui est permis par le langage, sans tenir compte de ce qui est recommandé ou non.

nom_de_variable

nom-de-variable

nom_de_variable2

nom:de:variable

NomDeVariable

2eme_variable

_nom_de_variable

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER

QUIZ 25.3 — MOTS-CLEFS 1 POINT

Quels sont les noms de variables suivants autorisés en PYTHON ?

var

if

True

true

prénom_garçon

β1

VÉRIFIER

AFFICHER LA RÉPONSE

RÉINITIALISER



QUIZ 26 — TYPE NUMÉRIQUES

3 POINTS

QUIZ 26.1 — CALCUL DE PUISSANCES 1 POINT

Comment calculer la 16^{ième} puissance de 2? 2 ** 16 2 && 16 2 | 16 1 << 16**VÉRIFIER****AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 26.2 — NOMBRES FLOTTANTS 1 POINT

Parmi les expressions suivantes, quelles sont celles qui retournent un flottant dont la valeur est 1.0?

 (3.0 + 3.0) / 5 (3.0 + 3) / 5 (3 + 3) / 5 (3.0 + 3.0) // 5 (3.0 + 3) // 5 (3 + 3) // 5**VÉRIFIER****AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 26.3 — NOMBRES COMPLEXES 1 POINT

Comment saisir au clavier la valeur du complexe 3 + 2i?

 3 + 2i 3 + 2j 3 + 2 * 1j 3 + 2 * 1.j 3 + 2.j 3 + 2 * i 3 + 2 * j**VÉRIFIER****AFFICHER LA RÉPONSE****RÉINITIALISER**

CONCEPTS ÉLÉMENTAIRES

POVOIR ÉCRIRE DE PETITS PROGRAMMES au terme de sa lecture est l'objectif annoncé de ce chapitre. Les concepts essentiels du langage sont ainsi évoqués, dont les plus importants seront repris de manière plus détaillée par la suite.

Aussi, sont abordées les chaînes de caractères, les notions de séquences et de listes, pour ensuite exposer comment les structures de conditionnelles et de boucles d'itération communes à tout langage de programmation sont exprimées en PYTHON. Un aperçu du concept important de compréhension de liste puis une introduction aux modules sont finalement formulés.

1 Chaînes de caractères

Les programmes informatiques d'un langage dit de « haut niveau » comme PYTHON s'écrivent au moyen de chaînes de caractères afin d'être lisible par un être humain. Il faut donc connaître comment ces chaînes de caractère sont codées par l'ordinateur et comment les manipuler pour rédiger des programmes efficents.

1.1 Codage, décodage, Unicode

Avant de se lancer dans la programmation à proprement parler, les notions d'encodage et de décodage sont fondamentales à comprendre. En particulier, le langage PYTHON supporte l'Unicode en natif. Que cela recouvre-t-il et comment procéder ?

1.1.1 Encodage et manipulation

Considérons pour débuter la suite de lettres suivante : « JEFAISUNMOOCSURPYTHON ». Il faut quelques instants au cerveau pour décrypter sa signification ou, autrement dit décoder l'information en « JE FAIS UN MOOC SUR PYTHON ».

En informatique, les opérations ne se réalisent pas avec des lettres mais en manipulant des flux de bits. La question qui peut alors se poser est de savoir comment passer de flux de bits à la notion de lettre.

Le principe est le même que précédemment en appliquant une opération de décodage. Cela va consister à découper le flux de bits en une

SOMMAIRE

1 Chaînes de caractères
1.1 Codage, décodage, Unicode 299
1.1.1 Encodage et manipulation ■
1.1.2 Outils et formatage
1.2 Compléments et application 312
1.2.1 Expressions régulières ■ 1.2.2 Exercices
2 Notions de séquences et de listes
2.1 Définition et manipulation 328
2.1.1 Séquences ■ 2.1.2 Listes
2.2 Compléments et application 333
2.2.1 Séquences, listes et <i>slicing</i> ■
2.2.2 Méthodes spécifiques aux listes ■
2.2.3 Mutabilité et immutabilité des objets ■ 2.2.4 Tris de listes
3 Introduction aux conditionnelles, itérations et fonctions
3.1 Test <i>if-else</i> 341
3.1.1 Utilisation ■ 3.1.2 Blocs d'instruction et syntaxe
3.2 Factorisation de code 342
3.2.1 Boucle <i>for</i> ■ 3.2.2 Fonctions
3.3 Compléments et application 344
3.3.1 Bonnes pratiques ■ 3.3.2 Fonctions et valeur de retour ■ 3.3.3 Exercices
4 Introduction aux compréhensions de listes et aux modules
4.1 Compréhensions de liste 359
4.1.1 Syntaxe et utilisation ■ 4.1.2 Exercices
4.2 Modules 362
4.2.1 Attributs ■ 4.2.2 Bibliothèque standard et autres
5 Que faire de ces ressources ? Quiz

1. Le codage ASCII est la première transcription binaire des caractères alphanumériques. Seul l'alphabet latin sans lettre accentuée est codé pour s'adapter à l'anglais (voir chapitre 4 § 1.2.2).

suite de blocs d'une certaine taille qui chacun ont un sens selon une convention donnée au préalable.

Avec le codage ASCII — *American Standard Code for Information Interchange*¹ —, les différents blocs sont d'une taille de 7 bits. Pour s'en faire une idée, prenons un exemple de flux de bits (cf. figure 9.1).

11000011101100110100111000111100101	Flux de bits
1100001 1101100 1101001 1100011 1100101	Blocs de 7 bits
97 108 105 99 101	Conversion décimale
a l i c e	Jeu de caractères ASCII
a l i c e	Police Fira Sans Bold Italic

FIGURE 9.1 — Codage ASCII (la conversion décimale n'existe pas dans un ordinateur).

Ainsi, lors de la lecture d'un contenu textuel sur Internet ou provenant du disque dur, il s'agit de flux de bits qui sont décodés. De manière inverse, lorsque des données textuelles sont téléchargées sur Internet ou enregistrées sur disque dur, elles sont codées en flux de bits selon une convention déterminée.

Toutefois, on peut remarquer que le codage ASCII n'autorise que 128 caractères différents ($2^7 = 128$). C'est largement insuffisant, ne serait-ce que pour les langues d'Europe de l'Ouest comme le français ou l'allemand. C'est pourquoi pendant longtemps, des codages dits « ASCII étendus » sur 8 bits ($2^8 = 256$) ont été utilisés. Cela reste insuffisant pour pouvoir universellement coder tous les caractères existants, notamment pour les langues orientales et asiatiques.

Pendant longtemps, une multitude de tables de jeux de caractères a coexisté, provoquant des incompatibilités de lecture et de diffusion des données textuelles ; les ordinateurs n'étant pas configurés avec tous les jeux de caractères.

Pour pallier ce problème, le projet Unicode, débuté en 1991, a pour ambition de coder tous les caractères existants. À ce jour, cette opération est réalisée pour plus de 120 000 d'entre eux. Unicode utilise différents types de codage : UTF-8, UTF-16 et UTF-32. Sans rentrer dans les détails, il faut savoir qu'ils résultent d'un compromis entre compacté du codage et rapidité de décodage. En pratique, le codage UTF-8 est le plus usité qui possède la propriété d'être entièrement compatible avec l'ASCII standard.

3

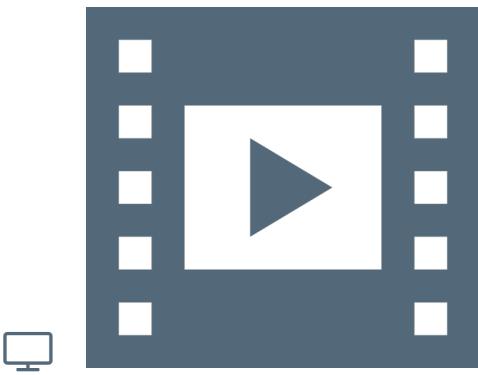
PYTHON a l'avantage d'être lui aussi compatible avec Unicode, au prix d'un contrôle strict de son utilisation. La règle est simple : sans mention explicite contraire provenant d'anciens documents toujours actifs, il faut impérativement indiquer au sein des programmes l'emploi du codage UTF-8.

A. CARACTÈRES ACCENTUÉS

La question des caractères accentués dépasse très largement le cadre de PYTHON et s'avère parfois relativement scabreuse. Qu'est-il possible de faire et quelles sont les prérogatives ?

Avec Unicode, le modèle qui établit la correspondance entre un caractère et un octet est cassé. Aussi en PYTHON 3, lorsqu'il s'agit de manipuler des données provenant de diverses sources de données, les alternatives sont les suivantes :

- le type **byte** est approprié pour charger en mémoire les données binaires brutes, sous forme d'octets ;



VIDÉO 9.1 — Codage et décodage des caractères alphanumériques.

- le type `str` est approprié pour représenter une chaîne de caractères qui, à nouveau, *ne sont pas forcément des octets*;

Le passage de l'un à l'autre de ces types se réalise par des opérations d'encodage et décodage. Pour toutes les opérations de codage et décodage, il est nécessaire de connaître l'encodage utilisé.

On peut appeler les méthodes `encode` et `decode` sans préciser l'encodage (dans ce cas PYTHON choisit l'encodage par défaut du système hôte). Cela dit, il est de loin préférable d'être explicite et de sélectionner son encodage. En cas de doute, il est recommandé de spécifier explicitement `utf-8` — au détriment d'encodages anciens comme `cp1252` (Windows) et `iso8859-*` —, que de laisser le choix au système.

PYTHON 3 supporte l'Unicode par défaut. On peut donc désormais utiliser sans aucun risque des accents ou des cédilles dans les chaînes de caractères. Il faut cependant être vigilant à deux choses :

1. les ordinateurs n'ont pas forcément les polices de caractères nécessaires pour afficher tous les caractères existants;
2. bien que les caractères Unicode soit possibles pour les noms de variables, il est recommandé dans la mesure du possible d'écrire les codes en anglais, comme c'est le cas de la quasi-totalité du code des bibliothèques. Ceci est particulièrement important pour les noms de lignes et de colonnes dans un jeu de données, afin de faciliter les transferts entre logiciels; la majorité des logiciels n'acceptant pas les accents et cédilles dans les noms de variables.

Ainsi, il faut bien distinguer les chaînes de caractères qui doivent par nature être adaptées au langage des utilisateurs du programme, et le code source qui, pour sa part, est destiné aux programmeurs et doit donc éviter d'employer autre chose que de l'anglais.

B. OÙ ET COMMENT UTILISER DES ACCENTS?

i- **NOM DE VARIABLE ET IDENTIFICATEUR** — S'il n'était pas possible en PYTHON 2 d'utiliser un caractère accentué dans un nom de variable — ou pour un identificateur au sens large —, c'est à présent toléré avec PYTHON 3.

In[1] `nb_élèves = 12 # pas recommandé, mais autorisé par le langage`

On peut même utiliser des symboles, comme par exemple

In[2] `from math import cos, pi as Π
θ = Π / 4
cos(θ)`

Out[2] `0.7071067811865476`

Il est fortement recommandé de ne pas employer cette possibilité sans être extrêmement familier avec les caractères Unicode.

Enfin, pour être exhaustif, il faut savoir que seule une partie des caractères Unicode est autorisée dans ce cadre,. C'est heureux parce que certains caractères, comme par exemple l'espace non-sécable, pourraient, s'ils étaient autorisés, être la cause de milliers d'heures de debugging à frustration garantie!

DOCUMENTATION OFFICIELLE
https://docs.python.org/3/reference/lexical_analysis.html#identifiers

ii- **CHAÎNES DE CARACTÈRES** — On peut naturellement mettre des accents dans les chaînes de caractères. Cela dit, les données manipulées par un programme proviennent pour l'essentiel de sources externes, comme une base de données ou un formulaire Web et non, le plus souvent, directement du code source. Les chaînes de caractères présentes dans du vrai code sont bien souvent limitées à des messages de *logging* et en anglais, donc sans accent.

Lorsqu'un programme doit interagir avec les utilisateurs et « parler » leur langue, c'est une bonne pratique de créer un fichier spécifique, que

l'on appelle *fichier de ressources*, qui contient toutes les chaînes de caractères spécifiques à une langue. Ainsi, la traduction du programme consistera à simplement modifier ce fichier de ressources.

- iii – COMMENTAIRES** — Bien entendu, on peut mettre dans les commentaires n'importe quel caractère Unicode et notamment des caractères accentués si on choisit malgré tout d'écrire le code en français.

C. ENCODAGE

La mémoire — ou le disque — d'un ordinateur ne permet que de stocker des représentations binaires. Il n'y a donc pas de façon « naturelle » de représenter un caractère comme 'A', un guillemet ou un point-virgule.

On utilise pour cela un encodage, par exemple le code US-ASCII stipule, pour faire simple, qu'un 'A' est représenté par l'octet 65 qui s'écrit en binaire **01000001**. Il se trouve qu'il existe plusieurs encodages, bien sûr incompatibles, selon les systèmes et les langues...

Le point important est que pour pouvoir ouvrir un fichier « proprement », il faut non seulement disposer du *contenu* du fichier, mais aussi connaître l'*encodage* qui a été employé pour l'écrire.

- i – PRÉCAUTIONS D'ENCODAGE DES CODES SOURCES** — L'encodage ne concerne pas simplement les objets chaîne de caractères, mais également l'ensemble du code source. PYTHON 3 considère que le code source utilise par défaut l'encodage UTF-8. Il est conseillé de conserver cet encodage ; c'est celui qui offre le plus de flexibilité.

Il est malgré tout possible de changer l'encodage du code source en faisant figurer *en première ou deuxième ligne* des fichiers, une déclaration comme :

```
In[] # -*- coding: <nom_de_l'_encodage> -*
```

ou plus simplement, comme ceci :

```
In[] # coding: <nom_de_l'_encodage>
```

Notons que la première option est également interprétée par l'éditeur de texte **Emacs** pour utiliser le même encodage. En dehors de l'utilisation d'**Emacs**, la deuxième option, plus simple et donc plus « pythonique », est à préférer.

Le nom UTF-8 fait référence à Unicode — ou pour être précis, à l'encodage le plus répandu parmi ceux qui sont définis dans la norme Unicode (cf. infra). Sur certains systèmes plus anciens on peut être amenés à utiliser un autre encodage. Pour déterminer la valeur à employer, on peut saisir dans un interpréteur interactif :

```
>>> # À exécuter sur la machine hôte
>>> import sys # (ici Linux Ubuntu MATE 20.04 LTS)
>>> print(sys.getdefaultencoding())
utf-8
```

Par exemple avec d'anciennes versions de **WINDOWS** (en principe de plus en plus rares) on peut être amenés à écrire :

```
In[] # coding: cp1252
```

La syntaxe de la ligne **coding** est précisée dans la documentation officielle et dans le [PEP 263](#).

- ii – GRAND MALENTENDU** — Si un fichier contenant du français est encodé avec, disons, ISO/IEC 8859-15 — à savoir Latin-9; on constate que dans la table, un caractère « € » se matérialise par un octet '0xA4', soit 164. Imaginons maintenant que l'on essaye d'ouvrir ce même fichier

depuis un vieil ordinateur Windows configuré pour le français. Si on ne lui donne aucune indication sur l'encodage, le programme qui va lire ce fichier va utiliser l'encodage par défaut du système, c'est-à-dire CP1252. L'octet ‘`0xA4`’ correspond au caractère « `¤` » à la place de « `€` ».

Contrairement à ce qu'on pourrait espérer, ce type de problème ne se règle pas en ajoutant une balise `# coding: <nom_de_l'_encodage>`, qui n'agit que sur l'encodage utilisé pour lire le fichier source en question (celui qui contient la balise). Pour pallier correctement ce type de problème, il faut préciser explicitement l'encodage à utiliser pour décoder le fichier, donc avoir un moyen fiable de déterminer cet encodage; ce qui n'est pas toujours aisément d'ailleurs. Cela signifie que pour être totalement propre, il faut pouvoir préciser explicitement le paramètre `encoding` à l'appel de toutes les méthodes qui sont susceptibles d'en avoir besoin.

iii – POURQUOI CELA FONCTIONNE EN LOCAL ? — Lorsque le producteur — programme qui écrit le fichier — et le consommateur — programme qui le lit — tournent sur le même ordinateur, tout fonctionne bien parce que les deux programmes se rapportent à l'encodage par défaut.

Il y a toutefois une limite, si on utilise un Linux configuré² de manière minimale, il se peut que l'encodage par défaut soit l'US-ASCII qui ne « connaît » pas un simple « `é` », ni *a fortiori* « `€` ». Pour écrire du français, il faut donc au minimum que l'encodage par défaut de l'ordinateur contienne les caractères accentués, comme par exemple : ISO 8859-1 (Latin-1), ISO 8859-15 (Latin-9), CP1252 et UTF-8, ce dernier étant à clairement préférer lorsque c'est possible.

². Par exemple, en version serveur ou embarqué.

iv – HISTOIRE DES ENCODAGES — Jusque dans les années 1980, les ordinateurs ne « parlaient » pour l'essentiel que l'anglais. La première vague de standardisation avait créé l'encodage dit ASCII, ou encore US-ASCII (voir aussi chapitre 4 « Codage binaire » § 1.2.2).

Le code US-ASCII s'étend sur 128 valeurs, soit 7 bits, mais est le plus souvent implémenté sur un octet pour préserver l'alignement, le dernier bit pouvant être utilisé par exemple pour ajouter un code correcteur d'erreur — ce qui à l'époque des modems n'était pas superflu. Bref, la pratique courante était alors de manipuler une chaîne de caractères comme un tableau d'octets.

Dans les années 1990, pour satisfaire les besoins des pays européens, ont été définis plusieurs encodages alternatifs, connus sous le nom de ISO/IEC 8859-* , nommés aussi Latin-* . Idéalement, on aurait pu et certainement dû définir un seul encodage pour représenter tous les nouveaux caractères, mais entre toutes les langues européennes, le nombre de caractères à ajouter était substantiel; cet encodage unifié aurait largement dépassé 256 caractères différents, il n'aurait donc pas été possible de tout faire tenir sur un octet.

On a préféré préserver la « bonne propriété » du modèle un caractère correspond à un octet, ceci afin de conserver le code existant qui aurait sinon dû être retouché ou réécrit. Dès lors, il n'y avait pas d'autre choix que de définir plusieurs encodages distincts. Par exemple, pour le français on a utilisé ISO/IEC 8859-1 (Latin-1), pour le russe ISO/IEC 5589-5 (Latin/Cyrillic). À ce stade, le ver était dans le fruit. Depuis cette époque, pour ouvrir un fichier il faut connaître son encodage.

Lorsque l'on a ensuite cherché à manipuler aussi les langues asiatiques, il a de toute façon fallu définir de nouveaux encodages beaucoup plus larges. C'est ce qui a été fait par le standard Unicode qui définit trois nouveaux encodages :

- UTF-8 : un encodage à taille variable, à base d'octets, qui maximise la compatibilité avec US-ASCII ;
- UTF-16 : un encodage à taille variable, à base de mots de 16 bits ;
- UTF-32 : un encodage à taille fixe, à base de mots de 32 bits.

Ces standards couvrent le même jeu de caractères (113 021 tout de même dans la dernière version). Parmi ceux-ci le plus utilisé est certainement UTF-8. Un texte ne contenant que des caractères du code US-ASCII initial peut être lu avec l'encodage UTF-8.

Pour être enfin tout à fait exhaustif, si on sait qu'un fichier est au format Unicode, on peut déterminer quel est l'encodage qu'il utilise en se fondant sur les quatre premiers octets du document. Ainsi dans ce cas particulier (lorsqu'on est sûr qu'un document utilise un des trois encodages Unicode) il n'est plus nécessaire de connaître son encodage de manière « externe ».

1.1.2 Outils et formatage

Cette section s'attache au maniement des chaînes de caractères et à leur encodage. On peut tout de suite noter que PYTHON 3 gère désormais nativement l'Unicode, ce qui n'était pas le cas auparavant; cela offre un très bon argument pour passer à cette nouvelle version.

A. OBJETS CHAÎNES DE CARACTÈRES

Créer une chaîne de caractère est très simple, il suffit de l'entourer d'apostrophes ou bien de guillemets si la chaîne contient elle-même des apostrophes. Pour la manipuler, on l'affecte à une variable.

```
>>> 'spam'
'spam'
>>> "spam"
'spam'
>>> s = 'spam'
```

Les chaînes de caractères sont des *objets immuables*, c'est-à-dire qu'une fois créés ils ne sont plus modifiables. Les chaînes de caractères comportent de nombreuses méthodes. Ces objets étant immuables, chaque fonction va retourner un nouvel objet chaîne de caractères.

Comment connaître toutes ces méthodes applicables aux chaînes de caractères ? PYTHON est un langage auto-documenté, c'est-à-dire que l'aide peut être interrogée depuis l'interpréteur, soit par l'ajout d'un point d'interrogation à la commande recherchée (depuis IPYTHON ou un notebook) comme par exemple `str?` pour les objets chaînes de caractères, soit en saisissant « `help` » devant la commande (depuis IDLE).

Pour par exemple lister toutes les méthodes associées aux chaînes de caractères, il faut utiliser la fonction *built-in* « `dir` » : `dir(str)`. On constate que certaines méthodes débutent et se finissent par des doubles underscores. Elles seront détaillée ultérieure car il s'agit de propriétés avancées liées aux méthodes spéciales.

Le nom des méthodes est souvent explicite. Prenons pour exemple les méthodes « `title` » et « `replace` ». La première met en capitale la première lettre de chaque mot de la chaîne de caractères, la seconde remplace un mot donné de la chaîne de caractères.

```
>>> 'un mooc sur python'.title()
'Un Mooc Sur Python'
>>> s = "le chocolat c'est bon"
>>> s.replace('chocolat', 'spam')
"le spam c'est bon"
```



Erratum : À 5:05, les `f-string` sont introduites depuis PYTHON 3.6 et non 3.5.

VIDÉO 9.2 – Chaînes de caractères I.

BLOC-NOTE VIDÉO 9.2 (copier-coller)

```
'spam'
"spam"
s = 'spam'
str?
dir(str)
'un mooc sur python'.title()
s = "le chocolat c'est bon"
s.replace('chocolat', 'spam')
s
'123'.isdecimal()
n = 'sonia'
age = 30
 "{} {}".format(n, age)
 "{} a {} ans".format(n, age)
 f"{{n}} a {{age}} ans"
```

On note au passage que si l'on veut introduire des apostrophes dans une chaîne de caractères, il suffit de la mettre entre guillemets; cela évite l'emploi du caractère d'échappement *backslash* « \ ».

D'autres méthodes permettent de conduire des tests et d'établir des comparaisons. Supposons par exemple avoir la chaîne de caractère de l'entier '123'. Avant de la convertir, on peut s'assurer que cela représente bien un nombre décimal. Cela se réalise avec la méthode `isdecimal` qui retourne ici `True`. Chacun est invité à consulter l'intégralité des méthodes sur les chaînes de caractères, car elles sont très puissantes et à même de très souvent répondre aux besoins.

```
>>> '123'.isdecimal()
True
```

Le formatage d'une chaîne de caractères se réalise en PYTHON avec l'instruction « `format()` » ou, de manière expressive et à utiliser systématiquement, à l'aide des *fstrings* depuis PYTHON 3.5 (cf. infra).

```
>>> n = 'sonia'
>>> age = 30
>>> "{} {}".format(n, age)
'sonia 30'
>>> "{} a {} ans".format(n, age)
'sonia a 30 ans'
>>> f"{n} a {age} ans"
'sonia a 30 ans'
```

B. SUPPORT UNICODE

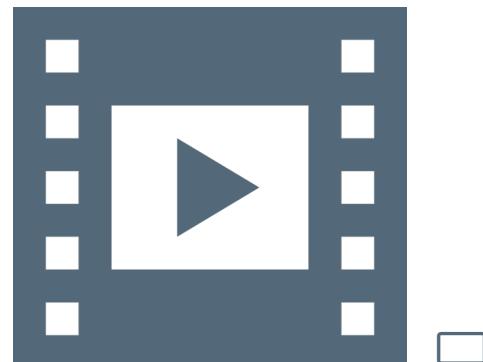
Il est permis de saisir directement les caractères Unicode dans un interpréteur, mais il aussi possible d'entrer le code du caractère comme par exemple « \u03a6 » pour obtenir la lettre grecque Phi « Φ ».

```
>>> "noël, été"
'noël, été'
>>> "\u03a6"
'Φ'
>>> "\u0556"
'Փ'
```

Par ailleurs, en saisissant « \u0556 », on obtient aucun affichage alors que le caractère attendu correspond à la trente-huitième lettre de l'alphabet arménien, Feh. Cela provient simplement de la police de caractères de l'interpréteur qui ne dispose pas de ce symbole. Il suffit d'aller changer la fonte dans les options de l'interpréteur. Par exemple, on a bien la lettre Feh « Ֆ » avec la police DejaVu Sans.

```
>>> s = "un noël en été"
>>> print(s)
un noël en été
>>> s.encode('utf8')
b'un no\xc3\xabl en \xc3\x9at\xc3\x9a9'
>>> en = s.encode('utf8')
>>> en
b'un no\xc3\xabl en \xc3\x9at\xc3\x9a9'
>>> en.decode('utf8')
'un noël en été'
>>> s.encode('latin1')
b'un no\xeb1 en \xe9t\xe9'
```

Les opérations de codage et décodages de chaînes de caractères sont simples en PYTHON. Dès qu'on utilise des caractères, on fait appel au type « `str` ». Dès que l'on manipule des flux de bits, on a recours au



VIDÉO 9.3 – Chaînes de caractères II.

BLOC-NOTE VIDÉO 9.3 (copier-coller)

```
"noël, été"
"\u03a6"
"\u0556"
s = "un noël en été"
print(s)
s.encode('utf8')
en = s.encode('utf8')
en
en.decode('utf8')
s.encode('latin1')
```

type « **bytes** ». PYTHON permet facilement de passer de l'un à l'autre à l'aide des méthodes **encode** et **decode** (voir supra encart IDLE).

Lorsqu'on encode une chaîne de caractères, PYTHON retourne une ligne débutant par « **b** » pour bytes, puis une suite de glyphes qui correspondent à des octets. En effet, PYTHON offre un affichage sous forme de lettre ou forme chiffre lorsque l'octet correspond au code d'un caractère ASCII. Il ne s'agit que d'une facilité et le codage est bien réalisé en octet. Inversement, on revient à une chaîne de caractère par la fonction **decode**. On peut noter que, bien qu'il soit très fortement indiqué d'utiliser UTF-8, d'autres encodages, tel que Latin-1 sont possibles.

C. MANIPULATION DES CHAÎNES DE CARACTÈRES

Même après des années de pratique, il est toujours difficile de se souvenir de toutes les méthodes sur les chaînes de caractères. Aussi, le recours à la documentation³ embarquée est efficace : **help(str)**.

Il est donc illusoire de les citer toutes et on se restreint ici aux plus employées. De plus, elles sont abordées de manière simple et, bien souvent, il est possible de passer en argument des options permettant de ne travailler que sur une sous-chaîne ou sur la première ou la dernière occurrence d'une sous-chaîne. Renvoi est fait à la documentation pour obtenir toutes les précisions voulues.

i – DÉCOUPAGE-ASSEMBLAGE : **split** ET **join** — Les méthodes **split** et **join** permettent respectivement de découper une chaîne selon un séparateur pour obtenir une liste et, à l'inverse, de reconstruire une chaîne à partir d'une liste.

```
>>> 'abc=:def=:ghi=:jkl'.split(':=')
['abc', 'def', 'ghi', 'jkl']
>>> '=:='.join(['abc', 'def', 'ghi', 'jkl'])
'abc=:def=:ghi=:jkl'
```

Attention toutefois si le séparateur est un séparateur, la liste résultat contient alors une dernière chaîne vide. En pratique, on utilise la méthode **strip** avant la méthode **split** pour éviter ce problème.

```
>>> 'abc;def;ghi;jkl;'.split(';')
['abc', 'def', 'ghi', 'jkl', '']
>>> ';' .join(['abc', 'def', 'ghi', 'jkl', ''])
'abc;def;ghi;jkl;'
```

ii – REMPLACEMENT : **replace** — La méthode **replace** est très pratique pour remplacer une sous-chaîne par une autre, avec une limite éventuelle sur le nombre de substitutions à effectuer.

```
>>> "abcdefabedefabedef".replace("abc", "zoo")
'zoodefzoodefzoodef'
>>> "abcdefabedefabedef".replace("abc", "zoo", 2)
'zoodefzoodefabcde'
```

Plusieurs appels à **replace** peuvent être chaînés comme suit.

```
>>> "le [x] qui [y]".replace("[x]", "chat").replace("[y]", "miaule")
'le chat qui miaule'
```

iii – NETTOYAGE : **strip** — On pourrait par exemple utiliser **replace** pour enlever les espaces dans une chaîne, ce qui peut être utile pour « nettoyer » la chaîne de caractères.

```
>>> " abc:def:ghi ".replace(" ", "")
'abc:def:ghi'
```

Toutefois, bien souvent, on préfère faire appel à la méthode `strip` qui ne s'occupe que du début et de la fin de la chaîne, tout en gérant également les tabulations et autres retour à la ligne.

```
>>> " \tune chaîne avec des trucs qui dépassent \n".strip()
'une chaîne avec des trucs qui dépassent'
```

On peut appliquer la méthode `strip` avant `split` pour éviter le problème du dernier élément vide (cf. supra).

```
>>> 'abc;def;ghi;jkl;'.strip(';').split(';')
['abc', 'def', 'ghi', 'jkl']
```

iv – RECHERCHER UNE SOUS-CHAÎNE — Plusieurs outils permettent de chercher une sous-chaîne. Il existe `find` qui renvoie le plus petit index où on trouve la sous-chaîne.

```
>>> "abcdefcdefghefghijk".find("def") # Indice première occurrence
3
>>> "abcdefcdefghefghijk".find("zoo") # ou -1 si chaîne absente
-1
```

La méthode `rfind` fonctionne comme `find` mais en partant de la fin de la chaîne. À noter que le résultat correspond toujours au début de la chaîne — en PYTHON les indices commencent à zéro donc la notation `ma_chaine[n]` permet d'accéder à la position `n+1` dans la chaîne.

```
>>> "abcdefcdefghefghijk".rfind("fgh") # En partant de la fin
13
>>> "abcdefcdefghefghijk"[13]
'f'
```

La méthode `index` se comporte comme `find`, mais en cas d'absence elle lève une exception⁴ plutôt que de renvoyer `-1`.

^{4.} Ce concept est abordé plus tard.

```
>>> "abcdefcdefghefghijk".index("def")
3
```

```
>>> try:
...     "abcdefcdefghefghijk".index("zoo")
... except Exception as e:
...     print("OOPS", type(e), e)
...
OOPS <class 'ValueError'> substring not found
```

Mais le plus simple pour chercher si une sous-chaîne est dans une autre chaîne est d'utiliser l'instruction⁵ `in`.

^{5.} Cette instruction est présentée avec les séquences.

```
>>> "def" in "abcdefcdefghefghijk"
True
```

La méthode `count` compte le nombre d'occurrences d'une sous-chaîne. On peut également mentionner les méthodes de commodité `startswith` et `endswith`.

```
>>> "abcdefcdefghefghijk".count("ef")
3
>>> "abcdefcdefghefghijk".startswith("abcd")
True
>>> "abcdefcdefghefghijk".endswith("ghijk")
True
```

On note ici la supériorité en terme d'expressivité des méthodes « pythoniques » `startswith` et `endswith` en remarquant que :

`chaîne.startswith(sous_chaine) == chaîne.find(sous_chaine) == 0`

```
chaine.endswith(sous_chaine)
chaine.rfind(sous_chaine) == (len(chaine) - len(sous_chaine))
```

- ✓ – CHANGEMENT DE CASSE — Pour conclure sur la présentation des méthodes applicables aux chaînes de caractères, sont exposées celles concernant des changements de casse qui parlent d'elles-mêmes.

```
>>> "monty PYTHON".upper()
'MONTY PYTHON'
>>> "monty PYTHON".lower()
'monty python'
>>> "monty PYTHON".swapcase()
'MONTY python'
>>> "monty PYTHON".capitalize()
'Monty python'
>>> "monty PYTHON".title()
'Monty Python'
```

D. FORMATAGE DES CHAÎNES DE CARACTÈRES

On désigne par formatage les outils qui permettent d'obtenir une présentation fine des résultats, que ce soit pour améliorer la lisibilité lorsqu'on s'adresse à des humains ou pour respecter la syntaxe d'une fonctionnalité à laquelle on peut vouloir passer les données pour un traitement ultérieur.

- ✓ – FONCTION `print` — Jusqu'à présent la fonction `print` a presque toujours été appelé pour afficher les résultats. Comme on l'a vu, celle-ci réalise un formatage sommaire : elle insère un espace entre les valeurs qui lui sont passées.

```
>>> print(1, 'a', 12 + 4j)
1 a (12+4j)
```

La seule subtilité notable concernant `print` est que, par défaut, elle ajoute un saut de ligne à la fin. Pour éviter ce comportement, on peut passer à la fonction un argument `end`, inséré au lieu du saut de ligne.

```
>>> print("une", "seule", "ligne") # Première ligne
une seule ligne
>>> def affiche(): # Seconde ligne en deux appels
...     print("une", "autre", end=' ')
...     print("ligne")
...
>>> affiche()
une autre ligne
```

Il faut aussi remarquer que `print` est capable d'imprimer n'importe quel objet. Cela a déjà été le cas avec des listes et des tuples, mais cela peut se réaliser avec des modules.

```
>>> import math
>>> print('le module math est', math) # Impression d'un module
le module math est <module 'math' (built-in)>
```

En anticipant un peu sur la suite, voici comment `print` présente les instances⁶ de classe.

```
>>> class Personne: # Définition de la classe Personne
...     pass
...
>>> personne = Personne() # Création d'une instance
>>> print(personne) # Affichage d'une instance de classe
<__console__.Personne object at 0x7f9307bdcc40>
```

6. Point d'inquiétude, ces notions sont appréhendées dans un chapitre ultérieur.

On atteint assez vite les limites de la fonction `print`:

- d'une part, on peut vouloir formater une chaîne de caractères sans nécessairement l'imprimer ou bien en tout au moins pas immédiatement;
- d'autre part, les espaces ajoutées peuvent être plus néfastes que vraiment profitables;
- enfin, on peut aussi avoir besoin de préciser un nombre de chiffres significatifs ou de choisir comment présenter une date.

C'est pourquoi il est plus courant de formater les chaînes — c'est-à-dire de calculer des chaînes en mémoire —, sans nécessairement les imprimer de suite, d'où la présentation des **f-strings**.

ii – CONCEPT DE F-STRING — Depuis la version 3.6 de PYTHON, on peut utiliser les *f-strings*. C'est le mécanisme de formatage⁷ le plus simple et le plus agréable à employer.

```
>>> prenom, nom, age = 'Jean', 'Dupont', 35 # Quelques variables
>>> f'{prenom} {nom} a {age} ans' # Un premier f-string
'Jean Dupont a 35 ans'
```

⁷. Bien que désormais obsolètes, il est recommandé de lire les sections consacrées à la méthode `format` et à l'opérateur `%` (cf. infra), car encore massivement présents dans les codes existants.

On remarque d'abord que la chaîne commence par « `f"` », c'est bien sûr pour cela qu'on l'appelle un *f-string*. On peut bien entendu ajouter le « `f` » devant toutes les formes du type *string*, qu'ils commencent par « ' », « " », « ''' » ou encore « """ ».

Ensuite, on note que les zones délimitées entre accolades { } sont remplacées par les valeurs assignées aux variables. La logique d'un *f-string*, c'est tout simplement de considérer l'intérieur d'une expression entre accolades comme du code PYTHON, de l'évaluer et d'en utiliser le résultat. Cela signifie qu'on peut conduire des calculs dans les expressions entre accolades et faire appel à des fonctions.

```
>>> f"Dans 10 ans {prenom} aura {age + 10} ans" # Calcul
'Dans 10 ans Jean aura 45 ans'
>>> notes = [12, 15, 19]
>>> f"Weus avons pour l'instant {len(notes)} notes" # Appel
"Nous avons pour l'instant 3 notes"
```

Les *f-strings* fournissent ainsi une méthode très simple et expressive pour formater des données dans des chaînes de caractère. Pour être bien clair : un *f-string* ne réalise pas d'*impression*, il faut donc le passer à `print` si l'impression est souhaitée. Reste à étudier comment chaque expression entre accolades est formatée, par exemple comment choisir le nombre de chiffres significatifs sur un flottant.

iii – FORMATAGE DES EXPRESSIONS AU SEIN DES F-STRINGS — Il arrive qu'on ait besoin de spécifier plus finement la façon dont une valeur doit être affichée ; cela se fait en précisant un format à l'intérieur des expressions entre accolades.

```
>>> from math import pi
>>> f"2xPi avec seulement 2 chiffres après la virgule {2*pi:.2f}"
'2xPi avec seulement 2 chiffres après la virgule 6.28'
```

Dans les accolades de l'exemple ci-dessus, on peut mettre avant les deux points n'importe quelle expression — ne comportant bien entendu pas de deux points —, opération arithmétique, appel de fonction, etc. Autrement dit, tout ce qui constitue l'expression à évaluer. Après les deux points, la place est réservée pour préciser un format d'affichage — ici, un nombre flottant avec deux chiffres significatifs après la virgule.

Pour forcer un petit entier à s'afficher sur quatre caractères, avec des zéro ajoutés au début si nécessaire, on peut saisir ce qui suit.

```
>>> x = 15
>>> f"{x:04d}"
'0015'
```

On fait appel ici au format **d** (toutes ces lettres **d**, **f**, **g** viennent des formats ancestraux de la **libc** des systèmes UNIX comme **printf**). Avec **04d** on précise qu'on veut une sortie sur 4 caractères et qu'il faut remplir à gauche si nécessaire avec des 0.

Dans certains cas, on a besoin d'afficher des données en colonnes de largeur fixe, on utilise pour cela les formats « < », « ^ » et « > » pour afficher à gauche, au centre ou à droite d'une zone de largeur fixe.

```
>>> comptes = [ # Données à afficher
... ('Apollin', 'Dupont', 127),
... ('Myrtille', 'Lamartine', 25432),
... ('Prune', 'Soc', 827465),
...
>>> for prenom, nom, soldé in comptes:
...     print(f"{prenom:<10} -- {nom:^12} -- {soldé:>8} €")
...
Apollin    -- Dupont    --      127 €
Myrtille   -- Lamartine --  25432 €
Prune     -- Soc       --  827465 €
```

iv – MÉTHODE `format` — Avant l'introduction des *f-strings*, la technique recommandée pour faire du formatage était d'avoir recours à la méthode **format** qui est définie sur les objets **str**.

```
>>> prenom, nom, age = 'Jean', 'Dupont', 35
>>> "{} {} a {} ans".format(prenom, nom, age)
'Jean Dupont a 35 ans'
```

Dans cet exemple le plus simple, les données sont affichées en lieu et place des accolades, dans l'ordre où elles sont fournies. Cela convient bien lorsqu'on a peu de données. Si par la suite on veut changer l'ordre par exemple des **nom** et **prénom**, on peut bien sûr échanger l'ordre des arguments passés à **format** ou encore utiliser la *liaison par position*.

```
>>> "{1} {0} a {2} ans".format(prenom, nom, age)
'Dupont Jean a 35 ans'
```

Dans la pratique toutefois, cette forme est assez peu utile, on lui préfère⁸ souvent la *liaison par nom*.

```
>>> ("{} {} a {} ans"
... .format(le_nom=nom, le_prenom=prenom, l_age=age))
'Jean Dupont a 35 ans'
```

Dans ce premier exemple de liaison par nom, des noms différents ont été assignés pour les données externes et pour les noms apparaissant dans le format, pour bien illustrer comment la liaison se résout. On peut cependant être plus direct et employer les mêmes appellations que les noms de variables.

```
>>> ("{} {} a {} ans"
... .format(nom=nom, prenom=prenom, age=age))
'Jean Dupont a 35 ans'
```

v – FORMATAGE HISTORIQUE AVEC L'OPÉRATEUR % — La méthode **format** a été introduite assez tard dans PYTHON. Étant donné le volume de code écrit avec l'opérateur **%**, il semble important d'introduire cette construction. On ne doit cependant pas utiliser cet opérateur dans du

8. Petite digression : à remarquer l'usage des parenthèses, qui permettent de couper la ligne en deux, car sinon le code serait trop long pour la PEP8 (recommandations de style); on s'efforce toujours de ne pas dépasser 80 caractères de large, dans ce cas c'est utile notamment pour l'édition du document au format PDF.

code moderne ; la bonne manière de formater les chaînes de caractères est désormais le *f-string*.

Le principe de l'opérateur % est d'élaborer comme précédemment un « format » c'est-à-dire le patron de ce qui doit être rendu, auquel on passe des arguments pour « remplir » les trous. Les exemples déjà formulés sont repris ci-dessous.

```
>>> "%s %s a %s ans" % (prenom, nom, age)
'Jean Dupont a 35 ans'
```

L'ancienne formulation est bien moins lisible que les autres approches. On pouvait également avec cet opérateur recourir à un mécanisme de liaison par nommage, en passant par un dictionnaire (notion abordée prochainement).

```
>>> variables = {'le_nom': nom, 'le_prenom': prenom, 'l_age': age}
>>> "%(le_nom)s, %(le_prenom)s, %(l_age)s ans" % variables
'Dupont, Jean, 35 ans'
```

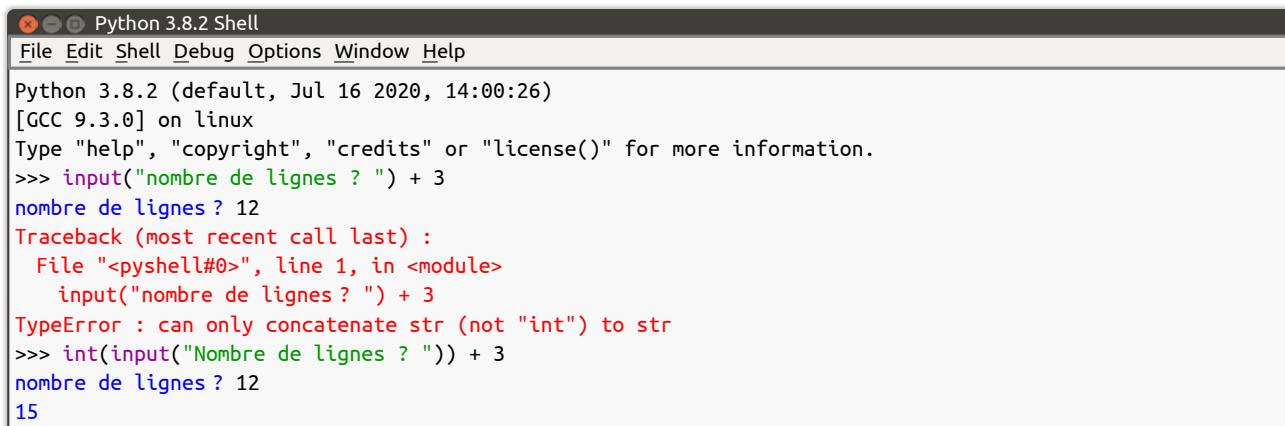
E. INTERACTION AVEC L'UTILISATEUR D'UN PROGRAMME

Occasionnellement, il est nécessaire de poser une question à l'utilisateur. C'est tout le propos de la fonction `input`.

```
In[1] nom_ville = input("Entrez le nom de la ville : ")
Out[1] Entrez le nom de la ville : Paris
In[2] print(f"nom_ville = {nom_ville}")
Out[2] nom_ville = Paris
```

4

Il faut bien avoir conscience que la fonction `input` renvoie toujours une chaîne de caractères (`str`). C'est assez évident, mais il est très facile de l'oublier et de passer cette chaîne directement à une fonction qui s'attend à recevoir, par exemple, un nombre entier, auquel cas les choses se passent mal. Dans un tel cas, il faut appeler la fonction `int` pour convertir le résultat en un entier.



The screenshot shows a terminal window titled "Python 3.8.2 Shell". The command line shows the following interaction:

```
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> input("nombre de lignes ? ") + 3
nombre de lignes ? 12
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    input("nombre de lignes ? ") + 3
TypeError: can only concatenate str (not "int") to str
>>> int(input("Nombre de lignes ? ")) + 3
nombre de lignes ? 12
15
```

La fonction `input` peut s'avérer bénéfique pour des premiers pas en PYTHON. Néanmoins, on l'utilise peu car les applications « réelles » viennent avec leur propre interface — souvent graphique — et disposent donc d'autres moyens pour interagir avec l'utilisateur.

Quant à elles, les applications destinées à fonctionner dans un terminal reçoivent traditionnellement leurs données de la ligne de commande, fonctionnalité du module `argparse` déjà rencontré au chapitre précédent « Notions fondamentales en PYTHON ».

1.2 Compléments et application

1.2.1 Expressions régulières

AVERTISSEMENT DES AUTEURS

Après avoir joué ce cours plusieurs années de suite, l'expérience montre qu'il est difficile de trouver le bon moment pour apprécier les expressions régulières. D'un côté il s'agit de manipulations de chaînes de caractères, mais d'un autre cela nécessite de créer des instances de classes et par suite d'avoir vu la programmation orientée objet. Du coup, les premières années elle étaient étudiées à la fin du cours, engendrant une certaine frustration. C'est pourquoi elle sont à présent introduites tôt, dans la séquence consacrée aux chaînes de caractères. Les étudiants qui seraient décontenancés par ce contenu sont invités à y retourner après la partie sur la programmation objet. Il semble important de savoir que ces fonctionnalités existent dans le langage, le détail de leur utilisation n'est toutefois pas critique et on peut parfaitement faire l'impassé sur ce complément en première lecture.

Une expression régulière est un objet mathématique qui décrit un ensemble de textes possédant des propriétés communes. Par exemple, s'il arrive d'utiliser un terminal et que l'on saisisse `$ ls *.txt` — ou bien `$ dir *.txt` sous Windows —, l'expression régulière `*.txt` désigne tous les fichiers dont le nom se termine par `.txt`. On dit que l'expression régulière filtre toutes les chaînes qui se terminent par `.txt` — la terminologie anglaise consacrée est le *pattern matching*.

Le langage PERL a été le premier à populariser l'utilisation des expressions régulières en les supportant nativement et non pas au travers d'une librairie. En PYTHON, les expressions régulières sont disponibles de manière plus traditionnelle, via le module `re` (pour *regular expressions*) de la librairie standard. Le propos de ce complément est d'en donner une première approche.

```
>>> import re
```

A. VUE D'ENSEMBLE

Pour le lecteur ne souhaitant pas approfondir, un premier exemple est proposé : on cherche à savoir si un objet chaîne est ou non de la forme `**.txt` et si oui, à calculer la partie qui remplace le « `*` ».

```
>>> regexp = "(.*)(.*).txt" # Objet expression régulière/pattern
>>> chaine = "abcdef.txt" # Chaîne de départ
>>> # Fonction qui calcule si la chaîne "matche" le pattern
>>> match = re.match(regexp, chaine)
>>> match is None
True
```

Le fait que l'objet `match` vaut `None` indique que la chaîne n'est pas de la bonne forme (il manque un tiret « - » dans le nom). En revanche, cela correspond avec une autre chaîne.

```
>>> chaine = "abc-def.txt" # Chaîne de départ
>>> # Fonction qui calcule si la chaîne "matche" le pattern
>>> match = re.match(regexp, chaine)
>>> match is None
False
```

Ici `match` est un objet, qui permet ensuite « d'extraire » les différentes parties.

```
>>> match[1]
'abc'
>>> match[2]
'def'
```

Bien entendu on peut s'attacher à des choses beaucoup plus élaborées avec `re`, mais en première lecture cette introduction doit suffire pour avoir une idée des possibilités avec les expressions régulières.

Avant de poursuivre, signalons qu'on emploie indifféremment les termes suivants pour désigner essentiellement la même chose :

- expression régulière;
- en anglais *regular expression*, d'où le nom du module `re`;
- en anglais raccourci *regexp*, *de facto* devenu un nom commun;
- en français on trouve aussi parfois le terme *d'expression rationnelle*, plus rare et un peu pédant;
- en anglais on utilise aussi facilement le terme de *pattern*;
- qui, du coup, est traduit par *motif*, d'un emploi assez rare.

Ensuite, selon les contextes, ces termes peuvent désigner des choses subtilement différentes; par exemple pour distinguer la chaîne qui spécifie un *pattern* de l'objet *regexp* qui en est déduit, mais à ce stade de la présentation on peut signaler tous ces termes et les assimiler en gros à la même notion.

B. ILLUSTRATION PAR L'EXEMPLE

Dans un terminal, `*.txt` est une expression régulière très simple. Le module `re`⁹ fournit le moyen de construire des expressions régulières très élaborées et plus puissantes que ce que supporte le terminal. C'est pourquoi la syntaxe des *regexprs* de `re` est un peu différente. Par exemple comme on vient de le voir, pour filtrer la même famille de chaînes que `*-*.*.txt` avec le module `re`, il a fallu écrire l'expression régulière sous une forme légèrement différente.

i- **MÉTHODE `findall`** — On se donne deux exemples de chaînes et on cherche tous les mots se terminant par « a » ou « m » dans une chaîne à l'aide de la méthode `findall`.

```
>>> sentences = ['Lacus a donec, vitae gravida proin sociis.',  
...                 'Neque ipsum! rhoncus cras quam.]'  
>>> for sentence in sentences:  
...     print(f"---- dans >{sentence}<")  
...     print(re.findall(r"\w*[am]\W", sentence))  
...  
---- dans >Lacus a donec, vitae gravida proin sociis.<  
['a ', 'gravida ']  
---- dans >Neque ipsum! rhoncus cras quam.<  
['ipsum!', 'quam.']}
```

Ce code permet de chercher toutes les occurrences de l'expression régulière, qui ici est définie par la chaîne : `r"\w*[am]\W"`.

Petite digression : cette manière de créer une chaîne en la préfixant par « `r` » s'appelle une *raw-string* – littéralement chaîne brute. L'intérêt est de ne pas interpréter les *backslash*.

```
>>> print("sans raw-string\nun newline")  
sans raw-string  
un newline  
>>> print(r"dans\nunraw-string")  
dans\nunraw-string
```

Dans une chaîne usuelle les caractères *backslash* ont une signification particulière, mais quand on crée une expression régulière, le but est de laisser les *backslashes* intacts, car c'est à la couche de *regexp* de les interpréter. Il n'y a aucune obligation à utiliser un *raw-string*, d'ailleurs on note qu'il n'y a pas de différence de nature entre un *raw-string* et une chaîne usuelle.

```
>>> raw = r'abc'  
>>> regular = 'abc'  
>>> # Comme on a pris une 'petite' chaîne ce sont les mêmes objets  
>>> print(f"both compared with is → {raw is regular}")  
both compared with is → True  
>>> # et donc a fortiori  
>>> print(f"both compared with == → {raw == regular}")  
both compared with == → True
```

Il se trouve que le *backslash* à l'intérieur des expressions régulières est d'un usage courant. C'est pourquoi on utilise fréquemment un *raw-string* pour décrire une expression régulière. Il désactive l'interprétation des *backslash* à l'intérieur de la chaîne. Par exemple, `\t` est in-

⁹. Il est conseillé de consulter la documentation du module `re` en parallèle de ce complément.

AVERTISSEMENT

Ce complément est amené à faire appel à des traits qui dépendent du `LOCALE`, c'est-à-dire de la configuration de l'ordinateur vis-à-vis de la langue. Il se peut que les résultats diffèrent légèrement en fonction de la station sur laquelle les codes sont exécutés. Pour ce document, la référence est un PC sous Ubuntu MATE 20.04 LTS configuré pour le français.

terprété comme un caractère de tabulation dans une chaîne usuelle. Sans *raw-string*, il faut doubler tous les *backslash* pour qu'il n'y ait pas d'interprétation non voulue.

Ceci étant posé, on peut détailler la signification de l'expression régulière `r"\w*[am]\W"` :

- `\w*`, chercher une sous-chaîne qui commence par un nombre, y compris nul (*) de caractères alphanumériques (`\w`). Ceci est défini en fonction de la variable d'environnement `LOCALE`;
- `[am]`, immédiatement après, trouver un caractère « `a` » ou « `m` »;
- `\W`, et enfin, détecter un caractère *qui ne soit pas alphanumérique*. Ceci est important puisqu'on cherche les mots qui se terminent par un « `a` » ou un « `m` », l'oublier fausserait le résultat.

```
>>> for sentence in sentences:
...     print(f"---- dans >{sentence}<")
...     print(re.findall(r"\w*[am]", sentence))
...
---- dans >Lacus a donec, vitae gravida proin sociis.<
['La', 'a', 'vita', 'gravida']
---- dans >Neque ipsum! rhoncus cras quam.<
['ipsum', 'cra', 'quam']
```

ii – MÉTHODE `split` — Une autre forme simple d'utilisation des *regexprs* est la méthode `re.split`, qui fournit une fonctionnalité voisine de celle pour les chaînes de caractères `str.split`, mais où les séparateurs sont exprimés comme une expression régulière.

```
>>> for sentence in sentences:
...     print(f"---- dans >{sentence}<")
...     print(re.split(r"\w+", sentence))
...
---- dans >Lacus a donec, vitae gravida proin sociis.<
['Lacus', 'a', 'donec', 'vitae', 'gravida', 'proin', 'sociis', '']
---- dans >Neque ipsum! rhoncus cras quam.<
['Neque', 'ipsum', 'rhoncus', 'cras', 'quam', '']
```

Ici l'expression régulière, qui bien sûr décrit le séparateur, est simplement `\w+` c'est-à-dire toute suite d'au moins un caractère non alphanumérique. Cela offre là un moyen simple et plus puissant de couper un texte en mots que `str.split`.

iii – MÉTHODE `sub` — Une troisième méthode utilitaire est `re.sub` qui permet de remplacer les occurrences d'une *regexp*.

```
>>> for sentence in sentences:
...     print(f"---- dans >{sentence}<")
...     print(re.sub(r"(\w+)", r"X\1Y", sentence))
...
---- dans >Lacus a donec, vitae gravida proin sociis.<
XLacusY XaY XdonecY, XvitaeY XgravidaY XproinY XsociisY.
---- dans >Neque ipsum! rhoncus cras quam.<
XNequeY XipsumY! XrhoncusY XcrasY XquamY.
```

L'expression régulière (le premier argument) contient ici un groupe : on a utilisé des parenthèses autour du `\w+`. Le second argument est la chaîne de remplacement, dans laquelle on a fait référence au groupe en écrivant `\1`, qui veut dire tout simplement « le premier groupe ». Donc au final, l'effet de cet appel est d'entourer toutes les suites de caractères alphanumériques par `X` et `Y`.

On regarde maintenant comment vérifier si une chaîne est conforme au critère défini par l'expression régulière, mais aussi extraire les mor-

ceaux de la chaîne qui correspondent aux différentes parties de l'expression. Pour ce faire, supposons qu'on s'intéresse aux chaînes qui comportent cinq parties : une suite de chiffres, une suite de lettres, des chiffres à nouveau, des lettres et enfin de nouveau des chiffres. On considère ces trois chaînes en entrée.

```
>>> samples = ['890hj000nnm890', # Cette entrée convient
...             '123abc456def789', # Celle-ci aussi
...             '8090abababab879', # Celle-ci non
...             ]
```

iv – MÉTHODE `match` — Pour commencer, on peut facilement tester si une chaîne vérifie ou non le critère voulu. On applique l'expression régulière à toutes les entrées.

```
>>> regexp1 = "[0-9]+[A-Za-z]+[0-9]+[A-Za-z]+[0-9]+"
>>> for sample in samples:
...     match = re.match(regexp1, sample)
...     print(f"{sample[:16]} → {match}")
...
890hj000nnm890 → <re.Match object; span=(0, 14), match='890hj000nnm890'>
123abc456def789 → <re.Match object; span=(0, 15), match='123abc456def789'>
8090abababab879 → None
```

Pour rendre ce résultat plus lisible, on définit une petite fonction de confort, puis on relance l'essai.

```
>>> def nice(match): # Visualisation d'une correspondance
...     # Le retour de re.match est soit None, soit un objet match
...     return "No" if match is None else "Match!"
...
>>> print(f"REGEXP={regexp1}\n")
REGEXP=[0-9]+[A-Za-z]+[0-9]+[A-Za-z]+[0-9]+

>>> for sample in samples:
...     match = re.match(regexp1, sample)
...     print(f"{sample[:16]} → {nice(match)})"
...
890hj000nnm890 → Match!
123abc456def789 → Match!
8090abababab879 → No
```

Ici plutôt que d'utiliser les raccourcis comme `\w`, on a préféré écrire explicitement les ensembles de caractères en jeu. De cette façon, on rend son code indépendant du `LOCALE` si c'est ce qu'on veut faire. Il y a deux morceaux qui interviennent tour à tour :

- `[0-9]+` une suite d'au moins un caractère dans l'intervalle `[0-9]`,
- `[A-Za-z]+` une suite d'au moins un caractère dans l'intervalle `[A-Z]` ou dans l'intervalle `[a-z]`.

Il est possible de nommer un morceau — un groupe — particulier de l'expression régulière pour convenance d'écriture ou de lecture. On se concentre sur une des deux entrées correctes et on appelle `needle` le groupe de chiffre central de l'expression régulière. Ce faisant, on peut demander de retrouver et imprimer la partie correspondante dans la chaîne initiale.

```
>>> haystack = samples[1]
>>> haystack
'123abc456def789'
>>> regexp2 = "[0-9]+[A-Za-z]+(?P<needle>[0-9]+)[A-Za-z]+[0-9]+"
>>> print(re.match(regexp2, haystack).group('needle'))
```

Dans l'expression ci-dessus, on a fait directement appel à un groupe nommé (?P<needle>[0-9]+), dans lequel :

- les parenthèses définissent le groupe,
- ?P<needle> spécifie que ce groupe pourra être référencé¹⁰ sous le nom de needle.

¹⁰ Cette syntaxe très absconse est héritée semble-t-il de PERL.

Pour terminer avec les exemples, on aborde un trait qui n'est pas présent dans tous les langages, à savoir restreindre un morceau de chaîne à être identique à un groupe déjà vu plus tôt dans la chaîne. On peut donc ici ajouter comme contrainte que le premier et le dernier groupes de chiffres soient identiques.

Comme précédemment, on a défini le groupe nommé id comme étant la première suite de chiffres. La nouveauté est la contrainte imposée sur le dernier groupe avec (?P=id). On n'obtient ainsi une correspondance qu'avec les entrées dans lesquelles le dernier groupe de chiffres est identique au premier.

```
>>> regexp3 = "(?P<id>[0-9]+)[A-Za-z]+(?P<needle>[0-9]+)[A-Za-z]+(?P=id)"  
>>> print(f"REGEXP={regexp3}\n")  
REGEXP=(?P<id>[0-9]+)[A-Za-z]+(?P<needle>[0-9]+)[A-Za-z]+(?P=id)  
  
>>> for sample in samples:  
...     match = re.match(regexp3, sample)  
...     print(f"{sample:>16} → {nice(match)}")  
...  
890hj000nnm890 → Match!  
123abc456def789 → No  
8090bababab879 → No
```

C. COMPILE DES EXPRESSIONS RÉGULIÈRES

Avant d'apprendre à écrire en propre une expression régulière, disons quelques mots du mode d'emploi de la bibliothèque.

i – FONCTIONS DE COMMODITÉ ET WORKFLOW — Une expression régulière décrite sous forme de chaîne, comme par exemple \w*[am]\W, peut se traduire dans un *automate fini* qui permet d'opérer le filtrage. Cela explique le *workflow* résumé en figure 9.2.

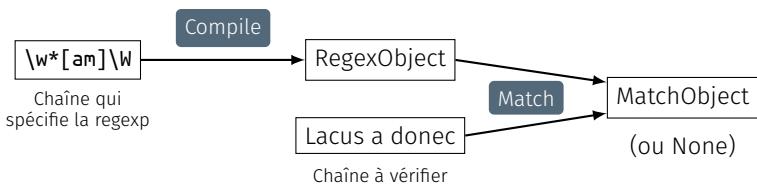


FIGURE 9.2 — Workflow d'une expression régulière.

Lorsqu'un même *pattern* est à appliquer sur un grand nombre de chaînes, la méthode recommandée pour utiliser la bibliothèque est :

- de compiler *une seule fois* la chaîne en automate, matérialisé par un objet de la classe `re.RegexObject`, en utilisant `re.compile`;
- puis d'employer cet objet autant de fois qu'il y a de chaînes.

Des fonctions de commodité du module ont été définies dans les exemples plus haut — il en est de même par la suite pour une meilleure lisibilité. Elles sont pratiques, par exemple, pour mettre au point une expression régulière en mode interactif, mais ne sont pas forcément adaptées dans tous les cas. Ces fonctions de commodité suivent toutes le même principe :

`re.match(pattern, sample)` `re.compile(pattern).match(sample)`

Donc à chaque fois qu'on utilise une fonction de commodité, on recompile la chaîne en automate, ce qui, dès qu'on a plus d'une chaîne à

traiter, représente un surcoût. La seconde version proposée ci-dessous ne compile qu'une fois la chaîne en automate et s'avère plus efficace.

```
>>> for sample in samples: # Imaginer 10**6 chaînes dans samples
...     match = re.match(regexp3, sample)
...     print(f"{sample:>16} → {nice(match)}")
...
 890hj000nnm890 → Match!
123abc456def789 → No
8090abababab879 → No
>>> re_obj3 = re.compile(regexp3) # On compile la chaîne en automate une seule fois,
>>> for sample in samples:      # ensuite on part directement de l'automate
...     match = re_obj3.match(sample)
...     print(f"{sample:>16} → {nice(match)}")
...
 890hj000nnm890 → Match!
123abc456def789 → No
8090abababab879 → No
```

ii – MÉTHODES SUR LA CLASSE `RegexObject` — Les objets de la classe `RegexObject` représentent donc l'automate à état fini qui est le résultat de la compilation de l'expression régulière. Pour résumer ce qu'on a déjà vu, les méthodes les plus utiles sur un objet `RegexObject` sont :

- `match` et `search`, qui cherchent une correspondance soit uniquement au début (`match`) ou n'importe où dans la chaîne (`search`),
- `findall` et `split` pour chercher toutes les occurrences (`findall`) ou leur négatif (`split`),
- `sub` (qui aurait pu sans doute s'appeler `replace`, mais c'est ainsi) pour remplacer les occurrences de `pattern`.

iii – MÉTHODES SUR LA CLASSE `re.MatchObject` — Pour exploiter les résultats, les méthodes disponibles sur la classe `re.MatchObject` sont documentées en détail. On en a déjà rencontré quelques-unes, en voici à nouveau un aperçu rapide :

- les méthodes `re` et `string` pour retrouver les données d'entrée du `match`;
- les méthodes `group`, `groups`, `groupdict` pour retrouver les morceaux de la chaîne d'entrée qui correspondent aux groupes de la `regexp`. On peut y accéder¹¹ par rang ou par nom.

¹¹. Comme vu plus haut avec `needle`.

```
>>> sample = "    Isaac Newton, physicist"
>>> match = re.search(r"(\w+) (?P<name>\w+)", sample)
>>> match.string
'    Isaac Newton, physicist'
>>> match.re
re.compile('(\w+) (?P<name>\w+)')
>>> match.groups()
('Isaac', 'Newton')
>>> match.group(1)
'Isaac'
>>> match.group('name')
'Newton'
>>> match.group(2)
'Newton'
>>> match.groupdict()
{'name': 'Newton'}
```

Comme on le voit pour l'accès par rang, les indices commencent à 1 pour des raisons¹² historiques. On peut aussi accéder au groupe 0 comme étant la partie de la chaîne de départ qui a effectivement été filtrée par l'expression régulière et qui, en général, est une sous-chaîne de la chaîne de départ.

¹². On pouvait déjà référencer 1 dans l'éditeur UNIX `sed` à la fin des années 1970 !

```
>>> match.group(0) # La sous-chaine filtrée
'Isaac Newton'
>>> sample # La chaîne de départ
' Isaac Newton, physicist'
```

On trouve également la méthode `expand` qui permet de faire une espèce de `str.format` avec les valeurs des groupes et la méthode `span` pour connaître les index dans la chaîne d'entrée pour un groupe donné.

```
>>> match.expand(r"last_name \g<name> first_name \1")
'last_name Newton first_name Isaac'
>>> sample # La chaîne de départ
' Isaac Newton, physicist'
>>> begin, end = match.span('name')
>>> sample[begin:end]
'Newton'
```

13. Notion abordé par la suite.

Nota bene : `seq[i:j]` est une opération de *slicing*¹³ — littéralement « trancher-partager » — qui retourne une séquence contenant les éléments de `i` à `j-1` de `seq`.

iv – DIFFÉRENTS MODES (FLAGS) — Il faut également noter qu'on peut passer à `re.compile` un certain nombre de *flags* — littéralement « drapeaux » — qui modifient globalement l'interprétation de la chaîne de caractères et qui peuvent rendre service.

La liste exhaustive de ces *flags* est disponible dans la documentation. Ils ont en général un nom long et parlant, et un alias court sur un seul caractère. Les plus utiles sont sans doute :

- `IGNORECASE` (alias `I`) pour ne pas faire de différence entre minuscules et majuscules;
- `UNICODE` (alias `U`) pour rendre les séquences `\w` et autres basées sur les propriétés des caractères dans la norme Unicode;
- `LOCALE` (alias `L`) cette fois `\w` dépend de la langue courante;
- `MULTILINE` (alias `M`) pour les *patterns* de début et fin de ligne;
- et `DOTALL` (alias `S`), pour faire du point « `.` » une correspondance avec tous les caractères, y compris une fin de ligne.

Comme c'est souvent le cas, on doit passer à `re.compile` un « OU » logique (caractère `|` ou « *pipe* ») des différents *flags* que l'on veut utiliser, c'est-à-dire comme suit.

```
>>> regexp = "a*b+"
>>> re_obj = re.compile(regexp, flags=re.IGNORECASE | re.DEBUG)
MAX_REPEAT 0 MAXREPEAT
    LITERAL 97
MAX_REPEAT 1 MAXREPEAT
    LITERAL 98

0. INFO 4 0b0 1 MAXREPEAT (to 5)
5: REPEAT_ONE 6 0 MAXREPEAT (to 12)
9.    LITERAL_UNI_IGNORE 0x61 ('a')
11.   SUCCESS
12: REPEAT_ONE 6 1 MAXREPEAT (to 19)
16.   LITERAL_UNI_IGNORE 0x62 ('b')
18.   SUCCESS
19: SUCCESS
>>> print(re_obj, "->", nice(re_obj.match("AabB"))) # On ignore la casse des caractères
a*b+ -> Match!
```

D. CONSTRUCTION D'UNE EXPRESSION RÉGULIÈRE

L'élaboration d'une expression régulière se veut ici synthétique (la documentation du module `re` en donne une version exhaustive).

La brique de base est le caractère quel qu'il soit et plusieurs approches sont alors possibles :

- un seul caractère, à citer tel quel, en le précédent d'un *backslash* « \ » s'il a par ailleurs un sens spécial dans le micro-langage de *regexp*s (comme « + », « * », « [», etc.);
- l'*attrape-tout* (*wildcard*), un point « . » signifie « n'importe quel caractère »;
- un *ensemble* de caractères avec la notation [...] qui permet de décrire par exemple :
 - ▶ [a1=] un ensemble *in extenso*, ici un caractère parmi « a », « 1 » ou « = »,
 - ▶ [a-z] un intervalle de caractères, ici de « a » à « z »,
 - ▶ [15e-g] un mélange des deux, ici un ensemble qui contiendrait « 1 », « 5 », « e », « f » et « g »,
 - ▶ [^15e-g] une négation, qui a « ^ » comme premier caractère dans les crochets, ici tout sauf l'ensemble précédent;
- un *ensemble prédéfini* de caractères, lesquels peuvent alors dépendre de l'environnement (**UNICODE** et **LOCALE**) avec entre autres les notations :
 - ▶ \w les caractères alphanumériques et \W (les autres),
 - ▶ \s les caractères "blancs" - espace, tabulation, saut de ligne, etc. et \S (les autres),
 - ▶ \d pour les chiffres et \D (les autres).

```
>>> sample = "abcd"
>>> for regexp in ['abcd', 'ab[cd][cd]', 'ab[a-z]d', r'abc.', r'abc\.']:
...     match = re.match(regexp, sample)
...     print(f"{sample} / {regexp}<10s> → {nice(match)}")%
File "<stdin>", line 3
  print(f"{sample} / {regexp}<10s> → {nice(match)}")%
                                                ^
SyntaxError: invalid syntax

>>> print(nice(re.match(r"abc\.", "abc.")))
Match!
```

Pour ce dernier exemple, comme un *backslash* est ajouté devant le point (« . »), il faut que la chaîne d'entrée contienne vraiment un point.

- i- EN SÉRIE OU BIEN EN PARALLÈLE — Par analogie avec les montages électriques, on a vu jusqu'ici le montage en série : les expressions régulières sont mises bout à bout et filtrent (**match**) séquentiellement la chaîne en entrée du début à la fin. On a *un peu* de marge pour spécifier des alternatives, lorsqu'on fait par exemple "ab[cd]ef", mais c'est limité à *un seul* caractère. Si on veut reconnaître deux mots qui n'ont pas grand-chose à voir comme **abc** ou **def**, il faut en quelque sorte mettre deux *regexp*s en parallèle, c'est ce que permet l'opérateur « | ».

```
>>> regexp = "abc|def"
>>> for sample in ['abc', 'def', 'aef']:
...     match = re.match(regexp, sample)
...     print(f"{sample} / {regexp} → {nice(match)}")
...
abc / abc|def → Match!
def / abc|def → Match!
aef / abc|def → No
```

- ii- FIN(S) DE CHAÎNE — Selon que l'on utilise **match** ou **search**, on précise si on s'intéresse uniquement à une correspondance en début (**match**) de chaîne ou n'importe où (**search**) dans la chaîne.

Mais indépendamment de cela, il peut être intéressant de « coller » l'expression en début ou en fin de ligne. Pour ce faire, il existe des caractères spéciaux :

- `^` lorsqu'il est utilisé comme un caractère (c'est à dire pas en début de []) signifie un début de chaîne;
- `\A` a le même sens (sauf en mode `MULTILINE`), recommandé de préférence à `^` qui est déjà pas mal surchargé;
- `$` correspond à une fin de ligne;
- `\z` est voisin de `$` mais pas tout à fait identique.

Se reporter à la documentation pour le détail des différences. Attention aussi à entrer le `^` correctement, il faut le caractère ASCII et non un voisin dans la ménagerie Unicode.

```
>>> sample = 'abcd'
>>> for regexp in [ r'bc', r'\Aabc', r'^abc',
...                   r'\Abc', r'^bc', r'bcd\Z',
...                   r'bcd$', r'bc\Z', r'bc$' ]:
...     match = re.match(regexp, sample)
...     search = re.search(regexp, sample)
...     print(f"{sample} / {regexp:5s} match → {nice(match):6s},"
...           f" search → {nice(search)}")
...
abcd / bc      match → No      , search → Match!
abcd / \Aabc   match → Match!, search → Match!
abcd / ^abc    match → Match!, search → Match!
abcd / \Abc    match → No      , search → No
abcd / ^bc    match → No      , search → No
abcd / bcd\Z   match → No      , search → Match!
abcd / bcd$   match → No      , search → Match!
abcd / bc\Z    match → No      , search → No
abcd / bc$    match → No      , search → No
```

On a en effet bien le *pattern* `bc` dans la chaîne en entrée, mais il n'est ni au début ni à la fin.

iii – PARENTHÉSAGE OU GROUPEMENT — Pour pouvoir faire des montages élaborés, il faut pouvoir introduire des parenthèses.

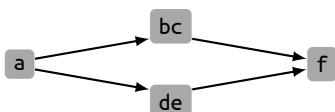


FIGURE 9.3 – Flux avec parenthésage.

```
>>> # Parenthèses dans une RE pour mettre en ligne :
>>> # un début 'a', un milieu 'bc' ou 'de' et une fin 'f'
>>> regexp = "a(bc|de)f"
>>> for sample in ['abcf', 'adef', 'abef', 'abf']:
...     match = re.match(regexp, sample)
...     print(f"{sample:>4s} → {nice(match)}")
...
abcf → Match!
adef → Match!
abef → No
abf → No
```

Les parenthèses jouent un rôle additionnel de groupe, ce qui signifie qu'on peut retrouver le texte correspondant à l'expression régulière comprise dans les parenthèses.

Ainsi, pour la première correspondance de l'exemple ci-dessus, on n'a utilisé qu'un seul groupe de parenthèse « `()` » et le morceau de chaîne qui correspond à ce groupe se trouve donc être le seul groupe retourné par `MatchObject.groups()`.

```
>>> sample = 'abcf'
>>> match = re.match(regexp, sample)
>>> print(f"{sample}, {regexp} → {match.groups()}")
abcf, a(bc|de)f → ('bc',)
```

iv – COMPTER LES RÉPÉTITIONS — Pour gérer les répétitions, on dispose des opérateurs suivants :

- * l'étoile signifiant n'importe quel nombre d'occurrences, même nul — par ex., `(ab)*` pour indiquer '' ou 'ab' ou 'abab' ou etc.;
- + le plus qui signifie au moins une occurrence — c.-à.-d. `(ab)+` pour ab ou abab ou ababab ou etc.;
- ? qui indique une option, c'est-à-dire zéro ou une occurrence — autrement dit `(ab)?` *matche* '' ou ab;
- {n} pour exactement n occurrences de `(ab)` — c.-à.-d. `(ab)3` qui serait exactement équivalent à ababab;
- {m,n} entre m et n fois inclusivement.

```
>>> # NB: la construction [op(elt) for elt in iterable] est une compréhension de liste étudiée plus tard.
>>> # Elle retourne une liste des résultats de l'opération op sur chaque élément de la liste de départ
>>> samples = [n*'ab' for n in [0, 1, 3, 4]] + ['babab']
>>> for regexp in ['(ab)*', '(ab)+', '(ab){3}', '(ab){3,4}']:
...     # on ajoute |A|Z pour matcher toute la chaîne
...     line_regexp = r"\A{} \Z".format(regexp)
...     for sample in samples:
...         match = re.match(line_regexp, sample)
...         print(f"{sample:>8s} / {line_regexp:14s} → {nice(match)}")
...
    / \A(ab)*\Z      → Match!
  ab / \A(ab)*\Z    → Match!
ababab / \A(ab)*\Z → Match!
abababab / \A(ab)*\Z → Match!
  baba / \A(ab)*\Z → No
    / \A(ab)+\Z    → No
    ab / \A(ab)+\Z  → Match!
  ababab / \A(ab)+\Z → Match!
abababab / \A(ab)+\Z → Match!
  baba / \A(ab)+\Z → No
    / \A(ab){3}\Z   → No
    ab / \A(ab){3}\Z → No
  ababab / \A(ab){3}\Z → Match!
abababab / \A(ab){3}\Z → No
  baba / \A(ab){3}\Z → No
    / \A(ab){3,4}\Z → No
    ab / \A(ab){3,4}\Z → No
  ababab / \A(ab){3,4}\Z → Match!
abababab / \A(ab){3,4}\Z → Match!
  baba / \A(ab){3,4}\Z → No
```

v – GROUPES ET CONTRAINTES — On a déjà vu un exemple de groupe nommé (voir `needle` plus haut). Les opérateurs de cette catégorie sont :

- (...) les parenthèses définissent un groupe anonyme;
- (?P<name>...) définit un groupe nommé;
- (?:...) permet de mettre des parenthèses mais sans créer un groupe, pour optimiser l'exécution puisqu'on n'a pas besoin de conserver les liens vers la chaîne d'entrée;
- (?P=name) qui ne *matche* que si l'on retrouve à cet endroit de l'entrée la même sous-chaîne que celle trouvée pour le groupe `name` en amont;
- enfin (?=...), (?!=...) et (?<=...) permettent des contraintes encore plus élaborées, est laissé au lecteur le soin d'expérimenter ces possibilités; à savoir toutefois que l'utilisation de telles constructions peut en théorie rendre l'interprétation de l'expression régulière beaucoup moins efficace.

vi – GREEDY VERSUS NON-GREEDY — Lorsqu'on stipule une répétition un nombre indéfini de fois, il se peut qu'il existe plusieurs fa-

çons de filtrer l'entrée avec l'expression régulière. Que ce soit avec *, + ou ?, l'algorithme va toujours essayer de trouver la séquence la plus longue, c'est pourquoi on qualifie l'approche de *greedy* – quelque chose comme « *glouton* » en français.

```
>>> line='<h1>Title</h1>' # Fragment d'HTML
>>> # Si on cherche un texte quelconque entre crochets c'est-à-dire l'expression régulière « <.*> »
>>> re_greedy = '<.*>'
>>> # On obtient ceci, on rappelle que group(0) montre la partie du fragment HTML qui matche la regexp
>>> match = re.match(re_greedy, line)
>>> match.group(0)
'<h1>Title</h1>'
```

Cela n'est pas forcément ce qu'on voulait faire, aussi on peut spécifier l'approche inverse, c'est-à-dire trouver la plus-petite chaîne qui *matche*, dans une approche dite non-*greedy*, avec les opérateurs :

- ▶ ***?** : * mais non-greedy;
 - ▶ **+?** : + mais non-greedy;
 - ▶ **??** : ? mais non-greedy.

```
>>> # Ici on va remplacer * par *? pour rendre l'opérateur * non-greedy
>>> re_non_greedy = re_greedy = '<.*?>'
>>> # Mais on continue à chercher un texte entre <> naturellement si bien que cette fois, on obtient
>>> match = re.match(re_non_greedy, line)
>>> match.group(0)
'<h1>'
```

vii – TRAITEMENT DES FINS DE LIGNE — Il peut être utile, pour conclure cette présentation des expressions régulières, de préciser un peu le comportement de la bibliothèque vis-à-vis des fins de ligne.

Historiquement, les expressions régulières telles qu'on les trouve dans les bibliothèques C, donc dans **sed**, **grep** et autres utilitaires UNIX, sont associées au modèle mental où on filtre les entrées ligne par ligne. Le module **re** en garde des traces.

```
>>> # Exemple de traitement des 'newlines'  
>>> sample = """une entrée  
... sur  
... plusieurs  
... lignes  
...  
"""  
>>> match = re.compile("(.*)").match(sample)  
>>> match.groups()  
('une entrée',)
```

On voit donc que l'attrape-tout '.' en fait ne recueille pas le caractère de fin de ligne \n, puisque si c'était le cas et compte tenu du côté *greedy* de l'algorithme, on devrait voir ici tout le contenu de **sample**. Il existe un *flag* **re.DOTALL** qui permet de faire du point un vrai attrape-tout qui capture aussi les instructions de nouvelle ligne.

```
>>> match = re.compile("(.*)", flags=re.DOTALL).match(sample)
>>> match.groups()
('une entrée\nsur\nplusieurs\nlignes\n.',)
```

Cela dit, le caractère *newline* est par ailleurs considéré comme un autre, on peut le mentionner dans une *regexp* comme les autres. Voici quelques exemples pour illustrer tout ceci.

```
>>> # Depuis Python 3, sans mettre de flag, \w matche l'Unicode
>>> match = re.compile("([\w ]*)").match(sample)
>>> match.groups()
```

```
('une entrée',)
>>> # Pour matcher les caractères ASCII avec \w, il faut mentionner le flag ASCII re.A
>>> match = re.compile("([\w]*)", flags=re.A).match(sample)
>>> match.groups()
('une entr',)
>>> # Si on ajoute \n à la liste des caractères attendus, on obtient bien tout le contenu initial
>>> match = re.compile("([\w \n]*)", flags=re.UNICODE).match(sample)
>>> match.groups()
('une entrée\nsur\nplusieurs\nlignes\n',)
```

viii – CONCLUSION — La mise au point d’expressions régulières est certes un peu exigeante et demande pas mal de pratique, mais permet d’écrire en quelques lignes des fonctionnalités¹⁴ très puissantes.

^{14.} C'est un investissement très rentable!

On peut enfin mentionner l’existence de sites Web qui évaluent une expression régulière de manière interactive et qui peuvent rendre la mise au point moins¹⁵ fastidieuse.

^{15.} Notamment <https://pythex.org/> ou <https://regex101.com>.

ix – POUR EN SAVOIR PLUS — Pour ceux qui ont quelques rudiments de la théorie des langages, il est connu qu’on distingue en général :

- l’analyse lexicale, qui découpe le texte en morceaux (qu’on appelle des tokens);
- et l’analyse syntaxique qui décrit pour simplifier à l’extrême, l’ordre dans lequel on peut trouver les tokens.

Avec les expressions régulières, on adresse le niveau de l’analyse lexicale. Pour l’analyse syntaxique, franchement au delà des objectifs de ce cours, il existe de nombreuses alternatives, parmi lesquelles :

- PYPARSING;
- PLY (Python Lex-Yacc);
- ANTLR, outil écrit en JAVA qui peut générer des parsers PYTHON.

1.2.2 Exercices

Sont ici proposés quelques exercices sur les expressions régulières. Quelques remarques¹⁶ avant de commencer :

^{16.} Pour ces exercices, il peut être profitable d’avoir sous la main :

- on se concentre sur l’écriture de l’expression régulière en elle-même, non pas sur l’utilisation de la bibliothèque;
- en particulier, tous les exercices font appel à `re.match` entre la regexp à élaborer et une chaîne d’entrée qui sert de jeux de test.

- ▶ la documentation officielle;
- ▶ l’accès à un site qui permet de mettre au point la regexp de manière interactive et rapide (cf. note précédente).



EXERCICE 7 — IDENTIFICATEURS PYTHON



Il est demandé d’écrire une expression régulière qui décrit les noms de variable en PYTHON. Pour cet exercice on se concentre uniquement sur les caractères ASCII. On exclut donc les noms de variables qui pourraient contenir des caractères exotiques comme les caractères accentués ou encore les lettres grecques.

Il s’agit donc de reconnaître toutes les chaînes qui commencent par une lettre ou un underscore « _ », suivi de lettres, chiffres ou d’autres underscores. Le tableau qui suit en donne quelques exemples.

TABLE 9.1 — Exemple de chaînes commençant par une lettre ou un underscore.

Chaîne	Match?	Chaîne	Match?
'a'	Vrai	' '	Vrai
'_'	Faux	'__'	Vrai
'aa1'	Vrai	'A1a'	Vrai
'1Aa'	Faux	'-aa1'	Faux

Solution page 325



EXERCICE 8 — LIGNES AVEC NOM ET PRÉNOM



On veut reconnaître dans un fichier toutes les lignes qui contiennent un nom et un prénom. Plus précisément, on cherche les chaînes qui :

- commencent par une suite de caractères alphanumériques (on peut utiliser l'instruction « \w » ou tiret haut « - » qui constitue le prénom;
- contiennent ensuite comme séparateur le caractère deux-points « : »;
- contiennent ensuite une suite — cette fois jamais vide — de caractères alphanumériques, qui constitue le nom;
- et enfin contiennent un deuxième « : » mais en option seulement.

On demande de construire une expression régulière définissant les deux groupes **nom** et **prenom** et rejetant les lignes qui ne satisfont pas ces critères.

Dans la correction — et ce sera pareil pour tous les exercices avec des groupes — la correction affiche *uniquement les groupes demandés*; ici on va montrer les groupes **nom** et **prenom**. On a parfaitement le droit d'utiliser des groupes supplémentaires, nommés ou pas, dans la *regexp*.

[Solution page 325](#)



EXERCICE 9 — NUMÉROS DE TÉLÉPHONE



On veut reconnaître des numéros de téléphone français :

- soit au format contenant 10 chiffres dont le premier est un 0;
- soit au format international commençant par +33 suivi de 9 chiffres.

Dans tous les cas on veut trouver dans le groupe **number** les 9 chiffres vraiment significatifs.

[Solution page 326](#)

¹⁷. Pour plus de précisions sur ce trait — laissé de côté pour ne pas trop alourdir le complément —, voir la documentation sur les expressions régulières et cherchez la première occurrence de `iLmsux`.

Vu comment sont conçus les exercices, on ne peut pas passer à la méthode `re.compile` un drapeau¹⁷ comme `re.IGNORECASE` ou autre. Cependant, on peut embarquer ces drapeaux dans la *regexp* elle-même; par exemple pour la rendre insensible à la casse de caractères, au lieu d'appeler `re.compile` avec le flag `re.I`, on utilise `(?i)` comme suit.

```
>>> import re
>>> # On peut embarquer les flags comme IGNORECASE directement dans la regexp, équivalent de faire ceci
>>> re_obj = re.compile("abc", flags=re.IGNORECASE)
>>> re_obj.match("ABC").group(0)
'ABC'
>>> re.match("(?i)abc","ABC").group(0) # ou de faire cela
'ABC'
>>> # Les flags comme (?i) doivent apparaître en premier dans la regexp
>>> re.match("abc(?i)","ABC").group(0)
'ABC'
```



EXERCICE 10 — DÉCORTIQUER UNE URL



On demande d'écrire une expression régulière qui permette d'analyser des URLs. Voici les conventions adoptées pour l'exercice :

- la chaîne contient les parties `<protocol>://<location>/<path>`;
- l'URL commence par le nom d'un protocole qui doit être parmi `http`, `https`, `ftp`, `ssh`;
- le nom du protocole peut contenir de manière indifférente des minuscules ou des majuscules;
- ensuite doit venir la séquence « `//` »;
- ensuite on va trouver une chaîne `<location>` qui contient :
 - ▶ potentiellement un nom d'utilisateur et s'il est présent, potentiellement un mot de passe;
 - ▶ obligatoirement un nom de `hostname`;
 - ▶ potentiellement un numéro de port;

- lorsque les quatre parties sont présentes dans <location>, cela fournit :
`<location> = <user>:<password>@<hostname>:<port>;`
- si l'on note entre crochets les parties optionnelles, cela donne :
`<location> = [<user>[:<password>]@]<hostname>[:<port>];`
- le champ <user> ne peut contenir que des caractères alphanumériques ; si le @ est présent, le champ <user> ne peut pas être vide ;
- le champ <password> peut contenir tout sauf un « : » et de même, si le « : » est présent le champ <password> ne peut pas être vide ;
- le champ <hostname> peut contenir une suite non-vide de caractères alphanumériques, *underscores*, ou « . » ;
- le champ <port> ne contient que des chiffres et est non vide si le « : » est spécifié ;
- le champ <path> peut être vide.

■ [Solution page 326](#)



SOLUTION 7 — IDENTIFICATEURS PYTHON (PAGE 323)

Un identificateur commence par une lettre ou un *underscore* et peut être suivi par n'importe quel nombre de lettre, chiffre ou *underscore*, ce qui se trouve être « \w » si on ne se met pas en mode Unicode.

La solution est donc : "[a-zA-Z_]\w*". Cette expression peut bien entendu également s'écrire en clair : "[a-zA-Z_][a-zA-Z0-9_]*".

```
>>> def layout(match): # Fonction de mise en forme
...     # le retour de re.match est soit "Vrai", soit "Faux"
...     return "Faux" if match is None else "Vrai"
...
>>> samples = ['a',' ','-','__', 'aa1','A1a','1Aa','-aa1']
>>> regexp_pythonid = "[a-zA-Z_]\w*"
>>> for sample in samples:
...     match = re.match(regexp_pythonid, sample)
...     print(f"{sample:>8} → {layout(match)}")
...
    a → Vrai
    → Faux
    - → Faux
    __ → Vrai
    aa1 → Vrai
    A1a → Vrai
    1Aa → Faux
    -aa1 → Faux
>>> regexp_pythonid_bis = "[a-zA-Z_][a-zA-Z0-9_]*"
>>> for sample in samples:
...     match = re.match(regexp_pythonid_bis, sample)
...     print(f"{sample:>8} → {layout(match)}")
...
    a → Vrai
    → Faux
    - → Faux
    __ → Vrai
    aa1 → Vrai
    A1a → Vrai
    1Aa → Faux
    -aa1 → Faux
```



SOLUTION 8 — LIGNES AVEC NOM ET PRÉNOM (PAGE 324)

L'exercice est fondé sur `re.match`, cela signifie que la correspondance est cherchée en début de chaîne, mais il faut bien mettre \Z à la fin de la `regexp`, sinon par ex. avec la sixième entrée le nom « Du Pré » sera reconnu

partiellement comme « Du » au lieu d'être rejeté à cause de l'espace. Du coup bien penser à toujours définir les *regexp* avec des *raw-strings*.

Par ailleurs, il faut remarquer l'utilisation de `:?` à la fin de la *regexp* pour signifier qu'on peut mettre ou non un deuxième séparateur « `:` ».

La solution donne : `r"\A(?P<prenom>[-\w]*)(?P<nom>[-\w]+):?\Z"`.

```
>>> agenda_strings = [
...     "Daniel:Durand",
...     "Jean:Dupont:",
...     "Jean:Dupont::",
...     ":Dupontel:",
...     "Jean-Noël:Dupont-Nemours",
...     "Charles-Henri:Du Pré",
...     "Charles Henri:DuPré",
... ]
>>> agenda = r"\A(?P<prenom>[-\w]*)(?P<nom>[-\w]+):?\Z"
>>> for data in agenda_strings:
...     match = re.match(agenda, data)
...     print(f"{data:>24} → {layout(match)}")
...
Daniel:Durand → Vrai
Jean:Dupont: → Vrai
Jean:Dupont:: → Faux
:Dupontel: → Vrai
Jean-Noël:Dupont-Nemours → Vrai
Charles-Henri:Du Pré → Faux
Charles Henri:DuPré → Faux
```



SOLUTION 9 — NUMÉROS DE TÉLÉPHONE (PAGE 324)

La même remarque que dans l'exercice précédent s'applique concernant `\Z`. Par ailleurs, il faut bien mettre un *backslash* le `+` devant les chiffres de l'international (+33) car sinon cela veut dire « un ou plusieurs ».

Ainsi, la solution s'écrit : `r"(\+33|0)(?P<number>[0-9]{9})\Z"`

```
>>> phone_strings = [
...     "0123456789",
...     "01234567890",
...     "012345678",
...     "1234567890",
...     "+33123456789",
...     "+3312345678",
...     "+330123456789"]
>>> phone = r"(\+33|0)(?P<number>[0-9]{9})\Z"
>>> for data in phone_strings:
...     match = re.match(phone, data)
...     print(f"{data:>16} → {layout(match)}")
...
0123456789 → Vrai
01234567890 → Faux
012345678 → Faux
1234567890 → Faux
+33123456789 → Vrai
+3312345678 → Faux
+330123456789 → Faux
```



SOLUTION 10 — DÉCORTIQUER UNE URL (PAGE 324)

Une première solution est apportée par la construction pas-à-pas des différents groupes constitutifs de l'URL

```

>>> url_strings = """
... http://www.google.com/a/b
... HttPS://www.google.com:8080/a/b
... http://user@www.google.com/a/b
... FTP://username:hispass@www.google.com/
... ssh://missing.ending.slash
... gopher://unsupported.proto.col/
... http://missing/hostname/
... """.split()

>>> # En ignorant la casse on pourra ne mentionner
>>> # les noms de protocoles qu'en minuscules
>>> i_flag = "(?i)"

>>> # Pour élaborer la chaîne (proto1/proto2/...)
>>> protos_list = ['http', 'https', 'ftp', 'ssh', ]
>>> protos = "(?P<proto>" + "|".join(protos_list) + ")"

>>> # À l'intérieur de la zone 'user/password', la partie
>>> # password est optionnelle - mais on ne veut pas le ':' dans
>>> # le groupe 'password' - il nous faut deux groupes
>>> password = r"(:(?P<password>[^:]*)?)"

>>> # La partie user-password elle-même est optionnelle
>>> # on utilise ici un raw f-string avec le préfixe rf
>>> # pour insérer la regexp <password> dans la regexp <user>
>>> user = rf"((?P<user>\w+){password}@)?"

>>> # Pour le hostname on accepte des lettres, chiffres,
>>> # underscore et '.' attention à backslash . car sinon
>>> # ceci va matcher tout, y compris /
>>> hostname = r"(?P<hostname>[\w\.\.]+)"

>>> # Le port est optionnel
>>> port = r"(:(?P<port>\d+))?"

>>> # Après le premier slash
>>> path = r"(?P<path>.*)"

>>> # On assemble le tout
>>> url = i_flag+protos+"://"+user+hostname+port+'/'+path

>>> for data in url_strings:
...     match = re.match(url, data)
...     print(f"{data:>40} → {layout(match)}")
...
    http://www.google.com/a/b → Vrai
    HttPS://www.google.com:8080/a/b → Vrai
    http://user@www.google.com/a/b → Vrai
    FTP://username:hispass@www.google.com/ → Vrai
    ssh://missing.ending.slash → Faux
    gopher://unsupported.proto.col/ → Faux
    http://missing/hostname/ → Faux

```

Une seconde solution apportée par un participant au Mooc fait appel à `re.X` (cf. <https://docs.python.org/fr/3/library/re.html#re.X>), ce qui donne une présentation beaucoup plus compacte.

```

>>> protos_list = ['http', 'https', 'ftp', 'ssh', ]
>>> url_bis = rf"""(?x)                      # verbose mode
...      (?i)                         # ignore case

```

```

...
...     (?P<proto>"|".join(protos_list))) # http|https|...
...     ://                                         # separator
...     ((?P<user>\w+){password}@)?           # optional user/pwd
...     (?P<hostname>[\w\.\.]+)                 # mandatory hostname
...     (:(?P<port>\d+))?                      # optional port
...     /(?P<path>.*?)                         # mandatory path
...
"""

>>> for data in url_strings:
...     match = re.match(url_bis, data)
...     print(f"{data:>40} → {layout(match)}")
...
http://www.google.com/a/b → Vrai
HTTPS://www.google.com:8080/a/b → Vrai
http://user@www.google.com/a/b → Vrai
FTP://username:hispass@www.google.com/ → Vrai
ssh://missing.ending.slash → Faux
gopher://unsupported.proto.col/ → Faux
http://missing/hostname/ → Faux

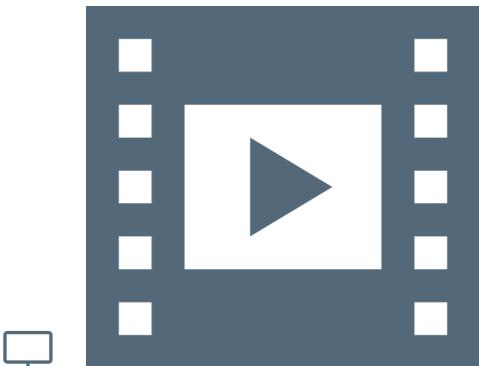
```

2 Notions de séquences et de listes

2.1 Définition et manipulation

2.1.1 Séquences

¹⁸. En pratique, les séquences concernent plus de types que ceux cités ici.



VIDÉO 9.4 — Notion de séquence.

En PYTHON, les séquences regroupent un ensemble de types¹⁸ tels que les listes, les tuples, les chaînes de caractères et les octets.

Une séquence en PYTHON est un ensemble fini et ordonné d'éléments indicés de 0 à n - 1 si l'on a n éléments. Prenons un exemple et commençons avec une chaîne de caractères : s = 'egg, bacon'. Quelles sont les opérations communes aux objets « séquence » ?

On peut tout d'abord vouloir accéder à chaque élément de la séquence avec la notation entre crochet; soit pour le premier élément s[0] et le dernier s[9]. Au passage, pour connaître le nombre d'éléments dans une séquence on utilise la fonction *built-in* len(s).

Toutes les séquences supportent le test appartenance, opération très puissante en PYTHON. Par exemple ici, est-que 'egg' est dans s. Ou inversement est-ce 'egg' n'est pas dans s. Ce test est relativement expressif et s'avère quasiment du langage naturel.

```

>>> s = 'egg, bacon'
>>> s[0]
'e'
>>> s[9]
'n'
>>> len(s)
10
>>> 'egg' in s
True
>>> 'egg' not in s
False

```

On peut ensuite faire de la concaténation de séquence en complétant la chaîne par une nouvelle chaîne de caractères ', and beans' pour obtenir un nouvel objet, ici de type chaîne de caractères.

Il existe également des opérations qui permettent de retrouver certains éléments, comme s.index(g) et s.count(g) pour respectivement

obtenir la première occurrence et le nombre d'occurrences de 'g'. On peut en outre faire appel aux fonctions *buit-in* `min(s)` et `max(s)` qui offre en retour le minimum et le maximum de la séquence, dans l'ordre lexicographique ici puisque la séquence est une chaîne de caractères.

Il existe une autre opération dite de *shallow copy*¹⁹ — littéralement « copie superficielle » — qui permet de faire une copie d'un nombre donné de fois de la séquence. Pour simplifier, on considère ici comme séquence le caractère 'x'.

```
>>> s
'egg, bacon'
>>> s + ', and beans'
'egg, bacon, and beans'
>>> s.index('g')
1
>>> s.count('g')
2
>>> min(s)
'
>>> max(s)
'o'
>>> 'x'*30
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

Une dernière opération abordée ici est celle de *slicing* — littéralement « taille, partage ». Elle permet de repérer et d'extraire les éléments d'une séquence selon le système d'indication.

Pour illustration, le point de départ est de reprendre la chaîne de caractères 'egg, bacon' en indiquant ses éléments (cf. ci-contre). L'opération de *slicing* se réalise en considérant une borne de gauche *inclue* et une borne de droite dont l'élément est *exclu*. Sa notation fait appel aux crochets et au glyphe de la double ponctuation, deux-points « : ».

Ainsi, `s[0:3]` retourne les trois premiers éléments de la chaîne de caractère, à savoir 'egg' et `s[5:10]`, les éléments de 5 à 9 : 'bacon'. Si la borne de gauche n'est pas spécifiée (`s[:3]`), cela parcourt tous les éléments du début de la séquence à la borne droite *exclue* ('egg' ici'). Inversement, si la borne de droite n'est pas spécifiée (`s[5:]`), cela parcourt tous les éléments de la borne de gauche *inclue* jusqu'à le fin de la séquence ('bacon' ici). Si aucune des bornes n'est spécifiée (`s[:]`), cela retourne une copie dite *shallow copy* de la séquence, mais pas l'objet de départ.

```
>>> s = 'egg, bacon'
>>> s[0:3]
'egg'
>>> s[5:10]
'bacon'
>>> s[:3]
'egg'
>>> s[5:]
'bacon'
>>> s[:]
'egg, bacon'
```

Pour parcourir les éléments d'une séquence, on peut introduire la notion de *pas*, par exemple un sur deux éléments si on donne un pas de deux : `s[0:10:2]`, soit ici la chaîne de caractère 'eg ao'. Si on considère l'instruction `s[::-2]`, soit parcourir la séquence du début à la fin par pas de deux, cela donne le même résultat. Si maintenant on saisi `s[:8:3]`, l'extraction des éléments se fait du début de la séquence jusqu'au huitième élément exclu par pas de trois et fournit : 'e,a'. A contrario,

¹⁹. Le concept de *shallow copy* sera repris et explicité plus avant par la suite. Il peut y avoir des effets de bords lorsque l'objet considéré est un objet mutable et non immuable comme une chaîne de caractères.

e g g , b a c o n
0 1 2 3 4 5 6 7 8 9

Indication d'une séquence.

`s[2:2:3]` parcourt la séquence du deuxième élément à la fin par pas de trois, ce qui conduit à '`gbo`'.

```
>>> s[0:10:2]
'eg ao'
>>> s[::-2]
'eg ao'
>>> s[:8:3]
'e,a'
>>> s[2::3]
'gbo'
```

```
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
e g g , b a c o n
0 1 2 3 4 5 6 7 8 9
```

Indiquez négatif d'une séquence.



VIDÉO 9.5 – Notion de liste.

BLOC-NOTE VIDÉO 9.3 (copier-coller)

```
a = []
type(a)
i = 4
a = [i, 'spam', 3.2, True]
a
a[0]
a[0] = 6
a
a[0] = a[0] + 10
a
a[1:3]
a[1:3] = [1, 2, 3]
a
a[1:3] = []
a
del a[1:2]
a
dir(list)
list.append?
help(list.append)
a
a.append('18')
a
a.extend([1, 2, 3])
a a = [1, 5, 3, 1, 7, 9, 2]
a.sort()
a
a = a.sort()
a
s = 'spam egg beans'
a = s.split()
a
a[0] = a[0].upper()
a
" ".join(a)
```

Que se produit-il si les indices dépassent la taille de la séquence ? L'instruction `s[100]` conduit assez logiquement à lever une exception et donne une erreur, mais si on introduit `s[5:100]` cela retourne '`bacon`' sans erreur. La raison est que PYTHON prend tous les éléments couverts par le *slice* et va chercher l'intersection avec les indices disponibles dans la chaîne de caractères considérée. Si on rentre l'instruction `s[100:200]`, le principe est le même : comme l'intersection est vide, cela retourne toujours un objet chaîne de caractère mais sans valeur.

```
>>> s[100]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> s[5:100]
'bacon'
>>> s[100:200]
''
```

Un autre aspect de l'opération de *slicing* est relatif aux indices négatifs. Le comportement est toujours le même mais la numération commence par le dernier élément. Cela s'avère pratique lorsque l'on veut accéder directement à la fin d'une séquence dont on connaît les derniers éléments. Ainsi `s[-10:-7]` va retourner les éléments des indices -10 inclus à -7 exclu, soit '`egg`'. Même si l'indication est négatif, la séquence est toujours parcourue de la gauche vers la droite. De même que précédemment, `s[:-3]` parcourt les éléments du début à l'indice -3 exclu pour donner '`egg, ba`'.

Pour ce qui concerne les pas, un pas négatif fait parcourir la séquence de la fin au début. C'est un moyen relativement pratique pour inverser une séquence, à savoir `s[::-1]` apporte '`nocab ,gge`'. Cette fois `s[2::-1]` inverse le parcours mais pas l'indice donc sélectionne les éléments d'indices 2 inclus à 0 exclu soit '`gg`' et `s[2::-1]` retourne la séquence en allant de deux jusqu'au début soit '`gge`'.

```
>>> s[-10:-7]
'egg'
>>> s[:-3]
'egg, ba'
>>> s[::-1]
'nocab ,gge'
>>> s[2::-1]
'gg'
>>> s[2::-1]
'gge'
```

2.1.2 Listes

Une liste est une séquence d'objets hétérogènes. Ce concept est donc à la fois souple et puissant. Une liste peut contenir n'importe quels types d'objets, néanmoins il est important de comprendre que

la liste ne stocke pas les objets en tant que tels mais les références vers ces objets. Par conséquent, la taille de l'objet liste est indépendante du type d'objets qui sont référencés.

Une liste est très malléable, on peut en augmenter la taille, la réduire, l'écartier au milieu pour rajouter des éléments à l'intérieur. C'est un objet dit *mutable*, c'est-à-dire qu'on peut le modifier en place, autrement dit le changer à l'endroit où il est stocké. L'avantage de cette mutabilité est que l'on a pas besoin de faire une copie de l'objet pour le modifier, ce qui est très efficace au niveau mémoire.

Une liste est une séquence, par suite toutes les opérations sur les séquences sont applicables aux listes : test d'appartenance, concaténation, les fonctions *built-in* `len`, `count`, `index`, etc.

Pour définir une liste, on utilise la notation entre crochets `a=[]` et `type(a)` retourne bien le type `list`. Une liste est composée d'éléments hétérogènes et même ceux référencé par une variable. On le vérifie en posant par exemple `i=4` puis `a=[i, 'spam', 3.2, True]`.

La liste est une séquence, on peut ainsi accéder à chacun de ses éléments, par exemple le premier `a[0]`. La liste étant un objet mutable, on peut également modifier un élément `a[0]=6`, voire directement faire une opération sur cet élément : `a[0]=a[0]+10`.

```
>>> a = []
>>> type(a)
<class 'list'>
>>> i = 4
>>> a = [i, 'spam', 3.2, True]
>>> a
[4, 'spam', 3.2, True]
>>> a[0]
4
>>> a[0] = 6
>>> a
[6, 'spam', 3.2, True]
>>> a[0] = a[0] + 10
>>> a
[16, 'spam', 3.2, True]
```

Là encore, comme la liste est une séquence, on peut appliquer les opérations de « *slicing* », par exemple `a[1:3]` retourne `['spam', 3.2]`.

On peut même faire des opérations d'affectation sur des *slices*. En posant `a[1:3]=[1,2,3]` on obtient `[16,1,2,3,True]`. L'affectation va effectuer deux opérations indépendantes. La première (`a[1:3]`) enlève tous les éléments d'indices 1 inclus à 3 exclu. La seconde opération (`[1,2,3]`) est d'insérer les éléments donnés à la place de ceux effacés. C'est un moyen très facile de supprimer des éléments; il suffit de les remplacer par une liste vide `a[1:3]=[]`. À noter que pour ce faire on peut aussi utiliser l'instruction `del a[1:2]`.

```
>>> a[1:3]
['spam', 3.2]
>>> a[1:3] = [1, 2, 3]
>>> a
[16, 1, 2, 3, True]
>>> a[1:3] = []
>>> a
[16, 3, True]
>>> del a[1:2]
>>> a
[16, True]
```

Les opérations sur les listes sont nombreuses. Pour les obtenir, on peut utiliser la fonction *built-in* `dir(list)` qui recense toutes les méthodes associées aux listes. Pour obtenir de l'aide sur une méthode particulière, il faut saisir `list.append?` avec l'interpréteur IPython et `help(list.append)` avec IDLE.

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__',
 '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
 '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy',
 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> help(list.append)
Help on method_descriptor:

append(self, object, /)
    Append object to the end of the list.
```

La méthode `append` rajoute un objet à la fin d'une liste comme suit `a.append('18')` pour la chaîne de caractère `18`. On peut également utiliser la méthode `extend` qui prend en entrée une séquence et va la rajouter à la fin de la liste : `a.extend([1, 2, 3])`. Le comportement est le même que faire un `append` sur chaque élément de la séquence.

La méthode `sort` trie une liste. Partant de `a=[1,5,3,1,7,9,2]`, on obtient `a=[1,1,2,3,5,7,9]`. Il faut bien noter que `sort` fonctionne en place, c'est-à-dire que la liste est triée sans copie temporaire et que la méthode ne retourne rien puisque la liste est triée en place.

```
>>> a.append('18')
>>> a
[16, True, '18']
>>> a.extend([1, 2, 3])
>>> a
[16, True, '18', 1, 2, 3]
>>> a = [1, 5, 3, 1, 7, 9, 2]
>>> a.sort()
>>> a
[1, 1, 2, 3, 5, 7, 9]
>>> a = a.sort()
>>> a
```

Une opération importante sur les listes est celle de passer une chaîne de caractère à une liste et vice-versa. C'est par exemple très utile dans la manipulation de fichiers traités avec PYTHON.

On part d'une chaîne de caractère, par exemple issue de la lecture d'un fichier, `s='spam egg beans'` que l'on souhaite afficher en colonnes. On emploie d'abord la fonction *built-in* `split` applicable aux chaînes de caractères `a=s.split()`. Le résultat est de découper la chaîne de caractère en utilisant l'espace comme séparateur. On obtient une liste qui contient trois éléments : `'spam'`, `'egg'` et `'beans'`.

La liste étant mutable, on peut tout à fait lui appliquer un traitement. Ici on souhaite mettre le premier élément en capitales : `a[0]=a[0].upper()`. Ensuite on peut de nouveau transcrire la liste en chaîne de caractères : `" ".join(a)`. La syntaxe signifie que l'on définit d'abord le séparateur (ici une espace) puis on juxtapose les éléments pour retrouver une chaîne de caractère avec le premier mot en lettres capitales.

```
>>> s = 'spam egg beans'
>>> a = s.split()
```

```
>>> a
['spam', 'egg', 'beans']
>>> a[0] = a[0].upper()
>>> a
['SPAM', 'egg', 'beans']
>>> " ".join(a)
'SPAM egg beans'
```

2.2 Compléments et application

2.2.1 Séquences, listes et *slicing*

Cette section reprend et complète ce qui a pu être abordé sur les *slices* en § 2.1.1. L'illustration est conduite sur la chaîne de caractères constituée des lettres de l'alphabet, mais est applicable aux listes aussi bien qu'aux tuples.

Une *slice* désigne toute une plage d'éléments d'une séquence. Les débutants ont parfois du mal avec les bornes. Il faut se souvenir que :

- les indices commencent comme toujours à zéro;
- le premier indice de début est inclus;
- le second indice de fin est exclu;
- on obtient en tout fin-debut items dans le résultat.

Ainsi, pour `chaine[2:6]`, le résultat contient $6-2 = 4$ éléments. Pour aider à mémoriser les conventions de début et de fin, on peut se rappeler qu'on doit pouvoir facilement juxtaposer deux *slices* qui ont une borne commune : `chaine[a:b] + chaine[b:c] == chaine[a:c]`

```
>>> chaine = "abcdefghijklmnopqrstuvwxyz"
>>> print(chaine)
abcdefghijklmnopqrstuvwxyz
>>> chaine[2:6]
'cdef'
>>> chaine[0:3] + chaine[3:7] == chaine[0:7]
True
```

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j
[x	x	x	[
				[x	x	x	x		
					[x	x	x	x	x

Slicing sur une séquence.

On peut omettre une borne dans une *slice* sur un objet. Trois cas de figure se présentent alors :

1. si on omet la première borne, la *slice* commence au début;
2. c'est pareil avec la fin de l'objet pour la seconde borne;
3. en ne spécifiant aucune borne, on fait une copie de l'objet.

On peut également utiliser des indices négatifs pour compter à partir de la fin de l'objet.

```
>>> chaine[:6]
'abcdef'
>>> chaine[24:]
'yz'
>>> chaine[:]
'abcdefghijklmnopqrstuvwxyz'
>>> chaine[3:-3]
'defghijklmnopqrstuvwxyz'
>>> chaine[-3:]
'xyz'
```

Il est possible de préciser un pas, de façon à ne choisir par exemple, dans la plage donnée, qu'un élément sur deux : `chaine[3:-3:2]`. Le pas est défini après un deuxième deux-points « `:` » ; ici on choisit un caractère sur deux dans l'intervalle `[3:-3]`. Comme on le devine, le troisième élément de la *slice*, ici 2, détermine le pas. On ne retient donc, dans la chaîne `defghi...` que `d`, puis `f` et ainsi de suite.

On peut préciser du coup la borne de fin (ici `-3`) avec un peu de liberté, puisqu'ici on obtiendrait un résultat identique avec `-4`.

Il est encore envisageable de spécifier un pas négatif. Dans ce cas, de manière un peu contre-intuitive, il faut préciser un début (le premier indice de la *slice*) qui soit plus à droite que la fin (le second indice).

Pour prendre un exemple, comme l'élément d'indice `-3`, c'est-à-dire `x`, est plus à droite que l'élément d'indice `3`, c'est-à-dire `d`. Évidemment si on ne précise pas le pas (qui revient à choisir un pas égal à `1`), on obtiendrait une liste vide. Si maintenant on précise un pas négatif, on obtient cette fois : `chaine[-3:3:-2]`.

```
>>> chaine[3:-3:2]
'dfhjlnprt'
>>> chaine[3:-4:2]
'dfhjlnprt'
>>> chaine[-3:3]
''
>>> chaine[-3:3:-2]
'xvtrpnljhf'
```

À nouveau, il faut se souvenir que ces mécanismes fonctionnent avec de nombreux autres types que les chaînes de caractères. En voici deux exemples — sur les listes et en utilisant la bibliothèque NumPy — qui anticipent tous les deux sur la suite, mais qui illustrent les vastes possibilités qui sont offertes avec les *slices*.

```
>>> liste = [1, 2, 4, 8, 16, 32]
>>> liste
[1, 2, 4, 8, 16, 32]
>>> liste[-1:1:-2]
[32, 8]
>>> liste[2:4] = [10, 20, 30]
>>> liste
[1, 2, 10, 20, 30, 16, 32]
```

La figure 9.4 expose une représentation imagée de ce qui se passe lorsqu'on exécute cette dernière ligne de code `liste[2:4]=[10,20,30]` ; cela revient en quelque sorte à remplacer la *slice* à gauche de l'affectation (ici `liste[2:4]`) par la liste à droite de l'affectation (ici `[10,20,30]`) — ce qui a pour effet de modifier la longueur de la liste.

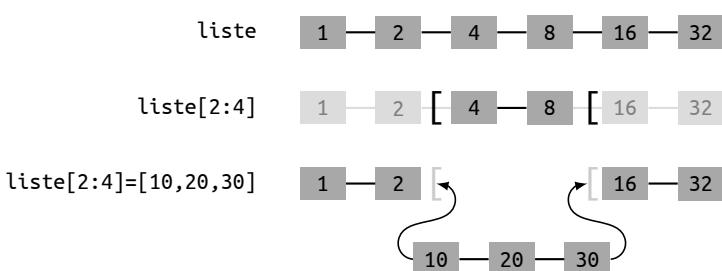


FIGURE 9.4 — Insertion d'éléments dans une liste avec une slice.

La bibliothèque NumPy permet de manipuler des tableaux ou des matrices de manière optimale. En anticipant (beaucoup) sur son usage (visité en détail par la suite), on présente ici un aperçu du potentiel des *slices* sur des objets NumPy.

```
>>> import numpy as np # Importation de la bibliothèque
>>> un_cinq = np.array([1, 2, 3, 4, 5]) # Construction d'un tableau ligne
>>> un_cinq
```

```

array([1, 2, 3, 4, 5])
>>> array = 10 * un_cinq[:, np.newaxis] + un_cinq # On le combine avec lui-même en utilisant une slice
>>> array
                                         # un peu magique pour former un tableau carré 5x5
array([[11, 12, 13, 14, 15],
       [21, 22, 23, 24, 25],
       [31, 32, 33, 34, 35],
       [41, 42, 43, 44, 45],
       [51, 52, 53, 54, 55]])
>>> centre = array[1:4, 1:4] # Sur ce tableau de taille 5x5, on peut aussi faire
>>> centre
                                         # du slicing et extraire le sous-tableau 3x3 au centre
array([[22, 23, 24],
       [32, 33, 34],
       [42, 43, 44]])
>>> coins = array[::-4, ::4] # On peut bien sûr également utiliser un pas
>>> coins
array([[11, 15],
       [51, 55]])
>>> tete_en_bas = array[::-1,:]
>>> tete_en_bas
array([[51, 52, 53, 54, 55],
       [41, 42, 43, 44, 45],
       [31, 32, 33, 34, 35],
       [21, 22, 23, 24, 25],
       [11, 12, 13, 14, 15]])

```

2.2.2 Méthodes spécifiques aux listes

Pour se remémorer les méthodes applicables aux listes, il faut saisir `help(list)` dans un interpréteur PYTHON. Dans un premier temps on peut ignorer les méthodes qui débutent et se terminent par un double underscore, restent alors les méthodes comprises entre `append` et `sort`: `clear`, `copy`, `count`, `extend`, `index`, `insert`, `pop`, `remove`, `reverse`.

Certaines de ces méthodes ont été vues en § 2.1.1 sur les séquences, c'est le cas notamment de `count` et `index`. Il s'agit à présent de faire état des autres méthodes.

```

>>> help(list)
Help on class list in module builtins:

class list(object)
| list(iterable=(), /)
|
| Built-in mutable sequence.
|
| If no argument is given, the constructor creates a new empty list.
| The argument must be an iterable if specified.
|
| Methods defined here:
|
|   __add__(self, value, /)
|     Return self+value.
|
|   __contains__(self, key, /)
|     Return key in self.
|
|   __delitem__(self, key, /)
|     Delete self[key].
|
|   __eq__(self, value, /)
|     Return self==value.

```

```
| __ge__(self, value, /)
|     Return self>=value.

| __getattribute__(self, name, /)
|     Return getattr(self, name).

| __getitem__(...)
|     x.__getitem__(y) <==> x[y]

| __gt__(self, value, /)
|     Return self>value.

| __iadd__(self, value, /)
|     Implement self+=value.

| __imul__(self, value, /)
|     Implement self*=value.

| __init__(self, /, *args, **kwargs)
|     Initialize self. See help(type(self)) for accurate signature.

| __iter__(self, /)
|     Implement iter(self).

| __le__(self, value, /)
|     Return self<=value.

| __len__(self, /)
|     Return len(self).

| __lt__(self, value, /)
|     Return self<value.

| __mul__(self, value, /)
|     Return self*value.

| __ne__(self, value, /)
|     Return self!=value.

| __repr__(self, /)
|     Return repr(self).

| __reversed__(self, /)
|     Return a reverse iterator over the list.

| __rmul__(self, value, /)
|     Return value*self.

| __setitem__(self, key, value, /)
|     Set self[key] to value.

| __sizeof__(self, /)
|     Return the size of the list in memory, in bytes.

| append(self, object, /)
|     Append object to the end of the list.

| clear(self, /)
|     Remove all items from list.

| copy(self, /)
```

```

| Return a shallow copy of the list.

| count(self, value, /)
|     Return number of occurrences of value.

| extend(self, iterable, /)
|     Extend list by appending elements from the iterable.

| index(self, value, start=0, stop=9223372036854775807, /)
|     Return first index of value.

| Raises ValueError if the value is not present.

| insert(self, index, object, /)
|     Insert object before index.

| pop(self, index=-1, /)
|     Remove and return item at index (default last).

| Raises IndexError if list is empty or index is out of range.

| remove(self, value, /)
|     Remove first occurrence of value.

| Raises ValueError if the value is not present.

| reverse(self, /)
|     Reverse *IN PLACE*.

| sort(self, /, *, key=None, reverse=False)
|     Sort the list in ascending order and return None.

The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
order of two equal elements is maintained).

If a key function is given, apply it once to each list item and sort them,
ascending or descending, according to their function values.

The reverse flag can be set to sort in descending order.

-----
| Static methods defined here:

| __new__(*args, **kwargs) from builtins.type
|     Create and return a new object. See help(type) for accurate signature.

-----
| Data and other attributes defined here:

| __hash__ = None

```

Pour commencer, donnons-nous une liste témoin `liste=[0,1,2,3]` à laquelle appliquer le différentes méthodes :

- la méthode `append` ajoute un élément à la fin d'une liste :
- la méthode `extend` réalise la même opération, mais avec *tous les éléments* de la liste qu'on lui passe en argument.

```

>>> liste = [0, 1, 2, 3]
>>> print('liste :', liste)
liste : [0, 1, 2, 3]
>>> liste.append('ap')
>>> print('liste', liste)

```

```

liste [0, 1, 2, 3, 'ap']
>>> liste2 = ['ex1', 'ex2']
>>> liste.extend(liste2)
>>> print('liste', liste)
liste [0, 1, 2, 3, 'ap', 'ex1', 'ex2']

```

Ces deux méthodes `append` et `extend` sont donc assez voisines.

Avant de voir d'autres méthodes de `list`, prenons le temps de comparer leur comportement avec l'addition de liste « + ». L'élément clé ici est que la liste est un objet *mutable*. Ainsi, `append` et `extend` modifient la liste sur laquelle elles travaillent, alors que l'addition crée un nouvel objet. C'est pour cette raison que :

- l'addition est disponible sur tous les types séquences — on peut toujours réaliser l'addition puisqu'on crée un nouvel objet pour stocker le résultat de l'addition;
- mais `append` et `extend` ne sont par exemple pas disponibles sur les chaînes de caractères, qui sont immuables.

```

>>> a1 = list(range(3)) # Pour créer une liste avec les n premiers entiers, on utilise
>>> print(a1)           # la fonction built-in range(), que l'on convertit en liste
[0, 1, 2]
>>> a2 = list(range(10, 13))
>>> print(a2)
[10, 11, 12]
>>> a3 = a1 + a2      # Le fait d'utiliser "+" crée une nouvelle liste
>>> print('a1', a1) # si bien que maintenant on a trois objets différents
a1 [0, 1, 2]
>>> print('a2', a2) # Comme on le voit, après une addition, les deux termes de l'addition sont inchangés.
a2 [10, 11, 12]
>>> print('a3', a3)
a3 [0, 1, 2, 10, 11, 12]

```

On reprend l'inventaire des méthodes de `list` avec la méthode `insert`. Comme son nom le suggère, elle permet d'insérer un élément à une certaine position (comme toujours les indices commencent à zéro). On peut remarquer qu'un résultat analogue peut être obtenu avec une affectation de `slice`; par exemple pour une insertion au rang 5 ici.

```

>>> print('liste :', liste) # Rappel du contenu de "liste"
liste : [0, 1, 2, 3, 'ap', 'ex1', 'ex2']
>>> liste.insert(2, '1 bis') # Insertion à l'index 2
>>> print('liste', liste)
liste [0, 1, '1 bis', 2, 3, 'ap', 'ex1', 'ex2']
>>> liste[5:5] = ['3 bis']
>>> print('liste', liste)
liste [0, 1, '1 bis', 2, 3, '3 bis', 'ap', 'ex1', 'ex2']

```

Quant à elle, la méthode `remove` détruit la première occurrence d'un objet dans la liste.

```

>>> liste.remove(3)
>>> print('liste :', liste)
liste : [0, 1, '1 bis', 2, '3 bis', 'ap', 'ex1', 'ex2']

```

La méthode `pop` prend en argument un indice; elle permet d'extraire l'élément ayant cet indice. En un seul appel on obtient la valeur de l'élément et on l'enlève de la liste. Si l'indice n'est pas précisé, c'est le dernier élément de la liste qui est visé.

```

>>> popped = liste.pop(0)
>>> print('popped :', popped, ' ; liste :', liste)
popped : 0  ; liste : [1, '1 bis', 2, '3 bis', 'ap', 'ex1', 'ex2']

```

```
>>> popped = liste.pop()
>>> print('popped :', popped, ' ; liste :', liste)
popped : ex2 ; liste : [1, '1 bis', 2, '3 bis', 'ap', 'ex1']
```

Enfin, la méthode `reverse`, comme son nom l'indique, renverse la liste; le premier élément devient le dernier. On peut alors remarquer ici que le résultat se rapproche de ce qu'on peut obtenir avec une opération de *slicing* comme suit. À la différence toutefois qu'avec le *slicing* c'est une copie de la liste initiale qui est retournée, la liste de départ quant à elle n'est pas modifiée.

```
>>> liste.reverse()
>>> print('liste :', liste)
liste : ['ex1', 'ap', '3 bis', 2, '1 bis', 1]
>>> liste2 = liste[::-1]
>>> print('liste2', liste2)
liste2 [1, '1 bis', 2, '3 bis', 'ap', 'ex1']
```

POUR EN SAVOIR PLUS
Documentation officielle sur les listes

2.2.3 Mutabilité et immuabilité des objets

Voici ci-dessous un fragment de code qui illustre le caractère immuable des chaînes de caractères. Le scénario est très simple, on crée deux variables `s1` et `s2` vers le même objet '`abc`', puis on fait une opération `+=` sur la variable `s1`.

Comme l'objet est une chaîne, il est donc immuable, on ne peut pas modifier l'objet directement; pour obtenir l'effet recherché (à savoir que `s1` s'allonge de '`def`'), PYTHON crée un deuxième objet.

```
>>> s1 = 'abc' # Deux variables vers le même objet
>>> s2 = s1
>>> s1 += 'def' # On essaie de modifier l'objet
>>> print(s1)
abcdef
>>> print(s2)
abc
```

Les listes sont des objets mutables. Voici par contraste ce qu'on obtient pour le même scénario mais qui, cette fois, utilise des listes.

```
>>> liste1 = ['a', 'b', 'c'] # Deux variables vers le même objet
>>> liste2 = liste1
>>> liste1 += ['d', 'e', 'f'] # On modifie l'objet
>>> print(liste1)
['a', 'b', 'c', 'd', 'e', 'f']
>>> print(liste2)
['a', 'b', 'c', 'd', 'e', 'f']
```

5

Ce comportement n'est pas propre à l'usage de l'opérateur `+=`, les objets mutables et immuables ont par essence un comportement différent, il est très important d'avoir ceci présent à l'esprit.

2.2.4 Tris de listes

PYTHON fournit une méthode standard pour trier une liste, qui s'appelle, sans grande surprise, `sort`.

```
>>> liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
>>> print('Avant le tri : ', liste)
Avant le tri :  [8, 7, 4, 3, 2, 9, 1, 5, 6]
>>> liste.sort()
```

```
>>> print('Après le tri : ', liste)
Après le tri : [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

On retrouve ici, avec l'instruction `liste.sort()` un cas d'appel de méthode (ici `sort`) sur un objet (ici `liste`). La première chose à remarquer est que la liste d'entrée a été modifiée — on dit « en place » ou « par effet de bord ». On aurait pu imaginer que la liste d'entrée soit restée inchangée et que la méthode de tri renvoie une copie triée de la liste, ce n'est pas le choix qui a été fait en PYTHON ; cela permet d'économiser des allocations mémoire autant que possible et d'accélérer sensiblement le tri.

En revanche, si on a besoin de faire le tri sur une copie de liste, la fonction `sorted` permet de le réaliser.

```
>>> liste1 = [3, 2, 9, 1]
>>> liste2 = sorted(liste1)
>>> print('Après le tri de la copie : ', liste2)
Après le tri de la copie : [1, 2, 3, 9]
```

Revenons à la méthode `sort` et aux *tris en place*. Par défaut la liste est triée par ordre croissant, si au contraire on souhaite l'ordre décroissant, on peut faire comme suit.

```
>>> liste = [8, 7, 4, 3, 2, 9, 1, 5, 6]
>>> print('Avant le tri :', liste)
Avant le tri : [8, 7, 4, 3, 2, 9, 1, 5, 6]
>>> liste.sort(reverse=True)
>>> print('Après le tri décroissant : ', liste)
Après le tri décroissant : [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

On a pas encore vu à quoi correspond cette formule `reverse=True` dans l'appel à la méthode — ceci est approfondi dans le chapitre sur les appels de fonction — mais dans l'immédiat on peut utiliser cette technique telle quelle.

Cette technique fonctionne très bien sur tous les types numériques à l'exception des complexes²⁰, ainsi que sur les chaînes de caractères.

Pour les chaînes de caractères, comme on s'y attend, il s'agit cette fois d'un *tri lexicographique*, dérivé de l'ordre sur les caractères. Autrement dit, c'est l'ordre du dictionnaire. Il faut souligner toutefois, pour les personnes n'ayant jamais été exposées à l'informatique, que cet ordre, quoique déterministe, est arbitraire en dehors des lettres de l'alphabet. Lorsque les chaînes contiennent des espaces ou autres ponctuations, le résultat du tri peut paraître surprenant.

```
>>> liste = ['spam', 'egg', 'bacon', 'beef']
>>> liste.sort()
>>> print('Après le tri : ', liste)
Après le tri : ['bacon', 'beef', 'egg', 'spam']

>>> 'a' < 'z' # Deux caractères minuscules se comparent selon l'ordre lexicographique
True

>>> 'Z' < 'a' # Si l'un est en minuscule et l'autre en majuscule, ce n'est plus le cas
True

>>> liste = ['abc', 'Zoo'] # Conséquence de 'Z' < 'a'
>>> liste.sort()
>>> print(liste)
['Zoo', 'abc']

>>> liste = [' zoo', 'ane'] # Attention ici la première chaîne commence par une espace
```

```
>>> liste.sort()          # et le caractère 'Espace' est plus petit
>>> print(liste)         # que tous les autres caractères imprimables
[' zoo', 'ane']
```

3 Conditionnelles, itérations et fonctions

Jusqu'ici les notions importantes abordées sont les concepts d'objet, de variable, de typage dynamique, mais aussi les types centraux comme les chaînes de caractères ou les listes et encore les opérations sur les séquences. Toutefois, cela ne suffit pas pour écrire de vrais programmes. Il faut en effet également maîtriser les tests, les boucles et les fonctions « Langages de l'informatique » § 1.1.3.

3.1 Test if-else

3.1.1 Utilisation

Le test **if-else** permet de faire de l'exécution conditionnelle, c'est-à-dire qu'un morceau de code va s'effectuer selon que le résultat d'un test est vrai ou faux.

On peut ainsi tester toute expression qui retourne un booléen comme (voir récapitulation partielle au [table 9.2](#)) les relations d'ordre, l'égalité, la différence ou les tests d'appartenance.

Supposons que l'on veuille afficher un message pour toute note supérieure à 10/20. On commence par définir une variable **note**. La syntaxe de l'instruction **if** est alors comme suit.

```
>>> note = 8
>>> if note > 10:
...     print('Reçu')
...     print('Bravo')
... else:
...     print('Recalé')
...
Recalé
```

On peut remarquer que l'on a systématiquement un deux-points « `:` » avant chaque bloc d'instruction. Un bloc d'instruction est une ensemble d'instructions qui sont toutes indentées, par convention, de quatre espaces vers la droite.

Ainsi, si le test est vrai on entre dans le premier bloc d'instruction, s'il est faux on se retrouve dans le bloc d'instruction de la clause **else**, également repérée par un deux-points.

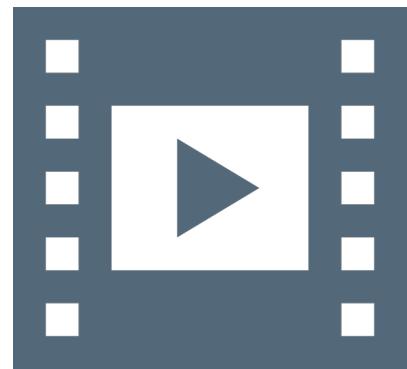
3.1.2 Blocs d'instruction et syntaxe

PYTHON est conçu autour du concept de bloc d'instruction. Dans d'autres langages, il n'y a pas cette notion et les blocs d'instructions sont explicitement délimités par des séparateurs comme les accolades.

Reprenons le même test, mais avec une syntaxe « à la JAVA ». Toutes les instructions finissent par un point-virgule pour déterminer leur fin et les blocs d'instructions sont eux délimités par des accolades.

```

note = 8;
if note > 10 {
    System.out.print("Reçu");
    System.out.print("Bravo !");
}
```



VIDÉO 9.6 — Conditionnelle.



TABLE 9.2 — Expressions applicables aux conditionnelles.

Relation	Expression
Supérieur	<code>a > b</code>
Supérieur ou égal	<code>a >= b</code>
Inférieur	<code>a < b</code>
Inférieur ou égal	<code>a <= b</code>
Égalité	<code>a == b</code>
Déférence	<code>a != b</code>
Appartenance	<code>a in s</code>
Absence	<code>a not in s</code>

```
else
{ System.out.print("Recalé"); }
```

Ce principe s'avère un problème connu en programmation puisque pour être capable de savoir où placer les accolades, on définit ce que l'on appelle des conventions de codage. La convention n'a aucun impact sur l'exécution du code, mais elle est adoptée pour faciliter l'écriture et la lecture du code.

Or, ces conventions de codage font partie d'écoles de programmation et certaines personnes préfèrent par exemple mettre les accolades en début et fin de ligne ou aligner les accolades sur les instructions. En fait, lire un code écrit avec une convention qui n'est pas la sienne rend extrêmement difficile l'interprétation ou la lecture de ce code.

En PYTHON, ce problème n'existe pas puisque la convention de codage fait partie de la syntaxe. Si on ne respecte pas la convention, le code ne s'exécute pas en indiquant une erreur de syntaxe. L'avantage est qu'il n'y a qu'une seule manière de présenter le code.

Seul est conservé dans la syntaxe un deux-point « : » précédant les blocs d'instructions. Il est issu de tests utilisateurs qui ont démontré que cela rendait plus aisée la lecture du code.

Le fait que PYTHON utilise l'indentation comme base de sa syntaxe n'a presque que des avantages, cela donne un code : toujours écrit de la même manière, bien présenté, facile à lire et à écrire. Il existe cependant un seul inconvénient, que cette syntaxe fondée sur l'indentation supporte mal le copier-coller. Dans ce cas de figure, la recommandation est naturellement de bien vérifier ce point lorsque l'on récupère des bouts de codes sur Internet ou depuis d'autres programmes.

Cette convention de codage qui décale les blocs d'instructions de quatre espaces vers la droite a tendance à créer des lignes un peu longues. Il est préconisé ici de ne pas dépasser 79 caractères par ligne de code. Même si aujourd'hui les écrans sont grands, l'idée est de pouvoir juxtaposer plusieurs éditeurs de texte. Quoi qu'il en soit, il est plus facile de lire des lignes courtes.

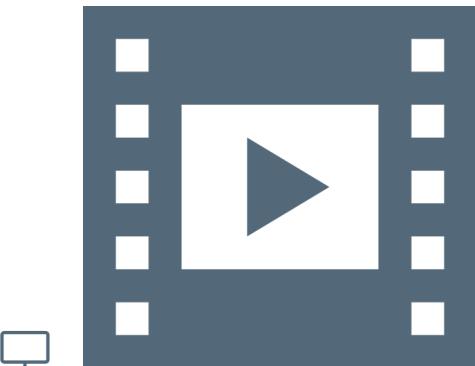
Par ailleurs, PYTHON permet de retourner sans peine à la ligne car tous ce qui est entre parenthèses, entre crochets ou entre accolades supportent le retour chariot sans générer des problèmes dans la syntaxe de PYTHON. Cela autorise une présentation du code bien lisible lorsque par exemple, on a une grande liste (crochets) ou un nombre important d'expressions (parenthèses).

3.2 Factorisation de code

Un principe fondateur de la programmation est ce qu'on appelle la *factorisation de code*. C'est ce qui permet de ne pas réécrire plusieurs fois un code qui fait la même chose. L'intérêt est évidemment de faciliter la maintenance et surtout la qualité du code. Sont abordées ici les notions de boucle **for** et les fonctions qui sont deux techniques traditionnelles de factorisation.

3.2.1 Boucle for

Supposons que l'on veuille afficher à l'écran les carrés des nombres allant de 0 à 9. On peut écrire `print(1**2)`, `print(2**2)`, `print(3**2)` et ainsi de suite... En continuant de cette manière, on se rend compte rapidement que c'est une tâche très fastidieuse. On peut alors automatiser cette tâche au moyen d'une boucle **for**.



VIDÉO 9.7 – Boucle *for*.

BLOC-NOTE VIDÉO 9.7 (copier-coller)

```
print(1**2)
print(2**2)
print(3**2)
for i in range(10):
    print(i**2)
for i in ['a', 3.5, True]:
    print(i)
a = [1, 2, 5, 8, 9]
for i in a:
    print(i**2)
b = [3.6, 18, 12, 25]
for i in b:
    print(i**2)
def carre(a):
    for i in a:
        print(i**2)
carre(a)
carre(b)
def carre(a):
    L = []
    for i in a:
        L.append(i**2)
    return L
carre(b)
b = carre(b)
```

La syntaxe d'une boucle `for` est donnée ci-dessous où `i` est une variable et `range(10)` est objet qui permet de parcourir tous les entiers allant de `0` à `9`. Avec un deux-point, la fin de ligne annonce un bloc d'instruction — ici `print(i**2)` —, qui débute à la ligne suivante décalée de quatre espaces vers la droite. La variable `i` référence chacun des entiers à chaque tour de boucle et `print(i**2)` affiche le carré de `i`.

```
>>> for i in range(10):
...     print(i**2)
...
0
1
4
9
16
25
36
49
64
81
```

On peut mettre n'importe quelle séquence voire itérable²¹ dans une boucle `for`. On peut par exemple considérer une liste contenant une chaîne caractère, un `float` et un booléen puis afficher simplement la variable `i` d'itération pour avoir chaque élément de la liste.

```
>>> for i in ['a', 3.5, True]:
...     print(i)
...
a
3.5
True
```

²¹. Concept abordé par la suite.

3.2.2 Fonctions

Reprendons l'exemple de l'affichage du carré et supposons que l'on ait deux listes `a=[1,2,5,8,9]` et `b=[3.6,18,12,25]`. Pour avoir leur carrés respectifs, on peut reprendre la démarche précédente avec deux boucles `for` distinctes. Là encore, c'est un peu fastidieux de réécrire deux fois la même boucle `for`. Une manière de factoriser ce code est d'utiliser ce que l'on appelle une fonction.

A. DÉFINITION

Une fonction est un morceau de code que l'on peut rappeler n'importe quand. Comme pour un test et une boucle, une fonction débute par le mot-clef `def`, suivi de son nom — ici `carre` — et des arguments d'entrée, pour terminer la ligne par deux-points indiquant un bloc d'instruction à suivre. Une fois définie, la fonction peut être appelé n'importe où dans le code.

```
>>> a = [1, 2, 5, 8, 9]
>>> for i in a:
...     print(i**2)
...
1
4
25
64
81
>>> b = [3.6, 18, 12, 25]
>>> for i in b:
...     print(i**2)
```

```

...
12.96
324
144
625
>>> def carre(x):
...     for i in x:
...         print(i**2)
...
>>> carre(a)
1
4
25
64
81
>>> carre(b)
12.96
324
144
625

```

B. RETOUR DE FONCTION

En pratique, avec une fonction on fait rarement des affichages avec des `print()`; en revanche l'objectif est plutôt d'obtenir un retour. L'instruction `return` permet le retour d'un traitement par une fonction. Dans notre exemple, il s'agit de la liste du carré des entiers qui se définit par une liste initiale vide à laquelle on ajoute à chaque tour de boucle le carré de la variable d'itération, avant de renvoyer la liste complète.

```

>>> def carre(a):
...     L = []
...     for i in a:
...         L.append(i**2)
...     return L
...
>>> carre(b)
[12.96, 324, 144, 625]
>>> b = carre(b)
>>> b
[12.96, 324, 144, 625]

```

3.3 Compléments et application

3.3.1 Bonnes pratiques

A. INDENTATION EN PYTHON

La pratique la plus courante est d'utiliser systématiquement une indentation de quatre espaces.

```

>>> if 'g' in 'egg': # Convention d'indentation à quatre espaces
...     print('OUI')
... else:
...     print('NON')
...
OUI

```

Pour écrire plusieurs tests imbriqués, il est donc nécessaire d'avoir une indentation de huit espaces.

```

>>> entrée = 'spam'
>>> if 'a' in entrée:

```

```

...
if 'b' in entree: # Imbrication à huit espaces
    cas11 = True
    print('a et b')
else:
    cas12 = True
    print('a mais pas b')
else:
    if 'b' in entree:
        cas21 = True
        print('b mais pas a')
    else:
        cas22 = True
        print('ni a ni b')
...
a mais pas b

```

Cette façon d'organiser le code peut paraître très étrange, notamment aux gens habitués à un autre langage de programmation, puisqu'en général les syntaxes des langages sont conçues de manière à être insensibles aux espaces et à la présentation.

Cependant, après s'y être familiarisé, cette pratique est très agréable. Par exemple, une fois qu'on a écrit la dernière ligne du code, on n'a pas à réfléchir à refermer le bon nombre d'accolades ou de « `end` ». Par ailleurs, comme pour tous les langages, les éditeurs connaissent cette syntaxe et aident à respecter la règle des quatre caractères d'espace.

Il faut par contre apporter quelques détails sur un problème fréquemment rencontré avec du code partagé entre plusieurs personnes, quand celles-ci utilisent des environnements différents.

6

Pour faire court, ce problème est susceptible d'apparaître dès qu'on utilise des *tabulations*, plutôt que des espaces, pour implémenter les indentations. Aussi, le message à retenir ici est de *ne jamais utiliser de tabulations dans un code PYTHON*. Tout bon éditeur devrait faire cela par défaut.

En version longue, il existe un code ASCII pour un caractère qui s'appelle Tabulation (alias `Ctrl+I`, qu'on note aussi `^I`) ; l'interprétation de ce caractère n'étant pas clairement spécifiée, il arrive qu'on se retrouve dans une situation comme la suivante.

Un utilisateur code avec l'éditeur VIM sous lequel il est possible de mettre des tabulations dans son code et de choisir la valeur de ces tabulations. Aussi il va dans les préférences, choisit Tabulation=4 et écrit un programme qu'il voit comme ceci :

```

1 if 'a' in entree:
2     if 'b' in entree:
3         cas11 = True
4         print('a et b')
5     else:
6         cas12 = True
7         print('a mais pas b')

```

Sauf qu'en fait, il y a mis un mélange de tabulations et d'espaces, le fichier se présente comme suit (avec `^I` pour tabulation).

```

1 if 'a' in entree:
2 ^If 'b' in entree:
3 ^I^Icas11 = True
4     ^Iprint('a et b')

```

```

5 ^Else:
6 ^I^Icas12 = True
7     ^Iprint('a mais pas b')

```

On peut remarquer le mélange de tabulations et d'espaces dans les deux lignes avec `print`. L'utilisateur envoie alors son code à un collègue qui utilise EMACS. Dans son environnement, EMACS affiche une tabulation comme 8 caractères. Du coup son collègue « voit » le code suivant :



```

1 if 'a' in entree:
2     if 'b' in entree:
3         cas11 = True
4         print('a et b')
5     else:
6         cas12 = True
7         print('a mais pas b')

```

Bref, c'est la confusion la plus totale. Aussi répétons-le, *n'utilisez jamais de tabulations* dans un code PYTHON.

Cela ne veut pas dire qu'il ne faut pas utiliser la touche `Tab` avec un éditeur — au contraire, c'est une touche très pratique —, mais il faut bien faire la différence entre le fait d'appuyer sur la touche `Tab` et le fait que le fichier sauve sur disque contient effectivement un caractère de tabulation. Les bons éditeurs proposent très certainement une option permettant de faire les remplacements idoines pour ne pas écrire de tabulation dans les fichiers, tout en permettant d'indenter le code correctement avec la touche `Tab`.

À cet égard, signalons enfin que PYTHON 3 est plus restrictif que PYTHON 2 et interdit de mélanger des espaces et des tabulations sur une même ligne. Ce qui n'enlève rien à la recommandation faite.

B. EXPLICATION DE LA RÈGLE D'INDENTATION

Pour les curieux, PYTHON n'impose pas que les indentations soient de quatre caractères. Aussi il est possible de rencontrer un code qui ne respecte pas cette convention et il faut, pour être tout à fait précis sur ce que PYTHON accepte ou non, expliciter ce qui est réellement requis par PYTHON.

La règle utilisée pour analyser un code, c'est que toutes les instructions d'un même bloc soient présentées avec le même niveau d'indentation. Si deux lignes successives — modulo les blocs imbriqués — ont la même indentation, elles sont dans le même bloc. Bien que légal, le code à suivre n'est pas recommandé.

```

>>> # Code accepté mais pas du tout recommandé
>>> if 'a' in 'pas du tout recommandé':
...     succes = True
...     print('OUI')
... else:
...     print('NON')
...
OUI

```

En effet, les deux blocs (après `if` et après `else`) sont des blocs distincts, ils sont libres d'utiliser deux indentations différentes (ici 2 et 6). A contrario, la construction ci-dessous n'est pas légale.

```

>>> # Ceci est incorrect et rejeté par Python
>>> if 'a' in entree:
...     if 'b' in entree:
...         cas11 = True

```

```

...         print('a et b')
...     else:
File "<stdin>", line 5
    else:
        ^
IndentationError: unindent does not match any outer indentation level
>>>     cas12 = True
File "<stdin>", line 1
    cas12 = True
    ^
IndentationError: unexpected indent
>>>     print('a mais pas b')
File "<stdin>", line 1
    print('a mais pas b')
    ^
IndentationError: unexpected indent

```

Ici les deux lignes `if` et `else` font logiquement partie du même bloc, elles doivent donc avoir la même indentation. Avec cette présentation le lecteur PYTHON émet une erreur et ne peut pas interpréter le code.

C. PRÉSENTATION DE CODE

On trouve dans la PEP-008 (en anglais) les conventions de codage qui s'appliquent à toute la bibliothèque standard et qui sont certainement un bon point de départ pour aider à trouver le style de présentation qui convient à chacun. Les sections plus particulièrement conseillées sont : l'indentation; les espaces; les commentaires.

ILLUSTRATION PAR L'EXEMPLE — On peut prendre en exemple le code du module `pprint` (comme *PrettyPrint*) de la bibliothèque standard ( `lib/python3.8/pprint.py`) qui permet d'imprimer des données de manière élaborée.



```

1 from modtools import show_module_html
2 import pprint
3 show_module_html(pprint)

```

La fonction du module — le *pretty printing* — est évidemment accessoire ici, mais on peut y voir illustré :

- le *docstring* pour le module aux lignes 11 à 35;
- les indentations, comme déjà mentionné à quatre espaces et sans aucune tabulation;
- l'utilisation des espaces, autour des affectations et opérateurs, des définitions de fonction, des appels de fonctions, etc.;
- les lignes de largeur « raisonnable » (79 caractères);
- la coupure des lignes pour respecter cette limite en largeur.

ESPACES — Comme on peut le voir dans `pprint.py`, les règles principales concernant les espaces sont les suivantes.

- ▶ S'agissant des affectations et opérateurs, on saisira `x = y + z`, non pas `x=y+z`, ni `x = y+z`, ni encore `x=y + z`. L'idée étant d'aérer de manière homogène pour faciliter la lecture.
- ▶ Pour les déclarations de fonction on entrera `def foo(x, y, z):` et non pas avec un espace en trop avant la parenthèse ouvrante `def foo (x, y, z):`, ni surtout sans espace entre les arguments `def foo(x,y,z):`

Il est important de noter qu'il s'agit ici de *règles d'usage* et non pas de règles syntaxiques; tous les exemples barrés ci-dessus sont en fait *syntaxiquement corrects*, l'interpréteur les accepterait sans souci; mais ces règles sont très largement adoptées et obligatoires pour intégrer du code dans la bibliothèque standard.

NOTE DE LA RÉDACTION

Bien qu'ayant installé le module `modtools`, il n'a pas été possible de faire appel à la fonction `show_module_html`. Cela fonctionne dans le Mooc initial avec *notebook*. Aussi le code source est directement consultable dans le répertoire de la bibliothèque standard  `python3.8/pprint.py` pour Ubuntu MATE.

COUPURES DE LIGNE — On va à présent zoomer dans le module `pprint.py` pour voir quelques exemples de coupure de ligne. Par contre avec ce qui précède, il s'agit ici surtout de règles syntaxiques, qui peuvent rendre un code non valide si elles ne sont pas respectées.

La fonction `pprint` (ligne 47) est une commodité (qui crée une instance de `PrettyPrinter`, sur lequel on envoie la méthode `pprint`). On voit là qu'il n'est pas nécessaire d'insérer un *backslash* (\) à la fin des lignes 50 et 51, car il y a une parenthèse ouvrante qui n'est pas fermée à ce stade.

De manière générale, lorsqu'une parenthèse ouvrante « (» — *idem* avec les crochets « [» et accolades « { » — n'est pas fermée sur la même ligne, l'interpréteur suppose qu'elle sera fermée plus loin et n'impose pas de *backslash*. On peut donc écrire :



```

1 valeurs = [
2     1,
3     2,
4     3,
5     5,
6     7,
7 ]
8 x = un_nom_de_fonction_tres_très_long(
9     argument1, argument2,
10    argument3, argument4,
11 )

```

À titre de rappel, signalons aussi les chaînes de caractères à base de """ ou ''' qui permettent elles aussi d'utiliser plusieurs lignes consécutives sans *backslash*.



```

1 texte = """Les sanglots longs
2 Des violons
3 De l'automne"""

```

Néanmoins, il existe des cas où le *backslash* est nécessaire. On peut l'illustrer par la ligne 536 du fichier `pprint.py`.

Dans tous les cas où une instruction est répartie sur plusieurs lignes, c'est naturellement l'indentation de la première ligne qui est significative afin de savoir à quel bloc rattacher cette instruction.

Bien noter enfin qu'on peut toujours mettre un *backslash* même lorsque ce n'est pas nécessaire, mais cette pratique est en règle générale évitée car les *backslash* nuisent à la lisibilité.

Il existe plusieurs outils liés à la PEP0008, pour vérifier si un code est conforme, ou même le modifier pour qu'il le devienne.

Cela donne un excellent prétexte pour parler un peu du site <https://pypi.python.org>, qui est la plateforme qui distribue les logiciels disponibles via l'outil `pip3` — acronyme récursif pour *Pip Installs Python* ou *PIP Installs Packages*. On peut notamment signaler ici :

- l'outil `pep8` pour vérifier;
- l'outil `autopep8` pour modifier automatiquement un code et le rendre conforme aux conventions d'usage.

Dans un autre registre, on peut se reporter à ce [lien](#) afin de savoir pourquoi on a choisi un délimiteur (le caractère deux-points « : ») pour terminer les instructions comme `if`, `for` et `def`.

D. *INSTRUCTION pass*

En PYTHON les blocs de code sont définis par leur indentation. Cette convention a une limitation lorsqu'on essaie de définir un bloc vide.

Voyons par exemple comment on définirait en C une fonction qui ne fait absolument rien :

C

```
1 /* une fonction C qui ne fait rien */
2 void foo() {}
```

Comme en PYTHON on n'a pas d'accolade pour délimiter les blocs de code, il existe une instruction `pass`, qui ne fait rien. À l'aide de cette instruction on peut à présent définir une fonction vide comme ci-dessous

```
1 # une fonction Python qui ne fait rien
2 def foo():
3     pass
```

Pour prendre un second exemple un peu plus pratique et pour anticiper un peu sur l'instruction `while` abordé plus loin dans le cours, voici un exemple d'une boucle vide, c'est à dire sans corps, qui permet de « dépiler » dans une liste jusqu'à l'obtention d'une certaine valeur. On voit qu'ici encore l'instruction `pass` a toute son utilité.

```
>>> liste = list(range(10))
>>> print('Avant : ', liste)
Avant : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> while liste.pop() != 5:
...     pass
...
>>> print('Après : ', liste)
Après : [0, 1, 2, 3, 4]
```

Dans les exemples à suivre on utilise une condition fausse. Imaginons qu'on parte d'un code hypothétique qui fasse ceci :

```
>>> condition = False # Utilisation d'une condition fausse
>>> if condition: # Version initiale
...     print("non")
... else:
...     print("bingo")
...
bingo
```

Supposons maintenant que l'on veuille modifier ce code pour simplement supprimer l'impression de « `non` ». La syntaxe du langage ne permet pas de simplement commenter le premier `print`.

```
>>> # En commentant le premier print la syntaxe devient incorrecte
>>> if condition:
... #     print("non")
... else:
File "<stdin>", line 3
  else:
  ^
IndentationError: expected an indented block
>>>     print("bingo")
File "<stdin>", line 1
  print("bingo")
  ^
IndentationError: unexpected indent
```

Évidemment ceci pourrait être récrit autrement en inversant la condition, mais parfois on s'efforce de limiter au maximum l'impact d'une modification sur le code. Dans ce genre de situation on préférera écrire plutôt le code suivant.

```
>>> # On peut s'en sortir en ajoutant une instruction pass
>>> if condition:
...     print("non")
...     pass
... else:
...     print("bingo")
...
bingo
```

Enfin, on peut également employer l'instruction `pass` pour définir une classe vide comme suit.



```
1 class Foo:
2     pass
3
4 foo = Foo()
```

3.3.2 Fonctions et valeur de retour

Une procédure est une fonction qui se contente de dérouler des instructions. Voici un exemple d'une telle fonction :

```
>>> def affiche_carre(n): # Définition
...     print("Le carré de", n, "vaut", n*n)
...
>>> affiche_carre(12) # Utilisation
Le carré de 12 vaut 144
```

Mais en fait, dans ce cas, il serait beaucoup plus commode de définir une fonction qui retourne le carré d'un nombre, afin de pouvoir écrire quelque chose comme `surface = carre(15)`, quitte à imprimer cette valeur ensuite si nécessaire. Jusqu'ici nous avons fait beaucoup appel à `print`, mais dans la pratique, imprimer n'est pas un but en soi.

Voici comment on pourrait écrire une fonction `carre` qui *retourne* (on dit aussi *renvoie*) le carré de son argument.

```
>>> def carre(n):
...     return n*n
...
>>> if carre(8) <= 100:
...     print('Petit appartement')
...
Petit appartement
```

²². Mot savant pour « comportement ».

La sémantique²² de l'instruction `return` est assez simple. La fonction qui est en cours d'exécution s'achève immédiatement et l'objet cité dans l'instruction `return` est retourné à l'appelant, qui peut employer cette valeur comme n'importe quelle expression.

Le terme même de fonction, si on se rappelle ses souvenirs de mathématiques, suggère qu'on calcule un résultat à partir de valeurs d'entrée. Dans la pratique, il est assez rare qu'on définit une fonction qui ne retourne rien.

En fait toutes les fonctions renvoient quelque chose. Lorsque le programmeur n'a pas prévu d'instruction `return`, PYTHON retourne un objet spécial, baptisé `None`. Voilà ce qu'on obtient si on essaie d'afficher le retour de la fonction `affiche_carre` qui ne retourne rien.

```
>>> retour = affiche_carre(15) # Premier appel provoquant l'impression d'une ligne
Le carré de 15 vaut 225
>>> print('retour =', retour) # Retour de la fonction
retour = None
```

L'objet `None` est un singleton prédéfini par PYTHON, un peu comme `True` et `False`. Ce n'est pas par contre une valeur booléenne.

Pour illustrer l'utilisation de `return` sur un exemple plus utile, considérons le code suivant.

```
>>> def premier(n):
...     """
...     Retourne un booléen selon que n est premier ou non.
...     Retourne None pour les entrées négatives ou nulles.
...     """
...     if n <= 0: # retourne None pour les entrées non valides
...         return
...     # Traitement du cas singulier (NB: elif est un raccourci pour else if)
...     # C'est utile pour éviter une indentation excessive
...     elif n == 1:
...         return False
...     # Recherche d'un diviseur dans [2..n-1],
...     # bien sûr on pourrait s'arrêter à la racine carrée de n mais ce n'est pas notre sujet
...     else:
...         for i in range(2, n):
...             if n % i == 0:
...                 return False # On a trouvé un diviseur, on peut sortir de la fonction
...     return True # À ce stade, le nombre est bien premier
... 
```

Cette fonction teste si un entier est premier ou non; il s'agit naturellement d'une version d'école, il existe d'autres méthodes beaucoup plus adaptées à cette tâche. On peut toutefois vérifier que cette version est fonctionnelle pour de petits entiers comme suit. On rappelle que le chiffre « 1 » n'est pas considéré comme un nombre premier.

```
>>> for test in [-2, 1, 2, 4, 19, 35]:
...     print(f"premier({test:2d}) = {premier(test)}")
...
premier(-2) = None
premier( 1) = False
premier( 2) = True
premier( 4) = False
premier(19) = True
premier(35) = False
```

Pour les besoins de la discussion, on a choisi de retourner `None` pour les entiers négatifs ou nuls, une manière comme une autre de signaler que la valeur en entrée n'est pas valide. Ceci n'est pas forcément une bonne pratique, mais elle permet ici d'illustrer que dans le cas où on ne mentionne pas de valeur de retour, PYTHON retourne `None`.

Comme on peut s'en convaincre en instrumentant le code — ce que l'on peut faire à titre d'exercice en ajoutant des fonctions `print` — dans le cas d'un nombre qui n'est pas premier, la boucle `for` ne va pas jusqu'à son terme. On aurait pu tirer profit de cette propriété pour écrire la fonction de manière légèrement différente, à l'instar de ce qui suit.

```
>>> def premier_sans_else(n):
...     """
...     Retourne un booléen selon que n est premier ou non
...     Retourne None pour les entrées négatives ou nulles
...     """
...     if n <= 0: # Retourne None pour les entrées non valides
...         return
...     if n == 1: # Traitement du cas singulier
...         return False
...     # Par rapport à la première version, on a supprimé la clause else: qui est inutile
```

```

...     for i in range(2, n):
...         if n % i == 0:
...             return False # On a trouvé un diviseur
...     return True # À ce stade c'est que le nombre est bien premier
...

```

C'est une question de style et de goût. En tout cas, les deux versions sont tout à fait équivalentes.

```

>>> for test in [-2, 2, 4, 19, 35]:
...     print(f"Pour n = {test:2d} : premier → {premier(test)}\n"
...           f"          premier_sans_else → {premier_sans_else(test)}\n")
...
Pour n = -2 : premier → None
premier_sans_else → None

Pour n =  2 : premier → True
premier_sans_else → True

Pour n =  4 : premier → False
premier_sans_else → False

Pour n = 19 : premier → True
premier_sans_else → True

Pour n = 35 : premier → False
premier_sans_else → False

```

DIGRESSION SUR LES CHAÎNES — On peut remarquer dans ce dernier morceau de code, en regardant bien le paramètre de `print`, qu'on peut accolter deux chaînes de caractères (ici deux f-strings) sans même les ajouter; un petit détail pour éviter d'alourdir le code.

```

>>> # Quand deux chaînes apparaissent immédiatement
>>> # l'une après l'autre sans opérateur, elles sont concaténées
>>> "abc" "def"
'abcdef'

```

3.3.3 Exercices



EXERCICE 11 — NOTES DES ÉLÈVES



On demande d'écrire une fonction qui prend deux arguments :

- ▶ une chaîne de caractères qui désigne le prénom d'un élève;
- ▶ un entier qui indique la note obtenue.

Elle devra retourner une chaîne de caractères selon que la note est :

- ▶ $0 \leq \text{note} < 10$
- ▶ $10 \leq \text{note} < 16$
- ▶ $16 \leq \text{note} \leq 20$

■ [Solution page 354](#)



EXERCICE 12 — SÉQUENCES ET SLICING



Commençons par créer une chaîne de caractères. Ne pas s'inquiéter en cas d'incompréhension du code d'initialisation²³ utilisé ci-dessous.

```

>>> import string # Constante définie dans le module string
>>> chaine = string.ascii_lowercase
>>> print(chaine) # Voici sa valeur
abcdefghijklmnopqrstuvwxyz

```

Pour chacune des sous-chaînes ci-après, écrire une expression de *slicing* sur `chaine` qui renvoie la sous-chaîne. La cellule de code doit retourner `True`. Par exemple, pour obtenir « `def` » : `chaine[3:6] == "def"`.

²³ Pour les plus curieux, l'instruction `import` permet de charger dans un programme une boîte à outils que l'on appelle un module. PYTHON vient avec de nombreux modules qui forment la bibliothèque standard. Le plus difficile avec les modules de la bibliothèque standard est de savoir qu'ils existent. En effet, il y en a un grand nombre et bien souvent il existe un module pour réaliser la tâche souhaitée. Ici en particulier nous utilisons le module STRING.

1. Écrire une *slice* pour obtenir « `vwx` » (ne pas hésiter à utiliser les indices négatifs) : `chaine[<code>] == "vwx"`.
 2. Écrire une *slice* pour obtenir « `wxyz` » (avec une seule constante) : `chaine[<code>] == "wxyz"`.
 3. Écrire une *slice* pour obtenir « `dfhjlnprtvxz` » (avec deux constantes) : `chaine[<vcode>] == "dfhjlnprtvxz"`.
 4. Écrire une *slice* pour obtenir « `xurolifc` » (avec deux constantes) : `chaine[<code>] == "xurolifc"`.
- En cas de difficulté pour répondre à ces questions, consulter § 2.2.1.

 [Solution page 354](#)



EXERCICE 13 — SÉQUENCES ET LONGUEUR



On donne une chaîne composite dont on sait qu'elle a été calculée à partir de deux chaînes `inconnue` et `connue` comme ceci : `composite = connue + inconnue + connue`. On fournit également la chaîne `connue`. Si par exemple, `connue = '0bf1'` et `composite = '0bf1a9730e150bf1'` alors `inconnue = 'a9730e15'`.

L'exercice consiste à écrire une fonction qui retourne la valeur de `inconnue` à partir de celles de `composite` et `connue`. On peut admettre que `connue` n'est pas vide, c'est-à-dire qu'elle contient au moins un caractère. On peut utiliser du *slicing* et la fonction `len()`, qui retourne la longueur d'une chaîne : `len('abcd')=4`.

 [Solution page 355](#)



EXERCICE 14 — LISTES



Écrire une fonction `laccess` qui a en argument une liste et qui renvoie :

- `None` si la liste est vide;
- sinon le dernier élément de la liste si elle est de taille paire;
- et sinon l'élément du milieu.

 [Solution page 355](#)



EXERCICE 15 — FONCTION DE DIVISIBILITÉ



L'exercice consiste à écrire une fonction nommée `divisible` qui retourne une valeur booléenne, laquelle indique si un des deux arguments est divisible par l'autre. On peut supposer les entrées `a` et `b` entiers et non nuls, mais pas forcément positifs.

 [Solution page 356](#)



EXERCICE 16 — FONCTION DÉFINIE PAR MORCEAUX



Obtenir en PYTHON une fonction définie par morceaux telle que :

$$\begin{aligned} f(x) &= \begin{cases} -x - 5 & \text{si } x < -5 \\ 0 & \text{si } x \in [-5, 5] \\ \frac{1}{5}x - 1 & \text{si } x \geq 5 \end{cases} \end{aligned}$$

 [Solution page 356](#)



EXERCICE 17 — COMPTAGE AU SEIN DES CHAÎNES



Sur les systèmes de type UNIX, la commande `wc` permet de compter le nombre de lignes, de mots et d'octets (ou de caractères) présents sur l'entrée standard ou contenus dans un fichier.

L'exercice consiste à écrire une fonction nommée `wc` qui prend en argument une chaîne de caractères et retourne une liste de trois éléments :

- ▶ le nombre de lignes (plus précisément le nombre de retours à la ligne);
- ▶ le nombre de mots (un mot étant séparé par des espaces);
- ▶ le nombre de caractères (uniquement le jeu de caractères ASCII).

Indice : la boucle `for` a été rapidement exposée, il faut toutefois savoir qu'on peut tout à fait résoudre l'exercice en utilisant uniquement la bibliothèque standard.

Remarque : usuellement, ce genre de fonctions retourne plutôt un tuple qu'une liste, mais ceux-ci n'ont pas encore été présentés.

 Solution page 358



SOLUTION 11 — NOTES DES ÉLÈVES (PAGE 352)

Trois solutions sont possibles.

```
>>> def label(prenom, note):
...     if note < 10:
...         return f"{prenom} est recalé"
...     elif note < 16:
...         return f"{prenom} est reçu"
...     else:
...         return f"Félicitations à {prenom}"
...
>>> label('Rémi', 10)
'Rémi est reçu'
>>> label('Marie', 17)
'Félicitations à Marie'
>>> label('Kevin', 8)
'Kevin est recalé'
>>> label('Jean', 14)
'Jean est reçu'
>>> # Juste pour illustrer cette construction
>>> def label_bis(prenom, note):
...     if note < 10:
...         return f"{prenom} est recalé"
...     elif 10 <= note < 16:
...         return f"{prenom} est reçu"
...     else:
...         return f"félicitations à {prenom}"
...
>>> # L'expression conditionnelle n'a pas été abordée
>>> # mais par curiosité on peut aussi faire comme ceci
>>> def label_ter(prenom, note):
...     return f"{prenom} est recalé" if note < 10 \
...     else f"{prenom} est reçu" if 10 <= note < 16 \
...     else f"félicitations à {prenom}"
...
```



SOLUTION 12 — SÉQUENCES ET SLICING (PAGE 352)

```
>>> chaine[-5:-2] == "vwx"
True
>>> chaine[-4:len(chaine)] == "wxyz"
True
>>> chaine[3:len(chaine):2] == "dfhjlnprtvxz"
True
>>> chaine[-3::-3] == "xurolifc"
True
```

SOLUTION 13 — SÉQUENCES ET LONGUEUR (PAGE 353)

Pour enlever à gauche et à droite une chaîne de longueur x , on peut faire `composite[x : -x]` or ici x vaut `len(connue)`

```
1 def inconnue(composite, connue):
2     return composite[ len(connue) : -len(connue) ]
```

Ce qui peut aussi s'écrire comme ci-dessous si on préfère.

```
1 def inconnue_bis(composite, connue):
2     return composite[ len(connue) :
3         -len(composite)-len(connue) ]
```

SOLUTION 14 — LISTES (PAGE 353)

```
>>> def laccess(liste):
...     """
...     Retourne un élément de la liste selon la taille
...     """
...     if not liste: # Si la liste est vide ne rien faire
...         return
...     if len(liste) % 2 == 0: # Liste de taille paire
...         return liste[-1]
...     else: # Liste de taille impaire
...         return liste[len(liste)//2]
...
>>> laccess([])
>>> laccess([1])
1
>>> laccess(['spam', 100])
100
>>> laccess(['spam', 100, 'bacon'])
100
>>> laccess([1, 2, 3, 100])
100
>>> laccess([1, 2, 100, 4, 5])
100
>>> laccess(['si', 'pair', 'alors', 'dernier'])
'dernier'
>>> laccess(['retourne', 'le', 'milieu', 'si', 'impair'])
'milieu'

>>> # une autre version qui utilise un trait pas encore vu
>>> def laccess_bis(liste):
...     # si la liste est vide il n'y a rien à faire
...     if not liste:
...         return
...     # l'index à utiliser selon la taille
...     index = -1 if len(liste) % 2 == 0 else len(liste) // 2
...     return liste[index]
```

Une fois que le code fonctionne, on peut regarder si par hasard il fonctionnerait aussi avec des chaînes de caractères.

```
>>> laccess("")
>>> laccess("a")
'a'
>>> laccess("ab")
'b'
```

```
>>> laccess("abc")
'b'
>>> laccess("abcd")
'd'
>>> laccess("abcde")
'c'
```

SOLUTION 15 — FONCTION DE DIVISIBILITÉ (PAGE 353)

Là encore, il y a deux formalisations possibles.

```
>>> def divisible(a, b):
...     "Renvoie True si un des deux arguments divise l'autre"
...     # b divise a si et seulement si le reste
...     # de la division de a par b est nul
...     if a % b == 0:
...         return True
...     # et il faut regarder aussi si a divise b
...     if b % a == 0:
...         return True
...     return False
...
>>> def divisible_bis(a, b):
...     "Renvoie True si un des deux arguments divise l'autre"
...     # On n'a pas encore vu les opérateurs logiques, mais
...     # on peut aussi faire tout simplement comme ça
...     # sans faire de if du tout
...     return a % b == 0 or b % a == 0
...
>>> divisible(10, 30)
True
>>> divisible(10, -30)
True
>>> divisible(-10, 30)
True
>>> divisible(8, 12)
False
>>> divisible(12, -8)
False
>>> divisible(-12, 8)
False
>>> divisible(-12, -8)
False
>>> divisible(10, 1)
True
>>> divisible(30, -10)
True
>>> divisible(-30, 10)
True
>>> divisible(-30, -10)
True
```

SOLUTION 16 — FONCTION DÉFINIE PAR MORCEAUX (PAGE 353)

Ici, trois formulations sont possibles.

```
>>> def morceaux(x):
...     if x <= -5:
...         return -x - 5
...     elif x <= 5:
```

```

...
    return 0
...
else:
    return x / 5 - 1

...
>>> def morceaux_bis(x):
...
    if x <= -5:
...
        return -x - 5
...
    if x <= 5:
...
        return 0
...
    return x / 5 - 1

...
>>> def morceaux_ter(x):
...
    if x <= -5:
...
        return -x - 5
...
    elif -5 <= x <= 5:
...
        return 0
...
    else:
        return x / 5 - 1

...
>>> morceaux(-10)
5
>>> morceaux(0)
0
>>> morceaux(10)
1.0
>>> morceaux(-6)
1
>>> morceaux(-4)
0
>>> morceaux(5)
0
>>> morceaux(-5)
0
>>> morceaux(4)
0
>>> morceaux(20)
3.0

```

En référence à la fonction `morceaux`, on veut visualiser le résultat. Autrement dit, en partant du code de la fonction, on veut produire sa courbe en utilisant NUMPY et MATPLOTLIB.

```

>>> # Importation des bibliothèques
>>> import numpy as np
>>> import matplotlib.pyplot as plt

>>> # Échantillons : X entre -10 et 20 et Y correspondants
>>> X = np.linspace(-10, 20)
>>> Y = np.vectorize(morceaux)(X)

>>> # On n'a plus qu'à dessiner
>>> plt.xlabel("Axe des abscisses")
Text(0.5, 0, 'Axe des abscisses')
>>> plt.ylabel('Axe des ordonnées')
Text(0, 0.5, 'Axe des ordonnées')
>>> plt.title('Fonction par morceaux')
Text(0.5, 1.0, 'Fonction par morceaux')
>>> plt.plot(X, Y)
[<matplotlib.lines.Line2D object at 0x7f92ffcf5b80>]

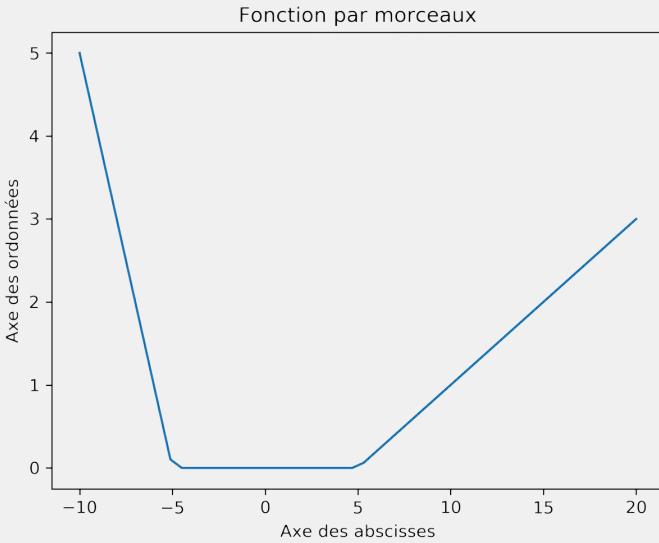
>>> # Affichage à l'écran
>>> #plt.show()

```

NOTE DE LA RÉDACTION

Par rapport à l'interactivité d'un *notebook JUPYTER*, nous sommes contraints d'enregistrer la figure, puis de l'insérer au document. En effet, l'affichage écran n'a lieu qu'au moment de la compilation *LATEX* avec l'extension *PYTHONLATEX*. L'affichage interactif est ainsi introduit en commentaire.

```
>>> # Pour affichage ultérieur
>>> plt.savefig('morceaux.png', transparent=True, dpi=300)
>>> plt.close()
```



SOLUTION 17 — COMPTAGE AU SEIN DES CHAÎNES (PAGE 353)

```
>>> def wc(string):
...     """
...         Compte les nombres de lignes, de mots et de caractères
...         Retourne une liste de ces trois nombres
...         (à noter qu'usuellement on renverrait plutôt un tuple)
...     """
...     # On peut tout faire avec la bibliothèque standard
...     nb_lines = string.count('\n')
...     nb_words = len(string.split())
...     nb_bytes = len(string)
...     return [nb_lines, nb_words, nb_bytes]
...
>>> wc('')
[0, 0, 0]
>>> wc('abc')
[0, 1, 3]
>>> wc('abc \t')
[0, 1, 5]
>>> wc('a bc \t')
[0, 2, 7]
>>> wc(' \tabc \n')
[1, 1, 7]
>>> wc(" ".join("abcdefg") + "\n")
[1, 7, 14]
>>> wc('''The Zen of Python, by Tim Peters)
... Beautiful is better than ugly.
... Explicit is better than implicit.
... Simple is better than complex.
... Complex is better than complicated.
... Flat is better than nested.
... Sparse is better than dense.
... ...
... '''),
([7, 38, 226],)
```

4 Compréhensions de listes et modules

4.1 Compréhensions de liste

Les listes sont au cœur de tout programme PYTHON. La liste est en effet un objet flexible pouvant référencer n'importe quel type d'objet.

Une manière simple de parcourir les listes est d'implémenter une boucle **for** pour appliquer une opération à chaque élément. Cette opération est tellement commune que PYTHON a inventé une nouvelle expression qu'on appelle *compréhension de liste*. Cette dernière permet, de manière simple et intuitive, d'adresser une opération aux éléments d'une liste et éventuellement d'ajouter une condition de filtre.

4.1.1 Syntaxe et utilisation

Supposons que l'on souhaite prendre le logarithme d'une liste d'entiers **a** = [1, 4, 18, 29, 13]. Il faut importer le module math qui implémente le logarithme. Essayons ensuite de mettre le résultat dans une nouvelle liste contenant le logarithme des éléments de **a**. On crée une liste vide **b**=[] et on établit une boucle **for** sur les éléments de **a** et de **b** pour aller chercher le logarithme.

```
>>> a = [1, 4, 18, 29, 13]
>>> import math
>>> b = []
>>> for i in a:
...     b.append(math.log(i))
...
>>> b
[0.0, 1.3862943611198906, 2.8903717578961645, 3.367295829986474, 2.5649493574615367]
```

Le concept de compréhension de liste permet une écriture plus ramassée et fournit directement une nouvelle liste, ici des logarithmes des éléments de **a**. Pour s'en convaincre, il suffit de comparer la syntaxe du résultat précédent avec la nouvelle formulation qui s'avère quasiment du langage naturel.

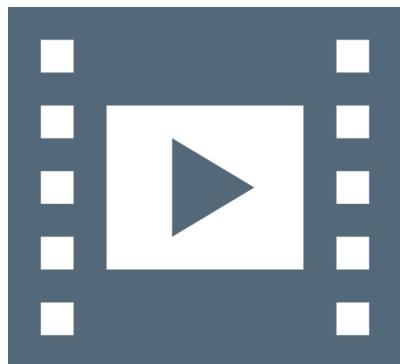
```
>>> b = [math.log(i) for i in a]
>>> b
[0.0, 1.3862943611198906, 2.8903717578961645, 3.367295829986474, 2.5649493574615367]
```

Supposons désormais que dans la liste **a** il y ait un entier négatif : **a.append(-1)**. On ne peut donc plus calculer le logarithme de cet élément. Néanmoins, avec une compréhension de liste, on peut rajouter un test comme suit pour contourner le problème.

```
>>> a.append(-1)
>>> a
[1, 4, 18, 29, 13, -1]
>>> b = [math.log(i) for i in a if i > 0]
>>> b
[0.0, 1.3862943611198906, 2.8903717578961645, 3.367295829986474, 2.5649493574615367]
```

On pourrait croire avec ces exemples que les compréhensions de listes sont réservées aux calculs sur des nombres entiers. En fait, cela n'est absolument pas le cas. Une compréhension de listes permet d'appliquer n'importe quelle opération, instruction et fonction sur n'importe quelle type de séquence.

Prenons maintenant un exemple avec des chaînes de caractères en partant d'une liste de prénoms dont certains ont une mauvaise capitalisation : **prenom = ['Alice','evE','sonia','BOB']** et essayons de



VIDÉO 9.8 – Compréhension de liste.



tous les mettre en lettres minuscules avec la méthode `lower()` des chaînes de caractères. On peut tout aussi bien mettre en capitale uniquement la première lettre des prénoms avec la méthode `title()`.

```
>>> prenom = ['Alice', 'evE', 'sonia', 'BOB']
>>> prenom
['Alice', 'evE', 'sonia', 'BOB']
>>> prenom = [p.lower() for p in prenom]
>>> prenom
['alice', 'eve', 'sonia', 'bob']
>>> prenom = [p.title() for p in prenom]
>>> prenom
['Alice', 'Eve', 'Sonia', 'Bob']
```

4.1.2 Exercices



EXERCICE 18 — FONCTION POLYNOMIALE



On se donne une fonction polynomiale : $P(x) = 2x^2 - 3x - 2$.

Il est demandé d'écrire une fonction `liste_P` qui prend en argument une liste de nombres réels x et qui retourne la liste des valeurs $P(x)$.

On suggère d'écrire une fonction `P` qui implémente le polynôme mais ça n'est pas strictement indispensable, seul le résultat de `liste_P` compte.

[Solution page 360](#)



EXERCICE 19 — MISE AU CARRÉ



On demande à présent d'écrire une fonction dans le même esprit que la fonction polynomiale précédente. Cette fois, chaque ligne contient, séparés par des points-virgules, une liste d'entiers et on veut obtenir une nouvelle chaîne avec les carrés de ces entiers, séparés par des deux-points.

À nouveau les lignes peuvent être remplies de manière approximative, avec des espaces, des tabulations ou même des points-virgules en trop, que ce soit au début, à la fin ou au milieu d'une ligne.

[Solution page 361](#)



SOLUTION 18 — FONCTION POLYNOMIALE (PAGE 360)

```
>>> def P(x):
...     return 2 * x**2 - 3 * x - 2
...
>>> def liste_P(liste_x):
...     """
...     retourne la liste des valeurs de P
...     sur les entrées figurant dans liste_x
...     """
...     return [P(x) for x in liste_x]
...
>>> def liste_P_bis(liste_x):
...     liste_y = []
...     for x in liste_x:
...         liste_y.append(P(x))
...     return liste_y
...
>>> a = list(range(5))
>>> liste_P(a)
[-2, -3, 0, 7, 18]
```

```
>>> b = list(range(-7, 8, 2))
>>> liste_P(b)
[117, 63, 25, 3, -3, 7, 33, 75]
>>> c = [-100, 0, 100]
>>> liste_P(c)
[20298, -2, 19698]
```

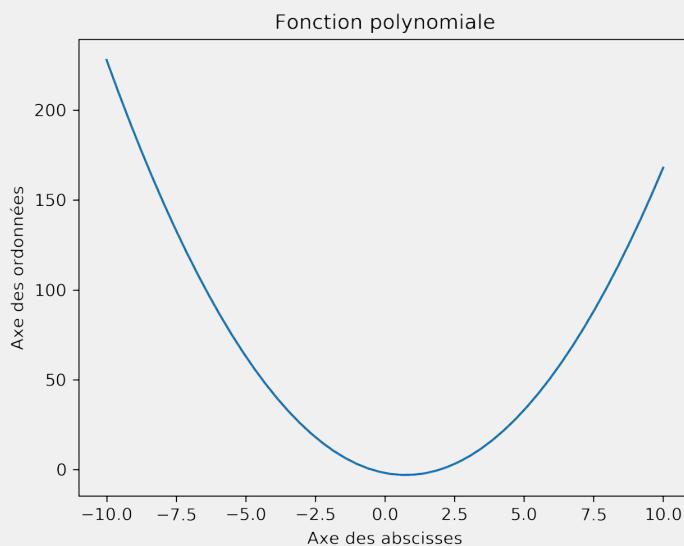
On peut ensuite visualiser le résultat

```
>>> # Importation des bibliothèques
>>> import numpy as np
>>> import matplotlib.pyplot as plt

>>> # Échantillons : X entre -10 et 10 et Y correspondants
>>> X = np.linspace(-10, 10)
>>> Y = liste_P(X)

>>> # On n'a plus qu'à dessiner
>>> plt.xlabel("Axe des abscisses")
Text(0.5, 0, 'Axe des abscisses')
>>> plt.ylabel('Axe des ordonnées')
Text(0, 0.5, 'Axe des ordonnées')
>>> plt.title('Fonction polynomiale')
Text(0.5, 1.0, 'Fonction polynomiale')
>>> plt.plot(X, Y)
[<matplotlib.lines.Line2D object at 0x7f92eef0e3d0>]

>>> # Affichage à l'écran
>>> #plt.show()
>>> # Pour affichage ultérieur
>>> plt.savefig('polynome.png', transparent=True, dpi=300)
>>> plt.close()
```



SOLUTION 19 — MISE AU CARRÉ (PAGE 360)

```
>>> def carre(line):
...     # On enlève les espaces et les tabulations
...     line = line.replace(' ', '').replace('\t','')
...     # La ligne suivante fait le plus gros du travail
...     # d'abord on appelle split() pour découper selon les ;'
```

```

...
    # dans le cas où on a des ';' en trop, on obtient dans
...
    # le résultat du split un 'token' vide, que l'on ignore
...
    # ici avec la clause 'if token' enfin on convertit
...
    # tous les tokens restants en entiers avec int()
...
    entiers = [int(token) for token in line.split(";")]
...
        # en éliminant les entrées vides qui
...
        # correspondent à des point-virgules en trop
...
        if token]
...
    # Il n'y a plus qu'à mettre au carré, retraduire
...
    # en strings et à recoudre le tout avec join et ':'
...
    return ":".join([str(entier**2) for entier in entiers])
...
...
>>> def carre_bis(line):
...
    # Pareil mais avec, à la place des compréhensions
...
    # des expressions génératrices pas encore étudiées
...
    # L'illustration ici est d'avoir exactement
...
    # le même code mais avec () au lieu de []
...
    line = line.replace(' ', '').replace('\t','')
...
    entiers = (int(token) for token in line.split(';'))
...
        if token)
...
    return ":".join(str(entier**2) for entier in entiers)
...
...
>>> carre("1;2;3")
'1:4:9'
>>> carre(" 2 ; 5;6;")
'4:25:36'
>>> carre("; 12 ; -23;\t60; 1\t")
'144:529:3600:1'
>>> carre("; -12 ; ; -23; 1 ;;\t")
'144:529:1'

```

4.2 Modules

Un module est un fichier PYTHON standard se terminant par l'extension « `.py` ». Lorsque l'on importe ce fichier avec l'instruction `import`, cela crée un objet module.

Un module contient un certain nombre de fonctions et d'opérations prédéfinies. L'idée d'un module est de regrouper dans le même fichier des opérations similaires. Ce faisant, un module peut se voir comme une boîte à outils à laquelle on fait appel quand on a besoin de l'ouvrir.

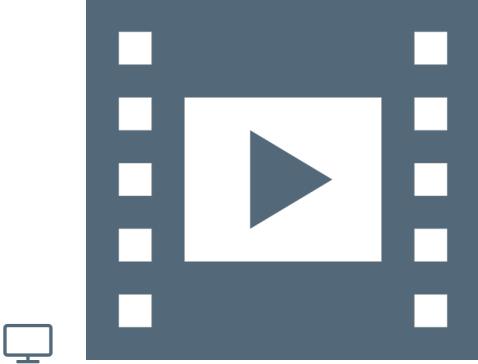
4.2.1 Attributs

On peut illustrer concrètement les choses à l'aide du module `RANDOM`. On peut vérifier sa nature par la fonction `print(random)`, qui renvoie que c'est bien un objet module.

On a accès à tous les attributs possibles du module avec la commande `dir(random)`. Pour mémoire, les attributs avec double *underscore* ne sont pas à manipuler directement.

On peut également avoir de l'aide sur un module en utilisant la fonction *buit-in* `help(random)`. Mentionnons au passage qu'il est rare d'utiliser l'aide directement car cela fait beaucoup de texte à traiter. Dans ce cas et en pratique, il est conseillé de consulter la documentation sur Internet.

L'aide se trouve néanmoins utile lorsqu'on veut regarder le fonctionnement d'une méthode singulière d'un module, comme `random.randint?` avec IPYTHON et `help(random.randint)` — retour aléatoire d'un entier entre deux bornes incluses — pour les autres interpréteurs.



VIDÉO 9.9 — *Modules*.

```

>>> import random
>>> print(random)
<module 'random' from '/usr/lib/python3.8/random.py'>
>>> dir(random)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI',
 '_Sequence', '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
 '__name__', '__package__', '__spec__', '__accumulate', '__acos', '__bisect', '__ceil', '__cos', '__e',
 '__exp', '__inst', '__log', '__os', '__pi', '__random', '__repeat', '__sha512', '__sin', '__sqrt', '__test',
 '__test_generator', '__urandom', '__warn', 'betavariate', 'choice', 'choices', 'expovariate',
 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariate',
 'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle',
 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']
>>> help(random.randint)
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.

>>> random.randint(1, 100)
51

```

4.2.2 Bibliothèque standard et autres

PYTHON est livré avec un grand nombre de modules — environ une centaine. C'est ce qu'on appelle la *bibliothèque standard*. Cela permet de réaliser de multiples opérations courantes.

L'intérêt de la bibliothèque standard est d'avoir la certitude de disposer des mêmes modules pour chacune des installations de PYTHON.

Mentionnons quelques opérations possibles et réalisables au moyen de la bibliothèque standard :

- ▶ programmation parallèle;
- ▶ programmation asynchrone;
- ▶ persistance de données (sérialisation);
- ▶ communication sur Internet (support de tous les protocoles);
- ▶ formatage et lecture des données;
- ▶ manipulation de fractions et de nombres décimaux;
- ▶ écriture d'expressions régulières;
- ▶ gestion de dates et calendriers;
- ▶ interaction avec le système de fichiers;
- ▶ création de fichiers et répertoires;
- ▶ compression de fichiers;
- ▶ codage d'interfaces graphiques.

Outre la bibliothèque standard, PYTHON peut intégrer des centaines de milliers de modules écrits par des groupes de développement ou des individus pour répondre à des usages spécifiques. Il suffit de les installer et de les charger pour en bénéficier, par exemple NUMPY et PANDAS en *data-science* ou MATPLOTLIB en visualisation de données.

5 Que faire de ces ressources ? Autoévaluation

NOTE DE LA RÉDACTION

La présentation des quiz du document suit plus ou moins celle de la plateforme FUN-Mooc. La fonctionnalité manquante — pas encore implémentée dans l'extension de style LATEX usitée — est relative à la comptabilisation des points et à leur enregistrement. Aussi, il appartient au lecteur de jouer le jeu dans l'autoévaluation de ses connaissances.

Le questionnaire à choix multiple* — QCM — à suivre porte sur le présent chapitre « Concepts élémentaires ».

* De manière traditionnelle en IHM, lorsqu'une seule réponse est correcte, les propositions sont précédées d'un cercle à cocher (radio button); en revanche, dans le cas de plusieurs solutions possibles, il s'agit de carrés (check box). En outre, après validation des réponses (« Vérifier »), leur explication s'affiche en marge ou infobulle (« Afficher la réponse »).

QUIZ 27 — CODAGE ET UNICODE

2 POINTS

QUIZ 27.1 — ENCODAGE 1 POINT

À quoi sert le décodage d'un flux de bits en informatique ?

- À obtenir un caractère.
- À obtenir une phrase.
- À obtenir un code que l'on utilisera pour trouver un caractère dans un jeu de caractères.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 27.2 — UNICODE 1 POINT

Cocher la ou les affirmations qui sont vraies.

- Unicode est un standard qui définit à la fois un jeu de caractères et des encodages.
- Unicode définit trois encodages UTF-8, UTF-16 et UTF-32.
- UTF-8 est l'encodage à favoriser.
- Unicode ne supporte pas certains caractères courants comme le symbole de l'euro.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 28 — CHAÎNES DE CARACTÈRES

4 POINTS

QUIZ 28.1 — MUTABILITÉ 1 POINT

Les chaînes de caractères en PYTHON sont mutables.

- Vrai.
- Faux.

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

QUIZ 28.2 — EXPRESSIONS 1 POINT

Cocher la ou les expressions qui définissent correctement une chaîne de caractères en PYTHON.

- 'spam"
- "spam"
- 'spam''

'spam'

....

"""

....

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 28.3 — FORMATAGE 1 POINT

On veut afficher la chaîne de caractères **le numéro d'appel des urgences est le 112** et on a les variables suivantes qui sont définies : `text = 'des urgences', numero = 112`. Quelle(s) solution(s) est(sont) correcte(s) ?

- 'le numéro d'appel {} est le {}'.format(text, numero).
- "le numéro d'appel text est le numero".
- "le numéro d'appel est le ".format(text, numero).
- f"le numéro d'appel text est le numero".

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 28.4 — ENCODAGE 1 POINT

Quelle(s) affirmation(s) est(sont) vraie(s) ?

- Sans spécifier explicitement l'encodage choisi, le codage utilisé sera dépendant de la machine sur laquelle on fait tourner le programme.
- Le type **str** est pour les chaînes de caractères décodées et le type **bytes** est pour les chaînes de caractères encodées.
- Lorsqu'une chaîne est décodée on ne peut plus l'encoder.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 29 — SÉQUENCES**3 POINTS****QUIZ 29.1 — LONGUEUR 1 POINT**

On se donne la variable suivante : `chaine="douarnenez"`. Parmi les expressions suivantes, quelles sont celles qui s'évaluent à **True** ?

- `chaine.index('z') == len(chaine)`.
- `chaine.index('z') == len(chaine) - 1`.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 29.2 — APPARTENANCE 1 POINT

On se donne la variable suivante : `chaine="douarnenez"`. Parmi les expressions suivantes, quelles sont celles qui s'évaluent à **True** ?

- `'daz' in chaine`.

'nez' in chaîne.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 29.3 — SLICING 1 POINT

On se donne la variable suivante : `chaîne="douarnenez"`. Parmi les expressions suivantes, quelles sont celles qui s'évaluent à `True` ?

`chaîne[-3:] == 'nez'`.

`chaîne[1:3] + chaîne[3:5] + chaîne[5:] == chaîne[1:]`.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 30 — LISTES 3 POINTS

QUIZ 30.1 — INSERTION 1 POINT

On a `liste = [0, 1, 2, 3]`. On veut modifier l'objet `liste` pour que sa valeur devienne `[0, 1, 4, 2, 3]`. Que faut-il faire ? Plusieurs réponses sont possibles.

`liste[2] = 4`

`liste[2] = [4]`

`liste.insert(2,4)`

`liste[2:3] = [4]`

`liste[2:2] = [4]`

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 30.2 — EXTRACTION 1 POINT

On a de nouveau `liste = [0, 1, 2, 3]`. On souhaite extraire et retourner le premier élément `0`, tout en la retirant de la liste. Plus précisément on veut affecter à la variable suivant la valeur `0` de telle sorte qu'après l'exécution, `liste` ne contienne plus que `[1, 2, 3]`. Que faut-il faire ? Plusieurs réponses sont possibles.

`suivant = liste[0]`

`suivant = liste.pop(0)`

`del liste[0]`

`suivant = liste[0]; del liste[0]`

`suivant = del liste[0]`

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 30.3 — TRI 1 POINT

On a cette fois `liste = [1, 0, 3, 2]`. On veut trier la liste en ordre décroissant et *en place*, c'est-à-dire sans dupliquer la liste ni ses éléments. Que faut-il faire ? Plusieurs réponses sont possibles.

`liste.sort(reverse=True)`

`liste.sort()`

`sorted(liste, reverse=True)`

`liste.sort(); liste.reverse()`

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 31 — CONDITIONNELLES ET SYNTAXE **2 POINTS**

QUIZ 31.1 — TEST **1 POINT**
Que peut-on dire du code suivant?

```
1 if 1 < 2
  print('oui')
else:
  print('non')
```

Ce code est invalide.

Ce code affiche « oui ».

Ce code affiche « non ».

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 31.2 — CONVENTION DE CODAGE **1 POINT**
Quelle(s) affirmation(s) est(sont) vraie(s)?

Des conventions de codage sont dans la syntaxe de PYTHON.

Il y a peu de contraintes sur la manière de présenter le code.

On utilise des accolades pour identifier les blocs d'instructions

Il faut toujours un « : » en fin de ligne précédant un bloc de code.

VÉRIFIER **AFFICHER LA RÉPONSE** **RÉINITIALISER**

QUIZ 32 — BOUCLE ET FONCTIONS **2 POINTS**

QUIZ 32.1 — BOUCLE **1 POINT**
Voici une boucle `for`, que va afficher cette boucle.

```
1 a = []
2 for n in [1, 2, '3', 4, 'FIN']:
  a.append(str(n))
3
4 print(", ".join(a))
```

Une erreur.

`1,2,'3',4,'FIN'`.

`1,2,3,4,FIN`.

- 1,
2,
3,
4,
FIN

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 32.2 — FONCTIONS 1 POINT**

Voici une fonction.

```
1 def to_str(a):
2     tmp = []
3     for i in a:
4         tmp.append(str(i))
5     return " ".join(tmp)
```

Quel(s) appel(s) va(vont) produire le retour suivant : '1 2 3'.

-
- to_str([1, 2, 3])

-
- to_str(['1', '2', '3'])

-
- to_str('123').

-
- [1, 2, 3].to_str()

-
- a = [1, 2, 3],
to_str

-
- to_str(['123'])

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER****QUIZ 33 — COMPRÉHENSION DE LISTE****1 POINT****QUIZ 33.1 — LISTES DE CARRÉS 1 POINT**On suppose avoir la liste : `entree = [1,2,3,4,5,6,7,8,9]`. Parmi les fragments suivants, lesquels sont valides en PYTHON ?

-
- carres = [for x in entree: x**2]

-
- carres = [x**2 for x in entree].

-
- carres = [for x in entree: if x % 3 == 0: x**2].

-
- carres = [x**2 for x in entree: if x % 3 == 0]

-
- carres = [x**2 for x in entree if x % 3 == 0]

VÉRIFIER**AFFICHER LA RÉPONSE****RÉINITIALISER**

V

ADDENDA

371 | RÉFÉRENCES

391 | GLOSSAIRE

RÉFÉRENCES

Bibliographie

[ABITEBOUL, ANDRÉ, et al., 2015] — Serge ABITEBOUL, Benjamin ANDRÉ, and Daniel KAPLAN, 2015 — “*Managing your digital life*”, *Communications of the Association for the Computing Machinery*, vol. 58, n°5, pp. 32–35 — ISSN: 0001-0782.

[ABITEBOUL et DOWEK, 2017] — Serge ABITEBOUL et Gilles DOWEK, 2017 — « *Le temps des algorithmes* », 1^{re} éd., Collection Essais. Éditions Le Pommier, Paris, 192 pages — ISBN : 978-2-7465-1175-0.

Depuis quelque temps, les algorithmes sont sur toutes les langues. Et ils inquiètent autant qu'ils fascinent. Avec eux, nous passons facilement d'un extrême à l'autre : nous nous réjouissons qu'ils nous facilitent la vie, mais redoutons qu'ils nous asservissent... Pour en finir avec cette vision manichéenne, cet ouvrage propose un nouveau regard sur notre époque, sur le temps des algorithmes. Pour commencer, cessons de considérer les algorithmes comme des êtres mystérieux, dotés d'intentions maléfiques. Après tout, les algorithmes sont des créations de l'esprit humain. Ils sont ce que nous avons voulu qu'ils soient. Les algorithmes sont avant tout des solutions, mais ces solutions ne sont pas neutres. S'ils sont à l'origine de transformations radicales des notions de travail, de propriété, de gouvernement, de responsabilité, de vie privée et même d'humanité, c'est à nous de décider de quel côté faire pencher la balance. Les algorithmes sont peut-être le premier outil à la mesure de nos aspirations. Cessons de les subir, en cherchant à les comprendre. C'est ainsi que nous pourrons être maîtres de notre destinée.

[BERNERS-LEE et al., 1994] — Tim BERNERS-LEE, Robert CAILLIAU, Ari LUOTONEN, Henrik Frystyk NIELSEN, and Arthur SECRET, 1994 — “*The World-Wide Web*”, *Communications of the Association for the Computing Machinery*, vol. 37, n°8, pp. 76–82 — ISSN: 0001-0782.

[BERRY, 2017] — Gérard BERRY, 2017 — « *L'Hyperpuissance de l'informatique. Algorithmes, données, machines, réseaux* », 1^{re} éd., Éditions Odile Jacob, Paris, 512 pages — ISBN : 978-2-7381-3953-5.

Ce n'est que récemment que l'on a commencé à mesurer à quel point l'informatique est en passe de transformer notre société, et même de la bouleverser de fond en comble. C'est à décrire et à analyser les fondements de l'hyperpuissance de l'informatique que Gérard Berry se consacre dans ce livre qui fera date.

Il montre en effet de façon non technique comment la science et la technologie informatiques mettent l'information au cœur de l'action, qu'elle soit produite par les hommes ou par les machines. Algorithmes, données, machines et réseaux conduisent surtout à un nouveau schéma mental bien différent de celui des siècles précédents, qui confère un pouvoir étonnant à ceux qui le comprennent et l'organisent.

Pour donner concrètement à comprendre le mode de pensée inhérent à l'informatique, Gérard Berry passe en revue cinq domaines de transformations massives : les télécommunications, Internet, la photographie et la cartographie, l'informatisation de la médecine, et celle en cours de

toutes les sciences. Il analyse ensuite en détail deux dangers de l'informatique, les bugs et les trous de sécurité, qui peuvent parfois transformer des systèmes informatisés en dangers publics, et montre comment la science moderne permet de mieux contrôler ces dangers. Enfin, l'auteur donne sa vision de l'évolution de l'informatique, bien loin des fantasmes trop souvent partagés. Pour la première fois, un livre qui explique tout de l'informatique, de son monde, ses fondements, ses applications, et la révolution qu'elle représente.

[BOBILLIER, 2012] — Sébastien BOBILLIER, 2012 — « Préparation à la certification Linux LPIC-2. Examens LPI 201 et LPI 202 », 2^e éd., Certifications. Éditions ENI, Saint Herblain, 434 pages — ISBN : 978-2-7460-7557-3.

Les examens LPI 201 et LPI 202 sont les deux examens qui permettent d'obtenir la certification LPIC-2 Advanced Level Linux Professional. Ce programme de certification du Linux Professional Institute est de plus en plus reconnu par les recruteurs qui voient dans cette certification un pré-requis à l'embauche ou à l'accès à un poste d'administrateur. Les examens LPI 201 et 202 prouvent aux professionnels que vous maîtrisez l'administration avancée d'un système LINUX quelle que soit la distribution : ils sanctionnent une compétence pratique en termes d'administration d'un réseau de petite ou moyenne taille (administration des services réseaux courants, gestion de la sécurité du réseau et des échanges...). Pour vous aider à préparer efficacement cette certification, ce livre couvre tous les objectifs officiels de la dernière version de l'examen (mise en place en juillet 2012) tant d'un point de vue théorique que d'un point de vue pratique. Il a été rédigé en français (il ne s'agit pas d'une traduction) par un formateur professionnel reconnu, également consultant, certifié LINUX. Ainsi, les savoir-faire pédagogique et technique de l'auteur conduisent à une approche claire et visuelle, d'un très haut niveau technique. Chapitre par chapitre, vous pourrez valider vos acquis théoriques, à l'aide d'un grand nombre de questions-réponses (110 au total) mettant en exergue aussi bien les éléments fondamentaux que les caractéristiques spécifiques aux concepts abordés. Chaque chapitre s'achevant par des travaux pratiques (32 au total) vous aurez les moyens de mesurer votre autonomie. Ces manipulations concrètes, au-delà même des objectifs fixés par l'examen, vous permettront de vous forger une première expérience significative et d'acquérir de véritables compétences techniques sur des mises en situations réelles. À cette maîtrise du produit et des concepts, s'ajoute la préparation spécifique à la certification : vous pourrez accéder gratuitement à un examen blanc en ligne, destiné à vous entraîner dans des conditions proches de celles de l'épreuve.

[BOLT, 1980] — Richard A. BOLT, 1980 — “Put-that-there: Voice and Gesture at the Graphics Interface”, Association for the Computing Machinery SIGGRAPH Computer Graphics, vol. 14, n°3, pp. 262–270 — ISSN: 0097-8930.

Recent technological advances in connected-speech recognition and position sensing in space have encouraged the notion that voice and gesture inputs at the graphics interface can converge to provide a concerted, natural user modality. The work described herein involves the user commanding simple shapes about a large-screen graphics display surface. Because voice can be augmented with simultaneous pointing, the free usage of pronouns becomes possible, with a corresponding gain in naturalness and economy of expression. Conversely, gesture aided by voice gains precision in its power to reference.

[BOTTE et al., 1989] — Marie-Claire BOTTE, Georges CANEVET, Laurent DEMANY et Christel SORIN, 1989 — « Psychoacoustique et perception auditive », 1^{re} éd., EMINTER/INSERM/SFA/CNET, Paris, 144 pages — ISBN : 2-85206-534-7.

Ce volume vient combler une lacune parmi les ouvrages français, dont très peu sont consacrés à la psychoacoustique. Les connaissances concernant les attributs élémentaires de la perception auditive — intensité subjective, hauteur, localisation —, ainsi que celles qui portent sur la perception de la parole continue, sont exposées de façon détaillée et didactique dans cet ouvrage. Un bilan de l'état des discussions théoriques et des enjeux des recherches actuelles accompagne l'exposé de chacun des thèmes traités.

[BROCA, 2018] — Sébastien BROCA, 2018 — « Utopie du logiciel libre », 2^e éd., Essais. LE PASSAGER CLANDESTIN, Paris, 380 pages — ISBN : 978-2-3693-5097-2.

Né dans les années 1980 de la révolte de hackers contre la privatisation du code informatique, le mouvement du logiciel libre a peu à peu diffusé ses valeurs et ses pratiques à d'autres domaines, dessinant une véritable « utopie concrète ». Celle-ci a fait sienne plusieurs exigences : bricoler nos technologies au lieu d'en être les consommateurs sidérés, défendre la circulation de l'information contre l'extension des droits de propriété intellectuelle, lier travail et accomplissement personnel en minimisant les hiérarchies. De GNU/Linux à Wikipédia, de la licence GPL aux Creative Commons, des ordinateurs aux imprimantes 3D, ces aspirations se sont concrétisées dans des objets techniques, des outils juridiques et des formes de collaboration qui nourrissent au-

jourd’hui une nouvelle sphère des communs. Dans cette histoire du Libre, les hackers inspirent la pensée critique (d’André Gorz à la revue *Multitudes*) et les entrepreneurs open source côtoient les défenseurs des biens communs. De ce bouillonnement de pratiques, de luttes et de théories, l’esprit du Libre émerge comme un déjà là propre à encourager l’inventivité collective. Mais il est aussi un prisme pour comprendre comment, en quelques décennies, on est passé du capitalisme de Microsoft — la commercialisation de petites boîtes des biens informationnels protégés par des droits de propriété intellectuelle — au capitalisme numérique des Gafa (Google, Amazon, Facebook, Apple), fondé sur l’exploitation de nos données et la toute puissance des algorithmes.

[BUSH, 1945] — Vannevar BUSH, 1945 — “As We May Think”, *The Atlantic Monthly*, n°176, pp. 101–108. Reprinted and discussed in ACM Interactions, 3(2), March 1996, pp. 35–67.

[CALLAHAN et al., 1988] — Jack CALLAHAN, Don HOPKINS, Mark WEISER, and Ben SHNEIDERMAN, 1988 — “An Empirical Comparison of Pie vs. Linear Menus”, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI’88. Association for the Computing Machinery, Washington, D.C., USA, pp. 95–100 — ISBN: 0-201-14237-6.

▶ *Menus are largely formatted in a linear fashion listing items from the top to bottom of the screen or window. Pull down menus are a common example of this format. Bitmapped computer displays, however, allow greater freedom in the placement, font, and general presentation of menus. A pie menu is a format where the items are placed along the circumference of a circle at equal radial distances from the center. Pie menus gain over traditional linear menus by reducing target seek time, lowering error rates by fixing the distance factor and increasing the target size in Fitts’s Law, minimizing the drift distance after target selection, and are, in general, subjectively equivalent to the linear style.*

[CGET, 2016] — — Commissariat général à l’égalité des territoires CGET, éd., 2016 — *En bref (11) : CPER 2015-2020 : soutenir l’investissement dans les territoires*.

[DOWEK et al., 2013] — Gilles DOWEK, Jean-Pierre ARCHAMBAULT, Claudio CIMELLI, Benjamin WACK, Emmanuel BACCELLI, Albert COHEN et Christine Eisenbeisand Thierry VIÉVILLE, 2013 — « *Économie du logiciel libre* », Noire. Eyrolles, Paris, 342 pages — ISBN : 978-2-212-13676-0.

▶ *Enfin un véritable manuel d’informatique pour les lycéens et leurs professeurs ! Les quatre concepts de machine, d’information, d’algorithme et de langage sont au cœur de l’informatique, et l’objet de ce cours est de montrer comment ils fonctionnent ensemble. En première partie, nous apprendrons à écrire des programmes, en découvrant les ingrédients qui les constituent : l’affectation, la séquence et le test, les boucles, les types, les fonctions et les fonctions récursives. Dans la deuxième partie, on verra comment représenter les informations que l’on veut communiquer, les stocker et les transformer — textes, nombres, images et sons. On apprendra également à structurer et compresser de grandes quantités d’informations, à les protéger par le chiffrement. On verra ensuite que derrière les informations, il y a toujours des objets matériels : ordinateurs, réseaux, robots, etc., qui font partie de notre quotidien. Enfin, on s’initiera à des savoir-faire utiles au XXI^e siècle : ajouter des nombres exprimés en base deux, dessiner, retrouver une information pardichotomie, trier des informations et parcourir des graphes.*

[ÉLIE, 2009] — François ÉLIE, 2009 — « *Économie du logiciel libre* », Accès Libre. Eyrolles, Paris, 186 pages — ISBN : 978-2-212-12463-7.

▶ *Cet ouvrage profond, documenté et illustré d’exemples contemporains, permet de comprendre enfin les mécanismes qui sous-tendent la dynamique des logiciels libres et open source, mais aussi leurs implications économiques. Il met en perspective l’évolution récente des modèles économiques pratiqués, propose quelques pistes de réflexion pour en appréhender les enjeux et prendre le train en marche.*

Écrit par François ÉLIE, acteur du logiciel libre en France, président de l’ADULLACT et membre de l’AFUL, il rappelle de façon percutante ce qu’est le logiciel libre et déchiffre les paradoxes apparents qui animent ses communautés d’amateurs, d’industriels, mais aussi de clients.

François ÉLIE, agrégé de philosophie, informaticien amateur et élu local UMP/LR (Angoulême), est co-fondateur et président de l’ADULLACT, association qui promeut depuis 2002 le développement mutualisé de logiciels libres métier. À ce titre, il est parmi les précurseurs de la cause des clients dans l’Open Source et intervient régulièrement (conférences, tables rondes, keynotes) sur les sujets qui touchent au logiciel libre ou aux nouvelles technologies dans le secteur public. Promoteur de la mutualisation par la demande et du développement de logiciels libres sur fonds publics, il a monté en mars 2003 la première forge publique dédiée aux applications métier destinées aux services publics (adullact.net, mars 2003) et inspiré en 2005 la proposition 35 de la déclaration du Sommet des Villes et des Pouvoirs Locaux visant à privilégier l’usage et le développement de logiciels libres dans les investissements des villes et régions. Lorsqu’il était en charge des Nou-

velles Technologies à Angoulême, il a fait distribuer aux angoumoisins en mars 2002 un CD-Rom de logiciels libres (26 000 exemplaires).

[ENGELBART, 1962] — Douglas C. ENGELBART, 1962 — “*Augmenting Human Intellect: A Conceptual Framework*”, Summary report project n°3578. Contract AF 49(638)-1024. Menlo Park, California: Stanford Research Institute. 134 pages.



This is an initial summary report of a project taking a new and systematic approach to improving the intellectual effectiveness of the individual human being. A detailed conceptual framework explores the nature of the system composed of the individual and the tools, concepts, and methods that match his basic capabilities to his problems. One of the tools that shows the greatest immediate promise is the computer, when it can be harnessed for direct on-line assistance, integrated with new concepts and methods.

[ENGELBART, 1968] — Douglas C. ENGELBART, 1968 — *A Research Center for Augmenting Human Intellect (AUGMENT 3954)*, 90 minutes video recording of live online hypermedia demonstration/presentation at the Fall Joint Computer Conference, San Francisco, CA, December 9, 1968. Stanford Research Institute.

[ENGELBART and ENGLISH, 1968] — Douglas C. ENGELBART and William K. ENGLISH, 1968 — “*A Research Center for Augmenting Human Intellect (AUGMENT 3954)*”, AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference. Vol. 33. Republished with articles n°4, 21, and 23 in *Computer Supported Cooperative Work: A Book of Readings*, Irene Greif [Ed.], Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988, pp. 81-105. Also republished with article n°3 in *New Media Reader*, Noah Wardrip-Fruin and Nick Montfort [Ed.], The MIT Press, 2003, Chapters 8 and 16. American Federation of Information Processing Societies, San Francisco, CA, pp. 395-410.



This paper describes a multisponsor research center at Stanford Research Institute in man-computer interaction.

For its laboratory facility, the Center has a time-sharing computer (65K, 24-bit core) with a 4.5 megabyte swapping drum and a 96 megabyte file-storage disk. This serves twelve CRT work stations simultaneously.

Special hardware completely removes from the CPU the burden of display refresh and input sampling, even though these are done directly out of and into core.

The display in a user's office appears on a high-resolution (875-line) commercial television monitor, and provides both character and vector portrayals. A relatively standard typewriter keyboard is supplemented by a five-key handset used (optionally) for entry of control codes and brief literals. An SRI cursor device called the "mouse" is used for screen pointing and selection.

The "mouse" is a hand-held X-Y transducer usable on any flat surface; it is described in greater detail further on.

Special-purpose: high-level languages and associated compilers provide rapid, flexible development and modification of the repertoire of service functions and of their control procedures (the latter being the detailed user actions and computer feedback involved in controlling the application of these service functions).

User files are organized as hierarchical structures of data entities, each composed of arbitrary combinations of text and figures. A repertoire of coordinated service features enables a skilled user to compose, study, and modify these files with great speed and flexibility, and to have searches, analyses data manipulation, etc. executed. In particular, special sets of conventions, functions, and working methods have been developed to aid programming, logical design, documentation, retrieval, project management, team interaction, and hard copy production.

[FOGEL, 2012] — Karl FOGEL, 2012 — « *Produire du logiciel libre* », 2^e éd., FRAMABOOK, Paris, 334 pages — ISBN : 978-2-9539187-5-5.



*Nous sommes heureux de publier cette traduction française de *Producing Open Source Software. How to Run a Successful Free Software Project*, dont le texte original est accessible sur ProducingOSS.com, dans de nombreuses autres langues. Grâce à son expérience du développement Open Source, Karl FOGEL détaille dans ce livre les éléments stratégiques les plus importants de la conduite de projet comme la bonne pratique du courrier électronique et le choix du gestionnaire de versions, mais aussi la manière de rendre cohérents et harmonieux les rapports humains tout en ménageant les susceptibilités... En somme, dans le développement Open Source peut-être plus qu'ailleurs, et parce qu'il s'agit de trouver un bon équilibre entre coopération et collaboration, les qualités humaines sont aussi décisives que les compétences techniques.*

[GLEIZES et BURRICAND, 2015] — François GLEIZES et Carine BURRICAND, 2015 — « *De plus en plus de foyers équipés de biens électroniques* », *Insee Focus* (20). Sous la dir. de INSEE,

[GRAVELEAU, 2016] — Séverin GRAVELEAU, 272016 — « *L'avenir appartient aux diplômés du numérique* », *Le Monde Campus*.

[HARTMANN et al., 2011] — Werner HARTMANN, Michael NÄF et Raimond REICHERT, 2011 — « *Enseigner l'informatique* », Collection Iris. Springer Verlag France, Paris, 176 pages — ISBN : 978-2-8178-0261-9.



Ce livre a pour but d'expliquer ce qu'un enseignement de l'informatique devrait être, ce qu'il n'est pas et ce qu'il ne doit surtout pas devenir. Il montre en particulier qu'il faut éviter la confusion entre le contenu d'un enseignement et le recours aux TIC; il explore méthodiquement certaines questions comme : existe-t-il une didactique de l'informatique ? une pédagogie vaut-elle mieux qu'une autre ? comment gérer la diversité au sein des groupes qui apprennent l'informatique ? pourquoi et comment aborder l'abstraction ? etc.

Chaque chapitre commence par une situation vécue exemplaire des difficultés rencontrées quotidiennement dans l'enseignement de l'informatique. Il s'en dégage de multiples problématiques auxquelles les auteurs apportent des réponses concrètes dans un style alerte et vivant.

Ce livre rendra les plus grands services à tous ceux qui doivent organiser ou prendre en charge des enseignements d'informatique. Il rappelle que des conditions matérielles adaptées sont un prélude à un enseignement de qualité, et propose une pédagogie pour mettre en place des cours efficaces et cohérents.

[HILLER and RUIZ, 1971a] — Lejaren A. HILLER and Pierre M. RUIZ, 1971a — “*Synthesizing musical sounds by solving the wave equation for vibrating objects*”, *Journal of the Audio Engineering Society*, vol. 19, n°6/7, pp. 462–470 (Part I) and 542–551 (Part II) — ISSN: 1549-4950.

[JEAN, 2011] — Benjamin JEAN, 2011 — « *Option libre. Du bon usage des licences libres* », Publié à la fois sous Licence Art Libre, GNU-FDL et Creative Commons By-Sa, ce texte est disponible au téléchargement, FramaSoft, Paris, 307 pages — ISBN : 978-1-4716-1234-3.



À l'heure où les manœuvres politiques et lobbyistes cherchent à étendre et renforcer les droits de propriété intellectuelle, un nouveau système construit sur la base des licences libres vient concurrencer et devancer le précédent en termes d'innovation et de création. Dans leurs fondements juridiques, les licences libres interrogeant à plus d'un titre les usages de la propriété intellectuelle en cours depuis des siècles. En moins de trente ans, le mouvement du Libre a réalisé une telle révolution, à la fois technique et culturelle, que le nombre de licences n'a cessé d'augmenter, formalisant de multiples manières les rapports entre les auteurs, les utilisateurs et l'œuvre.

Quelle stratégie adopter dans le choix d'une licence et comment concilier ce choix avec un modèle économique ? Quelles sont les compatibilités entre les licences et comment envisager l'équilibre entre les droits existants éprouvés (droit d'auteur, brevets, etc.) et la permissivité propre au Libre ? Dans cet ouvrage documenté, objectif et pédagogique, Benjamin Jean élaboré une base saine et pérenne de discussions et d'échanges entre tous les acteurs du Libre. Se livrant à un véritable inventaire des pratiques juridiques dans ce domaine, l'auteur nous permet de les appréhender finement et encourage à les perfectionner et les porter dans d'autres secteurs.

Après une présentation du cadre légal associé aux créations de l'esprit, initialement conçu comme un système en équilibre, l'ouvrage plonge le lecteur dans ce nouveau paradigme des licences libres. Tout en évitant la simple exposition de règles et de normes, c'est de manière méthodique que seront abordées les notions juridiques à la base des nouveaux rapports entre les acteurs. Une sérieuse analyse de la maturation de ce système et un repérage des bons réflexes et des principaux écueils, permettront enfin d'aborder une étude pratique et éclairée des quelques licences libres les plus utilisées.

[JOHNSON et al., 1989] — Jeff A. JOHNSON, Teresa L. ROBERT, William VERPLANK, David C. SMITH, Charles IRBY, Marian BEARD, and Kevin MACKEY, 1989 — “*The Xerox 'Star': A Retrospective*”, *IEEE Computer*, vol. 22, n°9, pp. 11–26. Reprinted in Ron M. Baecker, Jonathan Grudin, William Buxton, Saul Greenberg (Eds.), *Human Computer Interaction: Toward the Year 2000*, Morgan Kaufman Publishers, 1995 — ISSN: 0018-9162.



A description is given of the Xerox 8010 Star information system, which was designed as an office automation system. The idea was that professionals in a business or organization would have workstations on their desks and would use them to produce, retrieve, distribute, and organize documentation, presentations, memos, and reports. All of the workstations in an organization would be connected via Ethernet and would share access to file servers, printers, etc. The distinctive features of Star are identified, and changes to the original design are examined. A history of Star development is included. Some lessons learned from designing Star are related.

[KRUEGER, 1991] — Myron W. KRUEGER, 1991 — “*Artificial Reality II*”, 2nd ed., First ed. published in 1983, Addison-Wesley Professional, Boston, 304 pages — ISBN: 978-0-2015-2260-0.

➡ *Responsive environments, computer simulation, and data gloves are some of the ways in which people can explore artificial (or virtual) worlds for business, education, and entertainment. Artificial Reality II, a revised edition of the 1983 classic by pioneer researcher Myron Krueger, presents a view of our future interaction with machines, when computer systems will sense our needs and respond to them. In the book the author describes his personal odyssey through the development of artificial reality technology, from the original vision through current applications and beyond, to his vision of artificial reality as a paradigm for future computer-human interaction.*

[KRUEGER and WILSON, 1985] — Myron W. KRUEGER and Stephen WILSON, 1985 — “‘VIDEO-PLACE’: A Report from the Artificial Reality Laboratory”, *Leonardo*, vol. 18, n°3, pp. 145–151 — ISSN: 0024-094X.

➡ *The author argues that one of the computer’s most unique features, its ability to respond in real-time, has yet to be fully exploited. For the past 16 years he has been creating an interactive computer medium in which the computer perceives a participant’s actions and responds in real-time with visual and auditory displays. He describes his conceptual discovery process as well as work in progress.*

[KURTBACH and BUXTON, 1993] — Gordon KURTENBACH and William BUXTON, 24-Apr. 29, 1993 — “The Limits of Expert Performance Using Hierarchic Marking Menus”, *Proceedings of the INTERACT’93 and CHI’93 Conference on Human Factors in Computing Systems*. INTERCHI’93. Association for the Computing Machinery, Amsterdam, The Netherlands, pp. 482–487 — ISBN: 0-89791-575-5.

➡ *A marking menu allows a user to perform a menu selection by either popping-up a radial (or pie) menu, or by making a straight mark in the direction of the desired menu item without popping-up the menu. A hierarchic marking menu uses hierarchic radial menus and “zig-zag” marks to select from the hierarchy.*
This paper experimentally investigates the bounds on how many items can be in each level, and how deep the hierarchy can be, before using a marking to select an item becomes too slow or prone to errors.

[LALLEMENT, 2015] — Michel LALLEMENT, 2015 — « *L’âge du faire. Hacking, travail, anarchie* », La Couleur des idées. Le Seuil, Paris, 448 pages — ISBN : 978-2-02-119049-6.

➡ *De nouveaux lieux de conception, de production et de collaboration voient aujourd’hui le jour un peu partout dans le monde. Équipés de machines industrielles comme des plus récents équipements informatiques, les hackers inventent un nouveau modèle d’activité : le faire (make). À distance des exigences imposées par le marché et les grandes organisations bureaucratiques, les membres des hackerspaces et autres laboratoires de fabrication font du travail une fin en elle-même, sans que quiconque n’impose d’objectifs, de délais, de contraintes... Juste l’envie de faire pour soi.*

Fruit d’une enquête ethnographique menée dans la région de San Francisco, là où les chantres de la contre-culture libertaire côtoient les entrepreneurs de la Silicon Valley, ce livre plonge au cœur du mouvement faire. Il en décrit les origines historiques ainsi que ses multiples impacts sur l’économie et la société. Michel Lallement a partagé la vie des hackers, les a regardé inventer, bidouiller et s’organiser au quotidien dans des communautés frottées, pour certaines d’entre elles, aux principes de l’anarchisme. Il les a fait raconter et expliquer leurs vie, leurs choix, leurs idées. En expérimentant une utopie concrète, les hackers font plus qu’imaginer une autre manière de travailler. C’est une nouvelle grammaire du vivre ensemble que, sous nos yeux, ils sont en train de composer.

Michel Lallement est professeur du Conservatoire national des arts et métiers (CNAM), titulaire de la chaire d’Analyse sociologique du travail, de l’emploi et des organisations et membre du Laboratoire interdisciplinaire pour la sociologie économique (CNRS). Il est l’auteur de nombreux ouvrages de sociologie du travail.

[LANGTANGEN, 2016] — Hans Petter LANGTANGEN, 2016 — “*A Primer on Scientific Programming with Python*”, 5th ed., Texts in Computational Science and Engineering. SPRINGER, Berlin, 922 pages — ISBN: 978-3-662-49886-6.

➡ *The book serves as a first introduction to computer programming of scientific applications, using the high-level Python language. The exposition is example and problem-oriented, where the applications are taken from mathematics, numerical calculus, statistics, physics, biology and finance. The book teaches “Matlab-style” and procedural programming as well as object-oriented*

programming. High school mathematics is a required background and it is advantageous to study classical and numerical one-variable calculus in parallel with reading this book. Besides learning how to program computers, the reader will also learn how to solve mathematical problems, arising in various branches of science and engineering, with the aid of numerical methods and programming. By blending programming, mathematics and scientific applications, the book lays a solid foundation for practicing computational science.

[LAZARD et MOUNIER-KUHN, 2019] — Emmanuel LAZARD et Pierre-Éric MOUNIER-KUHN, 2019 — « *Histoire illustrée de l'informatique* », 2^e éd., EDP Sciences, Paris, 240 pages — ISBN : 978-2-7598-2375-8.

☞ *Comment fut inventé l'ordinateur ? Comment notre monde s'est-il numérisé ? Qui sont les héros de cette aventure ? Comment s'inscrit-elle dans l'évolution de l'humanité ? Connaitre l'histoire de l'informatique relève désormais de la culture générale. De la machine d'Anticythère au cyberespionnage et aux Big Data, des cartes perforées à l'Internet, des tabulatrices aux tablettes, ce livre vous propose un voyage dans le temps, une archéologie de notre environnement numérique. Laissez-vous emporter dans cette lecture fabuleuse, où chacun retrouvera ou découvrira ces merveilleuses créations de l'esprit humain.*

[LE GOFF, 2019] — Vincent LE GOFF, 2019 — « *Apprenez à programmer en Python* », 3^e éd., Collection OpenClassrooms. EYROLLES, Paris, 504 pages — ISBN : 978-2-212-67871-0.

[LE MONDE, 2016] — Collectif LE MONDE, 72016 — « *La numérisation facteur d'exclusion pour ceux qui cumulent précarité sociale et numérique* », *Le Monde*.

[LESSIG, 2004] — Lawrence LESSIG, 2004 — “Free Culture”, Traduction en français 2015, THE PENGUIN PRESS, New York, 346 pages — ISBN: 978-2-3693-5097-2.

[MARQUET et al., 2019] — Kevin MARQUET, Françoise BERTHOUD et Jacques COMBAZ, 2019 — « *Introduction aux impacts environnementaux du numérique* », 1024 — Bulletin de la Société informatique de France, n°13, pp. 85-97.

☞ *Notre civilisation fait aujourd'hui face à plusieurs menaces majeures qui sont le résultat de l'activité humaine : le réchauffement climatique, l'érosion de la bio-diversité qui est considérée comme la sixième extinction de masse des espèces, l'épuisement progressif des ressources naturelles, la dégradation des sols, etc. Dans le cadre de la COP21 de 2015 et des accords de Paris, les états se sont engagés à réduire les émissions de gaz à effet de serre afin de maintenir d'ici 2100 la hausse de la température mondiale en dessous de 2oC par rapport à l'ère pré-industrielle, ce qui implique en particulier de décarboner massivement et sans plus attendre notre énergie, mais aussi de maîtriser nos besoins énergétiques. Comme nous le verrons dans ce qui suit, le numérique n'est pas à ce jour la révolution attendue pour décarboner ou dématérialiser notre économie, mais bien au contraire un poste de consommation supplémentaire, voire un amplificateur des impacts environnementaux des autres secteurs de l'économie. À partir de ce constat, nous proposerons quelques pistes de réflexion et d'action dans le domaine de la recherche et de la formation pour réduire les impacts négatifs du numérique.*

[MONNIN, 2013] — Alexandre MONNIN, 2013 — « *Vers une philosophie du Web : le Web comme devenir-artefact de la philosophie (URLs, tags, ontologie (s) et ressources)* », Sciences de l'Homme et Société / Philosophie. Université Panthéon-Sorbonne, Paris I. eprint : <https://tel.archives-ouvertes.fr/tel-00879147>.

☞ *Cette thèse entend prendre acte de l'importance du Web d'un point de vue philosophique. Importance double : à la fois comme objet de recherche, qui, dans le sillage du Web Sémantique et de l'architecture du Web, à des titres divers, entre en résonance évidente avec les problématiques classiques de la métaphysique et de la philosophie du langage. Dans cette perspective, nous étudions quelques-uns de ses composants principaux (URL, ressources, tags, etc.). En parallèle, nous soulignons son importance au regard de la question du devenir de la philosophie elle-même. En effet, le travail entrepris ne s'est nullement contenté de projeter les concepts à priori d'une philosophia perennis. Il a consisté, au contraire, à interroger les architectes du Web eux-mêmes pour faire émerger leur métaphysique empirique, en observant les controverses qu'elle a suscitées. Prendre acte de la portée ontogénique d'une pratique telle que « l'ingénierie philosophique », selon l'expression de Tim Berners-Lee, pensée ici comme la production de nouvelles distinctions dans un monde en train de se faire, nous conduit à mener une réflexion plus vaste sur la nature de l'objectivation. Celle-ci rejoint en fin de compte des préoccupations politiques, dans la perspective de l'établissement d'un monde commun, auquel le Web participe activement.*

[MYERS, 1998] — Brad A. MYERS, 1998 — “*A Brief History of Human Computer Interaction Technology*”, ACM Interactions, vol. 5, n°2, pp. 44–54 — ISSN: 1072-5520.

 This article summarizes the historical development of major advances in human computer interaction technology, emphasizing the pivotal role of university research in the advancement of the field. Copyright 1996 – Carnegie Mellon University.

A short excerpt from this article appeared as part of "Strategic Directions in Human Computer Interaction," edited by Brad Myers, Jim Hollan, Isabel Cruz, ACM Computing Surveys, 28(4), December 1996.

This research was partially sponsored by NCCOSC under Contract N°N66001-94-C-6037, Arpa Order N°B326 and partially by NSF under grant number IRI-9319969.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NCCOSC or the U.S. Government.

[NELSON, 1993] — Theodor H. NELSON, 1993 — "Literary Machines 93.1. The report on, and of, project Xanadu concerning word processing, electronic publishing, hypertext, thinker-toys, tomorrow's intellectual revolution, and certain other topics including knowledge, education and freedom", Sausalito, California. 288 pages.

[PALOQUE-BERGES et al., 2013] — Camille PALOQUE-BERGES, Christophe MASUTTI, Pascal RUBERT, Valérie SCHAFER, Christopher KELTY, Stéphane RIBAS, Patrick GUILLAUD, Stéphane UBÉDA, Nicolas JULLIEN, Jean-Benoît ZIMMERMAN, Pierre-Amiel GIRAUD, Damien DJAOUTI, Thibaud HULIN, Sébastien BROCA, Benjamin JEAN, Primavera DE FILIPPI, Isabelle RAMADE, Robert VISEUR, Séverine GIORDAN, Adrienne ALIX, Jonathan CHIBOIS, Alexandre HOCQUET, Gérard GIRAUDON, Jean-Luc ARCHIMBAUD, François ÉLIE et Jean-Pierre ARCHAMBAULT, 2013 — « *Histoires et cultures du Libre. Des logiciels partagés aux licences échangées* », sous la dir. de Camille PALOQUE-BERGES et Christophe MASUTTI, un chapitre supplémentaire et deux textes additionnels sont disponibles au téléchargement, FRAMABOOK, Paris, 580 pages — ISBN : 978-2-9539187-9-3.

 Fruit de la collaboration inédite d'auteurs provenant d'horizons disciplinaires différents, par des approches thématiques et des études de cas, cet ouvrage propose une histoire culturelle du Libre non seulement à travers l'histoire de l'informatique, mais aussi par les représentations sociales, philosophiques, juridiques et économiques qu'a cristallisées le mouvement du logiciel libre jusqu'à nos jours.

A l'aide de multiples clés d'analyse, et sans conception partisane, ce livre dresse un tableau des bouleversements des connaissances et des techniques que ce mouvement a engendrés. Le lecteur saura trouver dans cette approche ambitieuse et prospective autant d'outils pour mieux comprendre les enjeux de l'informatique, des réseaux, des libertés numériques, ainsi que l'impact de leurs trajectoires politiques dans la société d'aujourd'hui.

[PERKINS et al., 1997] — Roderick PERKINS, Dan KELLER, and Frank LUDOLPH, 1997 — "Inventing the Lisa User Interface", ACM Interactions, vol. 4, n°1, pp. 40–53 — ISSN: 1072-5520.

[PERRET et JACQUEMELLE, 2014] — Xavier PERRET et Guy JACQUEMELLE, 2014 — « *Big Data. Le cinéma avait déjà tout imaginé!* », 1^{re} éd., Tout savoir sur... Éditions Kawa, Paris, 240 pages — ISBN : 978-2-36778-015-3.

 Les films de science-fiction aiment à imaginer ce que sera l'humanité dans les décennies à venir et sont parfois annonciateurs des bouleversements qui nous attendent. Souvenez-vous ! Terminator, La Mouche, La nuit des morts-vivants, Minority Report, Matrix, Dead Zone... Le déluge de données, la téléportation, l'alerte sur des crimes qui n'ont pas encore été commis, le voyage dans le temps et la résurgence du passé, l'affichage et l'utilisation d'images en réalité augmentée : ces films ont en commun d'avoir esquissé ce qui est devenu réalité.

Sans même nous en rendre compte, nous générerons de plus en plus de données : toutes nos recherches sur Internet, nos photos, vidéos et « likes » postés sur les réseaux sociaux, nos tweets, nos mails représentent d'immenses volumes de données et laissent des traces. Des millions de citoyens se déplacent également avec le GPS de leur téléphone dans leurs poches et fournissent nombre d'informations sur leurs habitudes et leurs comportements. Ces milliards d'empreintes numériques sont devenues si volumineuses qu'on les appelle les Big Data. Elles constituent un nouvel eldorado pour ceux qui savent les analyser, les comprendre et découvrir les secrets qu'elles recèlent.

Au travers de 11 films, cet ouvrage évoque des situations qui, il y a quelques années, nous paraissaient impossibles ou tellement lointaines. Il prouve, au travers de nombreux exemples, qu'aujourd'hui déjà, nous sommes allés au-delà de ce que l'on nous promettait et que ce mouvement ne peut que s'accélérer. Pour le meilleur, mais aussi parfois, pour le pire. Ce livre vous offre des clés pour comprendre ce nouveau monde, tirer profit de ces nouvelles opportunités et vous donne de précieux conseils pour protéger votre « patrimoine » numérique.

[PINTSCHER et al., 2013] — Lydia PINTSCHER, Armijn HEMEL, Markus KROËTZSCH, Evan PRODROMOU, Felipe ORTEGA, Leslie HAWTHORN, Kevin OTTENS, Jeff MITCHELL, Austin APPEL, Thiago MADEIRA, Henri BERGIUS, Kai BLIN, Ara PULIDO, Andre KLAPPER, Jonathan "Duke" LETO, Atul JHA, Rich BOWEN, Anne GENTLE, Shaun McCANCE, Runa BHATTACHARJEE, Guillaume PAUMIER, Federico Mena QUINTERO, Máirín Duffy STRODE, Eugene TROUNEV, Robert KAYE, Jono BACON, Alexandra LEISSE, Jonathan RIDDELL, Thorn MAY, Vincent UNTZ, Stuart JARVIS, Jos POORTVLIET, Sally KHUDAIRI, Nóirín PLUNKETT, Dave NEARY, Gareth J.GREENAWAY, Selena DECKELMANN, Till ADAM, Frank KARLITSCHER, Carlo DAFFARA, Till JAEGER et Shane COUGHLAN, 2013 — « *Libres conseils. Ce que nous aurions aimé savoir avant de commencer* », sous la dir. de Lydia PINTSCHER, FRAMABOOK, Paris, 373 pages — ISBN : 978-10-92674-04-0.



Les projets de logiciels libres bousculent le monde du logiciel parce qu'ils sont menés de façon originale, par des utilisateurs convaincus et impliqués. Chacun apporte quelque chose au mouvement à sa manière, selon ses aptitudes et ses connaissances. Cet engagement personnel et la puissance de la collaboration sur Internet donnent sa force particulière au logiciel libre.
Ce livre apporte 42 réponses à la question : « Qu'auriez-vous aimé savoir quand vous avez commencé à contribuer ? »
Les 42 auteurs font part de leur expérience et décrivent les nombreux talents différents qui doivent se combiner comme les briques de construction du projet : l'art du code bien sûr, mais aussi le design, les compétences de documentation et de traduction, le marketing et bien d'autres domaines... Si vous êtes débutant, cet ouvrage vous donnera une longueur d'avance. Et si vous avez contribué depuis un certain temps déjà, il vous donnera un aperçu précieux d'autres domaines et projets.

[RASKIN, 2000] — Jef RASKIN, 2000 — “*The Humane Interface. New Directions for Designing Interactive Systems*”, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 256 pages — ISBN: 978-0-2013-7937-2.



*The honeymoon with digital technology is over: millions of users are tired of having to learn huge, arcane programs to perform the simplest tasks; fatigued by the pressure of constant upgrades, and have had enough of system crashes. In *The Humane Interface*, Jef Raskin — the legendary, controversial creator of the original Apple Macintosh project — shows that there is another path. Raskin explains why today's interface techniques lead straight to a dead end, and offers breakthrough ideas for building systems users will understand — and love. Raskin reveals the fundamental design failures at the root of the problems so many users experience; shows how to understand user interfaces scientifically and quantitatively; and introduces fundamental principles that should underlie any next-generation user interface. He introduces practical techniques designers can use to improve their productivity of any product with an information-oriented human-machine interface, from personal computers to Internet appliances and beyond. The book presents breakthrough solutions for navigation, error management, and more, with detailed case studies from Raskin's own work. For all interface design programmers, product designers, software developers, IT managers, and corporate managers.*

"Deep thinking is rare in this field where most companies are glad to copy designs that were great back in the 1970s. The Humane Interface is a gourmet dish from a master chef. Five mice!" —Jakob Nielsen, Nielsen Norman Group, author of "Designing Web Usability: The Practice of Simplicity"

This unique guide to interactive system design reflects the experience and vision of Jef Raskin, the creator of the Apple Macintosh. Other books may show how to use today's widgets and interface ideas effectively. Raskin, however, demonstrates that many current interface paradigms are dead ends, and that to make computers significantly easier to use requires new approaches. He explains how to effect desperately needed changes, offering a wealth of innovative and specific interface ideas for software designers, developers, and product managers.

The Apple Macintosh helped to introduce a previous revolution in computer interface design, drawing on the best available technology to establish many of the interface techniques and methods now universal in the computer industry. With this book, Raskin proves again both his farsightedness and his practicality. He also demonstrates how design ideas must be built on a scientific basis, presenting just enough cognitive psychology to link the interface of the future to the experimental evidence and to show why that interface will work.

Raskin observes that our honeymoon with digital technology is over: We are tired of having to learn huge, arcane programs to do even the simplest of tasks; we have had our fill of crashing computers; and we are fatigued by the continual pressure to upgrade. The Humane Interface delivers a way for computers, information appliances, and other technology-driven products to continue to advance in power and expand their range of applicability, while becoming free of the hassles and obscurities that plague present products.

[RAYMOND, 2001] — Eric S. RAYMOND, 2001 — “*The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*”, 2nd ed., First edition October

1999, Revised edition January 2001, O'REILLY, Sebastopol, CA, 258 pages — ISBN: 978-0-596-00108-7.

☞ *The Cathedral and the Bazaar* is a must read for anyone who cares about the future of the computer industry or the dynamics of the information economy. Eric Raymond's persuasive manifesto defining the open source revolution has helped propel this collaborative approach to software development into the mainstream. As corporations continue to adopt technologies like Linux, Apache, and Perl, open source is proving to be the competitive advantage in the Internet Age. — Eric S. Raymond is an observer-participant anthropologist in the Internet hacker culture. His research has helped explain the decentralized open-source model of software development that has proven so effective in the evolution of the Internet. His own software projects include one of the Internet's most widely-used email transport programs.

[ROADS, 2007] — Curtis ROADDS, 2007 — « *L'audionumérique. Musique et informatique* », 2^e éd., Traduction et adaptation française : Jean de REYDELLET, DUNOD, Paris, 672 pages — ISBN : 978-2-1005-1249-2.

☞ *La musique électronique et informatique nécessite des outils sophistiqués, une grande habileté dans le travail de studio et une parfaite connaissance des techniques audionumériques.* Devenu au fil des années le livre de référence, *L'audionumérique* s'adresse à tous les techniciens et ingénieurs du son ainsi qu'aux musiciens compositeurs qui souhaitent découvrir ce domaine ou se perfectionner. Largement illustré de schémas, captures d'écran et photos, cet ouvrage présente l'ensemble des aspects théoriques, pratiques et esthétiques de la synthèse et du traitement d'un signal sonore dans un environnement numérique :

- ▶ la première partie traite de l'environnement et des outils : mixage, filtres numériques, spatialisation et réverbération, hauteur et rythme, méthodes d'analyse spectrale, MIDI, etc.
- ▶ la seconde partie aborde l'échantillonnage puis décrit dans le détail tous les types de synthèse sonore : additive, granulaire, pulsar, par modulation, formantique, graphique, etc.

[ROHAUT, 2011] — Sébastien ROHAUT, 2011 — « *Préparation à la certification Linux LPIC-1. Examens LPI 101 et LPI 102* », 3^e éd., Certifications. Éditions ENI, Saint Herblain, 661 pages — ISBN : 978-2-7460-7007-3.

☞ *Les examens LPI 101 et LPI 102 sont les deux examens qui permettent d'obtenir la certification LPIC-1 Junior Level Linux Professionnal. Ce programme de certification du Linux Professional Institute est reconnu par les recruteurs qui voient dans cette certification un pré-requis à l'embauche ou à l'accès à un poste d'administrateur. Les examens LPI 101 et 102 prouvent aux professionnels que vous maîtrisez les bases de l'administration système Linux quelle que soit la distribution : l'installation et la configuration complète d'un poste de travail ou d'un serveur et de tous les services essentiels associés, tant systèmes que réseaux. Cette nouvelle édition du livre tient compte des dernières nouveautés Linux (noyau 3.0, IPv6, nouvelles distributions, nouveau boot loader, etc.). Pour vous aider à préparer efficacement cette certification, ce livre couvre tous les objectifs officiels de la dernière version de l'examen (mise en place en juillet 2012), tant d'un point de vue théorique que d'un point de vue pratique. Il a été rédigé en français (il ne s'agit pas d'une traduction) par un formateur professionnel reconnu, Ingénieur Système et certifié Linux. Ainsi, les savoir-faire pédagogique et technique de l'auteur conduisent à une approche claire et visuelle, d'un très haut niveau technique. Chapitre par chapitre, vous pourrez valider vos acquis théoriques (536 au total), à l'aide d'un grand nombre de questions-réponses mettant en exergue aussi bien les éléments fondamentaux que les caractéristiques spécifiques aux concepts abordés. Chaque chapitre s'achève par des travaux pratiques (47 au total), vous aurez les moyens de mesurer votre autonomie. Ces manipulations concrètes, au-delà même des objectifs fixés par l'examen, vous permettront de vous forger une première expérience significative et d'acquérir de véritables compétences techniques sur des mises en situations réelles. À cette maîtrise du produit et des concepts, s'ajoute la préparation spécifique à la certification : vous pourrez accéder gratuitement à un examen blanc en ligne, destiné à vous entraîner dans des conditions proches de celles de l'épreuve. Sur ce site, chaque question posée s'inscrit dans l'esprit de la certification et, pour chacune, les réponses sont suffisamment commentées pour contrôler et identifier vos ultimes lacunes. À vous de juger quand vous serez prêt pour l'examen final !*

[ROSSING, AKHATOV, et al., 2007] — Thomas D. ROSSING, Iskander AKHATOV, Yoichi ANDO, Keith ATTENBOROUGH, Whitlow W. L. AU, Kirk W. BEACH, Mack A. BREAZEALE, Antoine CHAIGNE, Perry R. COOK, James COWAN, Mark F. DAVIS, Barbrina DUNMIRE, Neville H. FLETCHER, Colin GOUGH, William M. HARTMANN, Finn JACOBSEN, Yang-Hann KIM, William A. KUPERMAN, Thomas KURZ, Werner LAUTERBORN, Björn LINDBLOM, George C. MALING, Nils-Erik MOLIN, Brian C. J. MOORE, Alan D. PIERCE, Philippe Roux, Johan SUNDBERG, Gregory W. SWIFT, George S. K. WONG, and Eric D. YOUNG, 2007 — “*Springer Handbook of Acoustics*”,

ed. by Thomas D. ROSSING, 1st ed., Springer Handbooks. SPRINGER VERLAG, New York, 1182 pages — ISBN: 978-0-3873-0425-0.

▶ *Acoustics, the science of sound, has developed into a broad interdisciplinary field encompassing the academic disciplines of physics, engineering, psychology, speech, audiology, music, architecture, physiology, neuroscience, and others. The Springer Handbook of Acoustics is an unparalleled modern handbook reflecting this richly interdisciplinary nature edited by one of the acknowledged masters in the field, Thomas Rossing. Researchers and students benefit from the comprehensive contents spanning: animal acoustics including infrasound and ultrasound, environmental noise control, music and human speech and singing, physiological and psychological acoustics, architectural acoustics, physical and engineering acoustics, signal processing, medical acoustics, and ocean acoustics. This handbook reviews the most important areas of acoustics, with emphasis on current research. The authors of the various chapters are all experts in their fields. Each chapter is richly illustrated with figures and tables. The latest research and applications are incorporated throughout, e.g. computer recognition and synthesis of speech, physiological acoustics, psychological acoustics, thermoacoustics, diagnostic imaging and therapeutic applications and acoustical oceanography.*

[ROSSING, MOORE, et al., 2002] — Thomas D. ROSSING, Richard F. MOORE, and Paul A. WHEELER, 2002 — “*The Science of Sound*”, 3rd ed., ADDISON-WESLEY, San Francisco, 808 pages — ISBN: 0-8053-8565-7.

▶ *The Science of Sound is widely recognized as the leading textbook in the field. It provides an excellent introduction to acoustics for students without college physics or a strong background in mathematics. In the Third Edition, Richard Moore and Paul Wheeler join Tom Rossing in updating The Science of Sound to include a wide range of important technological developments in the field of acoustics. New exercises and review questions have been added to the end of each chapter to help students study the material.*

[ROUGIER, 2020b] — Nicolas P. ROUGIER, 2020b — “*Scientific Visualization – Python & Matplotlib. Scientific Python – Volume II*”, 1st ed., In print, Summer 2020., Bordeaux.

[RUIZ, 1970] — Pierre M. RUIZ, 1970 — “*A technique for simulating the vibrations of strings with a digital computer*”, Master’s thesis. Urbana, Illinois: Univerty of Illinois.

[SHNEIDERMAN, 1983] — Ben SHNEIDERMAN, 1983 — “*Direct Manipulation: A Step Beyond Programming Languages*”, IEEE Computer, vol. 16, n°8, pp. 57–69 — ISSN: 0018-9162.

[SMIERS et SCHIJNDEL, 2012] — Joost SMIERS et Marieke van SCHIJNDEL, 2012 — « *Un monde sanscopyright... Et sans monopole* », 2^e éd., FRAMABOOK, Paris, 334 pages — ISBN : 978-2-9539187-5-5.

▶ *Nous sommes heureux de publier cette traduction française de Producing Open Source Software. How to Run a Successful Free Software Project, dont le texte original est accessible sur ProducingOSS.com, dans de nombreuses autres langues. Grâce à son expérience du développement Open Source, Karl FOGEL détaille dans ce livre les éléments stratégiques les plus importants de la conduite de projet comme la bonne pratique du courrier électronique et le choix du gestionnaire de versions, mais aussi la manière de rendre cohérents et harmonieux les rapports humains tout en ménageant les susceptibilités... En somme, dans le développement Open Source peut-être plus qu’ailleurs, et parce qu’il s’agit de trouver un bon équilibre entre coopération et collaboration, les qualités humaines sont aussi décisives que les compétences techniques.*

[David Canfield SMITH et al., 1982] — David Canfield SMITH, Charles IRBY, Ralph KIMBALL, and Eric HARSLEM, 1982 — “*The Star User Interface: an Overview*”, Proceedings of the National Computer Conference. AFIPS'82. Association for the Computing Machinery, Houston, Texas, pp. 515–528 — ISBN: 0-88283-035-X.

▶ *In April 1981 Xerox announced the 8010 Star Information System, a new personal computer designed for office professionals who create, analyze, and distribute information. The Star user interface differs from that of other office computer systems by its emphasis on graphics, its adherence to a metaphor of a physical office, and its rigorous application of a small set of design principles. The graphic imagery reduces the amount of typing and remembering required to operate the system. The office metaphor makes the system seem familiar and friendly; it reduces the alien feel that many computer systems have. The design principles unify the nearly two dozen functional areas of Star, increasing the coherence of the system and allowing user experience in one area to apply in others.*

[STALLMAN et al., 2010] — Richard M. STALLMAN, Sam WILLIAMS et Christophe MASSUTI, 2010 — « *Richard Stallman et la révolution du logiciel libre. Une biographie autorisée* », Accès Libre. Traduction française, remaniée et augmentée, de l’ouvrage « *Free as in Free-*

dom » (2002) de Sam WILLIAMS et publié sous licence GNU-FDL par l'association FramasoFT et les Éditions Eyrolles. Disponible au téléchargement, FRAMABOOK-EYROLLES, Paris, 324 pages — ISBN : 978-2-212-12609-9.



Né en 1953, Richard Stallman est un programmeur américain hors pair considéré comme le « père » du logiciel libre. Son héritage est unanimement reconnu et son influence toujours plus grande sur nos sociétés actuelles de l'information et de la communication. Ses conférences en français débutent invariablement ainsi : « Je puis résumer le logiciel libre en trois mots : liberté, égalité, fraternité ». Cette biographie éclaire sans complaisance la vie de ce personnage autant décrié qu'encensé qui a révolutionné l'histoire du logiciel en particulier en initiant le projet GNU. À travers cet ouvrage, nous pouvons mieux connaître le parcours et les combats de cet homme hors du commun.

[STIEGLER, 2015a] — Bernard STIEGLER, 2015a — « *L'emploi est mort, vive le travail ! Entretien avec Ariel Kyrou* », Les Petits Libres. Fayard – Mille et une nuits, Paris, 120 pages — ISBN : 978-2-75-550746-1.

[STIEGLER, 2015b] — Bernard STIEGLER, 2015b — « *La Société automatique. L'avenir du travail* », La Couleur des idées. Fayard, Paris, 300 pages — ISBN : 978-2-21-368565-6.

[SUTHERLAND, 1963] — Ivan E. SUTHERLAND, 1963 — “*SketchPad: A Man-Machine Graphical Communication System*”, *Proceedings of the Joint Computer Conference*. AFIPS’63. Published in September 2003 as an electronic technical report by the University of Cambridge, Computer Laboratory, UK, ISSN: 1476-2986. Association for the Computing Machinery, Detroit, Michigan, pp. 329–346.



The Sketchpad system makes it possible for a man and a computer to converse rapidly through the medium of line drawings. Heretofore, most interaction between man and computers has been slowed down by the need to reduce all communication to written statements that can be typed; in the past, we have been writing letters to rather than conferring with our computers. For many types of communication, such as describing the shape of a mechanical part or the connections of an electrical circuit, typed statements can prove cumbersome. The Sketchpad system, by eliminating typed statements (except for legends) in favor of line drawings, opens up a new area of man-machine communication.

[SWINNEN, 2012] — Gérard SWINNEN, 2012 — « *Apprendre à programmer avec Python 3* », 3^e éd., Collection Noire. La version numérique de ce texte peut être téléchargée librement à partir du site : <http://inforef.be/swi/python.htm>, EYROLLES, Paris, 436 pages — ISBN : 978-2-2121-3434-6.



Reconnu et utilisé par les enseignants de nombreuses écoles et IUT, complété d'exercices accompagnés de leurs corrigés, cet ouvrage original et érudit est une référence sur tous les fondamentaux de la programmation : choix d'une structure de données, paramétrage, modularité, orientation objet en héritage, conception d'interface, multithreading et gestion d'événements, protocoles de communication et gestion réseau, bases de données... jusqu'à la désormais indispensable norme Unicode (le format UTF-8). On verra notamment la réalisation avec Python 3 d'une application Web interactive et autonome, intégrant une base de données SQLite. Cette nouvelle édition traite de la possibilité de produire des documents imprimables (PDF) de grande qualité en exploitant les ressources combinées de Python 2 et Python 3.

[TANENBAUM, 2008] — Andrew S. TANENBAUM, 2008 — « *Systèmes d'exploitation* », trad. par Jean-Alain HERNANDEZ et René JOLY. 3^e éd., Informatique. PEARSON FRANCE, Paris, 1070 pages — ISBN : 978-2-7440-7299-4.



Cet ouvrage présente de façon pédagogique et complète les systèmes d'exploitation. Andrew Tanenbaum, auteur de nombreux livres et concepteur de plusieurs systèmes d'exploitation (MINIX, Amoeba), expose dans un style clair des notions et des réalisations complexes. La conception de ce manuel est modulaire : les six premiers chapitres présentent les éléments fondamentaux qui composent un système d'exploitation (threads et processus, gestion de la mémoire, systèmes de fichiers, E/S). Les six chapitres suivants approfondissent des sujets tels que la sécurité, les systèmes d'exploitation multimédias, les systèmes multiprocesseurs et les techniques de conception des systèmes. La nouvelle édition de ce best-seller mondial traite des derniers développements technologiques des systèmes d'exploitation :

- ▶ *des chapitres entièrement refondus sur la sécurité informatique, les systèmes d'exploitation multimédias et les systèmes multiprocesseurs;*
- ▶ *une exploitation étendue des systèmes les plus répandus : un chapitre sur LINUX et un chapitre nouveau sur WINDOWS VISTA™;*
- ▶ *un chapitre complet sur la conception des systèmes d'exploitation;*

- ▶ une étude enrichie des interfaces graphiques, des systèmes d'exploitation multiprocesseurs, des systèmes sécurisés, des virus, des nouveaux algorithmes d'ordonnancement et de pagination;
- ▶ À la fin de la majorité des chapitres, une présentation des recherches actuelles sur le sujet;
- ▶ 450 exercices et problèmes, la plupart fortement actualisés.

[TANENBAUM et WETHERALL, 2011] — Andrew S. TANENBAUM et David J. WETHERALL, 2011 — « Réseaux », trad. par Jean-Alain HERNANDEZ et René JOLY. 5^e éd., Informatique. PEARSON FRANCE, Paris, 970 pages — ISBN : 978-2-7440-7521-6.



L'ouvrage de référence sur les réseaux, entièrement mis à jour pour inclure les technologies incontournables de demain.

Ce best-seller mondial a fait l'objet d'une révision minutieuse pour refléter les évolutions technologiques récentes, avec une attention particulière accordée aux avancées en matière de multimédia, aux réseaux 3G utilisés avec les smartphones et les terminaux de poche, aux étiquettes radio (RFID) et aux réseaux de capteurs, aux réseaux de distribution de contenu (CDN), et aux réseaux peer-to-peer. De nouveaux développements concernent également le temps réel (audio et vidéo à partir de fichiers ou en streaming), la vidéo à la demande et la téléphonie sur Internet (voix sur IP).

Andrew Tanenbaum et David Wetherall exposent de façon détaillée le fonctionnement interne des réseaux, depuis la couche physique jusqu'à la couche application. Cet ouvrage est organisé de la façon suivante :

- ▶ Couche physique (cuivre, fibre, sans fil, satellites, OFDM et CDMA, etc.);
- ▶ Couche liaison de données (principes des protocoles, détection et correction des erreurs, HDLC, PPP, etc.);
- ▶ Sous-couche MAC (Ethernet gigabit, 802.11, 802.16, Bluetooth et les RFID, sans fil à large bande, etc.);
- ▶ Couche réseau (algorithmes de routage, qualité de service en temps réel, IPv4, IPv6, etc.);
- ▶ Couche transport (programmation de sockets, UDP, TCP, RTP, contrôle de congestion, performances des réseaux, réseaux tolérants aux délais, etc.);
- ▶ Couche application (messagerie électronique, Web, PHP, DNS, streaming audio, etc.);
- ▶ Sécurité des réseaux (AES, RSA, chiffrement quantique, IPSec, sécurité du Web, sécurité 802.11 et Kerberos V5, etc.).

Ce livre présente en outre de nombreux exemples issus de l'Internet et des réseaux sans fil; il comporte plusieurs centaines d'exercices, tous entièrement corrigés sur le site compagnon.

[TISSEURAND, 2019] — Fabien TISSEURAND, 2019 — « Au cœur des réseaux. Des sciences aux citoyens », Collection Essais. Éditions Le Pommier, Paris, 168 pages — ISBN : 978-2-7465-1666-3.



Les réseaux : omniprésents dans notre vie, réelle comme virtuelle, ils jouent un rôle central pour décrypter les enjeux du monde contemporain, et ce particulièrement dans le contexte du Big Data. Entre la propagation de fake news, l'enregistrement toujours plus grand de nos traces sur le Web, leur exploitation pour déterminer nos goûts, nos habitudes et prédire notre comportement futur... il y a de quoi faire! Et pourtant, de notre utilisation des réseaux à leur compréhension, il y a un gouffre.

Ce livre grand public souhaite permettre au lecteur de reprendre le contrôle sur ces questions qui semblent si souvent techniques et hors de notre portée. En rendant accessibles les outils d'une science émergente qui s'attache à expliquer les « comportements » des réseaux, voire à les prédire... La science des réseaux vous conduira de découverte en découverte, des fameux « six degrés de séparation » censés relier toute paire d'individus au phénomène dit du « petit monde » selon lequel les amis de mes amis sont mes amis, vous comprendrez même les algorithmes de GOOGLE ou de FACEBOOK! Excusez du peu!

[TORVALDS et DIAMOND, 2001] — Linus B. TORVALDS et David DIAMOND, 2001 — « Il était une fois LINUX. L'extraordinaire histoire d'une révolution accidentelle », Traduction française, réalisée par Olivier ENGLER, de l'ouvrage « Just for Fun : The Story of an Accidental Revolutionary », OSMAN EYROLLES MULTIMÉDIA (OEM), Paris, 300 pages — ISBN : 978-2-7464-0321-5.



Il y a aujourd'hui dix ans, un étudiant finlandais nommé Linus TORVALDS s'enfermait plusieurs mois dans sa chambre, rideaux tirés, pour un long tête-à-tête avec son ordinateur. Le résultat : un nouveau système d'exploitation. Qu'allait-il en faire ? Le garder pour son usage personnel ? Le vendre à une société de logiciels ? Rien de tout cela. Linus décide de rendre le fruit de son travail librement accessible sur Internet en invitant toute personne intéressée à l'améliorer. L'UNIX libre de Linus, baptisé LINUX, était né et avec lui, une nouvelle manière de concevoir les logiciels qui allait bouleverser l'univers de l'informatique. La suite des événements fera date dans l'histoire. Linus Torvalds est devenu la figure emblématique du monde du logiciel libre. Son puissant système d'exploitation est aujourd'hui un acteur de tout premier plan de l'industrie informatique. La méthode de concep-

tion utilisée, nourrie de passion volontaire, fait de *LINUX* le plus vaste projet de création collective qui ait jamais existé. Pour la première fois, Linus TORVALDS raconte, dans ce livre, son étonnant parcours : sa fascination, tout enfant, pour la calculatrice de son grand-père, professeur de statistiques à l'Université d'Helsinki, sa première rencontre en 1981 avec un ordinateur, un Commodore VIC-20 et bien sûr les circonstances de la création du noyau *LINUX*, le composant essentiel du système *GNU/LINUX*. Linus TORVALDS est aujourd'hui l'informaticien le plus « convoité » de la planète. Il vit en Californie avec sa femme et ses filles. Il est le gourou incontesté de plusieurs millions de linuxiens convertis tout autour de la planète.

Un matin d'août 1991, du fin fond de la Finlande, Linus TORVALDS, alors âgé de 21 ans, lançait à travers le monde entier un message électronique désormais légendaire : « Hello à tous. Je travaille pour l'instant sur un système d'exploitation gratuit (ce sera un hobby et non une occupation professionnelle). J'aimerais que vous me donniez vos impressions positives ou négatives sur ce travail ». ».

[TRICOT, 2007] — André TRICOT, 2007 — « Apprentissages et documents numériques », Belin Sup Psychologie. Belin, Paris, 277 pages — ISBN : 978-2-7011-4409-2.



Les documents numériques à travers des supports comme le World Wide Web et le multimédia constituent une révolution dans l'histoire du document comparable à celle de l'invention de l'imprimerie par Gutenberg. Cette révolution en est à ses débuts mais elle modifie profondément la façon dont les hommes représentent des connaissances les enseignent et les acquièrent et sans doute le statut même de certaines connaissances humaines et de ceux qui les possèdent L'objectif de cet ouvrage est de faire la part entre les promesses et les résultats empiriques dans le domaine des documents numériques pour apprentissages Il est aussi de montrer comment deux processus cognitifs impliqués dans l'apprentissage — la compréhension et la recherche d'information — sont modifiés par les nouveaux documents et de faire le point sur les études ergonomiques qui peuvent permettre d'améliorer ces documents Enfin il s'agit de présenter les méthodes utilisées dans le domaine et de discuter de leur pertinence. Cet ouvrage est destiné aux étudiants en psychologie en ergonomie en interaction humains-machines (IHM) et en sciences du langage mais aussi aux enseignants documentalistes et à ceux des disciplines scientifiques et techniques. Il constitue un état de l'art conçu pour être utile aux concepteurs de documents pédagogiques dans le domaine de la formation professionnelle ou de l'enseignement académique



[VAN ZONEN, 2016] — Liesbet VAN ZONEN, 2016 — “Privacy concerns in smart cities”, *Government Information Quarterly*, vol. 33, n°3, pp. 472-480. This article is available under the Creative Commons CC-BY-NC-ND license and permits non-commercial use of the work as published, without adaptation or alteration provided the work is fully attributed. — ISSN: 0740624X.



In this paper a framework is constructed to hypothesize if and how smart city technologies and urban big data produce privacy concerns among the people in these cities (as inhabitants, workers, visitors, and otherwise). The framework is built on the basis of two recurring dimensions in research about people's concerns about privacy: one dimension represents that people perceive particular data as more personal and sensitive than others, the other dimension represents that people's privacy concerns differ according to the purpose for which data is collected, with the contrast between service and surveillance purposes most paramount. These two dimensions produce a 2 × 2 framework that hypothesizes which technologies and data-applications in smart cities are likely to raise people's privacy concerns, distinguishing between raising hardly any concern (impersonal data, service purpose), to raising controversy (personal data, surveillance purpose). Specific examples from the city of Rotterdam are used to further explore and illustrate the academic and practical usefulness of the framework. It is argued that the general hypothesis of the framework offers clear directions for further empirical research and theory building about privacy concerns in smart cities, and that it provides a sensitizing instrument for local governments to identify the absence, presence, or emergence of privacy concerns among their citizens.



[VIAL, 2013] — Stéphane VIAL, 2013 — « L'être et l'écran. Comment le numérique change la perception », 1^{re} éd., PRESSES UNIVERSITAIRES DE FRANCE, Paris, 336 pages — ISBN : 978-2-13-062786-9.



Les techniques ne sont pas seulement des outils, ce sont des structures de la perception. Elles conditionnent la manière dont le monde nous apparaît et dont les phénomènes nous sont donnés. Depuis près d'un demi-siècle, les technologies numériques nous apportent des perceptions d'un monde inconnu. Ces êtres qui émergent de nos écrans et de nos interfaces bouleversent l'idée que nous nous faisons de ce qui est réel et nous réapprennent à percevoir. Quel est l'être des êtres numériques ? Que devient notre être-dans-le-monde à l'heure des êtres numériques ? Le temps est venu d'analyser l'« ontophanie numérique » dans toute sa complexité. La prétendue différence entre le réel et le virtuel n'existe pas et n'a jamais existé. Nous vivons dans un environnement

hybride, à la fois numérique et non numérique, en ligne et hors ligne, qu'il appartient aux designers de rendre habitable

[WEISER, 1991] — Mark WEISER, 1991 — “*The Computer for the Twenty-First Century*”, *Scientific American*, vol. 265, n° 3, pp. 94–104 — ISSN: 0036-8733.

☞ Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence.

[WELLNER et al., 1993] — Pierre WELLNER, Wendy MACKAY, and Rich GOLD, 1993 — “*Back to the Real World*”, *Communications of the ACM*, vol. 36, n° 7, pp. 24–26. Special issue on Computer-Augmented Environments — ISSN: 0001-0782.

[ZIADÉ, 2009] — Tarek ZIADÉ, 2009 — « *Programmation Python. Conception et optimisation* », 2^e éd., Collection Blanche. EYROLLES, Paris, 586 pages — ISBN : 978-2-212-12483-5.

☞ Choisi par Google comme l'un de ses langages piliers et utilisé dans des projets d'envergure tels que YouTube, Python est omniprésent dans les applications web modernes. Open Source et portable, sa modularité et son orientation objet permettent de créer des applications de toutes tailles, génériques et maintenables.

Python : de la syntaxe à l'optimisation — Python est tout indiqué pour le développement d'applications web : serveurs de contenu, moteurs de recherche, agents intelligents, objets distribués... Il est également performant pour réaliser des scripts d'administration système ou d'analyse de fichiers textuels, pour gérer l'accès à des bases de données, pour servir de langage glue entre plusieurs applications, réaliser des applications graphiques classiques, etc.

Pour autant, le développeur n'exploitera vraiment sa puissance qu'en ayant acquis une certaine culture. C'est ce que ce livre permet d'acquérir par la description de techniques éprouvées dans tous les grands projets de développement en Python. Au-delà de la prise en main (installation des environnements d'exécution et de développement, rappels de syntaxe avec les primitives et la bibliothèque standard), cet ouvrage aborde les bonnes pratiques de développement Python, depuis les conventions de nommage et les design patterns objet les plus courants jusqu'à la programmation dirigée par les tests et l'optimisation de code.

Enrichie en nouveaux cas pratiques et exercices, cette édition mise à jour pour Python 2.6 détaille également le script de migration 2to3 vers Python 3 et présente la bibliothèque ctypes qui permet de manipuler les structures de données en C/C++.

Médiagraphie

[CASSAGNE, 2019] — David CASSAGNE, 2019 — « *Cours Python 3 pour la programmation scientifique* ». Licence Creative Commons BY-NC-SA 4.0.

[INTERSTICES COLL., 2020] — INTERSTICES COLL., 2020 — « *Interstices.info* ». Licence Creative Commons BY-NC-ND 4.0. Voir Mentions légales.

[PYTHON SOFTWARE FOUNDATION, 2020] — PYTHON SOFTWARE FOUNDATION, 2020 — “*Python*”.

[ROUGIER, 2017] — Nicolas P. ROUGIER, 2017 — “*From Python to numpy*”. Licence Creative Commons BY-NC-SA 4.0.

[ROUGIER, 2020a] — Nicolas P. ROUGIER, 2020a — « *Matplotlib tutorial* ».

[WIKIPEDIA COLL., 2020] — WIKIPEDIA COLL., 2020 — « *Wikipedia.org* ». Licence Creative Commons BY-SA 3.0 et GNU Free Documentation License GFDL. Voir Conditions d'utilisation.

Bibliographie PYTHON

[LANGTANGEN, 2016] — Hans Petter LANGTANGEN, 2016 — “*A Primer on Scientific Programming with Python*”, 5th ed., Texts in Computational Science and Engineering. SPRINGER, Berlin, 922 pages — ISBN: 978-3-662-49886-6.

☞ The book serves as a first introduction to computer programming of scientific applications, using the high-level Python language. The exposition is example and problem-oriented, where the applications are taken from mathematics, numerical calculus, statistics, physics, biology and finance. The book teaches “Matlab-style” and procedural programming as well as object-oriented programming. High school mathematics is a required background and it is advantageous to study classical and numerical one-variable calculus in parallel with reading this book. Besides learning how to program computers, the reader will also learn how to solve mathematical problems, arising in various branches of science and engineering, with the aid of numerical methods and

programming. By blending programming, mathematics and scientific applications, the book lays a solid foundation for practicing computational science.

[LE GOFF, 2019] — Vincent LE GOFF, 2019 — « Apprenez à programmer en Python », 3^e éd., Collection OpenClassrooms. EYROLLES, Paris, 504 pages — ISBN : 978-2-212-67871-0.

[SWINNEN, 2012] — Gérard SWINNEN, 2012 — « Apprendre à programmer avec Python 3 », 3^e éd., Collection Noire. La version numérique de ce texte peut être téléchargée librement à partir du site : <http://inforef.be/swi/python.htm>, EYROLLES, Paris, 436 pages — ISBN : 978-2-2121-3434-6.

Reconnu et utilisé par les enseignants de nombreuses écoles et IUT, complété d'exercices accompagnés de leurs corrigés, cet ouvrage original et érudit est une référence sur tous les fondamentaux de la programmation : choix d'une structure de données, paramétrage, modularité, orientation objet en héritage, conception d'interface, multithreading et gestion d'événements, protocoles de communication et gestion réseau, bases de données... jusqu'à la désormais indispensable norme Unicode (le format UTF-8). On verra notamment la réalisation avec Python 3 d'une application Web interactive et autonome, intégrant une base de données SQLite. Cette nouvelle édition traite de la possibilité de produire des documents imprimables (PDF) de grande qualité en exploitant les ressources combinées de Python 2 et Python 3.

[ZIADÉ, 2009] — Tarek ZIADÉ, 2009 — « Programmation Python. Conception et optimisation », 2^e éd., Collection Blanche. EYROLLES, Paris, 586 pages — ISBN : 978-2-212-12483-5.

Choisi par Google comme l'un de ses langages piliers et utilisé dans des projets d'envergure tels que YouTube, Python est omniprésent dans les applications web modernes. Open Source et portable, sa modularité et son orientation objet permettent de créer des applications de toutes tailles, génériques et maintenables.

Python : de la syntaxe à l'optimisation — Python est tout indiqué pour le développement d'applications web : serveurs de contenu, moteurs de recherche, agents intelligents, objets distribués... Il est également performant pour réaliser des scripts d'administration système ou d'analyse de fichiers textuels, pour gérer l'accès à des bases de données, pour servir de langage glue entre plusieurs applications, réaliser des applications graphiques classiques, etc.

Pour autant, le développeur n'exploitera vraiment sa puissance qu'en ayant acquis une certaine culture. C'est ce que ce livre permet d'acquérir par la description de techniques éprouvées dans tous les grands projets de développement en Python. Au-delà de la prise en main (installation des environnements d'exécution et de développement, rappels de syntaxe avec les primitives et la bibliothèque standard), cet ouvrage aborde les bonnes pratiques de développement Python, depuis les conventions de nommage et les design patterns objet les plus courants jusqu'à la programmation dirigée par les tests et l'optimisation de code.

Enrichie en nouveaux cas pratiques et exercices, cette édition mise à jour pour Python 2.6 détaille également le script de migration 2to3 vers Python 3 et présente la bibliothèque ctypes qui permet de manipuler les structures de données en C/C++.

Bibliographie RASPBERRY PI

[MOCQ, 2015] — François MOCQ, 2015 — « Raspberry Pi 2. Exploitez tout le potentiel de votre nano-ordinateur », Ressources informatiques. ENI ÉDITIONS, Saint-Herblain, 662 pages — ISBN : 978-2-7460-9503-8.

L'objectif de ce livre est de fournir au lecteur des bases solides pour explorer les ressources offertes par le Raspberry Pi tant du point de vue du système d'exploitation que du développement et de l'interfaçage physique. Aucun prérequis en Linux, en programmation ou en électronique n'est nécessaire. Le livre a été écrit sur les modèles Pi 2, B+ et A+, toutes les informations concernant le matériel sont 100% compatibles avec le modèle Pi 3.

Après une présentation physique du Raspberry Pi, vous aurez un aperçu des systèmes d'exploitation compatibles avec cet ordinateur. Vous serez guidé pour installer rapidement le système d'exploitation de votre choix sur une carte SD et rendre votre Raspberry Pi opérationnel (l'écriture a été réalisée sur le système Wheezy). L'utilisation de NOOBS, outil d'installation d'un système, de récupération de la carte SD et de gestion du multiboot est expliquée en détail. Une première étape de découverte du système Linux à travers la ligne de commande précède la mise en œuvre du Raspberry Pi en mode graphique. Vous verrez comment utiliser des mémoires de masse externes (clé USB, disque dur USB) et faire démarrer le Raspberry Pi sur un de ces supports de stockage externes. Vous apprendrez à utiliser les environnements de développement disponibles pour le Raspberry Pi : en Scratch et en Python. La description du GPIO suivie d'exemples d'utilisation des ports d'entrée-sortie du Raspberry Pi et de mise en œuvre de cartes d'interface ouvrent la voie à des applications dans lesquelles le Raspberry Pi interagit avec le monde physique. Vous apprendrez

comment transformer votre Raspberry Pi en poste bureautique avec la suite LibreOffice (édition et impression), en média-center avec XBMC, en serveur web avec lighttpd et WordPress ou en caméra de vidéosurveillance capable de détecter un mouvement et de vous en avertir par email. Enfin, dans le chapitre sur le dépannage, vous découvrirez comment utiliser les voyants du Raspberry Pi pour établir un premier diagnostic. Les principaux dysfonctionnements constatés sur le Raspberry Pi sont également expliqués avec des solutions à mettre en œuvre pour les corriger.

Des éléments complémentaires sont en téléchargement sur le site www.editions-eni.fr.

[UPTON et HALFACREE, 2013] — Eben UPTON et Gareth HALFACREE, 2013 — « *Raspberry Pi. Le guide de l'utilisateur : premiers projets* », trad. par Olivier ENGLER. 1^{re} éd., PEARSON FRANCE, Montreuil, 224 pages — ISBN : 978-2-7440-2579-2.

☞ L'ouvrage couvre toutes les bases dont les utilisateurs peuvent avoir besoin pour tirer pleinement profit de leur Raspberry Pi, aussi bien sur les questions matérielles et logicielles (périphériques, utiliser l'environnement Linux) et sur les premiers pas en programmation. Ce livre permet de comprendre le matériel et ses principes de fonctionnement : installation, configuration, premiers projets (mise en place d'un home cinéma, création d'un serveur web, utilisation du Raspberry Pi comme d'un outil de travail), et premiers programmes avec Scratch et Python. Il permet de mettre en place des premiers projets simples, de créer des programmes soi-même, et d'apprendre les bases de deux langages de programmation simples : Scratch et Python.

Bibliographie (L^A)T_EX

[BITOUZÉ et CHARPENTIER, 2010] — Denis BITOUZÉ et Jean-Côme CHARPENTIER, 2010 — « *L^AT_EX, l'essentiel. Pour une prise en main rapide et efficace* », PEARSON EDUCATION FRANCE, Paris, 372 pages — ISBN : 978-2-7440-7451-6.

☞ L^AT_EX est un système qui permet de rédiger et de mettre en forme des articles de recherche, des polycopiés de cours, des mémoires, des thèses ou des rapports de stage, notamment s'ils comprennent des éléments mathématiques. La prise en main de ce puissant outil n'est pas immédiate. Elle nécessite que les débutants et « ceux qui l'ont été » soient accompagnés dans leur (re)découverte : c'est à eux que s'adresse plus particulièrement cet ouvrage. Ce livre présente l'essentiel de ce qu'il faut savoir pour composer des documents avec L^AT_EX. Le lecteur est guidé pas à pas dans son apprentissage grâce à un cours progressif, illustré d'exemples pratiques. Les auteurs exposent notamment :

- ▶ l'installation et la prise en main du système;
- ▶ la réalisation de formules et l'insertion de schémas;
- ▶ certaines compositions plus spécifiques (PSTricks, classe Beamer, etc.);
- ▶ la gestion des index avec xindy et des bibliographies avec biblatex.

Chaque chapitre se clôt par une série d'exercices, choisis avec soin, dont les corrigés détaillés sont disponibles sur Internet. L'ensemble permet d'assimiler, d'illustrer plus concrètement et de mettre en pratique les différentes fonctions présentées.

[KNUTH, 1984] — Donald E. KNUTH, 1984 — “*TeXbook. Computers & Typesetting*”, 3rd ed., Incorporates the final corrections made in 1996, and a few dozen more, ADDISON-WESLEY PROFESSIONAL, Reading, Massachusetts, 496 pages — ISBN: 0-201-13448-9.

☞ Here is the definitive guide to the use of TeX, written by the system's creator, Donald E. Knuth. TeX represents the state of the art in computer typesetting. It is particularly valuable where the document, article, or book to be produced contains a lot of mathematics, and where the user is concerned about typographic quality. TeX software offers both writers and publishers the opportunity to produce technical text of all kinds, in an attractive form, with the speed and efficiency of a computer system.

Novice and expert users alike will gain from The TeXbook the level of information they seek. Knuth warns newcomers away from the more difficult areas, while he entices experienced users with new challenges. The novice need not learn much about TeX to prepare a simple manuscript with it. But for the preparation of more complex documents, The TeXbook contains all the detail required. Knuth's familiar wit, and illustrations specially drawn by Duane Bibby, add a light touch to an unusually readable software manual.

The TeXbook is the first in a five-volume series on Computers and Typesetting, all authored by Knuth.

[KNUTH, 2003] — Donald E. KNUTH, 2003 — « *Le TeXbook. Composition informatique* », trad. par Jean-Côme CHARPENTIER. 1^{re} éd., VUIBERT, Paris, 555 pages — ISBN : 2-7117-4819-7.



T_EX est un logiciel de composition qui permet l'écriture et la publication de documents, articles ou livres, des plus simples aux plus complexes. Particulièrement adapté aux écritures mathématiques et à la rédaction de documents scientifiques et techniques (physique, chimie, mais aussi langues étrangères et linguistique), il est devenu un outil incontournable pour bon nombre d'étudiants et de chercheurs de toutes nationalités et de toutes disciplines.

Son succès est dû à plusieurs facteurs : qualité typographique des documents qu'il génère, modularité, efficacité, gratuité et disponibilité sur quasiment tous les systèmes informatiques.

Cet ouvrage s'adresse aux utilisateurs novices comme aux experts. Guide de prise en main mais aussi manuel de référence, il contient tous les détails requis en vue d'un usage avancé. Les chapitres suivent les étapes de la mise en œuvre du logiciel et d'une rédaction type. Les difficultés sont signalées de façon à permettre plusieurs niveaux de lecture. Enfin, il contient de nombreux exercices d'application, qui en font un excellent outil d'auto-apprentissage

[LACHAND-ROBERT, 1995] — Thomas LACHAND-ROBERT, 1995 — « *La maîtrise de T_EX et L_AT_EX* », 1^{re} éd., MASSON, Paris, 644 pages — ISBN : 2-225-84832-7.



Doit-on encore présenter T_EX ? Curieusement, ce logiciel très répandu dans le monde universitaire ne semble pas pénétrer facilement le monde de l'entreprise ou des particuliers. Pourtant, ce fantastique outil associe les capacités de logiciel de traitement de texte, notamment pour le formalisme mathématique et de PAO. Il devrait donc séduire plus d'un utilisateur. On le dit vieillot : T_EX a été conçu de manière à rester indémodable, donc d'utilisation constante et permettant d'intégrer les fichiers faits sous des versions antérieures. On le dit compliqué : T_EX est un logiciel compilable, c'est-à-dire que les modifications apportées au texte ne sont visibles qu'après traitement du fichier source. Cette singularité est une conséquence de la puissance de T_EX : l'utilisateur peut y définir de A à Z sa mise en page, mais aussi utiliser des modèles préexistants s'il le souhaite. On le croit restreint à un petit nombre d'utilisateurs : T_EX est probablement le premier traitement de texte performant utilisé aux « quatre coins » du globe. En effet, transmissibles par courrier électronique, les fichiers T_EX permettent une circulation rapide de l'information et de sa mise en forme, ainsi qu'une réponse, éventuellement modifiée. Parmi les nombreuses qualités de T_EX, notons son intangibilité, quels que soient le matériel et la configuration utilisés. En ce sens, T_EX préfigure la norme unique qui verra nécessairement le jour quand seront mises en œuvre les véritables autoroutes de l'information. Enfin, et ce n'est pas un point mineur, T_EX est un logiciel gratuit, contrairement à tous les autres logiciels équivalents.

Que lui manque-t-il pour devenir le traitement de texte ? Probablement un discours clair et précis sur son utilisation et ses possibilités. Ce livre, destiné à tout utilisateur intéressé par T_EX, tente de répondre à ces exigences tout en restant accessible au non-informaticien.

[LAMPORT, 1994] — Leslie LAMPORT, 1994 — “*L_AT_EX: A Document Preparation System. User's Guide and Reference Manual*”, 2nd ed., ADDISON-WESLEY, Indianapolis, 272 pages — ISBN: 0-201-52983-1.



L_AT_EX is a software system for typesetting documents. Because it is especially good for technical documents and is available for almost any computer system, L_AT_EX has become a lingua franca of the scientific world. Researchers, educators, and students in universities, as well as scientists in industry, use L_AT_EX to produce professionally formated papers, proposals, and books. They also use L_AT_EX input to communicate information electronically to their colleagues around the world.

With the release of L_AT_EX₂_E, the new standard version, L_AT_EX has become even more powerful. Among its new features are an improved method for handling different style of type, and commands for including graphics and producing colors. L_AT_EX₂_E makes available to all L_AT_EX users valuable enhancements to the software that have been developed over the years by users in many different places to satisfy a variety of needs.

This book, written by the original architect and implementer of L_AT_EX, is both the user's guide and the reference manual for the software. It has been updated to reflect the changes in the new release. This book begins with instructions for formatting simpler text, and progressively describes commands and techniques for handling larger and more complicated documents. A separate chapter explains how to deal with errors. An added appendix describes what is new and different in L_AT_EX₂_E. Other additions to the second edition include:

- ▶ Descriptions of new commands for inserting pictures prepared with other programs and for producing colored output;
- ▶ New sections on how to make books and slides;
- ▶ Instructions for making and index with the MakeIndex program, and an updated guide to preparing a bibliography with the BibT_EX program;
- ▶ A section on how to send your L_AT_EX documents electronically.

Users new to L_AT_EX will find here a book that has earned worldwide praise as a model for clear, concise, and practical documentation. Experienced users will want to update their L_AT_EX library. Although most standard L_AT_EX input files will work with L_AT_EX₂_E, to take advantage of the new features, a few L_AT_EX₂_E must first be learned. For users who want an advanced guide to L_AT_EX₂_E and to more than 150 packages that can now be used at any site to provide additional features, a useful companion

to this book is The L^AT_EX Companion, by GOOSSENS, MITTELBACH and SAMARIN (also published by Addison-Wesley).

[MITTELBACH et GOOSSENS, 2005] — Franck MITTELBACH et Michel GOOSSENS, 2005 — « *L^AT_EX Companion* », trad. par Jacques ANDRÉ, Benoît BELET, Jean-Côme CHARPENTIER, Jean-Michel HUFFLEN et Yves SOULET. 2^e éd., PEARSON EDUCATION FRANCE, Paris, 1008 pages — ISBN : 2-7440-7133-1.

☞ *Langage de programmation dédié à l'écriture et à la composition de documents structurés, L^AT_EX permet de rédiger des notes, des articles, des mémoires de thèse et des livres dans tous les domaines scientifiques (physique, mathématiques, chimie, statistique mais aussi économie ou philologie) et de les mettre en pages avec une qualité typographique remarquable.*

Cette nouvelle édition du L^AT_EX Companion, entièrement revue, présente 200 des plus récentes extensions écrites pour L^AT_EX. Rédigée par quelques uns des principaux développeurs du langage, elle offre au lecteur des conseils sur :

- ▶ *la mise en forme des documents;*
- ▶ *la création de graphiques;*
- ▶ *la création de tableaux;*
- ▶ *l'écriture de textes multilingues;*
- ▶ *la composition des mathématiques;*
- ▶ *la gestion des bibliographies;*
- ▶ *les polices de caractère;*
- ▶ *la génération de listes et d'index...*

Les principales difficultés techniques et typographiques et les moyens de les résoudre, sont présentées et illustrées par plus de 900 exemples, disponibles sur le site qui accompagne l'ouvrage. On y trouvera aussi une bibliographie commentée et de nombreuses annexes.

L^AT_EX Companion 2^e édition est destiné aux utilisateurs intermédiaires et confirmés voulant tirer profit de toutes les fonctionnalités de L^AT_EX.

[ROLLAND, 2005] — Christian ROLLAND, 2005 — « *L^AT_EX par la pratique* », 1^{re} éd., O'REILLY FRANCE, Paris, 560 pages — ISBN : 2-84177-073-7.

☞ *L^AT_EX est un outil de formatage de texte libre et gratuit, qui permet d'obtenir aisément une qualité typographique sans égale sur tous les systèmes d'exploitation majeurs. Après s'être imposé en leader du traitement de texte scientifique, il est promis à un avenir brillant, notamment en tant que plate-forme d'impression de documents SGML ou XML.*

Ce livre est le seul ouvrage en français spécifiquement consacré à L^AT_EX. Son approche est résolument pratique et s'adresse aussi bien aux débutants souhaitant se familiariser avec la syntaxe de L^AT_EX, réputée austère, qu'aux pratiquants réguliers qui veulent trouver rapidement une solution aux problèmes les plus complexes. En 18 chapitres et 4 annexes, vous serez à même de trouver rapidement l'information qui vous manque, quels que soient vos besoins et vos connaissances. Vous trouverez dans L^AT_EX par la pratique :

- ▶ *des chapitres introductifs, qui montrent comment créer rapidement des documents simples, contenant listes, tableaux ou formules mathématiques,*
- ▶ *tout sur l'inclusion de dessins et illustrations. Un chapitre est consacré aux techniques de base, un autre au graphisme avancé;*
- ▶ *des chapitres avancés, où l'on explore l'univers des mathématiques complexes ou des tableaux imbriqués;*
- ▶ *tous les renseignements nécessaires, exemples à l'appui, pour gérer de longs documents à l'aide de tables des matières, compteurs de figures, index, bibliographies, etc.;*
- ▶ *des explications détaillées concernant la création de classes et autres extensions personnelles;*
- ▶ *quelques digressions autour de L^AT_EX, y compris une présentation des principaux outils de conversion LaTeX vers HTML.*

[VOSS, 2010] — Herbert Voss, 2010 — “*Typesetting Mathematics with L^AT_EX*”, 1st ed., Updated for L^AT_EX2_E. 13th Printing, 2001., UIT CAMBRIDGE, Cambridge, UK, 304 pages — ISBN: 978-1-906-86017-2.

Médiagraphie (L^A)T_EX

[TEX SX, 2020] — TEX SX, 2020 — “*TeX StackExchange*”.

[TEX USERS GROUP, 2020] — TEX USERS GROUP, 2020 — “*TeX Live 2020*”.

GLOSSAIRE

A : B : C : D : E : F : G : I : L : M : N : O : P : R : S : T : U : V : W : X

A **ACCESSIBILITÉ** — Concept initialement issu du monde du handicap et désormais étendu à l'ensemble des citoyens pour signifier la facilitation de l'accès dans différents domaines. En informatique, on parle d'accessibilité pour désigner l'adaptation des systèmes numériques et le développement d'outils spécifiques dans le cas de handicaps, comme par exemple la malvoyance.

ALAN TURING — Alan TURING est un mathématicien et cryptologue britannique (1912-1954), auteur de travaux qui fondent scientifiquement l'informatique. Pour résoudre le problème fondamental de la décidabilité en arithmétique, il présente en 1936 une expérience de pensée que l'on nommera ensuite machine de Turing et des concepts de programmation et de programme, qui prendront tout leur sens avec la diffusion des ordinateurs dans la seconde moitié du XX^e siècle.

ALGORITHME D'AIDE À LA PRISE DE DÉCISION — Algorithme dont l'objet est d'apporter des informations utiles pour la prise de décision (par exemple en fournissant un classement pertinent, comme les algorithmes de recommandation et les moteurs de recherche ou encore, en répertoriant des éléments dans des catégories). Dans certains cas, ces algorithmes peuvent même être utilisés dans des processus entièrement automatisés.

ALGORITHME D'APPRENTISSAGE — Cela correspond à des algorithmes qui ajustent les paramètres de leurs calculs en fonction des exemples qui leur sont donnés ou des retours (positifs ou négatifs, comme des récompenses ou punitions) issus de calculs précédents. Cela permet d'adapter leur fonctionnement aux données fournies.

ALGORITHME DE PRÉDICTION — Algorithme qui permet de déterminer — pour de nouvelles données —, des caractéristiques à partir de la connaissance de données d'apprentissage préalablement agréées. En quelque sorte, l'algorithme apprend des règles en se fondant sur les données d'apprentissage et les applique à de nouvelles données. Cela se réalise, entre autres, à l'aide de probabilités, de statistiques et de régressions.

ALGORITHME D'OPTIMISATION — Famille d'algorithmes qui résolvent un problème par améliorations successives : on part d'une solution

initiale par défaut, on la modifie un peu dans un sens ou dans un autre et si une de ses modifications améliore la solution, on réitère le procédé. Un critère de gain à maximiser ou de coût à minimiser est donc à la base de ces méthodes.

ANONYMISER — Garantir qu'un jeu de données contenant des informations personnelles ne permet pas d'identifier des individus, donc évite d'accéder à des informations privées et confidentielles les concernants.

APACHE — APACHE est un serveur Web sous licence libre parmi les plus répandus sur Internet. Il s'agit d'une application fonctionnant initialement sur les systèmes d'exploitation de type UNIX, mais désormais porté sur tous les OS majeurs du marché.

APPLICATION — Une *application* ou un *applicatif* est, dans le domaine informatique, un programme — ou un ensemble logiciel — directement manipulé par l'utilisateur pour réaliser une tâche ou un ensemble de tâches élémentaires d'un même domaine formant un tout. Typiquement, un éditeur de texte, un navigateur Web, un client de messagerie, une visionneuse d'images, un lecteur multimédia, un jeu vidéo, sont des applications.

B **BIG DATA** — Parfois qualifié comme *données massives*, le terme *big data* désigne des ensembles de données qui s'avèrent tellement volumineux qu'ils en deviennent difficiles à traiter avec les outils traditionnels de gestion de base de données ou de l'information. Le *big data* offre de nouvelles perspectives en matière d'analyse de données, notamment à des fins prédictives, avec des retombées espérées dans de nombreux domaines : santé, environnement, gestion des risques, etc.

BITCOIN — Le *bitcoin* est une monnaie planétaire, cryptographique, fondée sur un système de transaction et contrôle, la *blockchain*.

BLOCKCHAIN — Technologie de stockage et partage d'information avec un protocole de gestion de données numériques, qui est :

- ▶ transparente, car chacun peut consulter l'ensemble des échanges, présents et passés;
- ▶ sans organe de contrôle, car établie sur des échanges de pair-à-pair — *peer-to-peer* en anglais;
- ▶ infalsifiable et sécurisée, car différents exemplaires existent simultanément à de nombreux endroits ce qui empêche d'en modifier un ou quelques-uns.

BSD — BSD est l'acronyme de *Berkeley Software Distribution*, suite logicielle provenant de l'Université de Californie située à Berkeley. À l'origine, il s'agit d'extensions apportées au système d'exploitation UNIX AT&T. Plusieurs systèmes *Open Source* sont fondés sur cette base, en incluant des éléments issus d'autres projets, y compris du projet GNU : un noyau BSD, la bibliothèque C BSD — différente de celle du projet GNU —, des utilitaires — interpréteur de commandes, gestionnaire de fichiers, compilateurs et éditeurs de liens —, le système X Window pour l'affichage graphique — BSD ne définit pas de « bureau graphique » tels que *GNOME* ou *KDE* —, etc. D'abord principalement dévolus aux serveurs — les couches réseaux des BSD ont été portées dans bien d'autres systèmes —, les BSD s'utilisent de nos jours aussi bien comme station de travail. On distingue essentiellement les distributions *FREEBSD*, *NETBSD* et *OPENBSD*.

C **CARTE MÈRE** — Circuit imprimé qui rassemble un ensemble diversifié de connecteurs et de ports (connectique dédiée de certaines fonctionnalités) permettant d'y agréger les différents composants de

l'ordinateur (processeur, cartes mémoire, carte graphique, disques durs et tout autre périphérique) et d'assurer leur liaison à travers des circuits imprimés dont le comportement est piloté par le **BIOS**, programme contenu dans la mémoire « morte » (ROM — *Read-Only Memory*) et lancé au démarrage pour notamment détecter et configurer tous les éléments connectés.

CENTRALITÉ D'INTERMÉDIARITÉ — La centralité d'intermédiairité est, parmi d'autres, un moyen de mesure de la centralité du sommet d'un graphe. Elle capture le nombre de fois où un noeud agit, dans un graphe, comme un point de passage le long du plus court chemin entre deux autres nœuds.

CERVEAU — C'est l'organe principal du système nerveux des animaux, il régule les autres systèmes d'organes du corps, en agissant sur les muscles ou les glandes, et constitue le siège des fonctions cognitives (mémoire, apprentissage, planification, etc.); cette cognition n'existe que parce le cerveau est incarné dans un corps en interaction avec un environnement.

CLOUD — Le *cloud* signifie, qu'au lieu de stocker ses données et d'effectuer ses calculs sur son propre ordinateur, les deux sont confiés à un centre de données se trouvant sur Internet.

COBOT — S'emploie comme contraction de « robot collaboratif » et désigne un robot en interaction réelle, directe ou télé-opérée avec un opérateur ou un utilisateur humain.

Co-conception — Approche également appelée *co-design*. Il s'agit de développer un système en collaboration avec l'utilisateur final, au-delà d'une simple validation d'étape de sa part.

CODE SOURCE — Le code source d'un programme est une suite d'instructions logiques écrite par un informaticien — aussi nommé programmeur ou développeur — dans un langage de programmation. Autrement dit, c'est un fichier éditable, aux instructions lisibles et compréhensibles par un être humain à condition qu'il en maîtrise la signification.

CONCEPTION CENTRÉE UTILISATEUR — Démarche de conception où les besoins des utilisateurs sont pris en compte à chaque étape.

CONCEPTION ITÉRATIVE — Approche de conception qui se réalise par bouclage de cycles répétés. Chaque cycle permet de tester, d'adapter et d'améliorer le produit.

CONCEPTION PARTICIPATIVE — Démarche de conception qui fait participer les utilisateurs. Il s'agit d'une démarche centrée utilisateur où est mis en avant le rôle actif de l'utilisateur.

CONNECTEUR — Élément qui assure une connexion physique pour établir une liaison électrique ou une transmission de données entre deux entités matérielles distinctes. En informatique, ce terme est également employé pour désigner un port — même si le connecteur caractérise aussi la fiche mâle de la connectique qui lui est associée.

COPYLEFT — Le concept de *Copyleft* est une méthode générale pour rendre libre un programme — ou toute œuvre relevant du droit d'auteur —, imposant à toutes les versions modifiées ou étendues de l'œuvre d'être libres également : en quelque sorte c'est une, voire la première, « contagion virale positive », au sens du partage de connaissances. Son invention est celle de **Don HOPKINS**, largement popularisée par **Richard STALMAN** dans le cadre du projet **GNU**.

CRYPTER — Anglicisme employé pour le mot *chiffrer*, c'est-à-dire opération qui vise à rendre secret un document par *chiffrement*, pour que sa compréhension soit impossible à toute personne qui n'a pas la clé de (dé)chiffrement.

- D** **DATA CENTER** — Un *Data center* ou centre de données en français est un site physique sur lequel se trouvent regroupés des équipements constituants des systèmes d'information : ordinateurs, baies de stockage, équipements réseaux et de télécommunications, etc.
- DEEP LEARNING** — L'apprentissage profond ou *deep learning* est une architecture qui associe en cascade plusieurs couches d'algorithmes de ce type pour hiérarchiser le problème et obtenir des performances bien plus importantes.
- DEBIAN** — **DEBIAN** est le nom d'une distribution non commerciale qui repose *exclusivement* sur des *logiciels libres*. Elle est développée et maintenue par l'association **DEBIAN**, dont une particularité est qu'elle suit un mode de gouvernance communautaire. La version **DEBIAN** pour RASPBERRY PI peut se télécharger sur le site de la fondation Raspberry, soit *via* un *installateur (NOOBS)*, soit *via* une *image de la distribution (RASPBIAN)*. Plus largement, la qualité et la stabilité reconnues de **DEBIAN** ont engendré de nombreuses distributions populaires la prenant comme fondement, par exemple **UBUNTU** ou **LINUX MINT**.
- DEEP BLUE** — *Deep Blue* est un superordinateur spécialisé dans le jeu d'échecs par adjonction de circuits spécifiques, développé par IBM au début des années 1990.
- DESIGN** — Le *design* recouvre les notions de création et de conception d'un produit ou d'un système.
- DIGITAL** — Anglicisme souvent employé comme synonyme de numérique. Pour en savoir plus sur le débat digital vs. numérique : <http://www.blogdumoderateur.com/numerique-ou-digital>.
- DISCRET** — Un objet est dit *discret* lorsque ses points sont écartés les uns des autres, le contraire étant dénommé un objet continu. L'opposition du continu et du discret est un thème philosophique important déjà interrogé par la philosophie grecque.
- DROIT PRÉEMPTIF** — Sur ce sujet voir la vidéo de Daniel LE METAYER sur le numérique, le droit et la vie privée.
- E** **EFFICACITÉ** — Concept lié à l'atteinte du résultat prévu.
- EFFICIENCE** — Terme signifiant l'atteinte d'un résultat avec le moindre effort ou le temps minimal.
- ERGONOMIE** — Discipline qui étudie les interactions existantes entre les êtres humains et les composantes d'un système. On y prend en compte les facteurs humains pour concevoir et évaluer des tâches afin de les rendre compatibles avec les besoins et les capacités des utilisateurs.
- ÉTHIQUE** — L'éthique est une réflexion de fond sur un sujet, qui permet ensuite de proposer des normes, limites et devoirs, dans le respect de certaines valeurs. La morale désigne l'ensemble de ces règles établies.
- EXÉCUTABLE** — Se dit de l'état dans lequel se trouve une tâche après son activation et avant qu'elle ne soit finie. Par extension, cela désigne également le nom du fichier d'un programme — dits de *script* pour les langages interprétés (BASH, PERL, PYTHON, etc. : traduction « à la volée » en langage machine) ou comme résultat d'une *phase de compilation* (traduction du langage de programmation en langage machine) pour les langages dits *compilés* (FORTRAN, PASCAL, C/C++, etc.) — prêt à être lancé pour réaliser les tâches qui lui incombent.
- EXTERNALISER** — Désigne le fait qu'une entreprise fasse appel à un prestataire extérieur pour certains pans de son fonctionnement plutôt

que d'embaucher en interne un employé ayant la compétence requise (par exemple pour le ménage, la comptabilité ou l'accueil téléphonique). Avec le numérique, il devient plus facile et plus rentable d'externaliser beaucoup de services.

F FIREFOX — Firefox, développé par la *Mozilla Foundation*, est un des logiciels libres multiplateformes les plus répandus et des plus performants pour la navigation sur le Web.

G GNU — GNU est l'acronyme récursif de *GNU's Not UNIX*. Cela qualifie un ensemble complémentaire de programmes utilitaires libres rattachés aux noyaux des systèmes d'exploitation de type UNIX. L'ambition initiale du projet était de développer un système d'exploitation complet en y adjoignant son propre noyau connu sous le nom de HURD, double acronyme récursif de *Hird of Unix-Replacing Daemons* (littéralement « *Multitude de démons*^a pour remplacer Unix ») puis HIRD, autrement dit, « *Hurd of Interfaces Representing Depth* » : jeu de mot avec « *herd* », qui signifie *troupeau*^b en anglais. Cependant, la montée en charge remarquable de l'opérationnalité de LINUX de 1991 à 1996, associée à sa licence de publication GPL en 1992, l'a imposé de fait comme noyau du projet GNU ; d'où la double affiliation de dénomination ces systèmes en tant que GNU/LINUX.

GNU/LINUX — GNU/LINUX ou bien, par raccourci innapproprié LINUX, est un système d'exploitation entièrement libre, produit d'un grand nombre d'entreprises et de bénévoles qui contribuent à son élaboration. Il est disponible sous la forme d'une myriade de distributions généralistes ou spécialisées (assemblage cohérent de multiples logiciels libres — noyau LINUX et outils GNU — pour fournir un système d'exploitation opérationnel), telles que : DEBIAN, UBUNTU, LINUX MINT, RED HAT, SUSE, RASPIAN, etc.

GPIO — GPIO, acronyme de *General Purpose Input/Output*, est un port d'entrée-sortie qui permet à une carte électronique — carte à micro-contrôleur ou carte-mère — de communiquer avec un périphérique quelconque. Ce type de port est programmable, permettant de s'adapter à une vaste diversité de périphériques, sous condition d'écrire un pilote adéquat.

GPU (GRAPHICAL PROCESSING UNIT) — Un GPU est un processeur supplémentaire dans nos ordinateurs dans lequel tous les calculs liés aux opérations graphiques d'affichage sont pré-cablés pour accélérer le traitement.

GRAPHE — Un *graphe* est un type de structure de données composé de sommets (aussi appelés points, noeuds) reliés par des liens (aussi appelés arêtes ou arcs).

GRAPHE ÉTIQUETÉ — Il s'agit d'un graphe dont les liens sont étiquetés par un chiffre, un symbole, une lettre, etc.

GRAPHE ORIENTÉ — Si les liens d'un graphe sont orientés (l'orientation est matérialisée par des flèches), cela signifie que la relation va dans un seul sens, elle est asymétrique. Le graphe est donc orienté.

^a. Les « *daemons* » sont des programmes qui tournent en tâche de fond et qui apportent des services de gestion spécifique au système d'exploitation.

^b. Bien sûr, sous entendu de *gnou/gnu*.

J INGÉNIER SYSTÈME — Métier de l'informatique qui prend en charge tout ce qui relève de l'exploitation des infrastructures informatiques matérielles et logicielles.

INTELLIGENCE ARTIFICIELLE — L'intelligence artificielle — IA — est, selon l'*Encyclopédie LAROUSSE*, « l'ensemble des théories et des techniques mises en œuvre en perspective de réaliser des machines capables de simuler l'intelligence ». Cela recouvre un ensemble

de concepts et de technologies plus qu'une discipline autonome constituée. Au regard d'une définition peu précise, certains, à l'instar de la [CNIL](#), introduisent ce sujet comme « le grand mythe de notre temps » (source [W](#)).

INTELLIGENCE ARTIFICIELLE LOGIQUE — Se référer aux explications sur ce courant de l'intelligence artificielle dans la vidéo de Nicolas P. ROUGIER : [Enjeux et histoire de l'intelligence artificielle](#).

INTERFACE HOMME-MACHINE — Ensemble de moyens mis en œuvre pour qu'un humain puisse interagir avec une machine. Par extension, l'IHM est aussi le domaine de recherche qui conçoit des systèmes informatiques et étudie les interactions entre êtres humains et machines.

INTERNET — L'[Internet](#) est un ensemble de standards et de technologies qui permettent de relier les ordinateurs et les réseaux entre eux.

INTERNET DES OBJECTS (IOT) — L'[Internet des objets](#) — en anglais *Internet of Things* —, représente l'extension d'Internet à des éléments et à des lieux du monde physique.

L LIBREOFFICE — [LIBREOFFICE](#) est une suite bureautique libre complète, dérivée du projet [OpenOffice.org](#), créée et gérée par [THE DOCUMENT FOUNDATION](#) suite au rachat en 2010 de SUN MICROSYSTEMS — alors détenteur de la marque — par ORACLE.

LIEN — Un lien est un élément qui établit des liaisons. Lorsqu'on parle de graphes, les liens sont ce qui relie les différents sommets ou nœuds.

LINGUISTIQUE — Discipline qui étudie le langage et le fonctionnement des langues par une approche descriptive.

LINUX — [LINUX](#) est le noyau du système d'exploitation [GNU/LINUX](#). Le noyau est le cœur du système, c'est lui qui s'occupe d'ordonner les tâches et de fournir aux logiciels applicatifs une interface de programmation pour utiliser le matériel au moyen de modules d'extensions spécifiques nommés « pilotes » — ou « drivers » en anglais. [LINUX](#), contraction entre *Linus* et *UNIX*, est initié par Linus TORVALDS en août 1991.

LOGICIEL EMBARQUÉ — Un système embarqué — ou logiciel embarqué — est défini comme un système électronique et informatique autonome, fonctionnant souvent en temps réel, spécialisé dans une tâche bien précise et intégré au sein d'une entité ou d'un objet : voiture, train, avion...

M MACHINE LEARNING — L'apprentissage automatique ou *machine learning* correspond à des algorithmes qui ajustent les paramètres de leurs calculs en fonction des exemples qui leur sont donnés. Cela permet d'adapter leur fonctionnement aux données fournies en entrée d'algorithme.

MATRICE D'ADJASCENCE — Une [matrice d'adjacence](#) est un outil mathématique qui permet de modéliser sous forme d'un tableau (ou matrice) les liens d'un graphe et donc, dans le contexte présent, les relations d'un réseau.

MÉMOIRE — Dispositif électronique qui sert à stocker des données (valeurs, instructions de programmes). Dans un ordinateur, il existe trois grands types de mémoire externe au processeur : la [mémoire vive \(RAM\)](#) (carte branchée sur la carte mère), la [mémoire morte \(ROM\)](#) (circuit intégré à la carte mère) et la [mémoire de masse](#) (disques durs, et autres supports de stockage périphérique). Le processeur intègre aussi plusieurs niveaux de mémoire, notamment de la [mémoire cache](#) (placée dans la puce qui contient le processeur) et les registres (internes au processeur).

MÉMOIRE VIRTUELLE — Mécanisme qui permet de simuler la présence d'un type de mémoire en utilisant un autre type (par exemple un disque dur). Il est utilisé par exemple pour simuler la présence de mémoire vive en utilisant de la mémoire de masse. Sous LINUX, ce type de mécanisme est dénommé « *swap* » où à l'installation du système d'exploitation, une partition du disque dur est réservée à cet effet (on conseille souvent une taille double de celle de la mémoire vive de l'ordinateur).

MÉMOIRE VIVE — La RAM, acronyme de *Random Access Memory* en anglais — littéralement « mémoire à accès aléatoire » nommée en français « mémoire vive » —, est un circuit imprimé sur une carte qui permet de stocker les données et les instructions employées dans un programme informatique. Contrairement à la mémoire dite « morte » en français — ROM, *Read-Only Memory*, littéralement « mémoire uniquement en lecture » —, le contenu de la mémoire vive peut être modifié et offre donc un espace temporaire de stockage dynamique pour le processeur.

MODÉLISATION — Représentation d'un objet réel ou abstrait, en éliminant les détails difficiles ou accessoires à reproduire, afin d'obtenir un résultat plus net à interpréter, un modèle est validé par son adéquation à des données (différentes de celles qui ont pu aider à le construire), donc à prédire des faits nouveaux; il peut-être mathématique ou informatique, mais aussi être un objet tangible (une maquette, un animal modèle pour certains fonctionnements biologiques).

MODÉLISATION DES DONNÉES — Représentation abstraite, le modèle de données ne définit pas seulement la structure de données mais aussi ce que les données veulent vraiment signifier (sémantique).

MONNAIE CRYPTOGRAPHIQUE — Monnaie électronique associée aux principes de la cryptographie mathématique pour valider les transactions et émettre la monnaie elle-même.

MySQL — MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde.

 **NEURONE ARTIFICIEL** — Un neurone artificiel est une fonction qui combine des entrées pour calculer si la valeur de la sortie est plutôt élevée ou basse au regard d'un seuil défini, son calcul étant ajusté par des paramètres.

NORME ISO 9241-2010 — Cette norme ISO est directement liée aux IHM. « L'ISO 9241-2010:2010 fournit des exigences et des recommandations relatives aux principes et aux activités de conception centrée sur l'opérateur humain, intervenant tout au long du cycle de vie des systèmes informatiques interactifs. Elle est destinée à être utilisée par les responsables de la gestion des processus de conception, et traite des manières dont les composants matériels et les logiciels des systèmes interactifs permettent d'améliorer l'interaction homme-système ».

 **ONTOLOGIE** — Ensemble structuré de tous les termes et concepts de même que de leurs relations qui entrent dans la description d'un domaine de connaissance. Une ontologie a pour but de permettre l'automatisation de raisonnements à propos des objets du domaine concerné. « L'ontologie est aux données ce que la grammaire est au langage » ([W](#)).

OPTIMISATION PAR ESSAIS/ERREURS — Cela correspond à un type d'algorithme qui ajustent les paramètres par « renforcement », une solution est testée et en fonction du retour positif ou négatif, un ajustement se fait pour aller vers un meilleur comportement. Le fait que le retour se passe après l'action qui a pu en être la cause, parfois bien après, impose d'avoir une représentation interne de ce qui se passe au cours du temps.

ORDONNANCEUR — Composant logiciel du système d'exploitation qui est en charge d'allouer du temps processeur aux processus. Le choix du processus élu pour s'exécuter sur le processeur à un temps donné est fait par une procédure d'ordonnancement et implique de gérer les changements de contexte.

OS/SYSTÈME D'EXPLOITATION — OS, acronyme de *Operating system*, est l'expression anglophone signifiant « système d'exploitation ». Il s'agit d'un ensemble de programmes qui gèrent l'utilisation des ressources matérielles d'un ordinateur : stockage des mémoires à accès rapide et de masse des disques durs, ordonnancement des calculs du processeur, communication avec les périphériques ou administration du réseau. Le système d'exploitation accepte ou refuse ces demandes, puis réserve les ressources en question pour éviter que leur utilisation n'interfère avec d'autres requêtes provenant d'autres logiciels.

P **PERCEPTRON** — Inventé en 1957 par Frank ROSENBLATT au laboratoire d'aéronautique de l'Université Cornell, le perceptron peut être vu comme le type de réseau de neurones le plus simple.

PÉRIPHÉRIQUE — Dispositif électronique connecté à un ordinateur, plus précisément sur un des ports dédiés de la carte mère.

PHP — *PHP* — *Hypertext Preprocessor*, acronyme auto-référentiel — est un langage de script avant tout utilisé pour produire des pages Web dynamiques via serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

PORT — Prise/connecteur permettant de brancher un périphérique à un ordinateur. Les ports sont placés sur la carte mère qui assure la liaison avec les autres périphériques et le processeur.

PRINCPIA MATHEMATICA — Œuvre en trois volumes d'Alfred NORTH WHITEHEAD et Bertrand RUSSELL, publiés en 1910-1913, *Principia mathematica* a pour sujet les fondements des mathématiques et est considérée comme un des livres les plus influents de l'histoire de la logique.

PROCESSEUR — Processeur ou CPU, pour *Central Processing Unit*. Dispositif électronique de calcul chargé d'exécuter les différentes instructions d'un programme. Ce faisant, les données numériques sont traitées, manipulées et échangées avec les autres composants de l'ordinateur — mémoire vive (RAM — *Random Access Memory*), disques durs et autres périphériques — via la carte mère. Quand le processeur est construit sur un unique circuit intégré, on parle de microprocesseur.

PROCESSUS — Forme que prend un programme quand il est exécuté au sein d'un système d'exploitation. Une instance de processus comprend en général : un ensemble d'instructions (souvent copiées dans la RAM depuis le disque dur), une place, appelée espace d'adressage, réservée en mémoire vive pour stocker les données qu'il manipule et les ressources matérielles et logicielles que le programme utilise.

PSYCHOLOGIE COGNITIVE — Branche de la psychologie qui étudie spécifiquement le fonctionnement de l'acquisition des connaissances chez l'être humain : langage, mémoire, perception, concentration, etc. La cognition — du latin *cognito* pour action de connaître —, est ainsi l'ensemble des activités mentales et mécanismes qui se rapportent à la connaissance.

R **RÉALITÉ AUGMENTÉE** — La réalité augmentée se définit comme la superposition de l'observable physique avec différents éléments (sons, images, traitements de l'information, etc.) calculés par un système informatique en temps réel. Cela désigne souvent les méthodes qui incrustent de manière réaliste des objets virtuels dans une séquence d'images. Néanmoins, cela s'applique aussi bien à toutes les perceptions proprioceptives : visuelles, comme tactiles ou auditives — (voir **W**).

RÉALITÉ VIRTUELLE — L'expression « réalité virtuelle » — multimédia immersif ou encore simulation — renvoie à une technologie numérique qui reproduit la présence d'un utilisateur en environnement artificiel généré au moyen de logiciels. Il s'agit donc d'un contexte dans lequel l'usager interagit pour vivre une expérience sensorielle qui peut inclure la vue, le toucher, l'ouïe et l'odorat : visuelle, haptique, sonore ou olfactive (cf. **W**).

RÉSEAU DE NEURONES ARTIFICIELS — Un réseau de neurones fait référence à l'assemblage d'un grand nombre de neurones (sous-entendu artificiels) pour permettre de faire un calcul entrée/sortie très sophistiqué.

RÉSEAU SOCIAL — Un réseau social associé au Web est relatif à une application qui utilise les sciences et technologies de l'information et de la communication pour mettre en relation des personnes. Hors monde numérique, un réseau social est un groupement de personnes qui a un sens.

ROBOTIQUE — La robotique regroupe l'ensemble des techniques permettant la conception et la réalisation de machines automatiques — ou robots — à même de réaliser des tâches précises, quelque soit leur champ d'application : industrie, domotique, transport, médecine, sciences, etc. (source **W**).

S **SATISFACTION** — Évaluation subjective par l'utilisateur de l'interaction avec un système.

SCIENCES COGNITIVES — Sous un vocable commun, les sciences cognitives regroupent l'ensemble des disciplines qui étudient les mécanismes de la connaissance.

STALLMAN RICHARD M. — Richard STALLMAN est à l'origine un brillant programmeur issu du MIT (entre autres, l'éditeur EMACS). Initiateur du mouvement du logiciel libre, il lance en 1983 le projet *GNU, GNU IS NOT UNIX* — acronyme récursif —, en fondant les bases juridiques du projet sur la Licence publique générale (*General public licence* — GPL) initialement prévue pour les codes de logiciels et la Licence publique générale restreinte (*Lesser general public licence* — LGPL) pour les bibliothèques associées. Bien que dévolues au domaine des logiciels, voire des matériels informatiques, ces licences peuvent tout aussi bien s'appliquer à toute œuvre relevant du droit d'auteur, en parallèle ou à l'instar des licences *Creative Commons*.

SÛRETÉ — La sûreté de fonctionnement d'un système informatique est son aptitude à remplir ses fonctions en dépit de pannes matérielles ou logicielles.

SYSTÈME D'EXPLOITATION — Un « système d'exploitation » — *Operating system* en anglais —, est un ensemble de programmes qui gèrent l'utilisation des ressources matérielles d'un ordinateur : stockage des mémoires à accès rapide et de masse des disques durs, ordonnancement des calculs du processeur, communication avec les périphériques ou administration du réseau. Le système accepte ou refuse ces demandes, puis réserve les ressources en question pour éviter que leur utilisation n'interfère avec d'autres requêtes provenant d'autres logiciels.

SYSTÈME EXPERT — De manière générale, c'est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier. Sur ce sujet voir aussi la vidéo de Nicolas P. ROUGIER : *Enjeux et histoire de l'intelligence artificielle*.

SYSTÈME UBIQUITAIRE — L'ubiquité ou l'omniprésence, est la capacité de se trouver en tout lieu ou à plusieurs endroits en même temps. En informatique, cela renvoie aux environnements dans lesquels ordinateurs et réseaux sont intégrés au monde réel et où l'utilisateur est connecté en permanence (cf. **W**).

T **TECHNIQUE D'ANONYMISATION** — Les techniques d'anonymisation se rapportent aux moyens de transformer des jeux de données — effacement de certains attributs, généralisation, bruitage et autres manipulations —, afin de rendre très difficile voire impossible la « ré-identification » ou l'inférence de connaissances sur des personnes (physiques ou morales).

TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION — Issue de la transcription de l'anglais ICT pour *Information and Communication Technologies*, Tic est une expression, essentiellement utilisée dans les mondes universitaires et de l'administration pour désigner le domaine couvert par la télématique — informatique, audiovisuel, multimédias, Internet, télécommunications, etc. — qui permet aux utilisateurs d'accéder, produire, manipuler, communiquer des informations sous toute forme : texte, son, image, vidéo, etc. (source **W**).

TORVALDS LINUS B. — **Linus TORVALDS** est un informaticien d'origine finlandaise reconnu pour avoir créé en 1991 le noyau **LINUX**. Depuis, il continue d'en diriger le développement où, à l'instar de Guido van Rossum pour **PYTHON**, de Larry Wall pour **PERL** et de Mark Shuttleworth pour **UBUNTU**, il en est considéré comme le « dictateur bienveillant ».

U **UNIX** — Système d'exploitation multitâche et multi-utilisateur, **UNIX** est créé en 1969 par Kenneth Thompson. Il repose sur un interpréteur ou superviseur (le shell) et de nombreux petits utilitaires, accomplissant chacun une action spécifique, commutables entre eux (mécanisme de « redirection ») et appelés depuis la ligne de commande. **LINUX** ou **BSD** — « *Berkeley Software Distribution* » — sont des dérivés libres et ouverts de type **UNIX**.

UTILISABILITÉ — Conformément à la norme ISO 9241-11, il s'agit du « degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». L'utilisabilité possède plusieurs critères : l'*efficacité*, l'*efficience* et la *satisfaction*.

V **VERROU SCIENTIFIQUE** — On parle de verrou scientifique pour un résultat pas encore obtenu mais dont dépendent d'autres résultats scientifiques ou techniques. Cela constitue une contrainte à lever pour qu'un domaine de recherche avance.

VISION PAR ORDINATEUR — Un ensemble d’algorithmes qui partent des valeurs des pixels des images pour en extraire des caractéristiques, localiser les objets qui y sont vus et les étiqueter.

VLC — [VLC MEDIA PLAYER](#) est un lecteur multimédia libre. Ce logiciel est multiplateforme et distribué sous licence GNU GPL. Il intègre les codecs nécessaires à la lecture de la plupart des formats audio et vidéo. De plus, le lecteur est capable de lire un grand nombre de flux réseaux.

WEB — Le [World Wide Web](#), parfois aussi appelé la *Toile*, est une des applications d’Internet qui permet de lier et consulter à distance des documents (par ex. un journal), des interfaces d’applications (par ex. un service de réservation), des données (par ex. la météo d’une ville), etc.

WEB SÉMANTIQUE — Extension du Web — à savoir syntaxique — dans laquelle l’information se voit associée à un sens bien défini, améliorant la capacité des logiciels à traiter l’information disponible sur le Web. Le Web sémantique repose sur des standards du [W3C](#) (organisme international de normalisation du Web) comme RDF (Resource Description Framework). L’expression a été inventée par Tim Berners-Lee, inventeur du Web et directeur du W3C.

WEB SOCIAL — Le Web social est une évolution particulière du Web caractérisée par l’interaction entre les utilisateurs et la production de contenus par ces derniers.