Algorithmique Appliquée

BTS SIO SISR

Introduction à la programmation et à l'algorithmique

Plan

- Intérêt du cours
- Discussion sur les algorithmes
- Histoire de l’algorithmique
- Définition formelle
- Architecture simplifiée d'un ordinateur
- Discussion sur les langages de programmation
- Introduction au langage Python
- Types numériques, expressions et objets en Python
- Variables et assignation

Intérêt du cours

★ ★ ★ ★ ★

👉 Intérêt de savoir programmer

Selon vous, à quoi sert de savoir programmer ?

Quelques exemples

- Automatiser votre travail pour rentrer plus tôt chez vous le soir ⏰
- Ecrire une App révolutionnaire et devenir millionnaire 💰
- Amuser vos amis avec vos propres jeux vidéos 👾
- Simplifier la mise en relation de milliers de personnes 👥

Et l'algorithmique dans tout ça ? 😲

L'algorithmique est au ❤️ de la programmation

Qu'est-ce que je vais apprendre d'utile ? 😐

- Le socle de la programmation : l'**algorithmique** 💻
- Un langage de programmation industriel : **Python** 🐍
- Les bases pour écrire un logiciel **robuste** 💪, **performant** 🔥 et **maintenable** 👍

Euh... Attendez, j'ai choisi SISR, pas SLAM 😠

- Ce cours fait bien parti de votre programme.
- De plus en plus d'entreprises s'organisent en mode **DevOps**.
- La capacité à **automatiser** des tâches est également capital dans votre futur métier.

Discussion sur les algorithmes

👉 Auriez-vous des exemples d'algorithmes en tête ?

Exemple informel 1

Aller sur Internet 💻

- Brancher l'ordinateur.
- Appuyer sur le bouton Power.
- Vérifier la connexion réseau.
- Cliquer sur l'icône du navigateur web.
- Entrer une URL.
- Tapper sur la touche *Entrée*.

Exemple informel 2

Cuisine 🍰

- Prendre les ingrédients.
- Suivre les étapes de la recette.
- Mettre au four.
- Tant que ce n'est pas cuit, attendre.
- Sortir le plat du four.

Exemple informel 3

Choisir le petit déjeuner ☕

- S'il reste du café:
  - Faire chauffer du café.
  - Prendre une tasse.
  - Verser le café.
- Sinon:
  - Prendre un paquet de céréales.
  - Prendre un bol.
  - Verser du lait.

Exemple 4

Multiplication calculée avec une somme

Mathématiques : $x \times y = \sum_{0}^{y-1} x$

Python :

```
def multiplie(x, y):
    resultat = 0
    for _ in range(y):
        resultat += x
    return resultat
```

👉 A vous de trouver des idées

Intuitions à ce stade...

Un algorithme :

- a des **entrées** et des **sorties**,
- comporte une **suite d'instructions**,
- peut comporter des **conditions**,
- peut comporter des **répétitions** (boucles).

Histoire de l'algorithmique

👉 **Selon vous, de quelle époque datent les premiers algorithmes ?**

Babylone

- Les mathématiques Babyloniennes de l'ancienne Mésopotomie (actuellement l'Iraq) montrent les premiers algorithmes.
- Des premières tablettes d'argile Sumérienne trouvées près de Bagdad remontent à **2500 ans avant notre ère**.
- Ces algorithmes étaient utilisés pour prédire la date et le lieux de certains événements astronomiques.

Egypte

- Les mathématiques de l'ancienne Egypte utilisaient également des algorithmes pour résoudre des problèmes arithmétiques.
- Certains papyrus, comme celui de Rhind, remontent à **1550 ans avant notre ère**. Ils cherchaient à calculer des volumes, des aires et à résoudre des problèmes géométriques... notamment en relation avec des pyramides !

Grèce Antique

- On doit aux mathématiques Hellenistiques, qui remontent à **300 ans** avant notre ère, de nombreux algorithmes encore célèbres aujourd'hui.
- Deux exemples notables :
  - Le *Siève d'Eratosthenes* : permet de trouver tous les nombres premiers jusqu'à une certaine limite.
  - L'*algorithme d'Euclide* : permet de calculer le Plus Grand Commun Dénominateur (PGCD).

Muhammad ibn Musa al-Khwarizmi

- Il était un mathématicien, astronome, géographe et professeur à la Maison de la Sagesse à Bagdad. Il a vécu au **9e siècle** de notre ère.
- Il était le mathématicien le plus lu au moyen-âge en Europe.
- Le mot "algorithme" vient de la **traduction du nom Al-Khwarizmi** d'abord en latin (Algorizmi) puis en ancien anglais (algorism). Ce mot avait alors une signification différente.

Ada Loveplace

- La comtesse Ada Loveplace a vécu au milieu du **19e siècle** au Royaume-Uni.
- Elle a travaillé avec Charles Babbage sur des machines analytiques capables théoriquement d'exécuter n'importe quel algorithme.
- Ada est considérée comme pionière car elle a écrit les premiers algorithmiques dédiés à une exécution par un ordinateur.
- Un langage de programmation créé beaucoup à la fin du 20e siècle a été nommé Ada en sa mémoire.

Alan Turing

- Ce mathématicien, informaticien, scientifique et philosophe a vécu au **début du 20e siècle**.
- Ce personnage est historique au-délà de la discipline de l'algorithmique puisqu'il a fortement contribué à la victoire contre les Nazis lors de la 2e guerre mondiale.
- L'une de ses contributions essentielles est la **Machine de Turing**, qui est le premier ordinateur généraliste (par rapport à des ordinateurs spécialisés commes les calculatrices).

Donald Knuth

Ce professeur émérite en informatique de l'université de Stanford a contribué à la démocratisation de l'algorithmique au travers de ses ouvrages *The Art of Computer Programming*, sur lesquels il travaille toujours **aujourd'hui**.

Anecdotes récentes

- Google s'est construit autour de l'algorithme *Map Reduce*, inventé par ses fondateurs.
- Bill Gates a indiqué que Microsoft recrute n'importe quelle personne qui comprend les ouvrages de Knuth.
- Les géants Américains (GAFAM) et Chinois (BATX) utilisent tous des algorithmes et recherchent tous les meilleurs candidats dans ce domaine.
- L'Intelligence Artificielle est simplement un ensemble d'algorithmes travaillant sur des données massives.

Définition formelle

Un algorithme est une liste **finie** d'instructions décrivant un ensemble de calculs qui, lorsqu'ils sont exécutés sur un ensemble d'entrées, va passer par une séquence d'états bien définis et finalement produire une sortie.

Architecture simplifiée d'un ordinateur

Un champ limité d'actions

Un ordinateur ne sait faire que 2 choses :

- Faire des calculs,
- Se souvenir du résultat de ces calculs.

Mise en perspective

Un simple ordinateur de bureau standard sait :

- Faire beaucoup de calculs très rapidement : des **centaines de milliards de calculs par seconde**.
- Se souvenir qu'une quantité remarquable de données : 1 téraoctet (To), c'est **1 000 000 000 000 octets**. Changez l'unité en kilogrammes ou mètres...

Champ encore plus limité avant

Les premières machines étaient à **programme fixe** :

- Une calculatrice avec les opérations $+$, $-$, $\times$, $\div$.
- Un calculateur de trajectoire de missiles.
- Un solveur de système d'équations linéaires.
- Etc.

Les systèmes embarqués utilisent encore ce procédé pour des raisons de coût et de performance.

Capacité à exécuter différents programmes

Vos machines sont dites à **programmes stockés** :

- Elles ont des composants qui stockent les programmes.
- Elles ont des composants qui lui permettent d'exécuter les instructions de ces programmes.

Architecture logique simplifiée d'un ordinateur

Un ordinateur comporte :

- Des **entrées** (clavier, souris, ...) et des **sorties** (écran, son, ...).
- Des **mémoires** (disque, RAM, caches, ...).
- Des **calculateurs** (CPU, GPU, ...) ayant :
  - **Unité de contrôle** (compteur de programme, ...).
  - **Unités arithmétique et logique** (+, −, ...).

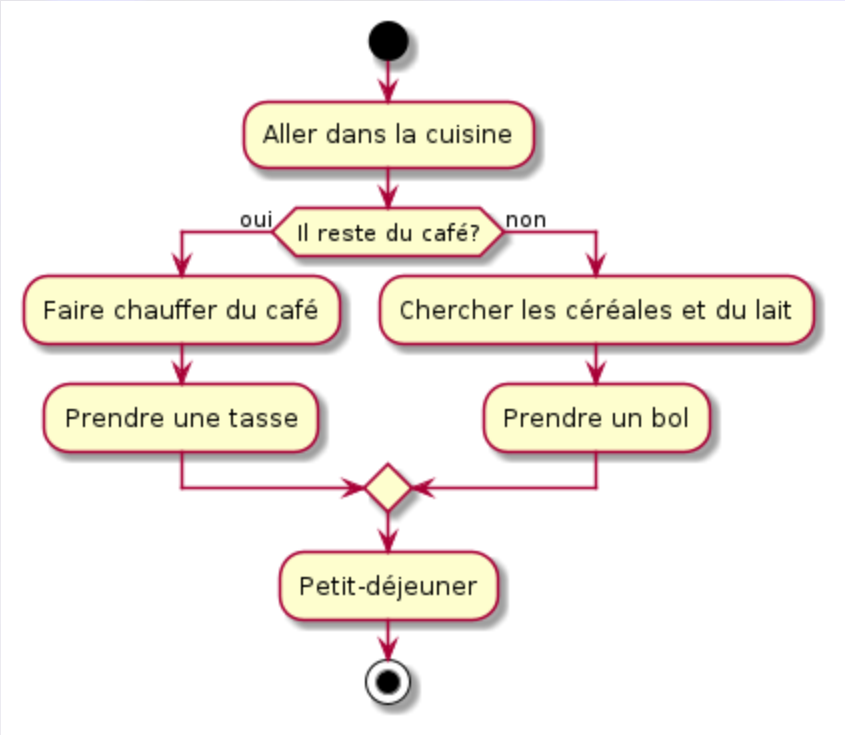
Cycle de vie simplifié d'un programme

- La séquence d'instructions du programme est chargé en mémoire.
- Un programme particulier appelé **interprêteur** exécute chaque instruction dans l'ordre :
  - Un **compteur de programme** pointe en mémoire vers la prochaine instruction à exécuter.
  - Cette instruction est envoyée à l'**unité arithmétique et logique** qui s'occupe de la résoudre.

Flot de contrôle

- Dans certains cas, sur la base d'un test, l'interprêteur saute vers une autre séquence d'instructions.
- C'est ce que l'on appelle le **flot de contrôle**.
- Cela nous permet d'écrire des programmes complexes.
- Une partie de ce premier cours et du prochain est dédié à l'étude des **structures de contrôle**.

Graphe de flot de contrôle

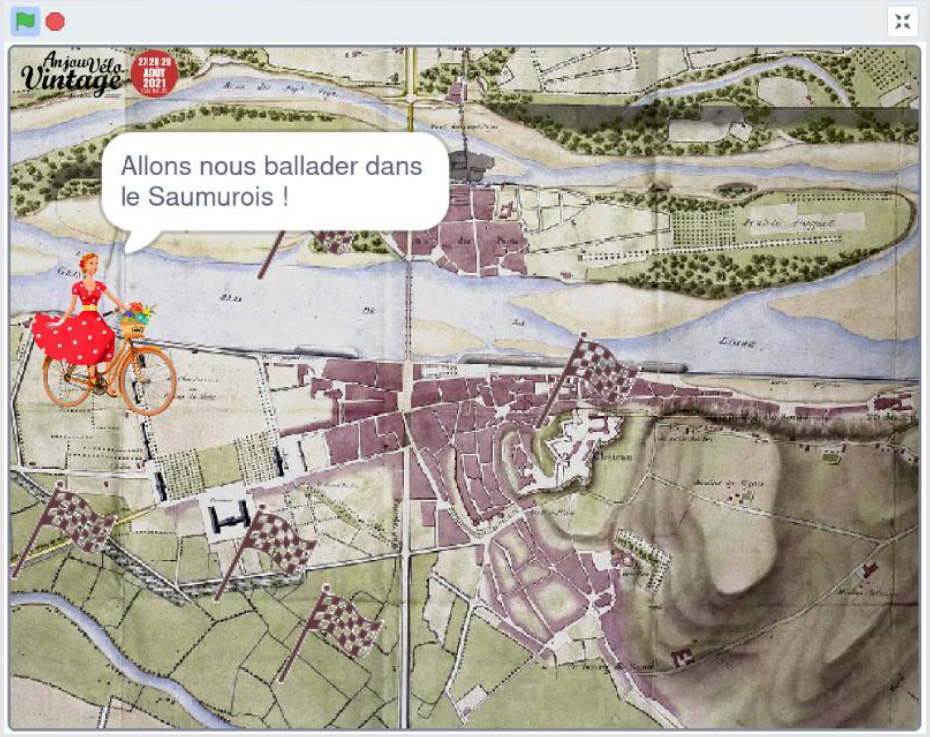


TP 01 - Démarrer avec Scratch

- **Scratch** est un langage de programmation graphique à destination des plus jeunes.
- Créé par le **MIT**.
- **Démo** : Editeur Scratch.

TP : Jeu Vidéo avec Scratch

- **Lien** vers le sujet de TP.
- **Contexte** : Anjou Vélo Vintage.



Discussion sur les langages de programmation

Intuition

- Un langage de programmation offre une **syntaxe** pour spécifier les **instructions** que l'interprêteur doit exécuter.
- Il existe de nombreux langages de programmation.
- **Scratch** et **Python** sont des langages de programmation.

Machine de Turing

- En 1936, Alan Turing : spécification de la **Machine de Turing Universelle**.
- Avec quelques **instructions élémentaires**, et une **mémoire**, il est possible d'écrire **n'importe quel programme** exécutable par une machine.

Thèse de Church Turing

Si une fonction est exécutable par une machine, alors une Machine de Turing peut être programmée pour l'exécuter.

➡️ Un algorithme peut donc être écrit dans **n'importe quel** langage de programmation.

Indécidable : tout n'est pas possible

- Certains problèmes **ne peuvent pas** être résolus par un algorithme.
- **Problème d'arrêt** (Halting Problem) : il est impossible d'écrire un programme capable de prédire si un autre programme arbitraire s'exécute indéfiniment.
- Cela signifie par exemple qu'il est impossible pour un programme de prédire si un autre programme fonctionne correctement.

Turing-complet

- Un langage de programmation est dit **Turing-complet** s'il permet de simuler une machine de Turing.
- Autrement dit, il permet d'écrire tout programme qu'une machine peut exécuter.

Retour sur Scratch

- Scratch 🐱 est **Turing-complet**.
- Scratch comporte de **nombreuses instructions** sous forme de blocs.
- On peut donc écrire n'importe quel algorithme avec Scratch.
- Le programme suivant calcule 1 + 1 :

Autre exemple : Brainfuck

- Brainfuck 🧠 est **Turing-complet**.
- C'est un langage *minimal* qui comporte seulement **8 instructions élémentaires**.
- Ce langage montre qu'il est possible d'écrire n'importe quel programme avec une **syntaxe illisible** pour un être humain.
- Le programme suivant calcule 1 + 1 :

```
+>+[-<+>]<
```

Caractéristiques d'un langage

- **Primitives élémentaires** : litéraux (`42` , `3.14`), chaînes de caractères (`"yo"`), opérateurs (`/` , `+`).
- **Syntaxe** : ensembles de séquences bien formées (exemple : `1 + 1` est bien formé, mais pas `1 1`).
- **Sémantique** : associe une signification aux séquences de symboles bien formés.

Paradigmes de programmation

- **Langage déclaratif** : suite de déclarations de faits ou états.
  - *Exemples* : HTML, CSS, CSV.
- **Langage impératif** : suite d'instructions à exécuter.
  - *Exemples* : Python, Scratch, Brainfuck.
- Il existe d'autres paradigmes : orienté-objet, orienté-aspect, fonctionnel, générique, etc.

Compilé ou interprêté

- Langage **compilé** : le code source du programme est traduit en *code machine binaire* exécuté directement par le matériel (processeur, carte graphique, etc.).
  - *Exemples* : Java, C, C++, Rust.
- Langage **interprêté** : le code source du programme est lu par un programme appelé *interprêteur* qui exécute ce code.
  - *Exemples* : Python, Scratch, JavaScript, HTML, CSS.

Bas niveau ou haut niveau

- **Bas niveau** : manipulation de données et d'opérations proches de la machine.
  - *Exemple* : Déplace le contenu d'un registre de 32bit vers cet autre registre.
  - *Langages* : Assembleur, C.
- **Haut niveau** : manipulation d'abstractions de haut niveau fournies par le langage.
  - *Exemple* : Affiche un bouton OK à telle localisation.
  - *Langages* : Java, Python, JavaScript.

Général ou spécifique à un domaine

- **Généraliste - General Purpose** : langage applicable à une vaste variété de domaines.
  - *Exemples* : Python, Java, C, C++, Assembleur.
- **Domain-Specific Language (DSL)** : les opérations primitives du langage sont spécifiques à un domaine particulier.
  - *Exemples* : SQL, HTML, CSS.

Introduction au langage Python

Quelques mots d'introduction

- Langage généraliste, interprêté, de haut niveau, et multi-paradigmes (impératif et orienté-objet).
- Langage simple à apprendre.
- Communauté très active : très nombreux outils, tutoriels et très bonne documentation.
- Hommage aux Monty Python.

Quelques dates

- **1991** : Création du langage par Guido van Rossum.
- **2000** : Sortie de la version 2.0 et début de l'essor.
- **2008** : Sortie de la version 3.0 non-compatible avec la version 2.0 ; peu utilisé au début.
- **2015** : Sortie de TensorFlow et essor exponentiel de l'usage du langage.
- **2020** : Fin de vie de la version 2.0.
- **2021** : *Vous* apprenez Python !

Quelques applications écrites avec Python

- **RankBrain** : algorithme derrière Google.
- **YouTube** : streaming vidéo.
- **Rover Persévérance (Mars)** : communication entre rover et satellite.
- **Abaqus/CAE** : interface graphique 3D de calcul par élements finis.

Quelques domaines de prédilection

- **Intelligence Artificielle** : Machine Learning, Deep Learning, Data Science.
- **Automatisation de tâches** : Scripts, Intégration Continue, Déploiement Continue.
- **Recherche** : Recherche Opérationnelle, Calcul Scientifique.
- **Backend Applications** : Développement Web côté serveur.
- **Internet of Things** : Développement de prototypes.

Mais...

- Moins *performant* et moins *structuré* que d'autres langages comme C++ et Java.
- Ne peut s'exécuter dans un navigateur web directement contrairement à JavaScript.

Types numériques, expressions et objets en Python

Déclarations

- Un programme (également appelé **script**) Python est composé de **déclarations**.
- Chaque déclaration dit à l'interpêteur Python ce qu'il doit faire.
- Par exemple :

```
print("Bonjour tout le monde")
```

➡️ demande à l'interprêteur d'afficher `Bonjour tout le monde`.

Objet

- En Python, **tout est objet**.
- Chaque objet possède un **type**.
- Un type peut être :
  - **Scalaire** s'il est indivisible.
  - **Non-scalaire** s'il est décomposable.

Types scalaires

- **int** : nombre entier naturel (`0` , `8` , `-12`).
- **float** : nombre réel, dit à virgule flottante (`1.5` , `3.14` , `-6e10` , `5e-6`).
- **bool** : vrai ou faux (`True` , `False`).
- **None** : représente l'absence de valeur (`None`).

Expressions

- On combine des objets et des **opérateurs** pour former des **expressions**.
- Le résultat de l'évaluation d'une expression s'appelle **valeur de l'expression**.
- Exemple :

```
1 + 1
```

```
2
```

Autres exemples d'expressions

```
4.5 + 2.3
```

```
6.8
```

```
-5 < 3
```

```
True
```

```
2 == 3
```

```
False
```

Opérateurs arithmétiques

- `+` : addition.
- `-` : soustraction.
- `*` : multiplication.
- `/` : division.
- `%` : modulo (reste de la division).
- `//` : division entière.
- `**` : puissance.

Comparaisons

- < : strictement plus petit que.
- <= : plus petit ou égal.
- > : strictement plus grand que.
- >= : plus grand ou égal.
- == : égal.
- != : différent.

Opérateurs Booléen

- `and` : ET logique.
- `or` : OU logique.
- `not` : NON logique.

Démo

Utilisation d'un shell Python

Démonstration de l'utisation d'opérateurs pour former des expressions

Variables et assignation

Variable

- Il est possible de **lier** un objet à un **nom** :

```
x = 42
```

- On dit que `x` est une **variable** liée à un objet de type `int` et dont la valeur est `42`.

Assignation

- Le fait de lier un objet à une variable s'appelle une **assignation**.
- Il est possible de réassigner un nouvel objet à une variable :

```
x = 42
y = x    # y vaut 42
x = 314
z = x    # z vaut 314
```

Typage dynamique

- Une variable peut se voir assigner n'importe quel type d'objet.
- En particulier, il est possible (bien que **déconseillé**) d'assigner un nouvel objet d'un type différent du type initial : on parle de **typage dynamique**.

```
x = 42
y = x      # y vaut 42
x = True
z = x      # z vaut True
```

TP 02 - Python avec Jupyter Notebook

- **Jupyter Notebook** est un environnement de développement Python.
- Adapté pour la recherche, l'enseignement, le prototypage, etc.
- **Binder** offre la possibilité d'exécuter un carnet Jupyter dans une page web.
- **Démo** : launch binder

TP : Familiarisation avec Python et Jupyter Notebook

Lien vers le sujet de TP.