INTRODUCTION AU LOGICIEL (INL) :

Le but de ce cours est de comprendre ce qu'est un logiciel, comment on le conçoit et comment on le distribue.
En informatique, un logiciel est un **ensemble de séquences d'instructions** interprétables par une **machine** et d'un jeu de **données** nécessaires à ces **opérations**.

Au cœur de la machine, il y a toujours le processeur, le cerveau de la machine, c'est lui qui gère toutes les informations, lorsqu'on programme, on « s'adresse » directement au processeur, et en fonction de ce dernier, on ne lui parlera pas de la même manière.

Un bon informaticien doit savoir bien découper l'information et très vite, à partir des informations fournis par le client, il visualise comment les réaliser (esprit d'analyse).
Le client va expliquer ce qu'il compte faire et c'est à l'informaticien d'extirper les informations et de dissocier les données des opérations à faire, jamais il ne parlera de données, sa demanda sera probablement très implicite, il faudra donc aller chercher un petit peu.

Les données :

Prenons l'exemple de Flappy Bird :
Les données ?
-Position de l'oiseau (x, y)
-Tap utilisateur
-Obstacles
-État du jeu (arrêté ou en cours)
-Éléments graphiques

Pour Word :
-Contenu
Avec le style, les images et les tableaux
-Nom du fichier
-Droite d'écriture (sans elles, à chaque enregistrement de données, le logiciel bug)
Définition de Métadonnées : Données sur une données
Les métadonnées sont des données qui en décrivent d'autres. Dans la plupart de ses usages informatiques, le préfixe méta signifie « définition ou description de référence ».
Les métadonnées synthétisent des informations élémentaires sur les données, elles facilitent la recherche et la manipulation d'instances de données particulières.

Exercice : pour Facebook
-Les photos
-Les noms, prénoms, marques
-Les commentaires
-Les pubs (très ciblées)
-Les amis
-Les liens
-Post
-Curriculum Vitae
…

Démarches :

Le schéma de base de données est la manière de procéder pour créer un logiciel, la première étape est de récolter et de stocker les données, sans penser à leur traitement ou à leur utilité, la deuxième étape consiste à les structurer, et seulement enfin, on encode les données dans un SGBD (Système de gestion de base de données).
→ Voir Slides du cours pour un peu plus d'information
SQL (Structured Query Language) : langage qui permet de sélectionner des éléments désirés en fonction de critères
-On va sélectionner les données en fonction des requêtes et des besoins.

L'information et son reflet :
https://interstices.info/tout-a-un-reflet-numerique/

Les opérations :

Quand les données sont rassemblées, il faut maintenant les traiter de manière à accéder à la requête.

Reprenons l'exemple de Flappy Bird :
Les opérations ?
-Faire monter l'oiseau
-le faire tomber
-Vérifier les obstacles
-Afficher les images successives de l'oiseau
Comment animer l'oiseau ?
→ Sprites : un ***Sprite***, ou **lutin**, est dans le jeu vidéo un élément graphique qui peut se déplacer sur l'écran.
→ Tiles : ensemble des éléments dynamiques d'un jeu et qui peuvent constituer le décor, peuvent se confondre avec les Sprites

Pour Word :
-Écrire du texte
-Copier/Coller
-Enregistrer
-Redimensionner la fenêtre

Pour Facebook :
-S'inscrire
-Se connecter
-Ajouter un ami
-Écrire un poste
-Rechercher
-Liker

Qui intervient dans le développement ?

-Client et Utilisateur
Client→ Celui qui **fait** la requête
Utilisateur→ Celui qui va **utiliser** le logiciel
-L'analyste

→ Celui qui va parler au client et étudier ses besoins pour en faire une liste
-Le développeur
→Il prend la liste de l'analyste et encode de manière à faire le logiciel souhaité
-Analyste-développeur
→Celui qui écoute le client, récolte ses besoins et les exauce (reprend les fonctions des deux précédents)
-Les testeurs
→ Celui qui test le logiciel et signale ce qui ne va pas
-Administrateur système
→ Gérer le linux, le système pour vérifier que tout se passe, notamment pour la sécurité du système
-Tous les autres
→Ceux qui vont intervenir dans le développement : graphistes (pour le design), comptables (pour les comptes), juristes (pour la charte du logiciel), etc.
GDPR → Loi européenne sur la protection des données, tous les développeurs de logiciel doivent la signer

Combien de personnes et de temps ?

-Tout dépend du projet
-Et de la méthodologie (pour le temps en tout cas)
Les méthodologies aident à travailler mieux, en suivant dans l'ordre certaines étapes, elle peut même nous indiquer combien de temps il faut pour chaque étape et avec qui on est censé les faire. (Voir slides sur « gérer le développement »)

Comment produire un logiciel ?

Plusieurs étapes :

1) Préanalyse : on discute avec le client pour savoir ce qu'il veut avec un maximum de détails (premières rencontres, budget, temps…) après cela, on sait si on va vers le GO (le projet nous parle, on ne sait pas encore ce qu'on veut faire mais on sent que c'est possible) ou le non-GO (le projet ne semble pas abordable ou le client a donné des raisons de douter)
2) Analyse : maintenant qu'on sait qu'il est possible de travailler sur le projet, on cherche une manière de parvenir à satisfaire le client et pour cela, il faut comprendre le problème (dialogue avec le client et/ou les utilisateurs, trouver les données et les opérations, …)
3) La conception : On apporte une solution conceptuelle qui fonctionne sur papier, via des prototypes (Proto.IO).
4) Le développement : On crée une application à partir de la conception
5) Les tests : On teste l'application développée (tests d'une opération, d'une fonctionnalité, de l'application entière, etc. {pas savoir})
6) Le déploiement : On installe l'application chez l'utilisateur (mise en place de l'infrastructure nécessaire, installation des logiciels tiers, formation des utilisateurs, etc.) et on le lance ensuite sur une plateforme (App store, Windows store)

-UML : Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.
-**Mockup** : En informatique, le terme mockup désigne un prototype d'interface utilisateur. Un mockup a ainsi pour rôle de présenter les idées sur l'utilisation d'un logiciel.
→ Idée reçue : c'est la faute de l'ordinateur !

Qu'est-ce que « développer un logiciel ? »

-Langage binaire (0 et 1 bien ordonné)
Rappel :
1 bit = 0 ou 1
1 byte/octet= 8 bits
-Langage naturel (Demander le nom de l'utilisateur, son mot de passe→ si les infos sont correctes alors accepter la connexion, sinon afficher un message d'erreur)
-Langage de programmation (Java, C++, Kotlin (google), etc.)
-Algorithmique
-Outils de développement
-Niveau de langage
-Performance
Entre le langage naturel et le langage binaire ? → langage de programmation

-Un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent
-**Un algorithme** est une suite finie et non-ambiguë d'instructions permettant de donner la réponse à un problème.

Chaque langage de programmation à sa grammaire (et parfois aussi chaque version d'un même langage de programmation à sa propre grammaire. Ex : Grammaire Java version 1 différent Grammaire Java version 11)

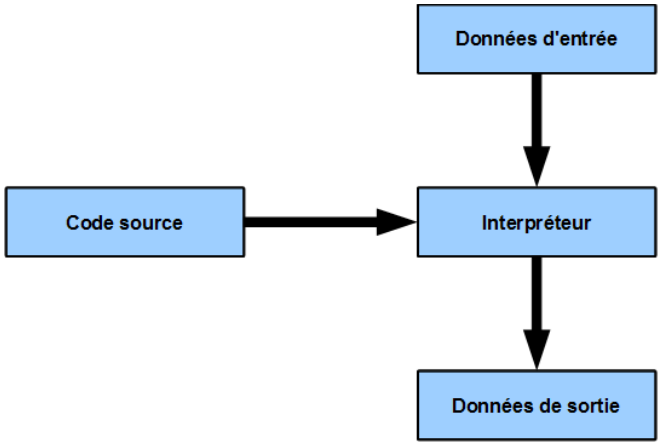
Buts :

- Permettre de déterminer si un texte est valide pour ce langage
- S'il est valide, on peut le transformer en langage binaire que l'ordinateur comprendra

-Programme (.text) → Programme (.exe)
Compilateur
Le compilateur :
Logiciel permettant de :
-Vérifier que le texte correspond à la grammaire du langage
-Transformer un texte valide dans un langage interprétable
Le processeur ou interpréteur peut alors exécuter le code produit
Un compilateur → Un langage

Langage compilé et interprété :

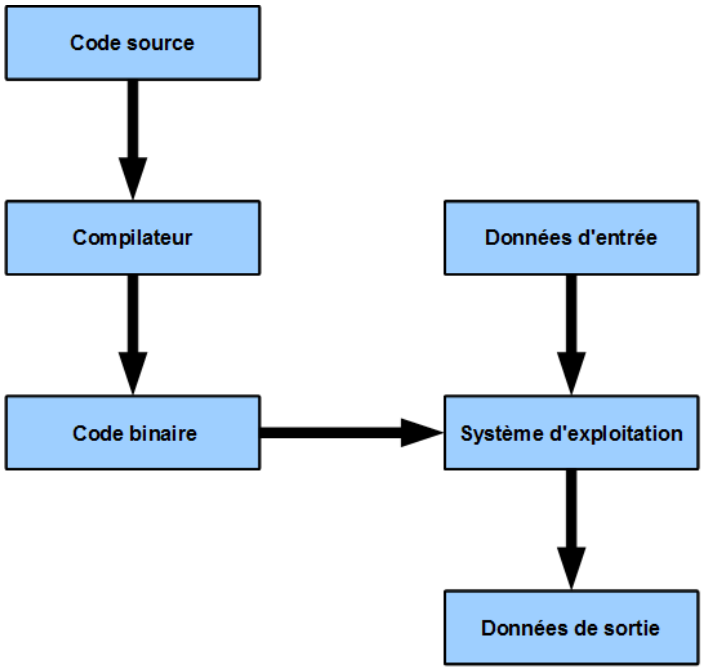
On peut distinguer deux grands types de langages : les langages interprétés et les langages compilés.
Pour les langages supportés sur le site on a :
-langages interprétés : Java (+ JavaScool) et Python ;
-langages compilés : C, C++, Pascal et OCaml.
Langages interprétés
Dans ces langages, le code source (celui que vous écrivez) est interprété, par un logiciel qu'on appelle interpréteur. Celui-ci va utiliser le code source et les données d'entrée pour calculer les données de sortie :



L'interprétation du code source est un processus « pas à pas » : l'interpréteur va exécuter les lignes du code une par une, en décidant à chaque étape ce qu'il va faire ensuite.

Langages compilés

Dans ces langages, le code source (celui que vous écrivez) est tout d'abord compilé, par un logiciel qu'on appelle compilateur, en un code binaire qu'un humain ne peut pas lire mais qui est très facile à lire pour un ordinateur. C'est alors directement le système d'exploitation qui va utiliser le code binaire et les données d'entrée pour calculer les données de sortie :



Principales différences :

On pourrait discuter très longtemps des avantages et inconvénients des différents types de langages mais les deux points qui sont les plus intéressants sont les suivants :

Dans un langage interprété, le même code source pourra marcher directement sur tout ordinateur.

Avec un langage compilé, il faudra (en général) tout recompiler à chaque fois ce qui pose parfois des soucis.

Dans un langage compilé, le programme est directement exécuté sur l'ordinateur, donc il sera en général plus rapide que le même programme dans un langage interprété.

En pratique

Selon le langage que vous choisissez il y aura, ou non, cette étape de compilation. Ne soyons donc pas étonnés si, dans le prochain cours, il faut 1 ou 2 étapes selon le langage choisi.
→ « Paradigme » ou « manière de penser »
Un paradigme est une représentation du monde, une manière de voir les choses, un modèle cohérent du monde qui repose sur un fondement défini.

Outils de développement :

Software Development Kit (SDK)
→ Un kit de développement logiciel, Software Development Kit (**SDK**) ou devkit en anglais, est conçu pour un ou plusieurs langages de programmation ; une ou plusieurs cibles (plateforme, jeux vidéo, etc.).
Environnement de développement ou Integrated Development Environment (IDE)
→ vi, vim, Notepad++, NetBeans, Eclipse, XCode (OS X & iOS), Android Studio, etc.
→En programmation informatique, un **environnement de développement** est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels. Il comporte un éditeur de texte destiné à la programmation, des fonctions qui permettent, par pression sur un bouton, de démarrer le compilateur ou l'éditeur de liens ainsi qu'un **débogueur** en ligne, qui permet d'exécuter ligne par ligne le programme en cours de construction. Certains environnements sont dédiés à un langage de programmation en particulier.

SDK :

Compilateur : Un compilateur est, en informatique, le terme utilisé pour désigner un programme qui transforme un code source écrit dans un langage de programmation
Interpréteur : En informatique, un interprète, ou interpréteur est un outil ayant pour tâche d'analyser, de traduire et d'exécuter les programmes écrits dans un langage informatique.
Débogueur : Un débogueur (ou *débugueur*, de l'anglais debugger) est un logiciel qui aide un développeur à analyser les bogues d'un programme. Pour cela, il permet d'exécuter le programme pas-à-pas, d'afficher la valeur des variables à tout moment, de mettre en place des points d'arrêt sur des conditions ou sur des lignes du programme ...
Librairies : En informatique, une bibliothèque logicielle est une collection de routines, qui peuvent être déjà compilées et prêtes à être utilisées par des programmes.

Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en byte code destiné à la machine virtuelle Java. Il existe plusieurs éditions de JDK, selon la plate-forme Java considérée.
L'environnement d'exécution Java (JRE), parfois nommé simplement « Java », est une famille de logiciels qui permet l'exécution des programmes écrits en langage de programmation Java, sur différentes plateformes informatiques.
→ JDK différent de JRE

Syntaxe vs Sémantique :

Problème de syntaxe :
Mon chat est **blan.**

Problème de sémantique :
Mon chat **gris** est blanc.

-Syntaxe → Problème lié à l'orthographe, un programme avec des fautes de syntaxe ne marchera pas.
-Sémantique→ Problème lié à ce que fait le programme, un programme peut être écrit dans une bonne syntaxe mais sera faux s'il ne fait pas ce qui était demandé ou n'a pas de sens.

Générations & Niveaux de langages :

1ère génération (bas niveau) :
2ème génération (bas niveau)
3ème génération (haut niveau)
4ème génération (haut niveau)
5ème génération (très haut niveau)
→ Plus un langage a un niveau bas, plus son niveau de langage est proche du processeur

- 1ère et 2ème génération (bas niveau) : Proche du matériel informatique, généralement dépendant du matériel utilisé.

→ ligne de code ≅ Une opération informatique

2ème génération (bas niveau) : langage assembleur (MOV AX, 3)

- 3ème et 4ème génération (haut niveau) : Plus proche du langage naturel, généralement indépendant du matériel utilisé.

→ligne de code ≅ Plusieurs opérations informatiques.

-3ème génération (haut niveau) : Fortran, Cobol, Basic, Pascal, C++, Java, Python, Ruby…
Langages utilisables pour tout type de programme
-4ème génération (haut niveau) : SQL, Matlab, Maple…
Langages utilisables pour un problème spécifique => Langage dédié ou DSL (Domain Specific Language)
-5ème génération (très haut niveau) : Prolog, Haskel…
Langages grâce auxquels le développeur exprime son problème et où la résolution est automatisée (Du moins, c'est le but !)

Performance :

Langage de bas niveau :
→ beaucoup à écrire mais performant
Langage de haut niveau :
→ moins à écrire mais moins performant

Résumé :

-Langage binaire
-Langage naturel
-Langage de programmation
-Algorithmique

-Niveau de langage
-Performance

Science, technique et art :

Démarche informatique :

-Client ?
Produit fini ?
-Équipe de développement ?

En réalité, plus compliqué :
-Client (Sponsor), Utilisateur final ?
-Exécutables, ressources, documents, formations ?
-Analyste, manager, Dev, Ops ?
→ Voir slides sur les métiers et schémas

Projet (informatique) :

Définition : Tout ce qui touche à la réalisation d'un (d'une partie d'un) système informatique par une équipe de développement. Il y a un début et une fin, avec des délivrables.
• Exemple : « Créer un site web pour une entreprise de confitures pour gérer facilement les commandes. »

SI : Système d'Information :

• Définition : Toutes les informations collectées sur le projet. Les besoins de l'utilisateur final dans le logiciel mais aussi l'organisation de l'entreprise, l'esprit de l'entreprise, les règles internes de l'entreprise, …
• Exemple : il existe différentes confitures, différents packagings de pots, il y a 5 employés dans l'entreprise, l'entreprise est jeune et dynamique, …
• Ne prendre en compte que les informations liées au projet (pas « le patron a trois enfants », ou « la secrétaire chausse du 39 ».
Durant le premier entretien, il peut arriver que le client dérape dans l'explication de ce qu'il veut pour le programme.

SIA : Système d'Information Automatisé :

• Définition : la partie automatisée du système d'information (SI), tout ce qui tourne sur un processeur.
• Le SIA est aussi appelé le système informatique.
• Exemple : le site web qui est online, la base de données qui tourne, les données qui évoluent, …

Étapes de développement (récapitulatif détaillé) :

• Appréhender le problème (pré-étude)
• Analyser le problème
• Comprendre finement le problème (analyse)
• Définir et modéliser la solution (conception)
• Réaliser (implémenter) la solution
• Tester

• Déployer (mise en production)

→ Cycle de vie d'un projet
Maintenance évolutive : La maintenance évolutive est une branche de la maintenance principalement évoquée en matière de maintenance des logiciels. Elle consiste à faire évoluer une application, par exemple à la suite de demandes d'utilisateurs, pour modifier son comportement ou pour proposer de nouvelles fonctions.
Maintenance corrective : La maintenance corrective est l'élimination d'une avarie ou d'une altération dans le fonctionnement d'un élément matériel (appelé « bien » ou « entité » dans le jargon de la spécialité) par sa réparation, sa restauration à l'état antérieur ou son remplacement.
Planification et replanification :
Planification de la manière dont on va procéder pour faire ce que le client nous demande en temps et en heure, lorsqu'un imprévu ou un retard apparait (ce qui arrive souvent) et que le processus prend du retard, on procède à une replanification, il est inutile de garder le même planning et de se dépêcher pour tenir les engagements, mieux vaut modifier le programme de façon à faire l'essentiel et de répondre au maximum d'attente.

1) Pré-étude
   • Premières discussions avec le client
   • Résultat : Contrat qui fixe les règles du développement, appelé
   • Software Development Plan (SDP) (Méthodologie RUP)
   • Charte de projet
   • Plan d'approche
   • Project agreement
   • Quality plan

Software Development plan (SDP) :

• Doit être approuvé par les personnes impliquées :
• sponsor (financement) • responsable informatique (développement)
• responsable opérations (mise en place infrastructure réseaux, machines, logiciels, …)
• propriétaire (utilisateur final)

SDP :
•Contenu
• Domaine de définition (« scope »)
• Résultats (livrables, productions, « délivrables »)
• Planning
• Organisation
• Prix du projet
→ **Planning de Gantt** : Le diagramme de Gantt est un outil utilisé en ordonnancement et en gestion de projet et permettant de visualiser dans le temps les diverses tâches composant un projet. Il s'agit d'une représentation d'un graphe connexe, valué et orienté, qui permet de représenter graphiquement l'avancement du projet.

→ Établir un Budget :
• Depuis le planning
• estimer la charge de travail de chaque activité en nombre de jours-homme pour chaque activité
(Pour faire travailler une personne qualifiée pendant toute une journée entière : 650 euros.)

• 1 mois-homme ≃ 20 jours-homme

• 1 année-homme ≃ 200 jours-homme

• en déduire le nombre de personnes et/ou la durée de chaque activité
→ Voir exemple dans slides

Contraintes d'un projet :

Atteindre la meilleure qualité dans les budgets et le temps définis. (Qualité du SI, budget, échéance).
Suite dans l'AA Analyse I...

…comment designer un système d'information (UML)
… quel lien avec le développement
… comment les méthodes aident à organiser le travail
…

Chapitre 4 : Le déploiement

Rappel :
Comment produit-on un logiciel ?
-Préanalyse
-Analyse
-Conception
-Développement
-Tests
-Déploiement

Distribuer ? Déployer ?

Distribution/Déploiement : ensemble des activités qui rend un système informatique utilisable par les utilisateurs finaux.

1. Activités de déploiement
2. Coût et valeur du logiciel
3. Focus sur la distribution
a. Canaux de distribution
b. Stratégie de revenu
c. Droit d'auteur et licences

1. Activités de déploiement :

1. Produire le logiciel
2. Conditionner le logiciel
3. Distribuer le logiciel
4. Installer le logiciel

5. Effectuer la conversion
6. Assister les utilisateurs
7. Acceptation formelle
8. Tests bêta

- 1.Produire le logiciel :

-Ajouter à l'exécutable des scripts d'installation, une documentation utilisateur, des données de configuration…

- 2.Conditionner le logiciel :

-Conditionner les divers délivrables sur des supports, serveur, …

- 3.Distribuer le logiciel :

-Canaux de distribution
Définition : Le canal de distribution est la matérialisation du chemin suivi par un bien de son producteur au consommateur. On distingue généralement les canaux directs, courts et longs. L'ensemble des canaux utilisés forme le circuit de distribution d'un produit.
-Droit d'auteur et licences (voir § suivants)

- 4.Installer le logiciel :

-L'installation est souvent faite par l'utilisateur sauf pour de grands systèmes techniquement complexes.

- 5.Effectuer la conversion :

-Pour passer de l'ancien système au nouveau.
-Conversion de données dans un nouveau format.
-En maintenant, ou non, le système opérationnel.

- 6.Assister les utilisateurs :

-Formations
-Aide en ligne
-Helpdesk Vidéos ...
-Implique souvent la mise en place de suivi et de résolution de problèmes (maintenance)

- 7.Acceptation formelle :

-Tests qui assurent formellement que le produit est conforme aux spécifications
-Contractuel
-Implique le client et le fournisseur

- 8.Tests Bêta :

-Tests effectués par un échantillon représentatif d'utilisateurs.
-Mise en place de procédures pour enregistrer et analyser leurs réactions.

2. Coût et valeur d'un logiciel

Les coûts de :
- Main-d'œuvre pour la conception et la réalisation
- Licences associées au génie logiciel et à la production des applications développées
- Déploiement (équipe système, réseaux, …)
- Infrastructure (serveurs, salles, …)
- Formations
- Maintenance corrective et évolutive

→ Facile à quantifier

- Tout logiciel a une valeur :
  - Valeur d'usage
    - Découle du service qu'il rend
  - Valeur intrinsèque

- Expertise contenue au sein du code source
- Maintenabilité, extensibilité, réutilisabilité

Définition :
Valeur objective des marchandises et des actions d'une entreprise, basée sur la quantité de travail utile à la production des marchandises.
→ Très difficile à quantifier
Coût ≠ Valeur
→ Coût : coût factuel d'un produit, toutes les dépenses nécessaires pour faire le logiciel
→ Valeur : tous les bénéfices reçus grâce à l'application

3. Focus sur la distribution

- Canaux de distribution
- Stratégie de revenus
- Droit d'auteur et licences

a. Canaux de distribution :

- Commerce de détail
- Téléchargement
- Vente liée
- Fournisseur de solution (vente de plusieurs solutions complètes : pour une personne qui démarre une boite, une grosse entreprise peut lui fournir des logiciels, pc, tablettes, etc. pour l'aider).
- Suite
- App store
- …

b. Stratégie de revenu :

- Contrat de licences
- Contrat d'après-vente
- Vente liée
- Vente de logiciel spécifique (logiciel de niche : logiciel spécialement pour une clientèle réduite prête à payer)
- Vente de services liés à l'utilisation du logiciel
- Achats in-app - publicité
- …

4. Droit d'auteur et licences :

- Droit d'auteur
- Licence propriétaire
- Partagiciel
- Gratuiciel (Freeware avec achat in-app)
- Logiciel libre

Droit d'auteur :

- Vous évoluez dans un pays de droits
  - La création / la copie / l'usage d'un logiciel sont réglementés (voir votre futur cours de droit)

Licence ?

-21 Idée reçue : Je peux utiliser comme je veux tout ce que je trouve sur le web
→ Une licence de logiciel est un contrat « par lequel le titulaire des droits du logiciel autorise un tiers à poser des gestes qui autrement les enfreindraient »

Sur quoi ?

- Votre logiciel
- Toutes les librairies que vous utilisez
- Toutes les ressources que vous utilisez (images, …)
- Tous les logiciels tiers/services que vous utilisez

Limitation des droits :

- Classiquement, une licence limite les droits d'usage :
    - Interdiction de diffusion publique
    - Interdiction de reproduction, même partielle
    - ...

Absence de licence :

Si un logiciel est distribué sans licence :
- Pas de reproduction, adaptation, distribution sans l'accord exprès de l'auteur.
- L'auteur n'est pas protégé par une clause de non responsabilité contre des dommages occasionnés par l'usage de son logiciel

c
- Le fournisseur peut arrêter la maintenance du logiciel.

- Partagiciel (Shareware) :

Logiciel privatif diffusé gratuitement mais pour lequel une contribution est demandée (période d'essai). Ex : antivirus, Netflix, spotify, etc.
C'est donc un logiciel privatif mais distribué autrement.
→ Logiciel avec une période d'essai puis on demande de payer

- Gratuiciel (Freeware) :

Logiciel privatif gratuit mais ne donnant pas nécessairement d'autres droits. Parfois pas même celui de redistribution.

- Logiciel libre (Free software) :

Logiciel donnant de nombreux droits à ses utilisateurs.
-N'est pas équivalent à un Freeware !
-N'est pas nécessairement gratuit !
-Le logiciel libre et ouvert : révolution ou évolution ? Fiche pratique du CECIL : Les logiciels libres
→ Logiciel libre => Libre d'utilisation

→ Freeware = privatif mais sans source / Free software = non privatif mais accès aux sources

- Licence libre :

-Innovation juridique (et non pas technique)
-Permet de nouveaux modèles économiques adaptés à l'économie immatérielle. (Coût de transaction négligeable grâce à Internet)
-MII, GPL, LGPL, BSD…
→ 4 libertés communes :
● Liberté d'exécuter le programme pour tout usage
● Liberté de redistribuer des copies du programme reçu
● Liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins
● Liberté de rediffuser le programme modifié par ses soins

Petite mise en garde :

● Il y a beaucoup de types de licences libres
● Elles s'appuient sur le droit de la propriété intellectuelle
● L'auteur d'un logiciel original peut choisir une licence libre existante ou en créer une selon ses besoins.

Et l'Open Source ?

● Logiciel libre => 1 communauté
● Open Source => 1 autre communauté
  ○ Difficile de définir les différences profondes
  ○ L'open source peut être vu comme une méthodologie de développement au travers de la réutilisation du code source.

« L'open source est **une méthodologie de développement ;** le logiciel libre est **un mouvement social** ». Richard Stallman (extrait de développez.com)

Exercices :

● Quelle est la licence / Quelles sont les licences ?
● Puis-je l'utiliser pour tous mes projets ?
  ○ gratuit
  ○ commerciaux
● Suis-je limité dans les licences de mes projets développés avec ces technologies ? Java ? Ubuntu ? Windows 10 ? Google drive ? poESI ? RedHat ?
→ Voir Références (ou pas)