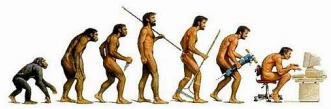
http://licence-info.univ-lyon1.fr/LIFAMI

LIFAMI
MATH INFO APP

LIFAMI
APPLICATIONS EN MATH ET INFO
CULTURE GÉNÉRALE DE PROGRAMMATION

Alexandre Meyer
Equipe SAARA, Laboratoire LIRIS
Université Lyon 1



1

LIFAMI
APPLICATIONS EN MATH ET INFO

Culture générale de programmation

2

Programmer (ou coder) ?

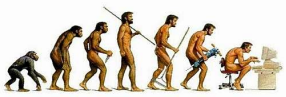
- Pourquoi faire ? Résoudre des problèmes
- Quel type de problème ?
  - Un ordinateur est meilleurs qu'un humain (ordinaire) dans peu de chose
    - Plus grande mémoire (mais moins bien organisée)
    - Tâches répétitives sans se lasser
  - Mais il ne comprend rien au monde
  - ➔L'ordinateur est un (bon?) outils pour l'homme
- Finalement c'est quoi coder ?

3

Programmer (ou coder) ?

- Finalement c'est quoi coder ?
  - Il faut tout expliquer à un ordinateur par une série de commandes, d'instructions que l'on peut répéter facilement (**boucle**)
  - On peut regrouper des instructions dans des groupes d'instructions=des **fonctions/procédures**
  - Il faut organiser les données, par ex. avec des **structures/tableaux**
- Ceci se fait dans un langage précis
  - Chaque langage aura ses spécificités et sa cible d'applications

➔Il faut du temps pour apprendre à coder
- Etre maniaque, ordonnée, précis
- Aimer ca (voir ca comme un jeu)



4

IEEE : top 10 des meilleurs langages de programmation de l'année 2019

En fonction de votre cible d'applications, certains langages sont à éviter

Rank	Language	Type	Score
1	Python		100.0
2	Java		96.3
3	C		94.4
4	C++		87.5
5	R		81.5
6	JavaScript		79.4
7	C#		74.5
8	Matlab		70.6
9	Swift		69.1
10	Go		68.0

5

IEEE : Top 10 des langages en 2016

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Open Ranking
1. C++		100.0
2. Python		98.7
3. C		97.9
4. Java		95.9
5. Swift		91.6
6. JavaScript		87.6
7. C#		85.8
8. Ruby		85.7
9. PHP		84.8
10. R		84.7

6

En 2019 langages d'après Github

Langages populaires en 2019 d'après Github

1. Javascript
2. Python
3. Java
4. PHP
5. C#
6. C++
7. TypeScript
8. Shell
9. C
10. Ruby

https://learnworthy.net/top-10-most-popular-language-of-2019-according-to-github/

7

Les langages indispensables d'après

http://dailygeekshow.com/programmation-internet-langage/

1. **Le Java** L3 : LIFAP7 et beaucoup en M1
2. **Le C** L1 : LIFAP1, LIFAMI
3. **Le C++** L2 : LIFAP3, LIFAP4; L3 : LIFAP6
4. **L'Objective-C** (propriétaire apple, proche du C++)
5. **Le C#** (propriétaire microsoft, proche du java)
6. **Le PHP** L1 : LIFASR2 L2 : LIFBDW1
7. **Le Python** L3 : LIFPROJET + M1
8. **Le Ruby**
9. **Le SQL** L2, L3 : LIFBDW?
10. **Le JavaScript** L2 : LIFAP5 L3 : LIFPROJET

8

Les langages indispensables d'après

http://dailygeekshow.com/programmation-internet-langage/

1. Le Java L3 : LIFAP7 et beaucoup en M1
2. Le C L1 : LIFAP1, LIFAMI
3. Le C++ L2 : LIFAP3, LIFAP4; L3 : LIFAP6
4. L'Objective-C
5. Le C#

Ces 5 langages ont

- **une syntaxe assez proche : {}, for, etc.**
- **sont typés (les variables doivent avoir un type)**

9

Les langages en Licence Info (Lyon 1)

- Le Java L3 : LIFAP7 et beaucoup en M1
- Le C L1 : LIFAP1, LIFAMI
- Le C++ L2 : LIFAP3, LIFAP4; L3 : LIFAP6
- Le PHP L1 : LIFASR2; L2 : LIFBDW1
- Le SQL L3 : LIFBDW2
- Le JavaScript L2 : LIFAP5

- Shell L1 : LIFASR1
- Scheme L1 : LIFAP2
- Assembleur L1 : LIFASR3
- ProLog L2 : LIFPROLOG (en option)

➔ **Pas idiot de savoir raisonner en algo (pseudo-langage)**

10

Les langages en Licence Info (Lyon 1)

- Le C L1 : LIFAP1, LIFAMI
- Assembleur L1 : LIFASR3
- Le C++ L2 : LIFAP3, LIFAP4; L3 : LIFAP6

➔ **Plutôt plus proche de la machine que les autres (bonne chose pour vous former)**

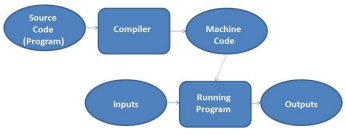
- Shell L1 : LIFASR1 ➔ **Spécifique au script**
- Scheme L1 : LIFAP2 ➔ **Programmation récursive**
- Le PHP L1 : LIFASR2; L2 : LIFBDW1 ➔ **pour le WEB**
- Le SQL L2 : LIFBDW1; L3 : LIFBDW2 ➔ **pour les Base de données**
- Le JavaScript L2 : LIFAP5 ➔**pour le WEB**
- ProLog L2 : LIFPROLOG (en option)
- Le C++ M1
- Le Java L3 : LIFAP7 et beaucoup en M1

➔ **Plutôt haut niveau**

11

Compilé ou interprété ?

- Compilateur (cas du C/C++)
  = un programme qui traduit
  un code source (.cpp pour nous) ➔ code assembleur ➔ exécutable
  + .exe (sous windows) peut s'executer seul
  - .exe compilé pour un processeur-X/OS-Y ne tournera pas ailleurs
  + programme rapide



12

Compilé ou interprété ?

- Interpréteur (sera le cas de Scheme, shell, etc.)
  = un programme qui exécute à la volée un code
  + code tournera sur toutes les machines disposant de l'interpréteur
  - plus lent

13

Paradigme de programmation

Un **paradigme de programmation** est un style fondamental de programmation informatique
- C'est une notion bien plus importante que les querelles de langages
- Un langage offre souvent plusieurs paradigmes de programmation

14

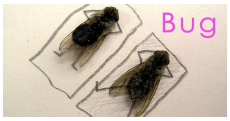
Paradigme de programmation

- Programmation **impérative**
  - Procédurale : LIFAP1, LIFAMI, etc.
  - Des procédures et des fonctions avec des boucles et des structures
- Programmation **déclarative**
  - Fonctionnelle : LIFAP3
    - Uniquement des fonctions, appels récursifs pour les itérations
  - Logique : LIFPROLOG
    - Description du problèmes et non de la recette menant à la solution
- Programmation **orientée objet**
  - En LIFAP3, LIFAP4, LIFAP6, LIFAP7, etc.
  - Les fonctions/procédures sont regroupées dans les structures
- Programmation évènementielle : LIFAP4, LIFIHM, etc.

15

Savoir débuguer quand on débute

Difficile quand on commence, voici quelques idées
- Il faut savoir ce qu'il y a dans les variables et par où on passe
  ➔faire des cout de debug
- Programmation incrémentale
  - Ecrire une fonction/procédure à la fois
  - Faire un main minimal
  ➔tester avec ce qu'on a, faire plusieurs tests
  Puis seulement passer à la suite si tout semble ok



= un insecte

1er plantage informatique était dû à un insecte coincé dans l'ordinateur

16

Savoir débuguer quand on débute

Si on a déjà écrit beaucoup de code sans avoir testé
- C'est mal, très mal !!!
  - Revient à trouver une aiguille dans une botte de foin
- Mettre en commentaire tout le code sauf une fonction et lancer le code comme ca, pour voir si ok ou non
- Décommenter la 2e fonction, seulement quand on est sûr que la 1ere fonction fonctionne
- Etc.

17