# Algorithmique et programmation (avec Python)

Cours de 16h - Université d'Angers

Axel-Cleris GAILLOTY

# Séances

vendredi 26 janvier : 14h-17h (salle 304)

vendredi 2 février : 14h-17h (salle 304)

vendredi 9 février : 14h-17h (salle 304)

vendredi 16 février : 14h-17h (salle 304)

vendredi 22 mars : 14h-18h (salle 304)

# A propos de moi

Axel-Cleris Gailloty

Développeur .NET/C# chez Groupe Créative (Nantes et Rennes)

Travaillé chez Harmonie Mutuelle

Travaillé pour la Poste (projet Automate)

Actuellement pour une filiale de Total (AS24)

Licence en Economie puis Master IEE à l'Université d'Angers

Commencé la programmation en 2017 avec R, puis Python en 2018, JavaScript en 2019, VBA en 2020, C# en 2021.

# Les sujets à aborder

Algorithmes et programmes

Les bases du langage Python

Webscraping

Développement d'interface graphique

Machine Learning

API

# Les ressources

Un répertoire Github qui contient le code qu'on écrira

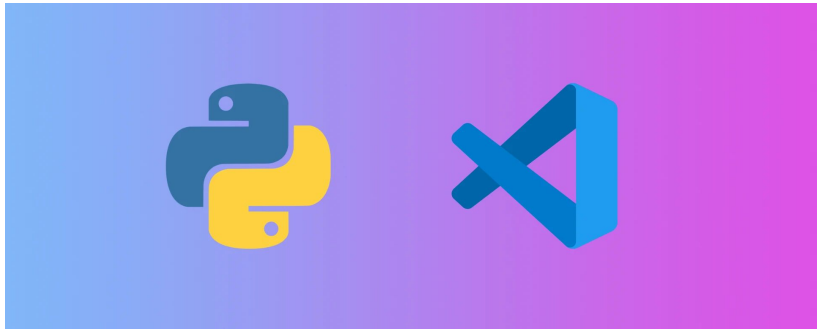
ALGOPY-2024

Python Crash Course

Python Cheatsheet - Python Cheatsheet

# Environnement de travail : Jupyter + VS Code

## Jupyter Lite



# Algorithmes et programmes

# Algorithmes et programmes

Algorithme : mot dérivé du nom du mathématicien Al Khwarizmi qui a vécu au 9ème siècle, et était membre de l'académie des sciences à Bagdad .

Un algorithme:

- est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat.
- est souvent exprimé avec une notation indépendante de tout langage de programmation.

# Algorithmes et programmes

Un algorithme prend des données en entrée, exprime un traitement particulier et fournit des données en sortie.

# Algorithmes et programmes

Un programme est la traduction d'un algorithme en un langage interprétable par un ordinateur.

# Savoir écrire un algorithme

Concevoir et écrire des algorithmes c'est décrire comment résoudre un problème particulier.

Il est donc indispensable de :

- Savoir formaliser son raisonnement
- Expliciter son raisonnement

L'ordinateur ne peut pas deviner ce que nous voulons.

# Savoir écrire un algorithme

Ecrire un algorithme c'est :

- Savoir expliquer comment faire un travail sans la moindre ambiguïté
- Utiliser un langage simple
- Définir une suite d'actions à entreprendre en respectant une chronologie imposée

Un algorithme est universel :

- Il ne dépend pas du langage dans lequel il est implémenté
- Ni de la machine qui exécutera le programme correspondant

Un même algorithme doit :

- Pouvoir être implémenté en VBScript et exécuté sur Windows
- Pouvoir être implémenté en bash et exécuté sur un système UNIX/Linux
- Pouvoir être implémenté en JavaScript et exécuté dans un navigateur
- Pouvoir être implémenté en Objective-C et exécuté sur un MacOS

# Les langages de programmation

# Les langages de programmation

Un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent.

Un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire, de significations, mais aussi d'un environnement de traduction censé rendre sa syntaxe compréhensible par la machine.

# Les langages de programmation

Vidéo : évolution des langages de programmation 1965-2022

NEW! Most Popular Programming Languages 1965 - 2022 - YouTube

# Comment l'ordinateur comprend le code qu'on écrit

Le processeur d'un ordinateur se charge d'exécuter les instructions qui lui sont fournies.

Le processeur ne comprend que le langage machine aussi appelé code machine ou encore code natif.

Un programme en code natif est composé d'instructions directement reconnues par un processeur. Le code natif est donc lié à une famille particulière de processeurs partageant le même jeu d'instructions.

# Comment l'ordinateur comprend le code qu'on écrit

Les programmeurs n'écrivent pas de code natif directement, ils rédigent des « programmes sources » en suivant les conventions d'un langage de programmation, et la traduction de ces programmes sources en code natif est faite par des programmes (assembleur, compilateur, édition de liens, etc.).

# Comment l'ordinateur comprend le code qu'on écrit

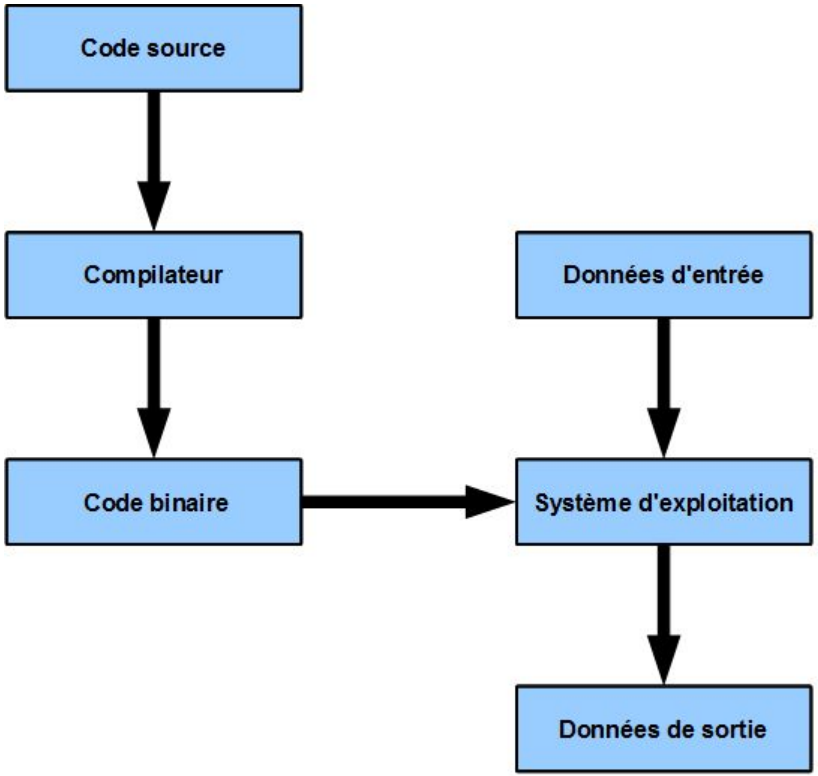
Pour plusieurs raisons comme la portabilité du code, la simplicité du langage, le domaine d'utilisation (...) tous les langages de programmation ne peuvent pas être compilés en code natif.

On distingue donc 3 principaux types de langages de programmation :

- Les langages compilés
- Les langages interprétés
- Langages compilés en bytecode (hybride)

# Comment l'ordinateur comprend le code qu'on écrit

*Les langages compilés*



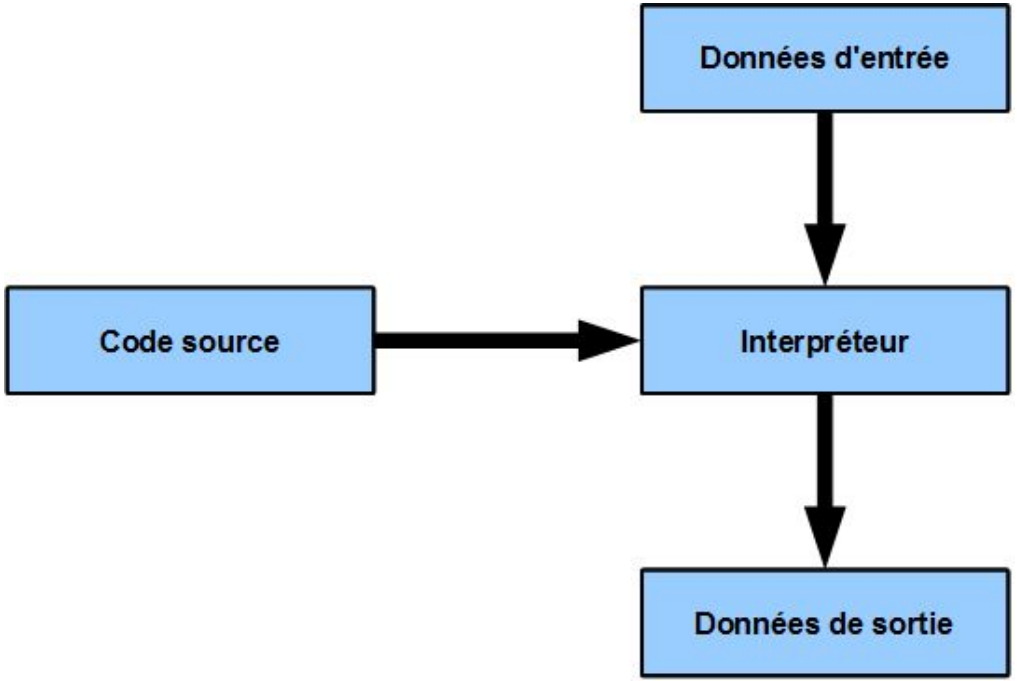
# Comment l'ordinateur comprend le code qu'on écrit

## Les 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 :

# Comment l'ordinateur comprend le code qu'on écrit

## Langages interprétés



# Comment l'ordinateur comprend le code qu'on écrit

## Langages interprétés

Dans ces langages, le code source (celui que vous écrivez) est interprété par un logiciel qu'on appelle interpréteur.

Le code source est lu ligne par ligne par l'interpréteur qui dit exactement au processeur ce qu'il doit faire.

# Comment l'ordinateur comprend le code qu'on écrit

Langages compilés vs langages interprétés

Dans un langage interprété, le code source est portable. Il suffit qu'il y ait un interpréteur installé sur la machine pour que le programme s'exécute.

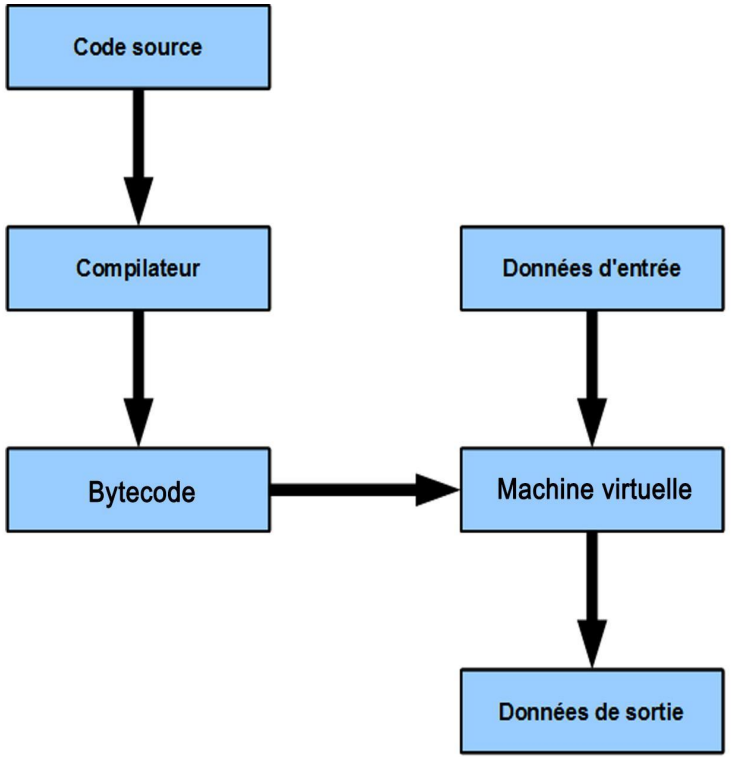
Avec un langage compilé, il faudra (en général) recompiler le code pour un système d'exploitation et un processeur donnés (.exe pour Windows, .dmg pour macOS)

Dans un langage compilé, le programme est directement compilé pour un processeur en particulier donc il sera en général beaucoup plus rapide que le même programme dans un langage interprété.

Il y a un tradeoff entre portabilité et performance.

# Comment l'ordinateur comprend le code qu'on écrit

## Langages compilés en bytecodes



# Comment l'ordinateur comprend le code qu'on écrit

Langages compilés en bytecodes

Pour ce type de langage, le code source est compilé en un code intermédiaire qui sera exécuté par une machine virtuelle. Au démarrage du programme, la machine virtuelle compile le bytecode en instructions natives pour le système d'exploitation et le processeur donnés. On appelle cela la compilation juste à temps (Just in Time Compilation)

Parce que le code source est compilé, il est beaucoup plus rapide à s'exécuter qu'un code source interprété mais avec une petite pénalité au démarrage du programme du fait de la compilation juste à temps.

Parce que le code source est compris par la machine virtuelle indépendamment de la plateforme, il peut être exécuté sur n'importe quel ordinateur sur laquelle la machine virtuelle est installée.

# Exemple catégorie de langages

Quelques langages compilés :

C, C++, Fortran, Go, Rust




# Exemple catégorie de langages

Quelques langages interprété :

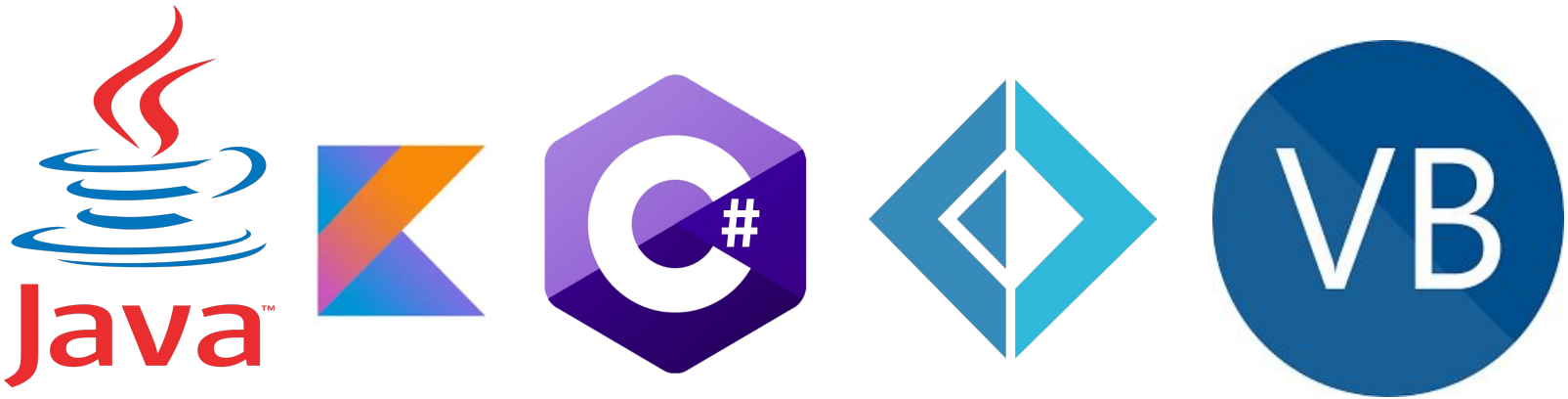
Python, R, JavaScript, Ruby, PHP, Perl




# Exemple catégorie de langages

Quelques langages compilés en bytecode :

Java, Kotlin, C#, F#, VB.NET,



# Exemples catégories de langages

Langage de programmation à usage général :

Ce sont des langages de programmation destinés à être utilisés pour tout domaine que touche la programmation.

Ils sont créés pour être versatiles. Ils possèdent des librairies pour faire certaines tâches spécifiques.

Exemple : Java, Python, C/C++, C#, F#, VB.NET, Rust, Go

Python possède un gros écosystème pour la Data Science comparable à R, Matlab bien que ce soit un langage à usage général.

# Exemples catégories de langages

Langage de programmation à usage dédié :

Un langage dédié  est un langage de programmation dont les spécifications sont conçues pour répondre aux contraintes d'un domaine d'application précis.

Une connaissance du domaine d'application est souvent requise pour utiliser les langages de programmation dédiés.

Exemple: R, SAS, Matlab, Erlang, Perl, PHP, JavaScript

Certains langages comme R, PHP et JavaScript deviennent de plus en plus universels grâce aux efforts de la communauté d'utilisateurs.

Avec R on peut aujourd'hui créer des applications web (R Shiny), des API (plumber) ...

# Langage de programmation et langage informatique

Il existe plusieurs langages informatiques.

Tout langage informatique n'est pas un langage de programmation.

Par exemple : SQL, XML, HTML, CSS, Latex sont des langages informatiques mais ne sont pas des langages de programmation