

# Introduction à la programmation en Python

## Les constructions élémentaires en Python

J. Boucher

Lycée Pierre-Paul RIQUET, Première NSI

21 août 2024

# Plan

## 1 COURS - I. L'environnement de développement

- 1. L'interprète Python
- 2. Mode interactif
- 3. Mode programme

## 2 COURS - II. Arithmétique, variables, instructions

- 1. Arithmétique
- 2. Variables
- 3. Entrées et sorties I/O
- 4. Les erreurs

# 1. L'interprète Python

## Fonctionnement d'un langage dit « interprété »

Le **langage de programmation Python** permet d'interagir avec la machine à l'aide d'un programme appelé **interprète Python**<sup>1</sup>.

On peut l'utiliser de deux manières différentes :

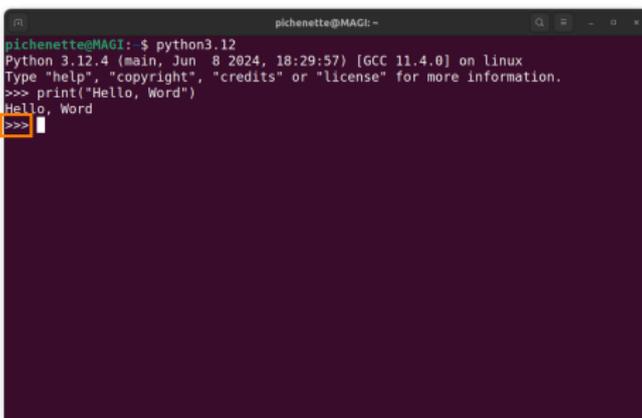
- En *mode interactif*; il permet de dialoguer avec l'interprète, instruction après instruction.
- En *mode programme*; une suite d'instructions est écrite dans un fichier, appelée programme ou code source, puis exécutée par l'interprète.

---

1. En informatique, un interprète, ou interpréteur, est un outil dont la tâche est d'analyser, de traduire et d'exécuter les programmes écrits dans un langage informatique.

## 2. Mode interactif

Le mode interactif de l'interprète Python se présente comme une calculatrice ; les trois chevrons `>>>` constituent l'invite de commande de Python, qui indique qu'il attend des instructions.



```
pichenette@MAGI: ~$ python3.12
Python 3.12.4 (main, Jun  8 2024, 18:29:57) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, Word")
Hello, Word
>>>
```

On lance l'interprète Python, ici `python3.12`, à partir d'un terminal<sup>2</sup>.

---

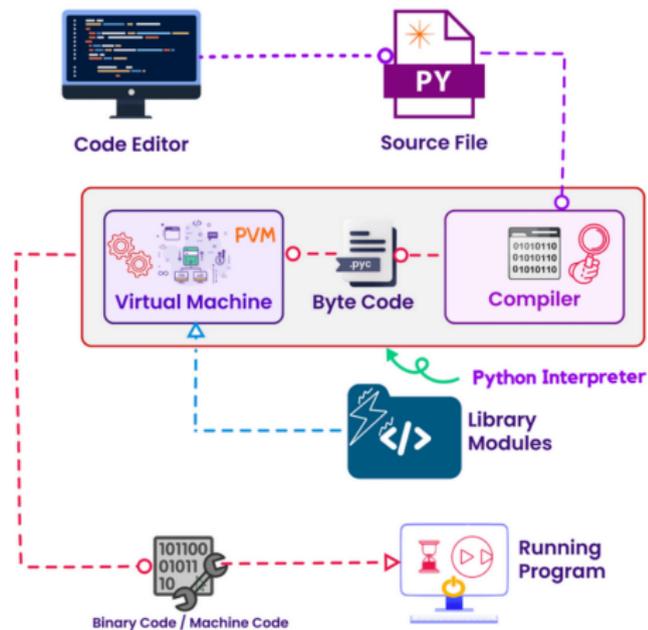
2. programme capturant toutes les entrées claviers et qui lui-même exécute un interpréteur de commandes interagissant avec le système d'exploitation

## 3. Mode programme

Un **environnement de développement**, ou IDE,<sup>3</sup> intègre toute la **chaîne d'outils** nécessaire pour programmer dans un langage donné.

Pour le langage Python, il sera recommandés d'utiliser, au choix :

-  IDLE
-  Visual Studio Code



Chaîne d'outils nécessaire à la génération et l'exécution d'un programme à partir d'un code écrit en Python.

### 3. Integrated Development Environment

# 1. Arithmétique

- En Python, on peut saisir des combinaisons arbitraires d'**opérations arithmétiques**, à l'aide des quatre opérations les plus communes :  $+$  ;  $-$  ;  $*$  ;  $/$  et de l'opération « élever à la puissance »  $**$ .

- Les règles de **priorité opératoire** sont les mêmes qu'en mathématiques :

$$() > ** > * \text{ ou } / > + \text{ ou } -$$

- De nombreux programmes manipulent des entiers naturels et nécessitent l'utilisation de la division euclidienne avec :
  - L'opérateur « *quotient dans division entière par* »  $//$  ;
  - L'opérateur « *reste dans la division entière par* » ou « *modulo* »  $\%$  .

**Attention :** les opérateurs  $//$  et  $\%$  de Python ne coïncident avec la division euclidienne que lorsque le diviseur est positif.

## 2. Variables

### Définition

Une **variable** peut se représenter comme une case mémoire de la machine exécutante. Elle se caractérise par :

- un **nom** (ou identificateur) ;
- une **valeur** ;
- un **type** ;
- une **adresse**.

Affection de la **valeur 1**  
à la variable **nommée x**

```
x = 1.0
```

**Type** de x

```
>>>type(x)  
<class 'float'>
```

État de l'interprète

```
x 1.0
```

**Adresse** de x

```
>>>id(x)  
139559917617488
```

## Instruction d'affectation

Une instruction d'affectation permet d'associer une **valeur** à un **nom** de variable.

$$x = 1.0 \quad \{ x \ 1.0 \} \quad (1)$$

$$x = x + 0.5 \quad \{ x \ 1.5 \} \quad (2)$$

$$y = x \quad \{ x \ 1.5 ; y \ 1.5 \} \quad (3)$$

$$y += 1.0 \quad \{ x \ 1.5 ; y \ 2.5 \} \quad (4)$$

### Remarques :

- 1 Une instruction d'affectation se lit toujours de droite à gauche :

$$x = x + 0.5$$

←

l'interprète commence par évaluer l'expression de droite  $x + 0.5$  à  $1.5$ , puis affecte cette valeur à la variable dont le nom  $x$  est donné à gauche du symbole  $=$ .

L'instruction  $x + 0.5 = x$  donne l'erreur

```
SyntaxError: cannot assign to expression here.
```

puisque  $x + 0.5$  ne fait pas référence à un nom de variable mais à une expression.

## Instruction d'affectation

Une instruction d'affectation permet d'associer une **valeur** à un **nom** de variable.

$$x = 1.0 \qquad \{ x \ 1.0 \} \qquad (1)$$

$$x = x + 0.5 \qquad \{ x \ 1.5 \} \qquad (2)$$

$$y = x \qquad \{ x \ 1.5 ; y \ 1.5 \} \qquad (3)$$

$$y += 1.0 \qquad \{ x \ 1.5 ; y \ 2.5 \} \qquad (4)$$

### Remarques :

- 2 Au cours de l'exécution de l'instruction (2) la valeur de  $x$  passe de 1.0 à 1.5.

$$\{ x \ 1.0 \} \rightarrow \{ x \ 1.5 \}$$

Cela illustre le *dynamisme* des variables informatiques, contrairement aux variables mathématiques dont la valeur ne change jamais.

L'interprétation mathématique de cette instruction est elle aussi très différente, et correspondrait à une égalité toujours fausse.

## Instruction d'affectation

Une instruction d'affectation permet d'associer une **valeur** à un **nom** de variable.

$$x = 1.0 \quad \{ x \ 1.0 \} \quad (1)$$

$$x = x + 0.5 \quad \{ x \ 1.5 \} \quad (2)$$

$$y = x \quad \{ x \ 1.5 ; y \ 1.5 \} \quad (3)$$

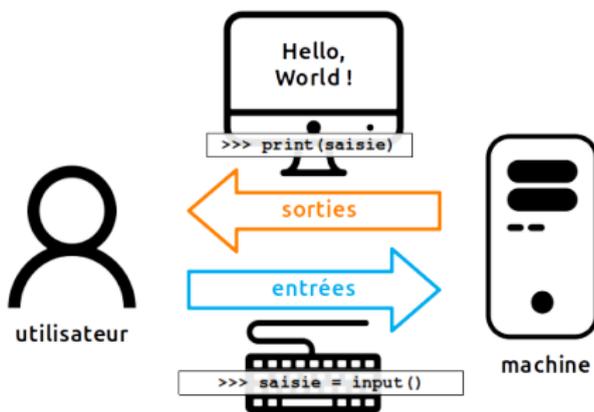
$$y += 1.0 \quad \{ x \ 1.5 ; y \ 2.5 \} \quad (4)$$

### Remarques :

- Les instructions (3) et (4) montre l'indépendance des variables  $x$  et  $y$  ;
  - L'instruction (3) permet de définir une nouvelle variable  $y$ , en l'initialisant avec la valeur associée à  $x$ .
  - L'instruction (4) permet de modifier la valeur de  $y$  en lui ajoutant 1.0 par l'utilisation de l'opérateur  $+=$ . C'est une « contraction » de l'instruction  $y = y + 1.0$ .
  - En observant l'état final de l'interprète, on note que seule la valeur de  $y$  est modifiée.

### 3. Entrées et sorties I/O

Les **instructions d'entrées/sorties** permettent l'interaction de l'utilisateur avec la machine.



- L'instruction `print` est une **instruction de sortie** permettant de faire afficher par la machine des messages à l'utilisateur du programme, au cours de son exécution ; le développeur peut donc par ce biais *instrumenter* son code source.
- L'instruction `input` est une **instruction d'entrée** permettant de capturer les caractères saisis au clavier par l'utilisateur du programme en cours d'exécution.

## 4. Les erreurs

(Informatique) Bogue [n. m.] : Francisation de l'anglais bug désignant un défaut dans la conception d'un programme, se manifestant par des anomalies de fonctionnement.

### Grands types d'erreur en programmation

- **erreurs de syntaxe** : le code ne respecte pas les règles « grammaticales » du langage. L'interprète interrompt l'exécution du programme et affiche un message d'erreur.
- **erreurs d'exécution** : erreur de programmation provoquant le blocage ou l'arrêt du programme en cours d'exécution (affectation de variable incorrecte, division par zéro, lecture d'un fichier inexistant...).
- **erreurs sémantiques** : le code est syntaxiquement correct et le programme s'exécute sans aucune erreur apparente (du point de vue de la machine). Néanmoins le programme ne réalise pas le résultat attendu.



- ```
>>> 1 = 1
SyntaxError
```
- ```
>>> a = 0 ; 1 / a
ZeroDivisionError
```
- ```
>>> a = a + 2 #on double la valeur de a
Aucune erreur pour la machine...
```