

# Des bases à l'introduction du langage Python

Yao Aristide

# Table des matières



<b>I - Objectifs</b>	<b>3</b>
<b>II - Introduction</b>	<b>4</b>
<b>III - Notions de base</b>	<b>5</b>
1. Langage de programmation : relation programmeur-machine .....	5
2. Nature d'un langage de programmation .....	5
3. Schéma de communication programmeur-machine .....	5
4. Programme de traduction .....	6
5. Compilateur vs Interpréteur : avantages et inconvénients .....	6
<b>IV - II. Introduction à Python</b>	<b>7</b>
1. Caractéristique du langage Python .....	7
2. Gestion du flux d'entrée et de sortie .....	7
3. Fonction de manipulation de sortie .....	8
4. Fonction de manipulation d'entrée .....	8
5. Premier pas avec la console Python (Vidéo) .....	8
<b>V - Exercice : Agencez dans l'ordre l'exécution d'un programme par un compilateur</b>	<b>9</b>
<b>VI - Exercice</b>	<b>10</b>
<b>VII - Exercice</b>	<b>11</b>
<b>VIII - Exercice</b>	<b>12</b>
<b>IX - Exercice</b>	<b>13</b>
<b>X - Exercice</b>	<b>14</b>
<b>XI - Exercice</b>	<b>15</b>
<b>XII - Conclusion</b>	<b>16</b>



# *Objectifs*

A la fin de cette leçon vous serez capable de :

- Définir un langage de programmation
- Décrire la communication homme-machine dans le cadre de l'activité de programmation
- Identifier les caractéristiques du langage Python
- Analyser et tester du code Python de base

# Introduction



L'activité essentielle d'un programmeur/administrateur consiste à proposer des solutions afin de résoudre des problèmes liés à une *gestion numérique* de différent type de données ou de matériel. Ce cours introduit d'abord les notions élémentaires que doit connaître un programmeur sans être exhaustif, ensuite l'architecture de fonctionnement de *python* et enfin la manipulation de l'interpréteur interactif.

# Notions de base



## Objectifs

- Définir les notions de base de la programmation
- Décrire la relation entre le programmeur et la machine
- Décrire les processus d'exécution d'un programme

## 1. Langage de programmation : relation programmeur-machine

L'activité de la *programmation* consiste à instruire une machine (ordinateur, smartphone, etc.) de manière à ce qu'elle reproduise exactement le fruit d'un raisonnement en sachant qu'elle ne possède aucune connaissance à priori sur ce raisonnement.

Ainsi un *langage de programmation* est un ensemble de mots-clés associé à un ensemble de règles très précises indiquant comment on peut assembler ces mots pour former des « phrases » que l'ordinateur puisse comprendre.

## 2. Nature d'un langage de programmation

Un *langage de programmation* peut être de *bas niveau* ou de *haut niveau*. Il est de *bas niveau* s'il est constitué d'instructions très élémentaires, très proche du langage de la machine.

Il est de *haut niveau* s'il contient des instructions d'un niveau élevé pour la machine. Ces instructions, facilement assimilables par le programmeur sont traduites par un *programme de traduction* en un grand nombre d'instructions machine élémentaires.

### Exemple

Langage *bas niveau*: Assembleur

Langage *haut niveau*: Langage C, Java, etc.

## 3. Schéma de communication programmeur-machine

Cette communication se déroule en trois étapes successives :

1. Le programmeur adresse à la machine un ensemble d'instructions grâce à un *langage de programmation*. Cet ensemble d'instructions est appelé *code source*<sup>1</sup>.
2. La machine fait appel à un *programme de traduction* qui se charge de traduire le *code source* en un langage qui est compréhensible par la machine (*langage machine*).
3. La machine (le processeur de la machine) *exécute* les instructions exprimées en *langage machine* afin d'afficher à l'écran les résultats à la demande.

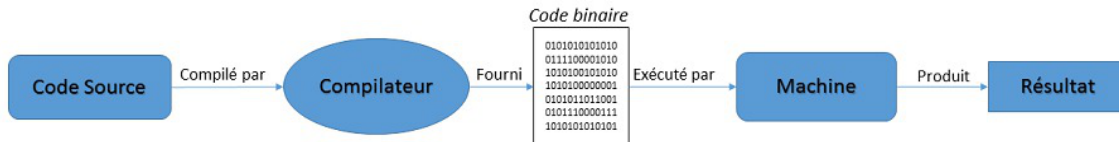
### Complément

<sup>1</sup>un *code source* est un programme qu'on écrit dans un langage de programmation quelconque sous forme de texte simple. Pour rédiger ce texte, on peut faire appel à toutes sortes de logiciels plus ou moins perfectionnés, à la condition qu'ils ne produisent que du texte brut, c'est-à-dire sans mise en page particulière ni aucun attribut de style (pas de spécification de police, donc, pas de gros titres, pas de gras, ni de souligne, ni d'italique, etc.)

## 4. Programme de traduction

Le *programme de traduction de code source* est soit un *interpréteur* ou un *compilateur* suivant la méthode de traduction utilisée . Il est :

- Un *compilateur* si le programme transforme le *code source* en une série d'instructions du *langage machine* via une *étape de compilation* ; la *compilation* se fait à la fin de l'écriture du *code source* en une fois et produit un *code binaire* qui est ensuite exécuté. Lors d'une *modification du code source*, il est nécessaire de *repasser par l'étape de la compilation* avant de constater le résultat de la modification. Le schéma suivant résume une *traduction de code source par un compilateur*



*Étapes de compilation d'un programme informatique*

- Un *interpréteur*, dans ce cas il n'existe pas d'*étape de compilation*, le programme traduit au fur et à mesure de la saisie, chaque ligne du code source en *instructions du langage machine*. Ces instructions en *langage machine* sont directement exécutées. Aucun code binaire n'est généré et le résultat d'une modification est constaté en relançant l'exécution du programme.

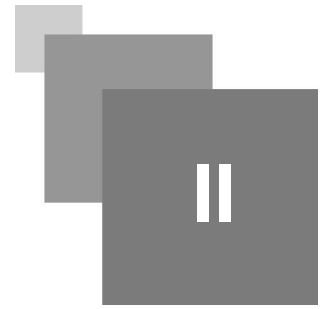
## 5. Compilateur vs Interpréteur : avantages et inconvénients

Le tableau suivant fait un état des différences entre les *deux natures de programme de traduction* sans être exhaustif

	Compilateur	Interpréteur
Avantages	<ul style="list-style-type: none"> <li>• L'exécution se déroule plus vite qu'un programme interprété.</li> <li>• Adapté pour les grands projets</li> </ul>	<ul style="list-style-type: none"> <li>• Possible de tester immédiatement toute modification apportée au code source sans passer par une phase de compilation.</li> <li>• Idéal pour une phase d'apprentissage d'un langage par un débutant</li> </ul>
Inconvénients	<ul style="list-style-type: none"> <li>• Nécessite de passer par la phase de compilation à chaque modification du code source.</li> <li>• Pas adapté pour des projets qui nécessitent des tests récurrents</li> </ul>	<ul style="list-style-type: none"> <li>• L'exécution fonctionne moins vite qu'un programme compilé.</li> <li>• Pas adapté pour les grands projets.</li> <li>• (Re)traduit le code source en langage machine à chaque exécution</li> </ul>

*comparaison entre langage compilé et langage interprété*

# II. Introduction à Python



## Objectifs

- Définir certaines caractéristiques du langage Python
- Comprendre les concepts de gestion de flux d'entrée et de sortie
- Analyser, tester des instructions python
- Prédire le résultat d'exécution de code python

## 1. Caractéristique du langage Python

*Qu'est ce que Python ?*

Python est un *langage de programmation haut niveau* qui est utilisé dans les domaines de la programmation web, de l'intelligence artificielle et du calcul scientifique. Il est parfaitement adapté à l'apprentissage de la programmation par un débutant car il présente les caractéristiques suivantes :

- Son téléchargement et son installation sont gratuits (i.e. il est libre de droit).
- Il est facile à comprendre, à écrire et à lire.
- Il utilise un mode de traduction interprété. Cependant il est adapté pour les grands projets
- Un programme Python est en moyenne au moins trois fois plus court qu'un programme écrit en langage C ou en Java.
- Il est doté d'une *communauté active*, ce qui permet au programmeur débutant de communiquer avec une palette de programmeurs expérimentés afin de l'aider à résoudre des problèmes qu'il pourrait rencontrer lors de son exercice.

Cette liste de caractéristique n'est pas exhaustive. Python possède encore de nombreuses caractéristiques qui font de lui un langage très populaire et très utilisé dans le monde de la programmation.

## 2. Gestion du flux d'entrée et de sortie

*Entrée et Sortie*

Lors de l'exécution d'un *programme informatique interactif*, les *entrées* sont les informations qui sont saisies au clavier par l'utilisateur du programme et les *sorties* sont les informations affichées à l'écran par le programme.


*Données d'entrée et Données de sortie*

Un programme informatique reçoit généralement des *données en entrée* et produit des *données en sortie*. Les *données en entrée* sont fournies, soit par le programmeur (*programme statique*), soit par l'utilisateur du programme (*programme interactif*) ; les *données en sortie* sont calculées par le programme en fonction de l'algorithme appliqué sur les *données d'entrées*.

### 3. Fonction de manipulation de sortie

*print*

En python, la fonction *print* est utilisée pour produire une *sortie*. Elle affiche une représentation textuelle de quelque chose à l'écran de la machine. Elle peut afficher le résultat d'une opération arithmétique ou encore une description textuelle. Pour afficher une description textuelle, le contenu du texte doit être encadré soit par le symbole apostrophe ('contenu') ou par le symbole double côte ("contenu").

 *Exemple : Afficher des sorties avec la fonction print*

---

```
>>>print(1+1)
2
>>>print("1+1")
1+1
>>>print("Nous sommes de la promotion DAS 2019 !")
Nous sommes de la promotion DAS 2019 !
```

 *Remarque*

---

lorsque le résultat de l'affichage est imprimé à l'écran, les doubles côtes ou les apostrophes n'apparaissent pas.

### 4. Fonction de manipulation d'entrée

*input*

Pour obtenir de l'utilisateur une donnée d'entrée, Python nous permet d'utiliser la fonction *input*. Elle demande à l'utilisateur de saisir une valeur au clavier à travers une description textuelle qui sera affichée à l'écran et qui est un paramètre de la fonction *input*. Elle retourne comme *données de sortie* la valeur saisie au clavier par l'utilisateur

 *Exemple : interagir avec l'utilisateur*

---

```
>>>input("donnez votre age : ")
donnez votre age : 20 -> valeur saisie par l'utilisateur
20 -> valeur retournée par la fonction input
>>>input('Entrez votre numéro de téléphone: ')
Entrez votre numéro de téléphone: 02229831
02229831
```

### 5. Premier pas avec la console Python (Vidéo)

*Vidéo*

<https://youtu.be/cpiO3xfAboI>, cette vidéo montre le fonctionnement des fonctions *input* et *print*, et initie l'apprenant à la manipulation de l'interpréteur interactif de Python



# Exercice : Agencez dans l'ordre l'exécution d'un programme par un compilateur



1. Exécution
2. compilateur
3. Code objet
4. Code source
5. Résultat

Réponse : \_ \_ \_ \_ \_

# Exercice

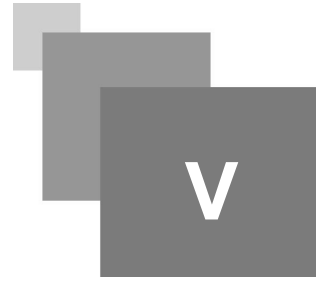


IV

*Un langage bas niveau est:*

- ☐ un langage qui est compréhensible par un non informaticien
- ☐ un langage dont les mots clés sont proches de la compréhension de l'ordinateur
- ☐ un langage machine

# Exercice



*Python est un langage de programmation qui utilise un mode de traduction*

- ☐ compilé
- ☐ interprété
- ☐ compilé et interprété
- ☐ binaire

# Exercice



VI

*Quel est le résultat de cette instruction ?*

```
>>>print("Je suis un bon etudiant")
```

- ☐ 'Je suis un bon etudiant'
- ☐ Je suis un bon etudiant
- ☐ "Je suis un bon etudiant"

# Exercice

VII

*Quel est le résultat de cette instruction?*

```
>>>print('input("donne ton age")')
```

- ☐ input("donne ton age")
- ☐ cette instruction est fausse
- ☐ 'input("donne ton age")'
- ☐ donne ton age

# Exercice

VIII

*Compléter l'instruction suivante pour que le résultat affiché à l'écran soit :*

*'Je suis UVCI'*

```
>>>print(" ")
```

# Exercice

IX

*Donnez le résultat de l'exécution du script suivant*

```
>>>print('5*3')
```



# Conclusion



Ce cours d'introduction à la programmation en général et à l'usage du langage python en particulier a permis de comprendre des notions fondamentales du domaine de développement d'application, de connaître l'architecture de fonctionnement de Python et de faire nos premiers pas avec l'interpréteur interactif de Python. La prochaine leçon sera consacrée à la manipulation des variables en Python.

