# Chapitre – Algorithmes et Python



**Python - Les principales instructions en Python**

La présente leçon vous introduit aux premières notions nécessaires pour programmer en Python. Python est un des langages informatiques qui permettent de traduire des *algorithmes* pour qu'ils soient exécutables par des machines.

| |
|---|
| On rappelle qu'un algorithme est une séquence ordonnée d'instructions exécutées de façon logique mais non intelligente. |

**--> Activité 01-01 Rappel de l'agorithmie et scratch + installation des logiciels de base.**

## 1) Particularités du langage Python

Point de repère - Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles.

Les principales caractéristiques de Python:
- Python est *portable* (cela signifie qu'il peut être adapté plus ou moins facilement en vue de fonctionner dans différents environnements d'exécution), non seulement sur les différentes variantes d'UNiX, mais aussi sur les OS propriétaires: MacOS, MS-DOS et les différentes variantes de Windows, par exemple.
- Python est *gratuit*, mais on peut l'utiliser sans restriction dans des projets commerciaux.
- Python convient aussi bien à des scripts d'une dizaine de lignes qu'à des projets complexes de plusieurs dizaines de milliers de lignes.
- La syntaxe de Python est très *simple* et permet d'écrire des programmes à la fois très compacts et très lisibles.
- La librairie standard de Python, et les paquetages contribués, donnent accès à une grande *variété* de services.
- Python est un langage qui continue à *évoluer*, soutenu par une communauté d'utilisateurs.

## 2) Variables : types de variables, création et affectation, opérations sur les variables

| Type | Pour qui ? | Exemples |
|---|---|---|
| **int** | Les entiers | ..., –5, –1, 0, 2, 6, ... |
| **float** | Les réels | –5.28, 13.0, 18.7 |
| **str** | Les chaînes de caractères (texte) | 'Janvier', '28b6', '13' |
| **bool** | Les booléens : uniquement deux valeurs possibles | True (Vrai) ou False (Faux) |
| **list** | C'est une liste ou tableau dans lequel on peut stocker plusieurs valeurs idéalement de même type | [0, 1, 2, 4], ['Janvier', 'Février', 'Mars'] |

La fonction `type(nom_variable)` permet de connaître le type de la variable.
La création d'une variable peut se faire à n'importe quel moment dans le programme mais annoncer les variables manipulées par le programme en début de code est une bonne pratique à favoriser. Pour créer une variable en python, il suffit de lui trouver un nom. Le nom de la variable doit être explicite et on utilisera pour le faire les caractères alphanumériques sans accent et le tiret de soulignement (underscore).
Ex.: `nom_variable`, `age_joueur` etc.

L'affectation est l'opération qui permet de définir la valeur ou le contenu d'une variable. En python l'affectation est définie avec le signe =
Ex. `longueur = 7.5` signifie que l'on affecte la valeur 7.5 à la variable appelée longueur
`Prenom_joueur = "Noémie"` signifie ……..

Les variables peuvent ensuite être manipulées, on peut réaliser dessus des opérations. Voici la liste des opérations les plus fréquentes que nous allons utiliser.
Avec des nombres, les opérations arithmétiques:

| | |
|---|---|
| *x + y* | *Addition* |
| *x – y* | *Soustraction* |
| *x * y* | *Multiplication* |
| *x / y* | *Division décimale de x par y* |
| *x // y* | *Quotient de la division euclidienne de x par y* |
| *x % y* | *Reste de la division euclidienne de x par y (Modulo)* |
| *x ** y* | *Puissance : $x^y$* |

Les opérations sur les chaînes de caractères (les chaînes de caractères sont des suites de signes, comme un mot qui est une suite de lettres):

| | |
|---|---|
| *ch1 + ch2* | *Concaténation (les chaînes ch1 et ch2 sont mises bout à bout)* |
| *len(ch1)* | *Nombre de caractères de la chaîne ch1* |
| *ch1[i]* | *$(i+1)^{ème}$ caractère de la chaîne ch1* |

exemple:
ch1='trou'  #attribue une chaîne trou à la variable ch1
ch2='duc' #attribue une chaîne duc à la variable ch1
print(ch1+ch2) # imprime la concatènation les deux chaines
CH=ch1+ch2 #attibue à la viriable CH la concatènation les deux chaines
print(len(CH))#imprime la longueur de la chaine CH
a=CH[3]#attibue à la variable a le 4 caractère de la chaine CH
print(a) #imprime la variable a

**IMPORTANT: L'insertion de commentaires : une nécessité**
En python (mais dans d'autres langages également), et dès que vous écrivez un programme un tout petit peu complexe, il faut commenter ce que vous faites.

Ex. ce petit programme très simple

```
age = int(56)        #attribue une valeur d'entier  à la variable âge

print('Vous avez', age, 'ans')   #affiche une phrase avec insertion de la variable
```

--> Annote chaque ligne:
ch1='trou'  #attribue une chaîne trou à la variable ch1
ch2='duc' #attribue une chaîne duc à la variable ch1
print(ch1+ch2) # imprime la concatènation les deux chaines
CH=ch1+ch2 #attibue à la viriable CH la concatènation les deux chaines
print(len(CH))#imprime la longueur de la chaine CH
a=CH[3]#attibue à la variable a le 4° caractère de la chaine CH
print(a) #imprime la variable a

### 3) Interaction avec l'utilisateur (entrées/sorties)

Comme on vient de le voir, la plupart du temps, dans les programmes, on demandera à l'utilisateur d'entrer des valeurs. Pour ce faire, en python, on utilise l'instruction `input()`.
Dans les parenthèses, on peut mettre, entre guillements, une phrase explicitant ce que l'on attend de l'ultilisateur.
Ex.

```
Prénom = input("Quel est votre prénom? ")
```

Cette instruction prendra les signes entrés par l'usager comme une chaîne de caractères. Si c'est une valeur numérique que vous attendez, il faudra combiner cette instruction `input()` à d'autres instructions. Voici ce que nous allons rencontrer le plus fréquemment :

| | |
|---|---|
| `input()` | Par défaut, ce qui est saisi par l'utilisateur est une chaîne de caractères. |
| `int(input())` | Permet de recueillir un entier entré par l'utilisateur (c'est utile pour l'âge, pour le nombre de personnes présentes ou absentes, etc.) |
| `float(input())` | Permet de recueillir un réel entré par l'utilisateur (c'est utile pour la température, , etc.) |

```
EXO:
Afficher la table de multiplication de x après demande à l'utilisateur.
```

### 4) Instructions conditionnelles : IF et ELSE, opérateurs logiques OR, AND, NOT, cas multiples: ELIF

En algorithmique, on peut vouloir effectuer des instructions selon qu'une condition soit satisfaite ou non. On parle alors d'instruction conditionnelle. En français, on peut résumer cette situation de la façon suivante "Si la condition est satisfaite, Alors on réalise une action, Sinon on réalise une autre action"
Des opérations que l'on dit booléennes (des tests qui peuvent se faire sur n'importe quel type de variable):

| | |
|---|---|
| $x < y$ | *Tester si x est inférieur à y* |
| $x <= y$ | *Tester si x est inférieur ou égal à y* |
| $x > y$ | *Tester si x est supérieur à y* |
| $x >= y$ | *Tester si x est supérieur ou égal à y* |
| $x == y$ | *Tester si x est égal de y* |
| $x != y$ | *Tester si x est différent de y* |

Attention à ne pas confondre en python le signe = qui est une affectation et le signe == qui est un test de vérification d'identité.

En Python, les **instructions conditionnelles** sont introduites par if (en minuscules) et il faut impérativement mettre **deux points** à la fin de la ligne exprimant la condition. Les instructions à effectuer à condition que

sont **indentées**(à la ligne mais avec un retrait... toujours le même!!!). Reprenons un exemple déjà mobilisé (on met en gras et en bleu les éléments de syntaxe conditionnelle)

```
age = int(input("Entrez votre âge : "))
if age<18:
   print("Vous n'êtes pas majeur!") #remarquez le décalage de la ligne
else:
   print ("vous avez l'âge requis")
```

Les conditions peuvent se combiner entre elles

| | |
|---|---|
| x and y | Intersection |
| x or y | Union |
| not y | Négation |

EXO:
Afficher la table de multiplication de x après demande à l'utilisateur. Et après lui avoir demander un mot de passe(qu'il connait!!!)

### 5) Boucles : boucles bornées, boucles non bornées

Pour exécuter plusieur fois un bloc d'instructions ((souvenez-vous du dessin d'un carré par exemple), on utilise ce que l'on appelle une boucle.
Il existe des boucles dont on connaît le nombre de répétitions, les boucles bornées, on utilise alors l'instruction `for` et la fonction `range()`. Cette fonction admet 3 paramètres (le début, la fin, le pas). Par défaut et si vous ne précisez rien, le début est 0, il est impératif de saisir un paramètre de fin et le pas est par défaut 1, autrement dit la fonction va par défaut aller de un en un.
Ex.

| | |
|---|---|
| `for i in range (5):`<br>`  print(i)` | Voici un programme qui va vous afficher 0 puis 1 puis 2 puis 3 puis 4 (en commençant par défaut à 0 et tant que i<5) |
| `for i in range (0,10,2):`<br>`  print(i)` | Voici un programme qui va vous afficher 0 puis 2 (mais pas 1 parce que on a demandé de sauter une valeur sur 2) puis 2 puis 4 puis 6 puis 8 (et s'arrête juste avant que i atteigne la valeur 10) |

Mais il existe aussi des boucles dont on ne connaît pas le nombre de répétitions, on les introduit par l'instruction `while` suivie d'un test (la boucle sera répétée tant que le test est vrai)
Une boucle while exécute des instructions à plusieurs reprises tant qu'une condition reste vraie. La syntaxe de la boucle while est:

```
while condition-de-continuation-de-la-boucle:
      Instruction(s)  # corps de la boucle
```

Une seule exécution du corps de la boucle est appelée une itération (ou répétition) de la boucle.

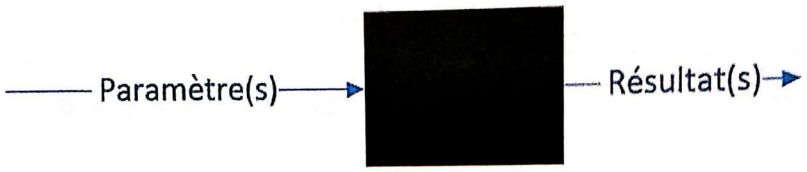
Exercice: voici un programme, quel sera le résultat?

```
somme = 0
i = 1
while i < 10:
   somme = somme + i
   i = i + 1
print("la somme est ", somme)
```

EXO:
Afficher la table de multiplication de x après demande à l'utilisateur. Et après lui avoir demander un mot de passe(qu'il connait!!!)
Le programme doit comporter une boucle de votre choix afin de réduire sa taille...

### 6) Fonctions

Au sens algorithmique, une fonction est une sorte de boîte noire qui admet en entrée des valeurs que l'on appelle des paramètres et qui retourne un résultat stockable dans une variable.




Pour créer une fonction:

```
def NomDeLaFonction(variable1, variable2...):
    A=varaible1*variable2
    return A
```

pour l'utiliser:

**NouvelleVariable=NomDeLaFonction(4, 78)** #permet d'affect à la variable NouvelleVariable une nouvelle valeur de 312...

Nous développerons **plus tard comment créer des fonctions dans Python** même si nous en avons déjà manipulé (la fonction `range()`, la fonction `print()` par exemple)

**Activité 01-02 annotation des instructions pythons, analyses de programmations.**

## 7) Bibliothèques et modules

Une bibliothèque ou un module est un paquet de fonctions regroupées parce qu'elles se rapportent à une thématique commune. Il existe ainsi dans Python des modules de fonctions créés par des utilisateurs. Pour accéder à un module, on utilise le mot-clé `import` suivi du nom du module que l'on veut utiliser, nous reverrons cela en détail.
Voici cependant le nom de modules que nous mobiliserons dès cette année :
`tkinter` : afficha ge graphique (nous utiliserons aussi `turtle`)
`math` : calcul mathématique
`random` : traitement aléatoire
`time` : gestion du temps
`pillow` : traitement d'images numériques
folium: création de document cartographique