Premiers pas en Python

1) Les types de variables en Python

Un programme manipule des **données** (nombre, texte …). Pour pouvoir manipuler ces données il faut garder leurs **valeurs** en mémoire. C'est le rôle des **variables**.

Une variable est un nom désignant une donnée. Une variable possède un type, ce qui permet à l'ordinateur de savoir quelles valeurs elle peut prendre et quelles opérations on peut effectuer avec.

Les principaux types de variables sont :

Types	Valeurs
`int`	nombres entiers relatifs
`float`	nombres décimaux
`complex`	nombres complexes
`bool`	booléens : True ou False
`str`	chaînes de caractères

Une variable est définie par son **nom**, son **type** et sa **valeur**.

La convention la plus fréquemment utilisée pour nommer les variables est la norme **lowerCamelCase**. Elle repose sur deux grands principes :

- Tout nom de variable commence par une lettre minuscule.
- Si le nom d'une variable se compose de plusieurs mots, la première lettre de chaque mot (sauf le premier) s'écrit en majuscule.

Exemple : `adresseClient` est un nom de variable qui respecte la norme lowerCamelCase.

2) Affectation de variables

syntaxe d'une affectation simple :

```
maVariable = valeur
```

syntaxe d'une affectation multiple :

```
variable1, variable2, variable3 = valeur1, valeur2, valeur3
```

En Python, le type de la variable est défini lors de la première affectation (appelée aussi initialisation).

On peut vérifier le type d'une variable avec l'instruction **type**.

```
type(maVariable)
```

```
>>> i=4
>>> type(i)
<class 'int'>
>>>
```

```
>>> nombre=4.5
>>> type(nombre)
<class 'float'>
>>>
```

```
>>> titre='Get Back'
>>> type(titre)
<class 'str'>
>>>
```

3) Conversions de types

Un programme peut forcer la conversion d'un type à un autre :

Opérations	Résultats
`int(x)`	x est un décimal ou une chaîne de caractères.
`float(x)`	x est un entier ou une chaîne de caractères.
`bool(x)`	x est un nombre ou un booléen. Le résultat vaut False uniquement si x vaut 0 ou ''
`str(x)`	x est un nombre ou un booléen

```
>>> p=3.14
>>> e=int(p)
>>> e
3
>>> c=str(e)
>>> c
'3'
>>> f=float(e)
>>> f
3.0
>>>
```

4) Affichage et saisie

En mode console on peut afficher le contenu d'une variable en tapant son nom (exemple précédent).

Pour afficher à partir d'un script on utilise l'instruction **print**.

Syntaxe de l'instruction print :

```
print(expression1,expression2)
```

expression1 et expression2 sont des données ou des variables, qui seront séparés par un espace à l'affichage.

```
>>> expression1='hello'
>>> print(expression1,'world')
hello world
>>>
```

On peut ajouter des paramètres optionnels **sep** et **end** qui imposent le caractère de séparation (par défaut il s'agit d'un espace) et le caractère de fin de ligne (par défaut il s'agit d'un retour à la ligne) :

```
print(expression1,expression2,sep='***',end='...')
```

```
>>> print ('hello','world',sep='***',end='...')
hello***world...
```

Pour faire saisir une donnée à l'utilisateur on utilise l'instruction **input**.

```
var = input(expression)
```

```
>>> nombre=input('choisir un nombre compris entre 1 et 20')
>>> print(nombre)
10
>>>
```

5) Opérations sur les variables

a) Opérations arithmétiques

Principales opérations arithmétiques en Python :

Opérations	Résultats
`x+y`	Somme de x et y
`x-y`	Différence de x et y
`x*y`	Produit de x et y
`x**y`	Élévation de x à la puissance y
`x/y`	Quotient réel de x par y
`x//y`	Quotient entier de x par y
`x%y`	Reste du quotient entier de x par y

L'affectation qui réalise le calcul et modifie l'opérande de gauche peut s'écrire de différentes façons :

Écriture 1	Écriture 2
`x+=y`	`x=x+y`
`x-=y`	`x=x-y`
`x*=y`	`x=x*y`
`x**=y`	`x=x**y`
`x/=y`	`x=x/y`
`x//=y`	`x=x//y`
`x%=y`	`x=x%y`

b) Opérations sur les chaînes de caractères

Une chaîne de caractères est une suite de caractères. On a accès à un caractère via sa position dans la chaîne (cette position est appelé l'indice). Comme dans la plupart des langages de programmations ces indices commencent à 0 (le premier caractère a pour indice 0). En Python, on peut accéder aux caractères avec des indices négatifs, en partant de la fin de la chaîne.

```
>>> chaine='chaine de caracteres'
>>> print(chaine)
chaine de caracteres
>>> print(chaine[0])
c
>>> print(chaine[2])
a
>>> print(chaine[-1])
s
>>>
```

On peut écrire une chaîne de caractères de différentes façons :

- Entre guillemets `" ceci est une chaine de caracteres "`
- Entre apostrophes `'ceci est une chaine de caracteres'`
- Entre triples guillemets `'''ceci est une chaine de caracteres'''`

Si un apostrophe doit se trouver dans la chaîne il faut soit écrire la chaîne entre guillemets soit insérer le caractère anti-slash « \ » avant l'apostrophe du message :

```
>>> print('c\'est parfait')
c'est parfait
>>>
```

```
>>> print ("c'est parfait")
c'est parfait
>>>
```

Principales opérations associées aux chaînes de caractères :

Opération	Résultat
`chaine1 + chaine2`	Concaténation de **chaine1** et **chaine2**
`chaine * n ou n * chaine`	Concaténation de **n** copies de **chaine**
`len(chaine)`	Nombre de caractères de **chaine**
`chaine.count(x)`	Nombre de **x** dans **chaine**
`chaine.index(x) ou chaine.find(x)`	Indice de **x** dans **chaine**
`chaine[i:j]`	Extraction des caractères situés de i à j-1
`chaine[i:j:p]`	Extraction des caractères situés de i à j-1 par pas de p
`chaine.split(motif)`	Renvoie une liste des sous-chaînes séparées par le motif
`chaine.replace(motif1,motif2)`	Remplace toutes les occurrences de motif1 par motif 2
`strip()`	Retire tous les espaces situés au début et à la fin de la chaîne.
`strip(motif)`	Retire tous les motifs situés au début et à la fin de la chaîne.
`x in chaine`	Teste si **x** appartient à **chaine**
`x not in chaine`	Teste si **x** n'appartient pas à **chaine**

```
>>> chaine1='vive les vacances'
>>> n=chaine1.count('v')
>>> print(n)
3
>>> chaine2=chaine1[1:3]
>>> print(chaine2)
iv
>>> chaine3=chaine2*3
>>> print(chaine3)
iviviv
>>> print(len(chaine3))
6
>>> chaine4=chaine3[0:5:2]
>>> print(chaine4)
iii
>>> liste1=chaine1.split()
>>> print(liste1)
['vive', 'les', 'vacances']
>>> chaine5='-'.join(liste1)
>>> print(chaine5)
vive-les-vacances
>>> liste2=chaine3.split('i')
>>> print(liste2)
['', 'v', 'v', 'v']
>>> chaine6=chaine5.replace('-','+')
>>> print(chaine6)
vive+les+vacances
>>> print(chaine5)
vive-les-vacances
>>>>>> chaine7='   help   '.strip()
>>> print(chaine7)
help
>>> chaine8=chaine7.strip('p')
>>> print(chaine8)
hel
>>> chaine9=chaine8+'lo'
>>> print(chaine9)
hello
>>>print(chaine9,chaine1)
hello vive les vacances
```