CHAPITRE 1 : VARIABLES, TYPES ET OPÉRATEURS

Une variable est un espace mémoire dans lequel il est possible de stocker une valeur (une donnée).

I. Le type int (integer : nombres entiers).

Pour **affecter** (on dit aussi assigner) la valeur 17 à la variable nommée age :

```
>>> nb = 17 # ceci est un commentaire
```

La fonction print affiche la valeur de la variable :

```
>>> print(nb) # Attention à la casse
17
```

La fonction type() retourne le type de la variable :

```
>>> type(nb)
<class 'int'>
```

int est le type des nombres entiers.

```
>>>  nb = nb + 1    # en plus court : nb+= 1
>>> nb
18
```

L'opérateur // donne le quotient de la division euclidienne :

```
>>> d = 450//360
>>> d
1
```

L'opérateur % donne le reste de la division euclidienne (opération modulo) :

```
>>> reste = 450 % 360
>>> reste
90
```

L'opérateur ** donne la puissance :

```
>>> Mo = 2**20
>>> Mo
1048576
```

II. Le type float (nombres en virgule flottante).

```
>>> b = 17.0        # le séparateur décimal est un point (et non une virgule)
>>> b
17.0
>>> type(b)
<class 'float'>
>>> a = 14.0/3.0
>>> b = 14/3
>>> a,b
(4.666666666666667,  4.666666666666667)
>>> type(a),type(b)
(<class 'float'>  <class 'float'>)
```

Les fonctions mathématiques :
Pour utiliser les fonctions mathématiques, il faut commencer par importer le module math :

```
>>> import math
```

La fonction dir() retourne la liste des fonctions et données d'un module :

```
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial',
'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Pour appeler une fonction d'un module, la syntaxe est la suivante :
module.fonction(arguments)
Pour accéder à une donnée d'un module : **module.data**

```
>>> math.pi                 # donnée pi du module math (nombre pi)
3.141592653589793
>>> math.sin(math.pi/4.0)   # fonction sin() du module math (sinus)
0.7071067811865475
>>> math.sqrt(2.0)          # fonction sqrt() du module math (racine carrée)
1.4142135623730951
```

Pour importer toutes les fonctions du module et éviter l'écriture module.fonction :

```
>>> from module import *
```

III. Le type str (string : chaîne de caractères).

```
>>> nom = 'Keller'                  # entre apostrophes (simple quote)
>>> nom, type(nom)                  # ne pas oublier la virgule
('Keller', <class 'str'>)
>>> prenom = "Stéphane"             # on peut aussi utiliser les guillemets (double quote)
>>> prenom
'Stéphane'
>>> nom,prenom
('Keller', 'Stéphane')
```

La **concaténation** désigne la mise bout à bout de plusieurs chaînes de caractères.
La **concaténation** utilise **l'opérateur +**

```
>>> ch = nom + prenom       # concaténation de deux chaînes de caractères
>>> ch
'KellerStéphane'
>>> ch = prenom + nom       # concaténation de deux chaînes de caractères
>>> ch
'StéphaneKeller'
>>> ch = prenom + ' ' + nom
>>> ch
'Stéphane Keller'
>>> ch = ch + ' 18 ans'     # en plus court : ch += ' 18 ans'
>>> ch
'Stéphane Keller 18 ans'
```

La fonction len() retourne la longueur (length) de la chaîne de caractères :

```
>>> len(ch)
22
>>> ch[0]                       # premier caractère (indice 0)
'S'
>>> ch[1]                       # deuxième caractère (indice 1)
't'
>>> ch[-1]                      # dernier caractère (indice -1)
's'
>>> ch = 'Aujourd'hui'
SyntaxError: invalid syntax
>>> ch  = 'Aujourd\'hui'                # séquence d'échappement \'
>>> ch
"Aujourd'hui"
>>> ch  = "Aujourd'hui"
>>> ch
"Aujourd'hui"
```

La séquence d'échappement \n représente un saut ligne :

```
>>> ch = 'Première ligne\nDeuxième ligne'
>>> print(ch)       #la fonction print affiche à l'écran avec le formatage demandé
Première ligne
Deuxième ligne
```

La séquence d'échappement \t représente une tabulation :

```
>>> ch = 'Gauche\tDroite'
>>> print(ch)
Gauche   Droite
```

On peut utiliser les triples guillemets (ou les triples apostrophes) pour encadrer une chaîne définie sur plusieurs lignes :

```
>>> ch = """Première ligne
Deuxième ligne"""
>>> print(ch)
Première ligne
Deuxième ligne
```

On ne peut pas mélanger les serviettes et les torchons (ici type str et type int) :

```
>>> ch = '17.45'
>>> type(ch)
<class 'str'>
>>> ch = ch + 2
Traceback (most recent call last):
  File "<pyshell#114>", line 1, in <module>
    ch=ch+2
TypeError: Can't convert 'int' object to str implicitly
```

La fonction float() permet de convertir un type str en type float

```
>>> nb = float(ch)
>>> nb, type(nb)
(17.45, <class 'float'>)
>>> nb = nb + 2                 # en plus court : nombre += 2
>>> nb
```

```
19.45
```

La fonction input() lance une invite de commande (en anglais : prompt) pour saisir une chaîne de caractères.

```
>>> # saisir une chaîne de caractères et valider avec la touche Enter
>>> ch = input("Entrer un nombre : ")
Entrer un nombre : 14.56
>>> ch, type(ch)
('14.56', <class 'str'>)
>>> nb = float(ch)              # conversion de type
>>> nb**2                       #élévation au carré
211.99360000000001
```

IV. Le type list (liste) et le type range.

Le premier élément d'une liste possède l'indice (l'index) 0.
Dans une liste, on peut avoir des éléments de plusieurs types.

```
>>> info = ['Stéphane' , 'Keller' , 17 , 1.75 , 72.5]
>>> # la liste InfoPerso contient 5 éléments de types str, str, int, float et float
>>> type(info)
<class 'list'>
>>> info
['Stéphane', 'Keller', 17, 1.75, 72.5]
>>> print('Prénom : ', info[0])                    # premier élément (indice 0)
Prénom :  Stéphane
>>> print('age : ', info[2])                      # le troisième élément a l'indice 2
age :  17
>>> print('Taille : ', info[3])                   # le quatrième élément a l'indice 3
Taille :  1.75
```

La fonction range() crée une liste d'entiers régulièrement espacés :

```
>>> li = range(10)  # range(début : 0 par défaut, fin non comprise, pas : 1 par défaut)
>>> li, type(li)
(range(0, 10), <class 'range'>)
>>> len (li)
10
>>> li[3]
3
>>> li = range(1,10,2) # range(début, fin non comprise, pas)
>>> li
range(1, 10, 2)
>>> len (li)
5
>>> li[3]
7
```

La méthode **append()** de la classe **list** ajoute un nouvel élément en fin de liste :

```
>>> li=[3,2,1]
>>> li.append(4)
>>> li
[3, 2, 1, 4]
>>>
```

```
>>> li.sort()                    # la méthode sort() trie les éléments
>>> li
[1, 2, 3, 4]
```

V. Le type bool (booléen)

Deux valeurs sont possibles : True et False

```
>>> a = True
>>> type(a)
<class 'bool'>
```

Les opérateurs de comparaison :

Opérateur	Signification	Remarques
<	strictement inférieur	
<=	inférieur ou égal	
>	strictement supérieur	
>=	supérieur ou égal	
==	Égal	Attention : deux signes ==
!=	différent	
in	dans	
not in	Non dans	

```
>>> b = 10
>>> b>8
True
>>> b==5
False
>>> b!=10
False
>>> 0<= b <=20
True
```

Les opérateurs logiques : and, or, not

```
>>> note=13.0
>>> mentionAB = note>=12.0 and note<14.0 # ou bien : mentionAB = 12.0 <= note < 14.0
>>> mentionAB
True
>>> not mentionAB
False
>>> note==20.0 or note==0.0
False
```

L'opérateur in s'utilise avec des chaînes (type str) ou des listes (type list) :

```
>>> ch = 'Bonsoir'
>>> #  la sous-chaîne 'soir' fait-elle partie de la chaîne 'Bonsoir' ?
>>> 'soir' in ch
True
>>> 'b' in ch
False
>>> li = [4,8,15]
>>> >>> 9 in li #  le nombre entier 9 est-il dans la liste ?
False
```