Stéphane KELLER – Lycée agricole Louis Pasteur

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 nb :

```
>>> 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
>>> nb = nb - 3     # en plus court : nb-= 3
>>> nb
15
>>> nb = nb * 2     # en plus court : nb*= 2
>>> nb
30
>>> a = 6*3-20
>>> a
-2
>>> b = 25
>>> c = a + 2*b
>>> b, c            # ne pas oublier la virgule
(25, 48)
```

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'>)
```

Notation scientifique :

```
>>> a = -1.784892e4
>>> a,type(a)
(-17848.92, <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
>>> math.sqrt(-5.0)
Traceback (most recent call last):
  File "<pyshell#65>", line 1, in <module>
    print(math.sqrt(-5.0))
ValueError: math domain error
>>> math.exp(-3.0)                  # fonction exp() du module math (exponentielle)
0.049787068367863944
>>> math.log(math.e)                # fonction log() du module math (logarithme népérien)
1.0
```

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[1:4]                # du deuxième au quatrième caractère (indice 1 inclus à 4 exclus)
'tép'
>>> ch[2:]                 # à partir du troisième caractère
'éphane Keller 18 ans'
>>> ch[-1]                 # dernier caractère (indice -1)
's'
>>> ch[-6:]                # à partir du sixième caractère en partant de la droite
'18 ans'
>>> 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 la fonction range.

Une liste est une structure de données.
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
```

On peut créer une liste de listes, qui s'apparente à un tableau à 2 dimensions (ligne, colonne) :

```
0       1       2
10      11      12
20      21      22
```

```
>>> li = [[0,1,2],[10,11,12],[20,21,22]]
>>> li[0]
[0, 1, 2]
>>> li[0][0]
0
>>> li[2][1]
21
>>> li[2][1]=69
>>> li
>>>  [[0, 1, 2], [10, 11, 12], [20, 69, 22]]
```

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

```
>>> # instanciation d'une liste vide
>>> amis = []                                  # ou bien : amis = list()
>>> amis.append('Nicolas')  # synthase générale : objet.méthode(arguments)
>>> amis
['Nicolas']
>>> amis.append('Julie')      # ou bien : amis = amis + ['Julie']
>>> amis
```

```
['Nicolas','Julie']
>>> amis.append('Pauline')
>>> amis
['Nicolas','Julie','Pauline']
>>> amis.sort()                # la méthode sort() trie les éléments
>>> amis
['Julie', 'Nicolas', 'Pauline']
>>> amis.reverse()             # la méthode reverse() inverse la liste des éléments
>>> amis
['Pauline', 'Nicolas', 'Julie']
```

La méthode **lower()** de la classe str retourne la chaîne de caractères en casse minuscule :

```
>>> # la variable ch est une instance de la classe str
>>> ch = "BONJOUR"          # ou bien : ch = str("BONJOUR")
>>> ch2 = ch.lower()        # on applique la méthode lower() à l'objet ch
>>> ch2,ch
('bonjour', 'BONJOUR')
```

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' ?
>>> a = 'soir' in ch
>>> a
True
>>> 'b' in ch
False
>>> li = [4,8,15]
>>> #  le nombre entier 9 est-il dans la liste ?
>>> 9 in li
False
>>> 8 in li
True
>>> 14 not in li
True
```

VI. Autres types.

Nous avons vu les types les plus courants.
Il en existe bien d'autres :

- long (nombres entiers de longueur quelconque, par exemple
- 428496177556201253695415910 2L)
- complex (nombres complexes, par exemple 1+2.5j)
- tuple (structure de données)
- set (structure de données)
- file (fichiers)
- dict (dictionnaire).
- ...

Pour visualiser, pas-à-pas l'exécution d'un programme :
http://www.pythontutor.com/visualize.html#mode=display

VARIABLES, TYPES ET OPÉRATEURS : TESTS

I. Afficher la taille en octets et en bits d'un fichier de 536 Ko. On donne : 1 Ko (1 Kilooctet) = 2^{10} octets !!! 1 octet = 1 byte = 8 bits.
II. Le numéro de sécurité sociale est constitué de 13 chiffres auquel s'ajoute la clé de contrôle (2 chiffres). Exemple : 1 89 11 26 108 268 91
La clé de contrôle est calculée par la formule : 97 - (numéro de sécurité sociale modulo 97)
Retrouver la clé de contrôle de votre numéro de sécurité sociale.
Quel est l'intérêt de la clé de contrôle ?
III. Afficher la valeur numérique de $\sqrt{(4,6^3 - 15/16)}$. Comparer avec votre calculatrice.
IV. À partir des deux variables prenom et nom, afficher les initiales (on prendra Léa pour le prénom et Martin pour le nom et l'affichage sera alors LM).
V. L'identifiant d'accès au réseau du lycée est construit de la manière suivante : initiale du prénom puis les 8 premiers caractères du nom (le tout en minuscule).
Exemple : Alexandre Lecouturier → alecoutur
À partir des deux variables Prenom et Nom précédentes, construire l'identifiant.