Variables

Les entiers (type int)

In [1]:
```
#on définit un entier et on l'affiche
number = 3
print(number)
```

```
3
```

In [2]:
```
#on vérifie le type de la variable number
type(number)
```

Out[2]:
```
int
```

Les nombres flottants (type float)

In [3]:
```
#on définit un flottant
number_float = 3.0
print(number_float)
```

```
3.0
```

In [4]:
```
#on vérifie le type de la variable number_float
type(number_float)
```

Out[4]:
```
float
```

In [6]:
```
#on peut transformer un float en int...
number_int = int(number_float)
print(number_int)
print(type(number_int))
```

```
3
<class 'int'>
```

In [7]:
```
#... et vice versa, int en float
number_float = float(number_int)
print(number_float)
```

```
3.0
```

Les listes (type list)

In [8]:
```
#on définit une liste simple de nombres
liste_numbers = [1,1,1,2,3,6,5]
print(liste_numbers)
```

```
[1, 1, 1, 2, 3, 6, 5]
```

In [9]:
```
#on vérifie le type de la variable liste_numbers
type(liste_numbers)
```

Out[9]:
```
list
```

In [10]:
```
#on peut transformer une liste en set...
liste_set = set(liste_numbers)
print(liste_set)
type(liste_set)
```

```
{1, 2, 3, 5, 6}
```

Out[10]:
```
set
```

In [11]:
```
#... et vice versa, set en liste
liste = list(liste_set)
print(liste)
```

```
[1, 2, 3, 5, 6]
```

In [12]:
```
#transformer un int en liste --> impossible
number_to_list = list(3)
print(number_to_list)

#la commande list() n'accepte que des iterables !
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-b902e64fa4d0> in <module>
      1 #transformer un int en liste --> impossible
----> 2 number_to_list = list(3)
      3 print(number_to_list)
      4
      5 #la commande list() n'accepte que des iterables !

TypeError: 'int' object is not iterable
```

Les chaines de caractères (type str)

In [13]:
```
#on définit une chaine de caractères
char = "la team mpx sont les + bo"
print(char)
```

```
la team mpx sont les + bo
```

In [14]:
```
#on vérifie le type de la variable char
type(char)
```

Out[14]:
```
str
```

In [16]:
```
#on peut transformer une nombre (int ou float) en str...
number = 8
char_number = str(number)
print(type(char_number))
print(char_number)
```

```
8
```

In [17]:
```
#... et vice versa de str en (int ou float)
char = "8"
number_char = int(number)
print(char_number)
```

```
8
```

In [18]:
```
#On peut aussi transformer un str en list...
char = "la team mpx sont les + bo"
liste_char = list(char)
print(liste_char)
```

```
['l', 'a', ' ', 't', 'e', 'a', 'm', ' ', 'm', 'p', 'x', ' ', 's', 'o', 'n', 't', ' ', 'l', 'e', '
s', ' ', '+', ' ', 'b', 'o']
```

In [20]:
```
#... et vice versa de liste en str
liste = [1,2,3,"la team mpx sont les + bo"]
char_liste = str(liste)
print(char_liste)
type(char_liste)
```

```
[1, 2, 3, 'la team mpx sont les + bo']
```

Out[20]:
```
str
```

Les tuples (type tuple)

In [21]:
```
#on définit un tuple
tup = (1,2)
print(tup)
```

```
(1, 2)
```

In [22]:
```
#On peut transformer un tuple en list...
liste = list(tup)
print(liste)
```

```
[1, 2]
```

In [23]:
```
for i in liste:
    print(i)
```

```
1
2
```

In [25]:
```
#On peut transformer un tuple en str...
char = str(tup)
print(char)
type(char)
```

```
(1, 2)
```

Out[25]:
```
str
```

In [27]:
```
for i in char:
    print(type(i))
```

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
```

Les booléens (type bool)

In [28]:
```
#un booléen est restreint à prendre deux valeurs : True et False
number_3 = 3
number_0 = 0
empty_liste = []
liste_num = [1,23,5,6,5]
char_empty = ""
char = "Team MPX"
print(f'Le booléen de {number_3} est', bool(number_3))
print(f'Le booléen de {number_0} est', bool(number_0))
print(f'Le booléen de {empty_liste} est', bool(empty_liste))
print(f'Le booléen de {liste_num} est', bool(liste_num))
print(f'Le booléen de {char} est', bool(char))
print(f'Le booléen de {char_empty} est', bool(char_empty))
```

```
Le booléen de 3 est True
Le booléen de 0 est False
Le booléen de [] est False
Le booléen de [1, 23, 5, 6, 5] est True
Le booléen de Team MPX est True
Le booléen de  est False
```

In [29]:
```
#une condition retourne toujours un booléen
print(f"The equality between {number_0} and 0 is", (number_0 == 0))
print(f'The equality between {liste_num} and [1,23,5,6,5] is', (liste_num == [1,23,5,6,5]))
print(f'The equality between {char} and oui is', (char == "oui"))
```

```
The equality between 0 and 0 is True
The equality between [1, 23, 5, 6, 5] and [1,23,5,6,5] is True
The equality between Team MPX and oui is False
```

Les nombres complexes

In [30]:
```
# on définit un nombre complexe z
z = 1+1j
print(z)
print(type(z))
```

```
(1+1j)
<class 'complex'>
```

In [31]:
```
# On peut récupérer la partie réel de z
reel = z.real
print(reel)
```

```
1.0
```

In [32]:
```
# On peut récupérer la partie imaginaire de z
imaginaire = z.imag
print(imaginaire)
```

```
1.0
```

Opérations mathématiques

In [ ]:
```
a = 12
b = 3
puissance = a ** b #a élevé à la puissance b

#definition de la division euclidienne : a = b*q + r
div_reste = b % a #reste de la division entre b et a = r
div_entier = b // a #division entière entre b et a
div = b / a #quotient entre b et a = q

multi = a * b
sous = a - b
addition = a + b
```

Conclusion

1. On affecte une valeur (int, float, str, list etc.) à une variable

2. Les transformations possibles sont :

1. str <--> int (ou float)
2. list <--> set
3. list <--> str

3. Suivez bien les consignes des exos qui demandent bien souvent des types bien précis pour les variables