Rappels sur Python

Les variables

Rappels

Déclaration ou modification d'une variable :

In [1]:

```
premiere_var = "Hello, world"
```

Affichage

In [2]:

```
print(premiere_var)
```

Commentaires :

In [3]:

```
# Mon premier commentaire
```

L'objet None

L'objet None permet de représenter l'absence de valeur d'une entité.

Zoom sur les types

Entier (int), flottant (float), booléen (True, False), liste (list), chaîne de caractères (string), dictionnaire.

In [4]:

```
entier = 2
flottant = 3,5
boole = True
chaine = "Hello,"
chaine_2 = 'world'
liste = [1,"Toto", 4.2]
dico = {
    "cle" : "valeur",
    "cle_2" : 2
}
```

Opérations possibles

Sur les entiers : Addition (+), soustraction (-), multiplication (*), division (/), quotient de la division euclidienne (//), reste de la division euclidienne (%), test (égalité : ==, différence : !=, <, >, <=, >=).

In [5]:

```
2+1 != 3
```

Out[5]:

```
False
```

Sur les flottants : Addition (+), soustraction (-), multiplication (*), division (/).

Sur les booléens : and, or, not.

Sur les chaînes : Concaténation (+), accès au $i^{ème}$ élément ([i], avec numérotation commençant à 0. Pour accéder au dernier élément, prendre i$= -1$), taille (len()), test d'appartenance (in).

In [6]:

```
chaine + " " + chaine_2
```

Out[6]:

```
'Hello, world'
```

Sur les listes : Concaténation (+), ajout d'un élément à la fin (.append()), accès au $i^{ème}$ élément ([i], avec numérotation commençant à 0. Pour accéder au dernier élément, prendre i$= -1$), taille (len()), supprimer un élément (del), test d'appartenance (in).

In [7]:

```
print(len(liste))
liste.append(7)
print(liste)
liste = [12] + liste
print(liste[0])
print(liste[-1])
print(liste)
del liste[1]
liste
```

```
3
[1, 'Toto', 4.2, 7]
12
7
[12, 1, 'Toto', 4.2, 7]
```

Out[7]:

```
[12, 'Toto', 4.2, 7]
```

Sur les dictionnaires : accès et modification à une valeur à partir d'une clé ([clé]), ajout d'une paire clé-valeur, suppression d'une paire (del), test d'existence d'une clé (in).

In [8]:

```
dico["cle"] = "nouv_valeur"
print("2 in dico :", 2 in dico)
print("'cle' in dico :", "cle" in dico)
dico
```

```
2 in dico : False
'cle' in dico : True
```

Out[8]:

```
{'cle': 'nouv_valeur', 'cle_2': 2}
```

Instructions basiques

Conditions

Syntaxe :

In [9]:

```
if True :  #condition (bool)
    None #opérations à effectuer si la condition est vérifiée
elif True :  #autre condition (bool)
    None #opérations à effectuer si la première condition n'est pas vérifiée, mais que
 la seconde l'est
else:
    None #opérations à effectuer sinon
```

Boucles

Boucle for

Syntaxe :

In [10]:

```
liste_de_valeurs = [0,1,3,5]

for k in liste_de_valeurs:
    print(k)#instructions
```

```
0
1
3
5
```

Pour énumérer des entiers :

In [11]:

```
for k in range(1, 10, 2): #range(premier entier, entier suivant le dernier, pas)
    print(k)
```

```
1
3
5
7
9
```

Par défaut, le premier entier est indexé à 0 et le pas à 1.

In [12]:

```
for k in range(10):
    print(k)
```

```
0
1
2
3
4
5
6
7
8
9
```

Boucle while

Syntaxe :

In [13]:

```
i = 0

while i != 10 : #condition (bool)
    i = i + 1
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

Fonction

Syntaxe :

In [14]:

```
def fonction(variable_1, variable_2):
    None #corps de la fonction
```

Pour que la fonction renvoie un résultat, on utilise la commande return (qui stoppe l'execution de la fonction)

In [15]:

```
def f(x,y):
    if x == y:
        return True
    else:
        return False

f(1,2)
```

Out[15]:

```
False
```

In [16]:

```
#remarque : ici, on pouvait directement écrire:

def f(x,y):
    return x == y

f(2,2)
```

Out[16]:

```
True
```