# td1a_correction_session1

April 14, 2021

## 1 1A.0 - Premiers pas en Python (correction)

La partie 1 ne nécessite pas de correction.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

### 1.0.1 Partie 2

Les erreurs sont corrigées de telle sorte que la ligne fautive est mise en commentaires ou que le commentaire indique la correction.

**Un oubli**

[2]:
```
a = 5
a = a + 4  # a + 4
print (a)
```

```
9
```

Si on affecte pas le résultat d'une opération à une variable, il est perdu.

**Une erreur de syntaxe**

[3]:
```
a = 0
for i in range (0, 10) :  # il manquait : en fin de ligne
    a = a + i
print (a)
```

```
45
```

**Une autre erreur de syntaxe**

[4]:
```
a = 0
for i in range (0, 10):
    a = a + i
    print (a)      # il manquait deux espaces
```

```
0
1
3
6
```

```
10
15
21
28
36
45
```

L'indentation est stricte en Python : elle indique la séparation entre des blocs d'instructions. Si les lignes ne sont pas bien alignées, l'interpréteur ne sait plus à quel bloc associer la ligne.

**Une opération interdite**

[5]:
```
a = 0
s = "e"
# print (a + s)  # pas de correction
```

Il n'existe pas de correction dans ce cas, il s'agit d'une opération interdite. De la même manière l'opération suivante est interdite aussi car on ajoute des informations de type différent :

[6]:
```
1 + "1"     # déclenche une exception
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-db0423a23ce2> in <module>()
----> 1 1 + "1"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Pour faire le calcule, il faut convertir les deux variables dans le même type :

[7]:
```
print(1 + int("1"))
print(str(1) + "1")
```

```
2
11
```

Le premier résultat est un entier. Le second est une chaîne de caractères. Les deux chaînes "1" + "1" ont été collées ensemble.

**Un nombre impair de...**

[8]:
```
a = 0
for i in range (0, 10) :
      a = (a + (i+2)*3)  # il manquait une parenthèse à la fin
print (a)
```

```
195
```

### 1.0.2 Partie 3

**Ecrire un programme qui calcule la somme des 10 premiers entiers au carré**

[9]:
```
s = 0
for i in range(1,11):   # range énumère les entiers de 1 à 11 exclu
    s += i
print(s)
```

```
55
```

Une autre version en écriture abrégée :

[10]:
```
r = sum ( i for i in range(1,11) )
print(r)
```

```
55
```

**Ecrire un programme qui calcule la somme des 5 premiers entiers impairs au carré**

[11]:
```
s = 0
for i in range(1,11):
    if i % 2 == 1 :
        print("ajoute",i)
        s += i
print("résultat",s)
```

```
ajoute 1
ajoute 3
ajoute 5
ajoute 7
ajoute 9
résultat 25
```

De la même manière, en écriture abrégée :

[12]:
```
r = sum ( i for i in range(1,11) if i%2 == 1 )
print(r)
```

```
25
```

**Ecrire un programme qui calcule la somme des qui 10 premmières factorielles : $\sum_{i=1}^{10} i!$**

[13]:
```
s = 0
f = 1
for i in range(1,11):
    f *= i
    s += f
print("résultat",s)
```

```
résultat 4037913
```

Il faut introduire une variable intermédiaire car le résultat qu'on cherche à calculer est la somme d'un produit. A chaque itération, on utilise le résultat de la somme à l'itération précédente ainsi que la valeur de la factorielle. La notation abrégée est plus difficile à écrire.