I11 - Programmation I : Python
TP 1

Semestre 1 2018

1 Premier pas avec *Idle*

Exercice 1. L'interprète

1. Créez un répertoire I11 et un répertoire TP1 dans votre répertoire de travail et faites de TP1 votre répertoire courant.
2. Lancez Idle. De quelle version de Python s'agit-il ? Assurez-vous que ce soit la version 3.1 ou une version ultérieure.
3. Taper dans l'interprète

```
>>> "Hello \n world"
```

4. Tapez

```
>>> print("Hello \n world")
```

Quelle différence ?

Exercice 2. Premier script

1. Créez un nouveau fichier dans l'éditeur de texte de Idle. Ecrire dans ce fichier la commande précédente.
2. Enregistrez ce fichier dans votre répertoire TP1 sous le nom `tp1_1`
3. Quelle est l'extension qui a été donnée à ce fichier ?
4. Exécutez ce script.
5. Où sont les résultats de ce script ?
6. Modifiez `tp1_1.py` comme suit

```
# Mon mon premier script
print("Hello \n world")
"Hello \n world"
```

7. Exécutez de nouveau le script ? Commentez ce qui se passe.
8. Faites le point sur les deux modes d'utilisation de Idle ou de Python en général.
9. Complétez le compte rendu avec CQFR de l'exercice.

2 Expressions et types simples

Exercice 3. Entiers

1. Testez dans l'interprète et commentez les résultats des expressions suivantes :

```
>>> type(4)
>>> 20 / 3
>>> 20 // 3
>>> 20 % 3
>>> _ ** 3
```

2. À quoi sert la variable prédéfinie '_' ?
3. Priorité des opérateurs. Testez et commentez :

```
>>> 7 + 3 * 4
>>> (7 + 3) * 4
>>> 15 // 2 **  2 % 3
```

4. Testez et commentez :

```
>>> 18 @ 3
>>> 15 / 0
```

5. Essayez de trouver la limite des nombres entiers en faisant :

```
>>> 1 * 1000
>>> _ * 1000 # Refaire cela plusieurs fois.
```

6. En supposant qu'un chiffre soit codé sur un seul octet en mémoire et en allant chercher la quantité de mémoire vive de votre système, quel est le plus grand nombre que vous puissiez obtenir ?
7. En considérant qu'un chiffre à l'écran a pour dimension 10 pixels de large et 20 pixels de haut et en utilisant la résolution de votre écran, combien d'écrans faut-il pour pouvoir afficher ce nombre ?
8. En considérant le temps qu'il vous faut pour taper dans l'interprète la commande

```
>>> _ * 1000
```

combien de temps vous faudrait-il pour atteindre ce nombre ?

Exercice 4. Booléens

1. Testez dans l'interprète et commentez le résultat :

```
>>> type(True)
>>> type(False)
```

2. Testez dans l'interprète et commentez :

```
>>> 1<2<5
>>> (1<2) or (2==0)
>>> not 1 == 2
>>> not 1 == 2 and 5<10
```

3. Priorité des opérateurs. Essayez et commentez :

```
>>> not (True or False) == not True and not False
>>> not (True or False) == (not True and not False)
```

Exercice 5. Flottants Ils sont notés par la présence d'un point décimal "." ou une notation exponentielle

1. "e" ou "E" dans le nombre :

```
>>> 2.718
>>> 3e8
>>> .2
>>> 1.
```

2. Le type flottant, tester dans l'interprète :

```
>>> type(4.56)
```

3. Les opérateurs arithmétiques sont les mêmes que ceux déjà vus. Cas particulier de la division entre entiers, essayez et commentez :

```
>>> 8 / 4
```

4. Autres opérateurs Expérimentez : que font ces opérateurs ? Fonctionnent-ils avec les autres types déjà vus ?

```
>>> 20.0 // 3
>>> 20.0 % 3
>>> 2.0 ** 10
```

Exercice 6. Chaînes de caractères

1. Le type chaîne `str`, tester dans l'interprète :

```
>>> type ("toto")
>>> type ('titi')
>>> ''' Voici
      une chaine de plusieurs
      lignes '''
>>> print (''' Voici
      une chaine de plusieurs
      lignes ''')
>>> """ "oui", j'aime bien faire du paddle """
>>> 'tata"
```

2. La longueur d'une chaîne :

```
>>> len('')
>>> len('taratata')
>>> len('Abra ca da bra')
```

3. La concaténation de chaînes :

```
>>> 'mon premier' + 'script' + 'en' + 'Python'
```

4. La répétition :

```
>>> ('la' * 4 + ' ') * 3
```

Exercice 7. Transtypage

1. Testez dans l'interprète et commenter

```
>>> float (2)
>>> int (2.3)
>>> int (5/2)
>>> int ('abc')
>>> str(125)
>>> float('2.3')
>>> float('2,3')
>>> bool('')
>>> bool('toto')
>>> bool(0)
>>> bool(12)
```

3 Affectation et entrée/sortie

EXERCICE 8. **Affectation** L'affectation est une instruction qui attribue une valeur à une variable.

```
>>> a = 1
```

il faut comprendre que l'on relie un contenu (la valeur 1) à un contenant (la variable a).

1. Tapez ces exemples et commentez :

```
>>> a = 1
>>> type(a)
>>> msg = "Quoi de neuf ?"
>>> type(msg)
>>> pi = 3.14159
>>> type(pi)
>>> x,y,z = 1,5,7
>>> x,y,z
```

2. Pour l'affectation `a=1`, il y création d'un nom "a" dans l'espace des noms qui **référence** un objet correspondant à la valeur entière 1, et lorsque vous écrivez `b=a`, il y a création d'un nom "b" qui référence le même objet que le nom a. On peut vérifier cela en utilisant la fonction `id()` de Python qui retourne l'identificateur unique associé à l'objet (en fait son adresse en mémoire) :
   Tester

```
>>> a = 1
>>> id(a)
>>> b = a
>>> id(b)
>>> a = 2
>>> id(a)
```

3. Testez les instructions suivantes et faire le suivi des références des variables avec la fonction `(id(var))` (donne la référence de var), déduisez en une représentation graphique de la mémoire pour l'exécution des instructions suivantes.

```
# Affectation et reference
a = int (input())
a = a + 1
b = 3
x = b
a = 2*x + a
b == 3
nul = x == 0
y = a = b
```

EXERCICE 9. **Entrée/Sortie** Les entrées (saisies de valeurs via le clavier) se font avec la fonction `input()` et les sorties (affichage à l'écran) se font avec la fonction `print()`.

1. Tester dans l'interprète et commenter :

```
>>> ch = input()
>>> type (ch)
>>> ch
>>> print (ch)
>>> a = int (input("Entrer votre age:" ))
>>> a
>>> print ("Age : ",a)
```

2. Écrivez un script tp1_2.py qui permette de saisir deux valeurs entières dans deux variables a et b et d'afficher leur somme, différence, produit et rapport.

3. Améliorez l'affichage en utilisant le fait que print peut prendre plusieurs paramètres pour afficher une indication avant chaque valeur :

```
Somme = 20
```

4. Utilisez enfin l'affichage avec des chaînes formatées pour produire une sortie plus conviviale :

```
Somme de 12 et 8 = 20
```

5. Vous devriez obtenir un affichage final par exemple :

   Exemple d'éxécution :

```
>>>
 a = 12
 b = 8
 Affichages simples :
 20
 4
 96
 1.5
 Affichages ameliores :
 Somme = 20
 Difference = 4
 Produit = 96
 Rapport = 1.5
 Affichages coorectement presentes:
 Somme de 12 et 8 = 20
 Difference de 12 et 8 = 4
 Produit de 12 et 8 = 96
 Rapport de 12 sur 8 = 1.5
```

6. Écrivez un script qui effectuera succéssivement les calculs suivant à partir de données saisies en entrée par l'utilisateur :
   (a) calcul du périmètre et l'aire d'un cercle à partir du rayon
   (b) calcul de l'aire d'un trinagle à partir de la longueur des trois cotés (formule de Héron)
   (c) conversion d'une température en dégrés celsius en température en degrés fahrenheit
   (d) conversion d'un nombre de secondes en heures/min/sec.

Exemple d'éxécution :

```
>>>
 Rayon du cercle : 2.5
 Perimetre = 15.7075, Aire = 19.634375

 Longueur du premier cote : 2
 Longueur du deuxieme cote : 1.7
 Longueur du troisieme cote : 3
 Aire = 1.616089

 Temperature en degre Celsius : 22
 Temperature en degre fahrenheit : 71.6

 Nombre de secondes : 15451
 Heures/minutes/secondes : 4h17m31s
```