Informatique

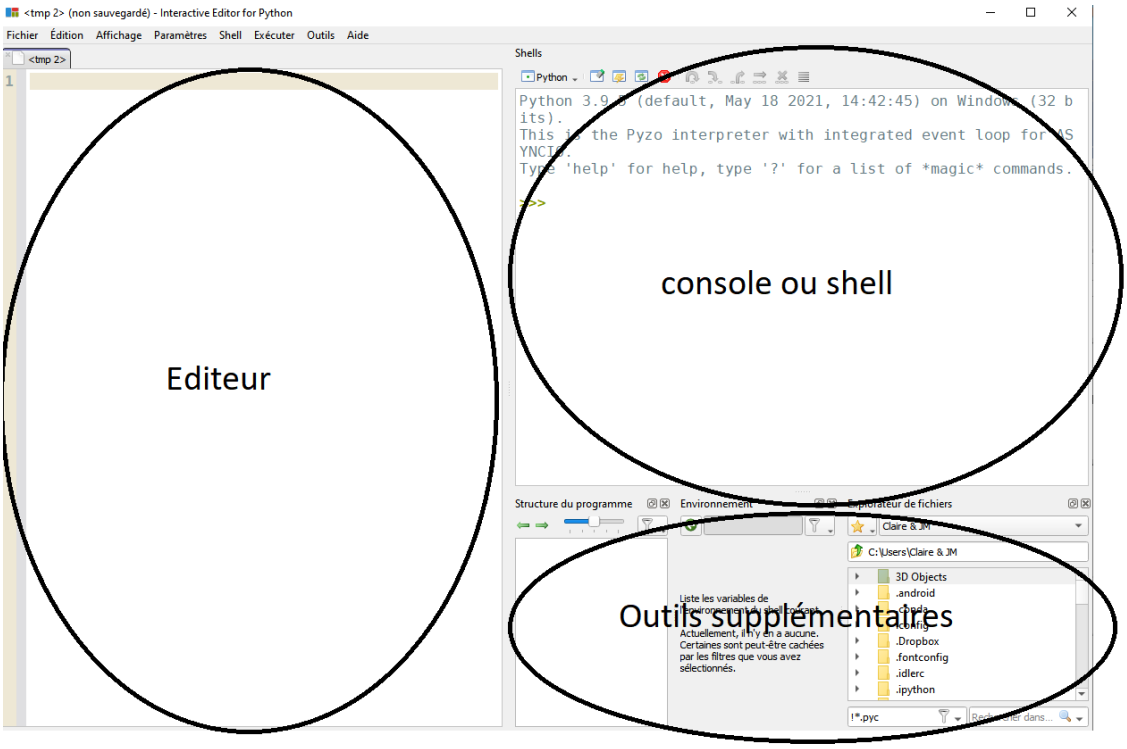
0
Bases Python

TD 0-1
Prise en main

Consignes générales pour les TP : Commencer la séance en créant un dossier Informatique dédié sur votre PC ou votre clé et pour chaque nouveau TP, créer un fichier adapté dans ce dossier.

Exercice 1: Bases python

Lancer PYZO (ou Spyder). Repérer les différentes fenêtres composant l'environnement :



Une instruction tapée dans la console et validée par la touche « Entrée » est immédiatement exécutée. Les instructions tapées dans la console ne peuvent pas être sauvegardées.

Question 1: Taper et exécuter dans la console les instructions suivantes tout en comprenant ce qu'il se passe

```
x = 42
y = 42.0
type(x)
type(y)
x = x+y
type(x)
```

```
z = (1>0)
z
type(z)
t = (0==1) or (1!=2)
t
```

```
L = [1,3,6,9,12]
type(L)
L[0]
L[1]
len(L)
L[4]
L[5]
L[-1]
L[-2]
L[1:4]
L[4:1]
L+L
L.append(15)
L
dir(L)
help(L.append)
L = L.append(18)
L
```

Question 2: Si cela n'a pas déjà été fait, ouvrir un nouveau fichier (CTRL+N ou File/New) et l'enregistrer (CTRL+S ou File/Save) sous le nom TP0-1.py un dossier personnel. Ecrire dans ce fichier le code ci-dessous, le comprendre et l'exécuter (F5 ou Run/Exécute file)

```
def somme1(n):
    """Renvoie 0+1+2+...+n
    Précondition: n doit être un entier naturel"""
    s = 0
    for k in range(n+1):
        s += k
    return s
test = somme1(5)
print(test)
```

Question 3: Taper ensuite dans la console les instructions suivantes (et comprendre)

```
Somme1(10)
Somme1(-2)
Somme1(120)==120*121/2
help(somme1)
```

Remarque : on pourra ajouter `print(s,k)` dans la bouche for de somme1 afin d'identifier le comportement de la fonction aux différents appels

Question 4: Ecrire une fonction factorielle1(n) qui, étant donné un entier naturel n, renvoie n! , à l'aide d'une boucle for. On rappelle que n! = 1×2×3×…×n. Pensez à tester votre fonction (avec des valeurs vérifiables !!)

Question 5: Ecrire une fonction factorielle2(n) qui, étant donné un entier naturel n, renvoie n! à l'aide d'une boucle while

Question 6: Dans votre fichier, taper et exécuter les instructions suivantes et vérifier les valeurs de L1,L2,L3,L4,L5 (vérification dans la console par exemple)

```
L1 = [i for i in range(10)]
L2 = [compteur**2 for compteur in range(7)]
L3 = [j+1 for j in range(-2,5)]
L4 = [j for j in range(1,8,2)]
L5 = []
for k in range(5):
    L5.append(k**2)
```

Question 7: Ecrire une fonction generer_liste_entiers_1(deb,fin) qui, étant donnés deux entiers deb et fin (deb<fin), renvoie la liste des entiers compris entre deb et fin exclus. On utilisera une boucle for. Pensez à tester votre fonction après écriture

Question 8: Ecrire une fonction generer_liste_entiers_2(deb,fin) qui, étant donnés deux entiers deb et fin, renvoie la liste des entiers compris entre deb et fin tous deux exclus. On utilisera une boucle while

Question 9: Ecrire une fonction liste_diviseurs(n) qui, étant donné un entier naturel n, renvoie la liste des diviseurs positifs de n qui sont strictement inférieurs à n (on pourra préalablement tester dans la console les instructions a//b et a%b avec des entiers a et b)

Exemple de test :

```
>>> liste_diviseurs(50)
[1, 2, 5, 10, 25]
```

Question 10: Ecrire une fonction somme2(L) qui, étant donnée une liste de nombres L, renvoie la somme des éléments de L en itérant sur les indices de L

Question 11: Ecrire une fonction somme3(L) qui, étant donnée une liste de nombres L, renvoie la somme des éléments de L en itérant sur les éléments de L

Un nombre entier non nul est dit parfait s'il est la somme de ses diviseurs positifs stricts.

Question 12: En utilisant les fonctions précédentes écrire une fonction est_parfait(n) qui renvoie True si l'entier n est parfait et False sinon.

Par exemple, 6 est parfait (1+2+3=6) mais 8 ne l'est pas (1+2+4=7)

Question 13: Ecrire une fonction liste_parfait_1(N) qui renvoie la liste des premiers nombres parfaits inférieurs ou égaux à N en utilisant une boucle for

Exemple :

```
>>> liste_parfait_1(1000)
[6, 28, 496]
```

Question 14: Ecrire une fonction liste_parfait_2(N) qui renvoie la liste des N premiers nombres parfaits en utilisant une boucle while.

Question 15: Ecrire un code permettant de stocker dans la variable quatrieme_nombre le $4^{ème}$ nombre parfait (en utilisant la fonction liste_parfait_2)