

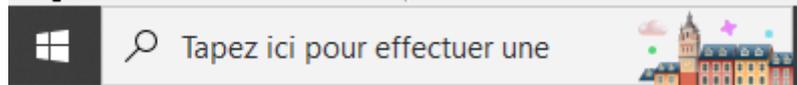
TP 0 : Expérimentation

Un cours efficace, rappels de lycée : <http://python.lycee.free.fr/>

Pour s'entraîner en autonomie : <http://www.france-ioi.org/algo/chapters.php>

Utiliser Spyder Anaconda 3

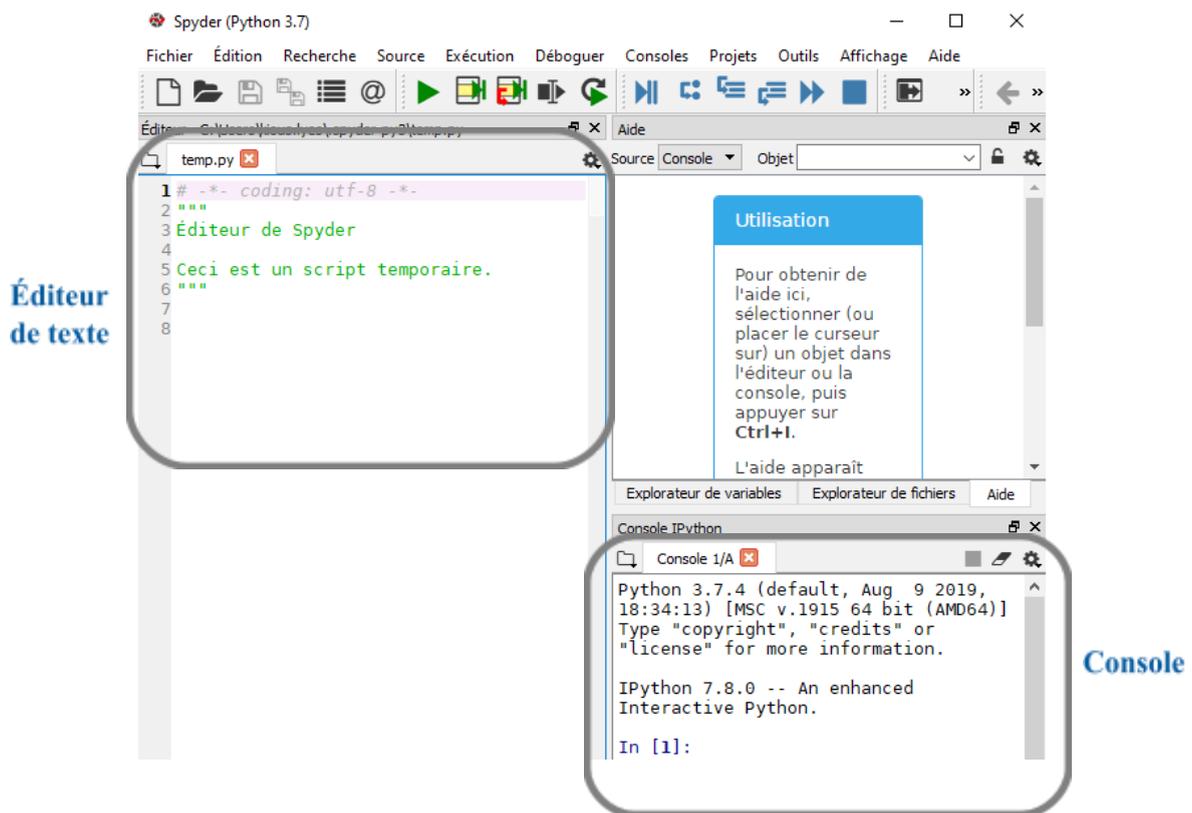
Sur un ordinateur du réseau Sainte Anne, en bas à gauche :



Taper : « spyder » et lancer :



Dans un premier temps on exécutera les instructions dans la console (interprète de commandes) à la suite de l'invite de commande (le prompt).



1. Opérations élémentaires

Exécuter (une à une) les instructions suivantes et en déduire les fonctions ci-dessous :

7+3	7*3	type(2<1)	(2<3) and (2<1)
7.0-3	7/3	2+1 == 4	(2+1!= 4) and not(2+1==4)
abs(-7.3)	7**2	not(2+1) == 4	12.54
type(2.0)	7//3	2+1! = 4	a = 2/3
type(2)	7%3	(2<3) or (2<1)	b = 1.89
int(2.3)	2<1	(5<3) or (2<1)	a,b = b,a
			print(a,b)

abs		%	
type		true	
float		false	
int		bool	
int		==	
*		!=	
/		=	
**		print	
//			

2. Chaîne de caractères

Une chaîne de caractères est une séquence d'éléments de type **str** c'est à dire caractère.

Exécuter (une à une) les instructions suivantes et en déduire les fonctions ci-dessous.

```

type('salut')      str(2+3)          ch[6:10]          'rné' in ch
c='bon'            ch='belle journée' ch[6:]           len(ch)
c+'jour'          ch[0]             'a' in ch        c = ch[0] + ch[7] + ch[10]
type('789')      ch[-2]             'a' not in ch    len(c)
                                                           str(2+3)+5      comment corriger cette dernière instruction ?
    
```

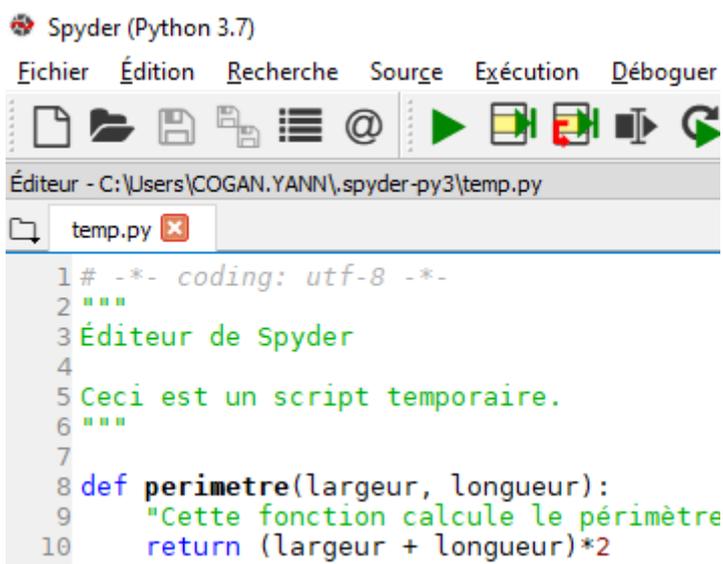
+ appliqué à deux données de type str	
in	
len	

Par la suite on va utiliser l'éditeur de texte (partie gauche de l'écran) et l'exécution des instructions se fera par le menu :

Exécution → Exécuter

ou le bouton : 

ou la touche F5



3. Fonction

La syntaxe de l'objet fonction sous python est :

```
def nom-fonction (liste des paramètres de la fonction) :  
    # Commentaire précisant l'action de la fonction,  
    # les variables d'entrée, et ce que retourne la fonction .  
    bloc d'instructions  
    return résultat de la fonction
```

L'appel de la fonction s'effectue par *nom-fonction(paramètre)*.

Exemple :

```
def perimetre(largeur, longueur) :  
    # Cette fonction calcule le périmètre d'un rectangle  
    # en fonction de sa largeur et de sa longueur"  
    return (largeur + longueur)*2
```

Tester cette fonction dans la console (où on remarque que la fonction définie dans l'éditeur de texte est disponible dans la console après que le texte ait été exécuté).

Exercice 3.1 Écrire une fonction **moyenne** qui prend en entrée trois nombres réels et retourne la moyenne de ces nombres.

Exercice 3.2 Que renvoie la fonction mystère suivante ?

```
def mystere (a,b) :  
    # Fonction mystère  
    # Variables : a entier positif et b entier strictement positif.  
    return a%b == 0
```

.....

4. Liste

Une liste est une séquence d'éléments qui peuvent être de types différents ou être de même type.

Une liste est mutable c'est à dire qu'il est possible de modifier un élément, d'en ajouter, d'en supprimer.

Les éléments d'une liste sont indexés : index du premier élément 0.

```
x = range(6)  
for n in x:  
    print(n)
```

5. Instructions conditionnelles

La syntaxe d'une instruction conditionnelle simple est :

```
if expression booléenne :  
    bloc d'instructions  
else :  
    autre bloc d'instructions
```

Exemple :

```
c = input("entrer un caractère :")  
mot = "bien"  
if c in mot:  
    res="le caractère est dans le mot"  
else:  
    res="essaie encore"  
print(res)
```

Cas d'alternatives multiples (if ... elif...else).

```
if expression booléenne 1:  
    bloc 1 d'instructions  
elif expression booléenne 2:  
    bloc 2 d'instructions  
else :  
    autre bloc d'instructions
```

Si le booléen 1 correspondant à `if` est vérifié (`True`) alors le bloc 1 est exécuté.

Sinon le booléen 2 est testé et s'il est vérifié alors le bloc 2 est exécuté.

Si le booléen 2 n'est pas vérifié alors le bloc correspondant à l'instruction `else` est exécuté.

Exemple

```
def categorie (n):  
    "entrée : nombre réel compris entre 0 et 20 retourne la catégorie correspondante."  
    categorie=["premier quart","deuxième quart","troisième quart", "quatrième quart"]  
    if n < 5:  
        return categorie[0]  
    elif n < 10:  
        return categorie[1]  
    elif n < 15:  
        return categorie[2]  
    else:  
        return categorie[3]
```

Boucle séquentielle (boucle for)

Les instructions sont exécutées au fur et à mesure du parcours d'une séquence. Une boucle séquentielle se termine toujours. La syntaxe d'une boucle séquentielle est :

```
for element in sequence :  
    bloc d'instructions
```

Exercice 5.1 On considère le bloc d'instructions suivant :

```
s=0
for k in range(5):
    s = s + k
```

Simuler la boucle ci-dessous en recopiant et complétant le tableau suivant et dire ce qu'elle fait.

tour de boucle	variable s	variable k
entrée de boucle	0	
1 ^{er} tour		
2 ^e tour		
...		
sortie de boucle		

Cette boucle

Écrire une fonction qui prend comme variable un entier naturel n et renvoie la somme des entiers de 0 à n .

Boucle conditionnelle (boucle while)

Les instructions de la boucle sont exécutées tant que la condition est vraie. Il faut être attentif au fait que si la condition est toujours vérifiée, la boucle peut tourner à l'infini. La syntaxe d'une boucle conditionnelle est :

```
while condition :
    bloc d'instructions
```

Exercice 5.2

```
i = 2
while i <= 10:
    print(f"Table de multiplication par {i} :")
    value = 1
    while value <= 10:
        print(f"{i} x {value} = {i * value}")
        value += 1
    i += 1
    print() # Un saut de ligne à chaque table
print("Et voilà !")
```

Que réalise la fonction `print(f "...{...}...")` ? :

Quel serait le code de l'instruction : `i+=1` ? :

Écrire le code d'un programme qui réalise le même affichage, mais avec l'instruction **for**.

Exercice 5.3

Écrire une fonction qui affiche une table de multiplication choisie par l'utilisateur.