NSi	Fonctions, boucles	TP2

Créer un répertoire TP2 pour y enregistrer les fichiers.

Une fonction peut prendre plusieurs paramètres ou un seul ou aucun. Tester les deux exemples qui suivent.

Exemple 1. Voici une fonction aire_cercle qui prend un paramètre, représentant le rayon d'un cercle, et renvoie l'aire de ce cercle.

```
def aire_cercle(r):
    """r est du type float
        renvoie l'aire du cercle de rayon r"""
    return 3.14159 * r * r
```

Exemple 2. Voici une fonction aire_rectangle qui prend deux paramètres, représentant la largeur et la longueur d'un rectangle, et renvoie l'aire de ce rectangle.

```
def aire_rectangle(larg, long):
    """larg et long sont du type float
        renvoie l'aire du rectangle de dimension larg et long"""
    return larg * long
```

Exercice 1

Créez un fichier PGM1.py et définissez une fonction ligne_car(n, car) où n est un entier naturel et car de type str, qui renvoie une chaîne composée de n fois la chaîne car.
Par exemple, l'expression ligne_car(3, "bon") a pour valeur "bonbonbon".

Exercice 2

Créez un nouveau fichier PGM2.py et définissez une fonction volume_boite(x1,x2,x3) qui renvoie le volume d'une boîte parallélépipédique dont on fournit les trois dimensions x1, x2, x3 en arguments.
Par exemple, l'expression volume_boite(2,3,4) a pour valeur 24.

Exercice 3

Créez un nouveau fichier PGM3.py et définissez une fonction qui attend le prix hors taxes (pht) d'un article et un taux de TVA, puis qui retourne le prix toutes taxes comprises (pttc) de l'article.
Rappels : tva = pht * taux_tva / 100 et pttc = pht + tva
Dans le programme principal, on entrera ensuite dans cet ordre le prix hors taxes et le taux de TVA (de type float), puis on affichera le prix TTC calculé par appel à la fonction.
Par exemple, le prix TTC d'un article de prix HT 171.5 au taux de TVA 8.8 % est 186,592.

Avec une boucle conditionnelle **while** ou "boucle non bornée"
Le principe est le même qu'avec la structure conditionnelle if : si une condition est réalisée, un bloc d'instructions est exécuté. Mais ici, à la fin de l'exécution du bloc, on vérifie si la condition est encore réalisée et si c'est le cas, on exécute à nouveau le bloc d'instructions. Et ainsi de suite, on recommence tant que la condition est réalisée.

Exercice 4

Etudier le programme suivant et expliquer le résultat renvoyé par la fonction mystere.

```
def mystere(x, n):
    cpt = n
    res = 0
    while cpt > 0:
        res = res + x
        cpt = cpt - 1
    return res
```

Exercice 5

Etudier le programme suivant et expliquer le résultat renvoyé par la fonction mystere.

```
def mystere(n):
    cpt = 0
    while n > 0:
        r = n % 10
        n = n // 10
        cpt = cpt + r
    return cpt
```

Peut-on remplacer la boucle while par une boucle for ?

Exercice 6

Dans un fichier PGM6.py, écrire une fonction multiples17 qui affiche tous les multiples de 17 inférieurs strictement à un entier n. Le nombre n est un paramètre de la fonction. Utiliser dans le code une boucle while.

Exercice 7

Dans un fichier PGM7.py, écrire une fonction qui prend en paramètre un nombre entier de secondes et affiche le nombre de minutes et de secondes correspondants. Par exemple si le paramètre vaut 275, alors la fonction affiche 4 minutes et 35 secondes.

Les seules opérations autorisées sont l'addition et la soustraction. On utilisera une boucle while.

Avec une boucle **for** ou "boucle bornée"

Il s'agit d'une structure itérative : on répète l'exécution d'un bloc d'instructions et le nombre de répétitions est bien déterminé.

Exercice 8

Voici ci-dessous un code avec une boucle for. Recopier-le dans un fichier PGM8.py et tester-le.

```
for i in range(4):
    print("i a pour valeur", i)
```

Ecrire ensuite un code qui produit le même résultat mais en remplaçant la boucle for par une boucle while.

De manière générale, une boucle for peut toujours être remplacée par une boucle while.

Exercice 9

Écrire dans un fichier PGM9.py une fonction qui affiche les 20 premiers termes de la table de multiplication par 7, en signalant au passage (à l'aide d'un astérisque) ceux qui sont des multiples de 3. Exemple : 71421*283542*49

Exercice 10

Écrire dans un fichier PGM10.py une fonction qui calcule les 50 premiers termes de la table de multiplication par 13, mais n'affiche que ceux qui sont des multiples de 7.

Exercice 11

Écrire dans un fichier PGM11.py une fonction qui affiche la suite de symboles suivante :

```
*
**
***
****
*****
```

Exercice 12

Écrire dans un fichier PGM12.py une fonction qui prend une chaîne de caractères en paramètre et renvoie la même chaîne mais avec le symbole * inséré entre les caractères. Par exemple : si la chaîne passée en paramètre est "informatique", la chaîne renvoyée est "i*n*f*o*r*m*a*t*i*q*u*e".

Exercice 13

Écrire dans un fichier PGM13.py une fonction qui calcule et affiche les carrés des n premiers entiers non nuls. La fonction prend en paramètre un nombre n de type int.

Exercice 14

Écrire dans un fichier PGM14.py les deux fonctions qui suivent puis les tester avec différentes valeurs de n de type int.

```
def affiche1(n):
    for i in range(n):
        print("iteration", i)
        print("bonjour")
        if i == 2:
            break
    print("fin itération", i)

def affiche2(n):
    for i in range(n):
        print("iteration", i)
        print("bonjour")
        if 2 < i < 6:
            continue
    print("fin itération", i)
```

Exercice 15

Recopier dans un fichier PGM15.py la fonction qui suit.

```
def affiche3(n):
    for i in range(n):
        print(i)
```

Modifier le code afin d'afficher uniquement les multiples de 5.
1ère méthode : modifier uniquement la ligne for i in range(5).
2ème méthode : utiliser l'instruction continue.
3ème méthode : ? ? ?

Exercice 16

Écrire dans un fichier PGM16.py une fonction qui prend deux paramètres n et p de type int et affiche un rectangle de n lignes et p colonnes formé d'étoiles.
Par exemple, pour n = 3 et p = 4, la fonction affiche :

```
****
****
****
```

Exercice 17

Écrire dans un fichier PGM17.py une fonction qui prend un paramètre n de type int et affiche le motif ci-contre avec n le nombre de lignes. Pour cet exemple, n = 5.

```
 ##
###
 ##
###
 ##
```

Exercice 18 Rectangle vide

Créez un nouveau fichier PGM18.py et définissez une fonction qui attend un caractère et deux nombres entiers c et h, puis qui dessine le rectangle vide de h lignes et c colonnes dessiné avec le caractère donné. Par exemple, si le caractère est *, que h = 5 et c = 6, la fonction dessine le motif ci-contre :

```
******
*    *
*    *
*    *
******
```

Exercice 19 Phrase encadrée

Créez un nouveau fichier PGM19.py et définissez une fonction qui attend une phrase et un caractère, puis qui encadre la phrase avec le caractère donné en laissant des espaces autour de la phrase. Par exemple, si le caractère est % et que la phrase est "Homer et Bart vont à la plage", la fonction affiche :

```
  %%%%%%%%%%%%%%%%%%%%
%                      %
 % Homer et Bart vont à la plage %
%                      %
  %%%%%%%%%%%%%%%%%%%%
```

Exercice 20 Portée des variables

Devinez ce que va afficher le code suivant lors de son exécution. Vérifiez sur machine.

```
x, y, z = 1, 3, 5

def f(x, y, u):
    x = y + 1
    z = u + 3
    y = x + z
    return x, y, z

print(f(x, y, z))
print(x, y, z)
```