Activité : Les fonctions

Introduction

L'écriture d'un programme peut devenir très rapidement long avec des morceaux de codes qui se répètent. Pour rendre le programme plus lisible et éviter de réécrire des lignes de codes identiques, on peut dans de nombreux langages de programmation utiliser des **fonctions**.

Les fonctions regroupent en un même endroit les lignes de codes qui se répètent. Ensuite, les fonctions sont appelées quand le programme en a besoin.

Dans un exercice sur les boucles, nous avons été amenés à créer des dessins avec différents motifs qui pouvaient se répéter de nombreuses fois. On va réaliser les mêmes figures en utilisant des fonctions.

La syntaxe en python

- Une fonction en python est introduite par le mot clef **def** suivi du nom de la fonction, de 2 parenthèses et les deux points :
- Les instructions de la fonction sont toutes **indentées** à l'intérieur de la fonction
- le résultat ou la valeur de la fonction est renvoyé au programme avec le mot clef **return**

Entrée [1]:
```
def mafonction():
    """
    Dans une fonction on peut avoir:
    -> des variables internes
    -> des instructions conditionnelles
    -> des boucles
    """
    a = 5
    """
    Après exécution des instructions, la fonction renvoie une valeur:
    -> un nombre entier ou flottant
    -> un booléen True ou False
    -> une chaine de caratères
    -> tout type de valeur
    Remarques :
    1) les variables internes à la fonction sont détruites après avoir
    renvoyé la valeur avec return
    2) return met fin à l'exécution de la fonction
    """
    return a
```

Entrée [2]:
```
# On appelle la fonction et on mémorise la valeur renvoyée dans la variable v
v=mafonction()
```

Entrée [3]:
```
# On affiche le contenu de la variable v
print(v)
```

```
5
```

Dans l'exemple ci-dessus :

- **Entrée [1] :** contient la fonction avec les instructions à réaliser lorsqu'elle est appelée ;
- **Entrée [2] :** on appelle la fonction et la valeur retournée sera affectée à la variable v ;
- **Entrée [3] :** on récupère la valeur de la variable v renvoyée par la fonction et on l'affiche.

Partie 1 : avec une fonction

Pour réaliser le dessin composé de 10 lignes de 10 fois la lettre X, on propose le programme suivant :

```
# On initialise le motif avec du vide, pas de caractère
motif=""

# Une première boucle qui va réaliser 10 lignes
for _ in range(10):
    # Une seconde boucle qui ajoute 10 fois la lettre X au motif
    for _ in range(10):
        motif=motif+"X"
    # On sort de la seconde boucle, le motif vaut "XXXXXXXXXX",
    # on ajoute alors au motif le caractère spécial \n qui équivaut à aller à la ligne.
    motif=motif+"\n"

# La première boucle est terminée, on 10 lignes de motif qu'on affiche
print(motif)
```

```
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```

On va réaliser le même dessin en utilisant une fonction.

1) Créer la fonction **ligne_X()** qui renvoie un motif composé de 10 fois la lettre **X** et du caractère spécial de retour à la ligne.

2) Écrire le programme ci-dessus en remplaçant la seconde boucle par l'appel à la fonction **ligne_X()**. On utilisera une variable dessin à laquelle on affectera la valeur renvoyée par la fonction.

3) Vérifier que le dessin est bien réalisé.

Partie 2 : une fonction et des paramètres

La fonction précédente ne peut réaliser le dessin qu'avec 10 lignes de 10 **X**.

Il est possible de modifier cette valeur en passant un paramètre **n** à la fonction qui correspondra au nombre de X à afficher ainsi que le nombre de lignes du dessin.

Pour y parvenir, il faut :

1) Réécrire la fonction précédente en la renommant **ligne_nX(n)**.

- **a)** Ajouter le paramètre **n** entre les parenthèses de la fonction correspondant au nombre de X du motif.
- **b)** Remplace la valeur 10 par le paramètre **n** dans la fonction.

2) Réécrire le programme en y insérant la fonction **ligne_nX(n)** et en initialisant la valeur de n.

3) Vérifier votre programme en réalisant le dessin avec plusieurs valeurs de n.

En supplément :

4) On va modifier la fonction et le programme pour réaliser un dessin avec un autre caractère que le X.

- **a)** Ajouter un second paramètre **c** à votre fonction pour remplacer la lettre X par n'importe quel autre caractère.
- **b)** Modifier le programme pour réaliser le dessin avec le caractère choisi.

5) Ajouter des instructions pour que le nombre de lignes et le caractère soit saisi par l'utilisateur.

Partie 3 : un programme et plusieurs fonctions

Pour réaliser la figure suivante, on peut utiliser plusieurs boucles et des print.
Cette façon de faire n'est pas du tout optimisée.

```
for _ in range(10):
    print("X",end="")
print()
for _ in range(8):
    print("X",end="")
    for _ in range(8):
        print(" ",end="")
    print("X",end="")
    print()
for _ in range(10):
    print("X",end="")
print()
```

```
XXXXXXXXXX
X        X
X        X
X        X
X        X
X        X
X        X
X        X
X        X
XXXXXXXXXX
```

Vous allez réécrire ce code en utilisant 2 fonctions :

— une fonction qui renvoie la première et la dernière ligne ;

— une fonction qui renvoie les lignes intermédiaires avec du vide ;

Ci-dessous l'algorithme pour coder les deux fonctions et le programme :

```
def ligne_nX(n,c="X"):
    # On initialise le motif avec du vide, pas de caractère

        # On ajoute au motif le caractère passé en paramètre

    # on ajoute alors au motif le caractère spécial \n qui équivaut à aller à la ligne.

    # On renvoie le motif


def lignes_X_X(n,c):
    # On initialise le motif avec le caractère passé en paramètre

    # On ajoute les espaces avec une boucle for

    # On finit le motif en ajoutant le caractère et le retour à la ligne

    # On renvoie le motif


n=...
c=...
dessin=""
# On ajoute à dessin la première ligne remplie de caractères

# On ajoute avec une boucle les lignes blanches bordées par les caractères


# On finit le dessin avec la dernière ligne (comme la première)


print(dessin)
```

1) Écrire la première fonction en utilisant ce qui a déjà été fait ;

2) Écrire la seconde fonction en suivant l'algorithme ;

3) Terminer par le programme et faire des essais pour vérifier le bon fonctionnement.

Partie 4 : dernière figure

On donne le programme et la figure suivante :

```
n=int(input("n="))
# On crée la première ligne remplie de X
for _ in range(n):
    print("X",end="")
print()
# On crée les lignes suivantes jusqu'à la moitié
for k in range(1,n//2):
    for _ in range(k):
        print(" ",end="")
    print("X",end="")
    for _ in range(k+1,n-k-1):
        print(" ",end="")
    print("X",end="")
    for _ in range(n-k,n//2):
        print(" ",end="")
    print()
# On vérifie la parité du nombre de ligne:
# si n impair on écrit X, sinon rien
if n%2==1:
    for _ in range(n//2):
        print(" ",end="")
    print("X",end="")
    for _ in range(n//2,n):
        print(" ",end="")
    print()
# On crée les lignes dans la seconde moitié
for k in range(n//2+1,n-1):
    for _ in range(n-k-1):
        print(" ",end="")
    print("X",end="")
    for _ in range(n-k,k):
        print(" ",end="")
    print("X",end="")
    for _ in range(k+1,n):
        print(" ",end="")
    print()
# On termine avec la dernière ligne
for _ in range(n):
    print("X",end="")
print()
```

```
n=17
XXXXXXXXXXXXXXXXX
 X               X
  X             X
   X           X
    X         X
     X       X
      X     X
       X   X
        X X
         X
        X X
       X   X
      X     X
     X       X
    X         X
   X           X
  X             X
 X               X
XXXXXXXXXXXXXXXXX
```

Réécrire ce code en utilisant des fonctions.