Programmation avec Python
Initiation aux fonctions

NOM :

Les fonctions permettent d'ajouter de nouvelles instructions au langage de programmation.
La syntaxe est la suivante : **def** nomDeLaFonction(*liste de paramètres*):

- Attention : Pas de caractère spécial ou accentué dans le nom d'une fonction.

I-Fonction simple sans paramètres

En mode interactif, taper les lignes suivantes :

```
def table7( ):                  # Les parenthèses et les « : »sont obligatoires
        n=1                     # L'indentation est obligatoire
        while n<11:
                print n*7,      # la virgule pour un affichage sur la même ligne
                n=n+1


table7( )
```

↪ .....

Que fait cette fonction ? .....

II-Fonction avec paramètre

En mode interactif, taper les lignes suivantes :

```
def table(base):
        n=1
        while n<11:
                print n*base,
                n=n+1


table(13)
```

↪ .....

```
table(9)
```

↪ .....

Exercice 1 : En utilisant la fonction table( ) définie ci-dessus, écrire un programme qui affiche les tables de la table du 1 à la table du 20.

Exercice 2 : En important le module turtle, écrire une fonction carre(taille, couleur) qui dessine un carré en fonction de la longueur de son côté et dans une couleur précisée (terminer par a=**input( )** pour que le programme attende et ne referme pas la fenêtre tout de suite).

III-Variables locales, variables globales

Lorsque nous définissons des variables à l'intérieur du corps d'une fonction, ces variables ne sont accessibles qu'à la fonction elle même. On dit que ces variables sont des variables locales à la fonction.
C'est le cas des variables taille et couleur de la fonction carre(taille, couleur).
Les variables définies à l'extérieur d'une fonction sont des variables globales. Leur contenu est visible de l'intérieur d'une fonction, mais la fonction ne peut pas le modifier.

En mode interactif, taper les lignes suivantes :

```
def mask( ):
        p=20
        print p , q
p , q=15 , 38
mask( )
```

Analysez attentivement cet exemple pour comprendre ce qui se passe.

↪ .....

```
print p , q
```

↪ .....

Instruction global

On peut quand même modifier une variable globale par une fonction.
Étudiez ces deux exemples :

```
def monter( ):
        global a
        a=a+1
        print a
a=15
monter( )
```

↪ .....

```
monter( )
```

↪ .....

```
def monter2( ):
        a=a+1
        print a
a=15
monter2( )
```

↪ .....

```
monter2( )
```

↪ .....

IV-<u>« Vraies » fonctions et procédures</u>

Les fonctions écrites jusqu'à présent ne sont pas des fonctions au sens strict. Mais plus exactement des procédures.
Python utilise la même instruction pour les deux, ce qui n'est pas le cas d'autres langages de programmation.
Une « vraie » fonction doit renvoyer une valeur lorsqu'elle se termine.

<u>Exemples</u> :

```
def cube(w):
        return w*w*w
b=cube(2)
print b
```

↪ .....

```
def table(base):
        result=[]               # result est une liste vide
        n=1
        while n<11:
                b=n*base
                result.append(b)# ajout d'un terme à la liste
                n=n+1
        return result
ta9=table(9)
print ta9
```

↪ .....

```
print ta9[0]
```

↪ .....

```
print ta9[2:5]
```

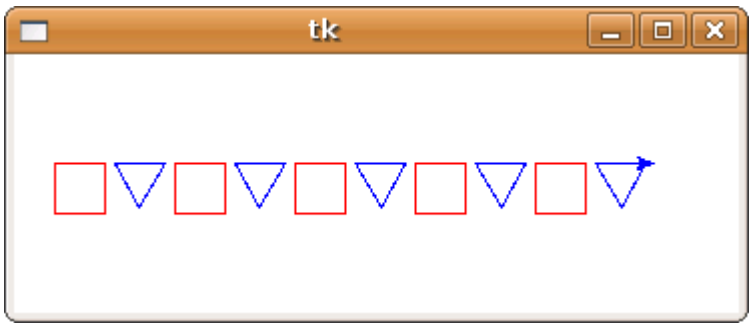
↪ .....

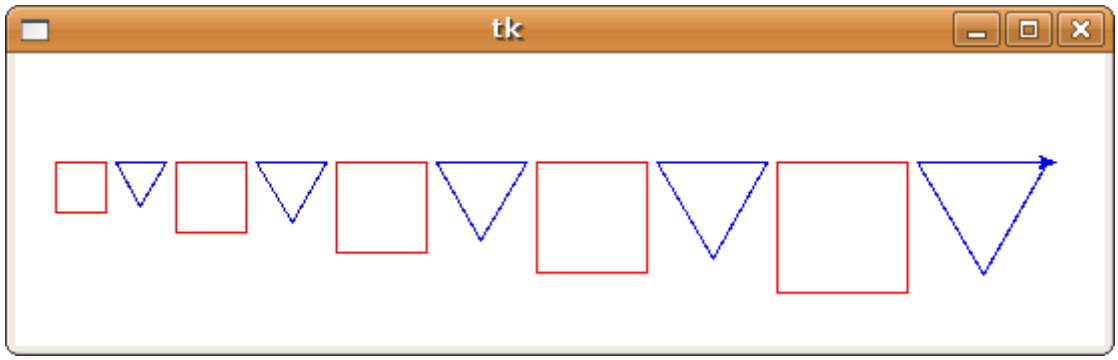
V-Un programme

Taper le programme suivant, l'enregistrer, puis l'exécuter.

```
from turtle import*
def carre(taille, couleur):
        "fonction qui dessine un carré de taille et de couleur déterminées"
        color(couleur)
        c=0
        while c<4:
                forward(taille)
                right(90)
                c=c+1
######Programme principal######
up()                            #relever le crayon
goto(-150, 50)                  #reculer en haut à gauche
#dessiner dix carrés rouges alignés :
i=0
while i<10:
        down()                  #abaisser le crayon
        carre(25, 'red')        #tracer un carré à l'aide de la fonction défini au début
        up()                    #relever le crayon
        forward(30)             #avancer plus loin
        i=i+1
a=input()                       #attendre
```

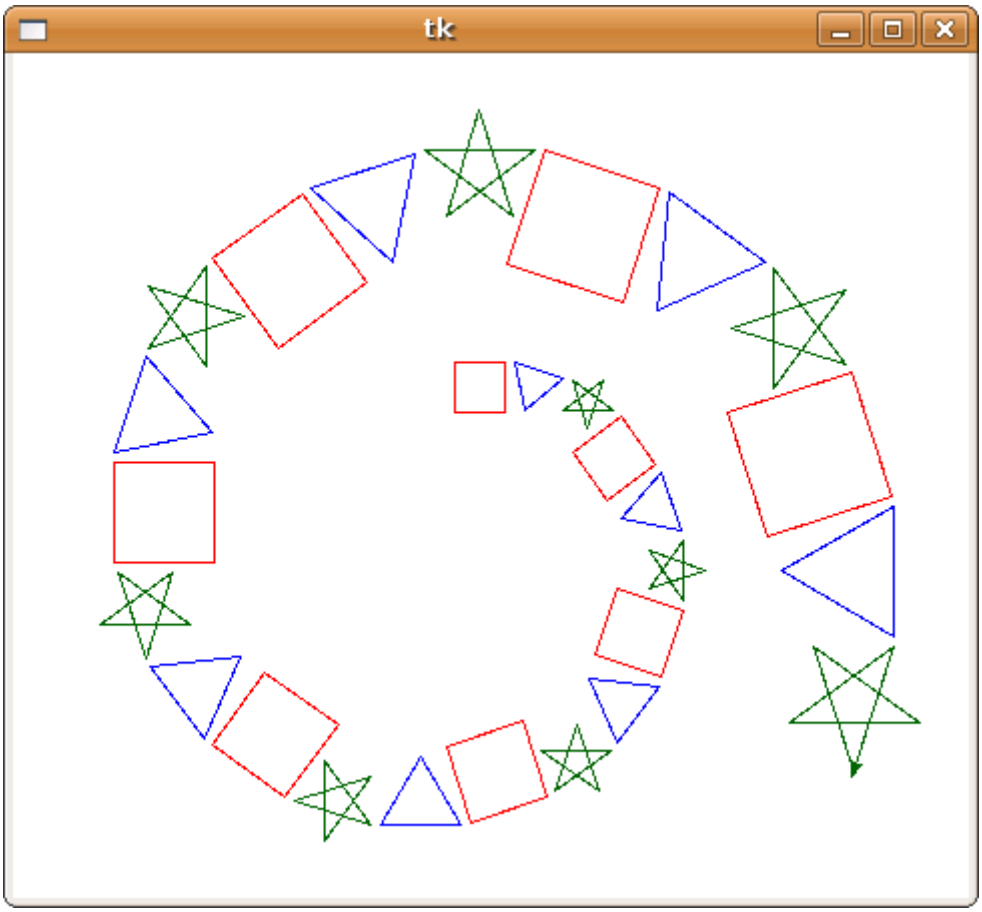
Exercice 3 : Définissez une fonction triangle(taille, couleur) capable de dessiner un triangle équilatéral d'une taille et d'une couleur bien déterminées.
Écrire des programmes qui feront appel aux fonctions triangle( ) et carre( ) à plusieurs reprises, avec des arguments variés pour dessiner une série de carrés et de triangles :





Exercice 4 : Écrire une fonction etoile5( ) spécialisée dans le dessin d'étoiles à 5 branches. Dans votre programme principal, insérez une boucle qui dessine une rangée horizontale de 9 petites étoiles de tailles variées :

Exercice 5 : Ajouter un paramètre angle aux trois fonctions carre( ), triangle( ) et etoile5( ), de manière à ce que les figures puissent être tracée dans différentes orientations.
Le programme principal devra tracer une spirale comme celle-ci (angle=18°) :



VI-Valeurs par défaut pour les paramètres

Dans l'interpréteur de commande, taper :

```
def politesse(nom, vedette='Monsieur'):
        print 'Veuillez agréer,", vedette, nom, "mes salutations distinguées."

politesse('Dupont')
```

↳ .....

```
politesse('Durand', 'Mademoiselle')
```

↳ .....

ⓘ Attention : Les paramètres sans valeur par défaut doivent précéder les autres dans la liste. La définition ci-dessous est incorrecte :

```
def politesse(vedette='Monsieur', nom):
```