1 Les fonctions

Une fonction est introduite par le mot-clé `def`, suivi du nom de la fonction, de ses paramètres (ou arguments) entre parenthèses s'il y en a, de deux points, puis d'un bloc d'instructions.

```
def ma_fonction(var1,var2,...):
        bloc instructions
        return ....
```

Ce bloc d'instructions s'appelle le corps de la fonction. Il doit être obligatoirement indenté et la fin de ce bloc marque la fin de la définition de fonction. L'instruction `return` (ou `return if ....`) qui veut dire renvoyer (ou renvoyer si...) est une instruction de sortie de la fonction. Toutes instructions écrites après ne sont pas prises en compte.

```
def moyenne( a,b):
    m=(a+b)/2
    return(m)
```

Après avoir exécuté le script, la fonction peut être appelée dans la console par exemple par `moyenne(5,8)` ou `print(moyenne(5,8))`
On peut aussi mettre les appels de fonctions dans le script lui-même souvent obligatoire dans les applications tablettes ou pour une carte microbit que vous utiliserez sans doute pour le thème "informatique embarquée".

Exemple d'une fonction qui fait défiler le message 'Hello !' sur une carte microbit :

A tester avec l'émulateur sur le site https://create.withcode.uk/

```
from microbit import*

def AfficheHello():
  display.scroll('Hello !')

AfficheHello() # on appelle la fonction dans le script
```

Un exemple avec le module PIL utile pour le traitement des photos numériques
Les photos numériques (au programme de SNT) sont des images matricielles composées de pixels qui sont associés à une ou trois valeurs (en général).
Si une photo est en noir et blanc : la valeur associée à un pixel vaut 0 ou 1.
Si une photo est en niveaux de gris : la valeur associée à un pixel est un entier entre 0 et 255 (2^8 valeurs possibles correspondant à 1 octet) correspondant à la luminosité (0 pour noir et 255 pour blanc).
Si une photo est en couleurs RGB : un pixel est associée à un triplet de valeurs (r, g, b) où r, g et b sont des entiers entre 0 et 255 correspondant respectivement à l'intensité de rouge, vert et bleu.

```
from PIL import Image # Image est un sous module du module PIL

def unecouleur(nbColonne,nbLigne,r,g,b,nomFichier):
   img = Image.new('RGB', (nbColonne, nbLigne), color = (r, g, b))
   img.save(nomFichier)
```

Exercice 1

1. Écrire dans un script, le sauvegarder, l'exécuter.
2. Dans la console, taper l'instruction suivante : `unecouleur(200,600,255,0,0,'unecouleur.png')`
   Pour voir le résultat, ouvrir le fichier `unecouleur.png` que vous trouverez dans le dossier où vous avez sauvegardé le script.
3. Faites des essais en modifiant les différents paramètres de la fonction.

Remarque :
Dans une fonction, l'instruction `print` renvoie un affichage, elle ne permet pas de récupérer la valeur d'une variable. Pour cela on utilise l'instruction `return`.

Voici deux fonctions, une qui renvoie l'aire d'un disque en fonction de la valeur du rayon et une autre qui renvoie le volume d'un cylindre en fonction du rayon et de la hauteur.
Écrire dans un script et tester avec les valeurs de votre choix.

```
from math import pi

def aireDisque(r):
    return(pi*r**2)

def volumeCylindre(r,h):
    base=aireDisque(r)
    volume=base*h
    return(volume)
```

Exercice 2

On dispose d'une image en couleurs qu'on souhaite afficher en niveaux de gris. Chaque pixel de l'image en couleurs est représenté par un triplet de nombres compris entre 0 et 255 pour les canaux respectifs de Rouge, de Vert et de Bleu (RGB). Pour la transformer en une image en niveaux de gris, il suffit de remplacer pour chaque pixel, le triplet `(r,g,b)` par un triplet `(gris,gris,gris)` où `gris` est un nombre entre 0 et 255.
La "recommandation 709" propose une moyenne pondérée à partir de `(r,g,b)` en prenant `gris=0.2126r+0.7152g+0.0722b`.

1. Écrire une fonction `enGris(t)` qui renvoie le triplet `(gris,gris,gris)` selon la "recommandation 709" où `t` est un tuple contenant 3 éléments.
   On rappelle que `t[0]` est le premier élément de `t` et que `g` doit être un entier.
2. Calculer le niveau de gris associé à un pixel jaune du logo Python pour lequel `r=254, g=220, b=95`.

Remarque : Le module PIL possède une fonction qui permet de récupérer le triplet `(r,v,b)` associé à un pixel.

Exercice 3

Le rectorat de Caen est à une latitude de 49.1794765 et à une longitude de −0.3829918.
Source : https://demarchesadministratives.fr/rectorat/caen-14000

On souhaite écrire une fonction permettant de convertir une donnée en degrés en une donnée en degrés, minutes et secondes.

1. Compléter la fonction `degreTodms(val)` pour obtenir une liste contenant le nombre de degrés, le nombre de minutes et le nombre de secondes correspondant à `val`

```
LatRectorat=49.1794765
LongRectorat=-0.3829918

def degreTodms(val):
    # val est un nombre réel entre 0 et 90
    nbDeg=int(val)# on rappelle que int(r) où r est de type float renvoie la troncature
    minutes=(val-valDeg)*60 # donne les minutes en degrés
    nbMin=.....
    secondes=.....# donne les secondes en minutes
    nbSec=.....
    return([nbDeg,nbMin,nbSec])
```

2. Que valent les coordonnées gps du rectorat en degrés, minutes et secondes ?

2 Structure conditionnelle

En Python, une structure conditionnelle est introduite par le mot clé `if` dont la syntaxe est :

```
if condition :
    bloc instructions
```

`condition` est une expression dont le résultat est un booléen. Si la condition est vraie, le bloc est exécuté.

Les points importants à ne pas oublier sont :
- les deux points après la condition
- l'indentation du bloc d'instructions qui doit être exécuté.

Pour exécuter un bloc d'instruction si la condition est fausse, on utilise le mot clé `else` avec la syntaxe suivante :

```
if condition :
    bloc instructions
else :
    bloc instructions
```

Cette structure est conseillée même s'il n'y a aucune instruction après `else`.
Dans ce cas, on utilise le mot clé `pass` et on écrit :

```
if condition :
    bloc instructions
else :
    pass
```

Exemple :
A partir d'un capteur de luminosité installée par exemple dans une pièce de la maison, on peut récupérer la valeur de l'intensité lumineuse. La fonction ci-dessous renvoie un conseil à l'occupant :

```
def lumiere(x):
"""renvoie un message en fonction de l'intensité lumineuse (en lux)  """
   if  x<400:
       return('allumer la lumière')
   else:
       return('éteindre la lumière')
```

Exercice 4

Dans python, le module `PIL` permet de manipuler des images et entre autres de récupérer la valeur associée à un pixel d'une image en niveaux de gris.
On souhaite transformer une image en niveaux de gris en une image en noir et blanc en remplaçant la valeur d'un pixel par 0 ou par 255 selon que celle-ci est inférieure à 127 ou non.

Compléter la fonction `seuil(x)` que l'on utilisera plus loin :

```
def seuil(x):
   if ..........:
       n=............
   else:
       n=............
   return(n)
```

Si on veut tester plusieurs conditions imbriquées, on peut utiliser le mot clé `elif` :

```
if condition1 :
    bloc instructions
elif condition2 :
    bloc instructions
elif ... :
       ...
else:
    bloc instructions
```

Exemple avec la carte microbit ; celle-ci possède un capteur de température et la fonction `temperature()` *renvoie la valeur de celle-ci.*

```
from microbit import*

def Message(t):
    if t<10:
        display.scroll("it's cold")
    elif t<19:
        display.scroll("it's not hot")
    else:
        display.scroll("it's hot")

Message(temperature())
Message(2)
Message(15)
Message(35)
```

Exercice 5

On considère une image en niveaux de gris. On souhaite transformer cette image en trois "couleurs" : noir (0) , gris moyen (127) et blanc (255) selon que la valeur du pixel d'origine est inférieur à 85, 116 ou 255.
Compléter la fonction `seuil2(x)` :

```
def seuil2(x):
   if x<85:
       n=..............
   elif ..............:
       n=..............
   else:
       n=.............
   return(n)
```

3 Structures de répétition

3.1 Boucles non bornées

En python, une boucle non bornée est introduite par le mot clé `while` dont la syntaxe est :

```
while condition :
       bloc instructions # sera répétée tant que condition est vraie
```

Exemple : faire clignoter la led centrale d'une carte microbit

A tester sur le site https://create.withcode.uk/.
La fonction `display.set_pixel(x,y,value)` allume le pixel situé à la colonne `x` et la ligne `y` avec une illumination `value` de 0 (éteint) à 9 (complètement allumé).

```
from microbit import *
from time import sleep

while True : # boucle infinie à n'utiliser que si on a un moyen de la stopper
  sleep(1)
  display.set_pixel(2,2,9)
  sleep(1)
  display.set_pixel(2,2,0)
```

Quelle est l'action de la fonction `sleep` du module `time` ?

3.2 Boucles bornées

En Python, la boucle de « répétition » est introduite par le mot clé `for` dont la syntaxe est :

```
for i in range(n):
        bloc instructions # sera répétée n fois
        #la variable i prenant successivement les valeurs 0, 1, ..., n-1
```

La fonction range peut prendre 1, 2 ou 3 arguments.

- `i in range(fin)` : `i` prend les valeurs de 0 à `fin`−1
- `i in range(debut,fin)` : `i` prend les valeurs de `debut` à `fin`−1
- `i in range(debut, fin, pas)` : `i` prend les valeurs `debut`, `debut` +`pas`, `debut` +2`pas`, ... jusqu'à ce que `fin`−1 soit dépassé.

On souhaite obtenir la liste des entiers de 1 *à n. Anticiper ce que renvoie la fonction liste et la modifier pour qu'elle renvoie la liste souhaitée.*

```
def liste(n):
    L=[]
    for e in range(n):
        L.append(e)
    return(L)
```

Remarques

- `list(range(10))` renvoie la liste $[0, 1, 2, 3, \ldots, 9]$.
- La boucle `for` en Python permet de décrire n'importe quelle séquence.
  Par exemple, `for c in "Bonjour"` : permet de parcourir tous les caractères de la chaîne "Bonjour".

  Écrire dans un script et tester.

```
for lettre in "bonjour":
    print(lettre)

for n in [1,2,3,4,5,6,7]:
    print(n)
```

Exercice 6

Compléter la fonction `somme(n)` pour qu'elle renvoie la somme $1 + 2 + 3 + \ldots n$

```
def somme(n):
    S=0
    for x in range(n+1) :  # Pour x allant de 0 à n faire
        S=.....................
    return(S)
```

Exercice 7

Un fichier au format PBM est un fichier en ASCII qui se compose comme suit :

— les caractères P1, suivis d'un retour à la ligne ou d'un espace,
— la largeur de l'image, en base 10, suivi d'un retour à la ligne ou d'un espace,
— la hauteur de l'image, en base 10, suivi d'un retour à la ligne ou d'un espace,
— la liste des pixels, ligne par ligne, de haut en bas et de gauche à droite.

1. Exécuter le programme `exoImage.py`
2. Ouvrir le fichier `lignes.pbm` qui a été crée avec NotePad++ et avec LibreOffice Draw. Observer et faire le lien avec le code python.
3. Modifier le programme python afin que l'image obtenue (à enregistrer sous le nom `colonnes.bpm`) ne soit pas une alternance de lignes noires et blanches mais une alternance de colonnes noires et blanches.

Exemple avec le module `PIL` et deux boucles for imbriquées
On utilise les fonctions `seuil()` et `seuil2()` définies plus haut. Il faut qu'elles soient écrites dans le même script.
La methode `getpixel(a,b)` permet de récupérer la valeur associée au pixel situé colonne a et ligne b.

```
from PIL import Image

def seuil(x):
    ..........

def seuil2(x):
    ..........

img=Image.open("kingfisherNB.jpg") # image de David Mark de Pixabay
nbColonne,nbLigne=img.size # on récupère le nombre de colonnes et de lignes de l'image

img1 = Image.new(img.mode,img.size) #on crée une image de même mode (ici 'L') que img et de même taille
#tous les pixels sont à 0

for numeroLigne in range(nbLigne):
    for numeroColonne in range(nbColonne):
        pixel=img.getpixel((numeroColonne,numeroLigne))
        p=seuil(pixel)
        img1.putpixel((numeroColonne,numeroLigne), p)

img1.save("kingfisher1.jpg")
img.close() # tout fichier ouvert doit être fermé
```

Exercice 8
Modifier le script précédent pour obtenir une image "kingfisher2.jpg" en utilisant la fonction `seuil2()`

Exercice 9
L'image "kingfisherNB.jpg" a été obtenu à partir de l'image "kingfisher.jpg" en couleurs de David Mark de Pixabay en utilisant le logiciel Gimp.

1. En utilisant la fonction `enGris` écrite précédemment selon la "recommandation 709", écrire un script qui permet de transformer l'image "kingfisher.jpg" en niveaux de gris sous le nom "kingfisher3.jpg".
2. Ouvrer "kingfisherNB.jpg" et "kingfisher3.jpg" et comparer les deux images.