I Les fonctions

I.1 Structure

La syntaxe générale est la suivante :

`def nom`(*liste des paramètres*) :
blocs d'instructions ;

I.2 Au regard du TD 2

Utilisation d'une fonction qui détermine le nombre de pièces d'une catégorie et qui renvoie comme valeur la monnaie restante.

```
from math import*
pEC=float(input("prix : "))
sEC=float(input("somme donnée : "))
mEC=round(sEC-pEC,2)
print("La monnaie est : ",mEC)
mE=floor(mEC)
mC=round((mEC-mE)*100)
#
# fonction np qui détermine le nombre de pièces de y centimes
#
def np(y) :
    n=mC//y # quotient entier de mC par y
    print(n," pièces de ",y," centimes")
    return mC%y # reste dans la division euclidienne de mC par y
#
#la fonction renvoie le reste de monnaie une fois enlevé n fois y centimes
#
print("Décomposition des centimes de la monnaie : ")
L=(50,20,10,5,2,1)
for x in L :
    mC= np (x) # appel de la fonction et affectation de mC par la valeur calculée
```

I.3 Remarques importantes

Remarque 1 Les paramètres d'une fonction, ainsi qui les variables créées à l'intérieur d'une fonction n'existent que localement. Par exemple, dans ce programme :

```
x=1
def suivant(x) :
    x=x+1
    return x
y=suivant(x+1)
print(x,y)
```

Quelles valeurs de x et de y seront affichées ?

I.4 D'autres exemples

1. Écrire un programme qui calcule la longueur de l'hypothénuse dans un triangle rectangle. L'utilisateur aura saisi les deux côtés de l'angle droit. (nombres réels)
   Vous vous servirez d'une fonction nommée **carre** avec le paramètre **cote** qui affiche et renvoie la longueur de l'hypoténuse.

2. Écrire un programme qui calcule et affiche la surface et le volume d'une sphère à partir d'un rayon saisie par l'utilisateur. (valeur réelle)
   Vous vous servirez d'une fonction nommée **surfvolsphere** avec le paramètre **rayon** qui affiche et renvoie les valeurs des variables locales **vol** et **surf**.
   En utilisant une boucle **for**, faire afficher les volumes et surfaces pour des valeurs de rayon allant de 10 à 20.

3. Écrire une fonction **conversion(h,m,s)** qui prend en paramètre un horaire en heures, minutes et secondes puis retourne cet horaire converti en secondes.
   Écrire un programme qui demande deux horaires à un utilisateur (heures, minutes, secondes) et qui utilise la fonction **conversion** pour calculer le temps écoulé entre les deux (en secondes).

4. Une fonction peut renvoyer une valeur booléenne (VRAIE (True) ou Fausse (False)). Écrire un programme qui demande la saisie d'un caractère à un utilisateur, utiliser une fonction **controle(y)** qui renvoie la valeur vraie si le caractère saisi est 'o'.

   Cet exercice nécessite l'utilisation d'une structure conditionnelle.
   L'instruction `if` est la structure de test la plus basique. Elle permet d'exécuter une série d'instructions si une condition est vraie. La syntaxe de cette expression est la suivante :

   `if` *condition* :
   bloc d'instructions ;

   La condition est un booléen (type `bool`).

II La boucle FOR

II.1 Structure

L'instruction `for... in` permet de faire parcourir à une variable l'intégralité d'une structure de données (comme les caractères d'une chaine ou une liste d'entiers) et de répéter un nombre de fois connu à l'avance un bloc d'instructions au fil de ce parcours.

Extrait du *mémento* : Struct. itérative POUR

`for` *variable* `in` *liste* :

liste peut être
- un mot, dans ce cas la variable est automatiquement de type caractère.
- `range(n)` : liste des entiers compris entre 0 et n-1.
- `range(p,n)` : liste des entiers compris entre p et n-1.
- `range(p,n,i)` : liste des entiers compris entre 0 et n-1, avec un pas de i
- une liste donnée explicitement avec le format de liste.

II.2 Exemples

Tester ces deux exemples :

```
for k in range(1,10) :
    print(k)
```

```
for l in 'Bonjour' :
    print(l)
```

EXERCICE 1 Écrire une ligne de commande qui affiche tous les multiples de 7 entre 0 et 70.

EXERCICE 2

1. Rédiger un programme avec les lignes qui suivent. **En parallèle**, écrire les valeurs successivement prises par les variables `k` et `s` dans ce programme. Que calcule ce programme ?

```
s=0
for k in range(1,6) :
    s=s+k
```

`k`	`s`
-	0
1	
2	

2. Rédiger un programme `factorielle.py` qui calcule et affiche le nombre $n!$ où n est une variable de type `int`. Tester votre programme avec $n = 3, 4$ et 12.

EXERCICE 3 Écrire un programme `voyelles.py` qui affiche le nombre de voyelles d'un mot (écrit en minuscule, sans caractère accentué). Tester votre programme sur le mot `informatique`.
Indication : utiliser une variable `compteur` initialisée à zéro ; à l'aide d'une boucle `for` parcourir chaque caractère du mot et incrémenter le `compteur` si la lettre est une voyelle.

EXERCICE 4 Écrire un progamme `moyenneAlea.py` qui calcule et affiche la moyenne d'un nombre donné (disons 1000) d'entiers entre 1 et 10 choisis *au hasard* (à l'aide de la fonction `randrange`).