TP 03

Expression, variable, fonctions et structures algorithmiques

Activité 1 : Fonctions en Python

Q 1 :

```
def moy_extr(L):
    """Renvoie la moyenne du premier et du dernier élément de L"""
    return (L[0]+L[-1])/2
```

Q 2 :

```
def incr_sans_effet_de_bord(L):
    """Renvoie une nouvelle liste identique à L,
    sauf le premier terme incrémenté de 1"""
    c = L.copy()
    c[0] = c[0]+1
    return c

def incr_avec_effet_de_bord(L):
    """Incrémente le premier élément de L de 1"""
    L[0] = L[0]+1
    return None
```

Activité 2 : Boucles IF, FOR, WHILE

Q 3 : Indentation vraie.

```
x=0
y=5
t=False
if x>=1:
    t=True
if y<=6:
    t=True
```

Indentation fausse.

```
x=0
y=5
t=False
if x>=1:
    t=True
```

```
    if y<=6:
        t=True
```

Q 4 :

```
from random import randrange
n= randrange(100) # Un entier aléatoire entre 0 et 99

if n <= 10:
    print("Trop petit")
elif n >= 50:
    print("Trop grand")
else:
    print("Juste comme il faut")
```

Q 5 :

```
def inv(n):
    """Somme les inverses des n premiers entiers naturels non nuls"""
    s = 0
    for k in range(n):
        x = 1/(k+1)
        s = s+x
    return s
```

Activité 3 : Algorithme glouton – Problème du rendu de monnaie

Q 6 : On peut utiliser une structure de liste puisque cette structure en sera pas amenée à évoluer. On peut choisir de donner les valeurs de billets en centime afin de n'utiliser que des entiers et éviter des erreurs d'arrondis.

```
valeurs=[2000, 1000, 500, 200, 100, 50, 20, 10, 5, 2, 1]
```

Q 7 :

```
def rendre_monnaie(cout,somme_client,valeurs):
    """
    Retourne une liste nombre_billets donnant le nombre de billets ou pièce àrendre ↙
        selon le type de billet ou pièce
    Keywords arguments :
    cout : somme àpayer
    somme_client : argent donné par le client
    """
    nombre_billets=[0]*len(valeurs)
    montant_a_rendre = somme_client - cout
    k=0#Indice du billet dans la liste valeur
    while montant_a_rendre>0:
        nombre_billets[k]=montant_a_rendre//valeurs[k]
        montant_a_rendre-=nombre_billets[k]*valeurs[k]
        k+=1
    return nombre_billets
```

Q 8 :

```
def afficher_rendu_monnaie(cout,somme_client,valeurs):
    cout=100*cout
    somme_client=100*somme_client
    nombre_billets=rendre_monnaie(cout,somme_client,valeurs)
    for k in range(len(nombre_billets)):
        if valeurs[k]>200:
            print(str(int(nombre_billets[k]))+' : billet de '+str(int(valeurs[k]/100)↙
                )+' euros')
        elif valeurs[k]>=100:
```

```
            print(str(int(nombre_billets[k]))+' : pièce de '+str(int(valeurs[k]/100))+' euros')
        else:
            print(str(int(nombre_billets[k]))+' : pièce de '+str(int(valeurs[k]))+' centimes')
```

Activité 4 : Structures de boucles

Q 9 :

```
def nb_exos():
    """Nombre d'exercices réalisés en 240 min"""
    t=0
    i=0
    #0 exos réalisés en 0 minute
    while t<=240:
        #invariant de boucle i exos réalisés en t<=240 minutes
        i=i+1
        #invariant de boucle (i-1) exos réalisés en t<=240 minutes
        t=t+(i)**(1/2)
#       invariant de fin de boucle : i+1 exos réalisé en t minutes
#sortie de boucle : n<=240<t avec n le nombre de minutes pour faire i exos-1
    return i-1
```

Activité 5 : Fonctions plus avancées

Q 10 :

```
from math import sqrt

def racine(n):
    """sqrt(n), entier si n est un carré parfait"""
    s = sqrt(n)
    if s==int(s):
        return int(s)
    else:
        return s
```