À rendre mercredi 18 septembre 2024

Devoir Maison n°1 : Reprise en main de python – *Corrigé*

À retenir

- `range(a,b)` renvoie les entiers compris entre `a` INCLUS et `b` EXCLUS (c'est-à-dire `b-1` INCLUS).

  `range(0,4)` renvoie donc 0, 1, 2, 3.
- La phrase « pour i allant de 0 à n inclus » se traduit en `for i in range(0,n+1):`
- La phrase « pour i allant de 1 à n inclus » se traduit en `for i in range(1,n+1):`
- `for i in range(1,n):` se traduit en « pour i allant de 1 à n EXCLUS, soit de 1 à $n-1$ INCLUS »

Exercice n°1 Boucles

R1. Écrire la fonction `somme(n:int)->int` qui prend en entrée un entier naturel `n` renvoie la somme $\sum_{k=1}^{n} k^3$.

Écrire l'instruction pour calculer $\sum_{k=1}^{99} k^3$.

Solution:

```
def somme(n:int)->int:
    """
    renvoie la somme carrés des n premiers entiers
    """
    assert type(n)==int
    assert n>=1
    S=0
    for k in range(1,n+1): # ATTENTION LIMITE DU RANGE
        S=S+k**3
    return S
>>> somme(99)
```

R2. Écrire la fonction `produit(n:int)->int` qui revoie la valeur de $\prod_{k=1}^{n} k^3$.

Solution:

```
def produit(n):
    p=1 # ATTENTION, il faut partir de 1  (et non de 0, sinon on
   obtient 0...)
    for k in range(2,n+1):
        p=p*k**3
    return p
```

R3. Écrire la fonction `somme_double(n:int)->int` qui renvoie la valeur de la somme double $\sum_{i=0}^{n}\sum_{j=i}^{n} ij$.

Solution:

```
def somme_double(n):
    S=0
    for i in range(0,n+1): # ATTENTION limite du RANGE
        for j in range(i,n+1): # ATTENTION limiteS du RANGE
            S=S+i*j
    return S
```

R4. Écrire la fonction `suite(n:int)->float` qui renvoie le terme u_n de la suite définie par récurrence $u_0 = 10$ et $u_{n+1} = \frac{1}{2}u_n - 3$.

Solution:

```
# en itératif :
def suite(n):
    u=10 # u0, initialisation
    for i in range(1,n+1): # i sert de compteur, il reste n termes à calculer
        u=1/2*u-3
    return u
# en récursif :
def suite_rec(n):
    if n==0:
        return 10 # renvoie u0
    return 1/2*suite_rec(n-1)-3   # appel récursif : calcul du rang n à partir du rang n-1
```

R5. Écrire le script python (on n'attend pas de fonction) qui calcule la somme des entiers entre 0 et 10000 inclus qui sont divisibles par 3 ou par 5.

Solution:

```
S=0
for i in range(0,10001):
    if i%3==0 or i%5==0:
        S=S+i
# PAS DE return
```

Exercice n°2 Listes

On considère les deux listes suivantes contenant pour la liste `J_annee` les dates des JO, et pour la liste `M_annee` le nombre de médailles obtenu chaque jour par la France. Par exemple :

```
J_2024=['27/07/2024','28/07/2024',...,'11/08/2024']
M_2024=[4,4,...,2]
```

Les fonctions suivantes seront écrites de façon générale pour des listes `J` et/ou `M` du type des exemples ci-dessus.

R1. Écrire la fonction `somme(M:list)->int` prenant en entrée une liste `M` d'entiers et renvoyant la somme des éléments de la liste.

Solution:

```
def somme(M:list)->int:
    assert type(M)==list
    S=0
    for i in range(len(M)):
        S=S+M[i]
    return M
# ou
def somme(M:list)->float:
    S=0
    for x in M:
        S=S+x
    return S
```

R2. Écrire la ligne à saisir dans la console pour obtenir le nombre total de médailles obtenu par la France en 2024.

Solution:

```
somme(M_2024)
```

R3. Écrire la fonction `moyenne(M:list)->int` prenant en entrée une liste `M` d'entiers et renvoyant la moyenne des éléments de la liste.

Solution:

```
def moyenne(M:list)->float:
    return somme(M)/len(M)
```

R4. Écrire la fonction `maximum(M:list)->int` prenant en entrée une liste `M` d'entiers (ou flottants), et renvoyant sa valeur maximale.

Solution:

```
def maximum(M:list)->int:
    m=u[0]
    for i in range(1,len(M)):
        if M[i]>m: # nouveau maximum local
            m=M[i]
    return m
def maximum(M:list)->int:
    m=u[0]
    for x in u:
        if x>m: # nouveau maximum local
            m=x
    return m
```

R5. Écrire la fonction `indice_min(M:list)->int` prenant en entrée une liste `M` d'entiers (ou flottants), et renvoyant le premier indice correspondant à ce minimum.

Solution:

```
def indice_min(M:list)->int:
    imin=0
    for i in range(1,len(M)):
        if M[i]<M[imin]: # nouveau minimum local
            imin=i
    return imin
```

R6. Écrire la fonction `jour_min(J:list,M:list)->str` qui prend les listes `J` et `M` et renvoie le jour où le moins de médaille a été obtenu.

Solution:

```
# en utilisant  la fonction précédente
def jour_min(J,M):
    return J[indice_min(M)]
# sans utiliser la fonction précédente
def jour_min(J,M):
    i_min=0
    j_min=J[0]
    for i in range(1,len(M)):
        if M[i]<M[i_min]:
            i_min=i
            j_min=J[i_min]
    return j_min
```