TD 4 (feuille d'exo 3) - Boucles for et while

Informatique
MPSI-PCSI - Lycée Thiers

Exercice 1.

Soit la suite $(u_n)_n$ définie par :

$$u_0 = 0 \quad \forall\, n \in \mathbb{N},\ u_{n+1} = u_n^2 + 1$$

Ecrire une fonction `u(n)` prenant en paramètre un entier positif n et qui retourne la valeur numérique du terme u_n de rang n de la suite $(u_n)_n$.

Exercice 1. Corrigé

```
def u(n):
    u = 0
    for k in range(n):
        u = u**2+1
    return u
```

À retenir : méthode générale pour obtenir le terme u_n d'une suite (u_n) définie à l'aide d'une relation de récurrence à 1 pas.

Exercice 2.

Écrire une fonction `somme(n,k)` prenant en paramètres deux entiers positifs `n` et `k` et qui retourne la valeur de la somme :

$$\sum_{a=1}^{n} a^k$$

Pour $k = 1, 2, 3$ comparer le résultat obtenu pour certaines valeurs de n que l'on choisira avec ce qu'on obtient par les formules connues :

$$\sum_{a=1}^{n} a = \frac{n(n+1)}{2} \quad ; \quad \sum_{a=1}^{n} a^2 = \frac{n(n+1)(2n+1)}{6} \quad ; \quad \sum_{a=1}^{n} a^3 = \frac{n^2(n+1)^2}{4}$$

Exercice 2. Corrigé

```
def somme(n,k):
    S = 0
    for a in range(1,n+1):
        S += a**k
    return S
```

À retenir : méthode générale pour calculer une somme.

Exercice 3.

Exercice 1. On admet (le démontrer ?) que :

$$\lim_{n\mapsto\infty} \sum_{k=1}^{n} \frac{1}{k} = +\infty$$

Déterminer le plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} \geqslant 10$.

Exercice 3. Enoncé

Corrigé. Tout d'abord la suite $(u_n)_n$ définie par $u_n = \sum_{k=1}^{n} \frac{1}{k}$ est strictement croissante : en effet $\forall\, n \in \mathbb{N}, \quad u_{n+1} - u_n = \frac{1}{n+1} > 0$,

et $\boxed{\text{la suite } (u_n) \text{ diverge vers } +\infty}$. En effet puisque $x \mapsto \frac{1}{x}$ est strictement décroissante sur $\mathbb{R}_+^*$ et admet ln pour primitive, $\forall\, k \in \mathbb{N}^*$:

$$\int_k^{k+1} \frac{1}{x}\,dx = \ln(k+1) - \ln(k) \leqslant \frac{1}{k}((k+1) - k) = \frac{1}{k}$$

En sommant l'inégalité pour k variant de 1 à n :

$$\sum_{k=1}^{n} \int_k^{k+1} \frac{1}{x}\,dx = \int_1^{n+1} \frac{1}{x}\,dx = \boxed{\ln(n+1) \leqslant \sum_{k=1}^{n} \frac{1}{k}}$$

Puisque $\lim_{n\mapsto\infty} \ln(n+1) = +\infty$, alors $\lim_{n\mapsto\infty} u_n = +\infty$.

Exercice 3. Enoncé

```
S = 0
k = 0
while S < 10:
    k += 1
    S += 1/k
print(k)
```

On obtient $n = 12367$. Vérifions-le :

```
S1 = 0
for k in range(1,12367):
    S1 += 1/k
print("Somme jusqu'à 12366 :", S1)
print("Somme jusqu'à 12367 :", S1+1/12367)
```

```
Somme jusqu'à 12366 : 9.99996214792161
Somme jusqu'à 12367 : 10.000043008275778
```

Exercice 4.

Exercice 2. Ecrire une fonction `sommeDouble(n)` qui retourne la valeur numérique du résultat de la somme double :

$$\sum_{k=0}^{n}\sum_{i=0}^{k}(i+k)$$

Comparer pour certaines valeurs de n avec le résultat obtenu par le calcul.

Exercice 4. Corrigé

```
def sommeDouble(n):
    S = 0
    for k in range(n+1):
        for i in range(k+1):
            S += i+k
    return S
```

À retenir : méthode générale pour calculer une somme double.

On peut comparer ce que retourne la fonction pour quelques valeurs de n avec ce qu'on obtient par le calcul en exprimant cette somme en fonction de n ; le calcul donne $\frac{n(n+1)(n+2)}{2}$.

Exercice 5.

Exercice 3.

1. Ecrire une fonction `syracuse(u,n)` qui prend en paramètre deux entiers positifs `u` et `n` et qui retourne la liste des $n+1$ premiers termes $u_0, u_1, \ldots, u_n$ de la suite $(u_n)_n$ de premier terme u_0 =`u` et définie par la relation de récurrence :

$$u_0 = u \qquad \forall\, n \in \mathbb{N},\ u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

2. Vérifier sur quelques essais la conjecture de Syracuse : "Quel que soit son premier terme $u_0 \in \mathbb{N}^*$, la suite de Syracuse est périodique à partir d'un certain rang".

Exercice 5. Corrigé

1.

```
def syracuse(u,n):
    L = [u]
    for k in range(n):
        if u%2==0:
            u = u//2
        else:
            u = 3*u+1
        L.append(u)
    return u
```

2. Dès que 1 apparaît la suite devient périodique $1, 4, 2, 1, 4, 2, 1, \textit{etc}\ldots$:

```
In [1]: 1 in syracuse(7,100)
Out[1]: True
In[2]: 1 in syracuse(71,100)
Out[2]: False
In[2]: 1 in syracuse(71,1000)
Out[2]: True
```

Exercice 6.

Exercice 4.

1. Soit a une variable de type entier ; quelle expression renvoie son chiffre des unités ? Comment obtenir son chiffre des dizaines ? des centaines ?
2. Écrire une fonction `sommeCube(n)` qui prend en paramètre un entier positif `n` et qui renvoie la somme des cubes de ses chiffres.
3. Écrire un script qui permettra d'obtenir tous les entiers entre 1 et 10 000 qui sont égaux à la somme des cubes de leurs chiffres. Par exemple :

$$153 \ : \ 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$$

Exercice 6. Corrigé

1. réponses :

a%10

(a//10)%10

(a//100)%10

2.

```
def sommeCube(n):
    S = 0
    while n > 0:
        S += (n%10)**3
        n = n//10
    return S
```

Exercice 6. Corrigé

3) Le mieux est de constituer une liste.

```
L = []
for n in range(1001):
    if n == somCube(n):
        L.append(n)
print(L)
```

ou par compréhension de liste :

```
L = [n for n in range(1001) if n == somCube(n)]
```

On obtient : `[0, 1, 153, 370, 371, 407]`.