Informatique tronc commun
Devoir n° 01

12 décembre 2015

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Ce devoir est à réaliser seul, en utilisant Python 3.
3. Nous vous conseillons de commencez par créer un dossier au nom du DS dans le répertoire dédié à l'informatique de votre compte.
4. La documentation Python officielle est disponible à l'adresse suivante.
   file :///usr/share/doc/python3.4/html/index.html
5. Vous inscrirez vos réponses sur la feuille réponse fournie. Attention : lisez attentivement le paragraphe suivant.

Fonctionnement du devoir

Vos réponses dépendent d'un paramètre α, unique pour chaque étudiant, qui vous est donné en haut de votre fiche réponse. On considère la suite u, définie comme suit.

$$u_0 = \alpha \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = (15\,091 \times u_n)\ [64\,007].$$

Nous vous en proposons l'implémentation suivante.

```
def u(alpha,n):
    """u_n, u_0 = alpha"""
    x = alpha
    for i in range(n):
        x = (15091 * x) % 64007
    return x
```

Pour s'assurer que vous avez bien codé la suite u, en voici quelques valeurs.
u(100,0) = 100
u(1515,987) = 37099
u(496,10**4) = 53781

Questions de cours.

Q1 Donner le reste et le quotient de la division euclidienne de $u_2 \times u_3$ par u_4.

Q2 Donner le plus petit entier $k \in [\![0, 1000[\![$ pour lequel u_k est maximal.

Q3 Donner $\log_{10}(u_2)$ (écrire juste dix décimales après la virgule).

Exercices.

Q4 Donner le plus petit entier naturel n tel que $u_n = 100$.

Q5 Donner le nombre d'entiers inférieurs ou égaux à 10 000 parmi les u_k, pour $k \in [\![0, 1\,000[\![$.

On rappelle l'algorithme d'Euclide : si a, b sont des entiers naturels, on considère la suite r définie par :

— $r_0 = a$, $r_1 = b$;

— si $r_{n+1} \neq 0$, r_{n+2} est le reste de la division euclidienne de r_n par r_{n+1}.

Alors, si r_n est le premier terme de r non nul ($n \geq 1$), le pgcd de a et de b est r_{n-1}.

Q6 Donner le PGCD de u_6 et u_7.

On appelle moyenne élaguée d'un tableau T la moyenne du tableau que l'on obtient quand on a enlevé à T toutes les occurences de son maximum ainsi que de son minimum.

On note $x\%100$ le reste de la division euclidienne de x par 100.

Q7 Donner la moyenne élaguée du tableau constitué des $u_k\%100$, pour $k \in [\![1\,000, 2\,000[\![$.

Soit la suite S définie par

$$\forall n \in \mathbb{N},\ S_n = \sum_{k=0}^{n} \frac{1}{u_k^2}\ .$$

On rappelle que la fonction `floor` donne la partie entière d'un nombre et peut être utilisée après avoir tapé

```
from math import floor
```

Q8 Donner S_{987} (écrire juste dix décimales après la virgule).

Q9 Donner $S_{1\,000\,000} - \lfloor S_{1\,000\,000} \rfloor$ (écrire juste dix décimales après la virgule).

On considère la suite v définie par :

$$v_0 = u_0 \ \text{ et } \ \forall n \in \mathbb{N},\ v_{n+1} = \sum_{k=0}^{n} (n + 1 - k) v_k\ .$$

Q10 Donner v_{15}.

On rappelle qu'une tranche d'un tableau est une suite de valeurs consécutives extraites de ce tableau.

Si x est un entier, on note $x\%5$ le reste de la division eudlidienne de x par 5.

Q11 Donner la taille de la plus grande tranche constituée uniquement de zéros dans le tableau des $u_k\%5$, pour $k \in [\![2\,000, 3\,000[\![$.

Q12 Donner le nombre de nombres premiers dans le tableau des u_k, pour $k \in [\![3\,000, 4\,000[\![$.

L'exercice suivant est certainement le plus difficile du devoir. Il n'est à traiter que par ceux qui auront déjà résolu les autres exercices.

Dans une pyramide de nombres, en partant du sommet, et en se dirigeant vers le bas à chaque étape, on cherche à maximiser le total de la somme des nombres traversés.

<u>**2**</u>
4 <u>**5**</u>
1 7 <u>**8**</u>
2 3 1 <u>**6**</u>
9 4 6 <u>**5**</u> 2

Sur cet exemple, la somme totale maximale est 26, obtenue en parcourant les nombres soulignés.

On peut voir la pyramide comme un graphe, parcourir les 16 chemins, et choisir celui qui a le plus grand total. Quand la pyramide a n niveaux, il y a 2^{n-1} chemins, ce qui rend vite cet algorithme inexploitable.

Un algorithme récursif est aussi possible, mais alors certains calculs sont effectués plusieurs fois, et là encore l'algorithme est trop long pour des pyramides de taille conséquente.

La méthode la plus rapide est celle utilisant la *programmation dynamique*. L'idée est résumée dans la séquence suivante :

2	2	2	2	26
4 5	4 5	4 5	20 24	
1 7 8	1 7 8	12 16 19		
2 3 1 6	11 9 7 11			
9 4 6 5 2				

À vous de comprendre cette idée et d'écrire un programme calculant ce total maximum pour des pyramides que l'on définit de la manière suivante : chaque pyramide est donnée dans un tableau, dont les éléments sont les lignes de la pyramide, représentées elles-mêmes dans un tableau. Par exemple, la pyramide de l'exemple sera représentée dans le tableau `[ [ 2 ], [ 4, 5 ], [ 1, 7, 8 ], [ 2, 3, 1, 6 ], [ 9, 4, 6, 5, 2 ] ]`.

Vous pourrez utiliser la fonction suivante pour construire ces pyramides :

```
def pyramide(alpha,n):
    p = []
    x = alpha
    for i in range(n):
        l = []
        for j in range(i+1) :
            y = x % 10
            l.append(y)
            x = (15091 * x) % 64007
        p.append(l)
    return p
```

Enfin, on rappelle que si $x \in \mathbb{N}$, on note $x\%10$ le reste de la division euclidienne par 10.

Q13 Donner la somme maximale pour la pyramide ayant 20 lignes, définie par les $u_k\%10$ (lus de haut en bas, de gauche à droite, la première ligne est $[u_0\%10]$, la seconde $[u_1\%10$, $u_2\%10]$ *etc.*).

Q14 Donner la somme maximale pour la pyramide ayant 100 lignes, définie par les $u_k\%10$ (lus de haut en bas, de gauche à droite, la première ligne est $[u_0\%10]$, la seconde $[u_1\%10$, $u_2\%10]$ *etc.*).