Informatique 1ere année, CPBX, TD 6: Boucle while

Carole Blanc, Paul Dorbec

1 Lancer de dé

Nous allons utiliser des appel de valeurs aléatoires. Au début de votre fichier, incluez la définition de fonction suivante :

```
import random

def un_de():
    return int(random.uniform(1,7))
```

La fonction `lancerDe()` simule le lancer d'un dé. C'est à dire que l'appel de cette fonction retourne aléatoirement un entier entre 1 et 6. Vous remarquerez que cette fonction n'attend aucun argument [1].

Question 1.1 *Écrivez une fonction* `nbLancers()` *qui ne prend rien en entrée et retourne le nombre de lancers fait avant d'obtenir un lancer qui vaut 6. Si les dés renvoient 1, puis 5, 2, et 6, la fonction retournera la valeur 3.*

Question 1.2 *On lance maintenant deux dés à la fois. Écrivez une fonction* `double()` *qui ne prend rien en entrée et retourne le nombre de lancers* [2] *faits avant d'obtenir un double (lorsque les deux dés ont la même valeur).*

Question 1.3 *Écrivez une fonction* `moyenneTentativesDouble(n)` *qui calcule la moyenne du nombre de tentatives avant d'obtenir un double, prise sur un échantillon de* `n` *doubles obtenus. Par exemple si* `n=10`*, on lancera les dés pour obtenir 10 fois des doubles et on calculera la moyenne du nombres de tentatives sur ces 10 séquences.*

1. pour faire un appel et affecter le résultat, on peut utiliser simplement `d=lancerDe()`. Attention, chaque appel donne une nouvelle valeur.
2. un lancer = lancer les deux dés

Question 1.4 *Test de l'uniformité : écrivez une fonction* `uniformite(n)` *qui lance* `n` *fois le dé et affiche ensuite le nombre de tirage de chaque valeur de 1 à 6. On testera par exemple avec* `n=6000`.

2 Ecriture décimale

Question 2.1 *On considère la fonction suivante :*

```
def mystere(n):
    s = 0
    while n > 0 :
        s = s + n % 10
        n = n // 10
    return s
```

Question 2.2 *Quelle est la valeur de* `mystere(2705)`. *De façon générale, que calcule la fonction* `mystere` *?*

Question 2.3 *Écrire une fonction* `nombreDeChiffres(n)` *qui retourne le nombre de chiffres contenus dans le nombre entier* `n`. *Par exemple, la valeur de* `nombreDeChiffres(2705)` *est* 4.

Question 2.4 *Écrire une fonction* `plusGrandChiffre(n)` *qui retourne le plus grand chiffre contenu dans le nombre entier* `n`. *Par exemple, la valeur de* `plusGrandChiffre(2705)` *est* 7.

3 Racine carrée

Question 3.1 *On rappelle que la somme des n premiers impairs est égale au carré de n.*
Écrire à l'aide d'une boucle `while` *et exclusivement de l'opération d'addition une fonction calculant la racine carrée entière par défaut d'un nombre entier positif donné ; calculer le nombre d'additions et de comparaisons à effectuer.*

4 Logarithme

Question 4.1 *Écrire une fonction* `logarithme(a,n)` *qui retourne le plus petit entier k tel que $a^k > n$ (on supposera $a > 1$).* Note : *python sait calculer a^k (notation* `a**k`*), mais il existe une solution simple et (légèrement) plus efficace qui utilise seulement la multiplication.*

Comment modifier cette fonction pour qu'elle calcule le plus grand entier k tel que $a^k \leq n$ (c'est la définition habituelle du logarithme entier*) ?*

5 Pour les plus rapides :

Suite de Conway : La suite de Conway est l'objet d'une énigme bien connue. Le premier terme de la suite $u_0 = 1$. Le terme u_{n+1} de la suite s'obtient en indiquant les chiffres composant le terme u_n. Par exemple, u_0 est composé de "un 1", et donc u_1 vaut "11". Ainsi, on obtient :

$$u_0 = 1, \quad u_1 = 11, \quad u_2 = 21, \quad u_3 = 1211, \quad u_4 = 111221, \quad u_5 = 312211, \ldots$$

Question 5.1 *Écrivez une fonction* `suiteConway(n)` *qui retourne le terme u_n de la suite de Conway ainsi définie. On pourra soit définir u_n comme un entier ou comme la liste de ses chiffres, ce deuxième choix étant sans doute plus facile.*

Dessin d'une fractale On considère la suite complexe suivante :

$$\begin{cases} z_0 = c \\ z_{n+1} = z_n^2 + c \end{cases}$$

Le langage python permet de manipuler facilement les nombres complexes :

- Pour initialiser une variable complexe il suffit d'écrire : `c = complex(re,im)` où `re` désigne la partie réeelle de `c` et et `im` la partie imaginaire de `c`.
- Les opérateurs arithmétiques classiques (+, -, *, /) fonctionnent naturellement sur les complexes.
- La fonction `abs(c)` permet de récupérer le module de `c`.

Question 5.2 *Définir une fonction* `suiteComplexe(x, y, size, n)` *qui retourne la valeur de n-ième terme de la suite (z_n) avec*

$$c = (x + iy) * 3/size - (2 + 1.5i)$$

.

Question 5.3 *Définir une fonction* `suiteComplexeDiverge(x, y, size)` *qui retourne le plus petit entier n tel que l'on n'a plus la condition :* *"n<256 et le module de z_n est inférieur à 2"*.

Question 5.4 *Définir la fonction* `mandelbrot(size)` *qui retourne une image de taille $(size, size)$ telle que le niveau de gris du pixel de coordonnées (x, y) est déterminé par la valeur calculée par* `suiteComplexeDiverge(x,y, size)`.

Dessiner l'image retournée par `mandelbrot(256)`.

Question 5.5 *Definir la fonction* `mandelbrotCouleur(size)` *sur le même schéma que la fonction* `mandelbrot(size)` *mais ce coup si la couleur de chaque pixel est :*

$$((n * 8) \,\%\, 256,\ (n * 32) \,\%\, 256,\ (n * 64) \,\%\, 256)$$

où n est la valeur retournée par la fonction `suiteComplexeDiverge(x, y, size)`.