PYTHON 06 : Etude d'un programme complet en Python

I./ Génération de nombres pseudo-aléatoires :

Il peut être intéressant, notamment pour réaliser un jeu, de pouvoir obtenir des nombres aléatoires, c'est-à-dire tirés au hasard, comme par exemple si on veut simuler un lancer de dé. Le problème est qu'un ordinateur est un système totalement déterministe et qu'aucun algorithme de calcul ne permet de créer des nombres totalement aléatoires. Certaines techniques permettent cependant de réaliser des tirages de nombres qui semblent suffisamment aléatoires pour un usage courant : on parle alors de nombres pseudo-aléatoires.

Pour pouvoir utiliser ces nombres pseudo-aléatoires, il faut utiliser une bibliothèque Python qui s'appelle **`random`**.

On l'appelle en utilisant la commande **`import random`** en début de programme.

Cette bibliothèque ajoute un certain nombre de fonctions à Python. Deux vont nous intéresser particulièrement.

La première est : **`random.random()`** : cette fonction renvoie un nombre flottant pseudo-aléatoire compris entre 0 et 1.

Exemple :

Dans la console, taper :

```
>>> import random
>>> random.random()
0.9206931218941108
>>> random.random()
0.7901151056746373
>>> random.random()
0.31390447925316234
```

On obtient une série de valeurs flottantes comprises entre 0 et 1. Les valeurs obtenues seront certainement différentes que celles indiquées dans le polycop.

La deuxième est : **`random.randint(a, b)`** : cette fonction renvoie un nombre entier pseudo-aléatoire compris entre **`a`** et **`b`** inclus.

Exemple :

Dans la console, taper :

```
>>> import random
>>> random.randint(1,10)
4
>>> random.randint(1,10)
1
>>> random.randint(1,10)
9
>>> random.randint(1,10)
7
```

On obtient une série de valeurs entières comprises entre 1 et 10. Les valeurs obtenues seront certainement différentes que celles indiquées dans le polycop.

Applications :

1./ Beaucoup de jeux utilisent des dés cubiques à 6 faces. Réaliser un programme qui demande à l'utilisateur un nombre de tirages, compris entre 1 et 10. L'ordinateur lancera alors un dé à 6 faces le nombre de fois demandé et affichera les résultats successifs.

2./ Dans le jeu des petits chevaux, on tire deux dés cubiques à 6 faces pour déterminer le déplacement du cheval. Faire un programme qui tire deux dés à 6 faces, et donne la somme obtenue. Si la somme est égale à 12, le programme devra afficher en plus « Vous avez obtenu un double 6 ! ».

3./ Dans la plupart des jeux de rôle (JdR), on utilise plusieurs types de dés : les dés classiques cubiques à 6 faces (ou D6), mais aussi des dés tétraédriques à 4 faces (ou D4), octaédriques à 8 faces (ou D8), décaédriques à 10 faces (ou D10), dodécaédriques à 12 faces (ou D12), icosaédriques à 20 faces (ou D20). Faire un programme qui demande à l'utilisateur quel type de dés il veut lancer (D4, D6, D8, D10, D12 ou D20), puis qui demande le nombre de dés à lancer, puis qui affiche le résultat des lancers.

4./ Il est parfois utile, dans certains jeux de rôle ou certains jeux de stratégie, d'obtenir une valeur comprise entre 0 et 99 quand certaines caractéristiques sont données en pourcentage. Il faudrait alors utiliser un D100, ou dé à 100 faces. Ce genre de dé existe, mais n'est pas très pratique : on préfère dans ces cas-là utiliser deux D10, l'un donnant le chiffre des dizaines, l'autre le nombre des unités. Par exemple, les deux dés ci-contre donnent la valeur de « 09 ». Faire un programme qui demande à l'utilisateur d'indiquer combien de lancers il désire réaliser. Le programme devra alors simuler pour chaque lancer, le tirage de deux dés à 10 faces et calculer la valeur correspondante entre 00 et 99.

II./ Jeu du nombre mystérieux :

Taper le programme suivant dans l'éditeur Python :

```
import random
running = 1
nombre_essais = 0
nombre_mystere = random.randint(1, 100)
while running == 1:
    nombre_essais = nombre_essais + 1
    print("Ceci est votre essai numéro", nombre_essais)
    correct = 0
    while correct == 0:
        nombre_joueur = int(input("Entrez un nombre entre 1 et 100 : "))
        if nombre_joueur >= 1 and nombre_joueur <= 100:
            correct = 1
        else:
            print("Le nombre entré est incorrect ! Recommencez !")
    if nombre_joueur < nombre_mystere:
        print("Le nombre cherché est plus grand que", nombre_joueur)
    elif nombre_joueur > nombre_mystere:
        print("Le nombre cherché est plus petit que", nombre_joueur)
    else:
        running = 0
print("Vous avez trouvé le nombre mystérieux en", nombre_essais, "coups")
```

1./ Lancer le programme et analyser son fonctionnement à l'aide des polys précédents.

2./ Modifier le programme de façon à ce que le programme choisisse un nombre mystérieux entre 1 et 1000.

3./ Modifier le programme pour qu'à la fin d'une partie le programme demande si le joueur veut refaire une partie, et relance le jeu selon la réponse.

2026
Bruno PEYROT

Document en ligne : https://tinyurl.com/5n6b97b6