Exercice 1

script disponible
BT_mystere.py

Énoncé

On a défini une fonction *mystere* dans le script ci-dessous.

1. Justifier que l'instruction mystere(45) donne 3.
2. Que donne mystere(154) ?
Plus généralement, que représente mystere(n) pour n ?

```
1 def mystere(n):
2     assert(n >= 0)  # vérifie que n est positif ou nul
3     while n >= 7:
4         n = n-7
5     return n
```

Solution

1. n prend dans ce cas la valeur 45. La vérification faire en ligne 2 ne pose donc pas de problème et l'exécution continue.
Une boucle Tant que… est programmée aux lignes 3 et 4. On passe dans la boucle tant que n est supérieur ou égal à 7 et, à chaque fois, n prend la valeur n -7.
1er passage : la valeur de n est 45, supérieure à 7. On passe donc dans la boucle et n prend la valeur 45 – 7 = 38
2e passage : la valeur de n est 38, supérieure à 7. On passe donc dans la boucle et n prend la valeur 38 – 7 = 31
3e passage : la valeur de n est 31, supérieure à 7. On passe donc dans la boucle et n prend la valeur 31 – 7 = 24
4e passage : la valeur de n est 24, supérieure à 7. On passe dans la boucle et n prend la valeur 24 – 7 = 17
5e passage : la valeur de n est 17, supérieure à 7. On passe dans la boucle et n prend la valeur 17 – 7 = 10
6e passage : la valeur de n est 10, supérieure à 7. On passe dans la boucle et n prend la valeur 10 – 7 = 3
La valeur de n est strictement inférieure à 7, on sort donc de la boucle.
La valeur de n, soit 3 est alors renvoyée.

2. De même mystere(154) renvoie 0 (obtenue en enlevant 22 fois de suite 7 au nombre 154).
mystere(n) représente le nombre positif ou nul obtenu en enlevant le plus grand nombre de fois possible 7 à n, c'est-à-dire le reste dans la division euclidienne de n par 7.

Exercice 2 Lancer un dé jusqu'à obtenir un 6

script disponible
BT_exoresolu2.py

Énoncé

Écrire un algorithme pour simuler le lancer d'un dé cubique, bien équilibré, jusqu'à l'apparition d'un 6 et indiquer combien le nombre de lancers ont été effectués.

Solution

Analysons le problème posé. Il faut simuler le lancer d'un dé, et répéter ce lancer jusqu'à obtenir un 6 ou, ce qui revient au même, tant que l'on n'a pas obtenu de 6.
On ne connaît donc pas à l'avance le nombre de répétitions mais on connaît la condition d'arrêt : obtenir 6.
Nous allons donc utiliser une boucle "Tant que …" ou "Répéter .... jusqu'à".

On a besoin de connaître le nombre de lancers effectués donc on a besoin d'une variable qui compte le nombre de lancer effectués.

<u>Algorithme</u>

VARIABLES : *d* entier entre 1 et 6, *compt* entier

TRAITEMENT :
Affecter à *d* un entier aléatoire entre 1 et 6
Affecter 1 à *compt* *# 1 lancer a été simulé*
Tant que *d* ≠6 Faire
Affecter à *d* un entier aléatoire entre 1 et 6
Affecter *compt*+1 à *compt*
Fin Tant que

SORTIE : Afficher *compt*

<u>Programme en Python</u>

```
1  import random
2  #nécessaire pour disposer de randint()
3
4  d = random.randint(1, 6)
5  #on prévoit un compteur
6  compt = 1
7  while d != 6:
8      d = random.randint(1, 6)
9      compt = compt + 1
10 print(compt)
```