Ex 1 - Une urne contient 5 billes rouges, 11 billes vertes et 2 billes jaunes. On tire une bille au hasard et on note sa couleur.
Compléter l'algorithme ci-dessous pour qu'il permette de simuler cette expérience.

```
from random import randint
a=randint(1,…)
if a<=5:
    print("rouge")
elif a<=…:
    print("vert")
else:
    print("……")
```

Ex 2 - La fonction ci-dessous simule 50 lancers successifs d'une pièce de monnaie équilibrée et retourne le nombre de « pile » obtenus.

```
from random import randint
def lancers():
    p=0
    for i in range(50):
        d=randint(0,1)
        if d==1:
          p=p+1
    return p
```

1) Tester cette fonction.
2) Comment a-t-on codé l'événement « obtenir pile » et « obtenir face » ?
3) Comment modifier cette fonction pour qu'elle retourne la fréquence des « pile » obtenus ?
4) Modifier cette fonction pour qu'elle accepte en paramètre une valeur n, et qu'elle simule ensuite n lancers de dés puis qu'elle affiche la fréquence des « pile ».
5) Proposer une astuce permettant de remplacer les 3 instructions qui sont dans la boucle for par une seule instruction.

Ex 3 - On lance un dé équilibré jusqu'à ce que l'on obtienne le numéro 6 sur sa face supérieure.

1) Écrire une fonction qui simule cette expérience et donne en sortie le nombre de lancers de dé qui ont été nécessaires afin d'obtenir 6.
2) Écrire une fonction qui appelle la précédente pour simuler 1000 fois l'expérience et renvoie la moyenne des nombres de lancers qui ont été nécessaires pour obtenir le numéro 6. Exécuter plusieurs fois cette fonction : Que remarque-t-on ?

Ex 4 - Le lièvre et la tortue.
On considère une piste de course contenant six cases alignées. Un lièvre et une tortue font une course selon la règle suivante : On lance un dé à six faces, parfaitement équilibré. Si l'on obtient un « 6 », le lièvre gagne directement, sinon, la tortue avance d'une case. Pour que le lièvre gagne, il suffit donc que le « 6 » sorte avant que la tortue ait parcourue les six cases de la piste. Pour que la tortue gagne, le « 6 » ne doit jamais sortir au cours des six premiers lancers de dés. Le but de l'exercice est de déterminer qui, de la tortue ou du lièvre, a le plus de chance de gagner.

1) Compléter la fonction ci-dessous. Que permet-elle de simuler ? Que permet de compter la variable n ? Qui a gagné quand la fonction retourne 1 ?

```
from random import randint
def partie():
    n=…
    while n<6 and randint(1,6)!=…:
        n=n+1
    if n==6:
        return 1
    else:
        return 0
```

2) Créer une fonction qui simule 1000 parties et retourne le nombre de fois où la tortue gagne. Que préfériez-vous être : lièvre ou tortue ?
3) On cherche la probabilité qu'a la tortue de gagner. Combien de parties faut-il simuler pour obtenir une estimation de cette probabilité à ±0,01 près ?

Ex 5 - On s'intéresse à la probabilité d'obtenir un 1quand on lance deux dés et que l'on ne garde que le plus petit.

1) A l'aide d'un tableau, représenter toutes les possibilités et en déduire la probabilité cherchée.
2) Écrire un algorithme qui permette de simuler 1000 fois cette expérience et qui affiche à la fin la fréquence des 1. Cette fréquence est-elle la plupart du temps proche de la probabilité trouvée dans la question 1 ?
3) On décide désormais de lancer 3 dés tout en continuant à ne garder que le plus petit.
Modifier l'algorithme précédent en utilisant la fonction min(…, …, …) qui retourne la plus petite des valeurs passées en paramètre.
4) Donner une estimation de la probabilité d'avoir 1 en gardant le plus petit des 3 dés. Avec quelle précision ?
5) Albéric, Barnabé et César ont cherché à calculer cette probabilité.
Albéric a trouvé $\frac{3x5^2}{6^3}$, Barnabé $\frac{6^2+5\times11}{6^3}$ et César $\frac{2\times6^2+4\times6}{6^3}$.
L'un d'entre eux semble-t-il avoir raison ?

Ex 6 - Sur les cinq dernières voitures que j'ai achetées, deux avaient un défaut de construction. Pourtant le concessionnaire m'a assuré qu'il n'y avait que 1 % des voitures de cette marque qui avaient un défaut à l'achat !

1) Écrire une fonction qui simule un échantillon de 5 voitures et retourne le nombre de celles qui ont un défaut. (On peut coder par 0 une voiture qui n'a pas de défaut et par 1 une voiture qui a un défaut. Attention, le 1 doit tomber avec une probabilité de 0,01 !).
2) Écrire une nouvelle fonction qui simule 1000 échantillons de 5 voitures et affiche le nombre d'échantillons où il y a au moins deux voitures défectueuses. Êtes-vous « un peu », « beaucoup », ou « énormément » malchanceux ?