I Le module `random`

Le module `random` contient des fonctions permettant la simulation du hasard. On présente ici certaines de ces fonctions. On importe ce module en le renommant : `import random as rd`

1 La fonction `randint(a,b)`

a et b étant deux entiers tels que $a \leqslant b$, `rd.randint(a,b)` renvoie un entier de l'intervalle $[\![a, b]\!]$.

Il y a **équiprobabilité** sur $[\![a, b]\!]$: $\forall k \in [\![a, b]\!]$, $\mathbf{P}(\texttt{rd.randint(a,b)} = k) = \dfrac{1}{\text{card}([\![a, b]\!])} = \dfrac{1}{b - a + 1}$.

Exemples : `rd.randint(1,2)` renvoie 1 dans 50% des cas, et renvoie 2 dans 50% des cas.

`rd.randint(1,6)` renvoie n'importe quel entier de $\{1, 2, 3, 4, 5, 6\}$ avec la probabilité $\dfrac{1}{6}$.

Remarques : Ces exemples permettent de simuler le jeu de *pile ou face*, ou de simuler le lancer d'un dé.

2 La fonction `random()`

Cette fonction renvoie un flottant "aléatoire" de l'intervalle $[0, 1]$. Elle suit une loi **uniforme** sur $[0, 1]$:

$$\forall a < b \in [0, 1],\ \mathbf{P}(a < \texttt{rd.random()} < b) = b - a.$$

Exemples : Il y a 10% de chances pour que `rd.random()` renvoie un flottant compris entre $0,54$ et $0,64$.
Il y a 25% de chances pour que `rd.random()` renvoie un flottant compris entre $0,1$ et $0,35$.

Remarques : La fonction `random` porte le même nom que le module `random`.
La fonction `random` n'accepte aucun argument, on l'appelle par la syntaxe : `rd.random()`

À tester : Dans le compilateur, utiliser plusieurs fois la commande :

```
>>> rd.random()          Out : 0.5877422084879713
                         Out : 0.0264512505538965
                         Out : 0.64122769310229851
```

3 La fonction `choice(L)`

Cette fonction renvoie un élément choisi au hasard et équiprobablement parmi les éléments d'un objet itérable `L` (par exemple une liste, une chaîne de caractères, une matrice).

Exemples : `rd.choice("Bonjour")` renvoie une lettre de ce mot, chacune avec la probabilité $\dfrac{1}{7}$.

`rd.choice(range(10))` renvoie n'importe quel entier $0 \leqslant k \leqslant 9$ avec la probabilité $\dfrac{1}{10}$.

II Simulation d'une expérience aléatoire

1 Lancer d'un dé bien équilibré

La fonction suivante simule le lancer d'un dé cubique bien équilibré :

```
def de() : return rd.randint(1,6)
```

1. Écrire une fonction simulant le lancer d'un dé bien équilibré à 12 faces (dodécaèdrique).
2. Écrire une fonction simulant la somme de deux dés cubiques bien équilibrés.
3. Écrire une fonction simulant la somme de n dés cubiques bien équilibrés ($n \geqslant 1$).
4. Écrire une fonction simulant le maximum de n dés cubiques bien équilibrés ($n \geqslant 1$).

2 Estimation informatique d'une probabilité

Pour estimer une probabilité, on simule un grand nombre de fois l'expérience, et on calcule la fréquence de l'événement dont on cherche la probabilité.
Si le nombre de simulations est grand, cette fréquence converge vers la probabilité de l'événement.

Exemple : On cherche la probabilité d'obtenir 3 en sommant deux dés cubiques bien équilibrés.

```
def probaTrois(N) :
    compteur = 0
    for _ in range (N) :
        compteur += (rd.randint(1,6) + rd.randint(1,6) == 3)
    return compteur / N
```

Estimer informatiquement les probabilités des événements suivants :
(*les dés ou pièces considérés ne sont pas truqués*)

1. : « Obtenir 12 en jetant 5 dés cubiques. »
2. : « Obtenir au moins 15 en jetant 5 dés cubiques »
3. : « Obtenir au moins deux fois plus de 'pile' que de 'face' en jetant 12 fois une pièce. »
4. : « Obtenir un écart maximal d'au moins 900 entre 10 entiers choisis au hasard entre 1 et 1000. »
5. : « Obtenir au moins 3 'pile' consécutifs en jetant 6 fois une pièce. »
6. : « Deux personnes au moins dans une classe de 36 élèves ont la même date d'anniversaire. »

3 Estimation informatique d'une moyenne

À partir d'une expérience aléatoire, on peut définir une valeur (on parle de *variable aléatoire*) dépendante du résultat de l'expérience aléatoire.
Exemple : on jette deux dés cubiques bien équilibrés, et on s'intéresse au maximum des résultats obtenus. Ce maximum (variant entre 1 et 6) est une variable aléatoire.
On peut s'intéresser à la *valeur moyenne* de cette variable aléatoire : on simule un grand nombre de fois l'expérience aléatoire, et on calcule la moyenne observée lors de ces simulations.

```
def moyenneMaxi(N) :
    somme = 0
    for _ in range(N) :
        somme += max(rd.randint(1,6) , rd.randint(1,6))
    return somme / N
```

Déterminer une estimation informatique de la moyenne des variables aléatoires suivantes :

1. La somme obtenue lors du lancer de 6 dés cubiques.
2. Le maximum obtenu lors du lancer de 6 dés cubiques.
3. L'écart maximal obtenu lors du lancer de 6 dés cubiques.
4. Le nombre de tirages (avec remise) pour piocher la boule noire dans une urne contenant 10 boules blanches et une boule noire.
5. Le nombre de tirages (sans remise) pour piocher la boule noire dans une urne contenant 10 boules blanches et une boule noire.
6. La position (x : abscisse, y : ordonnée) d'un mobile après 100 déplacements dans $\mathbf{Z}^2$ en partant de l'origine $(0, 0)$ et en se déplaçant aléatoirement de ± 1 unité horizontalement ou verticalement.
7. La distance à l'origine de ce même mobile après 100 déplacements.

4 Simulation d'une expérience aléatoire évolutive

(a) On dispose de deux urnes. Au départ, l'urne A contient 1 boule blanche et 4 boules noires, l'urne B contient 3 boules blanches et 2 boules noires.
On effectue une succession de tirages, en commençant par l'urne A : la boule piochée est ajoutée à l'urne B, et le tirage suivant se fait dans l'urne B. La boule piochée est alors ajoutée à l'urne A et on recommence. On veut simuler la couleur de la boule piochée au $n^{\text{ème}}$ tirage :

```
def tirage(n) :
    L = [[0,1,1,1,1], [0,0,0,1,1]]   # Modélisation des urnes : 0 = blanc, 1 = noir
    urne = 0                         # 0 pour l'urne A, 1 pour l'urne B
    for _ in range(n) :
        boule = rd.choice(L[urne])   # On pioche une boule
        L[urne].pop(-boule)          # pop(0) enlève la 1er valeur, pop(-1) la dernière
        urne = 1-urne                # On change d'urne
        if boule == 0 : L[urne] = [0] + L[urne]
        else : L[urne].append(1)
    return boule
```

Pour tout $n \geqslant 1$, on note B_n l'événement : « on pioche une boule blanche au $n^{\text{ème}}$ tirage. »
Écrire une fonction d'argument $n \geqslant 1$ renvoyant une liste d'estimations des probabilités de $B_1, \ldots, B_n$.

(b) Une urne contient au départ 1 boule noire et 3 boules blanches. On répète l'expérience suivante :

1. On pioche au hasard une boule de l'urne,
2. si la boule piochée est blanche, on la remet dans l'urne et on ajoute une nouvelle boule blanche dans l'urne,
3. si la boule est noire, l'expérience prend fin et on note B le nombre de boules blanches dans l'urne à ce moment.

Écrire une fonction qui renvoie une simulation de B.

On pourra poser `urne = [0, 1, 1, 1]` *pour décrire l'urne au début de l'expérience, et utiliser la fonction* `choice` *pour réaliser un tirage.*

Peut-on estimer informatiquement le nombre moyen de boules blanches de l'urne à l'issue de cette expérience ?

5 Probabilité uniforme sur un intervalle réel $[a, b]$

La fonction `random()` renvoie un flottant aléatoire entre 0 et 1.
Pour obtenir un flottant aléatoire entre a et b ($a < b$), on utilise : `rd.random()*(b-a) + a`

1. Écrire une fonction simulant le résultat (0 = échec, 1 = succès) d'une épreuve où la probabilité de succès vaut $p \in]0, 1[$.
2. Un joueur joue n fois à un jeu. À chaque fois, la probabilité de gagner vaut $p \in]0, 1[$.
   Écrire une fonction simulant le nombre de fois où le joueur gagne.
   Estimer informatiquement le nombre moyen de fois où le joueur gagne.
3. Simuler une variable aléatoire réelle (flottante) X uniformément répartie entre 2 et 8.
   Estimer la moyenne de X.
4. Soit $p_1 \in]0, 1[$. Un joueur joue n fois à un jeu. La première fois, il a une probabilité p_1 de gagner.
   Chaque fois suivante, sa probabilité de gagner vaut $p_{n+1} = (p_n)^2$ s'il a gagné la fois précédente, et $p_{n+1} = 1 - p_n$ s'il a perdu la fois précédente.
   Écrire une fonction simulant sous forme d'une liste le résultat des n parties (0 = défaite, 1 = victoire).
   Donner une estimation du nombre moyen de parties gagnées lorsque $n = 1000$ et $p_1 = 0,15$.
5. Dans un jardin rectangulaire de dimensions $a \times b$, on plante aléatoirement n arbres, en choisissant pour chaque arbre son abscisse uniformément dans $[0, a]$, et son ordonnée dans $[0, b]$.
   On appelle D la variable aléatoire égale à la distance minimale entre deux de ces arbres.

   Écrire une fonction `distance(a,b,n)` prenant pour arguments les dimensions du terrain et le nombre d'arbres, et renvoyant une simulation de D.

   Estimer informatiquement la distance minimale moyenne lorsque $a = b = 10$ mètres et $n = 6$ arbres.