TPC² — T.C.d'I.

SIMULATIONS D'EXPÉRIENCES ALÉATOIRES

§1. Dés

Le programme ci-dessous simule le lancer d'un dé ordinaire.

```
from random import randint as EntAléa
def Dé() :
    return EntAléa(1, 6)
```

Exercice 1

1) Écrire un programme `DeuxDés()` qui simule le lancer de deux dés, dont on fait la somme.

2) Écrire le programme `TroisDés()` qui simule le lancer de trois dés, dont on fait la somme.

Exercice 2 — Compléter le programme ci-dessous, qui étant donné un programme `X()` renvoyant des valeurs aléatoires, réalise un grand nombre de tirages de la variable aléatoire X et renvoie la fréquence de la `Valeur`.

```
NB_TIRAGES = 10 ** 6
def FréquenceEmpirique(X, Valeur) :
    nb = ...
    for _ in range(NB_TIRAGES) :
        if ... :
            nb += ...
    return nb / NB_TIRAGES
```

Petit exemple d'utilisation.

```
>>> FréquenceEmpirique(Dé, 6)
0.166493
```

Exercice 3 — Écrire un programme `MoyenneEmpirique(X)` qui étant donné un programme simulant une variable aléatoire numérique, renvoie une estimation de son espérance, en calculant une moyenne empirique sur un grand nombre de tirages.

```
>>> MoyenneEmpirique(Dé)
3.498303
```

Exercice 4 — On lance trois dés équilibrés : un à six faces (numérotées de 1 à 6), un à huit faces (numérotées de 1 à 8) et un à douze faces (numérotées de 1 à 12).

1) Avec Python, estimer la probabilité d'obtenir 10 (lorsqu'on fait la somme).

2) Écrire un programme qui calcule la valeur *exacte* de la probabilité de faire 10.

§2. Urnes

Le programme ci-dessous renvoie un élément au hasard dans une liste, avec équiprobabilité.

```
def Choisir(L) :
    k = EntAléa(0, len(L) - 1)
    return L[k]
```


40°

Exercice 5 — Comment simuler un dé truqué qui donne à la face « 6 » une probabilité deux fois plus grande que toutes les autres faces ?

```
>>> FréquenceEmpirique(DéTruqué, 6)
0.285743
```


70°

Exercice 6 — Une urne contient quinze boules rouges, une boule jaune, une boule verte, une boule marron, une boule bleue, une boule rose et une boule noire.

1) Écrire un programme qui simule le tirage (avec remise) de deux boules dans cette urne. Ce programme renverra un couple.

2) Estimer avec Python la probabilité que les deux boules piochées soient de la même couleur.


60°

Exercice 7 — Une urne contient une boule rouge et une boule verte. On réalise le jeu suivant : on pioche une boule au hasard. Si elle rouge, on s'arrête. Si elle est verte, on la remet dans l'urne, on ajoute une boule verte, et on recommence.

1) Écrire un programme qui simule ce jeu, et qui renvoie le nombre total de tirages effectuées.

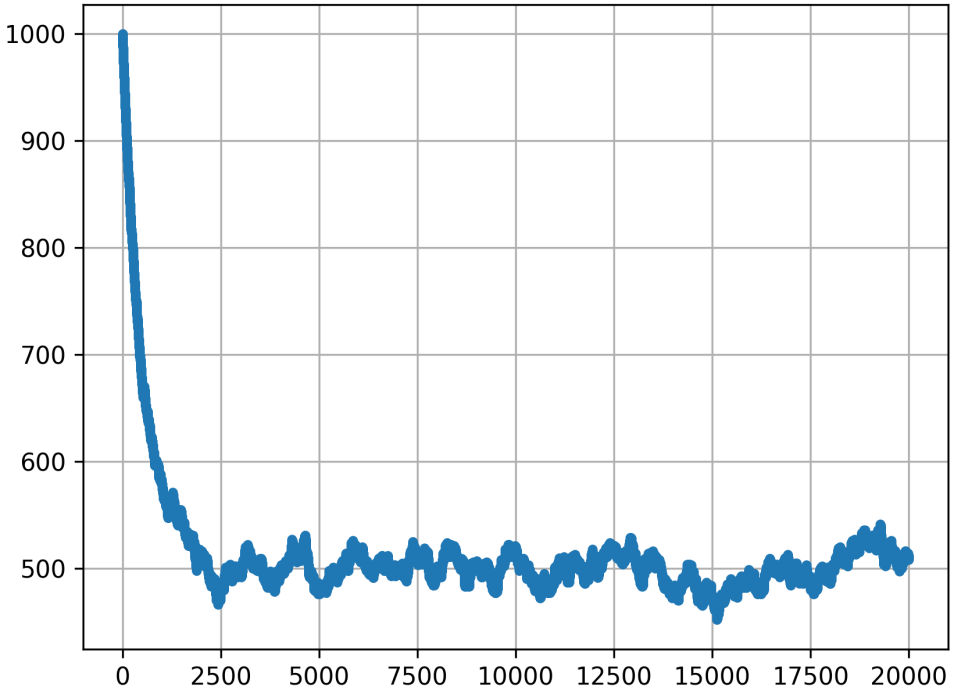
2) Quelle est la probabilité que le jeu s'arrête au premier tirage ? Et au deuxième ?

3) Estimer le nombre moyen de tirages effectués, lorsqu'on joue un grand nombre de parties (chaque partie commence évidemment avec une boule rouge et une boule verte dans l'urne).


70°

Exercice 8 — **Urnes d'Ehrenfest.** On dispose de deux urnes A et B, et d'un certain nombre de billes. Au début de l'expérience, on place toutes les billes dans l'urne A. À chaque étape, on choisit une bille au hasard, et on la change d'urne. Écrire un programme `Ehrenfest(NombreBilles, n)` qui simule n étapes de cette expérience, et qui trace le graphique représentant l'évolution du nombre de billes dans l'urne A.

```
>>> Ehrenfest(1000, 20000)
```



§3. Cartes

On donne le programme ci-dessous qui construit une liste représentant un jeu de cinquante-deux cartes. Les cartes sont représentées par des couples (h, c) avec h la hauteur de c la couleur.

```
def CréerJeu() :
    Cartes = []
    for h in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, "V", "D", "R"] :
        for c in ["♡", "♢", "♣", "♠"] :
            Cartes.append( (h, c) )
    return Cartes
CARTES = CréerJeu()
```

60°

Exercice 9 — Écrire un programme `Distribuer(`n`)` qui renvoie une main de n cartes (tirées sans remise).

```
>>> Distribuer(5)
[(10, '♠'), (3, '♢'), ('V', '♢'), (6, '♡'), (2, '♡')]
```

50°

Exercice 10

1) Écrire un programme `UnAs(Main)` qui prend en argument une liste de cartes et teste si elle contient au moins un as. Ce programme renverra un booléen.

2) Écrire un programme `DeuxAs(Main)` qui prend en argument une liste de cartes et teste si elle contient au moins deux as.

3) Estimer avec Python la probabilité d'obtenir (au moins) un as lorsqu'on prend deux cartes, puis la probabilité d'obtenir deux as (toujours lorsqu'on prend deux cartes).

§4. Points

On rappelle que le programme

```
from random import random
def Aléa(a, b) :
    return a + (b - a) * random()
```

renvoie un nombre réel choisi au hasard dans l'intervalle $[a\,;\,b[$, avec distribution uniforme.

20°

Exercice 11 — Écrire un programme `PointAléa(`$x_{\min}$`,` $x_{\max}$`,` $y_{\min}$`,` $y_{\max}$`)` qui renvoie un point tiré au hasard dans la région $[x_{\min}\,;\,x_{\max}[\times[y_{\min}\,;\,y_{\max}[$.

```
>>> PointAléa(−1, 1, −1, 1)
(0.2564315765715173, -0.9452228855290417)
```

50°

Exercice 12 — Dans le plan, rapporté à un repère orthonormé, on trace le cercle trigonométrique, ainsi que le carré $\mathscr{K} = [0\,;\,1] \times [0\,;\,1]$.

1) Quelle est l'aire du quart de disque « trigonométrique » contenu dans le carré $\mathscr{K}$?

2) Si l'on prend un point au hasard, avec distribution uniforme, dans $\mathscr{K}$, quelle est la probabilité qu'il soit à l'intérieur du cercle trigonométrique ?

3) Écrire un programme `PiSurQuatre(`n`)` qui réalise la simulation suivante : tirer n points dans le carré $\mathscr{K}$, et calculer la proportion de ceux qui sont à l'intérieur du cercle trigonométrique.

```
>>> PiSurQuatre(10 ** 6)
0.785687
```


40°

Exercice 13 — Écrire un programme `UnitaireAléa()` qui renvoie un point choisi au hasard sur le cercle trigonométrique (donc un couple (x, y) avec $x^2 + y^2 = 1$), avec distribution uniforme.

```
>>> UnitaireAléa()
(-0.7012966888045598, -0.7128695212111124)
```