

Feuille_info_11 : simulations.

Dans ce TD les fonctions Python seront supposées précédées de la ligne suivante qu'il ne sera pas nécessaire de rappeler par la suite.

```
import random as rd
```

Les parties 1 et 2 sont indépendantes. La partie 3 utilise les fonctions des questions 1) des Parties 1 et 2.

Dans un premier temps vous pouvez passer les questions 2) des parties 1 et 2.

Partie 1

- 1) Ecrire une fonction qui permet de simuler une variable aléatoire X suivant une loi discrète à support fini :

valeurs	x_0	x_1	$\cdots$	x_{n-2}	x_{n-1}
probabilités	p_0	p_1	$\cdots$	p_{n-2}	p_{n-1}

On pourra compléter le programme suivant :

```
def simulation(valeurs, probabilités):
    """ Simule la loi de probabilité qui à valeurs[k] associe probabilités[k],
        la somme des valeurs de la liste probabilités doit être égale à 1 """
    r = rd.random()

    return valeurs[i]
```

- 2) Faire la question **Q. 15** du sujet **MMI 2025**.

Partie 2

L'objectif de cet exercice est de récupérer les résultats d'un grand nombre de simulations dans un tableau.

Ici on suppose qu'une fonction aléatoire `simul_X()` est en mémoire et peut être utilisée dans tous les fonctions sans être mis en argument. Cette fonction `simul_X()` simule la réalisation d'une variable X .

- 1) Ici on suppose que X prend des valeurs entières appartenant à $\llbracket 0, n-1 \rrbracket$.
- a) Ecrire une fonction `simulation(N, n)` qui réalise une série de N simulations et qui renvoie la liste des fréquences d'apparition des valeurs de $\llbracket 0, n-1 \rrbracket$.
- b) Tester votre fonction avec la fonction :

```
def simul_X():
    return rd.randint(1,8)
```

- 2) Ici on suppose que X prend des valeurs quelconques

- a) Ecrire une fonction `simulation_2(N)` qui réalise une série de N simulations et qui renvoie le dictionnaire des fréquences d'apparition des valeurs prises par X .
- b) Tester votre fonction avec la fonction :¹

```
def simul_X():
    x = rd.sample(["Nokomie", "Aïcha", "Allan", "Lola", "Thomas", "Aïcha"], 2)
    return tuple(x)
```

- c) Expliquer comment la fonction précédente permet de tester, de manière empirique, si une variable aléatoire est discrète ou non.

1. On transforme la liste en tuple car les tuples, contrairement aux listes, peuvent être utilisés comme clés d'un dictionnaire.

Partie 3

On utilise ici les fonctions des questions 1) des **Parties 1 et 2**.

1) On considère la loi suivante :

$$X(\Omega) = \{0, 1, 2, 3\}, \quad (p_0, p_1, p_2, p_3) = (0.1, 0.3, 0.4, 0.2).$$

Écrire en Python les lignes permettant de définir les listes `valeurs` et `probabilites`, puis d'utiliser la fonction `simulation(valeurs, probabilites)` pour simuler une valeur de X .

2) On souhaite maintenant comparer la loi théorique et la loi empirique obtenue par simulation.

- a) Utiliser la fonction `simulation(N, n)` de la **Partie 2** pour estimer les fréquences d'apparition des valeurs de X pour $N = 10\,000$ simulations.
- b) À l'aide du module `matplotlib.pyplot`, représenter dans un même graphique :
 - un diagramme en barres représentant les probabilités théoriques (p_0, p_1, p_2, p_3) ;
 - un diagramme en barres représentant les fréquences empiriques obtenues lors des N simulations.
- c) Commenter brièvement : que constate-t-on lorsque N est grand ? Que se passe-t-il si l'on diminue fortement le nombre de simulations ?

3) Refaire l'étude de la question 2) pour une fonction `simul_X()` qui simule la loi binomiale de paramètres $\left(10, \frac{1}{2}\right)$.

