Dans ce TP on simule des cas aléatoires de façon à visualiser les résultats obtenus en cours.

1 Loi des fréquences avec le tableur

Une entreprise fabrique des cellules solaires. 8 % des cellules a une efficacité inférieure aux normes en vigueur. On assemble 100 de ces cellules pour former un panneau.

Nous allons simuler cette situation dans un **tableur**.

(1) Ouvrez le tableur et reproduisez ce que vous voyez ci-dessous.

	A	B	C	D
1	panneau 1	panneau 2	panneau 3	
2	=ALEA()<=8/100			
3				
4				

Notez la formule en A2. Cette formule renvoie VRAI dans 8 % des cas et simule donc l'éventualité d'une cellule défectueuse.

(2) Étendez la formule A2 vers le bas en nombre adéquate pour représenter un panneau.

(3) En dessous de cette colonne, on souhaite calculez le nombre de panneaux défectueux. Il suffit d'effectuer une somme. Les résultats VRAI seront comptés comme 1 et les FAUX comme 0 si bien que las SOMME donne le nombre de VRAI.

Écrivez en bas de la colonne A, la formule donnant le nombre de cellules défectueuses.

(4) En-dessous encore, écrivez la formule donnant la **fréquence** de cellules défectueuses dans un panneau.

Vous pouvez ajuster l'affichage pour l'avoir en %.

(5) Étendez toute la colonne vers la droite de façon à simuler de la même façon 200 panneaux.

Cela peut prendre quelques instants...

Vous pouvez déjà constater que les valeurs ne fluctuent pas en dehors d'un certain intervalle.

(6) Calculez (en utilisant le tableur) la moyenne et l'écart-type des fréquences.

On aimerait pouvoir tracer un diagramme mais c'est tout de suite très fastidieux avec le tableur. On va donc passer à un outil un peu plus efficace : Python.

2 Loi des fréquences avec le tableur

Avec EduPython, ouvrez un nouveau fichier.

Enregistrez-le tout de suite au nom loi_frequences.py.

(1) Commencez à importer les modules utiles :

```
1  import matplotlib.pyplot as plt # graphique
2  from random import random       # simulation aléa
3  import numpy as np              # fonctions mathématiques
```

(2) Pour simuler un panneau de n cellules, ayant chacune une probabilité p d'être défectueuse, et renvoyer la fréquence de cellule défectueuse, on écrira la fonction :

```
5  def frequence(n, p):
6      nombre_defectueuse = 0
7      for i in range(n):
8          if random() <= p:
9              nombre_defectueuse += 1
10     return nombre_defectueuse/n
```

Ce programme est simple. Essayez de le comprendre en même temps que vous le recopiez.

(3) Exécutez. Vous pouvez tester en console en écrivant la commande frequence(?,?) où il faut remplacer les ? par les valeurs adéquates pour notre problème.

Vous pouvez répéter l'opération pour voir comment évolue la fréquence.

(4) Pour simuler N panneaux, on utilisera la fonction suivante :

```python
def serie(N, n, p):
    resultats = []
    for i in range(N):
        f = frequence(n, p)
        resultats.append(f)
    return resultats
```

Recopiez et exécutez avant des tester en console. Vous pouvez prendre N = 200 pour reproduire la situation du tableur.

Notez que vous pourriez prendre sans problème N = 10000 ce qui aurait été très compliqué avec le tableur.

(5)

(6) Maintenant le tracé du graphique, avec Python et matplotlib, est très simple.

```python
# calcul pour 200 panneaux de 100 cellules
result = serie(200, 100, 8/100)
bornes = [i/100 for i in range(20)]
plt.hist(result, bins=bornes, density = True, edgecolor='k', linewidth=3)
plt.xticks(bornes, rotation=60)
plt.title('répartition des fréquences')
plt.show()
```

Exécutez et visualisez.

(7) En console, après avoir fermé la fenêtre du graphique, result calculez ci-dessus reste accessible. Vous pouvez alors exécuter les commandes, la première pour la moyenne, la seconde pour l'écart-type :

```
>>> np.mean(result)
>>> np.std(result)
```

(8) Vérifiez que l'on a bien $\sigma \approx \sqrt{\frac{p\cdot(1-p)}{n}}$

3 Loi des moyennes

On n'est pas très surpris de voir que la situation précédente tende vers une loi normale : il s'agit d'une loi binomiale et on sait déjà que la loi binomiale tend vers la loi normale.

Voyons le cas plus surprenant d'un dé. Prenons un dé dont les chances des faces est donné par [12, 13, 10, 20, 17, 28] ce qui signifie que le 6 à 28/100 chances d'apparaître alors que le 3 n'a que 10/100 chances.

Simulons un tel dé :

```python
# fichier loi_moyennes.py
import matplotlib.pyplot as plt
from random import randrange
import numpy as np

def simul_lancer_1():
    poids = [12, 13, 10, 20, 17, 28]
    a = randrange(100)
    t = 0
    for i in range(6):
        t += poids[i]
        if a < t:
            return i+1
```

(1) Reproduisez ce script, exécutez-le et testez en console.

(2) Reprenez une fonction similaire à serie dans le script précédent pour totaliser un grand nombre de lancer.

(3) faites un histogramme pour obtenir la répartition. Pour l'instant, on constate seulement que le dé est pipé et que le 6 est plus fréquent.

(4) Écrivez une fonction qui effectue K lancer de dés et fait le total (revient au même que faire la moyenne est plus parlant)

(5) Simulez N = 1000 lancers de K = 10 dés. Tracez le graphique pour constater qu'on a maintenant une loi normale.