

L'objectif de ce travail est d'introduire le chapitre sur les **Lois de Densité**.

Cécile se rend en métro de la station Gorge de Loup jusqu'à la station Cuire : Gorge → Bellecour (ligne D) + Bellecour → Hotel de Ville (ligne A) + Hotel de Ville → Cuire (Ligne C). Elle n'a pas consulté les horaires. A l'arrivée à Gorge de Loup et aux changements à Bellecour et Hotel de ville, elle a un temps d'attente aléatoire uniformément réparti entre 0 et 5 mn. A l'arrivée à Cuire, le temps total de son parcours est égal à T si elle a pris les 3 métros sans aucune attente. Dans le cas où Cécile attend 5 mn au départ et ensuite à chacun des 2 changements, le temps du parcours sera de T + 15 mn. On note X la variable aléatoire donnant en minutes, le temps d'attente cumulé : $0 \leq X \leq 15$. Quelle probabilité $p(2 < X < 2,5)$ a Cécile d'avoir un temps d'attente cumulé compris entre 2 mn et 2,5 mn ?



On se propose dans ce Tp de créer des fonctions pythons qui permettent de simuler cette situation un grand nombre de fois, en utilisant la fonction `random()`. Les codes pythons à créer seront écrits dans un fichier nommé `monNom_loisDensite.py` qui sera à uploader dans le répertoire *Devoir* en fin de séance.

Les premières lignes de ce fichier seront les suivantes. Elles permettent d'importer les librairies `random` et `matplotlib`.

```
from random import random
import matplotlib.pyplot as plt
```

1- Fonction `attente()` :

Ecrire le script de la fonction `attente()`. Cette fonction n'a pas de paramètre. Elle renvoie une liste de nombres aléatoires réels compris entre 0 et 5.

Par exemple l'exécution de cette fonction dans la console pourrait renvoyer :

```
>>> attente()
[1.3405078066351868, 0.5144854541851618, 1.8129063991440963]
```

2- Fonction `experience()` :

Ecrire le script de la fonction `experience(n)`. Cette fonction a comme paramètre un nombre entier n . Elle renvoie une liste de n listes générées avec la fonction `attente()` créée précédemment.

Par exemple l'exécution de cette fonction dans la console pourrait renvoyer :

```
>>> experience(5)
[[0.7771552463118053, 2.0496671862274356, 4.45565660366028],
 [3.6180348125817483, 4.637841643075493, 2.994106492110047],
 [0.35228019714051984, 2.199215200889231, 3.3172586433138873],
 [3.0272468751569663, 3.615950962410532, 0.7161957101877708],
 [3.6180448824694897, 1.7120289673303417, 0.2819391691790807]]
```

Cette liste renvoyée contient ici 5 simulations de trajets Gorge → Bellecour (ligne D) + Bellecour → Hotel de Ville (ligne A) + Hotel de Ville → Cuire (Ligne C).

3- Fonction `cumul()` :

Ecrire le script de la fonction `cumul(n)` . Cette fonction a comme paramètre un nombre entier n . Elle appelle la fonction `experience(n)` créée précédemment. Le début de son script est :

```
def cumul(n):  
    t = [0 for i in range(15)]  
    liste = experience(n)
```

Une liste `t` de 15 valeurs est initialisée à sa création à `[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]`. Une liste de simulation est créée en appelant la fonction `experience()`. Compléter le script afin que cette fonction renvoie la liste `t` qui contiendra :

- pour `t[0]`, le nombre de trajets parmi les n simulés dans liste, pour lesquels **la somme** des 3 temps est comprise entre 0 et 1 mn
- pour `t[1]`, le nombre de trajets parmi les n simulés dans liste, pour lesquels **la somme** des 3 temps est comprise entre 1 et 2 mn
- etc
- pour `t[14]`, le nombre de trajets parmi les n simulés dans liste, pour lesquels **la somme** des 3 temps est comprise entre 14 et 15 mn
-

Par exemple l'exécution de cette fonction dans la console pourrait renvoyer :

```
>>> cumul(50)  
[0, 0, 0, 2, 4, 10, 4, 10, 9, 3, 4, 3, 1, 0, 0]
```

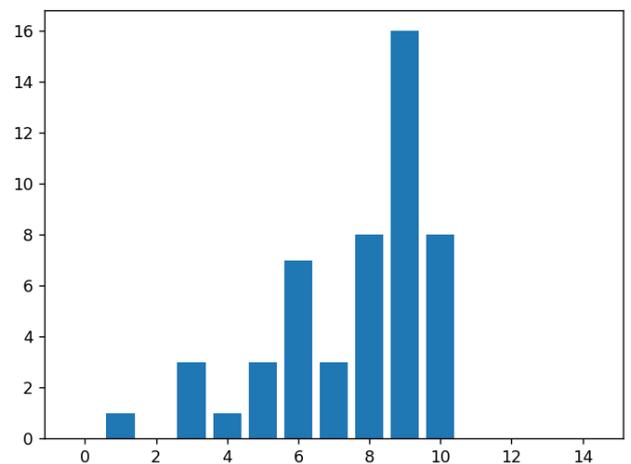
4- Fonction `trace()` :

Ecrire le script de la fonction `trace(n)` . Cette fonction a comme paramètre un nombre entier n . Elle affiche le diagramme barre relatifs à la liste générée avec la fonction `cumul(n)` écrite précédemment.

```
def trace(n):  
    names = [i for i in range(15)]  
    values = cumul(n)  
    plt.bar(names, values)  
    plt.show()
```

Par exemple l'exécution de cette fonction dans la console pourrait renvoyer :

```
>>> trace(50)
```



5- Conclusion :

⇒ Utiliser les fonctions créées précédemment pour simuler 100 000 trajets Gorge → Bellecour (ligne D) + Bellecour → Hotel de Ville (ligne A) + Hotel de Ville → Cuire (Ligne C).

⇒ Copier le diagramme barre obtenu dans un fichier nommé `monNom_loisDensite.doc`, à uploader dans le répertoire `Devoir` .

⇒ Répondre à cette question : « Comment expliquez-vous la forme gaussienne du diagramme barre ? »