TP Informatique

DECOUVERTE DES LOIS DE DENSITE

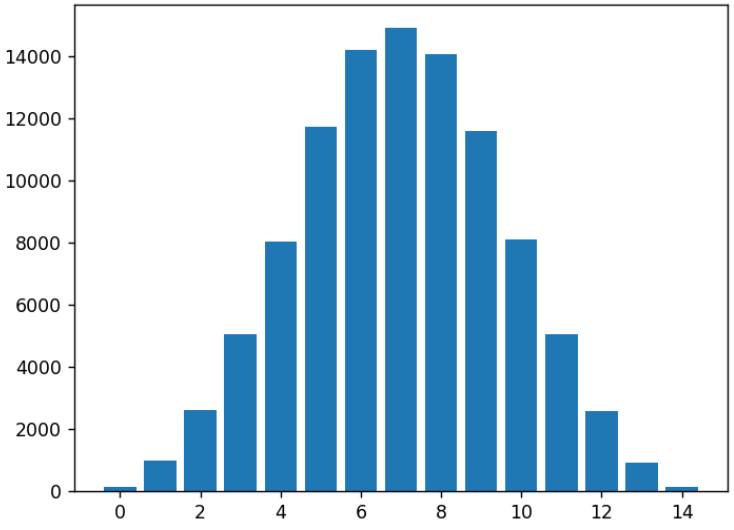
L'objectif de ce travail est de montrer que l'expérience aléatoire étudiée précédemment et encore décrite ci-dessous, peut se modéliser en utilisant une fonction de densité

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

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 ?

Les codes pythons à créer seront écrits dans le fichier nommé *monNom_loisDensite.py* qui a déjà été réalisé dans la séance précédente. Ce fichier comprend normalement les scripts des fonctions *attente()*, *experience()*, *cumul()* et *trace()*. L'exécution de `>>> trace(100000)` permet d'obtenir le diagramme ci-contre qui donne la répartition des 100000 temps d'attente dans les tranches [0 – 1 mn] , [1 – 2 mn] , [2 – 3 mn] , ……., [14 – 15 mn] . L'objectif de ce travail sera de sortir le même diagramme avec en plus la courbe de la fonction de densité

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$



Pour partir d'un code « propre », on donne ci-dessous le scripts des fonctions *attente()*, *experience()*, *cumul()* et *trace()* :

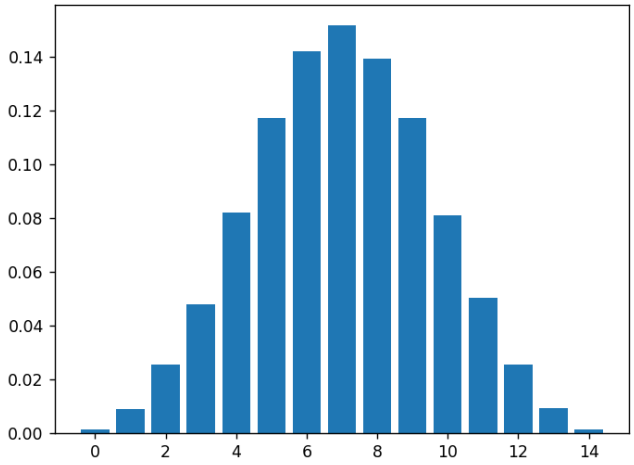
```python
from random import random
import matplotlib.pyplot as plt
from math import sqrt,exp,pi

def attente() :
    liste = []
    for i in range(3) :
        t = 5*random()
        liste.append(t)
    return liste

def experience(n) :
    liste = []
    for i in range(n) :
        liste.append(attente())
    return liste

def cumul(n):
    t = [0 for i in range(15)]
    liste = experience(n)
    for i in range(n) :
        s = liste[i][0] + liste[i][1] + liste[i][2]
        s = int(s)
        t[s] = t[s] + 1
    return t
```

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

1- Modification de la fonction *cumul()* :

Pour pouvoir comparer les résultats de la simulation avec ceux obtenus avec la fonction de densité, il nous faut dans un premier temps afficher un diagramme barre non pas de nombres, mais de fréquences.

Il s'agit de modifier légèrement la fonction *cumul()* afin de pouvoir obtenir après exécution de `trace(100000)` Le diagramme suivant :



⇨ Modifier le script de la fonction *cumul()* en divisant simplement les valeurs de la liste t par n :

```
def cumul(n):
    t = [0 for i in range(15)]
    liste = experience(n)
    moy,sig = moySig(liste)
    for i in range(n) :
        s = liste[i][0] + liste[i][1] + liste[i][2]
        s = int(s)
        t[s] = t[s] + 1/n
    return t,moy,sig
```

2- Création de la fonction *moySig()* :

Ecrire le script de la fonction *moySig()* . Cette fonction a comme paramètre une liste de liste du type de celle renvoyée par la fonction *experience()*. Elle renvoie les valeurs de la moyenne et de l'écart-type des cumuls de temps présent dans la liste en paramètre.

L'exécution suivante permet de mieux comprendre :

⇨ Ecrire le script de la fonction *moySig()* .

```
>>> liste = experience(5)

>>> liste
[[3.1805474060219607, 1.7833171943960024, 4.484164251963
323], [0.7462886631826043, 3.45830687149569, 2.969405975
4477644], [2.7818465430217323, 4.505697413864159, 2.1990
972938716675], [1.1886143227158075, 3.185891168215458, 4
.239626184671562], [1.1772001975801656, 3.32613806323183
43, 4.44545919358881]]

>>> moySig(liste)
(8.734320148653707, 0.8450333948334663)
```

```
def moySig(liste):
    s = 0
    for i in range(len(liste)) :
        s = s + liste[i][0] + liste[i][1] + liste[i][2]
    moy = s / len(liste)
    s = 0
    for i in range(len(liste)) :
        s = s + (liste[i][0] + liste[i][1] + liste[i][2]-moy)**2
    v = s / len(liste)
    sig = sqrt(v)
    return moy,sig
```

CORRIGE

3- Modification de la fonction *cumul()* :

⇨ Modifier la fonction *cumul()* pour y appeler la fonction *moySig()* mise au point dans le paragraphe précédent. Cette fonction *cumul()* devra renvoyer à présent la liste t et la moyenne et l'écart-type.

Par exemple :

```
>>> cumul(10)
([0, 0, 0, 0.1, 0.1, 0.2, 0.1, 0.1, 0.2, 0.1, 0, 0.1, 0, 0, 0], 7.145323851545657, 2.2848796360439185)
```

```
def cumul(n):
    t = [0 for i in range(15)]
    liste = experience(n)
    moy,sig = moySig(liste)
    for i in range(n) :
        s = liste[i][0] + liste[i][1] + liste[i][2]
        s = int(s)
        t[s] = t[s] + 1/n
    return t,moy,sig
```

4- Création de la fonction *fDensite()* :

Ecrire le script de la fonction *fDensite ()* . Cette fonction a comme paramètre 2 nombres $moy = \mu$ et $sig = \sigma$ et une liste de nombres $[t1, t2, \dots]$. Elle renvoie une liste contenant les images $[f(t1), f(t2), \dots]$ par la fonction $f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-\mu}{\sigma})^2}$.

On donne le test suivant pour valider son script :

```
>>> fDensite(7.5,2.0,[0,1,2])
[0.0004363413475228801, 0.0022159242059690038, 0.00876415024678427]
```

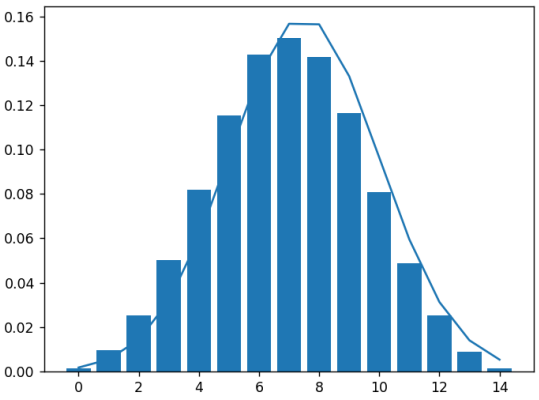
```
def fDensite(moy,sig,temps) :
    listeF = []
    for t in temps :
        t = t + 0.5
        y = 1/(sig*sqrt(2*pi))*exp(-0.5*((t-moy)/sig)**2)
        listeF.append(y)
    return listeF
```

CORRIGE

5- Modification de la fonction *trace()* :

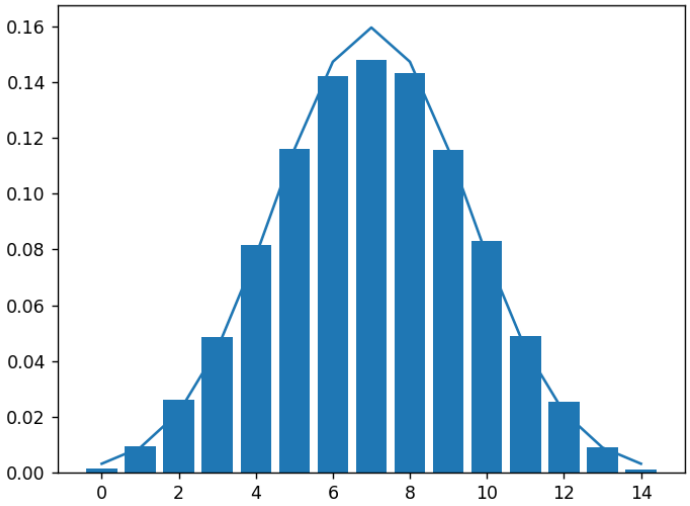
⇨ Modifier la fonction *trace()* pour y appeler la fonction *fDensite()* mise au point dans le paragraphe précédent et ainsi pouvoir tracer le diagramme barre et la courbe de la fonction de densité. Par simplicité, on donne le code de cette fonction :

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



En exécutant à présent `>>> trace(100000)` le résultat laisse apparaitre un léger décalage sur l'échelle des temps.

Ce défaut provient du fait que, dans la fonction *fDensite()* , les images sont calculées pour des valeurs de *t* qui ne correspondent pas au milieu des intervalles [0 – 1 mn] , [1 – 2 mn] , [2 – 3 mn] , ……., [14 – 15 mn]. En décalant les temps de la liste des temps *t* de 0.5 : `t = t + 0.5` on obtient une correspondance PARFAITE entre diagramme barre simulé et courbe calculée :



⇨ Déposer votre fichier dans le répertoire Devoir de votre classe.