SIMULATIONS D'EXPÉRIENCES ALÉATOIRES CONTINUES.

Les parties marquées d'un ⛰, ne doivent être abordées que si vous avez de l'avance.

Objectifs, motivations et premier pas

Le but de ce TP est de représenter les fonctions de densité et de répartition pour les lois classiques et de simuler ces lois.

Pour chacune des trois lois classiques, uniforme, exponentielle, normale, nous allons effectuer les tâches suivantes.

1. Utiliser les résultats mathématiques pour simuler des variables aléatoires suivant une loi en utilisant uniquement `rd.random()` qui simule une loi uniforme sur $[0;1[$. Les fonctions de simulation ont pour nom `loi(parametres,n)` où n est un entier, elles renvoient une liste de n valeurs.
2. Simuler un grand nombre d'expériences aléatoires en utilisant les fonctions de simulation.
3. Tracer le graphe des fréquences sous forme d'histogramme en utilisant `hist` du module matplotlib.pyplot
4. Tracer la fonction de densité théorique en utilisant `plot`. les noms des fonctions de densité sont sous la forme `loi_densite(x,paramètres)` où x désigne un réel.
5. Tracer l'histogramme des densités cumulées (répartition).
6. Tracer la fonction théorique de répartition en utilisant `plot`.

Remarque : Les fonctions du module `numpy.random` permettent de simuler un grand nombre de variables aléatoires. Les noms des fonctions utilisées sont les noms anglais des lois usuelles. Les paramètres peuvent être différents de ceux utilisés en mathématiques. Il ne faut pas hésiter à chercher de l'aide sur ces fonctions.

À programmer 1 (Fichier et Modules)**.**
Télécharger le fichier `TDcontinues.py`.
Importer les modules

- `numpy.random` avec l'alias `rd`
- `numpy` avec l'alias `np`
- `matlotlib.pyplot` avec l'alias `plt`

I Loi Uniforme.

I.1 Un exemple la loi : $\mathscr{U}([0;1[)$.

À programmer 2 (À tester)**.**
Dans le fichier se trouve le code suivant, à vous de le tester.

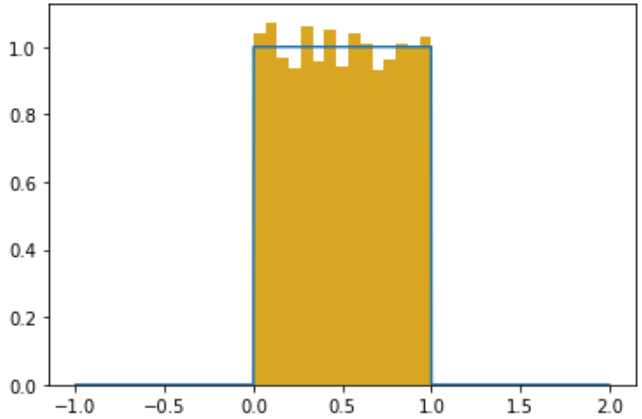
```
D=rd.random(10000)
#simulation d'un échantillonn de 10000 valeurs suivant une loi uniforme sur  [0,1[
plt.hist(D,color='goldenrod',density=True,bins=15)
#affichage d'un histogramme à 15 classes


def unif_densite(x):
    '''densité de la loi uniforme'''
    if x<0:
        return 0
    elif 0<x<1:
        return 1
    else:
        return 0

#tracé de la courbe représentative de la densité
debut=-1
fin=2
nbpas=1000
X=[debut + k*(fin-debut)/nbpas for k in range(nbpas)]
Y=[unif_densite(x) for x in X]
plt.plot(X,Y)

plt.show()
```

Vous devez obtenir :



Dans un premier temps on simule un grand nombre de fois une loi uniforme sur $[0;1[$, et on affiche un histogramme. Les arguments de la fonction `plt.hist` sont :

`D` : une série de valeurs

`bins` : nombre de classes et donc de barres de l'histogramme. L'étendue des données est séparée en `bins` classes de même largeur.

`density=True` : Permet d'afficher les fréquences, si `density=False` les effectifs de chaque classe sont représentés. Si `density=True` est choisie l'aire de la barre i est donnée par

$$\text{nombre de valeurs dans la classe } i / \text{effectif total} \times \text{largeur de la classe}$$

`cumulative` permet de choisir entre un graphe de distribution (Probability Density Function) ou de fonction de répartition (*Cumulative Distribution Function*). Si les options `cumulative` **et** `density` sont activées, la hauteur de la i-ème barre est donnée [1] par

$$\text{effectif cumulé des classes } 1,2,\ldots,i / \text{effectif total}$$

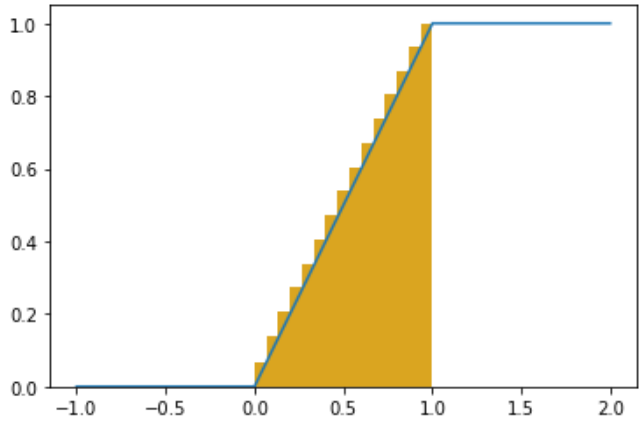
Puis on définit une fonction de densité de la loi uniforme sur $[0;1[$, et on superpose son graphe à l'histogramme précédent.

1. Dans le cas de classes de même largeur

À programmer 3.
En utilisant les explications précédentes copier le script précédent et le modifier pour

- Réaliser un grand nombre de simulations d'une variable aléatoire suivant la loi $\mathscr{U}([0;1[)$.
- Créer une fonction `uniforme_repartition(x)` qui pour tout réel x renvoie la valeur de la fonction de répartition d'une variable aléatoire suivant la loi $\mathscr{U}([0;1[)$.
- Afficher l'histogramme des fréquences cumulatives et l'allure de la fonction de répartition.

Vous devez obtenir



I.2 Simuler une loi $\mathscr{U}([a;b[)$

Sur papier 1 (Résultat mathématique)**.**
Soit a et b deux réels tels que $a<b$. Soit $X \hookrightarrow \mathscr{U}([0;1[)$ on pose $Y=(b-a)X+a$ Montrer que $Y \hookrightarrow \mathscr{U}([a;b[)$.

Nous allons donc, pour simuler une variable aléatoire suivant loi $\mathscr{U}([a;b[)$

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction $x \mapsto (b-a)x+a$.

À programmer 4 (Simulation)**.**
Écrire une fonction `uniforme(a,b,n)` qui renvoie une liste de n valeurs simulées d'une variable aléatoire suivant la loi $\mathscr{U}([a;b[)$.

À programmer 5 (Représentation d'une densité)**.**

1. Écrire une fonction `uniforme_densite(x,a,b)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `uniforme(a,b,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{U}([a;b[)$
3. Sur un même graphique faire afficher l'allure de la densité et l'histogramme des fréquences.

À programmer 6 (Représentation de la fonction de répartition)**.**

1. Écrire une fonction `uniforme_repartition(x,a,b)` qui renvoie la valeur de la fonction de répartition.
2. En utilisant la fonction `uniforme(a,b,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{U}([a;b[)$
3. Sur un même graphique faire afficher l'allure de la fonction de répartition et l'histogramme des fréquences cumulatives.

I.3 Loi uniforme sur $[\![1,p]\!]$.

À programmer 7 (Approche mathématique)**.**
Soit p un entier naturel non nul et $X \hookrightarrow \mathscr{U}([1;p+1[)$, on pose $Y=\lfloor X \rfloor$.
Montrer que $Y \hookrightarrow \mathscr{U}([\![1,p]\!])$.

À programmer 8 (Implémentation)**.**
En utilisant la fonction `uniforme(n,a,b)`, écrire une fonction `unifentier(n,p)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{U}([\![1,p]\!])$.

II Lois exponentielles.

II.1 Loi exponentielle par transfert.

Sur papier 2 (Calculs)**.**
Soit $X \hookrightarrow \mathscr{U}([0;1[)$ et soit $\lambda>0$. On pose $Y=-\frac{1}{\lambda}\ln(1-X)$. Montrer que $Y \hookrightarrow \mathscr{E}(\lambda)$.

Nous allons donc, pour simuler une variable aléatoire suivant loi $\mathscr{E}(\lambda)$:

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction $x \mapsto -\frac{1}{\lambda}\ln(1-x)$.

À programmer 9 (Simulation)**.**
En utilisant la fonction `rd.random()` écrire une fonction `exponentielle(lamb,n)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{E}(lamb)$.

À programmer 10 (Représentation d'une densité)**.**

1. Écrire une fonction `exponentielle_densite(x,lamb)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `exponentielle(lamb,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{E}(1)$
3. Sur un même graphique faire afficher l'allure de la densité et l'histogramme des fréquences.
4. Recommencer en changeant la valeur du paramètre λ et décrire la différence.

À programmer 11 (Représentation de la fonction de répartition)**.**

1. Écrire une fonction `exponentielle_repartition(x,lam)` qui renvoie la valeur de la densité choisie.
2. En utilisant la fonction `exponentielle(lamb,n)` simuler un grand nombre de fois une variable suivant la loi $\mathscr{E}(1)$
3. Sur un même graphique faire afficher l'allure de la fonction de répartition et l'histogramme

des fréquences cumulatives.

4. Recommencer en changeant la valeur du paramètre λ et décrire la différence.

II.2 ⛰ Loi géométrique.

Sur papier 3 (Approche mathématique)**.**
Pour tout nombre réel x, on note $\lfloor x \rfloor$ la partie entière de x.
Soit X la variable aléatoire suivant la loi exponentielle de paramètre λ ($\lambda > 0$).
On pose $Y = \lfloor X \rfloor$, Y est donc la partie entière de X.

1. Montrer que Y prend ses valeurs dans $\mathbb{N}$.
2. Pour tout k de $\mathbb{N}^*$, calculer $P(Y = k-1)$.
3. En déduire que la variable aléatoire $Y+1$ suit une loi géométrique dont on donnera le paramètre.

À programmer 12 (Implémentation)**.**
En utilisant les questions précédentes et la fonction `expo` précédente, écrire une fonction `geometrique(n,p)` qui renvoie une liste à n éléments dont chaque terme est choisi selon loi $\mathscr{G}(p)$.

III Cas général et loi normale.

III.1 Tracer des gaussiennes pour différents paramètres.

À programmer 13 (Implémentation)**.**
Écrire une fontion `normale_densite(x,m,sigma)` qui renvoie la valeur d'une densité de la loi normale $\mathscr{N}(m,\sigma^2)$.

À programmer 14 (Variations des paramètres)**.**

1. Sur un même graphique tracer les courbes représentatives des densités en fixant $\sigma = 1$ et en faisant varier le paramètre m. Décrire ce que vous constatez.
2. Sur un autre graphique tracer les courbes représentatives des densités en fixant $m = 0$ et en faisant varier le paramètre σ. Décrire ce que vous constatez.

III.2 ⛰ Résultats mathématiques.

Sur papier 4.
Soit $U \hookleftarrow \mathscr{U}(0,1)$, on note Φ la fonction de répartition de la loi normale centrée réduite.

1. Pourquoi Φ est elle **strictement** croissante et continue sur $\mathbb{R}$?
2. Montrer que Φ induit une bijection de $]-\infty;+\infty[$ dans un intervalle que l'on déterminera. Quelle est la monotonie de Φ^{-1}?
3. On note alors $Y = \Phi^{-1}(U)$, pour x réel montrer que
$$P(Y \leqslant x) = \Phi(x)$$
4. En déduire que Y suit la loi $\mathscr{N}(0,1)$.

Remarque : C'est la méthode que nous avons appliquée dans les cas précédents, en utilisant la bijection réciproque de la fonction de répartition en se limitant à un intervalle où la fonction de répartition est strictement croissante.

III.3 Simulation de la loi normale centrée réduite.

Nous allons donc, pour simuler une loi $\mathscr{N}(0,1)$.

- Choisir un nombre au hasard uniformément dans $[0;1[$.
- Lui appliquer la fonction Φ^{-1}.

À programmer 15 (À recopier)**.**
On vous donne le code pour calculer la bijection réciproque de Φ.

```
from scipy.stats import norm
def Finv(x):
        return norm.ppf(x)
    #ppf=Probability Point Function
```

À programmer 16 (Simulation)**.**
En utilisant la fonction `rd.random()` écrire une fonction `normale_cr(,n)` qui renvoie une liste de longueur n dont chaque terme est choisi selon loi $\mathscr{N}(0,1)$.

À programmer 17 (Représentation d'une densité)**.**
Représenter sur un même graphique un histogramme de densité et la valeur théorique de la densité pour la loi $\mathscr{N}(0,1)$.

III.4 Simulation dans le cas général.

Sur papier 5 (⛰)**.**
Montrer que si X suit la loi $\mathscr{N}(0,1)$ alors $Y = \sigma X + \mu$ suit la loi $\mathscr{N}(\mu,\sigma^2)$.

À programmer 18.
En utilisant ces informations écrire une fonction `normale(m,sigma,n)` qui renvoie une liste de n simulations d'une variable aléatoire suivant la loi $\mathscr{N}(m,\sigma^2)$.

À programmer 19 (Représentation d'une densité)**.**
Représenter sur un même graphique un histogramme de densité et la valeur théorique de la densité pour la loi $\mathscr{N}(1,1)$ puis pour $\mathscr{N}(0,2)$.

IV Reconnaissance de loi.

Le fichier fourni contient 4 séries de données mystères ; pour chacune de ces séries

1. Tracer un histogramme.
2. Proposer une famille de loi qui semble correspondre à cet histogramme.
3. Calculer la moyenne et l'écart-type de ces séries.
4. En déduire une estimation des paramètres de la loi ayant servi à générer ces séries.

V Autour de la loi normale.

À programmer 20 (Somme de deux variables aléatoires suivant des lois normales)**.**
On suppose que X suit une loi $\mathscr{N}(0,1)$ et que Y suit la même loi et que X et Y sont indépendantes.

Écrire un script qui simule un grand nombre de fois $X+Y$, tracer l'histogramme pour deviner la loi suivie par $X+Y$.

À programmer 21.
On suppose que $(X_1, X_2, \dots, X_n)$ suivent des lois $\mathscr{U}(]-1;1[)$ et sont indépendantes. On note

$$\overline{X_n}^* = \sqrt{3}\frac{X_1+\cdots+X_n}{\sqrt{n}}$$

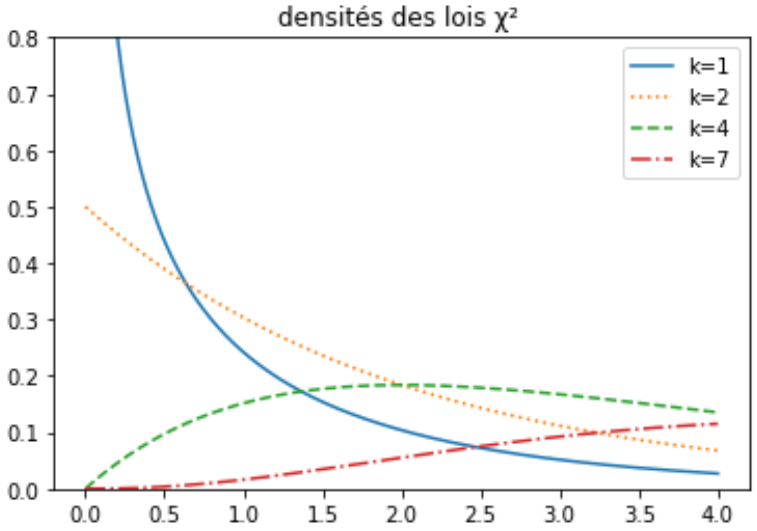
et on cherche à comprendre quelle est la loi suivie par cette variable aléatoire.

1. Écrire une fonction `Xbarre(n,N)` qui renvoie une liste de N simulations, où chaque valeur est une simulation de $\overline{X_n}^*$.
2. On fixe $n=10$, et on choisit N grand ; utiliser la fonction précédente et un histogramme pour deviner la loi suivie par $\overline{X_n}^*$.
3. En calculant la moyenne et l'écart-type donner une estimation des paramètres de cette loi.
4. Recommencer avec $n=20$.

Loi du χ^2. La loi du χ^2 prononcé « ki deux » ou « ki-carré » est une loi utilisée lors des tests statistiques.
La loi χ^2 à k degrés de liberté est définie comme la loi de la somme de k carrés de variables aléatoires suivant des lois $\mathscr{N}(0,1)$ et indépendantes

$$C_k = \sum_{i=1}^{k} X_i^2 \qquad X_1, \dots, X_k \text{indépendantes, de loi} \mathscr{N}(0,1)$$

À programmer 22 (Simulation de la loi du χ^2)**.**
En utilisant les fonctions précédentes, écrire une fonction `chi2(k,n)` qui renvoie une liste de n simulations de variable aléatoires suivant la loi du χ^2 à k degrés de liberté. Pour information on vous donner l'allure des graphes des densités des lois dde χ^2.



Loi de Student. La loi de Student (en anglais *t distribution*), utilisée en statistiques, est la loi suivie par le quotient d'une loi normale et de la racine d'une loi du χ^2 à k degrés de liberté.

$$S_k = \frac{N}{\sqrt{\frac{C_k}{k}}} \qquad N \text{ suit } \mathscr{N}(0,1), \quad C_k \text{ suit une loi du } \chi^2 \text{ à } k \text{ degrés de liberté, indépendantes}$$

À programmer 23 (Simulation de la loi de Student)**.**
En utilisant les fonctions précédentes, écrire une fonction `student(k,n)` qui renvoie une liste de n simulations d'une variable aléatoire suivant la loi de Student à k degrés de liberté. Pour information on vous donner l'allure des graphes des densités des lois de Student.