TP6 : Simulation de lois via la bibliothèque `numpy.random`

- Dans votre dossier `Info_2a`, créez le dossier `TP_6`.

I. Utilisation des fonctions de `numpy.random`

Dans ce TP, nous délaissons la fonction `rd.random` pour utiliser les fonctions de la bibliothèque `numpy.random` qui présentent l'avantage de pouvoir simuler très simplement l'observation d'un échantillon de `N` v.a.r. indépendantes.
On commencera donc par importer cette bibliothèque.

```
1  import numpy.random as nr
```

I.1. Simulation d'une v.a.r. suivant une loi géométrique

I.1.a) Utilisation de la fonction `nr.geometric`

- Évaluer plusieurs fois la commande `nr.geometric(0.4)`. Qu'obtient-on ?

- Évaluer maintenant la commande `nr.geometric(0.4,10)`. Qu'obtient-on ?

- Évaluer maintenant la commande `[nr.geometric(0.4,4) for k in range(2)]`. Qu'obtient-on ?

- Généraliser ce résultat.

- Écrire un programmme qui :
  - × demande à l'utilisateur d'entrer la valeur d'un paramètre `p`,
  - × demande à l'utilisateur d'entrer la valeur d'un paramètre `N`,
  - × réalise la simulation d'un échantillon de `N` v.a.r. indépendantes de loi géométrique $\mathcal{G}(\texttt{p})$, *(on stocke le résultat obtenu dans une variable* `Obs`*)*
  - × trace le diagramme des effectifs de cette simulation.

  On pourra utiliser les fonctions `tabul` (pour calculer les effectifs) et la fonction `bar` (pour représenter le digramme en bâtons correspondant).

(on enregistre ce programme sous le nom `distrib_geom.py`*)*

- Ajouter `rwidth=0.8, color = ‘‘red‘‘, edgecolor = ‘‘black‘‘` dans l'appel à la fonction `plt.hist`. À quoi servent ces arguments optionnels ?

- Comment modifier le programme précédent afin d'obtenir le diagramme des fréquences ?

I.1.b) Comparaison avec la distribution théorique

On propose d'ajouter les lignes suivantes au programme précédent.

```
1  lx = [k for k in range(1,11)]
2  ly = [(1-p)**(k-1) * p for k in range(1,11)]
3
4  plt.step(lx, ly)
```

- Que contient la variable `y` ? Que cela représente-t-il pour une v.a.r. X telle que $X \hookrightarrow \mathcal{G}(p)$?

- Le choix fait pour la variable `x` vous semble-t-il pertinent ?
  Comment la modifier de sorte à représenter autant de bâtons que dans le diagramme précédent ?

- Comment s'affiche les deux diagrammes obtenus ? Comment améliorer cet affichage ?

- On pourra enfin ajouter un titre et une légende à ce diagramme à l'aide des commandes suivantes.

```
plt.hist(Obs, normed = True, bins = x, color = ‘‘red‘‘,
        edgecolor = ‘‘black‘‘, hatch = ‘‘/‘‘,
        label = ‘‘Diagramme distribution approchée‘‘)
plt.step(lx, ly, label = ‘‘Diagramme distribution théorique‘‘)
plt.legend()
plt.title(‘‘Comparaison de la distribution théorique et de la distribution‘‘
        + ‘‘approchée pour une var suivant une loi géométrique‘‘)
```

I.2. Simulation d'une v.a.r. suivant une loi de Poisson

I.2.a) Utilisation de la fonction `grand`

▶ Si `N` et `lam` sont donnés, à quoi sert l'appel `nr.poisson(lam, N)` ?

▶ Quel commande permet d'obtenir la simulation de `m` échantillons de `N` v.a.r. indépendantes X_i suivant toutes la loi de Poisson de paramètre `lam` ?

I.2.b) Comparaison avec la distribution théorique

▶ Soit X une v.a.r. . Que signifie que $X \hookrightarrow \mathcal{P}(\lambda)$? Détailler.

▶ Dans un nouvel onglet **Python**, copier le programme `distrib_geom.py` et l'adapter au cas de la loi de Poisson. Pour la distribution théorique, on pourra se servir de la fonction `np.math.factorial`.

(on enregistre ce programme sous le nom `distrib_poisson.py`*)*

II. Bilan sur la bibliothèque `numpy.random`

▶ Chercher de l'aide en ligne sur la bibliothèque `numpy.random` sur le site numpy.org et prendre connaissance des informations contenues dans la section `Random sampling`. Compléter alors le texte ci-dessous.

`(N)`	génère un échantillon de `N` variables aléatoires suivant la loi $\mathcal{U}([0,1[)$.
`rd.randint(a,b,N)`	génère un échantillon de `N` variables aléatoires suivant la loi .
`(n,p,N)`	génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{B}(\texttt{n},\texttt{p})$.
`nr.geometric(p,N)`	génère un échantillon de `N` v.a.r. suivant la loi .
`(lam,N)`	génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{P}(\texttt{lam})$.
`nr.exponential(alpha,N)`	génère un échantillon de `N` v.a.r. suivant la loi $\mathcal{E}(\texttt{alpha})$.

II.1. Simulation d'une v.a.r. suivant une loi normale

II.1.a) Utilisation de la fonction `grand`

▶ Si `N`, `esp` et `sigma` sont donnés, à quoi sert l'appel `nr.normal(esp,sigma,N)` ?

II.1.b) Comparaison avec la distribution théorique

▶ Soit X une v.a.r. . Que signifie que $X \hookrightarrow \mathcal{N}(0,1)$? Détailler.

▶ Écrire la fonction `densiteNormaleCR`, densité de la loi normale centrée réduite.

- On considère un vecteur `Obs` contenant `N` valeurs comprises entre -5 et 5.
  On souhaite, grâce à `plt.hist`, générer le diagramme des fréquences de `Obs`. Comment le générer de sorte qu'il contienne 100 bâtons dont le premier commence en -5 et le dernier finit en 5 ?

- Dans un nouvel onglet **Python**, adapter les programmes précédents au cas de la loi normale centrée réduite.