TP5 : Simulation de v.a.r. suivant une loi usuelle discrète via la fonction `random`

▶ Dans votre dossier `Info_2a`, créez le dossier `TP_5`.

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

▶ Pour quel type d'expérience définit-on une v.a.r. suivant une loi de Bernoulli ?

> Considérons une expérience aléatoire possédant deux issues non forcément équiprobables. L'une de ces issues est nommée « succès » et se produit avec probabilité p ; l'autre est nommée « échec » et se produit avec probabilité $1-p$.
>
> Alors la v.a.r. X égale à 1 en cas de succès et 0 en cas d'échec (*i.e.* calculant le nombre de succès) suit la loi de Bernoulli de paramètre p.

▶ Que signifie qu'une v.a.r. X suit la loi de Bernoulli de paramètre p ? (détailler)

> $X \hookrightarrow \mathcal{B}(p)$ signifie : $X(\Omega) = \{0,1\}$ $\mathbb{P}(X=1) = p$ et $\mathbb{P}(X=0) = 1-p$

On considère la fonction **Python** suivante.

```
1  import random as rd
2
3  def bernoulli(p) :
4      r = rd.random()
5      if r < p :
6          z = 1
7      else :
8          z = 0
9      return z
```

▶ Copier ce programme et l'enregistrer sous le nom `bernoulli.py`.
Tester alors cette fonction sur plusieurs paramètres. Que renvoie-t-elle ?

> Elle renvoie 1 avec la probabilité p et renvoie 0 avec la probabilité $1-p$.

▶ Écrire une fonction `sampleBernoulli` qui :

- × prend en paramètre un entier `N`, *(représente le nombre d'observations souhaitées)*
- × prend en paramètre un réel `p` dans $]0,1[$,
- × renvoie une liste `listRes` contenant le résultat de la simulation de `N` v.a.r. indépendantes suivant la loi $\mathcal{B}(\texttt{p})$.

- Recopier cette fonction ci-dessous.

```
def sampleBernoulli(N, p) :
    listRes = [0 for k in range(N)]
    for k in range(N) :
        listRes[k] = bernoulli(p)
    return listRes
```

- Recopier le programme suivant.

```
p = 0.3
N = 10000
X = sampleBernoulli(N, p)

plt.hist(X, normed = False)
```

Que fait la commande `plt.hist` ? On commentera en particulier l'utilité de `normed`.

L'appel `plt.hist(X)` renvoie un histogramme.

- les positions des barres sont données par les valeurs apparaissant dans la liste `X`,
- la hauteur des barres est donnée par l'effectif des valeurs apparaissant dans la liste `X`.

L'argument optionnel `normed` permet de choisir si on souhaite que l'histogramme soir normalisé.

- Le résultat obtenu est-il cohérent ?

Le diagramme est cohérent : 1 apparaît environ 3000 fois et 0 apparaît environ 7000 fois.

- Le diagramme obtenu représente l'effectif de chaque classe.
Comment modifier l'appel pour obtenir un diagramme des fréquences ?

On change la dernière ligne en `plt.hist(X, normed = True)`.

II. Simulation d'une v.a.r. suivant une loi binomiale

- Pour quel type d'expérience définit-on une v.a.r. suivant une loi binomiale ?

On considère une expérience aléatoire qui consiste en une succession de n épreuves indépendantes, chacune d'entre elles ayant deux issues : succès obtenu avec probabilité p et échec obtenu avec probabilité $q = 1 - p$.
Autrement dit, l'expérience consiste à effectuer n épreuves de Bernoulli identiques et indépendantes.

Alors la v.a.r. donnant le nombre de succès de l'expérience suit la loi binomiale de paramètre (n, p).

- Que signifie qu'une v.a.r. X suit la loi binomiale de paramètre (n, p) ? (détailler)

$X \hookrightarrow \mathcal{B}(n,p)$ signifie : $X(\Omega) = [\![0, n]\!]$ et $\forall k \in [\![0, n]\!],\ \mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

- Écrire une fonction `binomiale` qui :
  - × prend en paramètre un entier `n`,
  - × prend en paramètre un réel `p` dans $]0,1[$,
  - × renvoie dans une variable `z` le résultat de la simulation d'une v.a.r. suivant la loi $\mathcal{B}(\mathtt{n},\mathtt{p})$.

  On pourra utiliser la fonction `sampleBernoulli`.

```
def binomiale(n, p) :
    z = sum(sampleBernoulli(n, p))
    return z
```

- Écrire une fonction `sampleBinomiale` qui :
  - × prend en paramètre un entier `N`, *(représente le nombre d'observations souhaitées)*
  - × prend en paramètre un entier `n`,
  - × prend en paramètre un réel `p` dans $]0,1[$,
  - × renvoie une liste `listRes` contenant le résultat de la simulation de N v.a.r. indépendantes suivant la loi $\mathcal{B}(\mathtt{n},\mathtt{p})$.

```
def sampleBinomiale(N, n, p) :
    listRes = [0 for k in range(N)]
    for k in range(N) :
        listRes[k] = binomiale(n, p)
    return listRes
```

- Détailler les commandes permettant d'obtenir le tracé d'un diagramme des effectifs à l'aide de la fonction `sampleBinomiale` et de la fonction `plt.hist`.

```
X = sampleBinomiale(N, n, p)
plt.hist(X, normed = True)
```

III. Simulation d'une v.a.r. suivant une loi géométrique

- Pour quel type d'expérience définit-on une v.a.r. suivant une loi binomiale ?

On considère une expérience aléatoire qui consiste en une succession infinie d'épreuves indépendantes, chacune d'entre elles ayant deux issues : succès obtenu avec probabilité p et échec obtenu avec probabilité $q = 1 - p$.
Autrement dit, l'expérience consiste à effectuer une infinité d'épreuves de Bernoulli identiques (même paramètre) et indépendantes (le résultat de l'une ne dépend pas du résultat des autres).

Alors la v.a.r. donnant le rang d'apparition du premier succès de l'expérience suit la loi géométrique de paramètre p.

▶ Que signifie qu'une v.a.r. X suit la loi géométrique de paramètre p ? (détailler)

$X \hookrightarrow \mathcal{G}(p)$ signifie : $\quad X(\Omega) = \mathbb{N}^* \quad$ et $\quad \forall k \in \mathbb{N}^*,\ \mathbb{P}(X = k) \ = \ p\,(1-p)^{k-1}$

On considère la fonction **Python** suivante.

```
1  def geometrique(p) :
2      rang = 1
3      aux = bernoulli(p)
4      while aux == 0 :
5          rang = rang + 1
6          aux = bernoulli(p)
7      return rang
```

▶ Copier ce programme et l'enregistrer sous le nom `geometrique.py`.
Tester alors cette fonction sur plusieurs paramètres. Que calcule-t-elle ?

Elle simule une v.a.r. suivant une loi géométrique de paramètre p. Plus précisément :

× on simule les résultats successifs d'épreuves de Bernoulli de paramètre p,

× on s'arrête dès le premier succès rencontré,

× le compteur `rang` calcule le rang d'apparition de ce succès, qui est le résultat renvoyé.

▶ Est-on sûr que cette fonction s'arrête toujours ?

Si `aux` vaut toujours `0` (les épreuves de Bernoulli produisent **toutes** un échec), la boucle est infinie. Cependant, la terminaison est assurée ici car un succès a lieu `presque sûrement`.

▶ Écrire une fonction `sampleGeometrique` qui :

× prend en paramètre un entier `N`, *(représente le nombre d'observations souhaitées)*

× prend en paramètre un réel `p` dans $]0, 1[$,

× renvoie une liste `listRes` contenant le résultat de la simulation de N v.a.r. indépendantes suivant la loi $\mathcal{G}(\mathtt{p})$.

```
1  def sampleGeom(N, p) :
2      listRes = [0 for k in range(N)]
3      for k in range(N) :
4          listRes[k] = geometrique(p)
5      return listRes
```

▶ Détailler les commandes permettant d'obtenir le tracé d'un diagramme des effectifs à l'aide de la fonction `sampleGeometrique` et de la fonction `plt.hist`.

```
1  X = sampleGeom(N, p)
2  plt.hist(X, normed = True)
```