Corrigé Code de partage avec Capytale : c11e-10114854

On utilisera les bibliothèques suivantes :

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
import scipy.special as sp
```

Approximation d'une loi binomiale par une loi de Poisson

1. Compléter la fonction suivante renvoyant les 11 premières valeurs de la loi de Poisson de paramètre `t`

```
def LoiPoisson(t):
    proba=np.exp(-t)
    loi=[proba]
    for k in range(1, 11):
        proba =proba*t/k
        loi.append(proba)
    return loi
```

On utilise ici une relation récursive sur les probabilités de la loi de Poisson :
pour $k \in \mathbb{N}^*, P(X=k) = \dfrac{\lambda}{k} P(X=k-1)$

2. En remarquant que, pour tous entiers $n \geqslant i \geqslant 1$:

$$\binom{n}{i} = \frac{n-i+1}{i}\binom{n}{i-1} \tag{1}$$

compléter la fonction suivante renvoyant les 11 premières valeurs de la loi binomiale $\mathscr{B}(n, t/n)$

```
def LoiBin(n, t):
    p=t/n
    q=1-p
    proba= q**n
    loi=[proba]
    for k in range(1, 11):
        proba = proba*(n-k+1)/k*p/q
        loi.append(proba)
    return loi
```

La relation nous permet de procéder de manière analogue :
pour $k \in \mathbb{N}^*, P(X=k) = \dfrac{n-k+1}{k} P(X=k-1)\dfrac{p}{q}$

3. Représenter graphiquement et sur un même graphique les 11 premières valeurs des deux lois calculées précédemment, pour $k = 5$

Il s'agit simplement d'utiliser les deux fonctions précédentes et de représenter les deux séries de valeurs.

```
x=[k for k in range(0,11)]
y1=LoiPoisson(3)
y2=LoiBin(5,3)
plt.close()
plt.plot(x,y1,'*')
```

```
plt.plot(x,y2,'.')
plt.show()
```

4. On considère le programme suivant, qui représente les lois binomiales $\mathscr{B}(k, 3/k)$ pour les valeurs de k entre 5 et 35, et affiche les premières valeurs de la loi de Poisson $\mathscr{P}(3)$

```
def Affichage():
    loipoisson=LoiPoisson(3)
    plt.ylim(0, 0.4)
    x=np.linspace(0, 10, 11)
    loibin=[]
    for k in range(5, 35, 5):
        loibin.append(LoiBin(k, 3))
        plt.close()
        for l in loibin:
            # plt.close()
            plt.plot(x, l, '+')
        plt.plot(x, loipoisson, '*')
        plt.show()
Affichage()
```

Commenter le résultat.

On remarque que les points de la loi binomiale se rapprochent de la loi de Poisson (on peut utiliser le `plt.close()` pour enlever les graphiques précédents, ce qui laisse à penser que la loi $\mathscr{B}(k, 3/k)$ converge en loi, quand k tend vers $+\infty$, vers $\mathscr{P}(3)$

Approximation d'une loi binomiale par une loi normale

Rappel : pour $p \in]0, 1[$ fixé, si $S_n \hookrightarrow \mathscr{B}(n, p)$, alors

$$S_n^* \xrightarrow[n\to+\infty]{\mathscr{L}} N \hookrightarrow \mathscr{N}(0, 1)$$

où $S_n^* = \dfrac{S_n - E(S_n)}{\sigma(S_n)}$ désigne la variable centrée réduite associée à S_n
Cela signifie que $\forall x \in \mathbb{R}$, $F_{S_n^*}(x) \xrightarrow[n\to+\infty]{} \Phi(x)$ car Φ est continue sur $\mathbb{R}$. On va illustrer ce résultat.
N.B. sous Python, la bibliothèque `scypy.special as sp` donne l'accès à la fonction de répartition Φ de la loi normale $\mathscr{N}(0, 1)$ par la commande : `sp.ndtr(x)` qui renvoie une valeur approchée de $\Phi(x)$

1. Représenter graphiquement la fonction Φ sur l'intervalle $[-5, 5]$

```
plt.close()
x=np.linspace(-5,5,100)
y=sp.ndtr(x)
plt.plot(x,y)
plt.show()
```

2. Représenter graphiquement la fonction de répartition de la variable S_{20}^* pour $p = \dfrac{1}{4}$

   On pourra remarquer que $P(S_n = k) = P\left(S_n^* = \dfrac{k - np}{npq}\right)$

3. On considère le programme suivant :

```
def ApproxBinNor():
    p=1/4
    q=3/4
```

```
    for n in range(20, 200, 10):
        plt.close()
        pbin=q**n
        F=[pbin]
        for i in range(1, n+1):
            pbin *= (n-i+1)/i*p/q
            F.append(F[-1]+pbin)
        mu=n*p
        sigma=np.sqrt(n*p*q)
        x=np.linspace(0, n, n+1)
        y=(x-mu)/sigma

        plt.plot(y, F, '+')
        x2=np.linspace(-5, 5, 100)
        y2=[sp.ndtr(t) for t in x2]
        plt.plot(x2, y2)
        plt.show()

ApproxBinNor()
```

Questions sur la fonction `ApproxBinNor()` :

1) A quoi correspondent les valeurs `mu` et `sigma` des lignes 12 et 13 ?

   Ces valeurs correspondent à la moyenne et à l'écart-type de S_n, ce sont donc la moyenne et la racine carrée de la variance d'une loi binomiale

2) Expliquer les lignes 10 et 11.

   On calcule les valeurs (et on les intègre dans une liste) de la fonction de répartition de la loi binomiale pour $k \in]\!]0, n[\!]$

3) Que trace-t-on dans la ligne 17 ?

   On représente la fonction de répartition mentionnée à la question précédente.

4) Qu'observez-vous à l'exécution du programme ?
   Justifier alors l'approximation $\mathscr{B}(n,p) \approx \mathscr{N}(np, npq)$

   On remarque que la fonction de répartition de la loi binomiale $\mathscr{B}(n,p)$ se rapproche de plus en plus, voire se confond, avec celle de la loi $\mathscr{N}(np, npq)$ ce qui n'est pas une surprise, puisqu'il y a convergence en loi comme nous l'avons vu en cours. L'approximation semble donc pertinente.

Questions mathématiques

5) Déterminer $S_n(\Omega)$ et $S_n^*(\Omega)$. On notera $S_n^*(\Omega) = \{x_0, \dots x_n\}$ où $x_0 < x_1 < \dots x_n$

6) Montrer que, pour $i \in \{0, \dots n\}$, $F_{S_n^*}(x_i) = \sum_{j=0}^{i} P(X_n = j)$