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, ...):
        proba =...
        loi.append(...)
    return ...
```

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=...
    q=...
    proba= q**n
    loi=[proba]
    for k in range(1, ...):
        proba = ...
        loi.append(...)
    return ...
```

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

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 $[-5,5]$
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 ?
2) Expliquer les lignes 10 et 11.
3) Que trace-t-on dans la ligne 17 ?
4) Qu'observez-vous à l'exécution du programme ?
   Justifier alors l'approximation $\mathscr{B}(n,p) \approx \mathscr{N}(np, npq)$

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