TP 17 annales EDHEC traduites en Python

EDHEC 2022

1. *Soit $n \in \mathbf{N}^*$, $p \in]0;1[$. On pose $q = 1 - p$.*

   On s'intéresse à un jeu vidéo au cours duquel le joueur doit essayer, pour gagner, de réussir, dans l'ordre, n niveaux numérotés $1, 2, \ldots, n$, ce joueur ne pouvant accéder à un niveau que s'il a réussi le niveau précédent. Le jeu s'arrête lorsque le joueur échoue à un niveau ou bien lorsqu'il a réussi les n niveaux du jeu.

   Pour tout entier $k \in [\![1; n-1]\!]$, on dit que le joueur a le niveau k si et seulement si il a réussi le niveau k et échoué au niveau $k+1$. On dit que le joueur a le niveau n si et seulement si il a réussi le niveau n, et on dit que le joueur a le niveau 0 si et seulement si il a échoué au niveau 1.

   On admet que la probabilité de passer d'un niveau à l'autre est constante et égale à p, la probabilité d'accéder au niveau 1 étant, elle aussi, égale à p.

   On note X_n le niveau du joueur et on admet que X_n est une v.a. définie sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ que l'on ne cherchera pas à déterminer.

   Compléter le script Python suivant afin qu'il permette de simuler ce jeu et d'afficher la valeur prise par X_n dès que l'utilisateur saisit une valeur pour p.

```python
import numpy.random as rd
p = float(input('Entrez la valeur de p dans ]0;1[ : '))
n = int(input('Entrez la valeur de n : '))
X = ......................
while (..........) and (rd.random() <= p) :
    X = ........................
print('Le niveau du joueur est : ', X)
```

2. *On admet que dans le sujet, on a démontré qu'une certaine suite $(S_n)_{n \in \mathbf{N}^*}$ est croissante et converge vers γ, qu'une autre suite $(T_n)_{n \in \mathbf{N}^*}$ est décroissante et converge vers γ (dont on ne connaît pas la valeur), avec*

$$\forall n \in \mathbf{N}^* \ , \ S_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n+1) \quad et \quad T_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n) \ .$$

   (a) Donner un encadrement de γ à l'aide des réels S_n et T_n.

   (b) En utilisant cet encadrement, préciser ce que représente S_n pour γ lorsque $T_n - S_n$ est inférieur ou égal à 10^{-3}.

   (c) Déterminer $T_n - S_n$, puis compléter le script Python suivant afin qu'il affiche une valeur approchée de γ à 10^{-3} près.

```python
from math import *
n = 1
s = 1 - log(2)
while ........ :
    n = .......
    s = s + .......
print(........)
```

EDHEC 2021

1. *On donne $a \in]0;1[$ et trois suites $(u_n)_{n \in \mathbf{N}}$, $(v_n)_{n \in \mathbf{N}}$ et $(w_n)_{n \in \mathbf{N}}$ vérifiant $u_0 = v_0 = 0, w_0 = 1$ et*

$$\forall n \in \mathbf{N} \ , \ \begin{cases} u_{n+1} = (2a+1)u_n + v_n \\ v_{n+1} = -a(a+2)u_n + w_n \\ w_{n+1} = a^2 u_n \end{cases} \ .$$

(a) Expliquer pourquoi le script Python qui suit ne permet pas de calculer et d'afficher les valeurs de u_n, v_n et w_n lorsque n et a sont entrés par l'utilisateur. *On pourra examiner attentivement l'intérieur de la boucle « for »* .

```
n = int(input('Entrez une valeur pour n : '))
a = float(input('Entrez une valeur pour a : '))
u = 0
v = 0
w = 1
for k in range(1, n+1):
    u = (2 * a + 1) * u + v
    v = - a * (a + 2) * u + w
    w = a * a * u
print(u,v,w)
```

(b) Modifier la boucle de ce script en conséquence.

2. *Deux joueurs A et B s'affrontent lors de lancers de pièces identiques donnant pile avec probabilité p, les lancers étant supposés indépendants.*

Pour la première manche, A et B lancent chacun leur pièce simultanément jusqu'à ce qu'ils obtiennent pile, le gagnant du jeu étant celui qui a obtenu pile en premier. En cas d'égalité, et en cas d'égalité seulement, les joueurs participent à une deuxième manche dans les mêmes conditions et avec les mêmes règles, et ainsi de suite jusqu'à la victoire de l'un d'entre eux.

Pour tout $k \in \mathbf{N}^*$, on note X_k (respectivement Y_k) la variable aléatoire égale au rang d'obtention du 1er pile par A (respectivement par B) lors de la k-ième manche.

(a) On rappelle que la commande `rd.geometric(p)` permet de simuler une variable aléatoire suivant une loi géométrique de paramètre p. Compléter le script Python suivant pour qu'il simule l'expérience décrite et affiche le nom du vainqueur ainsi que le numéro de la manche à laquelle il a gagné.

```
import np.random as rd
p = float(input('Entrez une valeur pour p : '))
c = 1
X = rd.geometric(p)
Y = rd.geometric(p)
while X == Y:
    X = .......
    Y = .......
    c = .......
if X < Y:
    ........
else:
    ........
print(c)
```

(b) *Dans un autre jeu, A parie sur le fait que la manche gagnée par le vainqueur du premier jeu le sera par un lancer d'écart, et B parie le contraire.*

Compléter le script suivant afin qu'une fois ajoutée au script précédent, elle permette de simuler le deuxième jeu

```
if ..... :
    print('A gagne le deuxieme jeu')
else :
    .......
```

HEC 2022

Pour les deux questions, on pose a un entier supérieur ou égal à 2, et $\gamma_1, \gamma_2, \ldots, \gamma_a$ des réels. À part cela, les deux questions sont indépendantes.

1. On définit F_T la fonction de répartition d'une variable aléatoire étudiée dans le sujet par

$$\forall t \in [0; a[\ , \ F_T(t) = 1 - \exp\left(-\gamma_{\lfloor t \rfloor + 1}(t - \lfloor t \rfloor)\right) \exp\left(-\sum_{k=1}^{\lfloor t \rfloor} \gamma_k\right) \ . \tag{1}$$

On suppose que le vecteur numpy `gammaTab` contient les valeurs $\gamma_1, \ldots, \gamma_a$.
Compléter la fonction suivante pour qu'elle renvoie la valeur de $F_T(t)$ obtenue dans l'égalité (1).

```
def F(t, gammaTab):
    produit = ...
    i = ...
    for k in range(1, i):
        produit = produit * exp( - gammaTab(k) )
    return 1 - exp( - gammaTab(i) * (........) ) * produit
```

2. *On pose $r \in \mathbf{R}_+^*$ et pour tout $k \in [\![1; a]\!]$,*

$$\alpha_k = r + \gamma_k \ .$$

On se donne aussi deux familles de réels $(A_k)_{k \in [\![0;a]\!]}$, avec $A_0 = 1$ et $(\theta_k)_{k \in [\![0;n]\!]}$ avec $\theta_0 = 1$. On admet que l'on peut obtenir les γ_k en résolvant le système linéaire suivant :

$$\forall k \in [\![1; a]\!] \ , \ \theta_k(1 - A_{k-1}(1 - \alpha_k)) - A_{k-1} = \theta_{k-1}(1 - A_{k-1}) \ . \tag{2}$$

   (a) Écrire γ_k en fonction de A_{k-1}, θ_k, θ_{k-1} et r, et ce pour tout $k \in [\![1; a]\!]$.

   (b) *On admet que pour tout $k \in [\![1; a]\!]$ on a*

$$A_k = A_{k-1} \mathrm{e}^{-\alpha_k} \qquad \text{et} \qquad \theta_k = \frac{1}{r + \dfrac{s_k}{1 - \delta}} \ ,$$

   où $\delta \in]0; 1[$ et les s_k sont des réels strictement positifs.
   Écrire une fonction Python qui prend en entrée a, r, δ et $(s_1, \ldots, s_a)$ et qui renvoie le vecteur `gammaTab` rempli en calculant les γ_k de proche en proche.

HEC 2021

On fixe un espace probabilité $(\Omega, \mathcal{A}, \mathbb{P})$, un sous-ensemble J de $\mathbf{R}_+$, Y une v.a. définie sur $(\Omega, \mathcal{A}, \mathbb{P})$ vérifiant $Y(\Omega) = J$, et une famille $(X_t)_{t \in J}$ de variables aléatoires définie sur $(\Omega, \mathcal{A}, \mathbb{P})$ à valeurs dans $\mathbf{N}$ et indépendantes de Y telles que pour tout $t \in J$, X_t suit la loi $\mu(t)$, $\mu(t)$ désignant une loi de probabilité de paramètre t. On définit alors Z sur le même espace probabilisé par

$$\forall \omega \in \Omega \ , \quad \text{si } Y = t \text{ alors } Z(\omega) = X_t(\omega) \ .$$

Autrement dit quelque soit $\omega \in \Omega$, on a $Z(\omega) = X_{Y(\omega)}(\omega)$. On dit alors que Z suit la loi $\mu(Y)$.

1. On considère le script Python suivant :

```python
import numpy.random as rd

def X(t) :
    result = 1
    while rd.random() > ..... :
        result =
    return result

Y = rd.random()
Z = .....
print(Z)
```

En considérant les notations précédentes avec $J =]0;1[$ et en notant Y la variable aléatoire dont `Y` est une simulation, compléter le script précédent pour que `Z` soit une simulation d'une v.a. qui suit la loi géométrique $\mathcal{G}(Y)$.

2. *On définit la suite* $(r_n)_{n\in\mathbf{N}}$ *par* $r_n = \dfrac{n+2}{n+1}$ *pour tout* $n \in \mathbf{N}$. *On note* Δ *la matrice ligne* $\begin{pmatrix}\alpha_0 & \alpha_1 \cdots \alpha_d\end{pmatrix}$, *où les* α_k *sont des réels de* $]0;1[$. *On a démontré dans le sujet*

$$z_0 = 1 \quad \textit{et} \quad \forall n \in \mathbf{N}\ ,\ z_{n+1} = r_n \sum_{k=0}^{\min(n,d)} \alpha_k z_{n-k}\ .$$

Écrire une fonction Python nommée `z` de variables `Delta` et `n` qui renvoie z_n si `Delta` représente Δ.

HEC 2020

1. Après avoir établi la formule $\dbinom{n}{k} = \dfrac{n}{k}\dbinom{n-1}{k-1}$ lorsque $k \in [\![1;n]\!]$, écrire une fonction Python **récursive** qui calcule les coefficients binomiaux.
2. *On considère des évènements* $(G_n)_{n\in\mathbf{N}^*}$ *vérifiant :*

$$\forall n \in \mathbf{N}^*\ ,\ \mathbb{P}(G_n) = \frac{p}{q} - \left(1 - \frac{p}{q}\right)\sum_{k=1}^{n}\binom{2k-1}{k}(pq)^k\ .$$

Écrire un script Python qui détermine n_p le plus petit entier n tel que $\mathbb{P}(G_n) \leqslant \varepsilon$ pour $p < 1/2$ et $\varepsilon > 0$ tous deux saisis au clavier par l'utilisateur.