Devoir d'Informatique # 1

à rendre le Lundi 03 Septembre 2018, 23h59

Lycée
Chaptal
2 BCPST A/B

Version du 15 septembre 2018, 16 :05

Le devoir est à rendre sous forme d'un fichier `.py` *à l'adresse mail jona.harter.maths@gmail.com, avant la date indiquée. Il devra présenter l'en-tête ci-dessous que vous complèterez avec les bons renseignements.*
Par ailleurs, vous êtes libres d'organiser votre travail comme vous le souhaitez : notamment vous avez le droit de créer des fonctions en début de fichier afin de les utiliser dans plusieurs exercices ensuite.
Enfin, toute question posée à l'adresse mail ci-dessus pendant les vacances est la bienvenue.

```
#*****************************************************
# 2BIOA ou B DM de vacances 2018
# Nom/Prénom: à compléter (écrit en python 3.x)
#*****************************************************
Vos réponses ici.
```

Exercice 1

Soit $(u_n)_{n\in\mathbf{N}}$ la suite définie par : $\begin{cases} u_0 = 0,\ u_1 = 1,\ u_2 = 2, \\ u_{n+3} = 2\,u_{n+2} + 3\,u_{n+1} - 2\,u_n\,,\ \forall\, n \in \mathbf{N}, \end{cases}$ et $(\mathrm{R}_n)_{n\in\mathbf{N}^*}$ la suite définie par : $\mathrm{R}_n = \dfrac{u_{n+1}}{u_n}$, $\forall\, n \in \mathbf{N}^*$.

□ **1**— Ecrire une fonction `ListeU(N)` renvoyant la liste des `N` premières valeurs de la suite $(u_n)_{n\in\mathbf{N}}$. Représenter graphiquement les 20 premières valeurs de la suite $(\mathrm{R}_n)_{n\in\mathbf{N}}$.

□ **2**— Tracer le graphe de la fonction polynomiale $\mathrm{P}(x) = x^3 - 2x^2 - 3x + 2$ sur l'intervalle $[-2, 4]$. Remarquer alors que P admet trois racines réelles λ, μ, ν telles que $|\lambda| > |\mu| > |\nu|$.

□ **3**— On admet qu'il existe $(a, b, c) \in \mathbf{R}^3$ tel que : $\forall\, n \in \mathbf{N}$, $u_n = a\,\lambda^n + b\,\mu^n + c\,\nu^n$.

■ **3.1. [à rendre sur feuille séparée]** Démontrer que la suite $(\mathrm{R}_n)_{n\in\mathbf{N}}$ converge vers λ en cherchant un équivalent de $(u_n)_{n\in\mathbf{N}}$.

■ **3.2.** En déduire, avec `Python`, une valeur approchée de λ. On considèrera la valeur *bonne* pour un n assez grand tel que $|\mathrm{R}_{n+1} - \mathrm{R}_n| < 10^{-5}$.

■ **3.3. [Bonus]** Il existe un lien important entre $(u_n)_{n\in\mathbf{N}}$ et le polynôme P : analysez ce lien.

Considérons à présent le polynôme $\mathrm{Q}(x) = x^3\mathrm{P}\left(\dfrac{1}{x}\right) = 1 - 2x - 3x^2 + 2x^3$. Quelles sont les racines de Q ? Quelle est la plus grande à présent ?
En formant une nouvelle suite récurrente $(v_n)_{n\in\mathbf{N}}$, associée au polynôme Q, appliquer la même stratégie pour déterminer une valeur approchée de ν.

■ **3.4. [Bonus]** En déduire une valeur approchée à à 10^{-5} près de μ à partir des deux questions précédentes.

SOLUTION.

□ **1**—

```
## Q1 ##
def ListeU(N):
    L=[0,1,2]
    for i in range(3,N):
        L.append(2*L[-1]+3*L[-2]-2*L[-3])
    return L

import matplotlib.pyplot as plt
def R(N):
    L=[]
    U=ListeU(N+2)
    for i in range(1,N+1):
        L.append(U[i+1]/U[i])
    return L
plt.subplot(211)
```

```
plt.plot(R(20),linestyle = 'none', marker = 'o', c = 'lime',
  markersize = 10)
plt.title('premiers termes de R')
## FinQ1 ##
```

□ **2**—

```
## Q2 ##
import numpy as np
X=np.linspace(-2,4,100)
def P(x):
    return x**3-2*x**2-3*x+2

Y=[P(x) for x in X]
plt.subplot(212)
plt.plot(X,Y)
plt.title('graphe de P')

plt.show()
## FinQ2 ##
```

□ **3**—

■ **3.1.** On forme donc le quotient de deux termes consécutifs. Soit $n \in \mathbf{N}$, alors :
$\dfrac{a\lambda^{n+1}+b\mu^{n+1}+c\nu^{n+1}}{a\lambda^n+b\mu^n+c\nu^n} = \dfrac{\lambda^{n+1}\left(a+b\left(\mu/\lambda\right)^{n+1}+c\left(\nu\lambda\right)^{n+1}\right)}{\lambda^n\left(a+b\left(\mu/\lambda\right)^n+c\left(\nu/\lambda\right)^n\right)} \sim_{n\to\infty} \dfrac{a\lambda^{n+1}}{a\lambda^n} = \lambda$. Nous avons utilisé ici le fait suivant : la condition $|\lambda| > |\mu| > |\nu|$ impose $1 > \left|\dfrac{\mu}{\lambda}\right| > \left|\dfrac{\nu}{\lambda}\right|$. On déduit alors :

$$\lim_{n\to\infty}\left(\frac{\mu}{\lambda}\right)^n = 0 = \lim_{n\to\infty}\left(\frac{\nu}{\lambda}\right)^n.$$

Ainsi, $\boxed{\lim_{n\to\infty} R_n = \lambda}$.
Ce résultat nous donne un moyen d'approcher λ à l'aide de $(R_n)_{n\in\mathbf{N}}$ pour n assez grand.

■ **3.2.**

```
## Q3.2 ##
eps=1e-5
n=0
LU=[0,1,2,3] # liste des u_n
LR=[2,1.5] #valeurs de R_1 et R_2
while abs(LR[n+1]-LR[n])>eps:
    LU.append(2*LU[-1]+2*LU[-2]-3*LU[-3])
    LR.append(LU[-1]/LU[-2])
    n+=1
print('valeur approchee de lambda :',LR[n])
## FinQ3.2 ##
```

■ **3.3.** En résolvant $Q(x) = 0$ dans $\mathbf{R}$, on constate que $Q\left(\dfrac{1}{\lambda}\right) = Q\left(\dfrac{1}{\mu}\right) = Q\left(\dfrac{1}{\nu}\right) = 0$. Comme $\dfrac{1}{\lambda} \neq \dfrac{1}{\mu} \neq \dfrac{1}{\nu}$ et $\deg Q = 3$, les racines de Q sont $\boxed{\dfrac{1}{\lambda}, \dfrac{1}{\mu} \text{ et } \dfrac{1}{\nu}}$, la plus grande est $\dfrac{1}{\nu}$.
En s'inspirant de ce qui précède, nous allons donc former la suite récurrente définie par :

$$v_{n+3} = \frac{1}{2}\left(3u_{n+2} + 2u_{n+1} - u_n\right).$$

D'après le rappel, on a :

$$v_n = a'\left(\frac{1}{\nu}\right)^n + b'\left(\frac{1}{\mu}\right)^n + c'\left(\frac{1}{\lambda}\right)^n, \quad \text{avec} \quad (a', b', c') \in \mathbf{R}^3.$$

Et donc de la même manière qu'en **Q1)**, on établit que $\lim_{n\to\infty}\left(\frac{v_{n+1}}{v_n}\right)=\frac{1}{\nu}$, la plus grande des racines de Q.

Notons que les conditions initiales n'ont aucune incidence sur la limite de $(v_n)_{n\in\mathbf{N}}$ nous reprenons donc $0,1,2$ comme avant.

■ **3.4.**

```
## Q3.4 ##
m=0
LV=[0,1,2,4]
LRprime=[2,1.5]
while abs(LR[n+1]-LR[n])>eps:
    LV.append((3/2)*LV[-1]+(2/3)*LV[-2]-(1/2)*LV[-3])
    LRprime.append(LU[-1]/LU[-2])
    m+=1
print('valeur approchee de nu :',1/LRprime[m])
## FinQ3.4 ##
```

{2-EXAMEN}

***Exercice* 2 – Calculs de probabilités de figures du jeu de Poker**

On considère un jeu de 32 cartes. Il est formé de couples de 8 valeurs ordonnées, `valeurs`=["7", "8", "9", "10", "V", "D", "R", "A"], et de 4 « *couleurs* » , `couleurs`=["T", "K", "C", "P"] (Trèfle , Carreau, Coeur, Pique). On distribue au hasard une « *main* », c'est-à-dire 5 cartes distinctes, et on s'intéresse à des mains particulières :

- ✓ les « *couleurs* » (5 cartes de même « *couleur* »);
- ✓ les « *quintes* » (5 cartes de valeurs qui se suivent);
- ✓ les « *quintes floches* » (5 cartes de même « *couleur* » et de valeurs qui se suivent).

□ **1**— Construire une fonction `CARTES` des 32 cartes à partir des deux listes `valeurs` et `couleurs`, chaque carte étant représentée par la paire [*valeur*, *couleur*].

INDICATION ▸ *On pourra commencer par créer une liste vide, que l'on remplit à l'aide d'une boucle* `for`.

□ **2**— Vérifier que le nombre de cartes obtenu est correct.

□ **3**— Écrire une fonction `TIRERMAIN` sans paramètre qui renvoie une liste de cinq cartes distinctes tirées au hasard.

INDICATION ▸ *On pourra utiliser la fonction* `sample` *du module* `random`. *Essayez de voir ce que donne :*

```
import random
random.sample([1,2,3],2)
```

□ **4**— Écrire une fonction `estCOULEUR` d'argument une main, et qui renvoie un booléen indiquant si cette main est une « *couleur* » ou pas.

□ **5**— Créer une liste `quintes` de toutes les suites possibles de cinq valeurs d'une quinte.

□ **6**— En déduire une fonction `estQuinte` d'argument une main et qui renvoie un booléen indiquant si cette main est une « *quinte* » ou pas.

INDICATION ▸ *Pour comparer deux listes de valeurs indépendamment de leur ordre, on pourra utiliser la fonction* `sorted`. *Testez* `sorted([1,3,2])`.

□ **7**— En déduire ensuite une fonction `estQuinteFloche` d'argument une main et qui renvoie un booléen indiquant si cette main est une « *quinte floche* » ou pas.

□ **8**— À partir de 50000 tirages aléatoires de mains, estimer la probabilité d'obtenir une « *couleur* », celle de tirer une « *quinte* » et celle de gagner une « *quinte floche* ».

SOLUTION.

□ **1**—

```
## Q1 ##
valeurs=['7','8','9','10','V','D','R','A']
couleurs=['T','K','C','P']
import random as rd

def CARTES():
    L=[]
    for v in valeurs:
        for c in couleurs:
            L=L+[[v,c]]
    print(len(L))
    return L
Jeu=CARTES()
len(Jeu)
## FinQ1 ##
```

□ **2**— Voir la question précédente.

□ **3**—

```
## Q3 ##
def TIRERMAIN():
    return rd.sample(Jeu,5)
## FinQ3 ##
```

□ **4**—

```
## Q4 ##
#On se donne donc une liste de 5 couples notée M. On commence par regarder si cette liste est
    de bonne taille.
def estCOULEUR(M):
    if len(M)!=5 or len(M[0])!=2 or len(M[1])!=2 or len(M[2])!=2 or len(M[3])!=2 or len(M[4])
        !=2:
        return "ce n'est pas une main"
    return M[0][1]==M[1][1]==M[2][1]==M[3][1]==M[4][1]
## FinQ4 ##
```

□ **5**—

```
## Q5 ##
quintes=[['7','8','9','10','V'],['8','9','10','V','D'],['9','10','V','D','R'],['10','V','D','R
    ','A'],['V','D','R','A','7'],['D','R','A','7','8'],['R','A','7','8','9'],['A','7','8','9',
    '10']]

quintescodees
    =[[7,8,9,10,11],[8,9,10,11,12],[9,10,11,12,13],[10,11,12,13,14],[11,12,13,14,7],[12,13,14,7,8],[13,14,7,8,9]


## FinQ5 ##
```

□ **6**—

```
## Q6 ##
def estQuinte(M):
    #on extrait les figures
    N=[]
    for i in range(len(M)):
        N=N+[M[i][0]]
    for i in range(len(M)):
        if N[i]=='V':
            N[i]=11
        elif N[i]=='D':
```

```
            N[i]=12
        elif N[i]=='R':
            N[i]=13
        elif N[i]=='A':
             N[i]=14
        else :
            N[i]=int(N[i])
    return sorted(N) in quintescodees
## FinQ6 ##
```

□ **7**—

```
## Q7 ##
def estQuinteFloche(M):
    return estQuinte(M) and estCOULEUR(M)
## FinQ7 ##
```

□ **8**—

```
## Q8 ##
nbcouleur=0
nbquinte=0
nbquintefloch=0
N=100000
for i in range(N):
    M=TIRERMAIN()
    nbcouleur+=int(estCOULEUR(M))
    nbquinte+=int(estQuinte(M))
    nbquintefloch+=int(estQuinteFloche(M))
print('couleur',nbcouleur/N)
print('quinte',nbquinte/N)
print('quinte floche',nbquintefloch/N)
## FinQ8 ##
```

Une simulation m'a donné par exemple les probabilités suivantes : couleur 0.00115, quinte 0.02037, quinte floche 0.00016.