ENSAE mardi 11 décembre 2012

Cette interrogation écrite compte pour 10 points ajoutés à l'ensemble des notes de la matière. Tous les documents sont autorisés. La durée est de 45 minutes. Vous devrez imprimer le résultat en n'omettant pas d'y ajouter votre nom et le numéro de l'énoncé qui vous aura été distribué.

1

1) La ligne suivante génère une suite aléatoire de nombre entiers compris entre 0 et 20 inclus.

```
s = [ random.randint(0,20) for i in range(0,10000) ]
```

Le nombre 0 apparaît en différentes positions, écrire une fonction qui calcule l'écart moyen entre deux positions consécutives. (2 points)

2) On tire quatre nombres au hasard et **sans remise** en utilisant la fonction `random.randint`$(0, 20)$. Calculer la probabilité pour que la somme des deux plus grands nombres soit inférieure à deux fois la somme des deux plus petits nombres (sur 10000 tirages par exemple). (2 points)

$$2 * (x_1 + x_2) \geqslant x_3 + x_4 \text{ avec } x_1 \leqslant x_2 \leqslant x_3 \leqslant x_4$$

3) Ecrire une fonction qui retourne tous les combinaisons C_n^p pour $0 \leqslant p \leqslant n \leqslant 100$ dans un dictionnaire. On utilisera pour cela la formule de récurrence issue du triangle de Pascal $C_n^p = C_{n-1}^{p-1} + C_{n-1}^p$ et le fait que $C_n^p = C_n^{n-p}$.

Rappel : on note aussi $C_n^p = \binom{n}{p}$, et $C_0^0 = 1$, $C_n^0 = C_n^n = 1$.

(3 points)

4) Créer une fonction qui réordonne les coefficients d'une matrice en disposant le plus grand nombre sur la première ligne et la première colonne. Le second nombre sera sur la première ligne, deuxième colonne et ainsi de suite. Elle prend en entrée une matrice (liste de listes) et qui retourne une matrice de même dimension. Par exemple :

$$\begin{pmatrix} 4 & 5 \\ 3 & 7 \\ 1 & 2 \end{pmatrix} \text{ deviendra } \begin{pmatrix} 7 & 5 \\ 4 & 3 \\ 2 & 1 \end{pmatrix}$$

Rappel : si `li` est une liste, `li.sort`() trie cette liste.

(3 points)

Correction

```
#coding:latin-1
# correction exerice 1
import random

# question 1, version 1 (version abrégée)
def ecart_moyen (s) :
    positions = [ j for j,e in enumerate(s) if e == 0 ]
```

```
    dd = [ positions[j] - positions[j-1] for j in range(1,len(positions)) ]
    # une division entière retourne 0, il faut multiplier par 1.0
    return 1.0*sum(dd) / len(dd)

s = [ random.randint(0,20) for i in range(0,10000) ]
print ecart_moyen(s)

# question 1, version 2 (version plus longue)
def ecart_moyen (s) :
    positions = [ ]
    for j in range(len(s)) :
        if s[j] == 0 : positions.append (j)
    ecart = 0
    for j in range(len(positions)-1) :
        ecart += positions[j+1] - positions[j]
    # on
    return ecart*1.0 / (len(positions)-1)

# question 1, version 3 (version courte + astuce)
def ecart_moyen (s) :
    # moyenne des écarts = (dernière position - première position) / (nombre de positions - 1)
    positions = [ j for j,e in enumerate(s) if e == 0 ]
    return (positions [-1] - positions [0])*1.0 / (len(positions)-1)

# question 2, version 1
def tirage_sans_remise () :
    l = [ random.randint(0,20) ]
    while len(l) < 4 :
        x = random.randint(0,20)
        if x not in l : l.append (x)
    return l

def moyenne () :
    s = 0
    for i in range(0,10000) :
        l = tirage_sans_remise ()
        l.sort()
        s1 = l[0]+l[1]
        s2 = l[2]+l[3]
        if s1*2 >= s2 : s+=1
    return s*1.0/10000

print "v1", moyenne()

# question 2, version 2
def tirage_sans_remise () :
    l = range(0,20)
    random.shuffle(l)
    return l[:4]

def moyenne () :
    s = 0
    for i in range(0,10000) :
        l = tirage_sans_remise ()
        l.sort()
        s1 = l[0]+l[1]
        s2 = l[2]+l[3]
        if s1*2 >= s2 : s+=1
    return s*1.0/10000
```

```
print "v2",moyenne()

# question 3
def combinaison (N = 100) :
    # res [ n,p ]
    res = { }
    res [0,0] = res [1,0] = res[1,1] = 1.0
    for n in range (2, N) :
        res [n,0] = 1.0
        res [n,n] = 1.0
        for p in range (1,n/2+1) :
            res [n,p]   = res [n-1,p] + res[n-1,p-1]
            res [n,n-p] = res [n,p]
    return res

if False :  #  pour éviter des affichages trop longs
    d = combinaison (100)
    for n,p in sorted(d) :
        print "c(n=%d,p=%s)=%f" % (n,p,d[n,p])

# question 4
def ordonne (mat) :
    # on concatêne toutes les lignes
    suml = [ ]
    for l in mat : suml += l
    # on trie
    suml.sort()
    # et on crée une autre matrice
    nc = len(mat[0])  #nombre de colonne
    res = [ suml[ i*nc: i*nc+nc] for i in range (0, len(mat)) ]
    return res

l = [[4,5], [3, 7], [1,2] ]
print ordonne(l) # affiche [[1, 2], [3, 4], [5, 7]]
```

2

1) La fonction `random.shuffle` effectue une permutation aléatoire d'une liste. Appliqué à une liste de listes (ou matrice), elle effectue une permutation aléatoire des lignes.

```
li = [[0, 1, 2, 4], [4, 5, 6, 8], [8, 9, 10, 12], [12, 13, 14, 16]]
random.shuffle(li)
print li  # example : [[8, 9, 10, 12], [12, 13, 14, 16], [4, 5, 6, 8], [0, 1, 2, 4]]
```

En utilisant une astuce, écrire une fonction qui effectue une permutation aléatoire des colonnes. (2 points)

2) Ecrire une fonction qui calcule le nombre moyen de voyelles dans une liste de mots. (3 points)

```
l = ["chat", "chats", "chien", "chiens", "cheval", "chevaux" ]
def nombre_moyen_voyelles (l) :
   ...
```

3) On tire quatre nombres au hasard et **avec remise** en utilisant la fonction `random.randint(0, 20)`. Calculer la probabilité pour que la somme des deux plus grands nombres soit inférieure à deux fois la somme des deux plus petits nombres (sur 10000 tirages par exemple). (2 points)

$$2 * (x_1 + x_2) \geqslant x_3 + x_4 \text{ avec } x_1 \leqslant x_2 \leqslant x_3 \leqslant x_4$$

4) On veut savoir combien de fois un texte contient une paire de mots adjacents provenant d'une liste de mots distincts.

```
mots  = ["paris", "texas", "montmartre", "wim", "wenders", "france"]
texte = "paris texas est un film de wim wenders ce paris n'est pas en france mais au texas"]
```

La fonction qu'on doit créer doit retourner 2 pour les deux paires trouvées "paris texas" et "wim wenders". Les paires de type "paris paris" ne sont pas comptées. (3 points)

Correction

```
#coding:latin-1
# correction exerice 2
import random

# question 1, version 1
# l'astuce consiste à transposer la matrice
def shuffle_colonne (li) :
    tr  = [ [ li [j][i] for j in range(len(li[0])) ] for i in range(len(li))]
    random.shuffle(tr)
    res = [ [ tr [j][i] for j in range(len(tr[0])) ] for i in range(len(tr))]
    return res

li = [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11], [12, 13, 14, 15]]
print shuffle_colonne (li)

# question 2
def nombre_moyen_voyelle (l) :
    s = 0   # nombre total de voyelles
    ts = 0  # nombre total de lettres
    for m in l :
        ts += len(m)
        for c in m :
            if c in "aeiouy" : s += 1
    return s * 1.0 / ts

l = ["chat", "chats", "chien", "chiens", "cheval", "chevaux" ]
print "voyelle", nombre_moyen_voyelle (l)  #  0.3333

# question 3
def tirage_avec_remise () :
    l = [ random.randint(0,20) for i in range(0,4) ]
    l.sort()
    return l

def moyenne () :
    s = 0
    for i in range(0,10000) :
        l = tirage_avec_remise ()
```

```
        l.sort()
        s1 = l[0]+l[1]
        s2 = l[2]+l[3]
        if s1*2 >= s2 : s+=1
    return s*1.0/10000

print "v1", moyenne()

# question 4, version 1
def compte_pairs (mots, texte) :
    p = 0
    for m in mots :
        for mm in mots :
            t = m + " " + mm
            if t in texte :
                p += 1
    return p

mots = ["paris", "texas", "montmartre", "wim", "wenders", "france"]
texte = "paris texas est un film de wim wenders ce paris n'est pas en france mais au texas"
print compte_pairs(mots, texte)  # affiche 2

# question 4, version 2
def compte_pairs (mots, texte) :
    dec = texte.split()
    p = 0
    for i in range(1,len(dec)) :
        if dec[i-1] in mots and dec[i] in mots :
            p += 1
    return p

mots = ["paris", "texas", "montmartre", "wim", "wenders", "france"]
texte = "paris texas est un film de wim wenders ce paris n'est pas en france mais au texas"
print compte_pairs(mots, texte)  # affiche 2
```