ENSAE TD noté, mardi 27 novembre 2012

Le programme construit au fur et à mesure des questions devra être imprimé à la fin du TD et rendu au chargé de TD. ***Il ne faut pas oublier de mentionner son nom inséré en commentaire au début du programme et de l'ajouter sur chaque page.*** *Les réponses autres que des parties de programme seront insérées sous forme de commentaires. Les squelettes de fonctions proposés ne sont que des suggestions. Les programmes fournis avec l'énoncé ne devront pas être rendus.* ***Il faudra aussi indiquer le numéro des exercices sous forme de commentaires.***

1

L'objectif de cet exercice est de colorier l'espace situé entre les deux spirales de la figure 1.

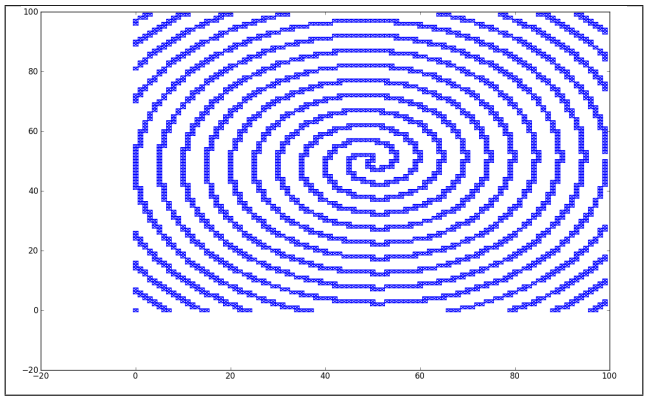

Figure 1 : *Double spirale*

Le code qui a servi à construire ces deux spirales vous est fourni en pièce jointe [1]. Ce code vous est donné, aucune question ne vous sera posée dessus. Il est présent à la fin de l'énoncé de l'exercice. **Afin d'éviter son inclusion (et son impression), votre programme devra impérativement commencer par :**

```
#coding:latin-1
import exoS
matrice = exoS.construit_matrice(100)
```

Pour visualiser la matrice et obtenir la figure 1, il faut écrire :

```
exoS.dessin_matrice(matrice)   # cette fonction place un point bleu pour une case contenant 1,
                               # rouge pour une case contenant 2,
                               # rien si elle contient 0
```

Au départ la matrice retournée par la fonction `construit_matrice` contient soit 0 si la case est vide, soit 1 si cette case fait partie du tracé d'une spirale. L'objectif de cet exercice est de colorier une zone de la matrice.

1) Ecrire une fonction qui prend comme entrée une matrice, deux entiers et qui retourne la liste

1. lien http://www.xavierdupre.fr/enseignement/complements_site_web/td_note_2013_novembre_2012_exoM.py

des voisins parmi les quatre possibles pour lesquels la matrice contient une valeur nulle. On fera attention aux bords du quadrillage. (2 points)

```
def voisins_a_valeur_nulle ( matrice, i, j ) :
    resultat = [ ]
    # ...
    return resultat
```

2) En utilisant la fonction précédente, écrire une fonction qui reçoit une liste de points de la matrice contenant une valeur nulle et qui retourne tous les voisins possibles contenant une valeur nulle pour tous les points de la liste. (2 points)

```
def tous_voisins_a_valeur_nulle ( matrice, liste_points ) :
    resultat = [ ]
    # ...
    return resultat
```

3) On a tous les éléments pour écrire l'algorithme de coloriage :

1. On part d'un point $(i0, j0)$ qui fait partie de la zone à colorier, qu'on insère dans une liste `acolorier` $= [(\texttt{i0}, \texttt{j0})]$.
2. Pour tous les points (i, j) de la liste `acolorier`, on change la valeur de la case (i, j) de 0 à 2.
3. Pour tous les points (i, j) de la liste `acolorier`, on regarde les quatre voisins $(i, j+1), (i, j-1), (i+1, j), (i-1, j)$. Si la matrice contient 0 pour un de ces voisins, on l'ajoute à la liste et on retourne l'étape précédente tant que cette liste n'est pas vide.

En utilisant la fonction précédente, écrire une fonction qui colorie la matrice à partir d'un point $(i0, j0)$. (3 points)

```
def fonction_coloriage( matrice, i0, j0) :
    # ...
```

4) Tester la fonction précédente avec le point de coordonnées $(53, 53)$ puis afficher le résultat avec la fonction `dessin_matrice`. (1 point)

Il y avait deux questions différentes selon les énonces.

5) Enoncé 1 : Ecrire une fonction qui retourne la surface coloriée. (2 points)

```
def surface_coloriee ( matrice ) :
    # ...
    return surface
```

6) Enoncé 4 : Créer une autre fonction identique à celle de la question 3 à ceci près qu'elle doit s'arrêter après que environ 1000 points distincts ont été coloriés. (2 points)

```
def fonction_coloriage_environ_1000 ( matrice, i0, j0 ) :
    # ...
    return surface
```

Programme générant la spirale

```
#coding:latin-1
import math

# cette fonction construit deux spirales imbriquées dans une matrice nb x nb
# le résultat est retourné sous forme de liste de listes
def construit_matrice (nb) :
    mat = [ [ 0 for x in range (0,nb) ] for y in range(0,nb) ]

    def pointij (nb,r,th,mat,c,phase) :
        i,j = r*th * math.cos(th+phase), r*th*math.sin(th+phase)
        i,j = int(i*100/nb), int(j*100/nb)
        i,j = (i+nb)/2, (j+nb)/2
        if 0 <= i < nb and 0 <= j < nb :
            mat[i][j] = c
        return i,j

    r = 3.5
    t = 0
    for tinc in range (nb*100000) :
        t  += 1.0 * nb / 100000
        th  = t * math.pi * 2
        i,j = pointij (nb,r,th,mat,1, 0)
        i,j = pointij (nb,r,th,mat,1, math.pi)
        if i >= nb and j >= nb : break

    return mat

# cette fonction reçoit une matrice sous forme de liste de listes contenant des entiers : 0,1,2
# à chaque valeur est associée une couleur :
# 0 pour blanc, 1 pour bleu, 2 pour rouge
def dessin_matrice (matrice) :
    import pylab
    colors = { 1: "blue", 2:"red"  }
    for i in range(0,len(matrice)) :
        for j in range (0, len(matrice[i])) :
            if matrice [i][j] in colors :
                pylab.plot ([i-0.5,i-0.5,i+0.5,i+0.5,i-0.5,i+0.5,i-0.5,i+0.5],
                            [j-0.5,j+0.5,j+0.5,j-0.5,j-0.5,j+0.5,j+0.5,j-0.5],
                        colors [ matrice[i][j] ])
    pylab.show()

if __name__ == "__main__" :
    matrice = construit_matrice(100)
    dessin_matrice(matrice)
```

Correction

```
#coding:latin-1
import td_note_2013_novembre_2012_exoS as exoS

# question 1, exo S (1 ou 4)
def voisins_a_valeurs_nulle (matrice,i,j) :
    res = []
    if i > 0                 and matrice[i-1][j] == 0 : res.append ( (i-1,j) )
    if i < len(matrice)-1    and matrice[i+1][j] == 0 : res.append ( (i+1,j) )
```

```
    if j > 0                 and matrice[i][j-1] == 0 : res.append ( (i,  j-1) )
    if j < len(matrice[i])-1 and matrice[i][j+1] == 0 : res.append ( (i,  j+1) )
    return res

# question 2, exo S (1 ou 4)
def tous_voisins_a_valeurs_nulle (matrice, liste_points) :
    res = []
    for i,j in liste_points :
        res += voisins_a_valeurs_nulle (matrice, i,j)
    return res

# question 3, exo S (1 ou 4)
def fonction_coloriage ( matrice, i0, j0) :
    # étage 1
    acolorier = [ ( i0, j0 ) ]
    while len (acolorier) > 0 :
        # étape 2
        for i,j in acolorier :
            matrice [i][j] = 2
        # étape 3
        acolorier = tous_voisins_a_valeurs_nulle ( matrice, acolorier )
        # on enlève les doublons car sinon cela prend trop de temps
        d = { }
        for i,j in acolorier : d [i,j] = 0
        acolorier = [ (i,j) for i,j in d ]

# question 5, exo S (version 1)
def surface_coloriee (matrice) :
    surface = 0
    for line in matrice :
        for c in line :
            if c == 2 : surface += 1
    return surface

# question 5, exo S (version 4)
def fonction_coloriage_1000 ( matrice, i0, j0) :
    acolorier = [ ( i0, j0 ) ]
    nb = 0                                  # ligne ajoutée
    while len (acolorier) > 0 :
        for i,j in acolorier :
            matrice [i][j] = 2
            nb += 1                         # ligne ajoutée
        if nb > 1000 : break                # ligne ajoutée
        acolorier = tous_voisins_a_valeurs_nulle ( matrice, acolorier )
        d = { }
        for i,j in acolorier : d [i,j] = 0
        acolorier = [ (i,j) for i,j in d ]

# question 4, exo S (1 ou 4)
matrice = exoS.construit_matrice(100)
fonction_coloriage  (matrice, 53, 53)
exoS.dessin_matrice(matrice)
print surface_coloriee (matrice) # retourne 3258

# question 5, exo S (version 4) vérification
matrice = exoS.construit_matrice(100)
fonction_coloriage_1000  (matrice, 53, 53)
exoS.dessin_matrice(matrice)
```

```
print surface_coloriee (matrice) # retourne 1002
```

2

Certaines questions suggèrent l'utilisation du module `numpy`, ne pas le faire ne sera pas pénalisé si les réponses proposées produisent des résultats équivalents.

Cet exercice utilise des données [2] fournies en pièce jointe. Elles comptabilisent le nombre d'équipements sportifs par canton. Il faudra utiliser le programme également fourni en pièce jointe [3] pour les récupérer sous forme de tableau. Ce code ne sera pas modifié durant l'examen excepté le dernier paramètre transmis à la fonction `construit_matrice` qui permet de ne considérer qu'une partie des données pour tester plus rapidement ses idées sur les premières lignes. On souhaite mesurer si les Français ont accès aux mêmes équipements sportifs sur tout le territoire français. **Afin d'éviter son inclusion (et son impression), votre programme devra impérativement commencer par :**

```
#coding:latin-1
import exoM
fichier_zip   = exoM.import_module_or_file_from_web_site ("equipements_sportif_2011.zip")
fichier_texte = exoM.unzip_fichier (fichier_zip)
# enlever le dernier paramètre 500 pour avoir le tableau complet
colonne, intitule, variables = exoM.construit_matrice (fichier_texte, 500)
    # colonne   : contient le nom des colonnes
    # intitule  : contient les deux premières colonnes du fichier textes avec du texte
    # variables : contient les autres colonnes avec des valeurs numériques
```

Les lignes suivantes permettent de convertir les informations extraites en un tableau `numpy` [4].

```
import numpy
intitule  = numpy.array(intitule)  # array et non matrix
variables = numpy.array(variables) # array et non matrix

# utilisation de numpy pour sélectionner des lignes spécifiques
print intitule [ intitule[:,1] == "Chevroux", : ]  # affiche [['01102' 'Chevroux']]
print variables[ intitule[:,1] == "Chevroux", : ]  # affiche [[  82.    1.   12 ...
```

1) Le tableau `intitule` a deux colonnes : le code postal et la ville. On veut créer un tableau `intitule3` qui contient trois colonnes : celles de `intitule` et le département déduit du code postal. Quelques fonctions utiles : (2 points)

```
print tab.shape                    # si tab est une matrice ou un tableau numpy à deux dimensions,
                                   # tab.shape est un couple (nb_lignes, nb_colonnes)
a = numpy.column_stack ( ( m, e ) ) # coller deux matrices, tableaux ayant le même nombre de lignes
```

Au final, la ligne ['01008','Ambutrix'] deviendra ['01008','Ambutrix','01'].

2) Construire la liste des départements, cette liste contient une unique instance du code du département. Elle doit être triée par ordre croissant. Il n'est pas recommandé d'utiliser `numpy` pour cette

2. lien http://www.xavierdupre.fr/enseignement/complements_site_web/equipements_sportif_2011.zip
3. lien http://www.xavierdupre.fr/enseignement/complements_site_web/td_note_2013_novembre_2012_exoM.py
4. Documentation : http://docs.scipy.org/doc/numpy/reference/

question. En anglais, tri se dit *sort*. (2 points)

3) Construire un tableau D de dimension $d \times v$ où d est le nombre de départements distincts, v est le nombre de variables (normalement 105). Le coefficient D_{ij} est la somme des valeurs pour la variable j et le département i. Si A désigne le tableau `variables`, B le tableau à trois colonnes de la question 1, C la liste des départements distincts (question 2) :

$$D_{ij} = \sum_{k|B_{k,3}=C_i} A_{kj}$$

L'objectif de cette question est d'agréger des données par départements alors qu'elles sont disponibles par canton. (3 points)

Remarque : l'instruction suivante pourrait être utile.

```
# crée une matrice de dimension nb_lignes x nb_colonnes initialisés à zéro
mvide = numpy.zeros ( ( nb_lignes, nb_colonnes) )
```

4) La colonne 5 du tableau D (la première colonne a l'indice 0) contient la population. Créer un autre tableau E qui vérifie : $E_{ij} = D_{ij}/D_{i5}$. (1 point)

5) Le programme fourni en pièce jointe contient une fonction `coefficient_gini` qui calcule le coefficient de Gini [5]. On l'utilise pour comparer le nombre d'équipements par habitants. Il vaut 0 si ce ratio est constant quelque soit le département, il vaut 1 si un seul département propose un certain type d'équipement. Entre 0 et 1, il indique l'inégalité de la distribution. Quel est l'équipement sportif le plus inégalitairement réparti sur tout le territoire ? (2 points)

Remarque : les lignes suivantes pourront vous aider.

```
li = list ( mat [:,i] )           # convertit une colonne d'un tableau numpy en une liste
print colonne[0][i+2]             # affiche le label de la colonne i
gini = exoM.coefficient_gini (li) # retourne le coefficient de Gini
                                  # pour la liste li
```

Programme préparant les données :

```
#coding:latin-1
import math, sys

# extrait les données depuis un site internet, puis les écrit à côté du programme
# ne fait rien si le fichier a déjà été téléchargé
def import_module_or_file_from_web_site (module) :
    import os
    if os.path.exists ("data\\equipement_sportifs_2011\\" + module) :
        return "data\\equipement_sportifs_2011\\" + module
    if not os.path.exists (module) :
        url = "http://www.xavierdupre.fr/enseignement/complements/" + module
        import urllib2
        if module.lower().endswith("zip") :
            f = urllib2.urlopen (url, "rb")
            t = f.read()
```

5. http://fr.wikipedia.org/wiki/Coefficient_de_Gini

```
            f.close()
            f = open(module, "wb")
            f.write(t)
            f.close()
        else :
            f = urllib2.urlopen (url)
            t = f.read()
            f.close()
            f = open(module, "w")
            f.write(t)
            f.close()
    return module

# extrait le fichier texte contenu dans le fichier zip
# et l'enregistre à côté du programme
# ne fait rien si cela est déjà fait
def unzip_fichier (fichier_zip) :
    import zipfile, os
    file = zipfile.ZipFile (fichier_zip, "r")
    res = None
    for info in file.infolist () :
        filename = info.filename
        res = filename
        if not os.path.exists (filename) :
            data = file.read(filename)
            f = open (filename,"w")
            if sys.version.startswith("3.") :
                data = str (data, encoding="iso-8859-1")
                data = data.replace("\r","").split("\n")
                data = [ _ for _ in data if len (_) > 1 ]
                data = "\n".join(data)
            f.write (data)
            f.close()
    file.close ()
    return res

# construit le tableau extrait du fichier précédent
# les deux premières lignes contiennent la description des colonnes
# les autres lignes contiennent les données elles-même
# pour aller plus vite à chaque exécution, on peut limiter le nombre de lignes
# il faudra toutes les utiliser pour l'exécution final
def construit_matrice (fichier, stop_apres = -1) :
    def float_except(x) :
        try : return float(x)
        except : return -1
    f = open (fichier, "r")
    lines = [ line.replace("\n","").split("\t")[:107] \
            for line in f.readlines()[:stop_apres] ]
    f.close ()
    colonne = lines [:2]
    lines = lines [2: ]
    lines = [ line [:2] + [ float_except(x) for x in line [2:] ] \
            for line in lines if len(line)>5 ]
    intitule = [ line[:2] for line in lines ]
    lines = [ line[2:] for line in lines ]
    return colonne, intitule, lines

def coefficient_gini (valeurs) :
    #voir http://fr.wikipedia.org/wiki/Coefficient_de_Gini
```

```
    valeurs.sort()
    gini = 0
    s = 0
    for (i,v) in enumerate (valeurs) :
        gini += (i+1)*v
        s += v
    gini = 2*gini / (len(valeurs)*s) - (len(valeurs)+1.0)/len(valeurs)
    return gini

if __name__ == "__main__" :
    fichier_zip   = import_module_or_file_from_web_site ("equipements_sportif_2011.zip")
    fichier_texte = unzip_fichier (fichier_zip)

    # enlever le dernier paramètre 500 pour avoir le tableau complet
    colonne, intitule, variables = construit_matrice (fichier_texte, 500)

    import numpy
    intitule  = numpy.array(intitule)
    variables = numpy.array(variables)

    # affichage des colonnes
    for i in range (len(colonne[0])) : print (i,colonne[0][i], " --- ", colonne[1][i])

    # utilisation de numpy pour sélectionner des lignes spécifiques
    print (intitule  [ intitule[:,1] == "Chevroux", : ])
    print (variables [ intitule[:,1] == "Chevroux", : ])
```

Correction

```
#coding:latin-1
import numpy
import td_note_2013_novembre_2012_exoM as exoM

fichier_zip   = exoM.import_module_or_file_from_web_site ("equipements_sportif_2011.zip")
fichier_texte = exoM.unzip_fichier (fichier_zip)

# enlever le dernier paramètre 500 pour avoir le tableau complet
colonne, intitule, variables = exoM.construit_matrice (fichier_texte)

import numpy
intitule  = numpy.array(intitule)
variables = numpy.array(variables)

# question 1, exo M (2 ou 3)
code_postaux = [ intitule[i,0] [:2] for i in range (intitule.shape[0] ) ]
intitule3    = numpy.column_stack ( (intitule, code_postaux) )

# question 2, exo M (2 ou 3)
comptage = {}
for i in range (intitule3.shape[0]) :
    comptage [intitule3 [i,2] ] = 0
departements = [ k for k in comptage ]
departements.sort()

# question 3, exo M (2 ou 3)
D = numpy.zeros ( (len(departements), variables.shape[1] ) )
for i in range (len (departements)) :
    d = departements [i]
```

```
    for j in range (variables.shape[1]) :
        D [i,j] = variables [ intitule3 [:,2] == d, j ].sum()

# question 4, exo M (2 ou 3)
E = numpy.zeros ( D.shape )
for i in range (E.shape[0]) :
    E [i,:] = D[i,:] / D[i,5]

# question 5, exo M (2 ou 3)
ginis = []
for j in range (E.shape[1]) :
    li = list ( E [:,j] )
    gini = exoM.coefficient_gini (li)
    ginis.append ( (gini, colonne[0][j+2]) )
ginis.sort ()
for line in ginis : print line

# les dernières lignes du tableau sont :
#(0.86910090569180598, 'Domaine skiable')
#(0.88139092467853186, 'Sports nautiques avec au moins une aire de pratique couverte')
#(0.89326137963164931, 'Domaine skiable - nombre de pistes')
#(0.9348918282098031,  'Parcours sportif avec au moins un parcours couvert')
#(0.93902978850018792, 'Domaine skiable avec au moins une piste \xe9clair\xe9e')
#(0.94625459043715754, '\xc9quipement de cyclisme avec au moins une piste couverte')
#(0.95743849241598267, 'Sports nautiques - nombre de places en tribune')
#(0.97248425032547758, 'Domaine skiable avec au moins une piste couverte')
#(0.97718065858676906, 'Parcours sportif - nombre de places en tribune')
#(0.98637386313881081, 'Terrain de golf - nombre de places en tribune')
#(0.98969072164948457, 'Domaine skiable - nombre de places en tribune')
```