Informatique en CPGE (2015-2016)
Corrigé TD 2 : écriture et lecture dans un fichier
utilisation des bibliothèques Python

Exercice 1
Soit f définie sur $[0\ ;\ 5]$ par $f(x) = \sin(4x)\exp(-x^2+3x)$.

Partie A

1. Ecrire la définition de la fonction f.
2. Construire la liste **lx** des abscisses x variant de 0 à 5 avec un pas de 0.01 puis la liste **ly** des ordonnées correspondantes $y = f(x)$.
3. Tracer la courbe représentative de la fonction f.
4. Ecrire dans un fichier **"data.txt"** les coordonnées x et y des deux listes précédentes. Chaque ligne du fichier contiendra deux nombres x et y séparés par une tabulation ("\t").
5. Vérifier le contenu du fichier **"data.txt"**.

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.sin(4*x)*np.exp(-x**2+3*x)
lx=np.linspace(0,5,501)
ly=f(lx)

plt.plot(lx,ly)
plt.show()

fic=open('data.txt','w')
for i in range(len(lx)):
    fic.write(str(lx[i])+"\t"+str(ly[i])+"\n")
fic.close()
```

Partie B
On souhaite constituer un échantillon des données du fichier **"data.txt"**.

1. Ouvrir le fichier **"data.txt"** en lecture et construire la liste **Lx** des abscisses et la liste **Ly** des ordonnées en prenant une ligne sur dix du fichier (les lignes 0, 10, 20, ...).
2. Afficher sur deux colonnes "abscisses" et "ordonnées" le contenu des deux listes. Les nombres seront arrondis à 10^{-4} près.
3. Sur une même figure, tracer à l'aide des listes **lx** et **ly** la courbe représentative de la fonction f, et placer les points dont les abscisses et ordonnées sont contenues respectivement dans les listes **Lx** et **Ly**.

```
fic=open('data.txt','r')
Lx=[]
Ly=[]
i=0
for ligne in fic:
    if i%10==0:
        coord=ligne.split('\t')
        Lx.append(float(coord[0]))
        Ly.append(float(coord[1]))
    i+=1
fic.close()
print("abscisses | ordonnées")
for i in range(len(Lx)):
    print(round(Lx[i],4),"\t  |",round(Ly[i],4))
plt.plot(lx,ly)
plt.plot(Lx,Ly,'ro')
plt.show()
```

Partie C

1. Afin de résoudre par **dichotomie** l'équation $f(x) = 0$ sur un intervalle $[a\ ;\ b]$ à ϵ près, définir une fonction **dichotomie(f,a,b,eps)** qui prend en argument une fonction f, les bornes a et b d'un intervalle et la précision "eps" et qui renvoie la solution approchée α.
2. Utiliser cette fonction pour déterminer la solution α strictement positive de l'équation $f(x) = 0$ sur l'intervalle $[0\ ;\ 1]$ à 10^{-5} près.
3. Comparer le résultat avec celui obtenu par la fonction **bisect** puis par la fonction **newton** de la sous-bibliothèque **scipy.optimize**.
   Utilisation : **bisect(f,a,b)** et **newton(f,a)** (attention à la valeur de a).

```
from scipy.optimize import bisect, newton

def dicho(f,a,b,eps):
    while b-a>eps:
        m=(a+b)/2
        if f(a)*f(m)<0:
            b=m
        else:
            a=m
    return (a+b)/2
print(dicho(f,0.1,1,0.00001))
print(bisect(f,0.1,1))
print(newton(f,0.7)) #attention à la valeur de a !
```

Partie D

On souhaite déterminer une valeur approchée de l'intégrale I de f sur $[0\ ;\ \alpha]$.

1. Tracer la courbe repésentative de la fonction f sur l'intervalle $[0\ ;\ \alpha]$.
2. Ecrire une fonction **trapeze** qui prend en argument deux listes $x = (x_i)$ et $y = (y_i)$ de même longueur n et renvoie une valeur approchée de I par la méthode des trapèzes. Déterminer alors une valeur approchée de I. Rappel :

$$I \simeq \sum_{i=0}^{n-2} (x_{i+1} - x_i)\, \frac{y_{i+1} + y_i}{2}$$

3. Comparer le résultat avec celui obtenu par la fonction **trapz** puis par la fonction **quad** de la sous-bibliothèque **scipy.integrate**.
   Utilisation : **trapz(ly,lx)** (attention à l'ordre) et **quad(f,a,b)**.

```
from scipy.integrate import trapz, quad

alpha=dicho(f,0.1,1,0.00001)
lx=np.linspace(0,alpha,101)
ly=f(lx)
plt.plot(lx,ly)
plt.show()

def trapeze(lx,ly):
    s=0
    for i in range(len(lx)-1):
        s+=(lx[i+1]-lx[i])*(ly[i+1]+ly[i])/2
    return s
print(trapeze(lx,ly))
s2=trapz(ly,lx)
print(s2)
s3=quad(f,0,alpha)
print(s3)
```

Exercice 2

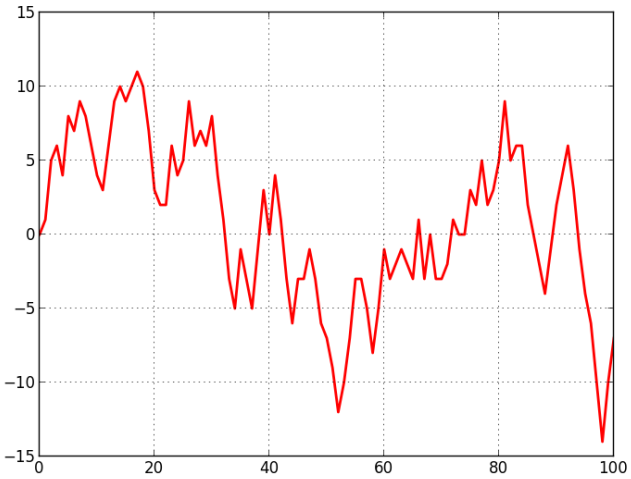
Etude d'une marche aléatoire : pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n + p_n$ avec $u_0 = 0$ et p_n un entier relatif aléatoire choisi de manière équiprobable entre -4 et 4, bornes comprises.

1. Ecrire un programme qui demande à l'utilisateur d'entrer le nom d'un fichier avec une extension "txt", par exemple "marche_alea.txt".
2. Compléter le programme afin d'écrire dans le fichier 101 lignes où sur chaque ligne seront écrites l'abscisse et l'ordonnée d'un point de coordonnées $(n\ ;\ u_n)$ pour n entier variant de 0 à 100.
3. Exécuter le programme et vérifier le contenu du fichier créé.
4. Compléter le programme afin d'obtenir sur une figure les points précédents reliés par des segments.

```
from random import randint

nom=input("nom du fichier ? ")
fic=open(nom,'w')
fic.write('0'+'\t'+'0'+'\n')
y=0
for i in range(1,101):
    y+=randint(-5,5)
    fic.write(str(i)+'\t'+str(y)+'\n')
fic.close()

import matplotlib.pyplot as plt
lx=[i for i in range(101)]
y=0
ly=[y]
for i in range(1,101):
    y+=randint(-4,4)
    ly.append(y)
plt.plot(lx,ly,"red",linewidth=2)
```

```
plt.grid()
plt.savefig("marche.png")
plt.show()
```

Exemple de figure :



Exercice 3

Il s'agit d'écrire un programe permettant d'obtenir des figures illustrant la construction graphique des termes d'une suite récurrente définie par u_0 et $u_{n+1} = f(u_n)$. La fonction f est définie sur $[0; 1]$ par $f(x) = ax(1 - x)$ avec $a \in]0; 4]$ et $u_0 \in]0; 1[$.

1. Le programme demande à l'utilisateur les valeurs de a, de u_0, et le nombre maximal d'itérations noté imax.
2. Ecrire la définition de la fonction f qui prend en argument x et renvoie la valeur de $f(x)$.
3. Tracer sur une même figure la courbe représentant f sur l'intervalle $[0\ ;\ 1]$ et la bissectrice (droite d'équation $y = x$).

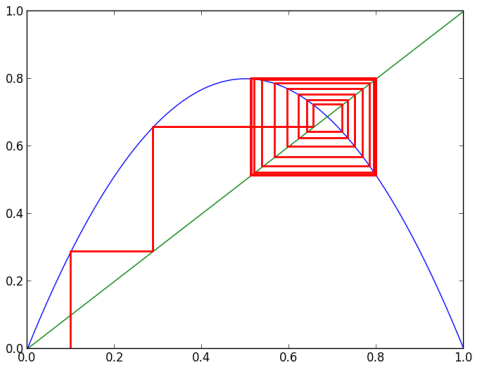
   Vérifier le bon fonctionnement du programme.
4. Pour le tracé des itérations, on crée deux listes contenant respectivement les abscisses et les ordonnées des points à placer. On commencera par écrire les coordonnées du premier point $(u_0; 0)$; puis, à l'aide d'une boucle, on complétera les deux listes avec les coordonnées des points $(x; y)$ et $(y; y)$ avec $y = f(x)$.
5. Tester le programme avec $a = 3,2$, $u_0 = 0,1$ et imax$= 10$.

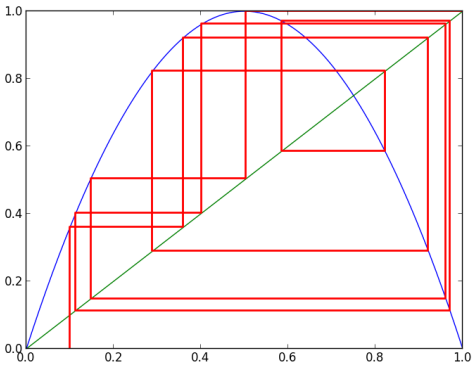
   Tester ensuite différentes valeurs pour u_0 et a, par exemple $a = 1$ avec $u_0 = 0,3$ ou $u_0 = 0,7$, puis $a = 1,6$, $a = 2,8$, $a = 3,2$ et $a = 4$. Déterminer des valeurs de a qui correspondent à différents types de graphiques (escalier, spirale, cycle, chaos).

```
a=float(input("Entrer la valeur de a (entre 0 et 4): "))
u0=float(input("Entrer la valeur de u0 (entre 0 et 1): "))
imax=int(input("Entrer le nombre d'itérations n (entre 1 et 20): "))

def f(x):
    return a*x*(1-x)
```

```
#tracé de la courbe
x=np.linspace(0,1,200)
y=f(x)
plt.plot(x,y)

#tracé de la bissectrice
plt.plot([0,1],[0,1])

#tracé des itérations
x=[u0]
y=[0]
# l'indice -1 renvoie le dernier élément de la liste
for i in range(imax):
    x.append(x[-1])
    y.append(f(x[-1]))
    x.append(y[-1])
    y.append(y[-1])

plt.plot(x,y,linewidth=2)
plt.savefig("iterations.png")
plt.show()
```

On obtiendra en particulier des figures du type suivant :

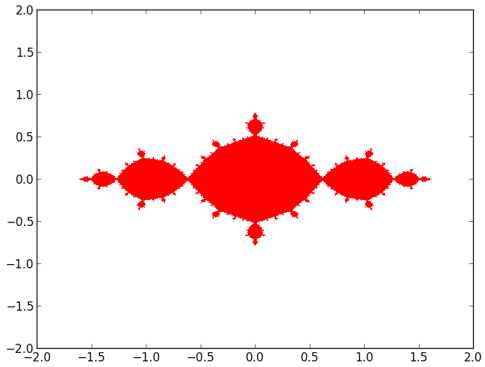


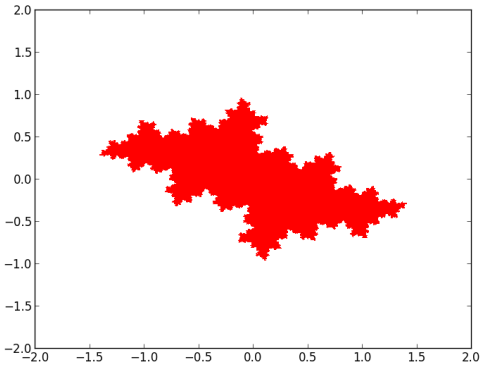


Exercice 4
Itérations dans le plan complexe :
pour $z_0 \in \mathbb{C}$ et $c \in \mathbb{C}$, on calcule les itérés $z_{n+1} = f(z_n) = z_n^2 + c$ pour $n \geq 0$.

1. On fixe le nombre complexe c$= -1$, (instruction : c=complex(-1, 0)) et un nombre d'itérations maximal **imax=20**.
   (a) Ecrire la définition d'une fonction **niveau_julia** qui prend en argument un complexe z et calcul les itérés de z tant que le nombre d'itérations est inférieur à **imax** et tant que le module des itérés est inférieur à 10. La fonction renvoie le nombre d'itérations effectuées.
   (b) Construction de l'ensemble de Julia pour c$= -1$: soit $z = x + iy$ où x prend 801 valeurs équidistantes comprises entre -2 et 2, et y prend 401 valeurs équidistantes comprises entre -1 et 1 ; on colore en rouge le point de coordonnées $(x\ ;\ y)$ si **niveau_julia(z)=imax**. On utilisera l'instruction **plt.plot([x] , [y] , ',' , color='red')** où ',' permet de tracer des pixels.
   (c) Construire l'ensemble de Julia pour c$= -0.5 + .05$i.

   On obtiendra les figures suivantes :
2. Construction de l'ensemble de Mandelbrot (ensemble des valeurs de c pour lesquelles les itérés de 0 ne s'échappent pas à l'infini).





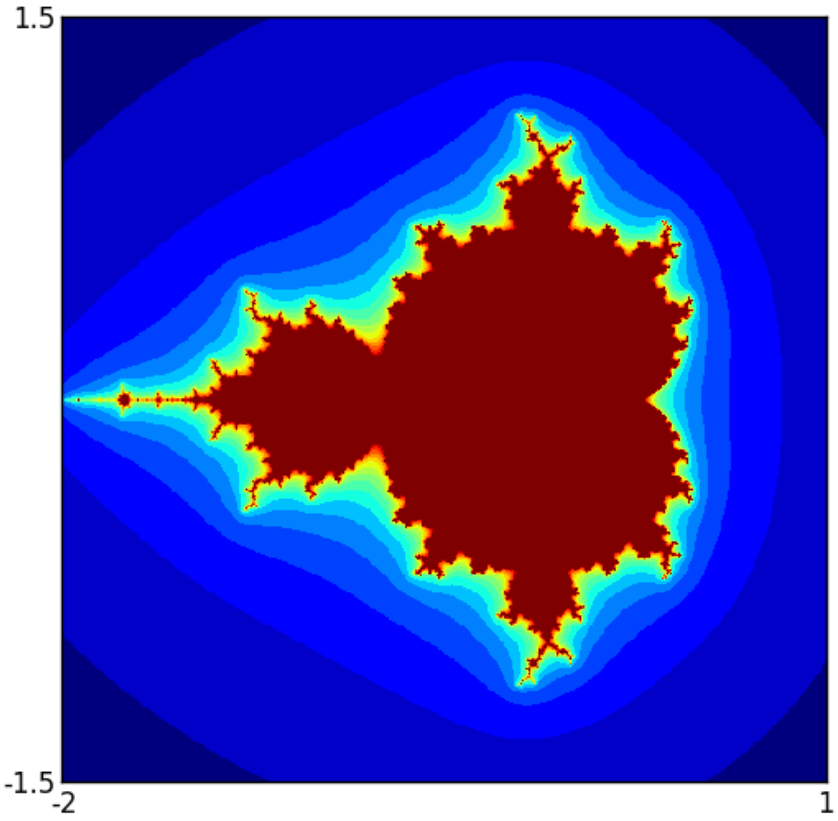
(a) On fixe maintenant $z_0 = 0$ et le nombre d'itérations maximal est toujours **imax=20**.
On s'intéresse aux valeurs de c pour lesquelles le module des itérés dépasse une certaine valeur.

(b) Ecrire la définition d'une fonction **niveau_mandelbrot** qui prend en argument un complexe c et calcul les itérés de $z_0 = 0$ tant que le nombre d'itérations est inférieur à **imax** et tant que le module des itérés est inférieur à 100. La fonction renvoie le nombre d'itérations effectuées.

(c) Représentation de l'ensemble de Mandelbrot : on colore chaque point du plan de coordonnées $(x\ ;\ y)$ selon la valeur renvoyée par la fonction **niveau_mandelbrot**. Soit $c = x + iy$ où x prend 512 valeurs équidistantes comprises entre -2 et 2, et y prend 512 valeurs équidistantes comprises entre $-1,5$ et $1,5$; on construit une matrice **mat** avec mat[i][j] prenant la valeur renvoyée par la fonction **niveau_mandelbrot(c)** où x_c est la ième valeur des 512 abscisses définies ci-dessus et y_c la j-ème valeur des 512 orconnées.
On utilise alors la fonction **matshow** de matplotlib après avoir transformé la matrice en tableau de type ndarray.
Les instructions sont :

```
matrice=np.array(mat)
plt.matshow(matrice)
plt.show()
```

On obtient la figure suivante :


1.5
-1.5
-2
1