TP 4 : Représentations graphiques - Corrigé

2 BCPST 2 - TP Informatique

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


def droite_tangente(x0):
    x = np.linspace(-10, 10, 100)
    y = np.sin(1/x)
    z = (x-x0)*(-1/x0**2)*np.cos(1/x0) + np.sin(1/x0)

    plt.plot(x,0*x,'k', linewidth = 0.5)
    plt.plot(0*y,y,'k', linewidth = 0.5)

    plt.plot(x, y)
    plt.plot(x, z)
    plt.plot(x0, np.sin(1/x0), 'xr' )


    plt.ylim(-np.pi/2,np.pi/2)


plt.subplot(2, 2, 1) ; droite_tangente(1)
plt.subplot(2, 2, 2) ; droite_tangente(2)
plt.subplot(2, 2, 3) ; droite_tangente(0.5)
plt.subplot(2, 2, 4) ; droite_tangente(-0.65)

#%% EX 2
plt.xlim(-5,5)
plt.ylim(-20,20)
x = np.linspace(-10, 10, 200)
plt.plot(x,1/x)
plt.plot(x,x**2)

plt.plot(x,-4*x-4)

plt.plot(-2,4,"xr")
plt.plot(-1/2,-2,"xr")

plt.plot([-10,10],[0,0],"k", linewidth = 0.5)
plt.plot([0,0],[-20,40],"k", linewidth = 0.5)
```

```
#%% Q4

X,Y = [],[]
n = 100; s = 0

for i in range(1,n):
    s += 1/i**2
    X.append(i)
    Y.append(s)

plt.plot(X,Y,"xb")

# Q5

a = np.pi**2/6
plt.plot([0,n],[a,a],"r")

plt.show()

#%%
#Q6

X = np.linspace(-10,10,1000)
Y = 1/np.sin(X)

plt.axis("equal")

plt.xlim(-10,10)
plt.ylim(-10,10)

plt.plot(X,0*X,"k", linewidth = 0.5)
plt.plot(0*Y,X,"k", linewidth = 0.5)
plt.plot(X,Y)
```

```
#%%  Q7

plt.axis("equal")

plt.xlim(-10,10)
plt.ylim(-10,10)


plt.plot([-10,10],[0,0],"k", linewidth = 0.5)
plt.plot([0,0],[-10,10],"k", linewidth = 0.5)

eps = 0.1

a = -10; b = 0
while(b > a):
    b -= np.pi
b += np.pi

while(a < 10):
    X = np.linspace(a+eps,b-eps,100)
    Y = 1/np.sin(X)
    plt.plot(X,Y, "b")
    a = b; b+= np.pi
    if(b > 10):
        b = 10
    print(a,b)

plt.show()
#%% Q8
def trace(a,b):


    plt.xlim(a,b)
    plt.ylim(-10,10)


    plt.plot([a,b],[0,0],"k", linewidth = 0.5)
    plt.plot([0,0],[-10,10],"k", linewidth = 0.5)

    eps = 0.1

    a0 = a; b0 = 0
    if(a < 0):
        while(b0 > a0):
            b0 -= np.pi
        b0 += np.pi
    else :
        while(b0 < a0):
            b0 += np.pi

    while(a0 < b):
        X = np.linspace(a0+eps,b0-eps,100)
        Y = 1/np.sin(X)
        plt.plot(X,Y, "b")
        a0 = b0; b0+= np.pi
        if(b0 > b):
            b0 = b

    plt.show()
```

```
trace(15,19)

#%%

#Exercice 5
def f(x) : return np.sqrt(x)


x = np.linspace(0, 10, 100)
y = f(x)

def ligne_brisee(u0):
    n = 100
    u = u0*np.ones(n)

    for i in range(1,n):
        u[i] = f(u[i-1])

    U = [] # U est la liste des abscisses de la ligne brisee
    V = [] # V est la liste des ordonn es de la ligne brisee

    for i in range(n-1):
        U.append(u[i]) ; V.append(u[i])
        U.append(u[i]) ; V.append(u[i+1])

    U.append(u[n-1]) ; V.append(u[n-1])

    plt.plot(x,y) # fonction f
    plt.plot(x,x) # droite y = x

    plt.xlim(0,max(u0,2)+0.5) #limite de x en fonctions des valeurs min et
   max atteintes
    plt.ylim(0,max(u0,2)+0.5) #limite de y en fonctions des valeurs min et
   max atteintes

    plt.plot(U,V, '--r') #Ligne bris e

    plt.plot(1,1,"or") # Point d'arriv e
    plt.plot(u0,u0,"xr") # Point de d part


# On va afficher les subplot    l'aide d'une boucle (pour ne pas tout
   r   crire)
# On mets donc les valeur    tester dans une liste :

l = [10,0.1, 0.5 , 5 ]

for i in range(4):
    plt.subplot(2, 2, i+1)
    ligne_brisee(l[i])

plt.show()
```

```
#%%

def f(x) : return 1/(1+x)


x = np.linspace(-10, 10, 100)
y = f(x)

def ligne_brisee(u0):
    n = 100
    u = u0*np.ones(n)

    for i in range(1,n):
        u[i] = f(u[i-1])

    U = [] ; V = []
    for i in range(n-1):
        U.append(u[i]) ; V.append(u[i])
        U.append(u[i]) ; V.append(u[i+1])

    U.append(u[n-1]) ; V.append(u[n-1])

    plt.plot(x,0*x,'k', linewidth = 0.5) #axe x
    plt.plot(0*x,x,'k', linewidth = 0.5) #axe y

    plt.plot(x,y) # fonction f
    plt.plot(x,x) # droite y = x

    plt.xlim(min(u0,0)-0.5,max(u0,2)+0.5) #limite de x en fonctions des
   valeurs min et max atteintes
    plt.ylim(min(u0,0)-0.5,max(u0,2)+0.5) #limite de y en fonctions des
   valeurs min et max atteintes

    plt.plot(U,V, '--r') # Ligne bris e

    x0 = (-1+np.sqrt(5))/2
    plt.plot(x0,x0,"or") # Point d'arriv e
    plt.plot(u0,u0,"xr") # Point de d part


l = [9,-5, 0 , 5 ]

for i in range(4):
    plt.subplot(2, 2, i+1)
    ligne_brisee(l[i])

plt.show()


#%% Exercice 6
plt.axis("equal")
Th = np.linspace(0,100,1000)
X = Th*np.cos(Th)
Y = Th*np.sin(Th)

plt.plot(X,Y)
```