TP 8 CODAGE DES NOMBRES

#%% Des grands nombres

''' 1) Flottants limités, entiers illimités'''

```
n = 0
while n < 1001:
   print('10 ^',n,' :',10**n)
   n += 1

n = 0
while n < 1001:
   print('10 ^',n,' :',10.**n)
   n += 1
```

#%% Nombres réels, décimaux

'''1) Écriture décimale de phi'''

```
from math import sqrt
phi = ( 1 + sqrt(5) ) / 2
print('phi :' , phi)
print('phi arrondi :' , round(phi,6))
```

'''2) Approximation fractionnaire de pi'''

```
from math import pi, floor, ceil
ecart = pi - 3
for j in range(100,10001):
   for i in range(floor(3.14*j),ceil(3.15*j)):
      if abs(i/j - pi) < ecart:
         approx = [i,j] ; ecart = abs(i/j - pi)
print('pi =',pi)
print(approx[0],'/',approx[1],'=',approx[0]/approx[1])
print('écart relatif : ',( approx[0]/approx[1]-pi)/pi)
```

#%% Calculs sur les flottants

'''3.1) limite float D.P. vers + infty'''

```
x = 1.
while x < 10*x:
   x *= 10 ; print(x)
#%%
n = 1
while float(2**n):
   print(n) ; n += 1
```

#%%

'''3.2) limite float D.P. vers 0'''

```
x = 1.
while x > x/10:
   x /= 10 ; print(x)
'''plus petit normalisé, avec mantisse à 1 : 1 x 2^(-1022) = 10^(-308)
rq : en dénormalisant on met 0 avant la virgule de la mantisse et pour tous les bits sauf le dernier : 1/2^52 x 2^(-1022) = 5.10^(-323)
'''
```

#%%

'''4.3) plus petit écart relatif entre float D.P.'''

```
x = 77 # comparer pour différentes valeurs !
e = 1 ; y = x + e
while y != x:
   e /= 2 ; y = x + e
print(e/x)
```

#%%

'''4.4) Successeur et prédécesseur de 1'''

```
x = 1. ; e = 1.
while x + e > x:
   e /= 2
print(e*2)
x = 1. ; e = 1.
while x - e < x:
   e /= 2
print(e*2)
```

#%% Erreurs classiques

'''4.1) Comparaison à 0'''

```
import math as m  ; import numpy as np
print(m.sin(m.pi) == 0) ; print(np.sin(np.pi) == 0)
print(m.cos(m.pi/2) == 0) ; print(np.cos(np.pi/2) == 0)

print(abs(m.sin(m.pi)) < 1e-10) ; print(abs(np.sin(np.pi)) < 1e-10)
```

```
#%%
'''4.2) Absorption'''
n = 1 ; a = 10. ; b = -a
while (a+b) + 1 == a + (b+1):
    n += 1 ; a *= 10 ; b = -a
print(n-1)

#%%
'''exemple de l'ordre de calcul d'une série harmonique'''
S1 = S2 = 0. ; fin = 10**6
for n in range(1,fin + 1):
    S1 += 1./n ; S2 += 1/(fin+1-n)
print('n :',n,' ; S1 :',S1)
print('n :',n,' ; S2 :',S2)

#%%
'''4.3) Cancellation : équation du second degré'''
from math import sqrt
def res1(a):
    D = a*a+4
    x1 = (-a-sqrt(D))/2 ; x2 = (-a+sqrt(D))/2
    return (x1,x2)
for a in [1e2,1e4,1e6,1e8,1e10,1e12,1e14,1e16]:
    X1 , X2 = res1(a)[0] , res1(a)[1]
    print('x1 :',X1,' x2 :',X2)
#%%
def res2(a):
    D = a*a+4
    x1 = (-a-sqrt(D))/2 ; x2 = -1/x1
    return (x1,x2)
for a in [1e2,1e4,1e6,1e8,1e10,1e12,1e14,1e16]:
    X1 , X2 = res2(a)[0] , res2(a)[1]
    print('x1 :',X1,' x2 :',X2)
```

#%% Incertitudes de calcul

```
'''5.1) Pendule et bifurcation
On travaille sur un vecteur Y = [theta, d(theta)/dt]
l'E.D. du mouvement theta'' + (g/L) sin(theta)
devient dY = [ Y[1] , -(g/L) sin(Y[0]) ]
'''
import matplotlib.pyplot as plt
import numpy as np

g0 = 9.81 ; l0 = 1.00
Vpi = 2 * np.sqrt(g0/l0)
# vitesse angulaire pour arriver à pi avec theta_0 = 0
T0 = 2*np.pi*np.sqrt(l0/g0) # période propre

def pendule( Y , g=g0 , L=l0):
    dY1 = Y[1] ; dY2 = - g/L * np.sin(Y[0])
    return np.array([dY1,dY2])

def euler(func,th0,dth0,tf,nb): # on part de t=0
    Y = np.zeros( (nb+1,2) ) # tableau de nb+1 [0,0]
    Y[0,0] , Y[0,1] = th0, dth0
    pas = tf/nb
    dates = np.linspace(0,tf,nb+1)
    for k in range(nb):
        Y[k + 1] = Y[k] + pas * func(Y[k])
    return dates, Y

#%% applications :

finT = 7*T0
Sim1 = euler(pendule,0,Vpi/1.0001,finT,12345)

plt.close()
plt.figure()
plt.plot(Sim1[0],Sim1[1][:,0])
plt.plot([0,finT],[np.pi,np.pi])
plt.plot([0,finT],[-np.pi,-np.pi])
plt.show()

#%%
```