```
### programme mystère de recherche du minimum dans une liste

def mystere(L):
    n=len(L)
    if n==0:
        None
    elif n==1:
        return (L[0])
    else:
        x=mystere(L[0:n-1])
        if x<=L[-1]:
            return (x)
        else:
            return (L[-1])

# >>> mystere([1,2,5,0.8,2,9,7,12])
# 0.8

### exercice 2 : miroir

def miroir_it(mot):
    tom=''
    for lettre in mot:
        tom=lettre+tom
    return (tom)

# >>> miroir_it('patricia')
# 'aicirtap'
# >>> miroir_it('eh!ca va la vache')
# 'ehcav al av ac!he'

def miroir_rec(mot):
    n=len(mot)
    if len(mot)==0:
        None
    elif len(mot)==1:
        x=mot
    else:
        x=miroir_rec(mot[0:n-1])
        x=mot[-1]+x
    return (x)

# >>> miroir_rec('patricia')
# 'aicirtap'

### exercice 3 factorielle
def fact_it(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return (fact)

# >>> fact_it(4)
# 24
# complexité n (boucle for)

def fact_rec(n):
    if n==0:
        fact=1
    else:
        fact=n*fact_rec(n-1)
    return (fact)

# >>> fact_rec(4)
# 24
# >>> fact_rec(1010)
# RuntimeError: maximum recursion depth exceeded in comparison

### EXERCICE 4 Fibonacci
### suite de fibonacci
#question 1
```

```
def fibonacci_it(n):
    """fibonacci itérative"""
    u0,u1=0,1
    un=1
    un1=u0+u1
    for i in range(2,n):
        un2=un+un1
        un=un1
        un1=un2
    return (un2)

# fibonacci_it(5)
# Out[5]: 5

#question 2
def fibonacci_rec(n):
    """fibonacci recursive avec renvoie de Un"""
    if n==0:
        un=0
    elif n==1:
        un=1
    else:
        un=fibonacci_rec(n-1)+fibonacci_rec(n-2)
    return (un)

# fibonacci_rec(5)
# Out[7]: 5

#question 3
def fibonacci_rec2(n):
    """fibonacci recursive avec renvoie de deux valeurs de Un"""
    if n==0:
        return (0,1)
    elif n==1:
        return (1,1)
    else:
        Un=fibonacci_rec2(n-1)[0]+fibonacci_rec2(n-1)[1]
        Un1=fibonacci_rec2(n-1)[1]
    return (Un1,Un)

# >>> fibonacci_rec2(5)[0]
# 5

### Exercice 5 calcul du déterminant

def extraire(A,lig,col):
    """fonction qui supprime la ligne et la colonne désignées"""
    del(A[lig])
    for i in range(len(A[0])-1):
        del(A[i][col])
    return (A)

# >>> extraire([[1,2,3],[4,5,6],[7,8,9]],0,0)
# [[5, 6], [8, 9]]

#question 2 calcul du déterminant
#cofacteur Cij=(-1)**(i+j)*Mij
#Le mineur Mij est le déterminant de la matrice obtenue en éliminant la ieme rangée et la
jeme colonne de A
#det(A)=C12M12+C11M11+C13M13 expansion suivant la première rangée

def determinant(A):
    n=len(A)
    if n==1:
        return A[0][0]
    else:
        C=0 #On initialise C
        for i in range(n):
            C+=((-1)**i) * A[i][0] * determinant(extraire(A,i,0))
    return C
```

```
# >>> determinant([[10]])
# 10


import numpy as np
A=[[1,2,3],[4,4,6],[7,8,9]]
print (np.linalg.det(A))
# 12

### exercice 6 - Bubble Bobble
#tracé du cercle de départ
import matplotlib.pyplot as plt


def cercle(x,y,r):
    """fonction qui trace le cercle de centre A(x,y) et de rayon r"""
    les_angles=np.linspace(0,2*np.pi,200)
    les_x=[x+r*np.cos(i) for i in les_angles]
    les_y=[y+r*np.sin(i) for i in les_angles]
    plt.plot(les_x,les_y)
    plt.show()

def bubble1(n,x=0,y=0,r=8):
    cercle(x,y,r)
    if n>1:
        bubble1(n-1,x+3*r/2,y,r/2)
        bubble1(n-1,x,y-3*r/2,r/2)

def bubble2(n,x=0,y=0,r=8,d=''):
    cercle(x,y,r)
    if n > 1:
        if d!='s':
            bubble2(n-1,x,y+3*r/2,r/2,'n')
        if d!='w':
            bubble2(n-1,x+3*r/2,y,r/2,'e')
        if d != 'n': #si le cercle est au nord on ne doit pas dessiner un cercle au sud
            bubble2(n-1,x,y-3*r/2,r/2,'s')
        if d != 'e':
            bubble2(n-1,x-3*r/2,y,r/2,'w')
```