```
################################################
# Exercice 1
################################################
## 1)
D={'info':6, 'modelisation': 7,'maths':12,'physique':13,'chimie':7, 'lettres':9, 'LV1':4}

print(D)

print(type(D))

D['LV2']= 2

print(D)

D['modelisation']= 6

print(D)


## 2)
print(D.items)

for x in D:
    print ( x )

for x in D:
    print ( D[ x ])



################################################
# Exercice 2
################################################

def C(n:int)->int:
    '''calcule Cn récursivement'''
    if n==0:
        return(1)
    else:
        S=0
        for j in range(n):
            S=S+C(j)
        return(S)
##
##1) Le cas Terminal est n=0, sinon il y a n appels récursifs
##pour des paramètres entiers naturels strictements plus petits,
##donc au bout de n séries d'appels au plus on atteint des cas terminaux
##
##2) il y a beaucoup de recalculs effectués,
##sans stockage en mémoire des calculs déjà effectués
##
##3)



dict = {}

dict[0]=1

def Cmem(n:int)->int:
    #test si C[n] est déja connu
    if n not in dict:
        res=0
        for j in range(n):
            res=res+Cmem(j)
         #Mise à jour du dictionnaire
        dict[n]=res
    else:
        res=dict[n]
    return res

print(dict)
```

```python
print(Cmem(100))

#print(dict)


#print(C(100))







################################################
# Exercice 3
################################################


import pylab
F = pylab.gca() # F peut être vue comme un objet 'figure'
def cercle(x, y, r):
    """ cercle de centre (x,y) et de rayon r """
    # création du cercle:
    cir = pylab.Circle([x, y], radius = r, fill = False)
    # ajout du cercle à la figure :
    F.add_patch(cir)

def CerclesRec(x, y, r):
    """ construction récursive de la figure """
    cercle(x, y, r)
    if r > 1:
        CerclesRec(x+3*r/2, y, r/2)
        CerclesRec(x, y-3*r/2, r/2)

# appel de la fonction CerclesRec
CerclesRec(0, 0, 8)
# pour placer toute la figure dans un repère orthonormé :
pylab.axis('scaled')
# affichage de la figure :
pylab.show()


## 1)

## 2)


## 3)

import pylab
F = pylab.gca() # F peut être vue comme un objet 'figure'
def cercle(x, y, r):
    """ cercle de centre (x,y) et de rayon r """
    # création du cercle:
    cir = pylab.Circle([x, y], radius = r, fill = False)
    # ajout du cercle à la figure :
    F.add_patch(cir)

def CerclesRec2(x, y, r,c):
    """ construction récursive de la figure """
    cercle(x, y, r)
    if r > 1:
        if c!='d':
            CerclesRec2(x+3*r/2, y, r/2,'g')
        if c!='g':
            CerclesRec2(x-3*r/2, y, r/2,'d')
        if c!='b':
            CerclesRec2(x, y-3*r/2, r/2,'h')
        if c!='h':
            CerclesRec2(x, y+3*r/2, r/2,'b')
```

```
# appel de la fonction CerclesRec
CerclesRec2(0, 0, 8,'m')
# pour placer toute la figure dans un repère orthonormé :
pylab.axis('scaled')
# affichage de la figure :
pylab.show()




################################################
# Exercice 4
################################################


Dc= {0:1,1:2,2:3}

print(Dc)

def U(n:int)->float:
    #test si U[n] est déja connu
    if n in Dc:
        res=Dc[n]
    else:
        res=U(n-3)*U(n-2)/U(n-1)
        Dc[n]=res
    return res

print(Dc)

print(U(10))


print(Dc)




################################################
# Exercice 5
################################################


def mystere(t ,k):
    if k==len(t)-1:
        return True
    if t[k]>t[k+1]:
        return False
    return mystere(t,k+1)

t=[6,9,4,8,12]

print(mystere(t,2))

## 1 ) mystere(t,2), mystere(t,3), mystere(t,4)->True

## 2 ) mystere(t,0), mystere(t,1)->False

## 3 )  La fonction mystere(t, k)retourne True si la suite
## d'entiers contenue dansle tableau t à partir de l'indice k est croissante
## et retourne False sinon.

## 4 )  Le pire des cas correspond  à l'appel
## mystere(t,k)pour un tableau croissant à partir de l'indice k.
## Il y a dans ce cas n-kappels récursifs en comptant l'appel initial.
```