Enginyeria de Sistemes TIC – *i*TIC

Informàtica-iTIC
problemes de teoria

Marta I. Tarrés-Puertas
Escola Politècnica Superior d'Enginyeria de Manresa
Universitat Politècnica de Catalunya.
versió 1: Setembre 2013

30 de gener de 2018

Índex

1 Estructures condicionals i iteratives **4**

2 Cadenes de caràcters **15**

3 Llistes **23**

4 Fitxers, diccionaris i tuples **39**

1 Estructures condicionals i iteratives

Exercici 1.1
Donat el següents fragments de codi, escriviu exactament, què es mostra per pantalla, després d'executar els següents fragments de codi.
[Apartat a]

```
def funcio1(x,t):
   return x**t
def funcio2(x,t):
   return funcio1(funcio1(t,x),x)

x=2
t=1
print "Returned value: ",funcio2(funcio2(x,t),t+x)
```

[Apartat b]

```
def fun2(t,x):
   return t-2*x
def fun1(a,b):
   return fun2(fun2(b,a),b+3)
x=5
t=2
print "Returned value: ",fun1(fun2(x,t),t+x)
```

Exercici 1.2 Donat el següent fragment de codi, escriviu exactament, què es mostra per pantalla.

```
def funcio1(x,t):
   if x>t:
      print "Bigger"
   elif x%2==0:
      print "Odd"
   else:
      print "Neither bigger nor odd"

def funcio2(x,t):
   funcio1(t,x)
   funcio1(x,t)
   funcio1(t-1,x)
```

```
x=5
t=4
funcio2(x,t)
```

EXERCICI 1.3 Donats el següent fragment de codi, seguiu el fluxe d'execució i escriviu exactament quins missatges es mostren per pantalla com a resultat de l'execució d'aquest script.

```
def test(x,y):
    if (y == 0):
        print("bad value for y!")
        s = 0
    else:
        print("good value for y!")
        s = x / y
    print("What?")
    return s

print test(test(20,2),0)
print test(10,test(5,2))
```

EXERCICI 1.4 Una cadena de producció reconeix el pes erroni dels seus productes en funció de dos paràmetres. Si el resutat de la suma d'aquests paràmetres dóna com a resultat un nombre múltiple de 2 i de 3, és un indicatiu de que el producte analitzat és erroni.

[Apartat a]

Per tal d'ajudar-los en la detecció de productes erronis, se'ns encomana el disseny d'una funció booleana de nom checkOddSum que ha de rebre dos paràmetres i ha de retornar True si i només si el resultat de la suma dels dos paràmetres correspon a un nombre múltiple de 2 i de 3. En cas contrari, ha de retornar False. Suposeu que aquesta funció es guardarà en el mòdul funcions.py

Exemples de funcionament:

checkOddSum(2,4) hauria de retornar **True**
checkOddSum(1,4) hauria de retornar **False**
checkOddSum(8,0) hauria de retornar **False**

[Apartat b]

Dissenyeu un script tal que demani dos nombres a l'usuari, efectui una crida a la funció anterior i escrigui per pantalla el missatge "Producte correcte" o bé "Producte Erroni".

EXERCICI 1.5 Escriviu exactament, què es mostra per pantalla, després d'executar els següents fragments de codi.

[Apartat a]

```
num=10
while num>3:
    print num
    num-=1
```

[Apartat b]

```
a = 0
b = 1
count = 0
max_count = 8
while count < max_count:
    count = count + 1
    old_a = a
    old_b = b
    a = old_b
    b = old_a + old_b
    print old_a,
```

[Apartat c]

```
n=13
k=5
while n > k:
    n-=1
    print n
    print k-1
```

EXERCICI 1.6 Escriviu exactament, què es mostra per pantalla, després d'executar el següent fragment de codi.

```
print "q1_1415"
k=10
j=10
while j>0:
    print j
    if j%2==0:
        j=j/2
    else:
        j+=1
```

EXERCICI 1.7 Desxifra què fa aquest script per qualsevol nombre positiu n.

```
n=19845
s=0
k,q=n,n
while k!=0:
    k=q/10
    r=q%10
    s+=r
    q=k
print s
```

Exercici 1.8 Escriviu exactament, què es mostra per pantalla, després d'executar cadascun dels fragments de codi que segueixen.

Apartat a)

```
num = 13
j=True
while num > 3 or not j:
   print num
   num = num - 1
```

Apartat b)

```
i=10
while i>0:
    print i
    if i%2!=0:
        i/=2
    else:
        i+=1
```

Apartat c)

```
def mira(t,x):
    if x<t:
        print "Not bigger"
    elif x%2==0:
        print "Not Odd"
    else:
        print "Bigger or odd"

def desxifra(x,t):
    mira(t,x)
    mira(x,t)
    mira(x-1,t-1)

r=4
m=5
desxifra(r,m)
```

Apartat d) Descriu també què fa la següent funció npi, $\forall n, n \in \epsilon, n > 0$.

```
import math
def npi(n):
    j=1
    while j<=n/2:
        if math.pow(j,2)==n:
            print j
        j+=1
```

Exercici 1.9 La sèrie ascendent. L'objectiu del problema a resoldre consisteix a demanar 10 nombres a l'usuari, i detectar si els nombres es troben ordenats ascendentment o no. A tal efecte, se us demana que dissenyeu un script òptim, tal que, demani com a màxim 10 nombres a l'usuari. En cas que cada nombre sigui més gran o igual que l'anterior, cal que es mostri per pantalla el missatge "Ascending order". Altrament cal escriure "Not Ascending order". A continuació segueixen exemples de funcionament.

```
EXEMPLE 1 de funcionament
Entri nombre: 3
Entri nombre: 8
Entri nombre: 4
Not Ascending order

EXEMPLE 2 de funcionament
Entri nombre: 2
Entri nombre: 8
Entri nombre: 15
Entri nombre: 34
Entri nombre: 45
Entri nombre: 56
Entri nombre: 67
Entri nombre: 67
Entri nombre: 89
Entri nombre: 98
Ascending order
```

Exercici 1.10 Escriu amb una línia què fa el següent programa.

```
min_n = 1
max_n = 100
right_answer = False
while not right_answer:
    mid_n = (max_n+min_n+1)/2
    print 'Is it ' + str(mid_n) + '? '
    answer = raw_input()
    if answer[0] == 'y':
        right_answer = True
    elif answer.startswith('higher'):
        min_n = mid_n + 1
    elif answer.startswith('lower'):
        max_n = mid_n - 1
    else:
        print "Sorry, I don't understand your answer."
print 'I got it!'
```

EXERCICI 1.11 El joc de Pedra-Paper-Tisores. Es pretén simular el conegut joc entre 2 jugadors del Pedra-Paper-Tisores.

[Apartat a] Es demana el disseny òptim d'una funció fructífera *PedraPaperTisores(jugA, jugB)*, on *jugA* correspon a un dels valors del jugador A (pedra/paper/tisores), i *jugB* correspon a un dels valors del jugador B (pedra/paper/tisores). La funció ha de retornar el jugador guanyador. A continuació segueixen exemples de funcionament.

```
>>> print PedraPaperTisores("pedra","paper")
jugador B
>>> ja=raw_input("Tria jugada (pedra/paper/tisores)")
tisores
>>> jb=raw_input("Tria jugada (pedra/paper/tisores)")
tisores
>>> print PedraPaperTisores(ja,jb)
Empat
```

[Apartat b] Suposant que la funció anterior està correctament implementada, dissenyeu un petit script tal que permeti jugar 10 vegades, indicant quin jugador guanya o si hi ha empat. Finalment, ha d'escriure per pantalla quin jugador ha guanyat més vegades, i quantes vegades han empatat. A continuació un exemple de funcionament suposant 4 vegades.

```
Jugador A. Tria jugada (pedra/paper/tisores) pedra
Jugador B. Tria jugada (pedra/paper/tisores) paper
Guanya jugador B
Jugador A. Tria jugada (pedra/paper/tisores) pedra
Jugador B. Tria jugada (pedra/paper/tisores) tisores
Guanya jugador A
Jugador A. Tria jugada (pedra/paper/tisores) paper
Jugador B. Tria jugada (pedra/paper/tisores) pedra
Guanya jugador A
Jugador A. Tria jugada (pedra/paper/tisores) tisores
Jugador B. Tria jugada (pedra/paper/tisores) tisores
Empat
RESUM DE PARTIDES:
Ha guanyat jugador A
Nombre partides amb empat: 1
```

EXERCICI 1.12 Un nombre enter és *màgic* si és el resultat de la suma dels nombres que hi ha des del nombre *1* fins a la meitat del propi nombre (pot ser inclòs o no). Per exemple, el nombre *6* és màgic donat que *6=1+2+3*, i també ho és el *10*, ja que *10=1+2+3+4*. El *8* no és un nombre màgic donat que no és igual ni a *1+2+3+4* ni a *1+2+3*. Es demana que creeu la funció *esMagic* tal que, donat un nombre corresponent a un valor enter positiu qualsevol, retorni *True* si es tracta d'un nombre màgic.

EXERCICI 1.13 Escriviu exactament, què es mostra per pantalla, després d'executar cadascun dels fragments de codi que segueixen.

Apartat a)

```
def potser(r):
    return r%2==0 and r%3==0

def castanya(k,s):
    j=k
    t=True
    m=s
    while j>0 and t:
        if potser(m):
            t=False
        else:
            print "upps"
            m+=1
        j-=1
    if not t:
        me="castanya"
    else:
        me="nose"
    return me

if __name__=='__main__':
    print castanya(3,5)
    print castanya(5,6)
    print castanya(-4,1)
```

Apartat b)

```
t=False
j=0
while j<3 and not t:
    i=1
    while i<4 and not t:
        if i%2==0:
            t=True
        else:
            print i,"Not parell"
        i+=1
    j+=1
    print j
if not t:
    print "kalidoscopius"
if j==3:
    print "not kalidoscopius"
```

Exercici 1.14 El costat del camp de fútbol. Dissenyeu una funció de nom *costatCampFutbol* tal que, demani a l'usuari l'àrea del camp de futbol i la longitud d'un dels seus costats i retorni la longitud del costat adjacent. A continuació segueix un exemple de funcionament.

```
>>> print costatCampFutbol()
Introdueix area del camp: 8250
Introdueix longitud costat: 110
75.0
```

Exercici 1.15 Quadrats perfectes. Dissenya la funció *quadratsPerfectes* tal que, donat un nombre positiu n com a paràmetre, escrigui els n primers quadrats perfectes que no són nombres parells.

```
>>> quadratsPerfectes(3)
1
9
25
```

Exercici 1.16 Apartat a) Descriu en **una línia** què fa la funció *misteri*, $\forall x, x \in \epsilon, x \geq 2$.

```
def misteri(x):
    i=1
    t,z=input("Enter number: "),input("Enter number: ")
    nose=abs(t-z)
    aux=z
    while i<x-1:
         z=input("Enter number: ")
         if nose>abs(t-z):
             nose=abs(t-z)
             aux=z
         i+=1
    print nose,aux
```

Apartat b) Descriu en **una línia** què fa la funció *niIdea*, $\forall n, n \in \epsilon, n > 0$

```
def niIdea(n):
    z=n/2
    print n
    while z>0:
        if n%z==0:
            print z
        z-=1
```

Exercici 1.17 El pàrquing NoMaTaBaLis, gestiona les tarifes d'aparcament de la següent manera,

- L'import per hora completa és de 2,60 euros
- El preu per fracció d'hora correspon a 0,043 per minut transcorregut
- El preu per dia a partir de 12 hores, és tarifa única de 30 euros

Apartat a) Suposant, que el màxim que estarà aparcat un cotxe al garaig són 24 hores, es demana que implementeu la funció fructífera *noAparquisAlCentre*, tal que, donat un valor numèric corresponent al nombre de minuts que es troba el cotxe al pàrquing, retorni l'import a pagar.

```
>>> print noAparquisCentre(120)
5.2
>>> print noAparquisCentre(720)
30
>>> print noAparquisCentre(60)
2.6
>>> print noAparquisCentre(30)
1.29
```

Apartat b) Suposant que la funció anterior està correctament implementada, dissenyeu un script òptim tal que permeti calcular el preu a pagar per a cadascun dels 100 cotxes aparcats. Addicionalment, cal que s'indiqui quin ha estat l'import màxim facturat, l'import mínim i quin ha estat el total facturat per a aquests 100 cotxes. Suposeu que els minuts introduïts són positius i correctes. A continuació segueix un exemple d'execució suposant 5 cotxes a facturar.

```
Enter minutes: 120
To pay 5.2 euros
Enter minutes: 60
To pay 2.6 euros
Enter minutes: 45
To pay 1.935 euros
Enter minutes: 15
To pay 0.645 euros
Enter minutes: 720
To pay 30 euros
----------------------
Total cash 40.38 euros
Maximum cost 30 euros
Minium cost 0.645 euros
```

Exercici 1.18 Nuggets. Al MacUpc es poden demanar tapes de nuggets que contenen 6, 9 o 20 peces. Escriu una funció boleana en Python, de nom *nuggets*, que accepti un enter, *num*, com a paràmetre i decideixi si és possible demanar una combinació de tapes que contingui aquest *num* peces. A continuació segueixen exemples de crides de la funció.

```
>>> print nuggets(46)
tapa1(6)--> 1 tapa2(9)--> 0 tapa3(20)--> 2
True
>>> print nuggets(195)
tapa1(6)--> 0 tapa2(9)--> 15 tapa3(20)--> 3
True
>>> print nuggets2(181)
tapa1(6)--> 0 tapa2(9)--> 9 tapa3(20)--> 5
True
>>> print nuggets2(5)
False
>>> print nuggets2(34)
False
```

Exercici 1.19 Escriviu exactament, què es mostra per pantalla, després d'executar cadascun dels fragments de codi que segueixen.

Apartat a)

```
omg = True
a = 0
b = 0
while omg:
    print "O"
    if a + b >= 24:
        omg = False
    a = a + 5
    b = b + 7
```

Apartat b)

```
a = 1
while a % 7 != 0:
    if a % 2 == 0:
        print "O",
    if a == 2:
        print "X"
    a = a + 1
```

Apartat c)

```
def f(a):
    a = a + 5
    return a

b = 0
f(b)
print b, ",",
b = f(b)
print b
```

Apartat d)

```
n = 10
i = 10
while i > 0:
    print i
    if i % 2 == 0:
        i = i / 2
    else:
        i = i + 1
```

Apartat e) Descriu també què fa la següent funció *npi*, $\forall n, n \in \epsilon, n > 0$.

```
def npi(n):
    j=n/2
    while j>=1:
        if j**2==n:
            print j
        j-=1
```

Exercici 1.20 El calendari. Dissenyeu òptimament la funció *calendari* tal que, donat un número de l'1 al 12, corresponent al mes en curs, i l'any en curs, retorni quants dies té aquell mes.

Recordeu que, es consideren anys de traspàs, els anys en què les dues darreres xifres de l'any són múltiples de 4, excepte si aquestes xifres són 00. Aleshores, cal tenir en compte les dues primeres xifres de l'any. Si són múltiples de 4, l'any també serà de traspàs. Per exemple, el 1996 va ser de traspàs, perquè 96 és múltiple de 4; el 1900, no fou de traspàs, perquè 19 no és múltiple de 4. Però el 2000, les dues darreres xifres també eren 00 i calia tenir en compte el 20. Com que és múltiple de 4, també va ser de traspàs. En resum: és de traspàs cada any múltiple de 4, excepte els múltiples de 100, que no ho són, i excepte els múltiples de 400, que sí que ho són. A continuació segueixen els exemples de funcionament.

```
i=1
while i<=12:
    print i,"2017:",calendari(i,2017),"1996:",calendari(i,1996),"1900:",
    print calendari(i,1900),"2000:",calendari(i,2000)
    i+=1
```

```
Resultat d'execució:
1 2017: 31 1996: 31 1900: 31 2000: 31
2 2017: 28 1996: 29 1900: 28 2000: 29
3 2017: 31 1996: 31 1900: 31 2000: 31
4 2017: 30 1996: 30 1900: 30 2000: 30
5 2017: 31 1996: 31 1900: 31 2000: 31
6 2017: 30 1996: 30 1900: 30 2000: 30
7 2017: 31 1996: 31 1900: 31 2000: 31
8 2017: 31 1996: 31 1900: 31 2000: 31
9 2017: 30 1996: 30 1900: 30 2000: 30
```

```
10 2017: 31 1996: 31 1900: 31 2000: 31
11 2017: 30 1996: 30 1900: 30 2000: 30
12 2017: 31 1996: 31 1900: 31 2000: 31
```

EXERCICI 1.21 Dissenyeu òptimament la funció *chequejaInput* tal que demani nombres a l'usuari fins que entri un nombre senar i divisible per 7. Quan l'hagi introduït, cal mostri per pantalla el missatge "Mission 7 odd accomplished" (acompanyat del número introduït senar i divisible per 7) i retorni el número d'intents realitzats. Chequegeu-ne el funcionament amb l'exemple que segueix.

```
>>> print str(chequejaInput())+" intents previs"
Entra nombre: 8
Entra nombre: 14
Entra nombre: 3
Entra nombre: 21
Number 21 Mission 7 odd accomplished
3 intents previs
```

EXERCICI 1.22 Dissenyeu òptimament la funció *triangleInvertitIticn* tal que, per qualsevol nombre positiu superior a 0, escrigui per cada línia, el número que rep com a paràmetre, decrementant en 2 unitats fins a arribar a l'1 o al 2. Aquest procediment s'ha de repetir per cada línia, començant cada línia pel valor 2 unitats inferior a l'anterior. El programa acaba quan escriu una línia amb un 1 o un 2. A continuació segueixen els exemples de funcionament.

```
>>> triangleInvertitIticn(20)
20 18 16 14 12 10 8 6 4 2
18 16 14 12 10 8 6 4 2
16 14 12 10 8 6 4 2
14 12 10 8 6 4 2
12 10 8 6 4 2
10 8 6 4 2
8 6 4 2
6 4 2
4 2
2
```

```
>>> triangleInvertitIticn(13)
13 11 9 7 5 3 1
11 9 7 5 3 1
9 7 5 3 1
7 5 3 1
5 3 1
3 1
1
```

2 Cadenes de caràcters

Exercici 2.1 Donat el següents fragments de codi, escriviu exactament, què es mostra per pantalla, després d'executar els següents fragments de codi.

[Apartat a]

```
def funcioMisteri2(p):
    s=""
    vocals="aeiouAEIOU"
    i=0
    if p[i] not in vocals:
        s=p[i+1:]+p[i]+"ey"
    else:
        s=p[i:]+"h"+"ey"
    return s
```

1. `print funcioMisteri2("provainformatica")`
2. `print funcioMisteri2("informatica")`
3. `print funcioMisteri2("")`

[Apartat b]

```
frase="joan@gmail.com pere@gmail.com ioannaT@epsem.es"
l=frase.split()
for a,b in enumerate(l):
    print a,b[:b.index("@")]
```

Exercici 2.2 Escriu què es mostra per pantalla en cada execució

```
s='INFos'
for element in range(len(s)):
    print element,s[element]
for element,element2 in enumerate(s):
    print element,element2
k = s.split(":")
print k[1]
print s[:4]
```

Exercici 2.3 Defineix una funció pura anomenada traversal que rebi una paraula i retorni la paraula al revés amb les vocals rodejades de parèntesis.

```
def traversal(l):
    """
```

```
    Retorna la paraula l al revés i amb les vocals rodejades de parèntesis
    >>> traversal('blueberry')
    'yrr(e)b(e)(u)lb'
    >>> traversal('')
    ''
    """
```

Exercici 2.4 L'Agència Catalana de Xifratge de dades ens demana el disseny i implementació d'una funció que permeti encriptar la paraula clau d'accés al correu electrònic d'un usuari. El mecanisme d'encriptació reb una paraula i es construeix una nova paraula aplicant les següents normes:

- Eliminar les vocals de la paraula original
- Convertir a majúscules els caràcters alfabètics
- Capgirar els caràcters
- Al final de la nova paraula s'hi afegeix el nombre de caràcters de la paraula original.

Dissenyeu la funció encriptaParaula, que rep una paraula i retorna la paraula encriptada. A continuació segueix la documentació de la funció.

```
def encriptaParaula(paraula):
    """
    Retorna el resultat d'encriptar la paraula
    >>> encriptaParaula("mariobrobros456")
    '654SRBRBRM15'
    >>> encriptaParaula("MTabc43aeiOU")
    '34CBTM12'
    >>> encriptaParaula("")
    '0'
    >>> encriptaParaula("qualsevol@simbol#")
    '#LBMS@LVSLQ17'
    """
```

Exercici 2.5 Completeu el docstring i els doctests de la funció que segueix, sent p una paraula que únicament conté nombres binaris.

```
def misteri2(p):
    """
    docstring
    >>> misteri2('0011')
    Apartat a)
    >>> misteri2('00110010')
    Apartat b)
    >>> misteri2('1100')
    Apartat c)
    >>> misteri2('0000')
```

```
    Apartat d)
    >>> misteri2('')
    Apartat e)
    """
    if p.startswith('0'):
        t=p.find('1')
        if t!=-1:
            s=p[t:]
        else:
            s=''
    else:
        s=p
    return s
```

EXERCICI 2.6 Escriviu exactament, què es mostra per pantalla, després de cada execució.

```
#Apartat a)
def yupif(s):
    s = s * 3 + "!"
    print "Zzz Zzz"

if __name__ == "__main__":
    m = "yupi yeah"
    yupif(m)
    print m.split()
```

```
#Apartat b)
def wondering(s):
    for i in range(4):
        r=""
        for c in s:
            r=r+c*i+":"
        print r

if __name__=='__main__':
    c="ok"
    wondering(c)
```

```
#Apartat c)
l1 = ['python', 'java', 'c']
l2 = l1[:]
l2.append('prog')
l3 = l2
l3[1] = 'lgs'
print [[i,v] for i,v in enumerate(l1)]
print [l2[i] for i in range(len(l2))][1]
print "-".join([d for d in l3])
```

EXERCICI 2.7 Completa els doctests que segueixen i afegeix un docstring òptim.

```
def niIdea(n):
    """
    >>> niIdea('avui classe aula info2.4')
    #Apartat 2.1)
    >>> niIdea(["avui classe aula info2.4",'melmelada'])
    #Apartat 2.2)
    >>> niIdea([1,2,3,4])
```

```
    #Apartat 2.3)
    """
    return 100*sum([int(c) for c in n if c.isdigit()])/float(len(n))

def misteri(s):
    """
    >>> misteri('aAAa')
    #Apartat 2.4)
    >>> misteri('AI')
    #Apartat 2.5)
    >>> misteri('Jjq')
    #Apartat 2.6)
    >>> misteri('')
    #Apartat 2.7)
    """
    t=True
    for ch in s:
        if ch.lower() != s[0].lower():
            t=False
    return t
```

EXERCICI 2.8 Modificant la paraula. Suposant que s correspon a una cadena de caràcters, i n correspon a un enter, Dissenya una funció pura, de nom, *repeteix(s,n,w)* tal que, donats s i n, afegeixi cada n caràcters a s, la cadena w.

```
def repeteix(s,n,w):
    """
    Afegeix cada n caràcters a s, la cadena w
    >>> repeteix('12345678910',2,'hack')
    '12hack34hack56hack78hack91hack0'
    >>> repeteix('',3,'inf')
    ''
    >>> repeteix('hola',3,'inf')
    'holinfa'
    >>> repeteix('zZZZ',1,'inf')
    'zinfZinfZinfZinf'
    >>> repeteix('inf',-1,'double')
    'inf'
    """
```

EXERCICI 2.9 **Encriptant frases**. Implementa òptimament la funció *encriptaFrase* tal que, donada una frase separada per espais, retorni el resultat d'encriptar la frase d'acord amb el procediment que segueix: per cada paraula de la frase, s'agafa la darrera lletra de la paraula, i així es va construint una nova paraula encriptada. A continuació segueixen els doctests.

```
def encriptaFrase(frase):
```

```
    """
    >>> encriptaFrase('python programming is funny and tricky')
    'ngsydy'
    """
```

Exercici 2.10 El filtrador de missatges corruptes.

[Apartat a] Es demana la implementació d'una funció de nom *missatgecorrupte(missatge, patro)* que permeti filtrar els missatges corruptes. Per filtrar un missatge corrupte se segueixen els següents passos:

- Els caràcters que no siguin lletres ni digits ni espais cal substituir-los per un espai en blanc
- Els caràcters que formin part del patro, cal substituir-los per *
- La resta de caràcters es mantenen
- Suposeu que patro conté únicament caràcters alfanumèrics

Adoneu-vos que hi pot haver missatges que no siguin corruptes, i el filtrat no tindrà efecte.

```
def missatgecorrupte(missatge, patro):
    """
    Retorna una còpia del missatge despres d'aplicar filtratge
    >>> missatge_corrupte('Cor$%&r&^up&ted te**xt', 'jkqxyzJKQXYZ')
    'Cor   r  up ted te  *t'
    >>> missatge_corrupte('aqn eKvil vxiruzrs dzid ttyhis!', 'jkqxyzJKQXYZ')
    'a*n e*vil v*iru*rs d*id tt*his '
    """
```

[Apartat b] Un cop dissenyada la funció, escriviu un petit script de nom *netejaMissatges.py*. Aquest script ha d'obtenir el patró i un nom de fitxer per la línia de comandes. Suposeu que aquest fitxer sempre existeix i no està buit. El fitxer conté un seguit de missatges, un per línia. Cal que l'script dissenyat escrigui en un nou fitxer que té el mateix nom que l'original afegint *rev* abans de l'extensió. En aquest nou fitxer hi ha d'haver els missatges corruptes filtrats, mantenint el format de cada missatge en una línia diferent. Finalment, ha de mostrar per pantalla el nombre de missatges corruptes. A continuació segueix un exemple de funcionament.

```
python netejaMissatges.py 'jkqxyzJKQXYZ' wrongMessages.txt
2 missatges corruptes

cat wrongMessages.txt
Cor$%&r&^up&ted te**xt
aqn eKvil vxiruzrs dzid ttyhis!
No problem

cat wrongMessagesrev.txt
Cor   r  up ted te  *t
a*n e*vil v*iru*rs d*id tt*his
No problem
```

Exercici 2.11 El verificador d'adreces de correu electrònic.

[Apartat a] Es demana la implementació d'una funció de nom *check* que permeti detectar si una adreça de correu electrònic és correcta. Per simplificar, suposarem que una adreça de correu electrònic és correcta si té el format, *nom@servidor.extensio*, on, el *nom* pot ser qualsevol cadena de caràcters de com a mínim 3 lletres. El *servidor* només pot contenir lletres alfabètiques i la longitud del servidor ha de ser com a mínim de 5 caràcters. Una *extensió* es considera correcta si correspon a: *cat*, *com*, o bé *es*.

```
def check(email):
    """
    return True if email has correct format
    >>> check('hola.hello@es')
    False
    >>> check('@.')
    False
    >>> check('pere@gmail.com')
    True
    >>> check('pere.gmail@com')
    False
    >>> check('ara@gmail22.com')
    False
    >>> check('pere@epsem.upc.es')
    False
    >>> check('jaonagmail.com')
    False
    """
```

[Apartat b] Un cop dissenyada la funció, escriviu un petit script de nom *rastrejador.py*, tal que, donada una seqüència d'emails obtinguda per la línia de comandes, escrigui per pantalla el nombre d'emails incorrectes i escrigui en un fitxer de nom *rightEmails.txt* els emails correctes. El format d'escriptura a fitxer esperat correspon a una línia per adreça de correu correcta. A continuació segueix un exemple d'execució.

```
python rastrejador.py pere@epsem.upc.edu joan@gmail.com hola.hello@es pere@gmail.com
 ara@gmail22.com
Detectats 3 emails incorrectes
cat rightEmails.txt
joan@gmail.com
pere@gmail.com
```

Exercici 2.12 Els nombres binaris.

[Apartat a] Dissenyeu i implementeu una funció *checkBinary* que rebi com a paràmetre una cadena composta de nombres binaris separats per espais en blanc, juntament amb una paraula que actua com a nombre binari a cercar, d'ara endavant, *binary target*, i mostri les posicions d'aquest *binary target* en la frase. Per exemple, si la cadena fos *001 100 111 101 010 001 001* i la *binary target* fos *001*, el resultat hauria de ser *0, 5, 6*, donat que *001* apareix en aquestes posicions en la cadena. La funció ha de retornar *False* si la *binary target* no apareix en la cadena.

```
def checkBinary(string,target):
    """
    return a list containing the positions of target in binary string.
    Return False if target not in string
    >>> checkBinary('001 100 111 101 010 001 001','001')
    [0, 5, 6]
    >>> checkBinary('001 100 111 101 010 001 001','1111')
    False
    >>> checkBinary('','1111')
    False
    >>> checkBinary('001 100 111 101 010 001 001','')
    False
    >>> checkBinary('','')
    False
    """
```

[Apartat b] Suposeu que esteu cercant *binary target* d'una llarga llista. Dissenyeu i implementeu la funció *showAllTargets* tal que, donada una cadena de nombres binaris i una llista de *binary target*, mostri les posicions on es troben els *binary target* de la llista. Utilitzeu la funció *checkBinary* de l'apartat anterior. A continuació segueixen els doctests.

```
def showAllTargets(string,ltarget):
    """
    >>> showAllTargets('001 100 111 101 010 001 001',['001','0000'])
    Binary target 001 found in positions
    0 5 6
    Binary target 0000 not found
    >>> showAllTargets('001 100 111 101 010 001 001',[])
    Empty Binary targets
    >>> showAllTargets('',['111'])
    Empty Binary string
    >>> showAllTargets('',[])
    Operation not allowed
    """
```

Exercici 2.13 Completa els doctests que segueixen i afegeix un docstring òptim.

```
Apartat a)
def ohmygod(s):
    """
    >>> ohmygod('01234abcA')
    #apartat a.1)
    >>> ohmygod('AaA')
    #apartat a.2)
    """
    seen=""
    for ch in s:
        if ch in seen:
```

```
            return False
        else:
            seen+=ch
    return True

Apartat b)
def npi(l):
    """
    >>> npi(['python', 'java', 'c'])
    #apartat b.1)
    """
    print [[i,v] for i,v in enumerate(l)]
    print [l[i] for i in range(len(l))][-1]
    print "-".join([d for d in l])
```

Exercici 2.14 Chequejador de Passwords. Un password es considera correcte si no té caràcters especials i té almenys 8 lletres. Se us demana que implementeu la funció que segueix de manera òptima.

```
def chequejaPassword(pwd):
    """
    retorna True si pwd és correcte
    >>> chequejaPassword('informatica')
    True
    >>> chequejaPassword('pere@22#avi')
    False
    """
```

Exercici 2.15 La paraula repetida. Escriu la funció *paraulaRepe* tal que, donada una frase separada per espais (assumeix que no hi ha caràcters de puntuació ni lletres majúscules), juntament amb una paraula, mostri la posició/les posicions en les que apareix aquesta paraula en la frase (la primera posició és la posició 0). A continuació segueixen els doctests.

```
def paraulaRepe(frase,p):
    """
    >>> paraulaRepe("we need to work hard to achieve good marks to","to")
    [2, 5, 9]
    >>> paraulaRepe("next holidays are next","good")
    []
    """
```

3 Llistes

Exercici 3.1 Donat el següents fragments de codi, escriviu exactament, què es mostra per pantalla, després d'executar els següents fragments de codi.

Apartat a)

```
a=[1,2,22,34]
b=a
a[-1]=13
print b
```

Apartat b)

```
j=[2,22,45]
j.insert(3,12)
print j
```

Apartat c)

```
a=[2,3,4]
b=[1,8,3]
a.extend(b)
print a
a.sort()
print a
```

Apartat d)

```
def funcioMisteri(l):
    jeje=0
    k=[]
    for element in l:
        jeje=jeje+element
        k+=[jeje]
    return k
```

4. `print funcioMisteri([2,4,6,7,8,9])`
5. `print funcioMisteri([])`

Exercici 3.2 Completa els doctests que segueixen i afegeix un docstring òptim.

```
Apartat a)
def idontknow(a,b):
    """
    >>> idontknow([1, 3, 5], [5, 3, 1])
    #apartat c.1)
```

```
    >>> idontknow([1, 3, 6, 9], [10, 14, 3, 72, 9])
    #apartat c.2)
    >>> idontknow([2, 3], [3, 3, 3, 2, 10])
    #apartat c.3)
    >>> idontknow([2, 4, 6], [1, 3, 5])
    #apartat c.4)
    """
    l=[]
    for e in a:
        if e in b:
            l+=[e]
    return l

Apartat b)
def uala(m):
    """
    >>> uala([[1, 2, 3], [2, 4, 11], [3, 8, 5]])
    #apartat d.1)
    >>> uala([[1, 1, 1, 1], [1, 0, 1, 1], [1, 0, 0, 1], [0, 0, 0, 1]])
    #apartat d.2)
    >>> uala([[1, 3], [2, 2]])
    #apartat d.3)
    """
    d1=sum(m[i][i] for i in range(len(m)))
    d2=sum(m[i][len(m)-1-i] for i in range(len(m)))
    return d1==d2
```

Exercici 3.3 Completa els doctests que segueixen.

```
Apartat a)
def mistery(dades):
    """
    >>> mistery([10, 20, 30, 40, 50])
    #apartat a.1)
    >>> mistery([])
    #apartat a.2)
    """
    t=len(dades)
    tmp=dades[t-1]
    for i in range(t):
        if i!=0:
            dades[i]=dades[i-1]
    dades[0]=tmp
    return dades

Apartat b)
def idontknow(s):
    """
```

```
    >>> idontknow('supercalifragilisticoexpialidoso')
    #apartat b.1)
    >>> idontknow('AaA')
    #apartat b.2)
    """
    z=[0]*len(s)
    for j,ch in enumerate(s):
        if ch in z:
            z[z.indexof(ch)]+=1
        else:
            z[j]=1
    return len(z)

Apartat c)
def none(l):
    """
    >>> none(['visualBasic', 'Java', 'Python', 'C'])
    #apartat c.1)
    """
    print [[v[-1],i] for i,v in enumerate(l)]
    print [l[i] for i in range(len(l))][i]
    print "#".join([d[0] for d in l])

Apartat d)
def npi(m):
    """
    >>> npi([[0, 2, 3], [2, 1, 11], [3, 8, 2]])
    #apartat d.1)
    >>> npi([[0, 2, 3, 4], [2, 1, 11, 5]])
    #apartat d.2)
    >>> npi([[1, 3], [2, 2]])
    #apartat d.3)
    """
    d1=[m[i][i] for i,v in enumerate(m) if i==v[i]]
    print d1
    d2=[m[i][len(m)-1-i] for i in range(len(m))]
    print d2
    return len([d1[i] for i in range(len(d1)) if d1[i]==d2[i]])
```

Exercici 3.4 Dissenyeu una funció pura tal que donada una llista de valors numèrics, corresponents a codis de productes, retorni la llista resultant d'eliminar-ne els codis alterats. Els codis alterats són aquells valors senars que ocupen una posició múltiple de 2 en la llista.

```
def eliminaMultiplesde2(l):
    """
    retorna la llista sense els senars que ocupen posicio multiple de 2
    >>> eliminaMultiplesde2([])
    []
```

```
    >>> eliminaMultiplesde2([4,-2,3,1,4,5,6,-10])
    [4, -2, 1, 4, 5, 6, -10]
    >>> eliminaMultiplesde2([0,2,4,6,7])
    [0, 2, 4, 6]
    """
```

EXERCICI 3.5 Apartat a) Dissenyeu una funció que, donada una llista d'elements l, retorni True si la llista conté algun nombre múltiple de 3.

```
def Multiples(l):
    """
    >>> Multiples([])
    False
    >>> Multiples([-9,12,25,44,33,125,67,245])
    True
    """
```

EXERCICI 3.6 Escriu una funció tal que donades dues matrius 300x300, correctament inicialitzades, retorni *True* si tenen els mateixos elements en la diagonal principal. Els següents doctests d'exemple són per matrius 3x3, cal generalitzar per qualsevol matriu quadrada (en particular, 300x300).

```
def checkMats(a,b):
    """
    retorna True si a i b, matrius quadrades no buides, son iguals en la diagonal
    >>> checkMats([[1,1,1],[1,2,3],[1,3,4]],[[1,0,0],[0,2,0],[1,3,4]])
    True
    >>> checkMats([[1,2,1],[1,2,3],[1,3,4]],[[1,0,0],[0,2,0],[1,3,0]])
    False
    """
```

EXERCICI 3.7 Dissenyeu una funció tal que, donada una llista d'enters, retorni *True* si és decreixent i tots els elements que la composen són positius. Completeu-la amb els doctests pertinents.

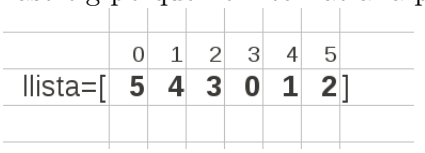
EXERCICI 3.8 Dissenyeu una funció pura de nom *misteri*, que, donada una llista k, retorna la llista decrementativa t, on l'element i-èssim de t correspon a la resta del nombre i de k amb l'*i-1*, excepte el primer. Escriu el codi python corresponent a la funció pura que implementi les especificacions dels doctests següents.

```
def misteri(k):
    """
    retorna la llista t decrementativa de k, on t[i]=k[i]-k[i-1], i>0
    >>> misteri([1,2,3,8,5,6])
    [1, 1, 1, 5, -3, 1]
    >>> misteri([])
    []
    """
```

Exercici 3.9 El filtrador d'enters de dues xifres. Dissenyeu una funció pura tal que, donada una llista d'enters, retorni la llista resultant d'eliminar els valors de més de 2 xifres numèriques.

```
def filtrador(llista):
    """
    Retorna la llista d'enters resultant d'eliminar els enters amb 3 xifres o mes
    >>> filtrador([-22,-223,40,58,112])
    [-22, 40, 58]
    >>> filtrador([-134,-200,123,100])
    []
    >>> filtrador([-1,2,3,4,-5])
    [-1, 2, 3, 4, -5]
    """
```

Exercici 3.10 **El rastrejador**. Suposant que tens una llista de mida aleatòria, correctament inicialitzada amb valors corresponents als índexs possibles de la llista. L'objectiu consisteix en rastrejar, des d'un posició donada, anant a l'índex emmagatzemat com a valor a cada posició, fins tornar a la posició 0. Per exemple, suposant la llista que mostra la figura i començant a l'índex 0, caldria rastrejar fins a la posició 5. En aquesta posició hi ha el valor 2. Caldria anar fins a la posició 2, on hi ha el valor 3. Caldria anar a la posició 3, on hi ha el valor 0, i s'acabaria el rastreig perquè hem tornat a la posició 0.



Se us demana que implementeu òptimament la funció *rastrejaPassos* d'acord amb el funcionament anterior, exemplificat a través dels doctest que segueixen. Fixeu-vos que la funció retorna el número de passos que ha calgut fer fins retornar a la posició 0, partint de la posició index. Suposeu també que els valors inclosos a la llista no porten mai a accessos que no permetin arribar a la posició 0.

```
def rastrejaPassos(l,index):
    """
    >>> rastrejaPassos([5,4,3,0,1,2], 0)
    4
    >>> rastrejaPassos([2,4,3,0,1,5], 3)
    1
    >>> rastrejaPassos([1,2,3,0],2)
    2
    >>> rastrejaPassos([], 1)
    Not data
    """
```

Exercici 3.11 **La matriu matemàtica**. Donada una llista de llistes que representa una matriu matemàtica de qualsevol ordre, se us demana que dissenyeu òptimament i implementeu una funcio pura de nom *canvi* que permeti l'operació d'intercanviar files per columnes, seguint l'ordre que es mostra en el doctest que segueix. Suposeu que la matriu està construïda correctament.

```
def canvi(m):
    """
    returns the resulting matrix of interchanging rows and columns in matrix m
    >>> canvi([[1, -1], [2, -2], [3, -3]])
    [[-1, -2, -3], [1, 2, 3]]
    >>> canvi([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
    [[3, 6, 9], [2, 5, 8], [1, 4, 7]]
    >>> canvi([])
    []
    """
```

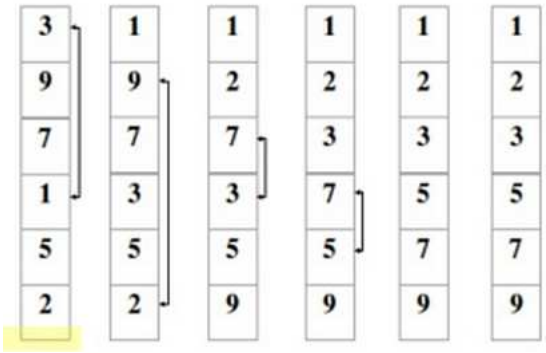
Exercici 3.12 Donada una matriu matemàtica de qualsevol ordre representada com a llista de llistes, se us demana que implementeu òptimament les funcions següents: 1) Dissenyeu la funció *sumaFiles(m)* tal que mostri el resultat de la suma dels seus elements per files, i 2) Dissenyeu la funció *sumaCols(m)* tal que mostri el resultat de la suma dels seus elements per columnes.

```
def sumaFiles(m):
    """
    escriu la suma dels valors de cada fila de m
    >>> sumaFiles([[2 ,3 ,4 ,6], [1, 1, 1, 1]])
    Valor fila: 0 --> 15
    Valor fila: 1 --> 4
    >>> sumaFiles([])
    """
def sumaCols(m):
    """
    escriu la suma dels valors de cada columna de m
    >>> sumaCols([[2, 3, 4, 6], [1, 1, 1, 1]])
    Valor columna: 0 --> 3
    Valor columna: 1 --> 4
    Valor columna: 2 --> 5
    Valor columna: 3 --> 7
    >>> sumaCols([])
    """
```

Exercici 3.13 **La llista ordenada.**

Una de les estratègies utilitzades en l'ordenació d'elements, més senzilles d'implementar, és la *ordenació per selecció*. El funcionament d'aquesta ordenació consisteix en buscar, per a cada iteració **i**, quin és l'**i**-èssim número més petit. Quan l'ha trobat l'intercanvia amb l'element que, en aquell moment, està a la posició **i** de la llista que s'està ordenant.

A continuació s'exemplifica gràficament el seu funcionament.



Se us demana que implementeu la funció **pura** *selectionSort* tal que, donada una llista de números correctes, retorni la llista ordenada ascendentment utilitzant l'estratègia *ordenació per selecció*.

Òbviament, la utilització de les funcions predefinides de Python en l'ordenació, suposarà una puntuació nul·la.

Exercici 3.14 La recaptació del teatre.

Se'ns ha encomanat el disseny d'un aplicatiu que permeti calcular la recaptació obtinguda en la venda d'entrades d'una sessió del teatre i detectar quin ha sigut el client estrella. El cap de projecte us comenta que les dades es gestionaran per mitjà d'una llista de tuples, on cada tupla contindrà dos valors: el primer serà un string amb el dni del client i el segon serà un enter amb la quantitat d'entrades comprades. Cada client apareix un únic cop a la llista. Un exemple podria ser:

```
[('44444444',5),('33333333',1),('22222222',10),('11111111',20),('55555555',1)]
```

Suposeu que el preu de cada entrada és de 10 euros.

[Apartat a] Dissenyeu la funció recaptacioFuncio(l), tal que donada una llista com la descrita, retorni la quantitat de diners que es guanyaran amb les entrades.

```
def recaptacioFuncio(l):
    """
    Retorna la recaptacio total d'entrades venudes
    >>> recaptacioFuncio([('44444444',10),('11111111',20)])
    300
    >>> recaptacioFuncio([])
    0
    """
```

[Apartat b] Dissenyeu la funció clientEstrella(l), tal que donada una llista com la descrita, retorni el codi del client que ha comprat més entrades per la funció. En cas d'empat, cal retornar el codi del primer client que es trobi a la llista.

```
def maximEntrades(l):
    """
    Retorna el codi del client que ha comprat mes entrades
    >>> maximEntrades([('44444444',10),('11111111',20),('33333333',10),
('22222222',20)])
```

```
    '11111111'
    >>> maximEntrades([])
    0
    """
```

[Apartat c] Dissenyeu un petit script tal que, donat el nom d'un fitxer que es reb per la línia de comandes, amb el format que es detalla a continuació, utilitzi les dues funcions anteriors per mostrar per pantalla la recaptació de la funció i el codi del client estrella. Les funcions anteriors es troben en el mòdul de nom funct.py

Contingut del fitxer teatre.txt

```
11111111 290
22222222 34
88888888 11
99999999 10
```

Exemple 1 d'execució:

```
python exercici1.py teatre.txt
Total recaptacio: 3450
Client estrella: 11111111
```

Exemple 2 d'execució:

```
python exercici1.py
Nom de fitxer no proporcionat
```

EXERCICI 3.15 Donada una matriu de qualsevol ordre, representada com a llista de llistes, es demana que dissenyeu de manera òptima la funció que permeti retornar la suma dels elements que es troben en el perímetre de la matriu. Per exemple, en negreta es mostren els elements de la matriu que es consideren elements del perímetre de la matriu.

1 3 2
2 1 **3**
3 2 1

```
def perimeter(matrix):
    """
    retorna la suma dels valors del perimetre de matrix
    >>> perimeter([[1, 3, 2], [2, 1, 3], [3, 2, 1]])
    17
    >>> perimeter([[1, 2, 4, 3], [2, 3, 1, 3], [3, 1, 2, 3], [4, 2, 2, 4]])
    33
    """
```

Exercici 3.16 **Sudokus.** Sudoku és un joc matemàtic l'objectiu del qual és omplir una quadrícula de 9x9 cel.les (81 caselles) dividida en subquadrícules de 3x3 (també anomenades "caixes" o "region") amb les xifres de l'1 al 9 partint d'alguns nombres ja disposats en algunes de les cel·les, tal i com es mostra a la imatge següent.

2			3					
8		4		6	2			3
	1	3	8			2		
				2		3	9	
5		7				6	2	1
	3	2			6			
	2				9	1	4	
6		1	2	5		8		9
					1			2

L'objectiu del joc és omplir cadascun dels espais lliures amb dígits entre l'1 i el 9 de manera que cada dígit aparegui exactament un cop en cada fila, en cada columna, i en cada "caixa". Cada puzzle Sudoku es construeix amb cura donat que només hi ha una única sol·lució. Un exemple correcte de sudoku és el que segueix.

2	7	6	3	1	4	9	5	8
8	5	4	9	6	2	7	1	3
9	1	3	8	7	5	2	6	4
4	6	8	1	2	7	3	9	5
5	9	7	4	3	8	6	2	1
1	3	2	5	9	6	4	8	7
3	2	5	7	8	9	1	4	6
6	4	1	2	5	3	8	7	9
7	8	9	6	4	1	5	3	2

Suposeu que se us demana la creació d'un script que comprovi si una sol·lució proposada és correcta. Donat que la tasca és excessivament complexa per a resoldre en un problema d'examen, se us demana una simplificació, que consisteix en comprovar si una "caixa" determinada conté els dígits de l'1 al 9. En l'exemple anterior, qualsevol de les "caixes" 3x3 ho és.

Si per exemple, l'usuari ha comés un error en una caixa, omplint el puzzle tal com segueix, la solució serà invàlida donat que conté dues vegades el valor 5 i el valor 6 no hi apareix cap cop.

2	7	5
8	5	4
9	1	3

[**Apartat 2.a**] Dissenyeu **òptimament** la funció *chequejaCaixa*, tal que donada una matriu 9x9 inicialitzada amb valors enters, retorni **True** únicament si la **caixa superior esquerra** és correcta. És a dir, conté els dígits de l'1 al 9. Si la caixa conté un enter fora del rang permès o bé conté valors duplicats, la funció ha de retornar **False**. En canvi, en l'exemple que segueix,

2	7	6
8	5	4
9	1	3

la funció retornaria **True**.

```
def chequejaCaixa(m):
    """
    >>> chequejaCaixa([[2,7,6,3,1,4,9,5,8],[8,5,4,9,6,2,7,1,3],[9,1,3,8,7,5,2,6,4],
[4,6,8,1,2,7,3,9,5],[5,9,7,4,3,8,6,2,1],[1,3,2,5,9,6,4,8,7],
[3,2,5,7,8,9,1,4,6],[6,4,1,2,5,3,8,7,9],[7,8,9,6,4,1,5,3,2]])
    True
    >>> chequejaCaixa([[2,7,5,3,1,4,9,6,8],[8,5,4,9,6,2,7,1,3],[9,1,3,8,7,5,2,6,4],
[4,6,8,1,2,7,3,9,5],[5,9,7,4,3,8,6,2,1],[1,3,2,5,9,6,4,8,7],
[3,2,5,7,8,9,1,4,6],[6,4,1,2,5,3,8,7,9],[7,8,9,6,4,1,5,3,2]])
    False
    """
```

[**Apartat 2.b**] Justifiqueu què us caldria modificar en la funció anterior per tal que permetés comprovar qualsevol de les caixes. (No cal el codi Python, només una justificació raonada dels canvis).

EXERCICI 3.17 El checkejador de sudokus. Donat un sudoku de qualsevol ordre, representat com matriu quadrada, amb llistes de llistes, es demana que dissenyeu de manera òptima la funció checkSudoku, tal que donat un sudoku retorni True si aquest és correcte. Per simplificar, suposarem que un sudoku és correcte si mai coincideixin dos números iguals en cada línia horitzontal ni vertical.

```
def check_sudoku(sudoku):
    """
    retorna True si el sudoku es correcte
    >>> check_sudoku([[1, 3, 2], [2, 1, 3], [3, 2, 1]])
    True
    >>> check_sudoku([[1, 2, 4, 3], [2, 3, 1, 3], [3, 1, 2, 3], [4, 2, 2, 4]])
    False
    """
```

EXERCICI 3.18 Resultats de les votacions. Dos partits polítics estan sent votats en en les diferents regions d'un país. En cada regió s'anota el nombre de vots de cadascun dels partits. Implementa la funció *resultatsVotacions*, tal que, donades 2 llistes que reb com a paràmetre, i que corresponen als vots obtinguts per cada partit en cada regió, la funció ha de retornar una llista amb 3 números: El primer element és el nombre de regions on ha guanyat el partit 1, el segon és el nombre de regions on ha guanyat el partit 2, i el tercer és el nombre de regions on han empatat.

Per exemple, si les votacions del partit 1 en la regió 0 han estat 5, en la regió 1 han estat 2, i en la regió 3 han estat 8, es representa com a $[5, 2, 8]$. El mateix per al partit 2, per exemple, $[0, 0, 9]$. La funció, donades les dues llistes anteriors, ha de retornar $[2, 1, 0]$, ja que: nombre partits guanyats per partit1: 2 (ha guanyat en vots a la regió 0 i a la 1), nombre partits guanyats per partit2: 1 (ha guanyat en vots a la tercera regió) i nombre empats:0. A continuació segueixen els doctests.

```
def resultatsVotacions(partit1,partit2):
    """
```

```
    >>> resultatsVotacions([5, 2, 8], [0, 0, 9])
    [2, 1, 0]
    >>> resultatsVotacions([17, 13, 40, 100], [18, 10, 40, 0])
    [2, 1, 1]
    """
```

Exercici 3.19 Dissenyeu òptimament una funció, tal que, donada una llista de llistes corresponent a una matriu, retorni el número de files de la matriu amb tots els valors de la columna negatius. Expliqueu on apliqueu esquema/es de cerca/recorregut.

```
def filaneg(m):
    """
    retorna True si hi ha una fila a m amb tot negatius i False en cas contrari
    >>> filaneg([[1, 0, 0, 0], [-1, -2, -3, -4], [-2, -8, -7, -6]])
    2
    """
```

Exercici 3.20 Donada una matriu de pixels corresponent a una matrícula en blanc i negre, implementada com a llista de llistes on el valor 0 correspon al negre i el 255 al blanc, es demana que dissenyeu i implementeu les següents funcions.

[Apartat a] Funció *fBlack*, tal que reb com a paràmetre una matriu corresponent a una matrícula, i retorna True si hi ha una fila tota a negre en la matrícula i False en cas contrari. Completeu la documentació de la funció amb els doctests pertinents.

```
def fBlack(mat):
    """
    return True if there is a black row in mat
    >>> fBlack([[0,0,0,0],[0,255,255,0],[255,255,255,0],[0,255,255,255],
[255,255,255,255]])
    True
    """
```

[Apartat b] Funció *cWhite*, tal que reb com a paràmetre una matriu corresponent a una matrícula, i retorna el nombre de columnes en blanc dins la matrícula. Completeu la documentació de la funció amb els doctests pertinents.

```
def cWhite(mat):
    """
    return the number of white columns in mat
    >>> cWhite([[0,255,255,255],[0,255,255,0],[255,255,255,0],[0,255,255,255],
[255,255,255,255]])
    2
    """
```

Exercici 3.21 La matriu matemàtica complexa. Dissenyeu òptimament una funció, tal que, donada una llista de llistes amb valors enters, corresponent a una matriu matemàtica de qualsevol ordre, retorni True, si hi ha almenys 2 files a la matriu amb almenys 3 valors igual a 8 (Vegeu els exemples). Expliqueu on apliqueu esquema/es de cerca/recorregut. Nota: No es valorarà una solució que no funcioni per matrius de qualsevol ordre.

```
Matriu a) Hauria de retornar True        Matriu b) Hauria de retornar False
8  1  8  1  1  8                         8  8  8
8  8  8  4  2  8                         1  0 -1
1 -3  4  8  3  1                         1  1  2
1  1  1  8  8  8
```

```
def almenys3filesamb8(m):
    """
    retorna True si hi ha almenys 2 files a la matriu amb almenys 3 valors igual a 8
    >>> almenys3filesamb8([[8, 1, 8, 1, 1, 8], [8, 8, 8, 4, 2, 8], [1, -3, 4, 8, 3, 1],
 [1, 1, 1, 8, 8, 8]])
    True
    >>> almenys3filesamb8([[8, 8, 8], [1, 0, -1], [1, 1, 2]])
    False
    """
```

Exercici 3.22 **Les matrix.**

Una matriu dispersa és aquella en què la majoria de valors que conté són zeros. A tal efecte, es decideix emmagatzemar els valors no nuls de dita matriu en un diccionari. Les claus seràn una tupla amb l'indicador de número de fila i número de columna, i el valor associat a dita clau correspondrà a valors no nuls de la matriu. A partir d'ara, d'aquesta representació en direm, representació *dicc-dispersió*. Per exemple, la matriu,

```
0 0 0 1 0
0 0 0 0 3
3 1 0 0 0
0 2 0 0 0
0 0 0 0 1
```

es representaria utilitzant dicc-dispersió com a:

```
m={(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1}
```

Per a resoldre els següents apartats, suposeu que totes les dades proporcionades són correctes i no hi ha inconsistències.

[Apartat a] Crear la funció *sumDiagonal*, tal que donada una matriu quadrada en dicc-dispersió, en sumi els valors que composen la diagonal. En aquest cas, suposeu que la matriu que reb la funció sempre serà quadrada. A continuació segueixen els doctests.

```
def sumDiagonal(m):
    """
    >>> sumDiagonal({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2})
    0
    >>> sumDiagonal({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1})
    1
    """
```

[Apartat b] Crear la funció *GuessOrder*, tal que donada una matriu en dicc-dispersió, retorni una tupla (files,columnes) corresponent a l'ordre de la matriu. A continuació segueixen els doctests.

```
def guessOrder(m):
    """
    >>> guessOrder({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2})
    (4, 5)
    >>> guessOrder({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 1, (3, 1): 2, (4, 4): 1})
    (5, 5)
    """
```

[Apartat c] Crear la funció *compressRow*, tal que donada una matriu en dicc-dispersió, retorni el resultat de comprimir la informació de la matriu utilitzant el següent mètode:

1. s'obté la mitjana dels elements per fila
2. cal emmagatzemar l'indicador de la fila, el nombre de columnes per fila amb valors no nuls i el valor de la mitjana anterior

Per exemple, la matriu en dicc-dispersió

`m={(0, 3): 1, (1, 4): 3, (2, 0): 4, (2, 1): 1, (3, 1): 2, (4, 4): 1}`

quedaria comprimida com a:

`m={(0, 1): 1.0, (1, 1): 3.0, (2, 2): 2.5, (3, 1): 2.0, (4, 1): 1.0}`

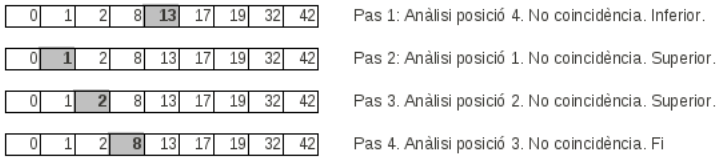
A continuació segueixen els doctests de la funció *compressRow*.

```
def compressRow(m):
    """
    >>> compressRow({(0, 3): 1, (1, 4): 3, (2, 0): 3, (2, 1): 2, (3, 1): 2})
    {(0, 1): 1.0, (3, 1): 2.0, (1, 1): 3.0, (2, 2): 2.5}
    >>> compressRow({(0, 0): 1, (0, 3): 2, (1, 0): 2, (1, 1): 1, (1, 3): 1, (2, 3): 1})
    {(1, 3): 1.3333333333333333, (0, 2): 1.5, (2, 1): 1.0}
```

Nota: La no utilització del format especificat en l'enunciat per la gestió de les matrius, comportarà una puntuació nul·la.

Exercici 3.23 **Cercant un element.** Una de les estratègies utilitzades per comprovar si un element es troba emmagatzemat en una llista és la *cerca binària*. Suposem una llista ordenada ascendentment. El funcionament consisteix en localitzar l'element del centre de la llista i comprovar si és el que busquem. Si aquest no coincideix amb l'element que es cerca, es determina en quina meitat de la llista ha de ser-hi. Es cerca en aquesta meitat repetint el procés les vegades que calgui fins trobar l'element, o bé fins que es determini que no es troba a la llista. Fixeu-vos que es tracta d'una estratègia divideix-i-venceràs, donat que en cada pas es minimitza la llargada de la llista a comprovar.

A continuació s'exemplifica gràficament el seu funcionament, suposant que:
la llista és: $[0, 1, 2, 8, 13, 17, 19, 32, 42]$, el valor a cercar correspon al 3.



Se us demana que implementeu la funció **pura** *binarySearch* tal que, donada una llista d'elements ordenada ascendentment, i un element a cercar, retorni **True** si l'element es troba a la llista i **False** en cas contrari, utilitzant l'estratègia *cerca binària*.

Òbviament, la utilització de les funcions predefinides de Python en la cerca, suposarà una puntuació nul·la. **No** es poden utilitzar *slices* ni la funció *del*. Únicament es permet que utilitzeu la funció que permet obtenir el nombre d'elements d'una llista.

Exercici 3.24 **Algorisme d'inserció.** Suposant una llista correctament ordenada en ordre ascendent, i un valor a inserir, se us demana el disseny **òptim** d'una funció **pura** tal que, donats aquests paràmetres, retorni una nova llista amb l'element inserit en la posició correcta, de manera que la llista continui estant ordenada ascendentment. La llista pot contenir valors repetits. Òbviament, la utilització de les funcions predefinides de Python sobre llistes suposarà una puntuació nul·la. Es pot utilitzar el mètode *len*, les llesques/slices i l'operació de concatenació de llistes. A continuació segueixen els doctests.

```
def insereixValor(valor,llista):
    """
    >>> insereixValor(4, [3, 5, 7, 8])
    [3, 4, 5, 7, 8]
    >>> insereixValor(18, [1, 9, 12])
    [1, 9, 12, 18]
    >>> insereixValor(-4, [1, 2, 3])
    [-4, 1, 2, 3]
    >>> insereixValor(2, [])
    [2]
    >>> insereixValor(45, [-1, 22, 33, 45, 104, 208])
    [-1, 22, 33, 45, 45, 104, 208]
    """
```

Exercici 3.25 **Kerning (Interespai entre lletres).** El kerning en tipografia correspon a l'espai que s'afegeix entre lletres per tal de millorar les funcions visuals en el printatge d'aquestes lletres. Un exemple, és el que es mostra a la Figura 2. (Font: Wikipedia). En el primer cas no s'ha aplicat kerning (Kerning=0) i per tant, es printa una lletra al costat de l'altra. En la resta de casos s'ha aplicat un Kerning donat, i una lletra s'ha aproximat (o encavalcat) a l'altra. La Figura 2, mostra altres exemples amb Kerning=1 i Kerning=2.

Per simplificar, suposarem que representarem les imatges de les lletres com a llistes bidimensionals de 0s i 1s indicant, per cada píxel de la imatge si és blanc (0) o negre (1). Per exemple, a continuació es mostra la representació de la lletra V i de la lletra A.

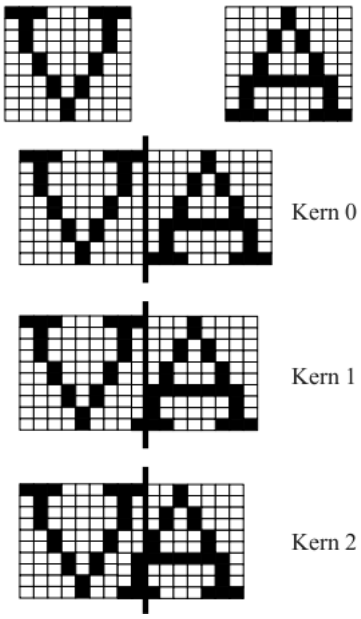


Figura 3.1: Exemple d'aplicació de kerning en el printatge de lletres

lletra V

1 1 1 0 0 0 **1 1 1**
0 **1** 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 **1** 0
0 0 **1** 0 0 0 **1** 0 0
0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 0 **1** 0 0 0 0
0 0 0 0 **1** 0 0 0 0

lletra A

0 0 0 0 **1** 0 0 0 0
0 0 0 0 **1** 0 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0
0 0 **1** 0 0 0 **1** 0 0
0 **1 1 1 1 1 1 1** 0
0 **1 1 1 1 1 1 1** 0
0 **1** 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 **1** 0
1 1 1 0 0 0 **1 1 1**

Se us demana que dissenyeu **òptimament** la funció *aplicarKerning*, tal que, donades dues matrius de 0s i 1s corresponents a lletres i un kerning, retorni la matriu resultant d'ajuntar les dues lletres amb el kerning indicat. Podeu suposar que l'alçada de les dues matrius serà la mateixa. No obstant, l'amplada de cada matriu pot diferir. Suposeu també que el kerning donat no és negatiu i és inferior a l'amplada de qualsevol de les dues imatges. Noteu que si el kerning és zero, la imatge resultant serà l'amplada de les dues imatges posades juntes. A mesura que el valor del kerning augmenti, l'amplada de la imatge resultant disminuirà. Suposeu que les matrius donades seran sempre correctes i no serán mai buides.

NOTA: No està permés l'ús de llesques/slices ni la còpia de llistes. El seu ús comportarà una puntuació nul·la.

A continuació se us mostra el resultat d'aplicar diferents valors de kernings.

lletra V

1 1 1 0 0 0 **1 1 1**
0 **1** 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 **1** 0
0 0 **1** 0 0 0 **1** 0 0
0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 0 **1** 0 0 0 0
0 0 0 0 **1** 0 0 0 0

lletra A

0 0 0 0 **1** 0 0 0 0
0 0 0 0 **1** 0 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0
0 0 **1** 0 0 0 **1** 0 0
0 **1 1 1 1 1 1 1** 0
0 **1 1 1 1 1 1 1** 0
0 **1** 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 **1** 0
1 1 1 0 0 0 **1 1 1**

kerning 0

1 1 1 0 0 0 **1 1 1** 0 0 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 0 0 **1 1 1** 0 0 0 **1 1 1**

kerning 1

1 1 1 0 0 0 **1 1 1** 0 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 0 **1 1 1** 0 0 0 **1 1 1**

kerning 2

1 1 1 0 0 0 **1 1 1** 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 **1 1 1** 0 0 0 **1 1 1**

kerning 3

1 1 1 0 0 0 **1 1 1** 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 0 **1** 0 0 0 0
0 **1** 0 0 0 0 0 **1** 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 0 **1** 0 **1** 0 0 0
0 0 **1** 0 0 0 **1** 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 0 **1** 0 0 0 **1** 0 0
0 0 0 **1** 0 **1** 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 **1 1 1 1 1 1 1** 0
0 0 0 **1** 0 **1** 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 0 **1** 0 0 0 0 0 **1** 0
0 0 0 0 **1** 0 **1 1 1** 0 0 0 **1 1 1**

4 Fitxers, diccionaris i tuples

EXERCICI 4.1 Escriu els resultats d'execució dels següents fragments de codi python

```
import sys
fhand = open(sys.argv[1])
count = 0
for line in fhand:
    if line.startswith('Subject:') :
        count = count + 1
print count
fhand.close()
```

EXERCICI 4.2 Escriu els resultats d'execució des següent fragment de codi python

```
fhand = open('mbox.txt')
count = 0
for line in fhand:
   count = count + 1
print count
fhand.close()
```

EXERCICI 4.3 Un fitxer de text de nom *reals.txt* conté un seguit de nombres reals correctes separats per espai en blanc, corresponents a les estadístiques de pesos d'un determinat producte. Suposeu que aquest fitxer existeix i no està buit. Se us demana que dissenyeu un script tal que, donat el fitxer amb el format especificat,

- Escrigui per pantalla quina és la mitjana aritmètica dels pesos.
- Escrigui en un fitxer anomenat *sreals.txt*, aquells pesos que són superiors a la mitjana.

A continuació es mostra un exemple d'execució amb el contingut del fitxer *reals.txt* i *sreals.txt*.

```
cat reals.txt
15.8 10.0 10.6 0.0 23.4 23.5 23.6 -12.0 11.9 45.8
Mitjana: 15.26
cat sreals.txt
15.8 23.4 23.5 23.6 45.8
```

EXERCICI 4.4 Un fitxer de text, de nom *notes.txt*, conté les notes de laboratori dels estudiants d'infoITIC, amb el format que es mostra en l'exemple. Una nota aprovada és aquella que té valors que comencin per A o B. En cas contrari, l'alume està suspès. Es demana que dissenyeu un script tal que, donat el fitxer *notes.txt*, hi afegeixi el percentatge d'alumnes amb una A i la resta d'alumnes. Comproveu a continuació el contingut del fitxer exemple.

```
Esteve Malacara, A
Laura Tomas, D
Xavier Cascot, A
Pere Rodella, C+
# A continuació segueix la informació que heu d'afegir al fitxer
As 50.0 %
altres 50.0 %
```

EXERCICI 4.5 Dissenyeu un script *comptatge.py* tal que, donat un conjunt de números, obtinguts des de la línia de comandes, escrigui quants cops s'ha introduït cada número. Per exemple,

```
python comptatge.py 11 22 11 44 22 11
11 3
22 2
44 1
```

EXERCICI 4.6

```
d={}
for element in range(12):
    d[element]=12
print d
```

EXERCICI 4.7 Dissenyeu un script que gestioni de manera òptima les pel·lícules "estrella" d'un videoclub virtual en un moment donat. A tal efecte cal dissenyar les següents funcions.

1. Disseny d'una funció que gestioni les dades de 100 usuaris i la pel·lícula que han triat com a pel·lícula estrella. Suposarem que guardem els dnis dels usuaris i per cada pel·lícula, guardem un codi numèric. Suposarem també que en la introducció de dades no s'introduiran dnis repetits i que tant les dades dels dnis com dels codis de pel·lícules són correctes.

   Exemple d'execució (suposant 3 usuaris):

   ```
   Entra dni usuari: 444444444
   Entra codi millor pel·lícula: 123
   Entra dni usuari: 999999999
   Entra codi millor pel·lícula: 44
   Entra dni usuari: 888888888
   Entra codi millor pel·lícula: 123
   ```

2. Disseny d'una funció que mostri per pantalla el codi de la pel·lícula estrella triada per un usuari. En cas que l' usuari no hagi estat donat d'alta, cal mostrar el missatge *Usuari inexistent al videoclub*. Altrament, cal mostrar el codi de la millor pel·lícula triada per l'usuari.

   Exemples d'execució:

   ```
   Entra dni a cercar: 999999999
   Millor pel·lícula per l'usuari 999999999: 44

   Entra dni a cercar: 111111111
   Usuari inexistent al videoclub
   ```

3. Disseny d'una funció que mostri per pantalla els usuaris que han triat una pel·lícula estrella determinada.

   Exemples d'execució:

   ```
   Entra codi de la pel·lícula a consultar: 123
   Llistat d'usuaris que han triat la pel·lícula 123 com a pel·lícula estrella:
   444444444
   888888888

   Entra codi de la pel·lícula a consultar: 209
   Llistat d'usuaris que han triat la pel·lícula 209 com a pel·lícula estrella:
   0 usuaris
   ```

Exercici 4.8 Escriu els resultats d'execució dels fragments de codi Python següents,

```
#Apartat a)
def words():
    f=open("a.txt","w")
    l=[['b','B'],['c','C'],['d','D'],['e','E'],['f','F'],['g','G'],['h','H']]
    for element in l:
        f.write(element[1].lower()+"\n")
    f.write(l[-1][0])
    f.write("\n")
    f.close()
if __name__=='__main__':
   words()

#Apartat b)
new={"Gener":1, "Febrer":2, "Març":3, "Abril":4, "Maig":5, "Juny":6}
new["Desembre"]=12
for item in new.keys():
      print "new[ ", item, " ] = ", new[item]

#Apartat c)
```

```
def buildCodeBook():
    letters ='abcdefghijklnopqrstuvwxyz'
    codeBook = {}
    key = 0
    for c in letters:
        codeBook[key] = c
        key += 1
    return codeBook

def decode(cypherText, codeBook):
    plainText = ''
    for e in cypherText:
        if e in codeBook:
            plainText += codeBook[e]
        else:
            plainText += ' '
    return plainText
if __name__=='__main__':
   print decode([51,-1,20,12,3], buildCodeBook())
```

Exercici 4.9 El joc dels símbols químics.

Donat el següent sistema de gestió de dades corresponent a un joc de símbols químics, *Dipse App for children: Play with Chemical Symbols*, on s'ha codificat un diccionari d amb clau el nom de l'element químic, i valor el símbol associat.

[Apartat a] Se us demana que dissenyeu de manera òptima la funció *eliminarSimbol1(d)*, tal que retorni el resultat d'eliminar de d els elements químics que tenen com a símbol un únic caràcter. Suposeu que el diccionari d es troba inicialitzat correctament. A continuació segueixen els doctests.

```
def eliminarSimbol1(d):
    """
    >>> eliminarSimbol1({'Hidrogen':'H', 'Liti': 'Li', 'Sodi':'Na','Potassi':'K'})
    {'Liti': 'Li', 'Sodi': 'Na'}
    >>> eliminarSimbol1({'Rodi': 'Rh', 'Liti': 'Li', 'Plata': 'Ag', 'Sodi':'Na'})
    {'Rodi': 'Rh', 'Liti': 'Li', 'Plata': 'Ag', 'Sodi': 'Na'}
    """
```

[Apartat b] El sistema ha de permetre una nova funcionalitat que permeti gestionar de manera òptima la classificació dels elements químics per famílies. En concret, *'Metalls alcalins'*, *'Alcalinoterrils'*, *'Lantanids'*, *'Actinids'*, *'Metalls de transicio'*, *'Altres metalls'*, *'Semi-metalls'*, *'No metalls'*, *'Halogens'* i *'Gasos nobles'*.

Cada element químic pertany a una única i família, i cada família té un conjunt d'elements químics que en formen part. La majoria de consultes que es realitzaran en l'aplicatiu seran per, donada una família, obtenir els símbols dels elements químics que en formen part.

Se us demana que:

- Describiu de manera raonada i detallada els canvis que calgui realitzar en l'estructura de gestió de dades. NOTA: Per a respondre aquest apartat no és necessari proporcionar codi python, únicament una explicació raonada.

Exercici 4.10 L'ajuntament de MaiMés ens demana que gestionem les multes d'aparcament recaptades. Heu de considerar que:

- L'ajuntament té un nombre indeterminat de multes de vehicles, i cada vehicle té un nombre indeterminat de multes.
- En una data determinada, un vehicle té una única multa.

De cada multa d'un vehicle ens interessa emmagatzemar la següent informació: data de la multa, codi d'infracció, quantitat i cal especificar si la multa ha estat pagada o no. Després d'un anàlisi d'especificacions, arribem a les conclusions que segueixen.

- Cal accedir de manera òptima a la cerca de multes d'un vehicle.
- Cal accedir de manera òptima a la cerca de multes d'un vehicle per data

Se us demana que justifiqueu i detalleu quin mecanisme de gestió de dades utilitzaríeu. Podeu incloure un exemple, però no heu d'escriure el codi Python per la lectura de dades, ni tampoc el codi Python que dugui a terme tota la gestió.

Exercici 4.11 Se'ns demana implementar una funcionalitat òptima que permeti gestionar l'encriptament i desencriptament de missatges que s'han d'enviar de manera privada entre un emissor i un receptor. En l'encriptació de missatges entre emissor i receptor es vol que cada lletra de l'abecedari se substitueixi per una altra lletra (a aquesta darrera l'anomenarem lletra encriptada). Suposarem que tant la lletra original com la encriptada han de correspondre a lletres minúscules.

Per exemple, es podria decidir que la lletra 'm' s'encripta amb una 'x', la lletra 'e' amb una 'z', la lletra 's' amb una 'f', la lletra 'a' com 'y' i la lletra 'g' com a 'w'. Per tant, amb aquesta codificació, el missatge 'message' s'encriptaria com a 'xzffywz'

Fixeu-vos que cada lletra té una única lletra encriptada i que cada lletra encriptada pertany a una única lletra original.

[Apartat a] Justifiqueu com emmagatzemaríeu de manera òptima la lletra original i la correspondència amb la lletra codificada.

[Apartat b] Dissenyeu una funció pura de nom *encripta*, tal que, donada una paraula, retorni la paraula encriptada.

[Apartat c] Dissenyeu una funció pura de nom *desencripta*, tal que, donada una paraula encriptada, retorni la paraula original.

Exercici 4.12 Dissenyeu una funció que, donada una llista d'elements l i un enter n, retorni una tupla amb els elements de la llista l que ocupin posicions múltiples de n.

```
def tuplaMultiples(l,n):
    """
    >>> tuplaMultiples([],2)
    ()
    >>> tuplaMultiples([1,-9,12,25,44,33,125,67,245],3)
    (1, 25, 125)
    """
```

Exercici 4.13 Escriu els resultat d'execució des següent fragment de codi python

```
t = 1,'a',3.14
print t
print t + ("bill",)
print "t * 3 equals: ", myTuple * 3
print t[0]
print t[-1]
print t[0:2]
print len(t)
```

Exercici 4.14 Coordenades GPS.

Una coneguda empresa vol implantar un aplicatiu que permeti fer amistats properes, tot utilitzant les coordenades GPS on s'ubica cada persona. A tal efecte, ens cal conéixer el funcionament de les coordenades GPS.

Las coordenades GPS estan formades por dues components que són latitud i longitud. Les dues principals unitats de mesura són: 1)coordenades decimals i 2) coordenades sexagesimals.

La latitud i longitud expresada com a 1) coordenades decimals, corresponen a valors decimals. En el cas de la latitud, són valors correctes aquells entre 0..90 i 0.. − 90. En el cas de la longitud, són valors correctes aquells entre 0..180 i 0.. − 180. Per exemple, un valor vàlid expressat en coordenades decimals per a una latitud correspondria al valor 42.1361.

En el cas d'expressió de la latitud i longitud com a 2) coordenades sexagesimals, cal tenir en compte que aquestes tenen tres components: graus, minuts i segons. Cadascun d'aquests components sol ser un número enter, però es pot usar un número decimal en els segons si es desitja major precisió. A diferència de les coordenades decimals, les sexagesimals no poden ser negatives. Per exemple, un valor vàlid expressat en sexagesimal per a una latitud correspondria a $42^o8'9.96''$. A partir d'ara, ho expressarem com a $42 : 8 : 9.96$

[Apartat a] Per tal de realitzar la conversió de coordenades sexagesimals a coordenades decimals, cal aplicar la següent fórmula:

$graus + (minuts/60) + (segons/3600)$

Per exemple, $42 : 8 : 9.96$ correspon a un format latitud/longitud $42 + (8/60) + (9.96/3600) = 42.1361$, sense pèrdua de decimals.

Per a resoldre els següents apartats, suposeu que totes les dades proporcionades són correctes i no hi ha inconsistències.

Se us demana crear la funció pura *sexagesimalDecimal*, tal que, donat un valor correcte corresponent a latitud/longitud en sexagesimal, retorni la corresponent coordenada decimal. A continuació segueixen els doctests.

```
def sexagesimalDecimal(a):
    """
    >>> sexagesimalDecimal('42:8:9.96')
    42.1361
    >>> sexagesimalDecimal('42:23:60.0')
    42.4
    """
```

[Apartat b] Per tal de realitzar la conversió de coordenades decimals a coordenades sexagesimals, cal aplicar el procediment següent. Prenent el resultat anterior, 42.1361 es converteix de la següent manera,

1. La part no decimal correspon als graus. En aquest cas, 42^o.

2. Multiplicar la part decimal per 60. Els valors han de ser entre el rang 0..23. Exemple: $0.1361 * 60 = 8.166$. Obtenim $8'$. La part decimal pendent és, en aquest cas, 0.166.

3. Multiplicar la part decimal resultant de l'apartat anterior, per 60. Els valors han d'estar entre el rang 0..60. Exemple: $0.166 * 60 = 9.96$. Obtenim $9.96''$.

Se us demana crear la funció pura *decimalSexagesimal* tal que, donat un valor correcte corresponent a latitud/longitud en decimal, retorni la corresponent coordenada sexagesimal. A continuació segueixen els doctests.

```
def decimalSexagesimal(a):
    """
    >>> decimalSexagesimal(42.1361)
    '42:8:9.96'
    >>> decimalSexagesimal(42.4)
    '42:23:60.0'
    """
```

[Apartat c] Per simplificar, suposarem que dues persones estan properes si tenen el mateix valor en graus per la latitud.

Donat un fitxer de text, el nom del qual ve donat per la línia de comandes, tal que conté un llistat de coordenades correctes en format decimal corresponents a la latitud i un identificador-persona, seguint el format següent,

```
42.4 user1
42.1361 user2
34.4 user3
11.2 user4
42.8 user5
34.6 user6
11.8 user7
10.2 user8
```

Es demana que dissenyeu la funció *parellesProperes*, tal que, retorni un llistat de tuples amb les combinacions de parelles properes. Per l'exemple anterior, el resultat hauria de ser,

```
[('user1', 'user2'), ('user1', 'user5'), ('user2', 'user5'), ('user3', 'user6'),
('user4', 'user7')]
```

L'exemple de crida a la funció *parellesProperes* és el que segueix.

```
import sys
if __name__=='__main__':
    if len(sys.argv)==1:
        print 'Manquen dades'
    else:
        print parellesProperes(sys.argv[1])
```

Exercici 4.15

Escriviu els doctests/resultats d'execució dels fragments de codi Python següents,

```
#Apartat a)
def desxifra(dusuaris):
    """
    >>> desxifra({'11111111':('Jack','Sparrow',False,[[],[]])})
    #apartat a.1)
    >>> desxifra({'22222222':('Clark','Kent',True,[[],[]]),'33333333':
('Indiana','Jones',True,[[], []])})
    #apartat a.2
    """
    k=0
    for element,valor in dusuaris.items():
       if valor[2]:
           k+=1
    return k
#Apartat b)
def misteri(A,B):
    """
    >>> misteri((1,2,3),(0,1,2,3,8))
    #apartat b.1
    >>> misteri((0,2,9,15,34),(2,10,9,12,17,15,34))
    #apartat b.2
    """
    for element in A:
        if element not in B:
           t=False
           break
    else:
        t=True
    return t
#Apartat c)
def f(s, d):
    for k in d.keys():
        d[k]=0
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 0
    return d
def addUp(d):
    result = 0
    for k in d:
        result += d[k]
    return result
if __name__=='__main__':
```

```
d1={}
d2=d1
d1=f('abbc', d1)
print addUp(d1)      #apartat c.1
d2 = f('bbcaa', d2)
print addUp(d2)      #apartat c.2
print f('', {})      #apartat c.3
```

EXERCICI 4.16 Escriu un script tal que demani paraules a l'usuari fins que s'introdueixi una paraula màgica. Una paraula màgica és aquella tal que conté el número 0 o bé conté algun símbol. Es considera símbol tot caràcter que no es ni lletra ni número. Finalment, l'script ha d'escriure per pantalla quantes paraules s'han introduït abans de la introducció de la paraula màgica.

EXERCICI 4.17 L'agència C-UPC ha detectat que un fitxer és corrupte si conté almenys una paraula que apareix en el fitxer 666 vegades o més. Donat el fitxer *chequeja.txt*, que conté paraules separades per espai en blanc, es demana que dissenyeu la funció *esCorrupte*, tal que retorni -1 si el fitxer és corrupte i el valor 22 en cas contrari. En cas que sigui corrupte, la funció ha de mostrar també quina és la paraula que ha portat a la detecció de fitxer corrupte. Expliqueu on apliqueu esquema/es de cerca/recorregut.

EXERCICI 4.18 Dissenyeu òptimament una funció, tal que, donada una llista de llistes amb valors enters, corresponent a una matriu de qualsevol ordre, retorni True, si hi ha un marc dins la matriu. Una matriu té un marc quan la primera i última fila, i la primera i última columna tenen totes valors iguals a 1. (Vegeu els exemples). Expliqueu on apliqueu esquema/es de cerca/recorregut. Nota: No es valorarà una solució que no funcioni per matrius de qualsevol ordre.

```
Matriu a) Hauria de retornar True
1  1  1  1  1  1
1  2  3  4  2  1
1 -3  4  8  3  1
1  1  1  1  1  1

Matriu b) Hauria de retornar False
1  1  1
1  0 -1
1  1  2

def conteMarc(m):
    """
    retorna True si hi ha una fila a m amb tot negatius i False en cas contrari
    >>> conteMarc([[1, 1, 1, 1, 1, 1], [1, 2, 3, 4, 2, 1], [1, -3, 4, 8, 3, 1],
[1, 1, 1, 1, 1, 1]])
    True
    >>> conteMarc([[1, 1, 1], [1, 0, -1], [1, 1, 2]])
    False
    """
```

EXERCICI 4.19 L'empresa AppCreator ens demana assessorament en la creació d'un joc de símbols químics que permeti, a través de la petició d'un nom de símbol químic, mostrar-ne el seu nom complet, i a l'inversa. Se us demana quina/es eina/es de gestió de dades utilitzaríeu. Justifiqueu la resposta. Nota: No se us demana el codi python que permeti la resolució del problema, únicament la descripció de les eines de gestió de dades que considereu necessària.

```
Un exemple d'execució:

Introdueixi nom símbol químic: Hidrogen
El símbol és H

Un altre exemple d'execució:

Introdueixi símbol: Na
El nom complet és Sodi
```

EXERCICI 4.20 **Acrònim-encriptació de missatges.** Com ja coneixeu, una de les formes d'encriptació de paraules és el xifratge Cèsar. A continuació se us demana que implementeu una altra estratègia de resolució, que anomenarem **acrònim-encriptació**. Per tal de codificar un missatge amb la tècnica *acrònim-encriptació*, simplement cal agafar la primera lletra de cada paraula, obtenint així un nou missatge. Per exemple, donat el missatge *"les vacances de primer quadrimestre estan a punt"*, el resultat de l'acrònim-encriptació seria el missatge *"lvdpqeap"*. Per tal de resoldre els apartats que segueixen, podeu suposar que els missatges no contindran lletres majúscules.

[**Apartat 1.a**] Se us demana que realitzeu la funció pura *encriptaAcronimous*, tal que, donat un missatge, retorni el missatge codificat segons la tècnica acrònim-encriptació. Suposeu que el separador de paraules dins el missatge és un espai en blanc, i que el missatge únicament contindrà lletres alfabètiques.

```
def encriptaAcronimous(s):
    """
    >>> encriptaAcronimous("les vacances de primer quadrimestre estan a punt")
    'lvdpqeap'
    """
```

[**Apartat 1.b**] En aquest apartat se us demana implementar la funció *decodificaAcronimous*, encarregada de decodificar un missatge secret encriptat amb *acrònim-encriptació* seguint l'estratègia que es detalla a continuació.

Donat un missatge secret, *decodificaAcronimous* construeix un nou missatge, on cada lletra del missatge original es canvia per una paraula que comenci per aquesta lletra. En cas que el caràcter no sigui alfabètic, aquest es manté. Finalment, la funció ha de retornar aquest nou missatge. Per exemple, *decodificaAcronimous* amb el missatge "nj, nr", pot retornar "nova jornada, nous reptes".

Per implementar la funció, utilitzeu la funció que se us proporciona resolta, de nom **obtenirParaula**, tal que, donat un caràcter, retorna una paraula aleatòria que comenci per aquest caràcter.

```
def decodificaAcronimous(s):
```

```
    """
    >>> decodificaAcronimous("nj, nr")
    'nova jornada, nous reptes'
    """
```

[**Apartat 1.c**] En aquest cas se us demana gestionar la funcionalitat que permet obtenir una paraula aleatòria que comenci per una lletra. En aquest cas se us demana **1.c.1)** un script que llegeixi prèviament un fitxer de text que contingui un conjunt de paraules, i emmagatzemi òptimament cada paraula en una estructura. Aquesta estructura ha de ser òptima de manera que donada una lletra, obtingui la llista de paraules del fitxer que comencin per aquesta lletra. Suposeu que el fitxer de paraules existeix amb el nom *paraules.txt* i que cada paraula en el fitxer està separada d'una altra per un espai en blanc. Tingueu en compte que les paraules del fitxer no segueixen cap ordre. I que una mateixa paraula pot estar repetida en el fitxer vàries vegades.

Finalment, us cal **1.c.2**) implementar la funció *obtenirParaula*, tal que utilitzi la estructura de dades anterior per, donat un caràcter, retornar una paraula aleatòria de la llista que comenci per aquest caràcter.

Se us demana que el codi Python dels passos c.1) i c.2) amb el funcionament esperat.

EXERCICI 4.21 Suposant que el següent script està contingut en exercici.py i que executem *python exercici.py sortida.txt soc un bon estudiant i avui aprovare* Què s'escriu per pantalla i què s'escriu en el fitxer sortida.txt després de l'execució d'aquest script exercici.py? Què conté sys.argv?

```
import sys
import string
f=open(sys.argv[1],"w")
print "Open out", sys.argv[1]
s=0; l=[]; d={}; vow="aeiouAEIOU"
for cad in sys.argv[2:]:
    cadAuxiliar=""
    for let in cad:
        if let in string.digits:
            s+=int(let)
        elif let not in vow:
            cadAuxiliar+=let
        else:
            d[let]=d.get(let,0)+1
    l+=[cadAuxiliar]
l.sort()
i=0
while i<len(l):
    print l[i]
    i+=1
llista=[]
for k,v in d.items():
    llista.append((v,k))
for element in llista:
    f.write(element[1]+"-"+str(element[0])+"\n")
f.close()
```

EXERCICI 4.22 Completa els doctests que segueixen.

```
def niIdea(d):
    """
    >>> niIdea({'Hidrogen':'H', 'Liti': 'Li', 'Sodi':'Na','Potassi':'K'})
    #apartat 2.1
    >>> niIdea({'Rodi': 'Rh', 'Liti': 'Li', 'Plata': 'Ag', 'Sodi':'Na'})
    #apartat 2.2
```

```
    """
    for clau,valor in d.items():
        if len(valor)==1:
            del d[clau]
    return d
```

Exercici 4.23 Escriu el resultat de l'execució.

```
def f(s, d):
    for k in d.keys():
        d[k]=0
    for c in s:
        if c in d:
            d[c] += 1
        else:
            d[c] = 0
    return d
```

```
def addUp(d):
    result = 0
    for k in d:
        result += d[k]
    return result
```

```
if __name__=='__main__':
    d1 = {}
    d2 = d1
    d1 = f('abbc', d1)
    print addUp(d1)
    d2 = f('bbcaa', d2)
    print addUp(d2)
    print f('', {})
```

Exercici 4.24 Les eleccions 11D. Dissenya òptimament una funció tal que donat un fitxer de text (resultats.txt) que conté els resultats de les eleccions, amb el format, per cada línia, el codi partit i numero vots, se us demana que dissenyeu òptimament la funció calculaResultat, tal que escrigui en un nou fitxer (resultatsnous.txt), línia per línia cada codi partit i el percentatge de vots que ha obtingut respecte al total. A continuació segueixen els exemples d'execució.

```
#exemple contingut resultats.txt
partit1 1620973
partit2 336375
partit3 366494
partit4 102870
partit5 734910
partit6 522209
partit7 29785
partit8 14390
partit9 1158
partit10 326
```

```
#exemple contingut resultatsnous.txt
partit1 43.4636639326%
partit2 9.01932972069%
partit3 9.82692003464%
partit4 2.75828598548%
partit5 19.7053752658%
partit6 14.0021557907%
partit7 0.798634665866%
partit8 0.38584364082%
partit9 0.0310498218255%
partit10 0.00874114155019%
```

Exercici 4.25 Detecció d'usuaris duplicats. Amb l'objectiu d'esbrinar si hi ha usuaris duplicats en dues xarxes socials, se us demana, que, donada la informació provinent de dues xarxes socials, retorneu el llistat d'usuaris que es troben en ambdues xarxes. Per simplificar, suposarem que d'un usuari emmagatzemarem el seu nick (únic per cada usuari) i la seva paraula d'accés (no encriptada). A continuació segueixen els doctests de la funció que heu de resoldre òptimament.

```
def dicciocopy(d1,d2):
    """
    >>> dicciocopy({'juliaRoberts':'jaja22','shakira':'idon24','brattPitt':'noende',
'angelJol':'pepeps'},
```

```
    {'brattPitt':'noende','acdc':'88arroba'})
    {'brattPitt': 'noende'}
    >>> dicciocopy({'bandarra': 'carxofa'}, {})
    {}
    """
```

EXERCICI 4.26 El traductor. Ens passen un fitxer de text de nom *traduccions.txt* amb el següent format: El fitxer consisteix d'una paraula anglesa, seguida de qualsevol nombre de línies de la forma *xx=translation*, on *xx* correspon a lletres estardarditzades corresponents a codis de llenguatge, tal com, *de* per alemany, *cat* per català o *fr* per francès. Un exemple de fitxer seria el que segueix.

```
Cancel
de=Abbrechen
es=Cancelar
fr=Annuler
close
de=Schlieben
cat=Tancar
fr=Fermer
OK
fr=Approuver
Open
de=Offnen
es=Abrir
fr=Ouvrir
Today
cat=avui
```

Aquest fitxer ens diu, per exemple que la paraula anglesa *Cancel* s'hauria de visualitzar en alemany com a *Abbrechen*. En cas que no hi hagi idioma de traducció per a una paraula, es deixa la paraula anglesa. En cas que la paraula anglesa no es trobi en el fitxer, la traducció és "NotF"

Suposeu que tots els caràcters són d'expanded Unicode i per tant, no hi haurà accents ni caràcters especials. Se us demana que implementeu òptimament en python la funció *translate*, tal que, donat un nom de fitxer, una frase correcta (en anglès), i un codi de traducció, retorni la frase traduïda.

Nota: Podeu utilitzar funcions addicionals per tal d'optimitzar el codi solució.

1. Expliqueu on apliqueu esquema/es de cerca/recorregut.

2. Expliqueu quina eina de gestió de dades òptima utilitzeu.

```
def translate(nom,frase,codi):
    """
    >>> translate("traduccions.txt","Cancel OK Open close","fr")
    'Annuler Approuver Ouvrir Fermer'
    >>> translate("traduccions.txt","Cancel all Open close Today","fr")
    'Annuler NotF Ouvrir Fermer Today'
    """
```

Exercici 4.27 **Query parameters.** Els Query parameters són una manera de gestionar la informació dins les URLs. Per exemple, quan cerquem a google informació sobre els query parameters, aquesta és la URL que genera automàticament Google.

```
http://www.google.com/search?sourceid=chrome&ie=UTF-8&q=query+parameters
```

Tal com es pot veure, Google emmagatzema la següent informació: que la cerca es va fer des del navegador chrome, que l'entrada de text era codificada amb format UTF-8, i que la cerca realitzada és 'query parameters.'

Més genèricament, els query parameters en les urls tenen el format que segueix.

```
http://website.com/?KEY1=VALUE1&KEY2=VALUE2...
```

Se us demana que:

[**Apartat 1.a**] Dissenyeu la funció *obtenirQueryParameters(url)* tal que, donada una URL, mostri per pantalla la informació dels query parameters. Per a l'exemple anterior, caldria que printés la informació que segueix. Adoneu-vos que el símbol + en les query parameters és indicador d'espai en blanc.

```
http://www.google.com/search?sourceid=chrome&ie=UTF-8&q=query+parameters
sourceid: chrome
ie: UTF-8
q: query parameters
```

O bé en un exemple de Facebook,

```
http://graph.facebook.com/search?q=watermelon&type=post
q: watermelon
type: post
```

Com que treballeu a Facebook/Google, necessiteu escriure aquestes funcions **òptimament**, o no podran ser evaluades. Podeu utilitzar les funcions predefinides de Python que necessiteu. A continuació segueixen els doctests.

```
def obtenirQueryParameters(url):
    """
    >>> obtenirQueryParameters("http://www.google.com/search?sourceid=chrome&ie=UTF-8
&q=query+parameters")
    sourceid: chrome
    ie: UTF-8
    q: query parameters
    >>> obtenirQueryParameters("http://graph.facebook.com/search?q=watermelon&type=post")
    q: watermelon
    type: post
    >>> obtenirQueryParameters("https://www.google.es/search?q=how+to+use+prezi
&client=ubuntu&channel=cs&aq=f&oq=how+to+use+prezi&aqs=chrome.0.57j65.18849
&sourceid=chrome&ie=UTF-8")
    q: how to use prezi
    client: ubuntu
    channel: cs
```

```
    aq: f
    oq: how to use prezi
    aqs: chrome.0.57j65.18849
    sourceid: chrome
    ie: UTF-8
    """
```

[**Apartat 1.b**] Convertiu la funció anterior en *booleana* i afegiu que retorni **True** si la consulta realitzada té a veure amb *how to*. En els doctests anteriors, únicament el darrer test retornaria **True**. Els dos primers retornarien **False**. **Nota:** No és necessari que copieu de nou íntegrament el codi de la funció, afegiu indicació del codi que us caldria afegir i la seva ubicació.

[**Apartat 1.c**] Suposant que disposeu de la funció *obtenirQueryParameters* de l'Apartat 1.b, correctament implementada, escriviu un script tal que donats un conjunt de URLs que es troben en el fitxer *queries.txt* amb el format: una línia per URL, escrigui en el fitxer *estadistiques.txt*, quin percentatge d'aquestes URLs té a veure amb consultes de tipus *how to*.

```
#contingut fitxer queries.txt
https://www.google.es/search?q=how+to+use+prezi&client=ubuntu&channel=cs&aq=f
&oq=how+to+us&aqs=chrome.0.59j57j6213.2339&sourceid=chrome&ie=UTF-8
https://www.google.es/search?q=how+to+program+in+java&client=ubuntu&channel=cs
&aq=f&oq=how+to+program&aqs=chrome.0.59j57j60j6213.2098&sourceid=chrome&ie=UTF-8
https://www.google.es/search?q=pictures+of+nutshell&client=ubuntu&channel=cs
&aq=f&oq=pictures+of+nutshell&aqs=chrome.0.57.3219&sourceid=chrome&ie=UTF-8
https://www.google.es/search?client=ubuntu&channel=fs&q=selection+sort+
algorithm&ie=utf-8&oe=utf-8&gfe_rd=cr&dcr=0&ei=6rdUWvPsCZLY8geNwpPYBg

#el contingut del fitxer estadistiques.txt hauria de ser
50%
```

Exercici 4.28 **Xarxes socials.** Suposant que teniu una xarxa social representada com un graf tal com mostra la Figura 1 esquerra.

Suposeu que dita xarxa s'ha representat amb un diccionari, de nom *xarxa*, on cada clau d'aquest serà el nick d'una persona, i el valor associat, una llista de nicks de les persones amb les quals és amic. Suposeu ara que cada persona de la xarxa social té un animal preferit. A tal efecte, es creu un nou diccionari, *animals*, on per cada clau corresponent al nick d'un usuari de la xarxa social, s'emmagatzema com a valor el seu animal preferit.

Direm que una persona de la xarxa social és un *hipster* si el seu animal preferit és diferent de tots els animals preferits dels seus amics. La Figura 1 dreta exemplifica aquesta situació. Per exemple, en la xarxa social, el conjunt de hipsters correspon a AngJolie, JuliaRoberts, Shakira i DavidBeckham. Se us demana que, implementeu **òptimament** la funció *hipster*, tal que, donats els dos diccionaris anteriors, retorni la llista de nicks d'aquells usuaris que són hipster. A continuació segueixen els doctests.

```
def hipster(d,e):
    """
    >>> hipster({'AngJolie': ['BradPitt', 'JuliaRoberts'], 'JuliaRoberts':
['AngJolie', 'BradPitt', 'Shakira', 'DavidBeckham'],'BradPitt': ['AngJolie',
'JuliaRoberts', 'TomCruise', Shakira'], 'TomCruise': ['BradPitt', 'Shakira'],
```

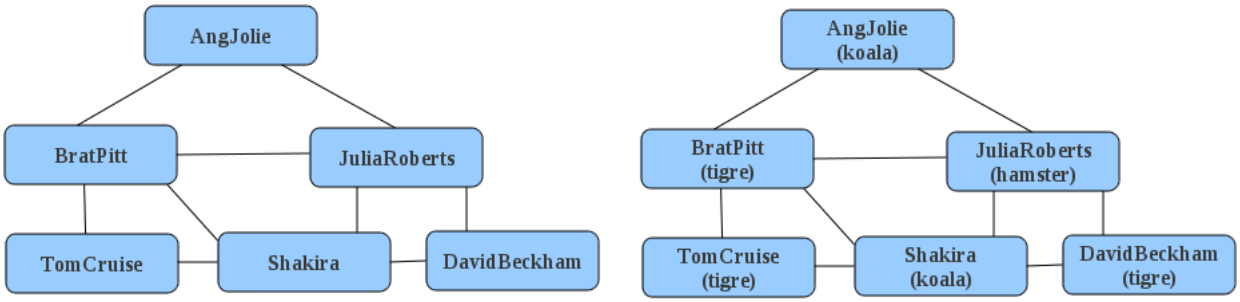


Figura 4.1: La xarxa social

```
'Shakira': ['TomCruise', 'BradPitt', 'JuliaRoberts','DavidBeckham'],
'DavidBeckham': ['JuliaRoberts', 'Shakira']}, {'AngJolie': 'koala',
'JuliaRoberts': 'hamster', 'BradPitt': 'tigre', 'TomCruise': 'tigre',
'Shakira': 'koala', 'DavidBeckham': 'tigre'})
    ['AngJolie', 'JuliaRoberts', 'Shakira', 'DavidBeckham']
    """
```