Segundo control de programación en Python de 2017

Supón un arquivo de texto con medidas de temperaturas cada certo tempo, no que cada liña contén as temperaturas rexistradas nun mes e o número de medidas pode variar dun mes a outro. Un exemplo deste arquivo sería (que tes que crear cun editor de textos):

```
5 4.5 7 8
10 13 9.6
2 4
20 17 21 24
9 23 16
34 40 32
4 7 5
26 34
23
```

Escribe un programa chamado `exame1.py` que lle pida ó usuario o nome do arquivo de texto e realice as seguintes operacións:

1. Escribe unha función chamada `temperaturaMedia()` (cos argumentos axeitados) que lea o arquivo de texto coas temperaturas e calcule a temperatura media de cada mes.

2. Chamar a función `temperaturaMedia()` e asigna ó vector **T** as temperaturas medias de cada mes. Calcular o polinomio de grao 3 que mellor representa os valores da temperatura (vector **T**). Representa gráficamente o curva do polinomio en azul e o vector **T** en verde. Se a temperatura máxima e mínima non están no intervalo [0,30], marcaas no gráfico cun asterisco vermello. Pon títulos os eixos e lendas no gráfico.

3. Calcula o número de veces que terías que concatenar **T** para que a suma de temperaturas fose maior que 1000. Visualiza o resultado na pantalla.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
nome=raw_input('Nome do arquivo: ')
def temperaturaMedia(nomeArquivo):
    temp=[];
    try:
        filein=open(nomeArquivo, 'r')
        for l in filein:
            temp.append(mean(array(l.rsplit(), dtype='float')))
        filein.close()
    except IOError:
        print "Erro abrindo ", nomeArquivo
    return array(temp)

T=temperaturaMedia(nome)
nmeses=len(T)
meses=range(1, nmeses+1)
p=polyfit(meses, T, 3)
x=linspace(1,nmeses, 20)
clf()
plot(meses, T, 'g*', label=u'Temp. media')
plot(x, polyval(p, x), 'b-', label=u'Polinomio')
tmin=min(T)
tmax=max(T)
```

```
if tmin < 0:
    plot(argmin(T)+1, tmin, 'ro', label=u'Minimo')
if tmax > 30:
    plot(argmax(T)+1, tmax, 'ro', label=u'Maximo')
grid(True)
xlabel('Meses')
ylabel('Temperatura media')
legend(loc='upper left')
show(False)
s=sum(T)
suma=s
n=1
while suma < 1000:
    suma = suma + s
    n= n+1
print "Necesito concatenar ", n, " veces"
```

Segundo control de programación en Python de 2017

Escribe un programa en Python chamado `exame2.py` que lea do teclado un número enteiro positivo n que sexa maior a 3 e realiza as seguintes operacións:

1. Calcule un vector **p** de dimensión n, onde cada elemento p_i ven dado pola expresión:

$$p_i = \int_0^1 5x \cos\left(\frac{i\pi x}{20}\right) dx \qquad i = 0, \ldots, n-1$$

   Calcula a integral usando `sympy` e ordea o vector **p** de menor a maior.

2. Define unha función `calculaMatriz()` cos argumentos axeitados que calcule unha matriz **a** cadrada de orde n, onde cada elemento a_{ij} será o número de elementos do vector **p** (comezando polo índice 0 e volvendo a comezar polo índice 0 cando se chegue ó final do vector) para que se cumpra a condición:

$$\sum_k p_k < (i+1)(j+1)^2$$

3. Chama a función `calculaMatriz()` desde o programa principal. Pide ó usuario o nome dun arquivo de texto e garda nel a fila i, $i = 0, \ldots, n-1$ de matriz **a** se:

$$\sum_{j=0}^{n-1} a_{ij} \leq \sum_{j=0}^{n-1} a_{ji}$$

   e, en caso contrario, garda un vector fila contendo os valores da columna i da matriz **a**. En ambos casos, garda os números como enteiros.

4. Realiza un mapa de calor da matriz **a** e gardao no arquivo `grafico.png`.

NOTA: Se consideras un $n = 4$, o vector **p** será:

```
p=  [ 2.5          2.48459987  2.43865237  2.36291148]
```

e o arquivo de texto resultante debe conter os seguintes datos:

```
1 1 2 2
2 4 5 7
2 5 12 20
2 7 15 27
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-

from matplotlib.pyplot import *
from sympy import *
from sys import exit
import numpy as np
n=-1
while n < 1:
    n=int(raw_input('n= '))

p=np.zeros(n)

x=symbols('x')
for i in range(n):
    aux=integrate(5*x*cos(i*pi*x/20), (x, 0,1))
```

```
    p[i]=aux.evalf()
p.sort()
print "p= ", p
def calculaMatriz(z):
    n=len(z)
    b=np.zeros([n,n])
    for i in range(n):
        for j in range(n):
            k=0; suma=0; nelementos=0
            umbral=(i+1)*(j+1)**2
            while suma< umbral:
                suma=suma + z[k]
                k=k+1
                nelementos= nelementos+1
                if (k == n):
                    k=0
            b[i,j]=nelementos
    return b

a=calculaMatriz(p)
af=a.copy();
for i in range(n):
    if sum(a[i]) > sum(a[:,i]):
        af[i]=a[:,i].copy()
nome=raw_input('Arquivo de saida: ')
try:
    np.savetxt(nome, af, '%i')
except IOError:
    exit('Erro escribindo en %s\n' % (nome))

clf(); imshow(a); title('Mapa de calor'); show(False)
```

Segundo control de programación en Python de 2017

Utilizando o editor de texto crea o arquivo `datos4.txt` que contén os vectores **x** (primeira liña) e **y** (segunda liña). Por exemplo, `datos4.txt` podería contér os datos:

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

Escribe un programa en python chamado `exame4.py` que pida ó usuario o nome dun arquivo e realice as seguintes operacións:

1. Define a función chamada `leeArquivo()` que lea un arquivo de texto e almacene as dúas primeiras liñas en dous vectores.
2. Chama a función `leeArquivo()` para asociar as variables **x** e **y** o contido do arquivo introducido polo usuario. Pídelle ó usuario outro arquivo mentres que a dimensión de **x** non sexa menor que a dimensión de **y** ou haxa algún erro abrindo o aquivo. Visualiza na pantalla os vectores **x** e **y** finais.
3. Visualiza na pantalla unha mensaxe informando se o vector **x** está contido ou non no vector **y**. No caso de que estea contido visualiza as posicións do vector **y** nas que se da a primeira coincidencia.
4. Xera a matriz **a** de orde n (sendo n a lonxitude do vector **y**) coa expresión:

$$a_{ij} = \frac{y_i^3 \cos(y_i \pi)}{e^{y_i - y_j}} \qquad i, j = 0, \ldots, n-1 \tag{1}$$

   e represéntaa como unha superficie en 3D, poñendolle títulos os eixos.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-

from numpy import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
def leeArquivo(nomeArquivo):
    v=[]; w=[]
    try:
        myfile=open(nomeArquivo, 'r')
        l=myfile.readline()
        aux=l.rsplit()
        for i in range(len(aux)):
            v.append(int(aux[i]))
        l=myfile.readline()
        w=array(l.rsplit(), 'int')
    except IOError:
        pass
    return [v, w]
nome=raw_input('Nome arquivo: ')
[x, y]=leeArquivo(nome)
while len(x) >= len(y):
    nome=raw_input('Nome arquivo: ')
    [x, y]=leeArquivo(nome)
print 'x= ', x
print 'y= ', y
nx=len(x); ny=len(y)
contido=False
for i in range(ny-nx):
    contido=True
```

```
    for j in range(nx):
        if y[i+j] != x[j]:
            contido=False
            break
    if contido:
        p=i
        break
if contido:
    print 'Vector x contido en y nas posicions %i-%i\n' % (p, p+nx)
else:
    print 'Vector x non contido en y'
a=zeros([ny,ny])
for i in range(ny):
    for j in range(ny):
        a[i,j]=y[i]**3*cos(y[i]* 6*pi)/exp(y[i]-y[j])

fig = figure()
ax = Axes3D(fig)
v=arange(ny)
X, Y=meshgrid(v,v)
ax.plot_surface(X,Y, a, rstride=1, cstride=1, cmap='hot')
ax.set_xlabel('eixo x')
ax.set_ylabel('eixo y')
ax.set_zlabel('eixo z')
show(False)
```

Segundo control de programación en Python de 2017

Escribe un programa chamado `exame5.py` que lea por teclado un número enteiro maior ou igual a $n = 10$ comprobando que o usuario introduce un número correcto. O programa debe realizar as seguintes operacións:

1. Crear un vector **x** de dimensión n cos n primeiros números de Fibonacci, dados por $x_0 = 1$, $x_1 = 1$, $x_i = x_{i-1} + x_{i-2}$, $i = 2, \ldots, n-3$ e amósao na pantalla.

2. Define unha función `transforma()`, cos argumentos axeitados, que convirta un vector **v** de dimensión n nunha matriz **b** cadrada de orde m que rechee o triángulo superior e inferior da matriz cos elementos do vector de xeito que $b_{ij} = b_{ji}$ con $i \neq j$. Polo tanto, para que o vector colla dentro do triángulo superior/inferior da matriz, a orde m debe verificar que $\frac{m^2-m}{2}$ sexa o mínimo número enteiro igual ou maior que n, e operando $m = \left\lceil \frac{1+\sqrt{1+8n}}{2} \right\rceil$ (no caso de $n = 10$, $m = 5$). Nos elementos da diagonal $b_{ii} = v_i$, $i = 0, \ldots, m-1$.

3. Desde o programa principal chama a función `transforma()` para convertir o vector **x** na matriz **a**. Representa a matriz **a** como un mapa de calor e ponlle un título e etiquetas nos eixos x e y. Garda o mapa de calor no arquivo `figura5.png`.

4. Calcula o número de elementos (k) da matriz **a** que será necesario sumar (percorrendo a matriz por columnas) para que dita suma sexa maior que a suma dos elementos do vector **x**. Garda no arquivo `resultados5.txt` a matriz **a** e o valor de k.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
from sys import *
n=0
while n< 10:
    n=int(input('n= '))
x=zeros(n)
x[0]=1; x[1]=1
for i in range(2,n):
    x[i]=x[i-1]+x[i-2]
print "x= ", x
def transforma(v):
    n=len(v)
    m=int(ceil((1+sqrt(1+8*n))/2))
    print "m= ", m
    b=diag(v[0:m])
    k=0
    for i in range(m):
        for j in range(i+1, m):
            b[i,j]=v[k]
            b[j,i]=v[k]
            k=k+1
            if k==n:
                break
        if k==n:
            break
    return b
a=transforma(x)
clf(); imshow(a)
title('Mapa de calor')
```

```
xlabel('Eixo x'); ylabel('Eixo y')
savefig('figura5.png')
show(False)

sx=sum(x)
y=a.flatten('F')
suma=y[0]
k=1
while suma<sx:
    suma = suma + y[k]
    k=k+1
try:
    savetxt('resultados5.txt', a, '%6d')
    fin=open('resultados5.txt', 'a')
    fin.write('k= %d\n' % (k))
    fin.close()
except IOError:
    exit('Erro escribindo resultados5.txt')
```

Segundo control de programación en Python de 2017

Crea un arquivo `datos6.txt` co editor de texto con números enteiros e liñas con distinto número de elementos. Por exemplo, co seguinte contido:

```
11 12 6
2 8
7 9 4 5
3
```

Escribe un programa en Python chamado `control6.py` que realice o seguinte:

1. Lea todos os elementos do arquivo `datos6.txt` ó vector **x** e chame á función `funcion6(...)` definida por ti e dalle os argumentos axeitados.

2. Esta función debe crear, a partir de **x**, un vector **y** (inicialmente baleiro) que engada os elementos múltiplos de 3 de **x** mentres a suma dos elementos de **y** non sexa superior a 50 (se non hai elementos suficientes en **x** para acadar este valor, volver a comezar polo principio tantas veces como sexa necesario). A función debe retornar o vector **y** creado.

3. O programa principal debe crear unha matriz **a** de dimensión n, sendo n o número de elementos de **y**, de modo que a_{ij} sexa a suma dos elementos de **x** superiores ou iguales, simultáneamente, a y_i e y_j.

4. Suma os elementos pares e impares da matriz **a** nas variables *sp* e *si* respectivamente. Se $si > sp$, pídelle ó usuario o nome dun arquivo para almacenar nel o vector **y** (nunha liña) e a matriz **a** (cada fila nunha liña) como números enteiros en columnas de 5 espacios de ancho. En caso contrario, realiza un histograma con todos os elementos da matriz **a**, poñendo títulos ó gráfico e os eixos.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from sys import exit
from matplotlib.pyplot import *
try:
    f=open('datos6.txt','r')
    x=[]
    for linha in f:
        aux=linha.rsplit()
        for i in aux:
            x.append(int(i))
    f.close()
except IOError:
    exit('Erro lendo datos6.txt')

def funcion6(v):
    n=len(v)
    w=[]; i=0
    while sum(w) < 50:
        if v[i]%3 == 0:
            w.append(v[i])
        i= i+1
        if i == n:
            i=0
    return w
y=funcion6(x)
m=len(y); n=len(x)
```

```
a=zeros([m,m])
x=array(x)
for i in range(m):
    for j in range(m):
        a[i,j]=sum(extract(x >= max(y[i],y[j]), x))
        # version non vectorizada
        #suma=0
        #for k in range(n):
            #if (x[k] >= y[i]) and (x[k] >= y[j]):
                #suma = suma + x[k]
        #a[i,j]=suma

z=a.flatten()
pares=where( z%2 == 0)[0]
impares=where(z%2 == 1)[0]
sp=sum(z[pares])
si=sum(z[impares])
if sp > si:
    nome=raw_input('Nome arquivo: ')
    savetxt(nome, vstack([y,a]), '%5d')
else:
    figure(1); clf(); hist(z)
    title('Histograma de a')
    xlabel('Valores en a'); ylabel('Numero elementos')
    show(False)
```

Segundo control de programación en Python de 2017

Crea co editor de texto un arquivo, chamado `datos7.txt`, cun contido do tipo:

```
3
1.5 2 1
0 -2.3 0.1
3.2 1 2
2
3.7
1
```

Escribe un programa en Python chamado `exame7.py` que realice:

1. Lea a primeira liña do arquivo `datos7.txt` a variable n e crea unha matriz **a** cadrada de orde n. Lea as n seguintes liñas e almacénaas nas n filas da matriz **a**. Lee as últimas n liñas e almacénaas nun vector **x**.

2. Calcula unha matriz **b** de orde n, onde cada elemento b_{ij} defínese como:

$$b_{ij} = \sum_{k=0}^{n-1} a_{ik} + \sum_{k=0}^{n-1} a_{kj} \qquad i, j = 0, 1, \ldots, n-1$$

3. O programa debe chamar a función `funcion7(...)`, cos argumentos axeitados e definida por ti, que devolva dúas matrices cadradas **c** e **d** de orde n. A matriz **c** debe ser inicialmente igual a **a**, e logo debes sumarlle a matriz **b** ate que

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_{ij} > 100$$

   Pola outra banda, cada elemento d_{ij} da matriz **d** debe calcularse como:

$$d_{ij} = \frac{x_j + x_k}{2} \qquad i, j = 0, 1, \ldots, n-1$$

   onde $k = (i + j)\,\%n$, e $x\,\%y$ é o resto da división enteira de x entre y.

4. No programa principal, calcula os vectores **y** e **z** resultantes de convertir as matrices **c** (por filas) e **d** (por columnas) en vectores respectivamente. Ordea de menor a maior o vector **y** e realiza unha figura con dúas gráficas que conteñan:

   - Axusta os valores (y_i, z_i), $i = 0, \ldots, n-1$ a un polinomio de orde 3 e representa os puntos reais e a curva do polinomio na primeira gráfica. Pon enreixado, lendas e título.
   - Estima o valor da interpolación lineal en 50 puntos equiespaciados entre o valor mínimo e máximo do vector **y**. Representa na segunda gráfica os valores reais en azul e os interpolados en verde. Pon enreixado, lendas e título.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from sys import exit
from matplotlib.pyplot import *
try:
    f=open('datos7.txt','r')
    n=int(f.readline())
    a=zeros([n,n])
    for i in range(n):
```

```
        a[i]=float_(f.readline().rsplit())
    x=zeros(n)
    for i in range(n):
        x[i]=float(f.readline())
    f.close()
except IOError:
    exit('Erro lendo datos7.txt')

b=zeros([n,n])
for i in range(n):
    for j in range(n):
        b[i,j]=sum(a[i,:])+sum(a[:,j])

def funcion7(a,v):
    c=a.copy()
    while sum(c) < 100:
        c = c + b
    n=len(v)
    d=zeros([n,n])
    for i in range(n):
        for j in range(n):
            k=(i+j)%n
            d[i,j]=(v[i]+v[k])/2
    return [c, d]

[c, d]=funcion7(a,x)
y=sort(c.flatten())
z=d.flatten('F')
yy=linspace(min(y), max(y), 50)
p=polyfit(y, z, 3)
figure(1); clf()
subplot(211)
plot(y, z, 'b*', label=u'Ptos reais')
plot(yy, polyval(p, yy), 'g-', label=u'Pol. orde 3')
grid(True); legend(loc='upper right')
title('Axuste a un polinomio grao 3')
subplot(212)
zz=interp(yy, y, z)
plot(y, z, 'b*', label=u'Ptos reais')
plot(yy, zz, 'gs', label=u'Ptos interpolado')
grid(True); legend(loc='upper right')
title('Interpolacion lineal')
show(False)
```