# Segundo control de programación en Python de 2018

---

Escribe un arquivo de texto con números enteiros, onde todas as liñas teñen o mesmo número de elementos. Por exemplo co seguinte contido:

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

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

1. Ler o arquivo a unha matriz `a`, comprobando que non hai erro na súa lectura. Sexa $n$ o número de filas e $m$ o número de columnas de `a`.

2. Define a función `transformaMatriz()`, cos argumentos axeitados, que transforme a matriz `a` nunha matriz `b`, da mesma dimension que `a`, cos elementos $b_{ij}$, $i = 0, \ldots, n-1$ e $j = 0, \ldots, m-1$ definidos por:

$$b_{ij} = \sum_{k=0}^{l-1} a_{ik} a_{kj} \tag{1}$$

   sendo $l = minimo(n, m)$.

3. O programa debe chamar a función `transformaMatriz()` para calcular a matriz `b` e visualizala na pantalla. Sexa $\mu$ o valor medio dos elementos da matriz `b`. O programa debe calcular os vectores **x**, **y** e **z** seguintes a partir de **b**:

   - O vector **x** conterá os elementos de **b** maiores que $\mu$.
   - O vector **y** será os elementos de **b** menores que $\mu$.
   - O vector **z** terá os elementos de **b** que hai que sumar (por columnas) para superar a metade da suma da matriz `b`.

4. Gardar no arquivo `saida1.txt` os vectores **x**, **y** e **z**, un en cada liña do arquivo.

5. Representa nun gráfico unha liña vermella co valor $\mu$ e, se o número de elementos de **x** e maior que os de **y**, o vector **x** (con marcas azuis), en caso contrario representa o vector **y** con marcas verdes.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
from sys import *
nome=raw_input('Nome do arquivo: ')
try:
    a=loadtxt(nome)
except IOError:
    print "Erro abrindo ", nomeArquivo
    exit()
def transformaMatriz(d):
    [nf, nc]=d.shape
    c=zeros([nf, nc])
    l=min(nf, nc)
    for i in range(nf):
        for j in range(nc):
            c[i,j]=dot(d[i,0:l], d[0:l, j])
    return c
b=transformaMatriz(a)
```

```
print 'b= ', b
mu=mean(mean(b))
print 'Valor medio= ', mu
x=extract(b > mu, b)
y=extract(b < mu, b)
bp=b.flatten('F')
sb=sum(sum(b))/2
suma=0
z=[]; k=0
while suma < sb:
    suma = suma + bp[k]
    k=k+1
    z.append(bp[k])
del z[-1]
def escribeVector(fid, x):
    for i in range(len(x)):
        fid.write('%5d' % (x[i]))
    fid.write('\n')
    return
try:
    f=open('saida1.txt', 'w')
    escribeVector(f, x)
    escribeVector(f, y)
    escribeVector(f, z)
    f.close()
except IOError:
    print 'Erro escribindo no arquivo saida1.txt'
clf()
nx=len(x); ny=len(y)
if nx > ny:
    plot(range(nx), x, 'b*')
    plot([0, nx], [mu, mu], 'r-')
else:
    plot(range(ny), y, 'g*')
    plot([0,ny], [mu, mu], 'r-')
show(False)
```

# Segundo control de programación en Python de 2018

---

Escribe un programa en Python chamado `exame2.py` que pida ó usuario unha cadea de caracteres coa expresión analítica dun polinomio `f(x)` e realice as seguintes operacións:

1. Deriva a función `f(x)` simbólicamente. Xera un vector **y** con $n = 50$ elementos equidistantes no intervalo [0,20] e calcula dous vectores **v** e **w** (ambos de dimensión $n$) que conteñan, respectivamente, o valor da función `f(x)` e da súa derivada nos puntos do vector **y**.

2. Representa nun gráfico os vectores **v** e **w** frente ó vector **y**, indicando con liñas de distinta cor cada vector. Pon título, etiquetas nos eixos e lendas no gráfico. Garda o gráfico no arquivo `figura2.png`.

3. Define unha función chamada `calculaMatriz()`, cos argumentos axeitados, que calula unha matriz cadrada `b` de orde $n$ a partir de dous vectores **v** e **w** de dimensión $n$. Cada elemento $b_{ij}$ ven dado pola expresión:

$$b_{ij} = \begin{cases} \displaystyle\sum_{k=0}^{i} v_k w_k & \text{se } i < j \\ \displaystyle\sum_{k=0}^{j} \frac{v_k + w_k}{j+1} & \text{se } j < i \\ \displaystyle\sum_{k=0}^{i} v_j w_k & \text{se } j = i \end{cases} \tag{2}$$

   con $i = 0, \ldots, n-1$ e $j = 0, \ldots, n-1$.

4. Desde o programa principal chama a función `calculaMatriz()` para calcular a matriz `b` e garda no arquivo `saida2.txt` os vectores **y**, **v**, **w** (un vector en cada liña utilizando reais con dous decimais), e a matriz `b` (unha fila en cada liña e utilizando tamén dous decimais).

5. Calcula a suma de todos os elementos da matriz **b** na variable *sb* e calcula o número de elementos do triángulo superior (sen a diagonal principal) que hai que sumar para que se supere o valor $sb/4$. En caso de que non se supere ese valor, suma todos os elementos do triángulo. Visualiza no terminal o número de elementos e os elementos sumados.

**NOTA:** Para evitar interferencias entre funcións que teñen o mesmo nome en distintos módulos utiliza un alias ( por exemplo `import numpy as np`). Podes probar coas funcións `x**2` ou `x**3-5*x**2`.

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


from sympy import *
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as npr

x=symbols('x')
expresion=raw_input('f(x)= ')
df=diff(expresion, x)
f=sympify(expresion)
n=50
y=np.linspace(0,20, n)
ff=eval('lambda x: '+ expresion)
dff=lambdify(x, df)
v=ff(y)
w=dff(y)
# alternativa
```

```
#v=np.zeros(n); w=np.zeros(n)
#for i in y:
    #v[i]=f.subs(x, i)
    #w[i]=df.subs(x, i)
plt.clf()
plt.plot(y, v, 'b*-', label=u'f(x)')
plt.plot(y, w, 'go--', label=u'df(x)')
plt.legend(loc='best'); plt.title('f(x) e df(x)'); plt.xlabel('x')
plt.show(False); plt.savefig('figura2.png')

def calculaMatriz(v,w):
    n=len(v)
    a=zeros(n,n)
    for i in range(n):
        for j in range(n):
            if i < j:
                a[i,j]=sum(v[0:i+1]*w[0:i+1])
            elif j < i:
                a[i,j]=np.mean(v[0:j+1]+w[0:j+1])
            else:
                a[i,j]=v[j]*sum(w[0:i+1])
    return a
b=calculaMatriz(v,w)
a=np.vstack([y, v, w, b])
np.savetxt('saida2.txt', a, fmt='%8.2f')
sb=sum(b)/4
s=0
elem=[]
for i in range(n):
    for j in range(i+1, n):
        s=s+b[i,j]
        elem.append(b[i,j])
        if s > sb:
            break
    if s > sb:
        break
print 'Elementos sumados: ', elem
print 'No. elementos: ', len(elem), ' e suman ', sum(elem)
print 'sb= ', sb
```

# Segundo control de programación en Python de 2018

---

Escribe un programa en Python chamado `exame3.py` que realice as seguintes operacións:

1. Defina unha función chamada `cifras()` que descompoña un número enteiro $x$ nas súas cifras. Dado un número enteiro, podes obter as súas cifras tomando os restos das divisións sucesivas do número por 10. Por exemplo, se $x = 4567$, a función `cifras()` debería devolver 7, 6, 5 e 4.

2. Xere un número enteiro aleatorio $x$ no intervalo $[10^5, 10^8]$.

3. Pida ó usuario números enteiros positivos de dúas cifras mentres que as súas cifras non estean no número $x$. Se o número non fose válido, volve a pedir outro número ó usuario. Almacena todos os números introducidos polo usuario no vector $\mathbf{y}$.

4. Sexa $\mathbf{v}$ un vector coas cifras de $x$, calcula o vector $\mathbf{z}$ e a matriz `b` seguintes:

   - Un vector $\mathbf{z}$ concatenando (xuntando horizontalmente) 10 veces o vector $\mathbf{v}$.
   - Unha matriz cadrada `b` de orde $n$ (sendo $n$ o número de elementos do $\mathbf{z}$), onde cada elemento $b_{ij}$ ven dado por:

$$b_{ij} = \sum_{k=0}^{l} z_k z_{n-1-k} \quad l = \text{maximo}(i, j), \quad i, j = 1, \ldots, n \tag{3}$$

5. Fai un mapa de calor da matriz `b`.

6. Garda nun arquivo chamado `saida3.txt` o vector $\mathbf{v}$ nunha única liña e o vector $\mathbf{y}$ noutra liña.

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

from matplotlib.pyplot import *
from numpy import *
from random import randint
def cifras(x):
    c=[]
    if x < 10:
        c.append(x)
    else:
        c.append(x %10)
        x= x/10
        while x > 9:
            c.append(x%10)
            x=x/10
        c.append(x)
    return c
x=randint(1e4,1e8)
v=cifras(x)
print "x= ", x, ' cifras= ', v
y=[]
while True:
    n=abs(int(raw_input('Numero enteiro de dos cifras: ')))
    y.append(n)
    if (n > 9) & (n < 100):
        cn=cifras(n)
        print 'n= ', n, ' cifras: ', cn
        if (cn[0] in v) & (cn[1] in v):
```

```
            break
print 'y= ', y
z=v[:]
for i in range(9):
    z=concatenate((z,v))
n=len(z)
b=zeros([n,n])
for i in range(n):
    for j in range(n):
        l=max(i,j)
        b[i,j]=dot(z[0:l+1], z[n-l-1:n])
clf(); imshow(b); show(False)
try:
    f=open('saida3.txt', 'w')
    for i in range(len(v)):
        f.write('%5d' % (v[i]))
    f.write('\n')
    for i in range(len(y)):
        f.write('%5d' % (y[i]))
    f.write('\n')
    f.close()
except IOError:
    print 'Erro escribindo no arquivo saida3.txt'
```

# Segundo control de programación en Python de 2018

---

Escribe un programa en Python chamado `exame4.py` que realice as seguintes operacións:

1. Pide ó usuario a expresión dunha sucesión númerica $x_n$ (podes lela como unha cadea de caracteres) e calcule simbolicamente a suma da serie
$$\sum_{n=1}^{\infty} x_n \tag{4}$$
asignando a variable $s$ o valor exacto da suma con un númerico real en punto flotante. Supón que a serie é converxente (por examplo $x_n$=`1/n**2` ou $x_n$=`n**2/2**n`).

2. Pide ó usuario un número enteiro $m$ no intervalo [30,50] (debes volver a pedilo ate que $m$ sexa un valor aceptabel).

3. Suma os $m$ primeiros termos da serie e calcula a diferencia $ds$ entre a suma exacta da serie $s$ e o valor aproximado coa suma dos $m$ termos. Crea unha matriz cadrada aleatoria `a` de orde $m$ con números enteiros aleatorios no intervalo [0, m].

4. Se a diferencia $ds$ é maior que $1e-5$, visualiza a matriz `a` como nun mapa de calor. Noutro caso, garda no arquivo `saida4.txt` a matriz `a` como números enteiros, escribindo unha fila en cada liña.

5. Define unha función `calcula()`, cos argumentos axeitados, que, a partir dunha matriz `a` de orde $m$ calcule a matriz `b` e un vector `x`. Cada elemento $b_{ij}$, $i,j = 0,\ldots,m-1$ ven dado por:
$$b_{ij} = \frac{a_{1i}}{2a_{2j}+1} \tag{5}$$
Os elementos do vector son:
$$x_i = \sum_{j=0}^{m-1} a_{ij} \quad i = 0,\ldots,m-1 \tag{6}$$

6. Desde o programa principal chama a función `calcula()` e visualiza a matriz `b` e o vector `x` na pantalla .

**NOTA:** Para evitar interferencias entre funcións que teñen o mesmo nome en distintos módulos utiliza un alias ( por exemplo `import numpy as np`).

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

import numpy as np
from matplotlib.pyplot import *
from sympy import *
import  numpy.random as nr

n=symbols('n')
sucesion=raw_input('xn= ')
xn=sympify(sucesion)
f=lambdify(n, xn)
s=Sum(xn, (n, 1, oo)).evalf()
m=0
while m<30 or m>50:
    m=int(raw_input('m= '));
sumandos=np.zeros(m)
for i in range(m):
    sumandos[i]=f(i+1)
sa=sum(sumandos)
```

```
di=s-sa
print 'Suma exacta= %.3g, suma aproximada= %.3g e diferencia= %.3g\n' % (s, sa, di)

a=np.round(m*nr.rand(m,m))
if di > 1e-5:
    imshow(a); title('Mapa de calor')
    show(False)
else:
    np.savetxt('saida4.txt', a, '%d')

def calculos(a):
    [nf,nc]=a.shape
    b=np.zeros([nf,nc])
    x=np.zeros(nf)
    for i in range(nf):
        for j in range(nc):
            b[i,j]=a[1,i]/(2*a[2,j]+1)
        x[i]=sum(a[i])
    return [b, x]

[b, x]=calculos(a)
print 'b= ', b
print 'x= ', x
```

# Segundo control de programación en Python de 2018

---

Escribe un arquivo de texto con números enteiros onde cada liña pode ter distinto número de elementos. Por exemplo co seguinte contido:

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

Escribe un programa chamado `exame5.py` que lle pida ó usuario o nome do arquivo de texto e realice as seguintes operacións:

1. Ler o contido do arquivo almacenando os números pares nun vector **x** e os números impares nun vector **y**.

2. Definir a función `funcionExame()`, pasándolle os argumentos axeitados, que mostre por pantalla $sx$ (suma dos elementos do vector **x**) e $sy$ (suma dos elementos do vector **y**), e lea por teclado números enteiros repetidamente. Se $sx > sy$, o número lido debe engadirse a **y**; en caso contrario, debe engadirse a **x**. Con cada valor engadido, debes actualizar o valor da suma e rematar o proceso de lectura de números cando $sx = sy$. A función debe devolver os vectores **x** e **y**.

3. Chamar a función `funcionExame()` desde o programa principal e visualiza na pantalla os vectores **x** e **y**. Crea unha matriz **a** de orde $n$, sendo $n$ o mínimo dos tamaños dos vectores **x** e **y**, de modo que se $i \geq j$, entón $a_{ij}$ debe ser $j$ mais o número de elementos de **y** superiores a $x_i$; se, polo contrario, $i < j$, entón $a_{ij}$ debe ser $i$ mais o número de elementos de **x** superiores a $y_i$.

4. Comprobar se os vectores **x** e **y** son de distinta lonxitude, e en tal caso, rechea o vector de menor dimensión co valor 5 ate que **x** e **y** teñan o mesmo número de elementos. Representa graficamente os valores dos vectores **x** (con marcas azuis) e **y** (con marcas verdes) frente á posición no vector. Representa no mesmo gráfico (con marcas vermellas) os valores de **x** que son maiores ou iguais ós correspondentes (na mesma posición) en **y**. Pon lendas e títulos ó gráfico.

```python
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
from sys import *

nome=raw_input('Nome arquivo: ')
x=[]; y=[]
try:
    fid=open(nome, 'r')
    for linha in fid:
        aux=linha.rsplit()
        n=len(aux)
        for i in range(n):
            m=int(aux[i])
            if m%2 == 0:
                x.append(m)
            else:
                y.append(m)
    fid.close()
except IOError:
    print "erro: o arquivo", nome
    exit()
```

```
def funcionExame(x, y):
    sx=sum(x); sy=sum(y)
    while sx != sy:
        print 'sx= %d, sy=%d\n' % (sx, sy)
        n=int(raw_input('n= '))
        if sx > sy:
            y.append(n)
            sy = sy + n
        else:
            x.append(n)
            sx=sx + n
    return [x, y]
[x, y]=funcionExame(x,y)
print 'x= ', x
print 'y= ', y
n=min(len(x), len(y))
x=array(x); y=array(y)
a=zeros([n,n])
for i in range(n):
    for j in range(n):
        if i >= j:
            a[i,j]=j+sum(y > x[i])
        else:
            a[i,j]=i+sum(x > y[i])
print 'a= ', a
nx=len(x); ny=len(y)
if nx < ny:
    for i in range(ny-nx):
        x=append(x, 5)
elif ny < nx:
    for i in range(nx-ny):
        y=append(y, 5)
clf()
n=range(len(x))
plot(n, x, 'b*', label=u'Vector x')
plot(n, y, 'g*', label=u'Vector y')
ind=where(x >= y)[0]
plot(ind, x[ind], 'ro', label=u'x>y')
xlabel('Indice'); ylabel('Valores')
grid(True); show(False)
```