Segundo control de programación en Python de 2019

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

```
1 2 3 4 5 6 7 8 9
1.7 9.4 19.8 35.3 54.3 78.9 108.4 139.8 178
```

Escribe un programa chamado `exame1.py` que pida por teclado o nome do arquivo e lea a primeira liña ó vector **x** e a segunda liña ó vector **y**, comprobando que non hai erros na lectura. Despois realice as seguintes operacións:

1. Sexa n o número de elementos dos vectores **x** e **y**. Calcula a ecuación da función $y = ax^b$ que mellor axusta os puntos $(x_i, y_i), i = 1, \ldots, n$. Visualiza a expresión da función. Representa nun gráfico bidimensional os puntos $(x_i, y_i), i = 1, \ldots, n$ con asteriscos azuis e a curva $y = ax^b$ con unha liña vermella. Pon títulos e lendas ó gráfico.

2. Crea un vector z con n elementos, onde cada z_i ven dado pola expresión:

$$z_i = \begin{cases} \cos \dfrac{\pi i}{4} & \text{se i par} \\ z_{i-1} - x_i & \text{resto dos casos} \end{cases} \tag{1}$$

3. Define unha función chamada `transforma()`, cos argumentos axeitados, que calcule a matriz **a** de orde n, onde cada elemento de $a_{ij} = x_i y_j$, e retorne un vector **v** co triángulo inferior da matriz **a** sen a diagonal principal e outro vector **w** co triangulo superior da matriz tamén sen a diagonal principal. Chama a función para que actúe sobre os vectores **y** e **z**, e visualiza os vectores retornados.

```
from numpy import *
from sys import *
from matplotlib.pyplot import *
nome=input('Nome do arquivo: ')
try:
    b=loadtxt(nome)
    [nf, n]=b.shape
    if nf==2:
        x=b[0]
        y=b[1]
    else:
        raise
except IOError:
    print('Erro abrindo arquivo %s\n' % nome)
    exit()
except:
    print('O arquivo ten %i linhas\n' % nf)
    exit()

p=polyfit(log(x), log(y), 1)
a=exp(p[1])
print('x= ', x); print('y= ', y)
print('y= %6.2f x ** %6.2f\n' %( a, p[0]))
xr=linspace(min(x), max(x), 50)
yr=a * xr**p[0]
clf(); plot(x, y, 'b*', label='Puntos')
plot(xr, yr, 'r-', label='y=ax**b')
title('Axuste'); xlabel('x'); ylabel('y'); legend(); show(False)
z=zeros(n)
```

```
for i in range(n):
    if i%2:
        z[i]=z[i-1]*x[i]
    else:
        z[i]=cos(pi*i/4)
print('z= ', z)
def transforma(y, z):
    n=len(y)
    a=y.reshape([n,1])*z.reshape([1,n])
    print(a)
    vv=[]; ww=[]
    for i in range(n):
        for j in range(n):
            if i > j:
                ww.append(a[i,j])
            elif j > i:
                vv.append(a[i,j])
    return [vv, ww]
[v, w]=transforma(y, z)
print('v= ', v)
print('w= ', w)
```

Segundo control de programación en Python de 2019

Escribe un programa chamado `exame2.py` que pida por teclado un número enteiro m maior que 100 comprobando que o número sexa correcto e realice as seguintes operacións:

1. Calcule catro vectores **x**, **y**, **z** e **v** co seguinte contido: **x** contén os números que son múltiplos de 3 e conteñen a cifra 3 no intervalo $[0, m]$, **y** contén os números do intervalo $[0, m]$ que conteñen a cifra 3 pero que non son multiplos de 3, **z** contén os números que son múltiplos de 3 pero non teñen o número 3 entre as súas cifras no intervalo $[0, m]$, e **v** contén o resto dos números nese intervalo. Podes comprobar se `i` ten a cifra 3 con `str(i).count('3')>0`.

2. Calcula o número de veces que hai que sumar o vector **x** para que a suma sexa maior que a suma dos elementos do vector **v**.

3. Pide por teclado o nome dun arquivo. Define a función `gardaVector()`, con argumentos axeitados, que garde un vector nunha liña ó final do arquivo introducido por teclado. Se no proceso de escritura non houbo erros devolve un 1 e, en caso contrario, devolve un 0. Chama a función `gardaVector()` catro veces para gardar o contido dos vectores **x**, **y**, **z** e **v** no arquivo.

4. Representa nun gráfico de liñas os vectores **x**, **y**, **z** e **v** (no eixo y) e no eixo x un vector de números enteiros no intervalo $[0, m]$. Pon en distintos cores os puntos de cada vector. Pon enreixado, lendas e etiquetas nos eixos. Garda a figura no arquivo `exame2.png`.

```
from numpy import *
from sys import *
from matplotlib.pyplot import *

m=1
while m<100:
    m=int(input('m= '))
def cifras(n):
    x=[]
    while n>9:
        x.append(mod(n,10))
        n=n/10
    return x
x=[]; y=[]; z=[]; v=[]
for i in range(m+1):
    c=cifras(i)
    if mod(i, 3)==0 and 3 in c:
        x.append(i)
    elif 3 in c:
        y.append(i)
    elif mod(i,3)==0:
        z.append(i)
    else:
        v.append(i)
#-- solución alternativa
#for i in range(m):
        #c=str(i)
        #if i%3==0:
                #if c.count('3')>0:
                        #x.append(i)
                #else:
                        #z.append(i)
        #else:
                #if c.count('3')>0:
```

```
                        #y.append(i)
                #else:
                        #v.append(i)
print('x= ', x); print('y= ', y)
print('z= ', z); print('v= ', v)
nveces=1; sumax=sum(x); sumav=sum(v)
while sumax < sumav:
    sumax = sumax + sumax
    nveces += 1
print('No. veces: ', nveces)
nome=input('Nome do arquivo: ')
def gardaVector(nome, x):
    try:
        f=open(nome, 'a')
        cad=''
        for i in x:
            cad= cad + '%s '% str(i)
        cad= cad + '\n'
        f.write(cad)
        f.close()
        return 1
    except:
        return 0
vect=[x, y, z, v]
for w in vect:
    if gardaVector(nome, w):
        print('Gardou vector correctamente')
w=range(m+1)
figure(1); clf()
plot(x,x, 'b*', label='vector x')
plot(y,y, 'go', label='vector y')
plot(z, z, 'rv', label='vector z')
plot(v, v, 'm*', label='vector v')
legend(); grid(True); xlabel('Eixo x')
ylabel('Eixo y'); show(False)
savefig('exame2.png')
```

Segundo control de programación en Python de 2019

Escribe un arquivo de texto libre con palabras e números. Por exemplo co seguinte contido:

```
A casa de Iria ten 2 portas,
8 xanelas e 5 habitacions. No tellado pousaron 7 aves.
```

Escribe un programa chamado `exame3.py` que pida por teclado o nome do arquivo e realice as seguintes operacións:

1. Define unha función chamada `analiza()`, cos argumentos axeitados, que conte o número de palabras e meta nunha lista os números que aparecen nun arquivo (**como números, non como cadeas**). Se hai erros lendo o arquivo, devolve unha lista de números baleira e o número de palabras sería 0.

2. Aplica a función `analiza()` ó arquivo introducido por teclado, almacenando no vector $\mathbf{x}$ os números e na variable p o número de palabras. Sea n o número de elementos de $\mathbf{x}$.

3. Lee un número real a por teclado no intervalo $[2, 5]$, verificando que o valor é correcto. Calcula un vector $\mathbf{y}$ de dimension n, tal que $y_i = ax_i^2$, $i = 0, \ldots, n-1$. Suma a cada elemento do vector $\mathbf{y}$ un número real aleatorio no intervalo $[-10, 10]$. Visualiza na pantalla o vector $\mathbf{y}$ antes e despois de sumarlle os números aleatorios.

4. Axusta os datos (x_i, y_i), $i = 0, \ldots, n-1$ a unha función exponencial do tipo $y = bx^c$. Representa gráficamente o vector de puntos $\mathbf{y}$ frente a $\mathbf{x}$ e a curva resultante do axuste. Pon título, lendas e etiquetas nos eixos ó gráfico.

5. Conta o número de veces que hai que dividir o vector $\mathbf{y}$ por 10 para que a suma suma sexa inferior a suma do vector $\mathbf{x}$

```
from numpy import *
from matplotlib.pyplot import *
from random import *
nome=input('Nome do arquivo: ')
def analiza(nome):
    y=[]; n=0
    try:
        f=open(nome, 'r')
        for linha in f:
            x=linha.rsplit()
            for yi in x:
                if yi.isdigit():
                    y.append(float(yi))
                else:
                    n +=1
        f.close()
        return [y, n]
    except:
        return [[], 0]
[x, p]=analiza(nome)
print('x= ', x)
print('O arquivo %s ten %d palabras' % (nome, p))
a=0
while a<=2 or a>=5:
    a=float(input('a= '))
y=a*array(x)**2; n=len(x)
print('y= ', y)
for i in range(n):
    y[i]= y[i] + 20*random()-10
```

```
print('y= ', y)
p=polyfit(log(x), log(y),1)
xr=linspace(min(x), max(x), 50)
yr=exp(p[1])*xr**p[0]
clf(); plot(x, y, 'b*', label='Puntos')
plot(xr, yr, 'g-', label='Axuste')
grid(True); title('Axuste');
xlabel('x'); ylabel('y'); show(False)
sx=sum(x); nc=0
while sum(y) > sx:
    nc = nc +1
    y=y/10
print('No. veces: %i\n' % nc)
```

Segundo control de programación en Python de 2019

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

1. Pide por teclado dous números enteiros m e n no intervalo $[20, 30]$, non deixando avanzar o programa ate que os números verifiquen a condición.

2. Crea unha matriz **a** de orde $n \times m$ onde $a_{ii} = \dfrac{2^i}{i+1}$, con $i = 1, \ldots, \text{mín}(n, m)$. Os elementos do triángulo inferior serán números aleatorios no intervalo $[\text{mín}(n, m), \text{máx}(n, m)]$, e os elementos do triangulo superior calcúlanse coa expresión: $a_{ij} = \dfrac{a_{i(j-1)}}{a_{ii}}$, con $i = 1, \ldots, \text{mín}(n, m) - 1$ e $j = i + 1, \ldots, m - 1$.

3. Garda a matriz a no arquivo `saida4.txt` utilizando reais con un ancho de 10 e 2 cifras decimais.

4. Crea unha gráfica cunha superficie en 3D na que a matriz **a** representa as alturas e os eixos x e y son os índices de acceso ós elementos da matriz. Pista: para evitar erros, intercambia os rangos de x e y na función `meshgrid()`.

5. Convirte a matriz **a** no vector **x** percorrendo a matriz por columnas. Atopa o valor de k tal que:

$$\sum_{i=0}^{k} x_i > \sum_{i=k+1}^{l-1} x_i, \qquad \text{con } l = nm \tag{2}$$

6. Engade ó final do arquivo `saida4.txt` o valor de k.

```
from numpy import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
import random
n=0; m=0
while n<20 or n>30:
    n=int(input('n= '))
while m<20 or m>30:
    m=int(input('m= '))
a=zeros([n,m])
nmin=min(n,m)
nmax=max(n,m)
rango=nmax-nmin
#************************************************
for i in range(nmin):
    a[i,i]=2**i/(i+1)
for i in range(1,n):
    fin=min(i,m)
    for j in range(fin):
        a[i,j]=random.random()*rango+nmin
        # a[i,j]=random.uniform(nmin, nmax)
for i in range(1, nmin):
    for j in range(i+1, m):
        a[i,j]= a[i, j-1]/a[i,i]
#************************************************
# alternativa
for i in range(n):
    for j in range(m):
        if i == j:
            a[i,j]=2**i/(i+1)
```

```
        elif i > j:
            a[i,j]=random.uniform(nmin,nmax)
        else:
            a[i,j]=a[i,j-1]/a[i,i]
#************************************************
print('a= '); print(a)
savetxt('saida4.txt', a, '%10.2g')
fig=figure(); clf(); ax=Axes3D(fig)
x=arange(n); y=arange(m)
X,Y=meshgrid(y,x); ax.plot_surface(X, Y,a)
title('Superficie en 3D'); show(False)
x=a.flatten('F')
s1=0; s2=sum(x); k=0
while s1<s2:
    s1=s1+x[k]
    s2=s2-x[k]
    k=k+1
print('k=%i' % k)
try:
    f=open('saida4.txt', 'a')
    cad='k= '+str(k)+ '\n'
    f.write(cad)
    f.close()
except IOError:
    print('Erro escribindo no arquivo saida4.txt')
```

Segundo control de programación en Python de 2019

Escribe co editor de texto un arquivo chamdado `datos5.txt` con números enteiros (o mesmo número de elementos en cada liña). Por exemplo co seguinte contido:

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

Escribe un programa chamado `exame5.py` que realice as seguintes operacións:

1. Lea o arquivo `datos5.txt` e almacene o datos en dúas matrices cadradas. Se non puidesen ser cadradas, remata o programa. Visualiza as matrices na pantalla.

2. Define a función `calcula()`, cos argumentos axeitados, que a partir de dúas matrices cadradas **a** e **b** de orde n, calcule unha matriz **c** de orde n, onde os elementos c_{ij}, $i, j = 0, \ldots, n-1$ calcúlanse do seguinte modo: para calcular c_{ij}, suma os elementos da fila i de **a** e a columna j de **b** (un elemento de cada unha en cada iteración) rematando cando a suma supere o producto $a_{ij}b_{ij}$ ou se acade o final de fila i e columna j, sendo c_{ij} o número de iteracións. Aplica a función `calcula()` ás matrices do apartado anterior.

3. Se a suma dos elementos pares da matriz **a** e maior que a suma dos elementos impares de **b** realiza o seguinte: 1) convirte a matriz **a** no vector **x** percorrendo **a** por filas; 2) convirte a matriz **b** no vector **y** percorrendo **b** por columnas; 3) representa nun gráfico os valores do vector **x** (en azul) e do vector **y** (en verde) frente ó índice de cada elemento; 4) pon en vermello os puntos do vector **x** que son superiores ós do vector **y** na mesma posición. No caso de que non se cumpra a condición de que a suma dos elementos pares da matriz **a** e maior que a suma dos elementos impares de **b** garda no arquivo `resultados5.txt` a seguinte información: na primeira fila os elementos da diagonal secundaria da matriz **c**; na segunda liña os elementos da matriz **a** que son múltiplos de 4; e na terceira liña o determinate das matrices **a**, **b** e **c**.

```
from numpy import *
from numpy.linalg import *
from matplotlib.pyplot import *
from sys import *

try:
    aux=loadtxt('datos5.txt')
    [nf, nc]=aux.shape
    resto= nf%2
    if nf != 2*nc:
        print('Filas = %d e columnas=%d . Arquivo non adecuado\n'% (nf, nc))
        exit()
except IOError:
    print('Erro lendo arquivo datos5.txt')
    exit()

a=aux[:nc, :]; print('a= ', a)
b=aux[nc:, :]; print('b= ', b)
def calcula(a, b):
    n=size(a,1)
    d=zeros([n, n], int)
    for i in range(n):
        for j in range(n):
```

```
            s=0; p=a[i,j]*b[i,j]
            for k in range(n):
                s=s + a[i,k] + b[k,j]
                if s > p:
                    break
            d[i,j]=k+1
    return d
c=calcula(a,b)
spa=sum(extract(a%2==0, a))
sib=sum(extract(a%2==1, b))
# alternativa
# spa = sum(a%2 == 0)
# sib = sum(a%2 == 1)
if spa > sib:
    x=a.flatten()
    y=b.flatten('F')
    figure(1); clf()
    plot(x, 'b*'); plot(y, 'g*')
    ind=where(x > y)[0]
    plot(ind+1, x[ind], 'ro')
    show(False)
else:
    try:
        f=open('resultados5.txt', 'w')
        #---------------------------------
        cad=''
        n=size(c, 1)
        for i in range(n):
            cad= cad + '%i '% c[i, n-i-1]
        cad=cad + '\n'; f.write(cad)
        #-----------------------------------
        #f.write(str(diag(a[::-1]))) #alternativa
        #-----------------------------------
        z=extract(a%4==0, a)
        cad=''
        for i in z:
            cad=cad + '%s '% str(i)
        cad=cad + '\n'; f.write(cad)
        v=extract(b < a, b)
        cad=''
        for i in v:
            cad=cad + '%s '% str(i)
        cad=cad + '\n'; f.write(cad)
        f.write('det(a)=%g det(b)=%g det(c)=%g' % (det(a), det(b), det(c)))
        f.close()
    except IOError:
        print('Erro escribindo en resultados5.txt')
```

Segundo control de programación en Python de 2019

Escribe un programa chamado `exame6.py` que realice as seguintes operacións:

1. Pida por teclado un número real a e un número enteiro n no intervalo $[20, 40]$ (verificando que n está no intervalo).
2. Crea os vectores $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$ de dimensión n dados polas expresións:

$$x_i = \left\lfloor ae^{i+1} \cos^2\left(\frac{\pi(i+1)}{40}\right)\right\rfloor \quad y_i = \left\lfloor ae^{i+1} \sin^2\left(\frac{\pi(i+1)}{40}\right)\right\rfloor \quad z_i = i+1 \quad i = 0, \ldots, n-1 \quad (3)$$

3. Crea un gráfico cunha curva en 3D, onde os vectores $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$ son mostras das ecuacións paramétricas $x(i)$, $y(i)$ e $z(i)$. Ponlle un título e gardao no arquivo `figura6.png`.
4. Engade números enteiros aleatorios no intervalo $[0, 100]$ ós vectores $\mathbf{x}$ e $\mathbf{y}$ ate que teñan os elementos suficientes para poder transformalos nas matrices cadradas $\mathbf{a}$ e $\mathbf{b}$ respectivamente de orde $m = \lceil \sqrt{n} \rceil$.
5. Define a función `calcula()`, cos argumentos axeitados, que calcule os elemento da matriz $\mathbf{c}$ da seguinte maneira: para calcular c_{ij}, con $i, j = 0, \ldots, m-1$, suma os elementos da fila i de $\mathbf{a}$ e da columna j de $\mathbf{b}$ (un elemento de cada unha en cada iteración), rematando cando a suma supere o producto $a_{ij}b_{ij}$, de modo que c_{ij} sexa o número de iteracións.
6. Finalmente, o programa principal garda a matriz $\mathbf{c}$ no arquivo `saida6.txt` como números enteiros se a suma da diagonal principal é maior que a suma da diagonal segundario. No caso contrario, visualiza a matriz `c` na pantalla.

```
from numpy import *
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
a=float(input('a? '));n=0
while n<20 or n>40:
        n=int(input('20<=n(int)<=40? '))
i=arange(n);z=i+1
t=pi*z/40;u=a*exp(z)
x=floor(u*cos(t)**2)
y=floor(u*sin(t)**2)
fig=figure(1);ax=fig.gca(projection='3d')
ax.plot(x,y,z,label='curva');ax.legend();show(False)
m=int(ceil(sqrt(n)));k=m*m-n
if k>0:
        x=append(x,int(100*random([1,k])))
        y=append(y,int(100*random([1,k])))
a=x.reshape(m,m);b=y.reshape(m,m)
def calcula(a,b):
        n=a.shape[0];c=zeros([n,n])
        for i in range(n):
                for j in range(n):
                        s=0;t=a[i,j]*b[i,j]
                        for k in range(n):
                                s+=a[i,k]+b[k,j]
                                if s>t:
                                        break
                        c[i,j]=k+1
if trace(a)>trace(a[::-1]):
        savetxt('saida6.txt',c,'%i ')
else:
        print('c=', c)
```