Segundo control de programación en Python de 2025

Escribe no editor dúas liñas con igual número de números en cada liña, como o exemplo seguinte, e gardao co nome `datos1.txt`:

```
25.2  72.1  78.7  93.2  84.9
18.5  17.1  32.9  73.6  71.4
```

Escribe un programa chamado `exame1.py` que realice as seguintes operacións:

1. Lea o arquivo `datos1.txt`, comprobando que non hai erro na lectura, e almacena a primeira e segunda liña nos vectores $\mathbf{x}$ e $\mathbf{y}$, respectivamente. Sexa n a lonxitude de ambos vectores. Mostra $\mathbf{x}$ e $\mathbf{y}$ na pantalla.

2. Para cada $i = 0, \ldots, n-1$, calcula $m_i = \lfloor x_i + y_{n-i-1} \rfloor$ e calcula j da seguinte maneira: 1) se m_i é par, j será o número de elementos de $\mathbf{x}$ que hai que sumar para superar $u_i = \sum_{k=0}^{i} y_k$. Se chegas ó final de $\mathbf{x}$ sen superar u_i, entón j será igual a $n-1$; e 2) se m_i é impar, chama á función `calculo(...)`, cos argumentos axeitados, que calcule j como o número de elementos no vector $a = (x_0, x_1, \ldots, x_i)$ superiores ó seu correspondente elemento no vector $b = (y_{n-i-1}, \ldots, y_{n-1})$. Garda nun arquivo, pide o seu nome por teclado e comproba erros, os valores de j para $i = 0, 1, \ldots, n-1$.

3. Garda tamén neste arquivo o número de elementos sumados, k, e a suma de elementos, s, no seguinte proceso: xera un número enteiro aleatorio m no intervalo [0, n) e, comezando polo número na posición m suma elementos do vector $\mathbf{y}$ ate que a suma supere o valor $u = \sum_{p=0}^{n-1} x_p$. Se chegamos o final do vector $\mathbf{y}$ sen acadar u, volve a comezar no principio.

4. Representa nun gráfico de liñas os puntos do vector $\mathbf{x}$ con asteriscos azuis, o vector $\mathbf{y}$ con puntos vermellos e liñas horizontais cos valores medios de ambos vectores. Pon enreixado, títulos nos eixos e lendas no gráfico.

```
from matplotlib.pyplot import *
from numpy import *
from random import *
from sys import *
try:
    a=loadtxt('datos1.txt')
    x=a[0]; y=a[1]
except IOError:
    print('Erro lendo datos1.txt'); exit()
#-----------------------------------------
def calculo(x,y, i):
    n=len(x)
    ind=where(x[0:i+1]>y[n-i-1:n])[0]
    return len(ind)
#-----------------------------------------
n=len(x); z=zeros(n)
try:
    nome=input('Nome arquivo: ')
    fout=open(nome, 'w')
    fout.write('%10s %10s\n'%('i', 'j'))
    for i in range(n):
        mi=floor(x[i]+y[n-i-1])
        if mi%2 == 0:
            ui=sum(y[0:i+1]); s=0
```

```
            for j in range(n):
                s=s + x[j]
                if s>ui:
                    break
        else:
            j=calculo(x,y, i)
        fout.write('%10d %10d\n'%(i,j))
    u=sum(x); m=randint(0, n-1); k=0; s=0
    while s<u:
        s=s+y[m]; m=m+1; k=k+1;
        if m == n:
            m=0
    fout.write('Suma=%g con %d elementos\n'%(s,k))
    fout.close()
except IOError:
    print('Erro escribindo en %s\n'%nome); exit()
figure(1); clf(); ind=arange(n)
plot(ind, x, 'b*', label='x')
plot(ind, y, 'ro', label='y')
xlabel('Indices'); ylabel('Valores')
mx=mean(x); my=mean(y)
plot([0, n-1],[mx,mx], 'g-', label='mx')
plot([0, n-1],[my,my], 'm-', label='my')
grid(True); legend(); show()
```

Segundo control de programación en Python de 2025

Crea un arquivo de texto co nome `datos2.txt` con distinto número de números enteiros en cada liña. Por exemplo o arquivo seguinte:

```
8  10  7  4  3  7  4  1
5  6  4  3  7  3  1  8  12  4
5  4  6  8
```

Escribe un programa chamado `exame2.py` que realice as seguintes operacións:

1. Lea os elementos do arquivo `datos2.txt`, comprobando erros, a unha matriz cadrada **a** por filas recheando os elementos que falten con elementos do arquivo seleccionados aleatoriamente. Visualiza a matriz **a** na pantalla. Se m é o número de elementos do arquivo, a orde n da matriz cadrada **a** podes calculala como $n = \lceil \sqrt{m} \rceil$.

2. Define unha función `calculos(...)`, cos argumentos axeitados, que calcule os vectores **x** e **y** de lonxitude n e o número k, onde $x_i = a_{ii}$ e $y_i = a_{i(n-1-i)}$ con $i = 0, 1, \ldots, n-1$. Sexa $u = \sum_{i=0}^{n-1} x_i + \sum_{i=0}^{n-1} y_i$, calcula k como o número de elementos da matriz **a** que hai que sumar, percorrendo a matriz por columnas sen saírte da mesma, para superar u.

3. Chama a función `calculos()` e representa nun gráfico o vector **x** con puntos azuis, o vector **y** con puntos verdes, e en vermello os elementos de **x** que son maiores que o seu correspondente no vector **y**. Pon enreixado, título e lenda ó gráfico.

4. Finalmente, divide todos os elementos da matriz **a** por 2 ate que $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_{ij} < k$, visualizando na pantalla o número de veces que se realizou este proceso e o valor de k.

```
from numpy import *
from matplotlib.pyplot import *
from sys import exit
from random import *
try:
    f=open('datos2.txt', 'r')
    aux=f.read()
    x=float_(aux.rsplit()); m=len(x)
    print('x= ', x)
    n=int(ceil(sqrt(m))); n2=n*n
    for i in range(m, n2):
        x=append(x, x[randint(0,m-1)])
    a=x.reshape([n,n])
    f.close()
except IOError:
    print('Erro lendo arquivo datos2.txt\n'); exit()
print("a=", a)
def calculos(a):
    n=a.shape[0]
    x=zeros(n); y=zeros(n)
    for i in range(n):
        x[i]=a[i,i]
        y[i]=a[i,n-1-i]
    z=a.flatten('F'); m=len(z)
    s=0; u=sum(x)+sum(y)
    for i in range(m):
```

```
        s=s+z[i]
        if s > u:
            break
    return [x, y, i]
[x,y,k]=calculos(a)
figure(1); clf()
t=range(n)
plot(t,x,'b*', label='x')
plot(t,y,'ro', label='y')
ind=where(x>y)[0]
plot(ind, x[ind], 'go', label='x>y')
grid(True); legend(); show()
sa=sum(a); nv=0
while sa > k:
    a=a/2; sa=sum(a); nv +=1
print('k= %d nv=%d'%(k,nv))
```

Segundo control de programación en Python de 2025

A serie de Fourier *F(x)* de orde m dunha función $f(x)$ ven dada por:

$$F(x) = A + \sum_{k=1}^{m}(a_k \cos kx + b_k \sin kx) \tag{1}$$

onde:

$$A = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx, \qquad a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kxdx, \qquad b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kxdx \tag{2}$$

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

1. Lea por teclado a expresión analítica dunha función *f(x)* como unha cadea de caracteres e a convirta nunha función lambda. Proba con $f(x) = x**2$, $f(x) = x**3$ ou $f(x) = (x-3)**2/7$.
2. Define unha función `coeficientes(...)`, cos argumentos axeitados, que convirta a cadea anterior a unha expresión simbólica coa función `sympify` e calcule os coeficientes A, a_k e b_k da serie de Fourier para $k = 1, \ldots, m$ usando as funcións `integrate()`, `sin()` e `cos()` do módulo **Sympy**.
3. Pide por teclado o número de termos do sumatorio m, asegurando que $m > 10$ (usa m=20 cando executes o programa). Chama á función `coeficientes(...)` anterior para calcular A e $\{a_k, b_k\}_{k=1}^{m}$. Representa gráficamente a función *f(x)*, usando a función lambda anterior, e a súa serie de Fourier *F(x)* para $n = 100$ valores equiespaciados no intervalo $[-\pi, \pi]$ (vector `x`), poñendo cores distintas a cada curva e engadindo lendas ó gráfico.
4. Garda no arquivo `saida3.txt`, comprobando erros, os coeficientes A, a_k e b_k para $k = 1, \ldots, m$ en dúas columnas con dúas cifras decimais e un ancho de campo de 10. Garda tamén o valor de $\sum_{i=0}^{n-1} |f(x_i) - F(x_i)|$.

```
from numpy import *
from matplotlib.pyplot import *
import sympy as sp
expr=input('f(x)= ')
f=eval('lambda x:'+expr)
#-----------------------------------------------------
def coeficientes(expr, m):
    x=sp.symbols('x')
    a=zeros(m+1); b=zeros(m+1)
    fx=sp.sympify(expr)
    a[0]=sp.integrate(fx,(x,-sp.pi, sp.pi)).evalf()
    for i in range(1, m+1):
        a[i]=sp.integrate(fx*sp.cos(i*x), (x,-sp.pi, sp.pi)).evalf()
        b[i]=sp.integrate(fx*sp.sin(i*x), (x,-sp.pi, sp.pi)).evalf()
    a=a/pi; b=b/pi
    return [a, b]
#-----------------------------------------------------
n=100; m=0
while m<10:
    m=int(input('m= '))
[a,b]=coeficientes(expr, m)
x=linspace(-pi, pi, n)
y=zeros(n); yF=zeros(n)
k=arange(1,m+1)
for i in range(n):
```

```
    y[i]=f(x[i])
    yF[i]=a[0]/2+sum(a[1:m+1]*cos(k*x[i])+b[1:m+1]*sin(k*x[i]))
figure(1); clf()
plot(x,y, 'b-', label='f(x)')
plot(x, yF, 'g-', label='F(x)')
legend(); show()
try:
    z=hstack((a.reshape([m+1,1]), b.reshape([m+1,1])))
    savetxt('saida3.txt', z, '%10.2f')
    f=open('saida3.txt','a')
    f.write('Diferencia: %g'%sum(abs(y-yF)))
    f.close()
except IOError:
    print('Erro no arquivo saida3.txt')
```

Segundo control de programación en Python de 2025

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

1. Lee un número enteiro n por teclado, non deixando avanzar o programa ate que o número estea no intervalo [5,15] e sexa un número impar. Xera unha matriz cadrada de orde n con números enteiros aleatorios no intervalo [1,20]. Representa nun gráfico 3D a matriz **a**, onde o valor da matriz representa a altura da superficie.

2. Define a función `suavizado(...)`, cos argumentos axeitados, que realiza as seguinte operación sobre unha matriz **a**: xera unha matriz **b** coas mesmas dimensións de **a**, onde cada elemento b_{ij} da matriz transformada obtense como a media aritmética dos 9 elementos contidos nunha submatriz 3×3 centrada en a_{ij}. Para os elementos da primeira ou última fila ou columna so se deben usar os elementos existentes.

3. Executa repetidamente a función `suavizado(...)` de modo que a matriz **b** dunha execución sexa a matriz **a** da seguinte execución. A repetición debe realizarse ate que $d = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} |a_{ij} - b_{ij}| < n$ ou se realicen 20 execucións. Visualiza na pantalla o número de iteracións e o valor de d.

4. Pide o nome dun arquivo por teclado e garda nel o seguinte para cada elemento das matrices **a** (matriz orixinal do primeiro punto) e **b** (última matriz do proceso do punto anterior): se $a_{ij} < b_{ij}$, escribe -10; se $a_{ij} > b_{ij}$, escribe 10; e se son iguais, escribe 0. Escribe cada fila da matriz nunha liña.

```
from numpy import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from numpy.random import *
n=0
while n<5 or n>15 or n%2==0:
    n=int(input("n= "))
a=randint(1,20, [n,n])
b=a.copy()
print(b)
clf(); fig = figure(1);
ax=fig.add_subplot(111,projection="3d")
x = arange(n); X, Y= meshgrid(x,x)
ax.plot_surface(X, Y, a); show()
#---------------------------------
def suavizado(a):
    n=a.shape[0]; b=zeros(a.shape)
    for i in range(n):
        k1=max(i-1,0); k2=min(i+2,n)
        for j in range(n):
            l1=max(j-1, 0); l2=min(j+2,n)
            b[i,j]=mean(a[k1:k2,l1:l2])
    return b
#---------------------------------
for i in range(20):
    c=suavizado(b)
    print('----------------------'); print(c)
    d=sum(abs(b-c))
    b=c.copy()
    if d<n:
        break
```

```
print('Iteracións: %d d=%g'%(i+1,d))
nome=input('Nome arquivo: ')
c=zeros(a.shape)
ind=where(a>b)
c[ind[0], ind[1]]=10
ind=where(a<b)
c[ind[0], ind[1]]=-10
try:
    savetxt(nome, c, '%5d')
except IOError:
    print('Erro escribindo en %s'%nome)
```

Segundo control de programación en Python de 2025

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

1. Lee por teclado un número n enteiro par maior que 10, non deixando avanzar o programa ate que n sexa correcto. Crea e visualiza na pantalla os seguintes vectores $\mathbf{x}$ e $\mathbf{y}$ de lonxitude n. Cada elemento do vector $\mathbf{x}$ se define como $x_i = 1 + i^2(n-i)$, para $i = 0, 1, \ldots, n-1$. Os elementos y_i do vector $\mathbf{y}$ son números enteiros aleatorios no intervalo $[-n, n]$.

2. Se a suma dos elementos de $\mathbf{y}$, dada pola expresión $sy = \sum_{i=0}^{n-1} y_i$, é maior que 0, calcula o vector $\mathbf{z}$ de lonxitude n dado por: $z_i = x_i + y_i$ para $i = 0, 1, \ldots, n-1$. Sexa $\mathbf{t}$ un vector dado por $t_i = i$ para $i = 0, 1, \ldots, n-1$. Axusta os puntos $\{(t_i, z_i)\}_{i=0}^{n-1}$ a un polinomio de orde 2. Representa nun gráfico de liñas, os puntos $\{(t_i, z_i)\}$ en azul e a ecuación do polinomio en verde. Resalta cun círculo vermello o valor máximo de $\mathbf{z}$. Pon lendas na figura.

3. Se $sy \leq 0$, chama a unha función `calcula(...)`, cos argumentos axeitados, que calcule un vector $\mathbf{v}$ de lonxitude n, e o valor p. Cada elemento v_i será o número de elementos de $\mathbf{y}$ que hai que sumar, comenzando na posición i para superar o valor x_i. Se chegas ó final do vector $\mathbf{y}$, volve a comezar ó principio. O valor p é a posición do último elemento sumado. Desde o programa principal, garda no arquivo `saida5.txt` en n liñas o seguinte: 1) v_i se $y_i > 1$; e 2) en caso contrario v_i/y_i con dúas cifras decimais. Almacena tamén o valor de p como un enteiro.

```
from numpy import *
from matplotlib.pyplot import *
from numpy.random import *
n=0
while n<=10 or n%2==1:
    n=int(input("n= "))
i=arange(n)
x=1+i**2*(n-i)
print('x= ', x)
y=randint(-n, n+1, n)
print('y= ', y)
def calcula(x, y):
    n=len(x); v=zeros(n)
    for i in range(n):
        s=0; k=0; l=i-1
        while s<x[i]:
            l= l+1
            if l ==n:
                l=0
            s =s+y[l]; k+=1
        v[i]=k
    return [v, l]
sy=sum(y)
if sy > 0:
    z=x+y
    t=arange(n); clf()
    p=polyfit(t,z,2)
    plot(t,z,'b*', label='z(i)')
    tt=linspace(0,n,100)
    plot(tt, polyval(p,tt), 'g-', label='Polinomio')
    mm=argmax(z); plot(mm, z[mm], 'ro',label='max. z')
    legend(); show()
else:
```

```
[v, p]=calcula(x,y)
try:
    f=open('saida5.txt', 'w')
    for i in range(n):
        if y[i] > 1:
            f.write('%d\n'%v[i])
        else:
            f.write('%.2f %d\n'%(float(v[i])/y[i], p))
    f.close()
except IOError:
    print('Erro escribindo en saida5.txt')
```

Segundo control de programación en Python de 2026

Crea co editor de texto un arquivo `datos6.txt` con letras na primeira columna e números na segunda. Por exemplo, o seguinte arquivo:

a 4
b 6
c 5
d 8
s 3

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

1. Lea o arquivo anterior e almacena as letras nunha lista e os números no vector **x**.
2. Define unha función `valor(...)`, cos argumentos axeitados, que calcule o valor dunha palabra. Este valor calcúlase sumando o valor de todas as súas letras segundo o valor que figura no arquivo anterior. Asume que todas as letras que non figuran neste arquivo teñen unha puntuación de 1.
3. O programa ten que ler palabras por teclado e almacenar no vector **y** o seu valor ate que se introduza o carácter 3. Representa nun gráfico os valores de todas as palabras como puntos azuis e debuxa unha liña co valor medio das puntuacións das palabras. Pon enreixado e títulos nos eixos da gráfica.
4. Calcula unha matriz **a** de dimensión $n \times m$, sendo n e m as lonxitudes dos vectores **x** e **y** respectivamente. A matriz **a** calculase como $a_{0j} = y_j$ para $i = 0$ e $j = 0, \ldots, m-1$ e para $i = 1, \ldots, n-1$ será:

$$a_{ij} = \frac{x_{i-1} y_i}{\left(\sum_{q=0}^{n-1} x_q\right)\left(\sum_{p=0}^{m-1} y_p\right)} \tag{3}$$

   Visualiza a matriz **a** na pantalla.

```
from numpy import *
from sys import *
from random import *
from matplotlib.pyplot import *
try:
    f=open('datos6.txt', 'r'); x=[]; letra=[]
    for l in f:
        p=l.rsplit()
        letra.append(p[0])
        x.append(float(p[1]))
    f.close()
except IOError:
    print('Erro lendo datos6.txt'); exit()
print('letra= ', letra); print('x= ', x)
def valor(letra, x, palabra):
    s=0
    for l in palabra:
        if l in letra:
            ind=letra.index(l)
            s = s + x[ind]
        else:
            s = s + 1
    return s
y=[]
while True:
    pal=input('Palabra: ')
```

```
    if len(pal) == 1:
        if pal == '3':
            break
    v=valor(letra, x, pal)
    y.append(v)
print('v= ', y); clf()
n=len(x);  m=len(y); y=array(y)
plot(range(m), y, 'b*')
me=mean(y)
plot([0, m-1], [me, me], 'g-')
grid(True); xlabel('palabras'); ylabel('Puntuación'); show()
a=zeros([n,m])
x=array(x); a[0]=y.copy()
for i in range(1,n):
    for j in range(m):
        a[i,j]=x[i-1]*y[j]
a=a/sum(x)/sum(y)
print('a= ', a)
```