Segundo control de programación en Python de 2024

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

1. Pida por teclado a expresión dunha función matemática *f(x)*, podes usar $(x^2 \sin x)$. Calcula simbólicamente a derivada, *fp*, e a integral indefinida *it*.

2. Define un vector **x** con n=100 valores equidistantes entre -2π e 2π, e crea tres vectores **y**,**z** e **v** avaliando as funcións *f*, *fp* e *it* sobre o vector **x**. Representa nun gráfico 2D a función **f**, derivada e integral indefinida con cores verde, vermello e azul respectivamente no intervalo $x \in [-2\pi, 2\pi]$. Pon títulos, lendas e enreixado ó gráfico.

3. Define a función `calcula(·)`, cos argumentos axeitados, que calcule: 1) unha matriz **a** cadrada de orde k=10, con elementos:

$$a_{ij} = \left(\frac{1}{p}\sum_{l=ip-p}^{ip-1} y_l\right)\left(\frac{1}{p}\sum_{l=jp-p}^{jp-1} y_l\right), \quad p = \left\lfloor \frac{n}{k} \right\rfloor; \quad i,j = 0..k-1 \tag{1}$$

   2) o numero de elementos, q, da matriz **a** que hai que sumar (percorrendo a matriz por columnas) para superar o valor $\left(\sum_{i=1}^{n} y_i\right) + \left(\sum_{j=1}^{n} z_i\right)$

4. O programa principal chama a función `calcula()` e garda no arquivo `datos1.txt` a matriz **a**, cada fila nunha liña con formato real de ancho 10 e 1 decimal. Despois garda o valor de q como un enteiro.

```
from matplotlib.pyplot import *
from numpy import *
from numpy.random import *
from sys import *
import sympy as sp

expr=input('f(x)= ')
x=sp.symbols('x')
f=sp.lambdify(x, expr)
fp=sp.lambdify(x, sp.diff(expr, x))
it=sp.lambdify(x, sp.integrate(expr, x))
n=100; t=linspace(-2*pi, 2*pi, n);
y=f(t); z=fp(t); v=it(t)
clf(); plot(t, y, 'g-', label='f(x)')
plot(t, z, 'r--', label='fp(x)')
plot(t, v, 'b:', label='it(x)')
xlabel('x'); ylabel('Value'); grid(True); legend(); show()
def calcula(y, z):
    n=len(y); k=10; a=zeros([k,k])
    p=int(n/k)
    for i in range(k):
        for j in range(k):
            a[i,j]=sum(y[i*p-p:i*p])/p*sum(y[j*p-p:j*p])/p
    q=0; s=0; u=sum(y)+sum(z)
    w=a.flatten('F'); m=len(w)
    while s<u:
        s=s + w[q]
        q=q+1
        if q == m:
```

```
            break
    return [a, q]
[a, q]=calcula(y,z)
try:
    savetxt("datos1.txt", a, '%10.1f')
    f=open("datos1.txt", 'a')
    f.write('q=%i\n'%q)
    f.close()
except IOError:
    print('Erro escribindo datos1.txt')
```

Segundo control de programación en Python de 2024

Crea un arquivo de texto 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  2  5  8
```

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

1. Pida por teclado o nome do arquivo de texto e lea os elementos do arquivo a un vector $\mathbf{x}$ mentres que a súa suma sexa inferior a 50 ou non se acade o final do arquivo. Comproba se hai erros na lectura. Sexa n a dimensión do vector $\mathbf{x}$.

2. Crea un vector $\mathbf{y}$ de dimensión n, onde $y_0 = 0$ e $y_i = x_i - x_{i-1}$ para $i = 1, 2, \ldots, n-1$. Visualiza na pantalla os vectores $\mathbf{x}$ e $\mathbf{y}$.

3. Define unha función chamada `calcula()`, decide os argumentos axeitados, que calcule unha matriz cadrada $\mathbf{a}$ e un vector $\mathbf{z}$, ambos de orde n. Cada elemento a_{ij} ven dado pola expresión:

$$a_{ij} = \begin{cases} \displaystyle\sum_{l=0}^{n-1} \sum_{k=0}^{i} \exp(-x_l) \sin(\pi y_k x_l) & \text{se } x_i > y_j \\ \displaystyle\sum_{l=0}^{n-1} \cos(x_j + y_l) & \text{en caso contrario} \end{cases}$$

   Cada elemento z_i, con $i = 0, 1, \ldots, n-1$ será o numero de elementos k que hai que sumar para que se cumpra esta relación:

$$\sum_{l=0}^{k} |y_l| > \sum_{j=0}^{i} x_j \tag{2}$$

   comezando por o inicio do vector $\mathbf{y}$, e se chega ao final sen que se acade a condición, volve a comezar polo principio.

4. Chama a función `calcula()` para obter a matriz $\mathbf{a}$ e o vector $\mathbf{z}$. Visualiza na pantalla o vector $\mathbf{z}$ e representa como un mapa de calor a matriz $\mathbf{a}$.

```python
from numpy import *
from matplotlib.pyplot import *
from sys import exit
nome=input('Nome arquivo: ')
try:
    f=open(nome, 'r')
    suma=0; fin=True; x=[]
    while suma<50 and fin:
        aux=int_(array(f.readline().split()))
        if len(aux) == 0:
            fin=False
        for i in aux:
            suma = suma + i
            x.append(i)
            if suma >= 50:
                break
except IOError:
```

```
    print('Erro lendo arquivo %s\n'%nome); exit()
print("x=", x)
n=len(x); y=zeros(n)
for i in range(1,n):
    y[i]=x[i]-x[i-1]
print("y=", y)
def calcula(x, y):
    n=len(x); a=zeros([n,n])
    for i in range(n):
        for j in range(n):
            if x[i] > y[j]:
                s=0
                for l in range(n):
                    for k in range(i+1):
                        s = s + exp(-x[l])*sin(pi*y[k]*x[l])
            else:
                s=0
                for l in range(n):
                    s = s + cos(x[j]*y[l])
            a[i,j]=s
    z=zeros(n)
    for i in range(n):
        umbral=sum(x[:i+1]); suma=0; k=0; j=0
        while suma < umbral:
            suma = suma + abs(y[j])
            k = k+1; j += 1
            if j == n:
                j=0
        z[i]=k
    return [a, z]
[a, z]=calcula(x,y)
clf();imshow(a);title("Mapa de calor");show(False)
```

Segundo control de programación en Python de 2024

Co editor de texto crea un documento con dúas liñas con números enteiros e almacénao no arquivo con nome `datos3.txt`. Por exemplo o arquivo seguinte:

6 4 3 7 3 10 8 12 4 10 9
5 2 6 2 5 8

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

1. Lea o arquivo `datos3.txt` e almacena a primeira liña no vector **x** e a segunda liña no vector **y**. Sexa n o número de elementos de **x** e m o número de elementos de **y**. Se $n < m$ engade números enteiros aleatorios no intervalo [a, b) ao vector **x** ate que teña a mesma lonxitude que **y**, sendo a e b o elemento mínimo e máximo do vector **y**. Se $n > m$, engade elementos o vector **y** ate que o seu tamaño sexa n. Engade elementos do vector **x**, sempre que non estean contidos en **y**.

2. Define unha función **calcula()**, debes decidir os argumentos, que a partir de un vector **z** e un escalar d, devolva o índice do vector **z** no que a suma dos elementos do vector ate ese índice supera o valor d. Se chega ao final do vector **z** sen acadarse o valor d, volve a comezar polo principio.

3. Crea unha matriz **a**, de orde n, onde cada elemento a_{ij} está definido pola expresión:

$$a_{ij} = x_i \cdot \text{calcula}(\mathbf{x}, 10y_i) \cdot \text{calcula}(\mathbf{y}, 5x_j) \tag{3}$$

4. Representa a matriz **a** como as alturas dun gráfico en 3D, considerando para representar os eixos X e Y un vector de índices de filas e columnas.

```
from numpy import *
from random import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from sys import exit
try:
    f=open("datos3.txt", 'r')
    x=int_(f.readline().split())
    y=int_(f.readline().split())
except IOError:
    print('Erro abrindo datos3.txt'); exit()
n=len(x); m=len(y)
if n<m:
    a=min(y); b=max(y);
    while n<m:
        d=randint(a, b); x=append(x, d); n=len(x)
if n>m:
    while m<n:
        for i in x:
            if i not in y:
                y=append(y, i); m=len(y)
                if m == n:
                    break
print("x= ", x)
print("y= ", y)
def calcula(z, p):
    s=0; j=-1; n=len(z)
    while s<p:
        j=j+1
        if j==n:
```

```
            j=0
        s = s + z[j]
    return j
a=zeros([n,n])
for i in range(n):
    for j in range(n):
        a[i,j]=x[i]*calcula(x, 10*y[i])*calcula(y, 5*x[j])
clf(); fig = figure();
ax=fig.add_subplot(111,projection="3d")
x = arange(n); X, Y= meshgrid(x,x)
ax.plot_surface(X, Y, a); show()
```

Segundo control de programación en Python de 2024

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

1. Define a función `calcula()`, cos argumentos axeitados, que calcule un vector $\mathbf{x}$ de tamaño n usando un número real p e onde cada elemento x_i con $i = 0, 1, \ldots, n-1$ defínese como:

$$x_i = 2^i \prod_{k=1}^{i} \left[p - \cos\left(\frac{(2k-1)\pi}{2i} \right) \right] \tag{4}$$

2. Utiliza a función `calcula()` para crear un vector $\mathbf{y}$ de dimensión n=10 con p=1.5. Sexa $\mathbf{t}$ o vector cos índices dos elementos do vector $\mathbf{y}$. Atopa os coeficientes dun polinomio de orde 3 que mellor aproxima os datos $(t_i, y_i), i = 0, 1, \ldots, n-1$. Representa nun gráfico en 2D en azul con asteriscos os puntos $(t_i, y_i), i = 0, 1, \ldots, n-1$, en verde o polinomio de orde 3, e con círculos vermellos os puntos que están por debaixo do polinomio.

3. Utiliza a función `calcula()` para crear un vector $\mathbf{z}$ de dimensión $2n$ nun punto aleatorio no intervalo (1,2). Crea os vectores $\mathbf{v}$ e $\mathbf{w}$ cos elementos de $\mathbf{z}$ nas posicións pares e impares respectivamente.

4. Pide por teclado un nome de arquivo e almacena nel os vectores $\mathbf{y}$, $\mathbf{v}$ e $\mathbf{w}$ da seguinte forma: na liña primeira y_0, v_0, w_0, na segunda y_1, v_1, w_1, na liña i y_i, v_i, w_i e, na última liña o valor medio dos tres vectores. Usa un ancho de campo de 10 e dúas cifras decimais.

5. Finalmente, calcula e visualiza a matriz $\mathbf{b}$ de orde n seguinte. Cada elemento b_{ij} con $i, j = 0, 1, \ldots, n-1$ será o número k de productos $y_l v_l$ que hai que sumar (se chegas ao final dos vectores, volve a comezar polo principio) para que se cumpra a seguinte condición:

$$\sum_{l=0}^{k} y_l v_l > \sum_{q=0}^{i+j+1} z_q \tag{5}$$

```
from numpy import *
from sys import *
from random import *
from matplotlib.pyplot import *
def calcula(p, n):
    x=zeros(n)
    for i in range(n):
        s=2**i
        for k in range(1,i+1):
            s = s *(p-cos((2*k-1)*pi/(2*i)))
        x[i]=s
    return x
m=10; y=calcula(1.5, m)
print("y= ", y)
t=arange(m)
p=polyfit(t, y, 3)
px=polyval(p, t)
clf(); plot(t, y, 'b*', label='puntos')
plot(t, y, 'g-', label='polinomio')
ind=where(y<px)[0]; n=len(ind)
plot(t[ind], y[ind], 'ro', label='inferiores')
legend(); grid(True); show()
d=random()+1; print("d= ", d)
k=2*m; z=calcula(d, k)
v=z[0:k:2]; w=z[1:k:2]
```

```
y=append(y, mean(y)); v=append(v, mean(v)); w=append(w, mean(w))
a=vstack((y, v, w)); a2=a.T
nome=input('Nome arquivo:')
savetxt(nome, a2, "%10.2g")
b=zeros([m,m])
for i in range(m):
    for j in range(m):
        u=i+j+1; umbral=sum(z[0:u+1])
        k=0; l=0; s=0
        while s<= umbral:
            s = s + y[l]+ v[l]
            k = k+1
            l=l+1
            if l == m:
                l=0
        b[i,j]=k
print("b= ", b)
print("k= ", k)
```

Segundo control de programación en Python de 2024

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

1. Lea un número enteiro impar n por teclado que sexa maior que 10, comprobando que n sexa un número válido. Calcula un vector $\mathbf{x}$ de tamaño n con elemento x_i dado pola expresión:

$$x_i = \exp\left(\frac{-i}{\sigma^2}\right)\sin\left(\frac{\pi i}{3}\right), \quad \sigma = \frac{n+1}{2}, \quad i = 0..n-1 \tag{6}$$

2. Crea un vector $\mathbf{y}$ de dimensión n, onde $y_0 = \frac{x_0 + x_1}{2}$, $y_{n-1} = \frac{x_{n-1} + x_{n-2}}{2}$ e $y_i = \frac{x_{i-1} + x_i + x_{i+1}}{3}$ para $i = 1, 2, \ldots, n-2$.

3. Debuxa nun gráfico 2D os valores de $\mathbf{x}$ en azul e os de $\mathbf{y}$ en verde con asteriscos. Debuxa tamén unha liña co valor medio dos valores do vector $\mathbf{x}$ en amarelo e marca en vermello os puntos do vector $\mathbf{y}$ que son maiores que os do vector $\mathbf{x}$ na mesma posición. Pon lendas, enreixado e títulos ó gráfico e gárdao no arquivo `grafico5.png`.

4. Define unha función `calcula()`, cos argumentos axeitados, que dado un vector $\mathbf{z}$ de lonxitude m, calcule unha matriz cadrada, $\mathbf{a}$, de orde m. Cada elemento a_{ij} será: 1) Se $\sum_{l=0}^{i} z_l > \sum_{l=j}^{m-1} z_l$ o número de elementos de $\mathbf{z}$ que hai que sumar para que se supere o valor $\sum_{l=0}^{i} z_l^4$ comenzando por z_0. Se chegues ó final, volve a comezar polo principio; e 2) en caso contrario, será a suma dos elementos de $\mathbf{z}$ que son menores a media do vector $\mathbf{z}$.

5. Calcula as matrices $\mathbf{d}$ e $\mathbf{b}$ aplicando a función `calcula()` aos vectores $\mathbf{x}$ e $\mathbf{y}$ respectivamente. Garda no arquivo `saida5.txt` os elementos d_{ij} da matriz $\mathbf{d}$ que son maiores que b_{ij}, cun ancho de campo de 8 e tres cifras decimais, de xeito que os elementos de cada fila se garden nunha liña distinta.

```
from numpy import *
from matplotlib.pyplot import *
n=0
while n<10 or n%2==0:
    n=int(input("n= "))
s=(n+1)/2; s2=s**2
t=arange(n); x=exp(-t/s2)*sin(pi*t/3)
y=zeros(n)
y[0]=(x[0]+x[1])/2
y[n-1]=(x[n-1]+x[n-2])/2
for i in range(1, n-1):
    y[i]=sum(x[i-1:i+2])/3
print("x= ", x)
print("y= ", y)
clf(); plot(t, x, 'b*', label='x')
plot(t, y, 'go', label='y')
mx=mean(x); plot([0, n-1],[mx, mx], 'y-', label='V. medio x')
ind=where(y>x)[0]; plot(t[ind], y[ind], 'ro', label='y>x')
legend(); grid(); savefig('grafico5.png')
def calcula(z):
    m=len(z); a=zeros([m,m]); mz=mean(z); nz=sum(z[z<mz])
    for i in range(m):
        u1=sum(z[0:i+1])
        u=sum(z[0:i+1]**4)
        for j in range(m):
```

```
            u2=sum(z[j:m])
            if u1 >u2:
                s=0; k=0; p=0
                while s<u:
                    s = s + z[p]
                    p = p+1; k=k+1
                    if p==m:
                        p=0
                a[i,j]=k
            else:
                a[i,j]=nz
    return a
d =calcula(x); b=calcula(y)
try:
    f=open("saida5.txt", 'w')
    for i in range(n):
        ind=where(d[i] > b[i])[0]
        if len(ind)>0:
            cad=''; v=d[i,ind]
            for j in v:
                cad = cad + '%10.2g'%j
            cad= cad + '\n'
            f.write(cad)
    f.close()
except IOError:
    print("Erro escribindo en saida5.txt")
```

Segundo control de programación en Python de 2024

Crea co editor de texto o seguinte arquivo `datos6.txt`:

```
5 pepe 2 −4 xaneiro
luns 8 −3 7 ana
6 casa 14 1 mar 9 13
```

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

1. Lea o arquivo anterior os números, descarte as cadeas de caracteres, e almacene os números no vector $\mathbf{x}$. Sexa n a lonxitude do vector $\mathbf{x}$. Crea un vector $\mathbf{y}$, de lonxitude n, onde cada elemento y_i con $i = 0, \ldots, n-1$ sexa un elemento de $\mathbf{x}$ seleccionado aleatoriamente.

2. Representa nun gráfico en 2D os puntos (x_i, y_i), $i = 0, \ldots, n-1$ en azul. Calcula os valores de a e b da función recíproca $y = \dfrac{1}{ax+b}$ que mellor axusta os puntos (x_i, y_i), $i = 0, \ldots, n-1$, e represéntaa no gráfico cunha liña verde. Representa tamén o valor medio do vector $\mathbf{y}$ cunha liña azul e os valores do vector $\mathbf{y}$ menores que a función recíproca en cor vermello. Pon enreixado e lendas ó gráfico.

3. Chamar á función `calcula()`, cos argumentos axeitados, que calcule un vector $\mathbf{z}$ e unha matriz cadrada $\mathbf{d}$ de orde n. O vector $\mathbf{z}$ debe ter n elementos, sendo z_i para $i = 0, 1, \ldots, n-1$ o número de valores en $\mathbf{x}$ iguais ou maiores que i. O elemento d_{ij}, con $i, j = 0, 1, \ldots, n-1$, debe ser, se $i \geq j$, o número de elementos de $\mathbf{y}$ iguais a x_i ou a x_j. Se $i < j$, o elemento d_{ij} debe ser o número de valores de $\mathbf{x}$ que hai que sumar, partindo de x_i, para superar y_j. No programa principal, chama á función `calcula()` e visualiza na pantalla $\mathbf{z}$ e $\mathbf{d}$.

```
from numpy import *
from sys import *
from random import *
from matplotlib.pyplot import *
try:
    f=open('datos6.txt', 'r'); x=[]
    for l in f:
        p=l.rsplit()
        for i in p:
            if i.isdigit():
                x.append(float(i))
    f.close()
except IOError:
    print('Erro lendo datos6.txt'); exit()
n=len(x); print('x= ', x)
y=zeros(n)
for i in range(n):
    y[i]=choice(x)
    #alternativa m=randint(0,n-1); y[i]=x[m]
print('y= ', y); x=array(x)
clf(); plot(x, y, 'b*', label='(x,y)')
p=polyfit(x, 1.0/y, 1)
yr=1.0/(p[0]*x+p[1])
ind=argsort(x)
plot(x[ind], yr[ind], 'g-', label='Axuste')
ind=where(y < yr)[0]
plot(x[ind], y[ind], 'ro', label='y<yr')
my=mean(y)
plot([min(x),max(x)],[my, my], 'b-', label='media y')
```

```
grid(); legend(); show()
def calcula(x,y):
    n=len(x); z=zeros(n)
    for i in range(n):
        z[i]=len(where(x>=i))
    d=zeros([n,n])
    for i in range(n):
        for j in range(n):
            if i>=j:
                d[i,j]=len(where(y ==x[i])[0])+ len(where(y==x[j])[0])
            else:
                u=y[j]; k=i; s=0; l=0
                while s<u:
                    s = s +u
                    l = l+1; k=k+1
                    if k==n:
                        k=0
                d[i,j]=l
    return [z, d]
[z, d]=calcula(x,y)
print('z= ', z)
print('d= ', d)
```