Segundo control de programación en Python

Escribe un programa en Python chamado `exame1.py` que dada a serie:

$$\sum_{n=0}^{\infty} \frac{n^2+1}{5^n} \tag{1}$$

realice:

1. Define unha función chamada `sumaSerie()`, decide os argumentos que necesita, que aproxime a suma da serie considerando os termos que sexan maiores de 10^{-4} e devolva a suma da serie e un vector **x** cos sumandos.

2. O programa principal ten que chamar a función `sumaSerie` e gardar nun arquivo, pídelle ó usuario o seu nome, a seguinte información: a suma aproximada da serie, os sumandos (vectos **x**) nunha liña, o número de sumandos e a suma exacta da serie (calculada simbólicamente).

3. O programa principal ten que calcular a matriz **a** de orde n (n é o número de sumandos), onde cada elemento b_{ij}, $i, j = 0, \ldots, n-1$:

$$a_{ij} = \begin{cases} x_i & i = j \\ x_i x_j & i > j \\ x_i + x_j & i < j \end{cases} \qquad 0 \leq i, j < n$$

   e mostrar a matriz **a** na pantalla.

NOTA: con estes datos, o arquivo ten o contido:

```
Suma aproximada: 1.71869
Numero de termos: 9
1 0.4 0.2 0.08 0.0272 0.00832 0.002368 0.00064 0.0001664
Suma exacta serie: 1.71875
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from sympy import *
def sumaSerie():
    x=[]; n=0; suma=0; sumando = 1
    while sumando > 1e-4:
        x.append(sumando)
        suma = suma + sumando
        n=n+1
        sumando= (float(n)**2+1)/5**n
        print sumando
    return [suma, x]
[sumaa, x]=sumaSerie()
nome=raw_input('Nome arquivo: ')
ni=len(x) # numero de sumandos
try:
    out=open(nome, 'w')
    out.write('Suma aproximada: %g\n' % sumaa)
    out.write('Numero de termos: %d\n' % ni)
    linha=''
    for i in x:
        linha= linha + ('%g ' % i)
    out.write(linha)
```

```
    n=symbols('n')
    sumae=Sum((n**2+1)/5**n, (n,0, oo)).evalf()
    out.write('\nSuma exacta serie: %g\n' % sumae)
except IOError:
    print 'Erro escribindo no arquivo %s\n' % nome
a=zeros([ni,ni])
for i in range(ni):
    a[i,i]=x[i]
    for j in range(i+1, ni):
        a[i,j]= x[i]+x[j]
    for j in range(i):
        a[i,j]=x[i]*x[j]
print 'Matriz a: ', a
```

Segundo control de programación en Python

Crea co editor un arquivo de texto chamado `exame2.dat` co seguinte contido:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13
-2 0 2 1 5 4 11 14 20 23 30 40 43 50
```

Escribe un programa en Python chamado `exame2.py` que:

1. Lea o arquivo `exame12.dat` e almacene a primeira liña no vector **x** e a segunda no vector **v**. Ambos vectores conteñen n elementos.
2. Define unha función chamada `derivadaNumerica()`, cos argumentos axeitados, que devolva un vector **y**, de lonxitude $n-1$ de modo que $y_i = \frac{v_{i+1} - v_i}{x_{i+1} - x_i}$, $0 \leq i < n-1$.
3. O programa principal chama a función `derivadaNumerica` e visualiza **y** na pantalla.
4. Axusta os valores do vector **v** a un polinomio p de orde 2, e visualiza na pantalla os coeficientes do polinomio. Calcula o valor do polinomio nos valores de **x**, almacénaos no vector **z** e visualízao.
5. Representa gráficamente todos os vectores na mesma gráfica (**v** en vermello con cadrados, **y** en azul con asteriscos, **z** en verde con círculos). Ponlle a gráfica unha enreixado e as seguintes lendas: valores orixinais para o vector **v**, derivada numérica para o vector **y** e valores axustados para o vector **z**.
6. Calcula a suma das diferencias (en valor absoluto) entre os valores de **v** e os valores sobre o polinomio axustado (vector **z**). Visualiza na pantalla o número de elementos m que se necesita sumar para que esta suma sexa menor que 5.

NOTA: Con estes datos, debes obter:

```
y=  [ 2.   2.  -1.   4.  -1.   7.   3.   6.   3.   7.  10.   3.   7.]
Coeficientes polinomio:  [ 0.30151099  0.11332418 -1.16071429]
z=  [ -1.16071429  -0.74587912   0.27197802   1.89285714   4.11675824
   6.94368132  10.37362637  14.40659341  19.04258242  24.28159341
  30.12362637  36.56868132  43.61675824  51.26785714]
m:  4
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
try:
    a=loadtxt('exame2.dat')
    x=a[0]; v=a[1]
    def derivadaNumerica(v,x):
        n=len(x)
        y=zeros(n-1)
        for i in range(n-1):
            y[i]=float(v[i+1]-v[i])/(x[i+1]-x[i])
        return y
    y=derivadaNumerica(v,x)
    print 'y= ', y
    p=polyfit(x, v, 2)
    z=polyval(p, x)
    print 'Coeficientes polinomio: ', p
    print 'z= ', z
```

```
    clf()
    n=len(x)
    plot(x,v,'rs-', label='Valores orixinais')
    plot(x[0:n-1],y,'b*-',label='Derivada numerica')
    plot(x,z,'go-', label='Valores axustados')
    grid(); legend(loc='upper left'); show(False)
    d = 0
    i=0
    for i in range(n):
        d= d + abs(z[i]-v[i])
        print d
        if d > 5:
            break
    print 'Numero de termos: ', i
except IOError:
    print 'Erro lendo arquivo exame2.dat'
```

Segundo control de programación en Python

Escribe un programa en Python chamado `exame3.py` que:

1. Lea unha expresión matemática por teclado dependente da variable x. Proba por exemplo con $f(x) = e^{-x} \operatorname{sen} x$.
2. Calcule (simbólicamente) o valor da integral indefinida de $f(x)$ nun vector **t** de n=100 puntos equiespaciados no intervalo $[0, \pi)$ e almacene a integral no vector **y**.
3. Define unha función `integralNumerica(...)`, cos argumentos axeitados, que calcule numéricamente a integral indefinida de $f(x)$ no vector **t** segundo a expresión (almacena o resultado no vector **z**):
$$z_{i+1} = z_i + f(t_i)\Delta t, \qquad 0 \leq i < n \tag{2}$$
4. Desde o programa principal chama a función `integralNumerica` para calcular a integral numérica de $f(x)$.
5. Representa gráficamente os vectores **y** e **z** e a función $f(x)$ na mesma gráfica (**y** en vermello con cadrados, **z** en verde con círculos e $f(x)$ cunha liña azul). Ponlle a gráfica enreixado e as seguintes lendas: integral simbólica para o vector **y**, integral numérica para o vector **z** e `f(x)` para a función.
6. Calcula a suma s das diferencias (en valor absoluto) entre os valores da integral simbólica e numérica (vectores **y** e **z**). Se o valor de s é inferior a 50 mostra na pantalla unha mensaxe informando de que a aproximación numérica é aceptable.
7. Garda no arquivo chamado `saida3.txt` os vectores **t**, **y** e **z** (un vector en cada liña) con 3 cifras decimais.

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from sympy import *
from numpy import *
from matplotlib.pyplot import *
fx=raw_input('f(x)= ')
x=symbols('x')
ifx=integrate(fx, x)
intf=lambdify(x, ifx)
n=100
t=linspace(0,pi, n)
y=zeros(n)
for i in range(n):
    y[i]=intf(t[i])
f=lambdify(x, fx)
v=zeros(n)
for i in range(n):
    v[i]=f(t[i])
def integralNumerica(v, t):
    n=len(v)
    z=zeros(n)
    z[0]=v[0]
    for i in range(1, n):
        z[i]=z[i-1]+v[i-1]*(t[i]-t[i-1])
    return z
z=integralNumerica(v,t)
clf()
plot(t, v, 'b-', label='f(x)')
```

```
plot(t, y, 'rs-', label='Integral simbolica')
plot(t, z, 'go-', label='Integral numerica')
grid(); legend(loc='lower right'); show(False)
s=sum(abs(y-z))
if s < 50:
    print 'Aproximacion numerica da integral aceptable'
try:
    savetxt('saida3.txt',vstack([t,y,z]), '%.3f')
except IOError:
    print 'Erro escribindo o arquivo saida3.txt'
```

Segundo control de programación en Python

Co editor de texto escribe a seguinte matriz no arquivo `exame4.txt`:

```
1 2 3 2
4 5 6 1
```

Escribe un programa en Python chamado `exame4.py` que realice o seguinte:

1. Pida ó usuario o nome do arquivo onde esta almacenada unha matriz, lea a matriz e calcule o número de filas n e columnas m. Se a matriz non é cadrada e $n > m$, debe repetir as $n - m$ primeiras columnas pola dereita da matriz (supón que existen $n - m$ columnas). Se, polo contrario, $m > n$, debe facer o mesmo pero coas filas e por debaixo.

2. Define unha función chamada `calcula(...)`, cos argumentos que creas convintes, que sume os elementos de **a** ata que a súa suma sexa maior que 10 e retorne o vector **x** cos valores dos elementos sumados e o valor da suma.

3. Chamar a función `calcula` e visualizar na pantalla a matriz **a** , o vector **x** e o valor da suma.

4. Fai un mapa de calor da matriz **a**, ponlle un título e gárdaa no arquivo `figura4.png`.

NOTA: Usando o arquivo anterior, a saída debe ser:

```
Matriz a:
[[ 1.  2.  3.  2.]
 [ 4.  5.  6.  1.]
 [ 1.  2.  3.  2.]
 [ 4.  5.  6.  1.]]
Vector x:  [1.0, 2.0, 3.0, 2.0, 4.0]
Suma:  12.0
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from sys import exit
from pylab import *
try:
    nome=raw_input('Nome arquivo: ')
    a=loadtxt(nome)
except IOError:
    exit('Erro lendo arquivo %s' % nome)
[n,m]=a.shape
if m > n:
    for i in range(m-n):
        a=vstack([a, a[i]])
if n > m:
    a=a.T
    for i in range(n-m):
        a=vstack([a, a[i]])
    a=a.T
n=max(n,m) # orde da matriz a
def calcula(a):
    y=a.flatten()
    n=len(y)
    x=[]
    s=0
```

```
    for i in range(n):
        x.append(y[i]);s=s+y[i]
        if s > 10:
            break
    return [x,s]
xx,ss=calcula(a)
print 'Matriz a: '
print a
print 'Vector x: ', xx
print 'Suma: ', ss
clf(); imshow(a); title('Matriz a'); show(False)
savefig('figura4.png')
```

Segundo control de programación en Python

Escribe un programa en Python chamado `exame5.py` que realice o seguinte:

1. Lea dous números enteiros, n e m, por teclado no intervalo [5,10], comprobando que os números están dentro do intervalo.

2. Crea un vector **x** de dimensión n con valores aleatorios no intervalo [-n,n] e un vector **y** de dimensión m, onde cada elemento y_i ven dado pola expresión $y_i = e^{-i} i^3$, $i = 0, \ldots, m-1$.

3. Define unha función `calculaMatriz()`, cos argumentos axeitados, que devolva unha matriz **a** de orde $m \times n$ onde cada elemento $a_{ij} = x_j y_i$, $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$.

4. Chama a función `calculaMatriz` e representa os valores da matriz **a** como as alturas dun gráfico en 3D.

5. Se a suma dos valores positivos da **a** e maior que o valor absoluto da suma dos valores negativos, garda a matriz **a** no arquivo `matriz5.txt` con 3 números decimais. En caso contrario, calcula e visualiza na pantalla o vector **z**, de dimension n dado por:

$$z_i = \sum_{j=0}^{m-1} x_i y_j \qquad i = 0, \ldots, n-1 \tag{3}$$

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from numpy.random import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D
from sys import exit
n=0; m=0
while n < 5 or n > 10 or m < 5 or m > 10:
    n=int(input('n= '))
    m=int(input('m= '))
x=-n+2*n*random(n)
t=arange(m,dtype='float')
y=exp(-t)*t**3
def calculaMatriz(x,y):
    nx=len(x); ny=len(y)
    a=zeros([ny,nx])
    for i in range(ny):
        for j in range(nx):
            a[i,j]=x[j]*y[i]
    return a
a=calculaMatriz(x,y)
fig=figure(1); clf()
ax=Axes3D(fig)
x=arange(n); y=arange(m)
X, Y=meshgrid(x,y)
ax.plot_surface(X,Y, a, rstride=1, cstride=1, cmap='hot')
show(False)
sp=sum(a[a>0])
sn=sum(a[a<0])
if sp > abs(sn):
    try:
        savetxt('matriz5.dat', a, '%10.3f')
```

```
    except IOError:
        exit('Erro escribindo en matriz5.dat')
else:
    z=zeros(n)
    for i in range(n):
        z[i]=x[i]*sum(y)
    print 'z= ', z
```