Segundo control de programación en Python

1. (**1 PUNTO**) Atopa os coeficientes do polinomio de orde 3 que axusta os puntos con coordenadas **x**=(0.1, 0.5, 2, 3.4) **y**=(-1.5, 2, 5.3, 0.7).

```
from numpy import *
polyfit([0.1,0.5,2,3.4],[-1.5,2,5.3,0.7],3)
```

2. (**1 PUNTO**) Calcula a integral definida $\int_0^{\pi} \frac{\sin x}{1+\cos^2 x} dx$

```
from sympy import *
x=symbols('x')
integrate(sin(x)/(1+cos(x)**2),(x,0,pi)).evalf()
```

3. (**8 PUNTOS**) Escribe unha función chamada `sumaSerie()`, cos argumentos axeitados, que aproxime a suma da serie:

$$\sum_{n=1}^{\infty} \frac{3n}{n^p + 6} \tag{1}$$

Inclúe só os sumandos superiores a 10^{-2} e gardaos nun vector **y**. Escribe un programa que chame á función `sumaSerie()` con $p = 3$, visualice na pantalla a suma de serie e represente gráficamente o valor de cada sumando (almacenado no vector **y**), poñendo en vermello os puntos superiores á media e en verde os inferiores. Se m é o número de sumandos incluidos na serie, constrúe unha matriz a de orde m cos seguintes valores:

$$a_{ij} = \begin{cases} 100 y_i y_j & i < j \\ \frac{1}{y_i} & i > j \\ y_i & i = j \end{cases}$$

Finalmente, o programa pide ó usuario o nome dun arquivo e garda nel a matriz a, con dúas cifras decimais e un ancho de campo de 10 caracteres e unha fila en cada liña. NOTA: o número de sumandos é $n = 17$.

```
from numpy import *
from matplotlib.pyplot import *
def sumaSerie(p):
  y=[]; n=1;  suma=0
  sumando=3.0/6
  while sumando > 1e-2:
      y.append(sumando)
      suma = suma + sumando
      n=n+1
      sumando = 3.0 * n/(n**p+6)
  return [suma, y]
# chamada a funcion
[s, y]=sumaSerie(3)
print 'Suma da serie: ', s
m=len(y)
yy=array(y)
my=mean(y)
t=where(yy > my)[0]
plot(t, yy[t], 'r*')
hold(True)
t=where(yy <= my)[0]
plot(t, yy[t], 'g*')
```

```
show(False)
a=zeros([m,m])
for i in range(m):
  for j in range(m):
    if i < j:
      a[i,j]=100*y[i]*y[j]
    elif i > j:
      a[i,j]=1.0/y[i]
    else:
      a[i,j]=y[i]
nome=raw_input('Introduce nome arquivo: ')
savetxt(nome, a, '%10.2f')
```

Segundo control de programación en Python

1. (**1 PUNTO**) Escribe o comando que interpola linearmente os puntos $\{(-1,2),(0,4),(1,5),(2,3),(3,-1)\}$ con $x \in [-1,3]$ e representa gráficamente os puntos anteriores en cor vermella e os puntos interpolados cunha liña azul.

```
x=[-1,0,1,2,3]; y= [2,4,5,3,-1]; x2 = arange(min(x),max(x),0.1)
y2 = interp(x2, x, y)
plot(x,y,'ro',x2,y2,'b')
```

2. (**1 PUNTO**) Escribe un comando que resolva a ecuación $x^2 + e^x = 0$.

```
from sympy import *
x=symbols('x')
solve(x**2+exp(x),x)
```

3. (**8 PUNTOS**) Escribe un programa que lea números por teclado ata que o usuario introduza un número entre 3 e 6 (ambos incluidos), que debe ser convertido a enteiro. O programa debe crear unha matriz **a** cadrada de orde n con valores $a_{ij} = 1 + i^2 + j$, con $i, j = 0, \ldots, n-1$. Logo, o programa debe chamar á función `calcula(...)`, que debe ter os argumentos axeitados e retornar a suma s dos elementos por riba da diagonal (que non se debe incluir), un vector **x** cos elementos impares da matriz e un vector **y** de dimensión n onde:

$$y_i = \begin{cases} 3 & \text{se } \sum_{j=0}^{n-1} a_{ij} \leq \sum_{j=0}^{n-1} a_{ji} \\ 4 & \text{se } \sum_{j=0}^{n-1} a_{ij} > \sum_{j=0}^{n-1} a_{ji} \end{cases}$$

Finalmente, o programa almacena no arquivo `control.dat` a suma s, o vector **x** e **y** (cada un nunha única liña) e a matriz **a** (unha fila en cada liña do arquivo). **NOTA**: para $n = 4$, o arquivo `control.dat` debe ter o seguinte contido:

```
s= 26
x=1 3 3 5 5 7 11 13
y=3 3 3 4
a=
1 2 3 4
2 3 4 5
5 6 7 8
10 11 12 13
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
n=0
while n<3 or n > 6:
  n=int(raw_input('Introduce numero: '))
a=zeros([n,n], dtype='float');
for i in range(n):
  for j in range(n):
    a[i,j]=1+i**2+j
def calcula(a):
```

```
  s=sum(a-tril(a))
  b=ravel(a) # b=a.flatten()
  x=b[b%2==1] # x=b[where(b%2)[0]]
  at=a.T
  n=a.shape[0]
  y=zeros(n)
  for i in range(n):
    if sum(a[i]) <= sum(at[i]):
      y[i]=3
    else:
      y[i]=4
  return [s,x, y]
[s, x, y]=calcula(a)
try:
    f=open('control.dat','w')
    f.write('s= %i\nx=' % s)
    for i in range(n):
        f.write('%d ' % x[i])
    f.write('\ny=')
    for i in range(n):
        f.write('%d ' % y[i])
    f.write('\na=\n')
    for i in range(n):
        for j in range(n):
            f.write('%d ' % a[i][j])
        f.write('\n')
    f.close()
except IOError:
    print 'error abrindo control.dat'
```

Segundo control de programación en Python

1. (**1 PUNTO**) Escribe os comandos necesarios para axustar os puntos { (0,0),(-5,2),(1,-3),(7,4),(8,-1),(-7,-8),(2,6),(-3,-5) } a un polinomio de orde 7, representando os puntos con círculos de cor azul e o polinomio cunha liña de cor vermella.

```
x=[0,-5,1,7,8,-7,2,-3]; y= [0,2,-3,4,-1,8,6,-5]; p=polyfit(x,y,7)
x2 = arange(min(x),max(x),0.1);y2=polyval(p,x2)
plot(x,y,'ob',x2,y2,'r')
```

2. (**1 PUNTO**) Escribe os comandos necesarios para calcular a suma da serie $\sum_{n=1}^{\infty} \frac{n+2}{(2n+1)(n+1)n^2}$ como número real en punto flotante.

```
from sympy import *
n=symbols('n')
Sum((n+2)/(2*n+1)/(n+1)/n**2,(n,1,oo)).evalf()
```

Escribe un programa que crear unha lista **x** de lonxitude $n = 5$ con valores $x_i = i^2 + i - 2$ para $i = 0, \ldots, n-1$. O programa debe chamar á función `suma(...)` cos argumentos axeitados, que debe sumar iterativamente (comenzando en x_0) os elementos do vector **x** mentres o resultado da suma sexa menor que 100 (cando se chegue a x_{n-1}, debe volverse a x_0). A función debe retornar a devandita suma s e o número m de elementos sumados. Entón, o programa debe crear a matriz **a**, cadrada de orde n, con valores $a_{ij} = x_i^2 + x_j$ para $i, j = 0, \ldots, n-1$. Finalmente, no arquivo `control.dat` almacenar na liña i o par de números x_i, a_{ii}, con $i = 0, \ldots, n-1$. **NOTA**: debes acadar o seguinte arquivo `control.dat`:

```
-2 2
0 0
4 20
10 110
18 342
```

```
n=5
x=zeros(n)
for i in range(n):
  x[i]=i**2+i-2
def calcula(x):
    n=len(x);s=0;i=0;m=0
    while s<100:
        s=s+x[i];i=i+1;m=m+1
        if i==n:
            i=0
    return [s,m]
[s,m]=calcula(x)
a=zeros([n,n])
for i in range(n):
  for j in range(n):
    a[i,j]=x[i]**2+x[j]
try:
    f=open('control.dat','w')
    for i in range(n):
        f.write('%i %i\n' % (x[i],a[i,i]))
```

```
    f.close()
except IOError:
    print 'erro escribindo control.dat'
```

Segundo control de programación en Python

---

1. (**1 PUNTO**) Escribe os comandos que dada unha matriz $\mathbf{a} = [[6,2,7,4],[3,8,9,1]]$ de tipo real, cambien os elementos de **a** maiores que 5 pola súa metade.

```
from numpy import *
a=array([[6,2,7,4],[3,8,9,1]],'float')
where(a>5,a/2,a)
```

2. (**1 PUNTO**) Escribe os comandos necesarios para calcular $\lim_{x\to 0}\left(\frac{1}{\sin^2 x}-\frac{1}{x^2}\right)$ e o polinomio de Taylor de orde 8 da función $\sin(\cos x)$ en $x=0$.

```
from sympy import *
x=symbols('x')
limit(1/sin(x)**2-1/x,x,0)
f=sin(cos(x))
f.series(x,0,9).evalf()
```

3. (**8 PUNTOS**) Crea un arquivo de texto chamado `matriz.dat` co seguinte contido:

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

Escribe un programa que lea dende o arquivo `matriz.dat` e almacene o contido na matriz **a**. Define unha función `calcula(...)`, cos seus argumentos axeitados, que retorne: 1) o vector **x** resultante de convertir a matriz **a** nun vector; 2) o vector **t** cos elementos de **x** que dan resto 3 cando se dividen por 4; e 3) unha matriz **b** das mesmas dimensión de **a** onde:

$$b_{ij}=\begin{cases} a_{ij}a_{ji} & a_{ij}<a_{ji} \\ \dfrac{a_{ij}}{a_{ji}} & a_{ij}>a_{ji} \\ a_{ij}^3 & a_{ij}=a_{ji} \end{cases}$$

O programa ten que chamar a función `calcula(...)`. Finalmente, mentres a suma dos elementos do vector **x** sexa maior que 10, modifica o vector **x** dividindo cada elemento por 2. O programa debe mostrar o vector **x** resultante, o elementos que dan resto 3 e a matriz **b** na pantalla. **NOTA**: debes obter:

```
x=  [ 0.0625  0.125   0.1875  0.25    0.3125  0.375   0.4375  0.5     0.5625
  0.5     0.4375  0.375   0.3125  0.25    0.1875  0.125   0.1875  0.25
  0.3125  0.0625  0.1875  0.375   0.3125  0.0625  0.125 ]
t=  [ 3.  7.  7.  3.  3.  3.]
b=  [[   1.          12.          21.           2.          1.66666667]
 [   3.         343.           1.33333333   3.          1.33333333]
 [   2.33333333  48.         125.          64.         15.        ]
 [   8.          27.          64.         125.          1.        ]
 [  15.          48.           1.66666667   1.          8.        ]]
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
a=loadtxt('matriz.dat')
def calcula(a):
  x=ravel(a); t=x[x%4==3]
  # alternativa t=x[where(x%4 == 3)[0]]
  [nf,nc]=a.shape; b=zeros([nf, nc])
  for i in range(nf):
    for j in range(nc):
      if a[i,j] > a[j,i]:
        b[i,j] = a[i,j]/a[j,i]
      elif a[i,j] < a[j,i]:
        b[i,j]=a[i,j]*a[j,i]
      else:
        b[i,j]=a[i,j]**3
  return [x,t, b]
[x, t, b]=calcula(a)
while sum(x) > 10:
    x = x/2
print 'x= ', x
print 't= ', t
print 'b= ', b
```

Segundo control de programación en Python

1. (**1 PUNTO**) Escribe os comandos que permiten calcular a derivada $\dfrac{\partial^2}{\partial x \partial y}\left(\dfrac{\ln(1+x^4+y^4)}{\sqrt{x^2+y^2}}\right)$.

```
from sympy import *
x,y=symbols('x y')
f=log(1+x**4+y**4)/sqrt(x**2+y**2)
diff(diff(f,x),y)
```

2. (**1 PUNTO**) Escribe os comandos que permiten interpolar linearmente os puntos {(1,3),(2,2),(3,5),(4,8),(5,6),(6,4),(7,2)} e representar gráficamente os puntos (en cor azul) e a función interpolante cunha liña vermella.

```
from numpy import *
x=[1,2,3,4,5,6,7];y=[3,2,5,8,6,4,2]
x2=linspace(min(x),max(x),100);y2=interp(x2,x,y)
plot(x,y,'ob',x2,y2,'r');grid(True)
```

3. (**8 PUNTOS**) Crea un arquivo de texto chamado `datos.dat` co seguinte contido:

```
15 7 -10 5
23 3 8
45
1 4 7 14
20 9
18
```

Escribe un programa que pida ó usuario o nome dun arquivo, lea números mentres que a súa suma sexa inferior a 100, e os almacene nun vector **x**. Logo, o programa ten que chamar a unha función `calcula(...)`, cos argumentos axeitados, que calcule: 1) os elementos de **x** que son múltiplos de tres; e 2) unha matriz **a** de orde $m \times m$, sendo m a número de elementos de **x**, con elementos a_{ij} dados por:

$$a_{ij} = \begin{cases} x_i x_j & \text{se } i+j \text{ par} \\ x_i + x_j & \text{se } x+j \text{ impar} \end{cases}$$

Finalmente o programa debe mostrar por pantalla o vector **x**, o vector **t** cos elementos de **x** que son múltiplos de 3, a suma s dos elementos múltiplos de 3, e a diagonal da matriz **a**. **NOTA**: co arquivo `datos.dat` anterior debes obter:

```
x=15 7 -10 5 5 23 3 8 45 1 4
t=13 3 45
s=63
diag(a)=225 49 100 25 529 9 64 2025 1 16
```

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
nome=raw_input('Nome arquivo: ')
# version con for
try:
```

```
  suma=0
  f=open(nome, 'r')
  rematar=False;x=[]
  for linha in f:
     v=linha.rsplit()
     for i in v:
       t=float(i);suma += t; x.append(t)
       if suma > 100:
             rematar=True
             break
     if rematar:
       break
  print 'Suma = ', suma
  f.close()
except IOError:
  print 'Erro no arquivo %s' % nome
#-----------------------------------------
# alternativa con while (menos estable se
# se acada o final do arquivo antes de sumar 100
#  try:
#    suma=0
#  f=open(nome, 'r')
#  rematar=False; x=[]
#  while suma < 100:
#    linha=f.readline();v=linha.rsplit()
#    if len(v) == 0:
#      break
#    for i in v:
#       suma += float(i)
#       v.append(float(i))
#       if suma > 100:
#          break
#  f.close()
#except IOError:
#   print 'Erro en open ou fin de arquivo'
#-----------------------------------------
def calcula(x):
  m=len(x);w=array(x);m3=w[where(w % 3 == 0)[0]];a=zeros([m,m])
  for i in range(m):
    for j in range(m):
      if (i+j)%2: #impar
        a[i,j]=x[i]+x[j]
      else:
        a[i,j]=x[i]*x[j]
  return [m3,a]
[m3, aa]=calculos(x)
print 'Vector x: ', x
print 'Vector multiplos 3: ', m3
print 'No. multiplos 3= ', len(m3), ' que suman ', sum(m3)
print 'Diagonal a: ', int_(diag(aa))
```