execute the following line to import numpy and scipy

In [1]:
```
import numpy as np
import scipy as sp
import scipy.optimize
```

Create a numpy array containing the numbers from 0 to 20

1. print the last value
2. print the last 4 values
3. print every second value

In [2]:
```
a = np.arange(21)
print( a[-1] )
print( a[-4:] )
print( a[0::2] )
```

```
20
[17 18 19 20]
[ 0  2  4  6  8 10 12 14 16 18 20]
```

Find indices of non-zero elements from [1,2,0,0,4,0]

In [3]:
```
a = np.array([1,2,0,0,4,0])
a.nonzero()
```

Out[3]:
```
(array([0, 1, 4]),)
```

return all values greather or equal to 2 from [1,2,0,0,4,0]

In [4]:
```
a[a>=2]
```

Out[4]:
```
array([2, 4])
```

Create a null vector of size 10

In [5]:
```
nv = np.zeros(10)
nv
```

Out[5]:
```
array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

Create a vector containing 50 evenly spaced values between 0 and 2

```
In [6]: np.arange(0,2,2./49)
```

```
Out[6]: array([ 0.        ,  0.04081633,  0.08163265,  0.12244898,  0.1632
        6531,
                0.20408163,  0.24489796,  0.28571429,  0.32653061,  0.3673
        4694,
                0.40816327,  0.44897959,  0.48979592,  0.53061224,  0.5714
        2857,
                0.6122449 ,  0.65306122,  0.69387755,  0.73469388,  0.7755
        102 ,
                0.81632653,  0.85714286,  0.89795918,  0.93877551,  0.9795
        9184,
                1.02040816,  1.06122449,  1.10204082,  1.14285714,  1.1836
        7347,
                1.2244898 ,  1.26530612,  1.30612245,  1.34693878,  1.3877
        551 ,
                1.42857143,  1.46938776,  1.51020408,  1.55102041,  1.5918
        3673,
                1.63265306,  1.67346939,  1.71428571,  1.75510204,  1.7959
        1837,
                1.83673469,  1.87755102,  1.91836735,  1.95918367,  2.
        ])
```

Construct a matrix by repeating the following row 5 times: [4, 0.2, 5.6, 1.2] (hint: np.repeat)

```
In [7]: np.repeat( np.array([[4, 0.2, 5.6, 1.2]]), 5, axis=0 )
```

```
Out[7]: array([[ 4. ,  0.2,  5.6,  1.2],
               [ 4. ,  0.2,  5.6,  1.2],
               [ 4. ,  0.2,  5.6,  1.2],
               [ 4. ,  0.2,  5.6,  1.2],
               [ 4. ,  0.2,  5.6,  1.2]])
```

Create a 4x4 identity matrix

```
In [8]: np.eye(4)
```

```
Out[8]: array([[ 1.,  0.,  0.,  0.],
               [ 0.,  1.,  0.,  0.],
               [ 0.,  0.,  1.,  0.],
               [ 0.,  0.,  0.,  1.]])
```

Create a 8x8 ceckboard matrix (values 0 and 1)

In [9]:
```
a = np.zeros((8,8))
a[::2, ::2] = 1
a[1::2, 1::2] = 1
a
```

Out[9]:
```
array([[ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.],
       [ 0.,  1.,  0.,  1.,  0.,  1.,  0.,  1.],
       [ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.],
       [ 0.,  1.,  0.,  1.,  0.,  1.,  0.,  1.],
       [ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.],
       [ 0.,  1.,  0.,  1.,  0.,  1.,  0.,  1.],
       [ 1.,  0.,  1.,  0.,  1.,  0.,  1.,  0.],
       [ 0.,  1.,  0.,  1.,  0.,  1.,  0.,  1.]])
```

create a copy of the following matrix and set the first column to zeros. Check that the original matrix is ualtered

In [11]:
```
A = np.arange(16).reshape((4,4))
B = A.copy()
B[:,0] = 0
print(A)
print(B)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[ 0  1  2  3]
 [ 0  5  6  7]
 [ 0  9 10 11]
 [ 0 13 14 15]]
```

Create a random vector of size 10 and sort it

In [12]:
```
a = np.random.rand(10)
a.sort()
```

compute the mean of the random array

In [13]:
```
a.mean()
```

Out[13]:
```
0.59790004955615994
```

find the smallest element of the random arary

In [14]:
```
a.min()
```

Out[14]:
```
0.061389734107945237
```

generate the follwing matrix (without explicitly writing it)

[[1, 6, 11],
[2, 7, 12],
[3, 8, 13],
[4, 9, 14],
[5, 10, 15]]

In [15]:
```
np.reshape( range(1,16), (3,5) ).T
```

Out[15]:
```
array([[ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14],
       [ 5, 10, 15]])
```

compute the matrix product of two matrices of your choice

In [16]:
```
A = np.array( [[1,2,3],[4,5,6],[7,8,9]] )
B = np.array([[0,0,1],[0,1,0],[1,0,0]])
C = np.dot(A,B)
print(C)
```

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

compute the matrix product of the following matrices

In [17]:
```
A = np.array([[1,6],[6,2],[5,8]])
B = np.array([[2,5],[0,4],[5,5]])
np.dot(A.T, B)
```

Out[17]:
```
array([[27, 54],
       [52, 78]])
```

compute the determinant of the matrix product

In [18]:
```
np.linalg.det(C)
```

Out[18]:
```
-6.6613381477509402e-16
```

find the smallest element in matrix A

In [19]:
```
A.min()
```

Out[19]:
```
1
```

use the matrix inverse function (np.linalg.inv) to solve the following linear equation system:

- 8 x + 2 y = 24
- -4 x + y = -8

In [20]:
```
A = np.array([[8,2], [-4, 1]])
b = np.array( [24,-8] )

np.dot( np.linalg.inv(A), b )
```

Out[20]:
```
array([ 2.5,  2. ])
```

compute the integral of the polynomial x^5 - 13x^3+ 3x^2

In [21]:
```
p = sp.poly1d([1, 0, -13, 3, 1])
p.integ()
```

Out[21]:
```
poly1d([ 0.2       ,  0.        , -4.33333333,  1.5       ,  1.
        ,  0.        ])
```

search in the scipy library for a function to compute a zero-crossing (root) of the above polynomial

In [22]:
```
x_z = sp.optimize.fsolve(p, 0)
x_z
```

Out[22]:
```
array([-0.18516006])
```

Linear regression

We have a dataset which relates the final univerity grade of students to their high-school grades and grades from SAT tests[1]. The dataset is given below and containts the following columns: 1. high school grade point average, 2. Math SAT score, 3. Verbal SAT score, 4. Computer science grade point average, 5. Overall university grade point average.

Compute a linear regression to predict the overall university grade point average from the remaining variables. Hint: the standard linear regression model reads:

$$y = X * \beta$$

and the coefficients β can be computed using the following formula:

$$\beta = (X^T X)^{-1} X^T y$$

compute the predicted values for the overall university grade (y) and the residuals ($y_i - data_i$). Compute the mean residual.

[1] source: http://onlinestatbook.com/2/case_studies/sat.html (http://onlinestatbook.com/2/case_studies/sat.html)

In [34]:
```
import pickle
import pylab
data = pickle.load(open('data.p', 'rb'))

y = data[:,4]
X = data[:,:3]

b = np.dot( np.dot( np.linalg.inv(np.dot(X.T,X)), X.T ), y )
res = np.dot(X,b)
(y - res).mean()
```

Out[34]:
```
0.0036509585767943546
```

In [ ]:
```

```