NumPy

In [ ]:

```
pip install numpy
```

In [ ]:

```
import numpy
numpy.__version__
```

Vectors

In [ ]:

```
my_list = [1,2,3]
import numpy as np
arr = np.array(my_list)
print("Type/Class of this object:",type(arr))
print("Here is the vector\n--------------------\n",arr)
```

matrices

In [ ]:

```
my_mat = [[1,2,3],[4,5,6],[7,8,9]]
mat = np.array(my_mat)
print("Type/Class of this object:",type(mat))
print("Here is the matrix\n----------\n",mat,"\n----------")
print("Dimension of this matrix: ",mat.ndim,sep='') #ndim gives the dimensison, 2 for a
print("Size of this matrix: ", mat.size,sep='') #size gives the total number of element
print("Shape of this matrix: ", mat.shape,sep='') #shape gives the number of elements a
print("Data type of this matrix: ", mat.dtype,sep='') #dtype gives the data type contai

my_mat = [[1.1,2,3],[4,5.2,6],[7,8.3,9]]
mat = np.array(my_mat)
print("Data type of the modified matrix: ", mat.dtype,sep='') #dtype gives the data typ
print("\n\nEven tuples can be converted to ndarrays...")

b = np.array([(1.5,2,3), (4,5,6)])
print("We write b = np.array([(1.5,2,3), (4,5,6)])")
print("Matrix made from tuples, not lists\n----------------------------------------")
print(b)
```

'arange' and 'linspace'

In [ ]:

```
print("A series of numbers:",np.arange(5,16)) # A series of numbers from low to high
```

In [ ]:

```
print("Numbers spaced apart by 2:",np.arange(0,11,2)) # Numbers spaced apart by 2
```

In [ ]:

```
print("Numbers spaced apart by float:",np.arange(0,11,2.5)) # Numbers spaced apart by 2
```

In [ ]:

```
print("Every 5th number from 50 in reverse order\n",np.arange(50,-1,-5))
```

In [ ]:

```
print("21 linearly spaced numbers between 1 and 5\n------------------------------------
print(np.linspace(1,5,21))
```

Zeroes, Ones, empty, and Identity matrix

In [ ]:

```
print("Vector of zeroes\n--------------------")
print(np.zeros(5))
print("Matrix of zeroes\n-------------------")
print(np.zeros((3,4))) # Notice Tuples
```

In [ ]:

```
print("Vector of ones\n---------------------")
print(np.ones(5))
print("Matrix of ones\n---------------------")
print(np.ones((5,2))) # Note matrix dimension specified by Tuples
print("Matrix of 5's\n---------------------")
print(5*np.ones((3,5)))
```

In [ ]:

```
print("Empty matrix\n-------------\n", np.empty((3,5)))
```

In [ ]:

```
mat1 = np.eye(4)
print("Identity matrix of dimension", mat1.shape)
print(mat1)
```

Random number generation

In [ ]:

```
print("Random number generation (from Uniform distribution)")
print(np.random.rand(2,3)) # 2 by 3 matrix with random numbers ranging from 0 to 1, Not
```

In [ ]:

```
print("Numbers from Normal distribution with zero mean and standard deviation 1 i.e. st
print(np.random.randn(4,3))
```

In [ ]:

```
print("Random integer vector:",np.random.randint(1,100,10)) #randint (low, high, # of s
print ("\nRandom integer matrix")
print(np.random.randint(1,100,(4,4))) #randint (low, high, # of samples to be drawn in
print("\n20 samples drawn from a dice throw:",np.random.randint(1,7,20)) # 20 samples d
```

Reshaping, min, max, sort

In [ ]:

```
from numpy.random import randint as ri
a = ri(1,100,30)
b = a.reshape(2,3,5)
c = a.reshape(6,5)
print ("Shape of a:", a.shape)
print ("Shape of b:", b.shape)
print ("Shape of c:", c.shape)
print("\na looks like\n",'-'*20,"\n",a,"\n",'-'*20)
print("\nb looks like\n",'-'*20,"\n",b,"\n",'-'*20)
print("\nc looks like\n",'-'*20,"\n",c,"\n",'-'*20)

A = ri(1,100,10) # Vector of random interegrs
print("\nVector of random integers\n",'-'*50,"\n",A)
print("\nHere is the sorted vector\n",'-'*50,"\n",np.sort(A, kind='mergesort'))

M = ri(1,100,25).reshape(5,5) # Matrix of random interegrs
print("\n5x5 Matrix of random integers\n",'-'*50,"\n",M)
print("\nHere is the sorted matrix along each row\n",'-'*50,"\n",np.sort(M, kind='merge
print("\nHere is the sorted matrix along each column\n",'-'*50,"\n",np.sort(M, axis=0,
```

]:

```
print("Max of a:", a.max())
print("Max of b:", b.max())
print("Max of a location:", a.argmax())
print("Max of b location:", b.argmax())
print("Max of c location:", b.argmax())
```

Indexing and slicing

]:

```
arr = np.arange(0,11)
print("Array:",arr)
print("Element at 7th index is:", arr[7])
print("Elements from 3rd to 5th index are:", arr[3:6])
print("Elements up to 4th index are:", arr[:4])
print("Elements from last backwards are:", arr[-1::-1])
print("3 Elements from last backwards are:", arr[-1:-6:-2])

arr = np.arange(0,21,2)
print("New array:",arr)
print("Elements at 2nd, 4th, and 9th index are:", arr[[2,4,9]]) # Pass a list as a inde
```

]:

```
mat = np.array(ri(10,100,15)).reshape(3,5)
print("Matrix of random 2-digit numbers\n--------------------------------\n",mat)

print("\nDouble bracket indexing\n------------------------")
print("Element in row index 1 and column index 2:", mat[1][2])

print("\nSingle bracket with comma indexing\n----------------------------------")
print("Element in row index 1 and column index 2:", mat[1,2])
print("\nRow or column extract\n----------------------")

print("Entire row at index 2:", mat[2])
print("Entire column at index 3:", mat[:,3])

print("\nSubsetting sub-matrices\n--------------------------")
print("Matrix with row indices 1 and 2 and column indices 3 and 4\n", mat[1:3,3:5])
print("Matrix with row indices 0 and 1 and column indices 1 and 3\n", mat[0:2,[1,3]])
```

Conditional subsetting

In [ ]:

```
mat = np.array(ri(10,100,15)).reshape(3,5)
print("Matrix of random 2-digit numbers\n--------------------------------\n",mat)
print ("Elements greater than 50\n", mat[mat>50])
```

Slicing keeps the original reference, be aware of mutating the original array

In [ ]:

```
mat = np.array([[11,12,13],[21,22,23],[31,32,33]])
print("Original matrix")
print(mat)
mat_slice = mat[:2,:2]
print ("\nSliced matrix")
print(mat_slice)
print ("\nChange the sliced matrix")
mat_slice[0,0] = 1000
print (mat_slice)
print("\nBut the original matrix? WHOA! It got changed too!")
print(mat)

# Little different way to create a copy of the slixed matrix
print ("\nDoing it again little differently now...\n")
mat = np.array([[11,12,13],[21,22,23],[31,32,33]])
print("Original matrix")
print(mat)
mat_slice = np.array(mat[:2,:2]) # Notice the np.array command to create a new array no
print ("\nSliced matrix")
print(mat_slice)
print ("\nChange the sliced matrix")
mat_slice[0,0] = 1000
print (mat_slice)
print("\nBut the original matrix? NO CHANGE this time:)")
print(mat)
```

Array operations (array-array, array-scalar, universal functions)

In [ ]:

```
mat1 = np.array(ri(1,10,9)).reshape(3,3)
mat2 = np.array(ri(1,10,9)).reshape(3,3)
print("\n1st Matrix of random single-digit numbers\n----------------------------------
print("\n2nd Matrix of random single-digit numbers\n----------------------------------

print("\nAddition\n------------------\n", mat1+mat2)
print("\nMultiplication\n------------------\n", mat1*mat2)
print("\nDivision\n------------------\n", mat1/mat2)
print("\nLineaer combination: 3*A - 2*B\n-----------------------------\n", 3*mat1-2*mat

print("\nAddition of a scalar (100)\n------------------------\n", 100+mat1)

print("\nExponentiation, matrix cubed here\n----------------------------------------\n"
print("\nExponentiation, sq-root using pow function\n---------------------------------
```