In [1]:

```
#Numpy Python Continued
```

In [3]:

```
import numpy as np
#This is used to import numpy package with alias np
```

In [4]:

```
#Syntax of arange which is similar to range in core python
#np.arange([start,] stop[, step,], dtype=None)
x = np.arange(10)
```

In [5]:

```
print 'The generated range values are ', x
```

```
The generated range values are  [0 1 2 3 4 5 6 7 8 9]
```

In [7]:

```
print 'The type of the object x is ', type(x)
```

```
The type of the object x is  <type 'numpy.ndarray'>
```

In [8]:

```
x = np.arange(1,10,0.5, dtype=float)
```

In [9]:

```
print 'The type of the object x is ',  type(x)
```

```
The type of the object x is  <type 'numpy.ndarray'>
```

In [10]:

```
print 'The generated ranges values of the object x is', x
```

```
The generated ranges values of the object x is [ 1.   1.5  2.   2.5  3.
 3.5  4.   4.5  5.   5.5  6.   6.5  7.   7.5  8.
  8.5  9.   9.5]
```

In [12]:

```
print 'The number of elements in the array object x is ', x.size
```

```
The number of elements in the array object x is  18
```

In [13]:

```
print 'The total size of the object x is ', x.nbytes
```

```
The total size of the object x is  144
```

In [14]:

```
print 'The element size of the object x is ', x.itemsize
```

```
The element size of the object x is  8
```

In [19]:

```
x = np.arange(1,10,0.5, dtype=np.float32) #changing the internal data type
```

In [20]:

```
print 'The number of elements in the array object x is ', x.size
```

```
The number of elements in the array object x is  18
```

In [21]:

```
print 'The total size of the object x is ', x.nbytes
```

```
The total size of the object x is  72
```

In [23]:

```
print 'The element size of the object x is ', x.itemsize
```

```
The element size of the object x is  4
```

In [24]:

```
#Performance comparision of numpy arange v/s range function
```

In [55]:

```
%%capture compared_results
# Regular Python
%timeit python_range = range(1,10000)

#Numpy
%timeit numpy_arange = np.arange(1,10000)
```

In [56]:

```
print 'Performance comparision over two operations', compared_results
```

```
Performance comparision over two operations 10000 loops, best of 3: 20.5 u
s per loop
The slowest run took 5.76 times longer than the fastest. This could mean t
hat an intermediate result is being cached.
100000 loops, best of 3: 3.87 us per loop
```

In [61]:

```
#Mathematical operations on array
```

In [58]:

```
x = np.array([[1,2,3],[4,5,6],[7,8,9]])
y = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

In [59]:

```
print 'The array object x is', x
print 'The array object y is', y
```

```
The array object x is [[1 2 3]
 [4 5 6]
 [7 8 9]]
The array object y is [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [62]:

```
#Addition of two arrays
```

In [69]:

```
array_addition = x + y
print 'The addition of two array elements are\n', array_addition
```

```
The addition of two array elements are
[[ 2  4  6]
 [ 8 10 12]
 [14 16 18]]
```

In [70]:

```
array_subtraction = x - y
print 'The subtraction of two array elements are\n', array_subtraction
```

```
The subtraction of two array elements are
[[0 0 0]
 [0 0 0]
 [0 0 0]]
```

In [71]:

```
array_multiplication = x * y
print 'The multiplication of two array elements are\n', array_multiplication
```

```
The multiplication of two array elements are
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]
```

In [72]:

```
array_division = x  / y
print 'The division of two array elements are\n', array_division
```

```
The division of two array elements are
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

In [76]:

```
#Create a multidimensional array with Zero fills
```

In [74]:

```
zfill = np.zeros((3,3))
```

In [75]:

```
print 'The values of the zero  fill array of 3x3 dimension is\n', zfill
```

```
The values of the zero  fill array of 3x3 dimension is
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
```

In [77]:

```
#Create a multidimentional array with Zero fill having data type of int
```

In [78]:

```
zfill = np.zeros((3,3),dtype=np.int64)
```

In [79]:

```
3print 'The values of the zero  fill array of 3x3 dimension is\n', zfill
```

```
The values of the zero  fill array of 3x3 dimension is
[[0 0 0]
 [0 0 0]
 [0 0 0]]
```

In [86]:

```
#Creating a linear spaced values / samples (Vectors) linspace(start, stop, num=50, endp
oint=True, retstep=False, dtype=None)
```

In [81]:

```
linvalue = np.linspace(1,5)
```

In [82]:

```
print 'The generated lineraly spaced values are\n', linvalue
```

```
The generated lineraly spaced values are
[ 1.          1.08163265  1.16326531  1.24489796  1.32653061  1.40816327
  1.48979592  1.57142857  1.65306122  1.73469388  1.81632653  1.89795918
  1.97959184  2.06122449  2.14285714  2.2244898   2.30612245  2.3877551
  2.46938776  2.55102041  2.63265306  2.71428571  2.79591837  2.87755102
  2.95918367  3.04081633  3.12244898  3.20408163  3.28571429  3.36734694
  3.44897959  3.53061224  3.6122449   3.69387755  3.7755102   3.85714286
  3.93877551  4.02040816  4.10204082  4.18367347  4.26530612  4.34693878
  4.42857143  4.51020408  4.59183673  4.67346939  4.75510204  4.83673469
  4.91836735  5.        ]
```

In [89]:

```
linvalue = np.linspace(10,50,num=10,dtype=np.int64)
```

In [90]:

```
print 'The generated lineraly spaced values with 10 samples are\n', linvalue
```

```
The generated lineraly spaced values with 10 samples are
[10 14 18 23 27 32 36 41 45 50]
```

In [91]:

```
#Generating Random number arrays using numpy random
```

In [92]:

```
dataset = np.random.random((3,3))
```

In [93]:

```
print 'The generated random numbers for 3 x 3 array is\n', dataset
```

```
The generated random numbers for 3 x 3 array is
[[ 0.54283888  0.02027842  0.38800753]
 [ 0.62144766  0.62603785  0.22321634]
 [ 0.45693206  0.84869563  0.49225366]]
```

In [94]:

```
#Generating and Understanding Statistical functions
```

In [95]:

```
#max functionality max(object, axis=None, out=None, keepdims=False)
```

In [96]:

```
maxval = np.max(dataset)
```

In [97]:

```
print 'The max value returned from the object dataset is ', maxval
```

```
The max value returned from the object dataset is  0.848695625759
```

In [98]:

```
maxval = np.max(dataset, axis=0)
```

In [99]:

```
print 'The max value returned from the object dataset is ', maxval
```

```
The max value returned from the object dataset is  [ 0.62144766  0.8486956
3  0.49225366]
```

In [100]:

```
#min functionality min(object, axis=None, out=None, keepdims=False)
```

In [102]:

```
minval = np.min(dataset)
```

In [103]:

```
print 'The min value returned from the object dataset is ', minval
```

```
The min value returned from the object dataset is  0.0202784204495
```

In [106]:

```
minval = np.min(dataset, axis=1)
```

In [107]:

```
print 'The min value returned from the object dataset is ', minval
```

```
The min value returned from the object dataset is  [ 0.02027842  0.2232163
4  0.45693206]
```

In [108]:

```
#mean functionality mean(object, axis=None, dtype=None, out=None, keepdims=False)
```

In [113]:

```
meanvalue =  np.mean(linvalue)
```

In [114]:

```
print 'The mean value of given data set is', meanvalue
```

```
The mean value of given data set is 29.6
```

In [115]:

```
#median functionality median(object, axis=None, out=None, overwrite_input=False, keepdi
ms=False)
```

In [116]:

```
medianvalue = np.median(linvalue)
```

In [117]:

```
print 'The median value of the given data set is', medianvalue
```

```
The median value of the given data set is 29.5
```

In [118]:

```
#standard deviation functionality std(object, axis=None, dtype=None, out=None, ddof=0,
 keepdims=False)
```

In [119]:

```
stdvalue = np.std(linvalue)
```

In [120]:

```
print 'The standard deviation of the given data is ', stdvalue
```

```
The standard deviation of the given data is  12.8156154749
```

In [124]:

```
#sum functionalitye sum(object, axis=None, dtype=None, out=None, keepdims=False)
```

In [122]:

```
sumvalue = np.sum(linvalue)
```

In [123]:

```
print 'The sum value of the given data set is', sumvalue
```

```
The sum value of the given data set is 296
```

In [ ]:

```

```