Numpy

Numpy

- NumPy is short for Numerical Python.

- NumPy is a package that provides an N-dimensional array object, called ndarray, and used for mainly scientific computing.

- Unlike lists, ndarray items are all of the same type.
  - Every element in the memory take the same size.
  - ndarrays are stored at contiguous place in memory, so items can be accessed and manipulated very efficiently.

Numpy

- Python lists are slow to process. NumPy’s array is much faster.
- Numpy provides a lot of functions that work with ndarray efficiently.

```
import numpy as np
import time as t

startT = t.time()
lst = list(range(10000000))
newlst =[i*2 for i in lst]
endT = t.time()
print(endT - startT)
```

```
startT = t.time()
arr = np.arange(10000000)
arr = arr*2
endT = t.time()
print(endT - startT)
```

Creating Arrays

- There are several ways to initialize new numpy array.
  - One of the ways is to use array() function and pass a sequence such as a list or tuple.
  - We can use other functions that are dedicated initialize numpy arrays such as zeros(), ones(), arange(), linspace(), etc. We will see them later.

```
import numpy
arr = numpy.array([1, 2, 3])
print(arr)

import numpy as np
arr = np.array([1, 2, 3])
print(arr)
print(type(arr))
```

```
import numpy as np
arr = np.array( (1, 2, 3) )
print(arr)
print(type(arr))

arr = numpy.array( range(10) )
```

Dimensions in Arrays

- 0-D Arrays: the elements in an array. That is each array element is 0-D array.
- 1-D Arrays: An array that has 0-D elements.
- 2-D Arrays: An array that has 1-D elements.
- 3-D Arrays: An array that has 2-D elements.
- …any number of dimensions
- Examples:

```
a = np.array( 2 )
b = np.array( [ 2,3 ] )
c = np.array([ [2,3], [4,5] ])
d = np.array( [ [[2,3], [4,5]], [[6,7], [8,9]] ])
```

Dimensions in Arrays

- We can use the ndim attribute of an array object to get the dimension of the array.

```
a = np.array( 2 )
b = np.array( [ 2,3 ] )
c = np.array([ [2,3], [4,5] ])
d = np.array( [ [[2,3], [4,5]], [[6,7], [8,9]] ])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Dimensions in Arrays

- We can set the dimension of an array using the ndmin argument.

```
a = np.array([1,2], ndmin = 3)
print(a, a.ndim)   # 3
```

- To get the total number of elements in array, we can use the size attribute.

```
a = np.array([ [1,2],[2,3] ])
a.size # 4
```

Array shape and dtype

- The shape attribute of ndarray is a tuple of array dimensions. It defines the number of elements in each dimension.

```
ar= np.array([[1,2,3,4,5], [3,4,5,6,7],[1,3,4,6,7]])
print(ar.shape)  # (3,5)
```

- The dtype of ndarray defines the data type of the elements of array .

```
ar = np.array([2.1, 3, 4, 89])
print(ar.dtype)  #float64

ar = np.array([2, 3,4, 89])
print(ar.dtype)  #int32
```

```
ar = np.array([2, 3,4, 89], dtype=np.int16)
print(ar.dtype) #int16
```

Creating Arrays

- numpy.ones(), numpy.zeros()

```
a = numpy.ones((2,4)) # creates 2 rows, 4 columns of 1.0
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
a = numpy.ones((2,4), dtype=np.int32)
[[1 1 1 1]
 [1 1 1 1]]
a = numpy.zeros((2,4)) # creates 2 rows, 4 columns 0.0
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

Creating Arrays

- numpy.arange()

```
a = numpy.arange(10) # creates an array of 10 elements: 0,1,…,9
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
 a = numpy.arange(3,6, dtype=float)
[3., 4., 5.]
a = numpy.arange(2, 5, 0.7)
[2.  2.7 3.4 4.1 4.8]
```

Creating Arrays

- numpy.linspace()

```
a = numpy.linspace(2, 8, 4) # 4 elements between 2 and 8
[2. 4. 6. 8.]


a = numpy.linspace(2, 8, 6) # 6 elements between 2 and 8
[2.  3.2  4.4  5.6  6.8  8. ]      # notice that the differences b/w any two
                                     consecutive numbers are equal
```

Creating Arrays

- numpy.random.rand()

```
a = numpy.random.rand(2, 3) # 2*3 6 random elements between 0 and 1


[[0.31149921 0.09762174 0.75596525]
 [0.02047554 0.38856551 0.30889683]]
```

Creating Arrays

- empty(), fill(), full()

```
a = numpy.empty(3)
[1.48539705e-313 2.33419537e-313 3.81959242e-313]
a.fill(5)
[5. 5. 5.]
a.shape = (3,1)
[[5.]
 [5.]
 [5.]]
a = numpy.full((3,1),5)
```