Numpy Arrays

NumPy arrays basically are available in flavors: vectors and matrices. Vectors are strictly 1-dimensional (1D) arrays and matrices are 2D

Why use Numpy array? Why not just a list?

There are lot's of reasons to use a Numpy array instead of a standard python list object. Our main reasons are:

- Memory Efficiency of Numpy Array vs list
- Easily expands to N-dimensional objects
- Speed of calculations of numpy array
- Broadcasting operations and functions with numpy
- All the data science and machine learning libraries we use are built with Numpy

In [1]:

```
import numpy as np
```

In [2]:

```
# Example of creating array
my_list = [1,2,3]
my_array = np.array([1,2,3])
```

In [3]:

```
my_list
```

Out[3]:

```
[1, 2, 3]
```

In [4]:

```
my_array
```

Out[4]:

```
array([1, 2, 3])
```

In [5]:

```
# Creating the numpy array from objects
my_list = [1,2,3]
my_list
```

Out[5]:

```
[1, 2, 3]
```

In [6]:

```
np.array(my_list)
```

Out[6]:

```
array([1, 2, 3])
```

In [7]:

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

Out[7]:

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

In [8]:

```
np.array(my_matrix)
```

Out[8]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Built-in Methods to create arrays

arange = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.arange.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.arange.html)

Return evenly space values within a given interval

In [9]:

```
np.arange(0,10)
```

Out[9]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [10]:

```
np.arange(0,11,2)
```

Out[10]:

```
array([ 0,  2,  4,  6,  8, 10])
```

zeros and ones =https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.zeros.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.zeros.html)

Genrate array zeros and ones

In [11]:

```
np.zeros(3)
```

Out[11]:

```
array([0., 0., 0.])
```

In [12]:

```
np.ones(3)
```

Out[12]:

```
array([1., 1., 1.])
```

In [13]:

```
np.zeros((2,4))
```

Out[13]:

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

linspace = https://numpy.org/devdocs/reference/generated/numpy.linspace.html (https://numpy.org/devdocs/reference/generated/numpy.linspace.html)

Return evenly spaced number over a specified interval

In [14]:

```
np.linspace(0,10,3)
```

Out[14]:

```
array([ 0.,  5., 10.])
```

In [15]:

```
#linspace include the endpoint if we apply False to it it not consider it

np.linspace(5,50,10,endpoint=False)
```

Out[15]:

```
array([ 5. ,  9.5, 14. , 18.5, 23. , 27.5, 32. , 36.5, 41. , 45.5])
```

eye = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.eye.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.eye.html)

create an identity matrix

In [16]:

```
np.eye(5)
```

Out[16]:

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

Random

Numpy use random to create random numbers in array

rand = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.rand.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.rand.html)

it is use to create the random numbers between [0 , 1) interval

In [17]:

```
np.random.rand(5)
```

Out[17]:

```
array([0.01986564, 0.95377721, 0.64904178, 0.6204818 , 0.34869965])
```

In [18]:

```
np.random.rand(2,4)
```

Out[18]:

```
array([[0.35202741, 0.55333394, 0.87465856, 0.01692987],
       [0.27112875, 0.18894195, 0.21676078, 0.61773166]])
```

randn = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.rand.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.rand.html)

Returns a sample (or samples) from the "standard normal" distribution [σ = 1]. Unlike rand which is uniform, values closer to zero are more likely to appear.

In [19]:

```
np.random.randn(3)
```

Out[19]:

```
array([ 1.93913066,  0.5234611 , -0.93108175])
```

In [20]:

```
np.random.randn(2,4)
```

Out[20]:

```
array([[-0.26079239,  0.55264743, -0.3071679 , -0.61451923],
       [ 1.72363626,  0.42301345, -0.09784662, -1.92473877]])
```

randint = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.randint.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.randint.html)

Return random integers from low (inclusive) to high (exclusive)

In [21]:

```
np.random.randint(5,15)
```

Out[21]:

```
7
```

In [22]:

```
np.random.randint(1,11,9)
```

Out[22]:

```
array([ 9, 10,  5, 10,  4, 10,  5,  5,  5])
```

seed = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.seed.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.seed.html)

It is used toset a random state so random result same

In [23]:

```
np.random.seed(42)
np.random.rand(2)
```

Out[23]:

```
array([0.37454012, 0.95071431])
```

In [24]:

```
np.random.seed(42)
np.random.rand(2)
```

Out[24]:

```
array([0.37454012, 0.95071431])
```

Array Attributes and Methods

Let's discuss some useful attributes and methods for an array:

In [25]:

```
arr = np.arange(25)
ranarr = np.random.randint(0,50,10)
```

In [26]:

```
arr
```

Out[26]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

In [27]:

```
ranarr
```

Out[27]:

```
array([42,  7, 20, 38, 18, 22, 10, 10, 23, 35])
```

reshape = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.reshape.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.reshape.html)

Return data with new shape

In [28]:

```
arr.reshape(5,5)
```

Out[28]:

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

max, min, argmax, argmin

These are useful methods for finding max or min values.

argmin - Find location of min argmax- Find location of max

In [29]:

```
ranarr
```

Out[29]:

```
array([42,  7, 20, 38, 18, 22, 10, 10, 23, 35])
```

In [30]:

```
ranarr.max()
```

Out[30]:

```
42
```

In [31]:

```
ranarr.min()
```

Out[31]:

```
7
```

In [32]:

```
ranarr.argmin()
```

Out[32]:

```
1
```

In [33]:

```
ranarr.argmax()
```

Out[33]:

```
0
```

shape

Shape is an attribute that arrays have

In [34]:

```
arr.shape
```

Out[34]:

```
(25,)
```

In [35]:

```
arr.reshape(1,25)
```

Out[35]:

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
        16, 17, 18, 19, 20, 21, 22, 23, 24]])
```

In [36]:

```
# 1 row 25 column
arr.reshape(1,25).shape
```

Out[36]:

```
(1, 25)
```

In [37]:

```
# 25 row 1 column
arr.reshape(25,1)
```

Out[37]:

```
array([[ 0],
       [ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12],
       [13],
       [14],
       [15],
       [16],
       [17],
       [18],
       [19],
       [20],
       [21],
       [22],
       [23],
       [24]])
```

In [38]:

```
arr.reshape(25,1).shape
```

Out[38]:

```
(25, 1)
```

dtype = https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.ndarray.dtype.html (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.ndarray.dtype.html)

You can also grab the data type of the object in the array

In [39]:

```
arr.dtype
```

Out[39]:

```
dtype('int32')
```

In [40]:

```
arr2 = np.array([8.3, 2.4])
arr2.dtype
```

Out[40]:

```
dtype('float64')
```