Class 1 NumPy Creating Arrays and Indexing

April 13, 2020

0.0.1 NumPy – Numerical Python

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called axes.

Array oriented computing

Efficiently implemented multi-dimensional arrays

Designed for scientific computation

NumPy is very convenient to work with, especially for matrix multiplication and reshaping.

NumPy (or Numpy) is a Linear Algebra Library for Python, the reason it is so important for Finance with Python is that almost all of the libraries in the PyData Ecosystem rely on NumPy as one of their main building blocks. Plus we will use it to generate data for our analysis examples later on!

[7]:
```
#Importing NumPy in the name of np
import numpy as np
```

[10]:
```
my_list = [1,2,3]
my_list
```

[10]:
```
[1, 2, 3]
```

[11]:
```
np.array(my_list)
```

[11]:
```
array([1, 2, 3])
```

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

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

[9]:
```
np.array(my_matrix)
```

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

0.1 Built-in Methods

There are lots of built-in ways to generate Arrays

0.1.1 arange

Return evenly spaced values within a given interval.

[14]:
```
np.arange(0,10)
```

[14]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[15]:
```
np.arange(0,11,2)
```

[15]:
```
array([ 0,  2,  4,  6,  8, 10])
```

0.1.2 Full, zeros and ones

Generate arrays of zeros or ones and Generate Arrays in full of one number

[33]:
```
np.zeros(3)
```

[33]:
```
array([0., 0., 0.])
```

[17]:
```
np.zeros((5,5))
```

[17]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

[18]:
```
np.ones(3)
```

[18]:
```
array([1., 1., 1.])
```

[19]:
```
np.ones((3,3))
```

[19]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

[31]:
```
import numpy as np
x=np.full([3,4],10)
x
```

[31]:
```
array([[10, 10, 10, 10],
       [10, 10, 10, 10],
       [10, 10, 10, 10]])
```

0.1.3 linspace

Return evenly spaced numbers over a specified interval.

[20]:
```
np.linspace(0,10,3)
```

[20]:
```
array([ 0.,  5., 10.])
```

[21]:
```
np.linspace(0,10,50)
```

[21]:
```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
        2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
        3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
        4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
        5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
        6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
        7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
        8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
        9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

0.2 eye

Creates an identity matrix

[22]:
```
np.eye(4)
```

[22]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

0.3 Random

Numpy also has lots of ways to create random number arrays:

0.3.1 rand

Create an array of the given shape and populate it with random samples from a uniform distribution over `[0, 1)`.

[23]:
```
np.random.rand(2)
```

[23]:
```
array([0.24815076, 0.35774108])
```

[24]:
```
np.random.rand(5,5)
```

[24]:
```
array([[0.28130727, 0.25403702, 0.17749458, 0.17322046, 0.32408795],
       [0.96609571, 0.92121299, 0.93105032, 0.56956694, 0.96019604],
```

```
       [0.91748726, 0.52487437, 0.5285074 , 0.74287312, 0.37783356],
       [0.53596245, 0.95291456, 0.4824493 , 0.13054033, 0.0517984 ],
       [0.31271295, 0.21617196, 0.16929273, 0.91581897, 0.69063903]])
```

[3]:
```
np.random.randint(1,100)
```

[3]:
```
31
```

[4]:
```
np.random.randint(1,100,10)
```

[4]:
```
array([54,  8, 88, 35, 67, 14, 53, 56, 80, 62])
```

0.4 Array Attributes and Methods

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

[5]:
```
arr = np.arange(25)
ranarr = np.random.randint(0,50,10)
```

[6]:
```
arr
```

[6]:
```
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])
```

[7]:
```
ranarr
```

[7]:
```
array([36, 48, 45, 40, 22,  8, 32,  6, 27, 23])
```

0.5 Indexing a 2D array (matrices)

Indexing and Slicing

The general format is **arr_2d[row][col]** or **arr_2d[row,col]**. I recommend usually using the comma notation for clarity.

[1]:
```
import numpy as np
my_matrix = np.array([[1,2,3],[4,5,6],[7,8,9]])
my_matrix
```

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

[7]:
```
my_matrix[:,2]
```

[7]:
```
array([3, 6, 9])
```

[24]:
```
my_matrix[0:2]
```

[24]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

[25]:
```
my_matrix[0:2,1:3]
```

[25]:
```
array([[2, 3],
       [5, 6]])
```

[27]:
```
#row,col
my_matrix[:,2]
```

[27]:
```
array([3, 6, 9])
```

[28]:
```
my_matrix[1:2,1:2]
```

[28]:
```
array([[5]])
```

0.5.1 Great Job