1-NumPy-array

August 5, 2020

1 EE4708: Data Analytics Laboratory (R-Slot)

#Course: Introduction to NumPy

2 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Numpy is also incredibly fast, as it has bindings to C libraries. We will only learn the basics of NumPy, to get started we need to install it!

[1]:
```
import numpy as np
```

Numpy has many built-in functions and capabilities. We won't cover them all but instead we will focus on some of the most important aspects of Numpy: vectors,arrays,matrices, and number generation. Let's start by discussing arrays.

3 Numpy Arrays

NumPy arrays are the main way we will use Numpy throughout the course. Numpy arrays essentially come in two flavors: vectors and matrices. Vectors are strictly 1-d arrays and matrices are 2-d (but you should note a matrix can still have only one row or one column).

Let's begin our introduction by exploring how to create NumPy arrays.

3.1 Creating NumPy Arrays

3.1.1 From a Python List

We can create an array by directly converting a list or list of lists:

[19]:
```
my_list = [1,2,3]
my_list
```

[19]:
```
[1, 2, 3]
```

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

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

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

```
[20]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

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

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

3.2 Built-in Methods

There are lots of built-in ways to generate Arrays

3.2.1 arange

Return evenly spaced values within a given interval.

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

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

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

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

3.2.2 zeros and ones

Generate arrays of zeros or ones

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

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

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

```
[26]: 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.]])
```

[27]:
```
np.ones(3)
```

[27]:
```
array([ 1.,  1.,  1.])
```

[28]:
```
np.ones((3,3))
```

[28]:
```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

3.2.3 linspace

Return evenly spaced numbers over a specified interval.

[29]:
```
np.linspace(0,10,3)
```

[29]:
```
array([  0.,   5.,  10.])
```

[31]:
```
np.linspace(0,10,50)
```

[31]:
```
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.        ])
```

3.3 eye

Creates an identity matrix

[37]:
```
np.eye(4)
```

[37]:
```
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```

3.4 Random

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

3.4.1 rand

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

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

```
[47]: array([ 0.11570539,  0.35279769])
```

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

```
[46]: array([[ 0.66660768,  0.87589888,  0.12421056,  0.65074126,  0.60260888],
             [ 0.70027668,  0.85572434,  0.8464595 ,  0.2735416 ,  0.10955384],
             [ 0.0670566 ,  0.83267738,  0.9082729 ,  0.58249129,  0.12305748],
             [ 0.27948423,  0.66422017,  0.95639833,  0.34238788,  0.9578872 ],
             [ 0.72155386,  0.3035422 ,  0.85249683,  0.30414307,  0.79718816]])
```

3.4.2 randn

Return a sample (or samples) from the "standard normal" distribution. Unlike rand which is uniform:

```
[48]: np.random.randn(2)
```

```
[48]: array([-0.27954018,  0.90078368])
```

```
[45]: np.random.randn(5,5)
```

```
[45]: array([[ 0.70154515,  0.22441999,  1.33563186,  0.82872577, -0.28247509],
             [ 0.64489788,  0.61815094, -0.81693168, -0.30102424, -0.29030574],
             [ 0.8695976 ,  0.413755  ,  2.20047208,  0.17955692, -0.82159344],
             [ 0.59264235,  1.29869894, -1.18870241,  0.11590888, -0.09181687],
             [-0.96924265, -1.62888685, -2.05787102, -0.29705576,  0.68915542]])
```

3.4.3 randint

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

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

```
[50]: 44
```

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

```
[51]: array([13, 64, 27, 63, 46, 68, 92, 10, 58, 24])
```

3.5 Array Attributes and Methods

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

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

```
[56]: arr
```

```
[56]: 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])
```

```
[57]: ranarr
```

```
[57]: array([10, 12, 41, 17, 49,  2, 46,  3, 19, 39])
```

3.6 Reshape

Returns an array containing the same data with a new shape.

```
[54]: arr.reshape(5,5)
```

```
[54]: 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]])
```

3.6.1 max,min,argmax,argmin

These are useful methods for finding max or min values. Or to find their index locations using argmin or argmax

```
[64]: ranarr
```

```
[64]: array([10, 12, 41, 17, 49,  2, 46,  3, 19, 39])
```

```
[61]: ranarr.max()
```

```
[61]: 49
```

```
[62]: ranarr.argmax()
```

```
[62]: 4
```

```
[63]: ranarr.min()
```

```
[63]: 2
```

```
[60]: ranarr.argmin()
```

```
[60]: 5
```

3.7 Shape

Shape is an attribute that arrays have (not a method):

```
[65]: # Vector
      arr.shape
```

```
[65]: (25,)
```

```
[66]: # Notice the two sets of brackets
      arr.reshape(1,25)
```

```
[66]: 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]])
```

```
[69]: arr.reshape(1,25).shape
```

```
[69]: (1, 25)
```

```
[70]: arr.reshape(25,1)
```

```
[70]: 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]])
```

[76]:
```
arr.reshape(25,1).shape
```

[76]:
```
(25, 1)
```

3.7.1 dtype

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

[75]:
```
arr.dtype
```

[75]:
```
dtype('int64')
```