Introduction to Numpy

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

Introduction to Numpy

- Numpy is a Python package used for numerical computation and multi-dimensional array operations.
- It is a vast library of methods and modules that support a wide variety of operations.
- It is the fundamental building block of higher-level packages such as Pandas and TensorFlow.
- Numpy provides built-in objects (ndarrays) which are multi-dimensional arrays of homogeneous data type.
- It provides vectorized operations on multi-dimensional arrays which are very fast and efficient compared to iterative operations.

Importing Numpy and Creating Arrays

We can import the numpy package as follows:

```
import numpy
```

We can create aliases for the package we download so that we could call the methods more easily as follows:

```
import numpy as np
```

Creating a numpy array from a list is simple as follows:

```
import numpy as np
first_array = np.array([1,2,3,4,5])
```

Creating Multi-dimensional Arrays

Multi-dimensional arrays can be created as follows:

```
a = [1, 2, 3, 4, 5]
b = [1,4,9,16,25]
squares = np.array([a,b])
print(squares)
```

```
[[ 1  2  3  4  5]
 [ 1  4  9 16 25]]
```

```
# Checking the dimensions
print(squares.shape)
```

```
(2,5)
```

Checking the Type of the Array

Checking the type of the array is as easy as follows:

```
squares.dtype
dtype('int32')

# Creating a 3-dimensional array
# We can use the same lists again and add an
  additional dimensional array to it.
cubes = np.array([[a,b,[1,8,27,64,225]]])
cubes.ndim
3

# Accessing the elements using indexing
cubes[0][1]
array([  1,   4,   9,  16,  25])
```

Common Numpy Operations

```
a = np.array([1,2,3,4])
# Get the type of a
type(a)
numpy.ndarray
# assigning a float to an int array will
# truncate the decimal part.
a[0] = 5.6
print(a)
array([5, 2, 3, 4])
# Convert to list
b = a.tolist()
type(b)
list
```

Create Numpy Array from Random

```
# Create a numpy array from a random set of
  integers with a specific size
rand_array = np.random.randint(100, size=(6, 6))
```

```
[[14  36  45  51  94  59]
 [22  76  84  16  77  36]
 [ 4  62  76  45  32  94]
 [77  22  84  56  47  82]
 [18  54  10  86  88  81]
 [86  32   2  96  82  33]]
```

Slice and Dice a Numpy Array

```
# get only the column 3 through 4 from row 0
rand_arr[0,3:5]
array([51, 94])
# Get the elements at the bottom right corner
rand_arr[4:,4:]
array([[88, 81],
       [82, 33]])
# Get a complete row
rand_arr[2,:] # the 3rd row .. i.e index 2
array([ 4, 62, 76, 45, 32, 94])
# Get a complete row
rand_arr[:,3] # the 4th row.. i.e index 3
array([51, 16, 45, 56, 86, 96])
```

Alternate Rows and Columns

```
rand_arr[::2] # Getting alternative rows
array([[14, 36, 45, 51, 94, 59],
       [ 4, 62, 76, 45, 32, 94],
       [18, 54, 10, 86, 88, 81]])
rand_arr[:,::2] # Getting alternate columns
array([[14, 45, 94],
       [22, 84, 77],
       [ 4, 76, 32],
       [77, 84, 47],
       [18, 10, 88],
       [86,  2, 82]])
```

Taking Strides in The Array

```
# Lets print random array for the next exercise
rand_arr
array([[14, 36, 45, 51, 94, 59],
       [22, 76, 84, 16, 77, 36],
       [ 4, 62, 76, 45, 32, 94],
       [77, 22, 84, 56, 47, 82],
       [18, 54, 10, 86, 88, 81],
       [86, 32,  2, 96, 82, 33]])
rand_arr[2::2,::2]
array([[ 4, 76, 32],
       [18, 10, 88]])
```

Slices are References

```
# Lets create an array
a = np.array([1,2,34,5,563])
print(a)
array([  1,   2,  34,   5, 563])
# Lets take a slice of a and assign it to b
b = a[2:5]
# Lets add an item to b's 0th index position
b[0] = 252
# But turns out a changed as well along with b
a
```

That is because slices are references and not separate objects and hence any change made through the references pointing to a slice of an array creates changes in the original array. This is called broadcasting.

Summary