Package Expo: NumPy

MIST 6380: Group 10

Basics of Numpy Arrays

1. Attributes of Arrays
2. Indexing of Arrays
4. Joining & Splitting of Arrays
3. Slicing of Arrays
4. Reshaping of Arrays

Attributes of Arrays

01

Each array has 3 attributes

- ndim: number of dimensions
- shape : size of each dimension
- size: total size of the array

Other helpful attributes:

- dtype: data type of the array
- itemsize: size in bytes of each array element
- Nbytes: total size of array

```
x3 = np.random.randint(10, size=(3, 4, 5))
```

```
print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)
```

```
x3 ndim:  3
x3 shape: (3, 4, 5)
x3 size:  60
```

```
print("dtype:", x3.dtype)
```

```
dtype: int64
```

```
print("itemsize:", x3.itemsize, "bytes")
print("nbytes:", x3.nbytes, "bytes")
```

```
itemsize: 8 bytes
nbytes: 480 bytes
```

Array Indexing

02

Indexing in NumPy is similar to Python's standard list indexing.

You specify the desired index in square brackets

*One key difference between Python lists and NumPy arrays is that NumPy Arrays have a fixed type

```
In [5]: x1
Out[5]: array([5, 0, 3, 3, 7, 9])

In [6]: x1[0]
Out[6]: 5
```

```
In [10]: x2
Out[10]: array([[3, 5, 2, 4],
                [7, 6, 8, 8],
                [1, 6, 7, 7]])

In [11]: x2[0, 0]
Out[11]: 3
```

```
In [15]: x1[0] = 3.14159  # this will be truncated!
         x1
Out[15]: array([3, 0, 3, 3, 7, 9])
```

Array Slicing
03

NumPy slicing syntax:

X[start:stop:step]

The default for these values are:

- Start = 0
- Stop = size of dimension
- Step = 1

```
x
```

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

```
x[:5]  # first five elements
```

```
array([0, 1, 2, 3, 4])
```

```
x2
```

```
array([[12,  5,  2,  4],
       [ 7,  6,  8,  8],
       [ 1,  6,  7,  7]])
```

```
x2[:2, :3]  # two rows, three columns
```

```
array([[12,  5,  2],
       [ 7,  6,  8]])
```

Reshaping Arrays

04

In order to reshape an array, the size of the initial array must match the size of the reshaped array

- When possible, reshape will use a no-copy view of the initial array

You can also use newaxis as an alternative to reshape

```
x = np.array([1, 2, 3])

# row vector via reshape
x.reshape((1, 3))
```

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

```
# column vector via reshape
x.reshape((3, 1))
```

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

Joining & Splitting Arrays

05

Joining arrays is accomplished using mainly these methods in NumPy:

- np.concatenate
- np.vstack
- np.hstack

```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
```

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

Splitting arrays is accomplished using mainly these methods in NumPy:

- np.split
- np.hsplit
- np.vsplit

*N split-points prdocus N+1 subarrays

```
x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3, 5])
print(x1, x2, x3)
```

```
[1 2 3] [99 99] [3 2 1]
```

References

https://jakevdp.github.io/PythonDataScienceHandbook/02.02-the-basics-of-numpy-arrays.html

https://realpython.com/numpy-tutorial/

Time for a Demo