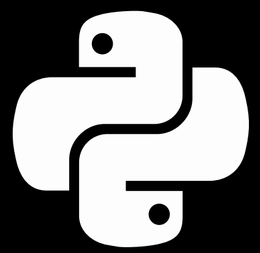


Working with Numpy arrays

Numpy

- NumPy is a Python library used for working with arrays
- It also has functions for working in domain of linear algebra, fourier transform, and matrices
- Numpy stands for numerical python

Why Numpy

- In Python we have lists that serve the purpose of arrays, but they are slow to process

- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists

- Arrays are very frequently used in data science, where speed and resources are very important

Why Numpy array over list

- Both are used to store collections of items, but they have some key differences
- Numpy arrays are stored at one continious place in memory
- Whereas python lists are not stored at one continious place in memory
- Since Numpy arrays are stored in a single block of memory, it makes accessing and processing elements faster

Installation

Import Numpy

Numpy Arrays

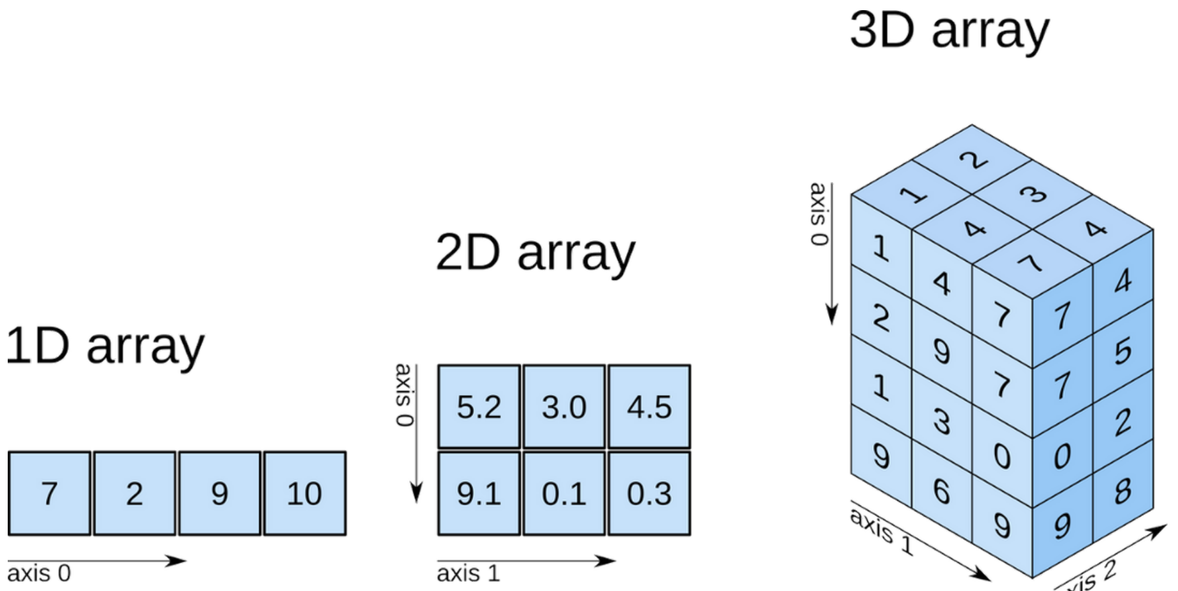
- NumPy is used to work with arrays

- The array object in NumPy is called ndarray

- We can create a NumPy ndarray object by using the array() function

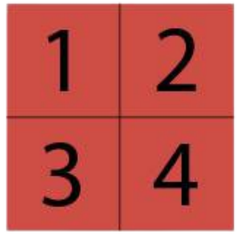
```
x = np.array([1, 2, 3, 4, 5])


print(x)


print(type(x))
```

Dimensions in Arrays



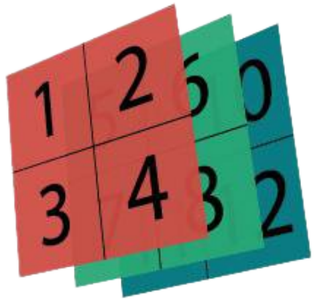
Vector

```
np.array([1, 2])
```



Matrix

```
np.array([[1, 2], [3, 4]])
```



3D Matrix

```
np.array([[[1, 2], [3, 4]],
          [[5, 6], [7, 8]],
          [[9, 10], [11, 12]]])
```

0 D Array

```
x = np.array(55)


print(x)
```

1 D Array

- An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array

```
x = np.array([1, 2, 3, 4, 5])


print(x)
```

1-D Array

2 D Array

- An array that has 1-D arrays as its elements is called a 2-D array

```
x = np.array([[1, 2, 3], [4, 5, 6]])


print(x)
```

1	9	1	7
9	1	7	1

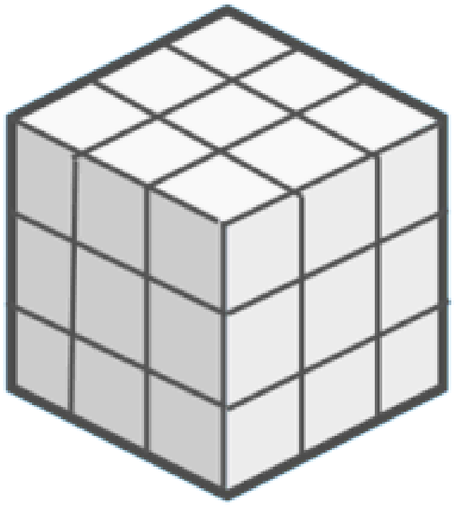
2-D Array

3 D Array

- An array that has 2-D arrays as its elements is called 3-D array

```
x = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])


print(x)
```

```
x = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
              --------------------   ----------------------
```



3-D Array

Array indexing

- Array can be accessed with the help of index number

```
x = np.array([1, 2, 3, 4])



print(x[0])
```

Access 2 D array

- 2 D array can be accessed with dimension and index number seperated by comma

```
x = np.array([[1,2,3,4,5], [6,7,8,9,10]])


print(x[0, 1])


print(x[1, 1])
```

Access 3 D array

- 3 D array can be accessed with dimensions and index number seperated by comma

```
x = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])


print(x[0, 1, 2])


x = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
             ---------------------- 0 (dimension)
                          --------- 1 (dimension)
                                ---- 2 (index)
```

Array slicing

- Slicing a numpy array is similar to string slicing in python

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



print(x[2:5])
```

Slicing 2 D array

- slice a 2D array by specifying the indices for each dimension

```
x = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(x[0, 0:3])

print(x[1, 0:3])
```

```
x = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

              ---------------   ----------------

                     0                  1
```

Slicing 2 D array

```
x = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])


print(x[0:2, 4])  - - - - - > from both elements, index 4 is returned


y = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])


print(y[0:2, 1:4]) - - - - > returns 2d array of 1 to 3rd index from both elements
```

Numpy copy

```
x = np.array([1, 2, 3, 4, 5])

y = x.copy()

x[0] = 10



print(x)

print(y)
```

Numpy view

```
x = np.array([1, 2, 3, 4, 5])

y = x.view()

x[0] = 10



print(x)

print(y)
```

Array shape

```
x = np.array([[1, 2, 3], [5, 6, 7]])



print(x.shape)
```

Reshaping array

```
# reshape 1-D into 2-D:


x = np.array(["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"])


y = x.reshape(5, 2)


print(y)
```

Reshaping array

```
# reshape 1-D into 3-D:


x = np.array(["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "l", "m"])


y = x.reshape(2, 3, 2)


print(y)
```

Reshaping array

```
# Convert any array into 1-D:


x = np.array([[1, 2, 3], [4, 5, 6]])


y = x.reshape(-1)


print(y)
```

Iterating array

```
# Iteration through 1-D:


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


for a in sample:
  print(a)
```

Iteration through 2-D

```
# Iteration through 2-D:


sample = np.array([[1, 2, 3], [4, 5, 6]])


1) for a in sample:
    print(a)


2) for a in sample:
    for b in a:
      print(b)
```

Iteration through 3-D

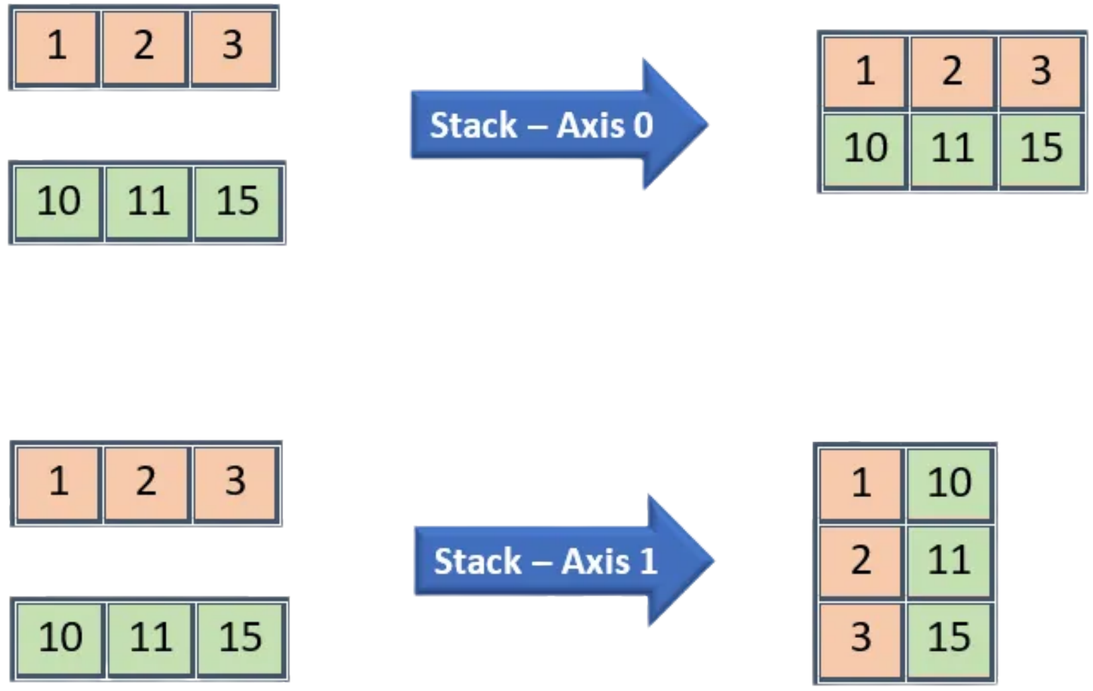
```
# Iteration through 3-D array:


sample = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])


for a in sample:
   for b in a:
    for c in b:
        print(c)
```

Join arrays

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


y = np.array([4, 5, 6])


z = np.concatenate((x, y))

print(z)
```

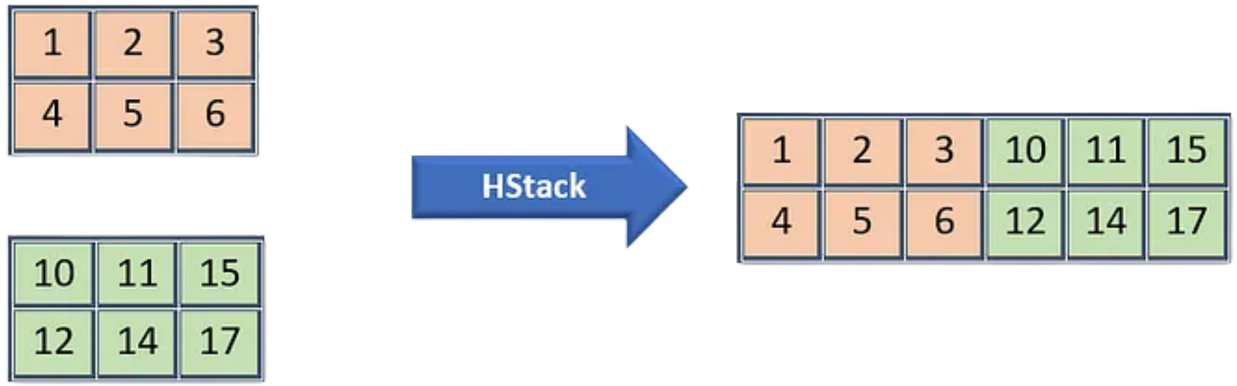
Join arrays

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


y = np.array([[4,5,6]])


sample1 = np.concatenate((x,y), axis=0)


sample2 = np.concatenate((x,y), axis=1)
```



hstack

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


y = np.array([4, 5, 6])


sample = np.hstack((x, y))
```

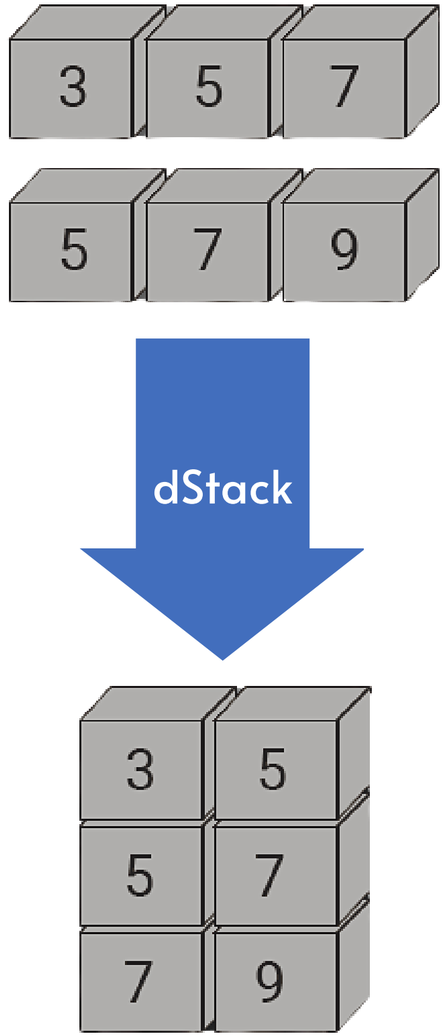


vstack

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


y = np.array([4, 5, 6])


sample = np.vstack((x, y))
```

dstack

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


y = np.array([4, 5, 6])


sample = np.dstack((x, y))
```



Array split

```
x = np.array([1, 2, 3, 4, 5, 6])



y = np.array_split(x, 4)

                        -------> split into 4 arrays

print(y)
```

split 2 D array

```
x = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])


y = np.array_split(x, 3)

                    -------> split into 3 arrays

print(y)
```

split along axis

```
x = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])


y = np.array_split(x, 3, axis=1)


print(y)
```

hsplit, vsplit, dsplit

```
a = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])


b = np.array([[[1, 2], [3, 4], [5, 6]], [[7, 8], [9, 10], [11, 12]]])


horizontal = np.hsplit(a, 2)

vertical = np.vsplit(a, 2)

depth = np.dsplit(b, 2)
```

Array search

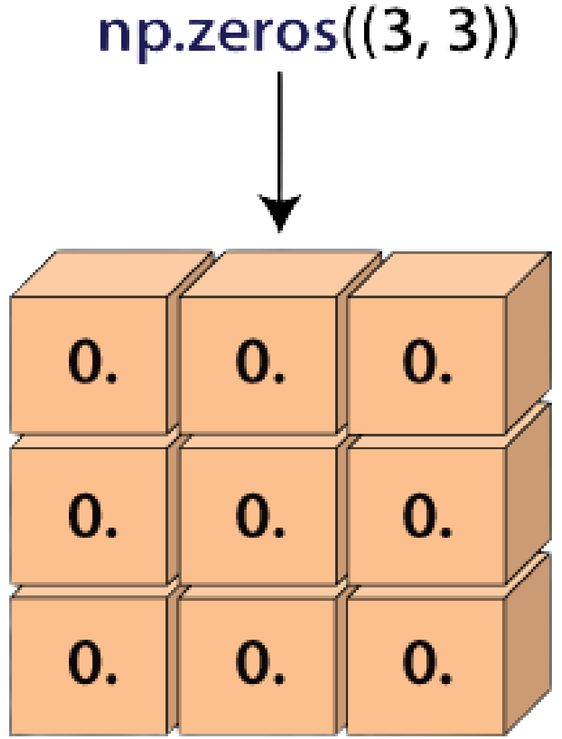
```
x = np.array(["a", "b", "a", "c", "d", "e", "a"])


new = np.where(x=="a")


print(new)
```

Sorting arrays

```
sample1 = np.array([3, 2, 0, 1])


print(np.sort(sample1))


sample2 = np.array([[3, 2, 4], [5, 0, 1]])


print(np.sort(sample2))
```

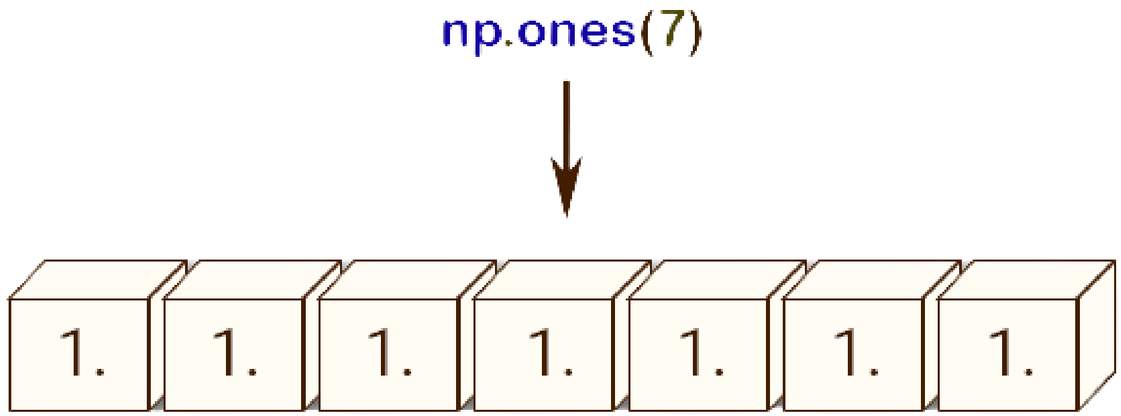
Numpy Zeros

```
x = np.zeros((3,3), dtype="i")
```



Numpy ones

```
x = np.ones((3,3), dtype="i")
```



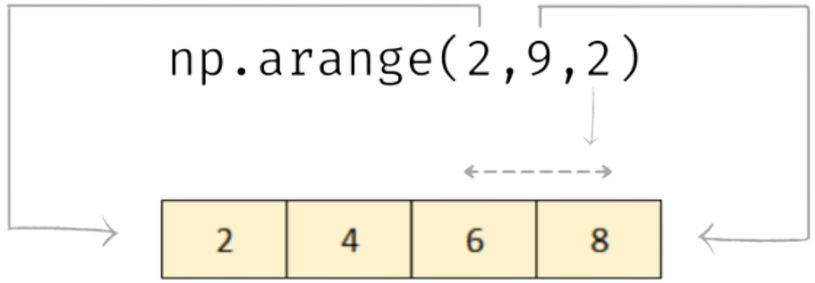
Numpy full

```
x = np.full((3,4),7)
```

7	7	7	7
7	7	7	7
7	7	7	7

Numpy arange

```
x = np.arange(2, 10)


y = np.arange(2, 10, 2)
```



Mathematical manipulation

```
a = np.array([1,2,3,4])

b = np.array([2,4,6,8])



print(np.add(a,b))

print(np.subtract(a,b))

print(np.multiply(a,b))

print(np.divide(a,b))
```

Mathematical manipulation

```
arr1=np.array([[1,2,3,4,5],[6,7,8,9,10]])

print(np.sum(arr1,axis=1))

print(np.sum(arr1,axis=0))

print(np.mean(arr1,axis=0))

print(np.mean(arr1,axis=1))
```

Mathematical manipulation

```
ran = np.random.random((2,4))


ran2 = np.random.randint(10,100,(3,5))


x = np.linspace(10,30,5)
```