Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

Amazon Web Services

MLOps with AWS

Masterclass

aws

Machine Learning Operations with AWS

Day -8

Amazon Web Services

MLOps with AWS

Masterclass

Pantech e Learning

DIGITAL LEARNING SIMPLIFIED

Numpy

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

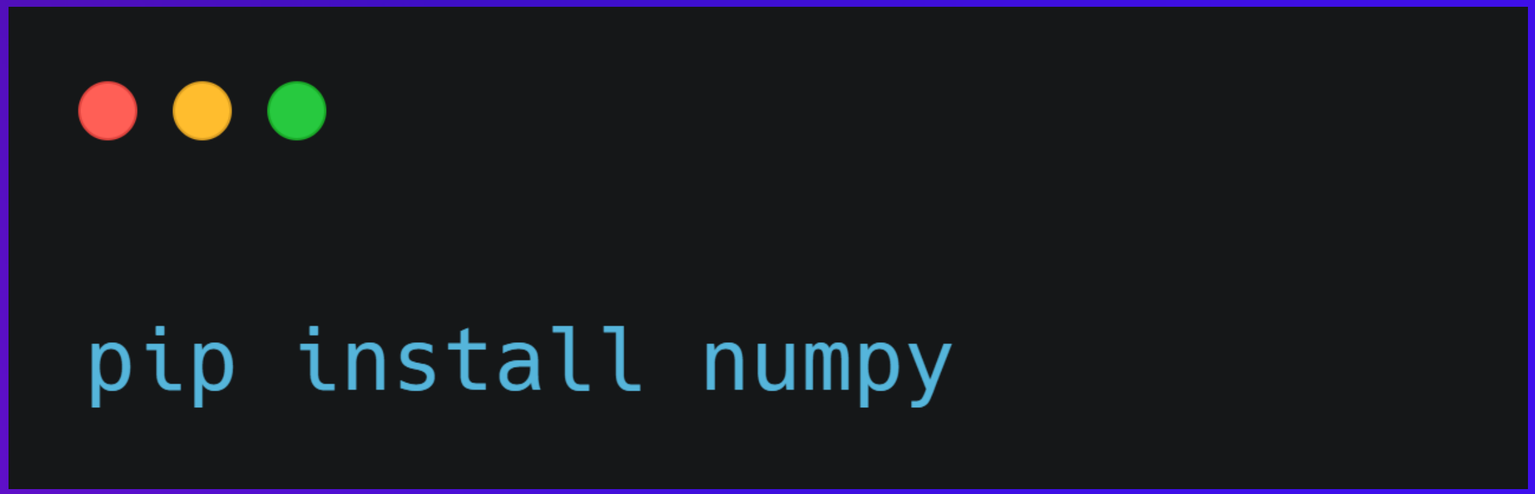
Why Numpy ?

- 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

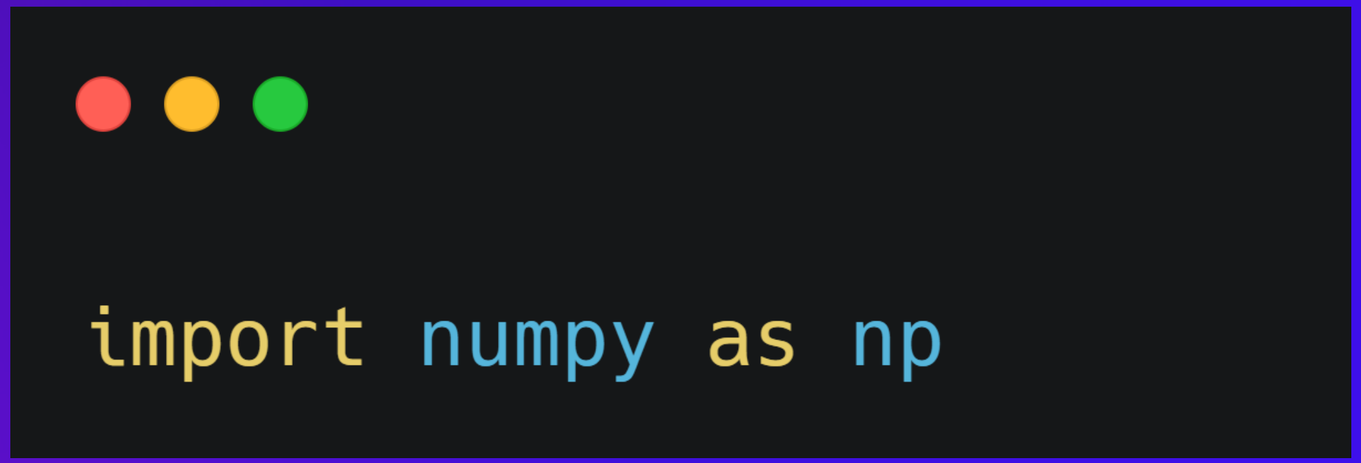
Why Numpy array over python list?

- 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

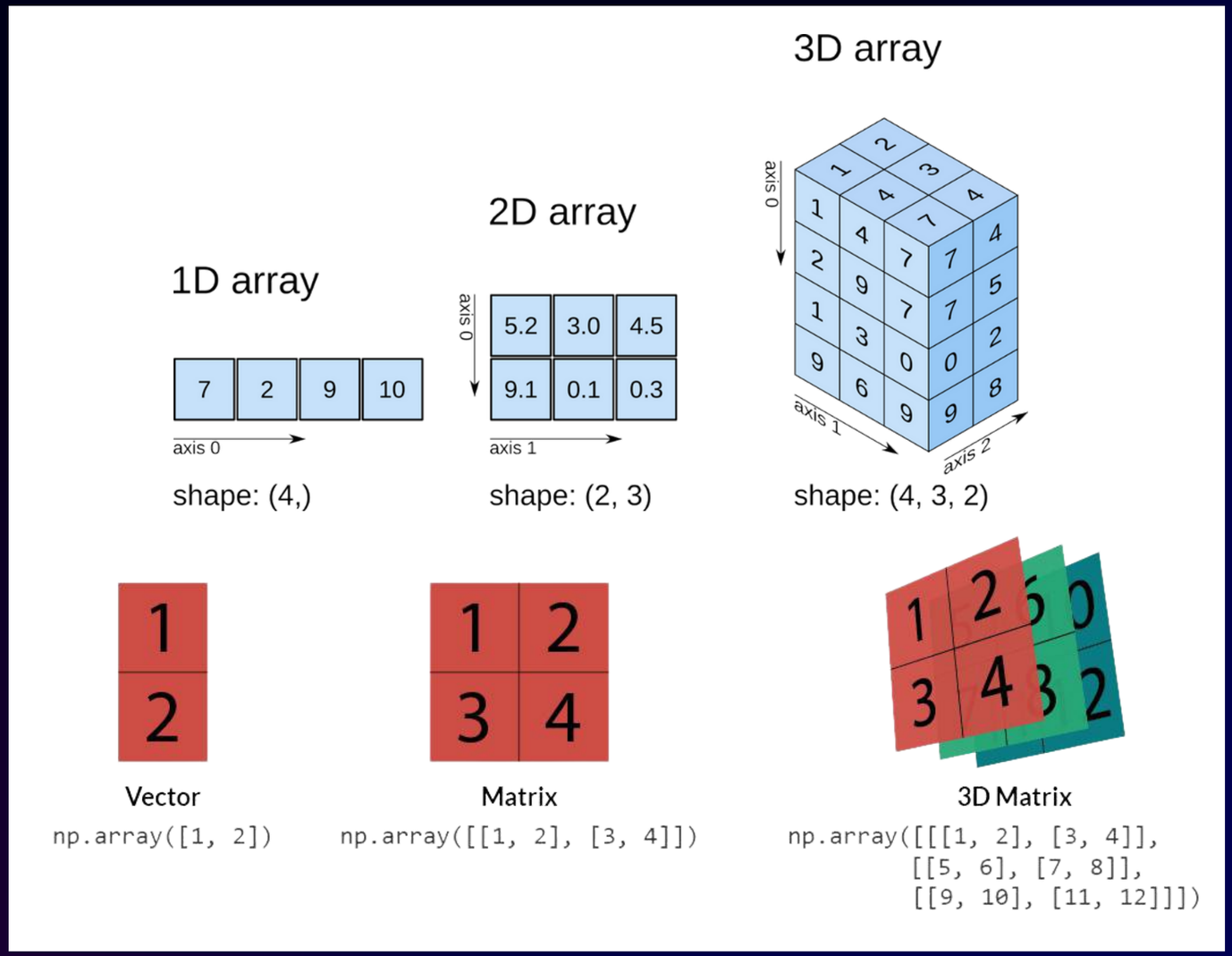


Numpy Array

- 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

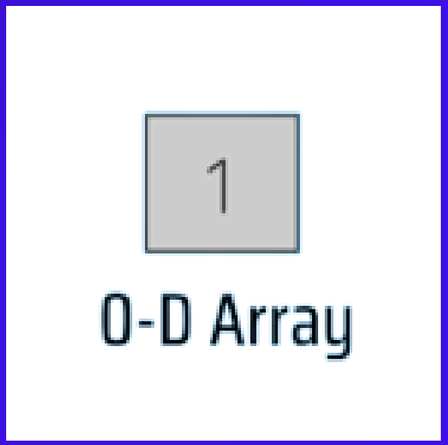
```
x = np.array([1, 2, 3, 4, 5])


print(x)


print(type(x))
```

Dimensions in Array



0-D array

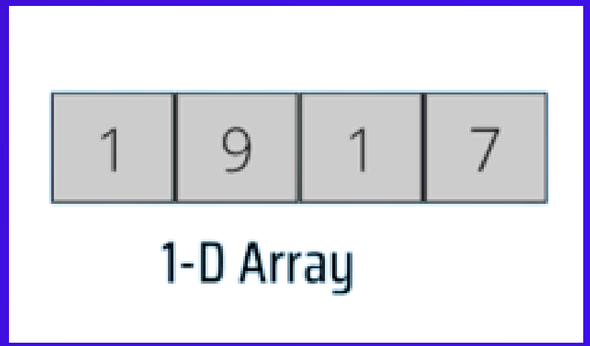
```
x = np.array(55)


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)
```



2-D array

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

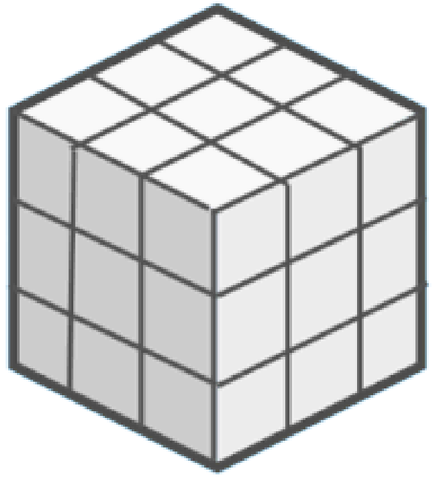
```
x = np.array([[1, 2, 3], [4, 5, 6]])


print(x)
```

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)
```

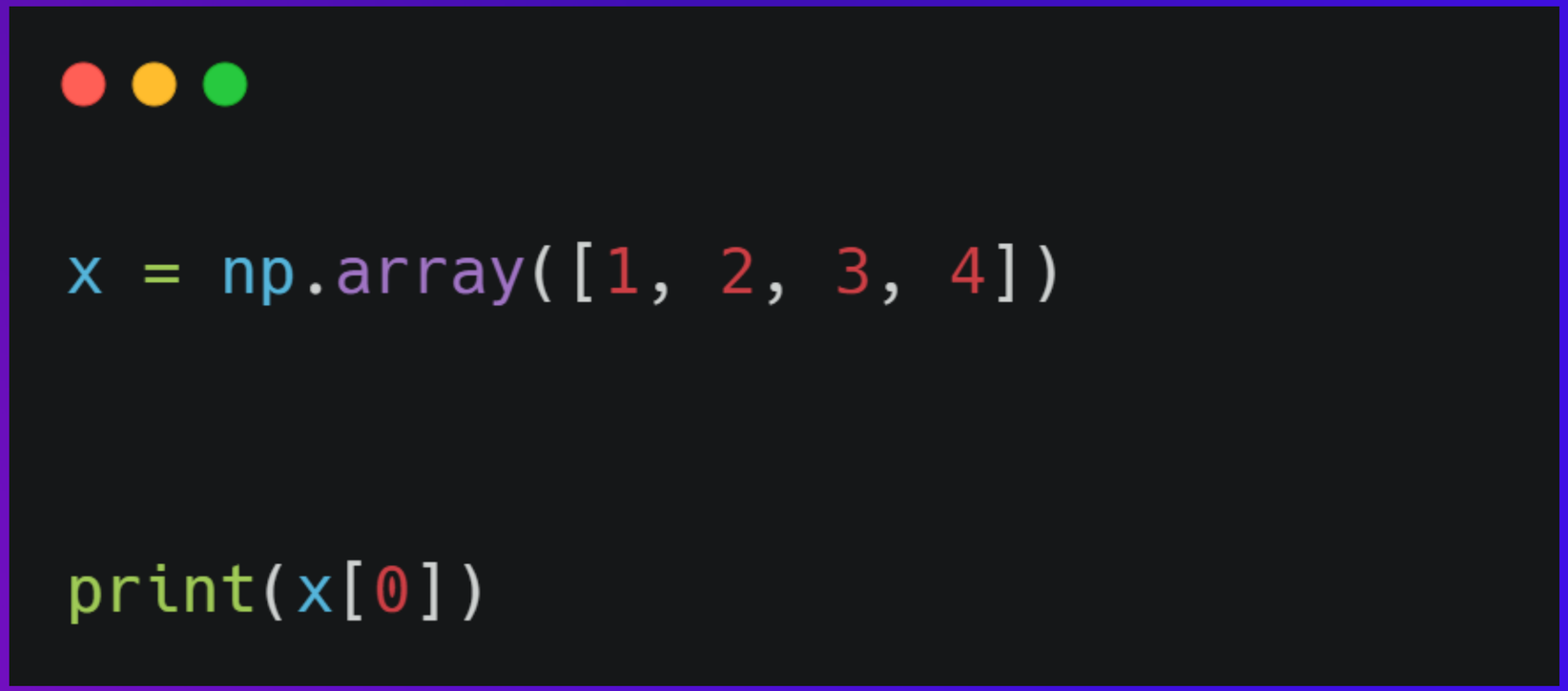
```
x = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
              --------------------    ----------------------
```



3-D Array

Array Indexing

Array can be accessed with the help of index number



Access 2D 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 3D 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])
```

2D Array Slicing

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
```

2D Array Slicing

```
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
```

Array Shape

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



print(x.shape)
```

Reshaping Array

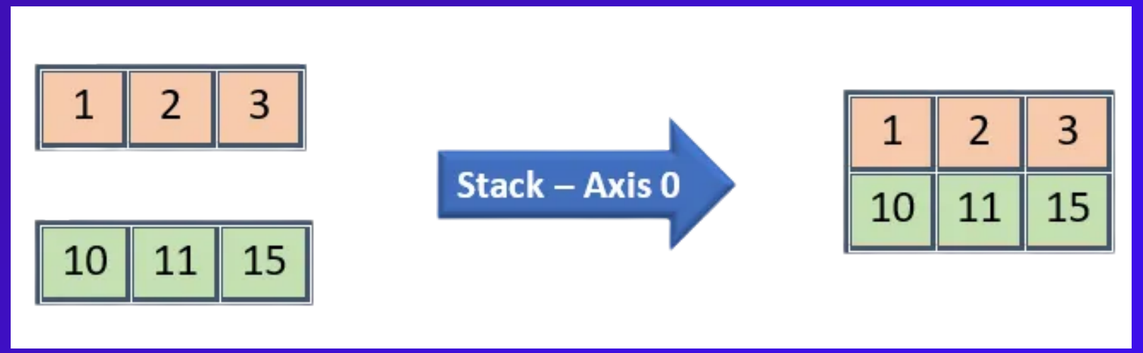
```
# Convert any array into 1-D:


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


y = x.reshape(-1)


print(y)
```

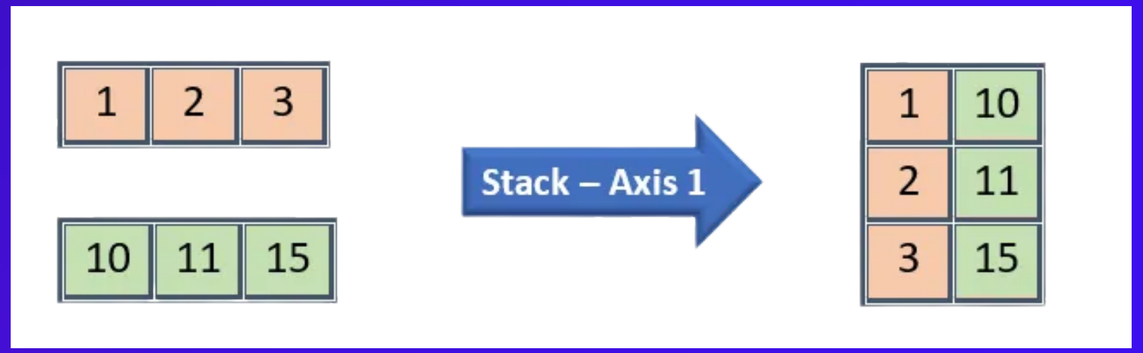
Join Arrays

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


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


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

print(z)
```

Join Arrays

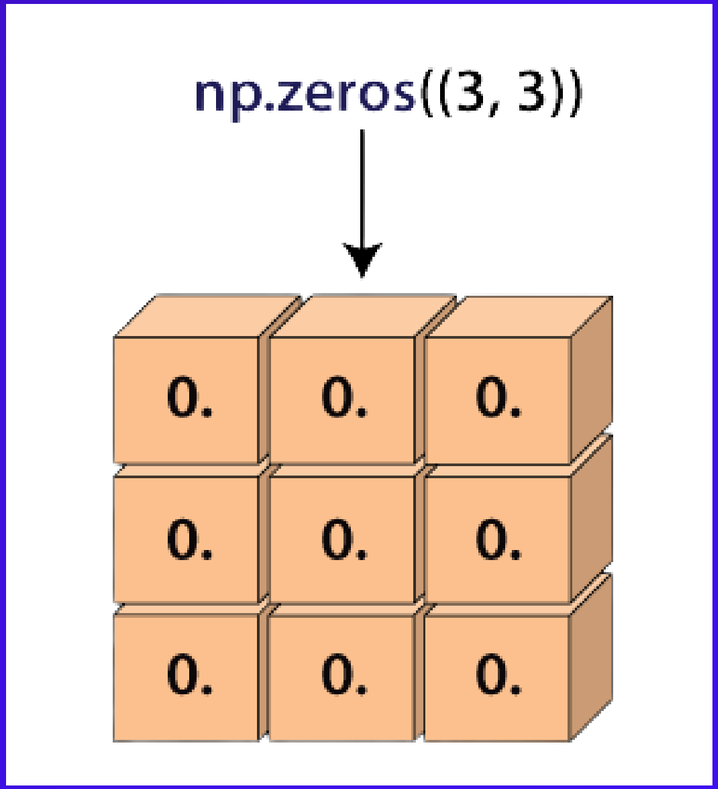
```
x = np.array([[1,2,3]])


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


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


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





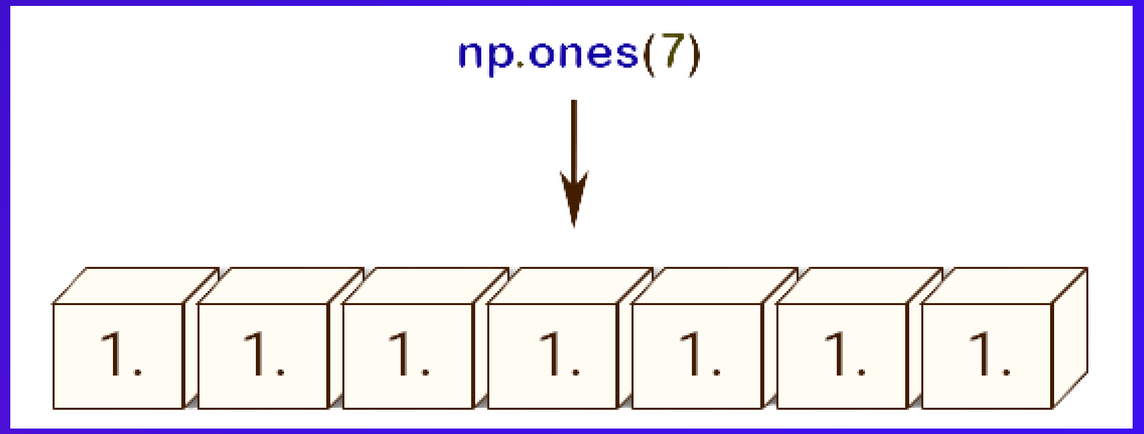
Numpy Zeros

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



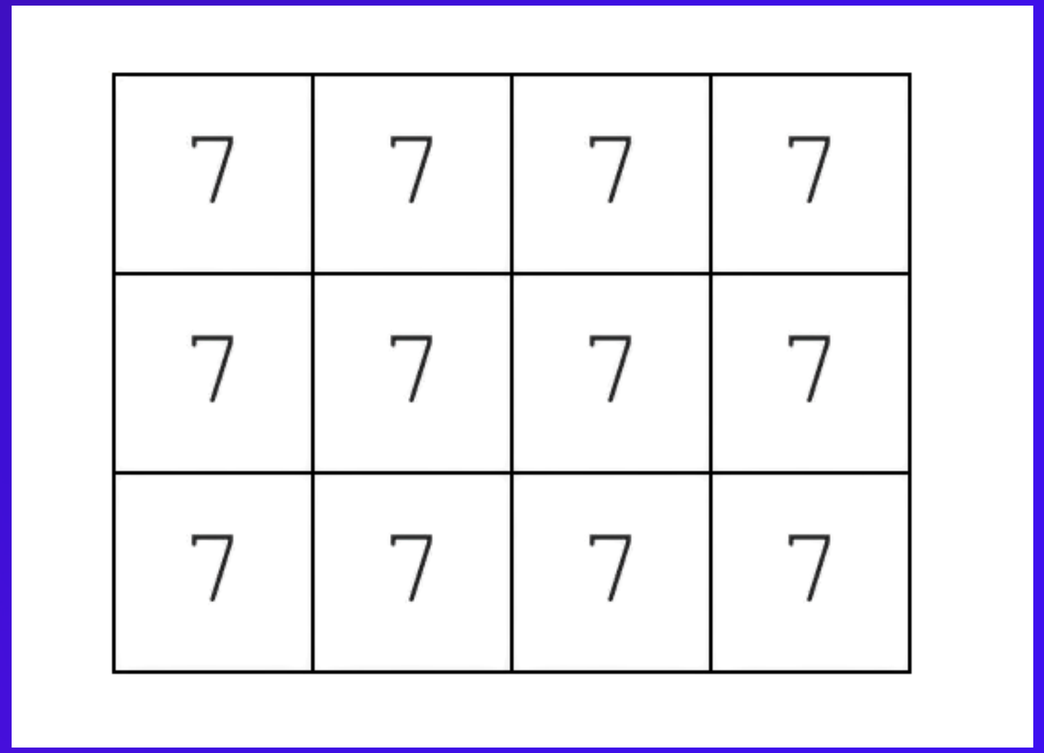
Numpy Ones

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



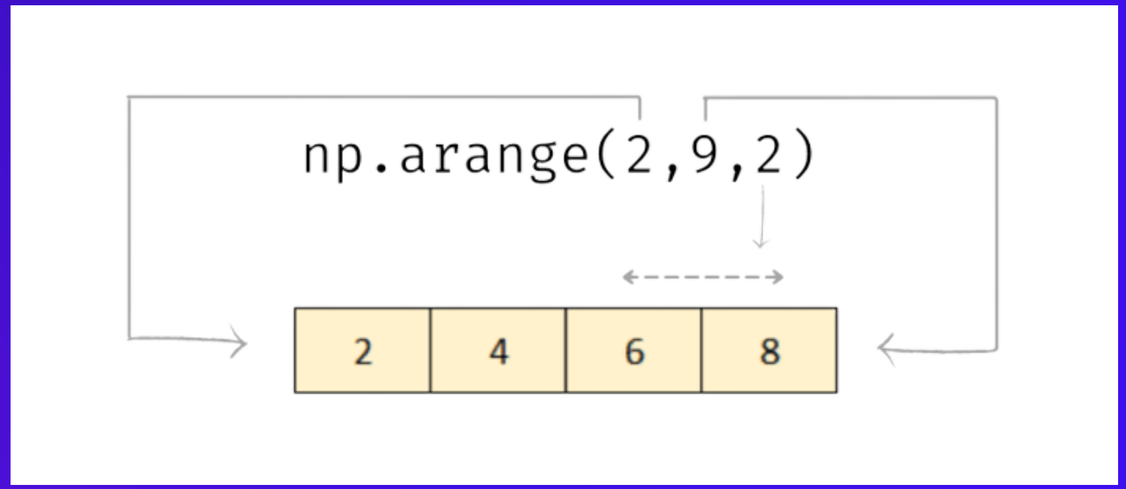
Numpy Full

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



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)
```

Thank you