```
import numpy as np
```

## Creating a numpy array

```
# Create an Numpy array from python list
x = [1,2,3]
arr = np.array(x)
print(arr)
```

```
[1 2 3]
```

```
# Creating a Numpy array having dimension 2*3 using a 2 dimensional python list
arr = np.array([[1,2,3],[4,5,6]])
print(arr)
```

```
[[1 2 3]
 [4 5 6]]
```

```
# Creating a Numpy array having dimension 2*3 using a python list and the reshape method
arr = np.array([1,2,3,4,5,6]).reshape(2,3)
print(arr)
```

```
[[1 2 3]
 [4 5 6]]
```

```
# Creating a Numpy array of dimension 2*3 having all values set to 0
# Here the parameter passed to the zeros method is a tuple
arr = np.zeros((2,3))
print(arr)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
# Creating a Numpy array of dimension 2*3 having all values set to 1
# Here the parameter passed to the ones method is a tuple
arr = np.ones((2,3))
print(arr)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
# Creating a Numpy array of dimension 2*3 having random values
# Here the parameter is directly passed to the rand method (not as a tuple, like in 'ones' and 'zeros')
# np.random.seed(10)
# Uncomment the above line for consistent results. Seed ensures that you get
# the same result everytime you execute a random operation
arr = np.random.rand(2,3)
print(arr)
```

```
# creating Numpy array using a range operation similar to python 'range' method
# np.arange(10) is same as np.arange(0,10)
arr = np.arange(10)
print(arr)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
# Total number of integers in arange(10) is 10 (0-9)
# and trying to reshape into 3,3 which needs only 9 elements results in an error
# When reshaping total number of elements should be equal to elements required in reshaped array
# np.arange(10) is same as np.arange(0,10)
arr = np.arange(10).reshape(3,3)
print(arr)
# If the above two lines are commented, the code below runs fine.
arr1 = np.arange(9).reshape(3,3)
print(arr1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-09f95c56aaa2> in <cell line: 5>()
      3 # When reshaping total number of elements should be equal to elements required in
reshaped array
      4 # np.arange(10) is same as np.arange(0,10)
----> 5 arr = np.arange(10).reshape(3,3)
      6 print(arr)
      7 # If the above two lines are commented, the code below runs fine.

ValueError: cannot reshape array of size 10 into shape (3,3)
```

SEARCH STACK OVERFLOW

## Basic operations that uses broadcasting

```
# Multiplication by 2 - Multiplies each element in the array by 2
arr = np.arange(9).reshape(3,3)
result = arr * 2
print(arr)
print("-"*30) #Prints 30 '-' as a separator for arr and result value.
print(result)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
------------------------------
[[ 0  2  4]
 [ 6  8 10]
 [12 14 16]]
```

```
# Multiplication by array having non similar dimension - Multiplies each element in the array by [2,3,5
# index of the element. ie: all elements in 1st column gets multiplied by 2, 2nd column elements gets m
```

```
# and elements in 3rd column gets multiplied by 5
arr = np.arange(9).reshape(3,3)
result = arr * [2,3,5]
print(arr)
print("-"*30) #Prints 30 '-' as a separator for arr and result value.
print(result)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
------------------------------
[[ 0  3 10]
 [ 6 12 25]
 [12 21 40]]
```

```
# the same is true for all operations like addition, subtraction and division.
arr = np.arange(9).reshape(3,3)
result = arr + [2,3,5]
print(arr)
print("-"*30) #Prints 30 '-' as a separator for arr and result value.
print(result)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
------------------------------
[[ 2  4  7]
 [ 5  7 10]
 [ 8 10 13]]
```

```
arr = np.arange(9).reshape(3,3)
result = arr - [2,3,5]
print(arr)
print("-"*30) #Prints 30 '-' as a separator for arr and result value.
print(result)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
------------------------------
[[-2 -2 -3]
 [ 1  1  0]
 [ 4  4  3]]
```

```
arr = np.arange(9).reshape(3,3)
result = arr / [2,3,5]
print(arr)
print("-"*30) #Prints 30 '-' as a separator for arr and result value.
print(result)
# Notice that here the output is
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
------------------------------
[[0.         0.33333333 0.4       ]
```

```
[1.5        1.33333333 1.        ]
[3.         2.33333333 1.6       ]]
```