```
Numpy is a Python Package, It stands for Numerical Python (NumPy)

It is a library consisting of multi-dimensional array and a collection of routines for processing these arrays



Operations using Numpy:

You can perform all mathematical and logical operators on an array.

Forurier Transformation and modifications of shapes can be done

Operations related to Linear Algebra, you can also generate random numbers
```

In [ ]:

```

```

Installing Numpy

```
pip install numpy
```

In [ ]:

```

```

importing library

In [161]:

```python
import numpy as np
```

In [ ]:

```

```

Numpy - ndarray Object

```
One of the most important object defined in Numpy is an N-dimesional array type called ndarry.
It describes the collection of items of the same data type.
Item in the collection can be accessed using zero based indexing
```

In [ ]:

```

```

In [162]:

```
np.array()
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-162-e4f3b47dc252> in <module>
----> 1 np.array()

TypeError: array() missing required argument 'object' (pos 1)
```

```
Object : An objects represents the data being put while creating the numpy array.

dtype : Desired data type of array -- optional argument

copy : By default true (you can allow the array to be copied) -- optional argument

order: Order by column (C), order by row (F) or by any (by default) -- optional argument

subok => returns an array for numpy  -- optional argument

ndmin => specify the minimum dimension needed for the array.


```

how to declare numpy array

In [163]:

```
my_arr = np.array([1,2,3,4,5])  #declaring single dimension array
```

In [164]:

```
print(my_arr)
```

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

In [165]:

```
type(my_arr)
```

Out[165]:

```
numpy.ndarray
```

In [ ]:

```

```

In [166]:

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

In [167]:

```
print(mul_d)
```

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

In [ ]:

```

```

In [168]:

```
arr_1 = np.array([1,2,3,4],dtype=complex)
```

In [169]:

```
print(arr_1)
```

```
[1.+0.j 2.+0.j 3.+0.j 4.+0.j]
```

In [ ]:

```

```

numpy Data-Types

```
bool_ => Boolean True or False  (but it stores as a byte)
int_ => Default integer type (same as C : integer 64 or integer 32 bit)
int8 => Byte(-128 to 127)
int16 => Integer(-32768 to 32767)
int32 => Integer (-2147483648 to 2147483647)
int64 => Integer (-9223372036854775808 to 9223372036854775807)
uint8 => Unsigned Integer (0 to 255)
unit16 => Unsigned Integer (0 to 65535)
unit32 => Unsigned Integer (0 to 4294967295)
unit64 => Unsigned integer (0 to 18446744073709551615)
float_ => Shorthand for float64
float 16  => Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float 32 => Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float 64 => Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_ => Shorthand for complex128.
complex64 => Complex number, represented by two 32-bit floats
complex128 => Complex number, represented by two 64-bit floats
```

In [ ]:

```

```

Numpy - Array Attributes

ndarray.shape :

```
 This array attribute returns a tuple consisting of array dimesnion. It can also be used to resize the array
```

In [170]:

```
arr = np.array([
    [10,20,30,40,50],
    [60,70,80,90,100]
])
```

In [171]:

```
print(arr)
```

```
[[ 10  20  30  40  50]
 [ 60  70  80  90 100]]
```

In [172]:

```
arr.shape
```

Out[172]:

```
(2, 5)
```

In [ ]:

```

```

In [173]:

```
# resizing the array
```

In [174]:

```
arr2 = np.array([
    [1,2,3],
    [4,5,6]
])
```

In [175]:

```
arr2.shape = (3,2)
```

In [176]:

```
arr2
```

Out[176]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [177]:

```
arr2.shape
```

Out[177]:

```
(3, 2)
```

In [ ]:

```

```

In [178]:

```
arr2 = arr2.reshape(2,3)
```

In [179]:

```
arr2
```

Out[179]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [ ]:

```

```

np.arange

```
This gives us an array of evenly spaced numbers
```

In [180]:

```
num = np.arange(30)
```

In [181]:

```
print(num)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23
 24 25 26 27 28 29]
```

In [182]:

```
# to check the dimension

num.ndim
```

Out[182]:

```
1
```

In [183]:

```
num = num.reshape(5,6)
```

In [184]:

```
num.shape
```

Out[184]:

```
(5, 6)
```

In [185]:

```
num.ndim
```

Out[185]:

```
2
```

In [ ]:

```

```

In [186]:

```
random = np.arange(24)
```

In [187]:

```
random
```

Out[187]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23])
```

In [188]:

```
random.ndim
```

Out[188]:

```
1
```

In [189]:

```
res = random.reshape(2,4,3)
```

In [190]:

```
print(res)
```

```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
```

In [191]:

```
res.ndim
```

Out[191]:

```
3
```

In [192]:

```
res.shape
```

Out[192]:

```
(2, 4, 3)
```

numpy.itemsize

This array attribute returns the length of each element of array in bytes

In [ ]:

```

```

In [193]:

```
num = np.array([1,2,3,4,5],dtype=np.float32)
```

In [194]:

```
num
```

Out[194]:

```
array([1., 2., 3., 4., 5.], dtype=float32)
```

In [195]:

```
num.itemsize
```

Out[195]:

```
4
```

In [ ]:

```

```

In [196]:

```
num1 = np.array([1,2,3,4,5])
```

In [197]:

```
num1.itemsize
```

Out[197]:

```
8
```

In [ ]:

```

```

numpy.empty

```
It creates an uninitialized array of specified size and dtype
```

In [198]:

```
x = np.empty([3,2],dtype=int)
```

In [199]:

```
print(x)
```

```
[[94441984802016              0]
 [             0              0]
 [             0              0]]
```

In [200]:

```
x.shape
```

Out[200]:

```
(3, 2)
```

numpy.zeros

```
It creates an uninitialized arrays of 0's
```

In [201]:

```
zero_arr = np.zeros([3,2],dtype=int)
```

In [202]:

```
zero_arr
```

Out[202]:

```
array([[0, 0],
       [0, 0],
       [0, 0]])
```

In [ ]:

```

```

numpy.ones

In [203]:

```
ones = np.ones([3,3],dtype=np.int8)
```

In [204]:

```
ones
```

Out[204]:

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]], dtype=int8)
```

In [ ]:

```

```

numpy.arange

In [205]:

```
np.arange(10)
```

Out[205]:

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

In [206]:

```
np.arange(10,dtype=float)
```

Out[206]:

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

In [207]:

```
np.arange(10,30)
```

Out[207]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26,
       27, 28, 29])
```

In [208]:

```
np.arange(10,30,2)
```

Out[208]:

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28])
```

In [ ]:

```

```

np.linspace

In [209]:

```
np.linspace(1,10,20)
```

Out[209]:

```
array([ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.8947368
4,
        3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.2631578
9,
        5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.6315789
5,
        8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.
])
```

In [210]:

```
1-2.28
```

Out[210]:

```
-1.2799999999999998
```

In [211]:

```
2.28-3.57
```

Out[211]:

```
-1.29
```

In [ ]:

```

```

np.logspace

In [212]:

```
np.logspace(1,10,10)
```

Out[212]:

```
array([1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05, 1.e+06, 1.e+07, 1.e+08,
       1.e+09, 1.e+10])
```

In [213]:

```
np.logspace(1,10,num=10,base=2)
```

Out[213]:

```
array([   2.,    4.,    8.,   16.,   32.,   64.,  128.,  256.,  512.,
       1024.])
```

In [ ]:

```

```

Numpy - Indexing & Slicing Technique

In [214]:

```
arr = np.arange(10,21)
```

In [215]:

```
print(arr)
```

```
[10 11 12 13 14 15 16 17 18 19 20]
```

In [ ]:

```

```

slicing

In [216]:

```
arr
```

Out[216]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

In [217]:

```
arr[0]
```

Out[217]:

```
10
```

In [218]:

```
arr[4:10]
```

Out[218]:

```
array([14, 15, 16, 17, 18, 19])
```

In [219]:

```
arr[4:10:2]
```

Out[219]:

```
array([14, 16, 18])
```

In [ ]:

```

```

slicing muliti-dimesnional array

In [220]:

```
mul_d = np.array([
    [10,20,30],
    [40,50,60],
    [70,80,90]
])
```

In [221]:

```
print(mul_d)
```

```
[[10 20 30]
 [40 50 60]
 [70 80 90]]
```

In [222]:

```
mul_d[2]
```

Out[222]:

```
array([70, 80, 90])
```

In [223]:

```
mul_d[0][2]
```

Out[223]:

```
30
```

In [224]:

```
mul_d[0:2]
```

Out[224]:

```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [225]:

```
mul_d[0:2,0:2]
```

Out[225]:

```
array([[10, 20],
       [40, 50]])
```

In [226]:

```
mul_d[1:3,1:3]
```

Out[226]:

```
array([[50, 60],
       [80, 90]])
```

In [227]:

```
mul_d
```

Out[227]:

```
array([[10, 20, 30],
       [40, 50, 60],
       [70, 80, 90]])
```

In [228]:

```
mul_d[0:2,0:3:2]
```

Out[228]:

```
array([[10, 30],
       [40, 60]])
```

```
only column as an output
```

In [ ]:

```

```

In [229]:

```
mul_d[0:3,1:2]
```

Out[229]:

```
array([[20],
       [50],
       [80]])
```

In [230]:

```
arr1 = np.arange(100,200,10)
```

In [231]:

```
arr1 = np.array([100, 110, 120, 130, 140, 150, 160, 170, 180])
```

In [232]:

```
arr1 = arr1.reshape(3,3)
```

In [233]:

```
arr1
```

Out[233]:

```
array([[100, 110, 120],
       [130, 140, 150],
       [160, 170, 180]])
```

In [234]:

```
np.diag(arr1)
```

Out[234]:

```
array([100, 140, 180])
```

In [ ]:

```

```

In [ ]:

```

```

to check for condition for a value in array

In [235]:

```
arr1
```

Out[235]:

```
array([[100, 110, 120],
       [130, 140, 150],
       [160, 170, 180]])
```

In [236]:

```
arr1>120
```

Out[236]:

```
array([[False, False, False],
       [ True,  True,  True],
       [ True,  True,  True]])
```

In [237]:

```
arr1[arr1>120]
```

Out[237]:

```
array([130, 140, 150, 160, 170, 180])
```

In [ ]:

```

```

In [ ]:

```

```

Broadcasting

```
You can use any arithmetic operators inside the arrray for performing operations
```

In [238]:

```
a1 = np.array([1,2,3,4])
b1 = np.array([10,20,30,40])
```

In [239]:

```
c = a1 * b1
```

In [240]:

```
print(c)
```

```
[ 10  40  90 160]
```

In [ ]:

```

```

In [241]:

```
m1 = np.array([
    [0.0,0.0,0.0],
    [10.0,10.0,10.0],
    [20.0,20.0,20.0],
    [30.0,30.0,30.0]
])
```

In [242]:

```
m1
```

Out[242]:

```
array([[ 0.,  0.,  0.],
       [10., 10., 10.],
       [20., 20., 20.],
       [30., 30., 30.]])
```

In [243]:

```
m2  = np.array([1.0,2.0,3.0])
```

In [244]:

```
m2
```

Out[244]:

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

In [245]:

```
new = m1 + m2
```

In [246]:

```
new
```

Out[246]:

```
array([[ 1.,  2.,  3.],
       [11., 12., 13.],
       [21., 22., 23.],
       [31., 32., 33.]])
```

Iterating into an Array

In [247]:

```
arr = np.arange(0,60,5)
```

In [248]:

```
arr
```

Out[248]:

```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55])
```

In [249]:

```
arr = arr.reshape(3,4)
```

In [250]:

```
arr
```

Out[250]:

```
array([[ 0,  5, 10, 15],
       [20, 25, 30, 35],
       [40, 45, 50, 55]])
```

In [ ]:

```

```

In [251]:

```
for item in np.nditer(arr):
    print(item)
```

```
0
5
10
15
20
25
30
35
40
45
50
55
```

numpy array manupulation

```
reshape => gives a new shape to an array without changing the data
flat => A 1-D iterator over array
flatten => Returns a copy of the array collapsed in 1-D
ravel : returns a continous flattend array
```

In [252]:

```
a = np.arange(8)
```

In [253]:

```
a
```

Out[253]:

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

In [254]:

```
a.reshape(4,2)


```

Out[254]:

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

In [ ]:

```

```

In [255]:

```
# np.ndarray.flat
```

In [256]:

```
arr = np.arange(10,18).reshape(4,2)
print(arr)
```

```
[[10 11]
 [12 13]
 [14 15]
 [16 17]]
```

In [257]:

```
arr[0][0]
```

Out[257]:

```
10
```

In [258]:

```
arr.flat[2]
```

Out[258]:

```
12
```

In [259]:

```
arr = np.arange(10,18).reshape(4,2)

```

In [260]:

```
arr
```

Out[260]:

```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17]])
```

In [261]:

```
arr = arr.flatten()  #this converts multi-d array to single d
```

In [262]:

```
arr
```

Out[262]:

```
array([10, 11, 12, 13, 14, 15, 16, 17])
```

In [263]:

```
arr = np.arange(10,18).reshape(4,2)
```

In [264]:

```
arr
```

Out[264]:

```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17]])
```

In [265]:

```
arr.ravel()
```

Out[265]:

```
array([10, 11, 12, 13, 14, 15, 16, 17])
```

In [ ]:

```

```

Transpose

In [266]:

```
arr = np.arange(12).reshape(3,4)
```

In [267]:

```
arr
```

Out[267]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [268]:

```
np.transpose(arr)
```

Out[268]:

```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [ ]:

```

```

In [ ]:

```

```

numpy.concatenate

In [269]:

```
arr1 = np.array([[1,2],[3,4]])

arr2 = np.array([[5,6],[7,8]])
```

In [270]:

```
arr1
```

Out[270]:

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

In [271]:

```
arr2
```

Out[271]:

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

In [272]:

```
arr3 = np.concatenate((arr1,arr2))  #axis 0 => vertical stack
```

In [273]:

```
arr3
```

Out[273]:

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

In [274]:

```
np.concatenate((arr1,arr2),axis=1) #axis 1 => horizontal stack
```

Out[274]:

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

In [ ]:

```

```

In [275]:

```
arr3
```

Out[275]:

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

In [276]:

```
arr3.reshape(2,4)
```

Out[276]:

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

In [ ]:

```

```

horizontal & verticle stack

In [277]:

```
arr1
```

Out[277]:

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

In [278]:

```
arr2
```

Out[278]:

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

In [279]:

```
np.hstack((arr1,arr2))
```

Out[279]:

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

In [280]:

```
np.vstack((arr1,arr2))
```

Out[280]:

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

In [ ]:

```

```

np.split

In [281]:

```
arr1 = np.arange(9)
```

In [282]:

```
arr1
```

Out[282]:

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

In [283]:

```
arr2 = np.split(arr1,3)
```

In [284]:

```
arr2
```

Out[284]:

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

In [ ]:

```

```

In [285]:

```
new_arr = np.arange(16).reshape(4,4)
```

In [286]:

```
new_arr
```

Out[286]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [287]:

```
np.hsplit(new_arr,2)
```

Out[287]:

```
[array([[ 0,  1],
        [ 4,  5],
        [ 8,  9],
        [12, 13]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11],
        [14, 15]])]
```

In [ ]:

```

```

In [288]:

```
np.vsplit(new_arr,2)
```

Out[288]:

```
[array([[0, 1, 2, 3],
        [4, 5, 6, 7]]),
 array([[ 8,  9, 10, 11],
        [12, 13, 14, 15]])]
```

In [ ]:

```

```

np.append

In [289]:

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

In [290]:

```
a
```

Out[290]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [291]:

```
a.shape
```

Out[291]:

```
(2, 3)
```

In [292]:

```
np.append(a,[7,8,9])
```

Out[292]:

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

In [ ]:

```

```

In [293]:

```
np.append(a,[[7,8,9]],axis=0) #vertical stack
```

Out[293]:

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

In [294]:

```
np.append(a,[[10,20,30],[40,50,60]],axis=1)
```

Out[294]:

```
array([[ 1,  2,  3, 10, 20, 30],
       [ 4,  5,  6, 40, 50, 60]])
```

In [ ]:

```

```

np.insert

In [295]:

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

In [296]:

```
a
```

Out[296]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [297]:

```
a[0]
```

Out[297]:

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

In [298]:

```
a[1]
```

Out[298]:

```
array([3, 4])
```

In [299]:

```
a[2]
```

Out[299]:

```
array([5, 6])
```

In [300]:

```
np.insert(a,4,[900,800])
```

Out[300]:

```
array([  1,   2,   3,   4, 900, 800,   5,   6])
```

In [301]:

```
a
```

Out[301]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [302]:

```
a
```

Out[302]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [303]:

```
np.insert(a,1,11,axis=1)
```

Out[303]:

```
array([[ 1, 11,  2],
       [ 3, 11,  4],
       [ 5, 11,  6]])
```

In [304]:

```
np.insert(a,1,[11],axis=0)
```

Out[304]:

```
array([[ 1,  2],
       [11, 11],
       [ 3,  4],
       [ 5,  6]])
```

In [ ]:

```

```

np.delete

In [305]:

```
a = np.arange(12).reshape(3,4)
```

In [306]:

```
a
```

Out[306]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [307]:

```
np.delete(a,5)
```

Out[307]:

```
array([ 0,  1,  2,  3,  4,  6,  7,  8,  9, 10, 11])
```

In [308]:

```
np.delete(a,1,axis=1) #delete column which ever specified
```

Out[308]:

```
array([[ 0,  2,  3],
       [ 4,  6,  7],
       [ 8, 10, 11]])
```

In [309]:

```
np.delete(a,1,axis=0)  # delete the row which ever specified
```

Out[309]:

```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [310]:

```
a
```

Out[310]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [311]:

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

In [312]:

```
arr
```

Out[312]:

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

In [ ]:

```

```

In [ ]:

```

```

In [313]:

```
np.delete(arr,np.s_[5:9])
```

Out[313]:

```
array([ 1,  2,  3,  4,  5, 10])
```

In [ ]:

```

```

np.unique

In [314]:

```
num = np.array([5,2,6,2,7,5,6,8,2,9])
```

In [315]:

```
num
```

Out[315]:

```
array([5, 2, 6, 2, 7, 5, 6, 8, 2, 9])
```

In [316]:

```
np.unique(num)
```

Out[316]:

```
array([2, 5, 6, 7, 8, 9])
```

In [317]:

```
num1 = num.reshape(2,5)
```

In [318]:

```
num1
```

Out[318]:

```
array([[5, 2, 6, 2, 7],
       [5, 6, 8, 2, 9]])
```

In [319]:

```
np.unique(num1)
```

Out[319]:

```
array([2, 5, 6, 7, 8, 9])
```

In [320]:

```
num
```

Out[320]:

```
array([5, 2, 6, 2, 7, 5, 6, 8, 2, 9])
```

In [321]:

```
val,index = np.unique(num,return_index=True)
```

In [322]:

```
val
```

Out[322]:

```
array([2, 5, 6, 7, 8, 9])
```

In [323]:

```
index
```

Out[323]:

```
array([1, 0, 2, 4, 7, 9])
```

In [ ]:

```

```

Numpy Strings

In [ ]:

```

```

numpy.char.add()

In [324]:

```
np.char.add(['hello'],['world'])
```

Out[324]:

```
array(['helloworld'], dtype='<U10')
```

In [ ]:

```

```

numpy.char.multiply()

In [325]:

```
np.char.multiply('Python ',5)
```

Out[325]:

```
array('Python Python Python Python Python ', dtype='<U35')
```

In [ ]:

```

```

numpy.char.center()

In [326]:

```
np.char.center(['punit','amit','mark'],30,fillchar='*')
```

Out[326]:

```
array(['************punit*************', '*************amit***********
**',
       '*************mark*************'], dtype='<U30')
```

In [327]:

```
name.center(30)
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-327-fb81c9718fb8> in <module>
----> 1 name.center(30)

NameError: name 'name' is not defined
```

In [ ]:

```

```

numpy.char.capitalize()

In [ ]:

```

```

In [ ]:

```
np.char.capitalize('steve jobs')
```

In [ ]:

```

```

numpy.char.title()

In [ ]:

```
np.char.title('steve jobs')
```

In [ ]:

```

```

np.char.lower()

In [ ]:

```
np.char.lower(['HELLO','PYTHON'])
```

In [ ]:

```

```

np.char.upper()

In [ ]:

```
np.char.upper(['hello','python'])
```

In [ ]:

```

```

np.char.split()

In [ ]:

```
np.char.split('Python,Java,C,C++',sep=',')
```

In [ ]:

```

```

In [ ]:

```
np.char.splitlines('Hello\nWelcome TO\nPython')
```

In [328]:

```
import numpy as np
```

np.char.strip()

In [ ]:

```

```

In [330]:

```
name = " punit "
```

In [331]:

```
name.strip()
```

Out[331]:

```
'punit'
```

In [332]:

```
np.char.strip(['apple','admin','java'],'a')
```

Out[332]:

```
array(['pple', 'dmin', 'jav'], dtype='<U5')
```

In [333]:

```
np.char.strip([' apple ',' mango ',' banana '],' ')
```

Out[333]:

```
array(['apple', 'mango', 'banana'], dtype='<U8')
```

In [ ]:

```
d-m-d
y/m/d
```

np.char.join()

In [334]:

```
np.char.join(['-','/'],['dmy','ymd'])
```

Out[334]:

```
array(['d-m-y', 'y/m/d'], dtype='<U5')
```

In [ ]:

```

```

np.char.replace

In [335]:

```
np.char.replace('This is Python Numpy Session','Python','Data Science')
```

Out[335]:

```
array('This is Data Science Numpy Session', dtype='<U34')
```

In [ ]:

```

```

In [ ]:

```
```

Mathematical Functions with Numpy

Trignometric Functions

In [336]:

```
a = np.array([0,30,45,60,90])
```

In [337]:

```
a
```

Out[337]:

```
array([ 0, 30, 45, 60, 90])
```

In [338]:

```
np.sin(a)
```

Out[338]:

```
array([ 0.        , -0.98803162,  0.85090352, -0.30481062,  0.8939966
6])
```

In [ ]:

```
```

In [339]:

```
np.cos(a)
```

Out[339]:

```
array([ 1.        ,  0.15425145,  0.52532199, -0.95241298, -0.4480736
2])
```

In [ ]:

```
```

In [340]:

```
np.cosh(a)
```

Out[340]:

```
array([1.00000000e+00, 5.34323729e+12, 1.74671355e+19, 5.71003695e+25,
       6.10201647e+38])
```

In [ ]:

```

```

In [341]:

```
np.tan(a)
```

Out[341]:

```
array([ 0.        , -6.4053312 ,  1.61977519,  0.32004039, -1.9952004
1])
```

In [ ]:

```

```

In [342]:

```
a = np.array([1.0,5.55,123,0.567,25.532])

np.around(a,decimals=1)
```

Out[342]:

```
array([  1. ,   5.6, 123. ,   0.6,  25.5])
```

In [345]:

```
np.around(a)
```

Out[345]:

```
array([  1.,   6., 123.,   1.,  26.])
```

In [348]:

```
np.around(10,decimals=2)
```

Out[348]:

```
10
```

In [ ]:

```

```

np.power()

In [349]:

```
arr = np.array([10,100,1000])
```

In [351]:

```
arr
```

Out[351]:

```
array([  10,  100, 1000])
```

In [352]:

```
np.power(arr,2)
```

Out[352]:

```
array([    100,   10000, 1000000])
```

In [354]:

```
np.power(3,3)
```

Out[354]:

```
27
```

In [ ]:

```

```

add, substract ...

In [355]:

```
a = np.arange(9).reshape(3,3)
```

In [356]:

```
a
```

Out[356]:

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

In [357]:

```
b = np.array([3,3,3])
```

In [358]:

```
b
```

Out[358]:

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

In [359]:

```
np.add(a,b)
```

Out[359]:

```
array([[ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [ ]:

```

```

In [360]:

```
np.subtract(a,b)
```

Out[360]:

```
array([[-3, -2, -1],
       [ 0,  1,  2],
       [ 3,  4,  5]])
```

In [ ]:

```

```

In [361]:

```
np.multiply(a,b)
```

Out[361]:

```
array([[ 0,  3,  6],
       [ 9, 12, 15],
       [18, 21, 24]])
```

In [ ]:

```

```

In [362]:

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

Out[362]:

```
array([[0.        , 0.33333333, 0.66666667],
       [1.        , 1.33333333, 1.66666667],
       [2.        , 2.33333333, 2.66666667]])
```

In [365]:

```
0%3
```

Out[365]:

```
0
```

In [364]:

```
np.mod(a,b)
```

Out[364]:

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

In [ ]:

```

```

In [366]:

```
np.remainder(a,b)
```

Out[366]:

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

In [ ]:

```

```

In [368]:

```
arr = np.array([-5.6j,0.2j,11.,1+1j])
```

In [369]:

```
arr
```

Out[369]:

```
array([-0.-5.6j,  0.+0.2j, 11.+0.j ,  1.+1.j ])
```

In [370]:

```
np.real(arr)
```

Out[370]:

```
array([-0.,  0., 11.,  1.])
```

In [ ]:

```

```

In [371]:

```
np.imag(arr)
```

Out[371]:

```
array([-5.6,  0.2,  0. ,  1. ])
```

In [ ]:

```

```

Statistical functions

In [372]:

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

In [373]:

```
a
```

Out[373]:

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

In [375]:

```
np.amin(a) #the overall min val
```

Out[375]:

```
2
```

In [376]:

```
np.amin(a,axis=1) #row wise min value
```

Out[376]:

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

In [378]:

```
np.amin(a,axis=0) #col wise min value
```

Out[378]:

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

In [ ]:

```

```

In [379]:

```
a
```

Out[379]:

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

In [381]:

```
np.amax(a)  #the overall max value
```

Out[381]:

```
9
```

In [382]:

```
np.amax(a,axis=1) #row wise max value
```

Out[382]:

```
array([7, 8, 9])
```

In [384]:

```
np.amax(a,axis=0) #col wise
```

Out[384]:

```
array([8, 7, 9])
```

In [ ]:

```

```

np.percentile

In [385]:

```
a = np.array([[30,40,70],[80,20,10],[50,90,60]])
```

In [386]:

```
a
```

Out[386]:

```
array([[30, 40, 70],
       [80, 20, 10],
       [50, 90, 60]])
```

In [387]:

```
np.percentile(a,50,axis=1) #percentile in rows
```

Out[387]:

```
array([40., 20., 60.])
```

In [388]:

```
np.percentile(a,50,axis=0) #percentile in cols
```

Out[388]:

```
array([50., 40., 60.])
```

In [ ]:

```

```

np.median

In [389]:

```
a = np.array([[30,65,70],[80,95,10],[50,90,60]])
```

In [390]:

```
a
```

Out[390]:

```
array([[30, 65, 70],
       [80, 95, 10],
       [50, 90, 60]])
```

In [391]:

```
np.median(a)
```

Out[391]:

```
65.0
```

In [392]:

```
np.median(a,axis=0)
```

Out[392]:

```
array([50., 90., 60.])
```

In [393]:

```
np.median(a,axis=1)
```

Out[393]:

```
array([65., 80., 60.])
```

In [ ]:

```

```

np.mean

In [394]:

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

In [395]:

```
arr
```

Out[395]:

```
array([[1, 2, 3],
       [3, 4, 5],
       [4, 5, 6]])
```

In [396]:

```
arr.mean()
```

Out[396]:

```
3.6666666666666665
```

In [397]:

```
np.mean(arr,axis=0)
```

Out[397]:

```
array([2.66666667, 3.66666667, 4.66666667])
```

In [398]:

```
np.mean(arr,axis=1)
```

Out[398]:

```
array([2., 4., 5.])
```

In [ ]:

```

```

In [399]:

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

```
arr
```

In [401]:

```
np.std(arr)
```

Out[401]:

```
1.118033988749895
```

In [ ]:

```

```

In [402]:

```
np.var(arr)
```

Out[402]:

```
1.25
```

In [ ]:

```

```

np.copy

In [403]:

```
a = np.arange(6)
```

In [404]:

```
print(a)
```

```
[0 1 2 3 4 5]
```

In [405]:

```
print(id(a))
```

```
140000097062416
```

In [ ]:

```

```

In [406]:

```
b = a
```

In [407]:

```
b
```

Out[407]:

```
array([0, 1, 2, 3, 4, 5])
```

In [408]:

```
a
```

Out[408]:

```
array([0, 1, 2, 3, 4, 5])
```

In [409]:

```
id(a)
```

Out[409]:

```
140000097062416
```

In [410]:

```
id(b)
```

Out[410]:

```
140000097062416
```

In [411]:

```
b.shape = 3,2
```

In [412]:

```
b
```

Out[412]:

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [413]:

```
print(a)
```

```
[[0 1]
 [2 3]
 [4 5]]
```

In [1]:

```
import numpy as np
```

shallow copy

In [2]:

```
arr1 = np.arange(6).reshape(3,2)
```

In [3]:

```
arr1
```

Out[3]:

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [4]:

```
arr2 = arr1.view()
```

In [6]:

```
arr2
```

Out[6]:

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [418]:

```
arr1
```

Out[418]:

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [7]:

```
id(arr1)
```

Out[7]:

```
140463511639984
```

In [8]:

```
id(arr2)
```

Out[8]:

```
140463511640272
```

In [9]:

```
arr2.shape = 2,3
```

In [10]:

```
arr2
```

Out[10]:

```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [11]:

```
arr1
```

Out[11]:

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [ ]:

```

```

matrix functions

In [12]:

```
import numpy.matlib
```

In [13]:

```
np.matlib.empty((2,2))
```

Out[13]:

```
matrix([[4.65021761e-310, 0.00000000e+000],
        [6.93983419e-310, 6.93983419e-310]])
```

In [ ]:

```

```

In [14]:

```
np.matlib.zeros((3,2))
```

Out[14]:

```
matrix([[0., 0.],
        [0., 0.],
        [0., 0.]])
```

In [ ]:

```

```

In [15]:

```
np.matlib.ones((3,3))
```

Out[15]:

```
matrix([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
```

In [ ]:

```python

```

In [17]:

```python
np.matlib.eye(5,5) #identity matrix
```

Out[17]:

```
matrix([[1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]])
```

In [ ]:

```python

```

In [18]:

```python
np.matlib.rand(3,3) #generating matrix for random value
```

Out[18]:

```
matrix([[0.94041174, 0.80928486, 0.13746363],
        [0.73310047, 0.51393815, 0.13549796],
        [0.87785017, 0.89153906, 0.44744869]])
```

In [ ]:

```python

```

In [ ]:

```python

```

In [19]:

```python
a = [[1,0],[0,1]]
b = [[4,1],[2,2]]
```

In [20]:

```python
a
```

Out[20]:

```
[[1, 0], [0, 1]]
```

In [21]:

```python
b
```

Out[21]:

```
[[4, 1], [2, 2]]
```

In [22]:

```
np.matmul(a,b)
```

Out[22]:

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

In [ ]:

```

```

In [23]:

```
a = np.arange(8).reshape(2,2,2)
b = np.arange(4).reshape(2,2)
```

In [24]:

```
a
```

Out[24]:

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

       [[4, 5],
        [6, 7]]])
```

In [25]:

```
b
```

Out[25]:

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

In [26]:

```
np.matmul(a,b)
```

Out[26]:

```
array([[[ 2,  3],
        [ 6, 11]],

       [[10, 19],
        [14, 27]]])
```

In [ ]:

```

```

In [27]:

```
a1 = np.array([[1,2],[3,4]])

b1 = np.array([[11,12],[13,14]])
```

In [28]:

```
np.dot(a1,b1)
```

Out[28]:

```
array([[37, 40],
       [85, 92]])
```

In [1]:

```

```

np.sort

In [2]:

```
arr = np.array([[3,4],[9,1]])
```

In [3]:

```
arr
```

Out[3]:

```
array([[3, 4],
       [9, 1]])
```

In [7]:

```
np.sort(arr) #sorting on the basis of row
```

Out[7]:

```
array([[3, 4],
       [1, 9]])
```

In [8]:

```
np.sort(arr,axis=0) #sorting on the basis of columns
```

Out[8]:

```
array([[3, 1],
       [9, 4]])
```

In [9]:

```
np.sort(arr,axis=1) #sorting on the basis of rows
```

Out[9]:

```
array([[3, 4],
       [1, 9]])
```

In [ ]:

```

```

In [10]:

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

In [11]:

```
x
```

Out[11]:

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

In [13]:

```
y = np.argsort(x)
```

In [14]:

```
y
```

Out[14]:

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

In [ ]:

```

```

In [15]:

```
name = ('punit','raju','amit','ravi')
year = ('F.Y','S.Y','S.Y','F.Y')
```

In [20]:

```
res = np.lexsort((year,name))
```

In [21]:

```
[name[i]+ " "+year[i] for i in res]
```

Out[21]:

```
['amit S.Y', 'punit F.Y', 'raju S.Y', 'ravi F.Y']
```

In [ ]:

```

```

In [33]:

```
surname = ('Hertz','Galilei','Hertz')
firstname = ('Henrich','Galileo','Gustav')
```

In [32]:

```
val =  np.lexsort((firstname,surname))
```

In [34]:

```
[firstname[i] + " "+surname[i] for i  in val]
```

Out[34]:

```
['Galileo Galilei', 'Gustav Hertz', 'Henrich Hertz']
```

In [ ]:

```

```

In [54]:

```
a1 = [1,5,1,4,3,4,4]
b1 = [9,4,0,4,0,2,1]
```

In [55]:

```
a1.sort()
```

In [56]:

```
a1
```

Out[56]:

```
[1, 1, 3, 4, 4, 4, 5]
```

In [58]:

```
b1.sort()
```

In [59]:

```
b1
```

Out[59]:

```
[0, 0, 1, 2, 4, 4, 9]
```

In [61]:

```
res = np.lexsort((b1,a1))
```

In [62]:

```
[(a1[i],b1[i]) for i in res]
```

Out[62]:

```
[(1, 0), (1, 0), (3, 1), (4, 2), (4, 4), (4, 4), (5, 9)]
```

In [ ]:

```

```

In [79]:

```
a = [1,5,1,4,3,4,4]
b = [9,4,0,4,0,2,1]
```

In [81]:

```
res= np.lexsort((b,a))
```

In [77]:

```
res
```

Out[77]:

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

In [ ]:

```

```

In [82]:

```
[(a[i],b[i]) for i in res]
```

Out[82]:

```
[(1, 0), (1, 9), (3, 0), (4, 1), (4, 2), (4, 4), (5, 4)]
```

In [22]:

```
x = np.arange(9).reshape(3,3)
```

In [23]:

```
x
```

Out[23]:

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

In [28]:

```
y = np.where(x > 5)
```

In [29]:

```
y
```

Out[29]:

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

In [30]:

```
x[y]
```

Out[30]:

```
array([6, 7, 8])
```

In [ ]:

```

```

In [ ]:

```

```

np.extract

In [83]:

```
a1 = np.arange(9).reshape(3,3)
```

In [84]:

```
a1
```

Out[84]:

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

In [85]:

```
condition = np.mod(a1,2) == 0
```

In [86]:

```
condition
```

Out[86]:

```
array([[ True, False,  True],
       [False,  True, False],
       [ True, False,  True]])
```

In [87]:

```
np.extract(condition,a1)
```

Out[87]:

```
array([0, 2, 4, 6, 8])
```

In [ ]:

```

```