Basics

```
a = 90
print(type(a))
```

```
<class 'int'>
```

```
a = 90
b = 98.5
print(type(a))
print(type(b))
```

```
<class 'int'>
<class 'float'>
```

```
a = 90
b = 98.5
print(type(a))
print(type(b))
```

```
<class 'int'>
<class 'float'>
```

```
print((a+b))
```

```
188.5
```

```
num = "90"
type(num)
```

```
str
```

Membership

```
lst1 = [90,67,45,23]
print(90 not in lst1)
```

```
False
```

```
lst1 = [90,67,45,23]
print(90 in lst1)
```

```
True
```

Functions

```
def addition (a,b):
    return a+b

addition(45,67)
```

```
112
```

```
def taxcal (S,T):
    Tax = ((T/100)*S)
    return Tax

taxcal(50000, 10)
```

```
5000.0
```

Write a program for tax deduction: 1.if salary is less than 10000,apply 5% tax 2.salary is more than 10000 but less than 50000,apply 10% tax 3.Salary is more than 50000 but less than 2,00,000 apply 15% tax

1. If salary is more than

```
def td(s):
    if(s<=0):
        return "invlid"
    if(s<10000):
        return s*0.05
    elif(s>=10000 & s<50000):
        return s*0.1
    elif(s>=50000 & s<200000):
        return s*0.15
    elif(s>=200000):
        return s*0.2
    else:
        return "invalid"

td(-1)
```

```
'invlid'
```

```
td(100000)
```

```
10000.0
```

loops

```
w=[67,45,23,50]
h=[160,127,140,130]
for i,j in zip(w,h): #zip function which iss used to two more list value iterate
   print(i/(j**2))
```

```
0.0026171875
0.0027900055800111 6
0.0011734693877551 02
0.0029585798816568047
```

```
for i in range(len(w)):
    print((w[i])/(h[i]*h[i]))

0.0026171875
0.00279000558001116
0.001173469387755102
0.0029585798816568047
```

Numpy

```
list1=[1,2,3,4]
list2=[5,6,7,8]
print(list1+list2)

[1, 2, 3, 4, 5, 6, 7, 8]

import numpy as np
a1=np.array([1,2,3,4])
a2=np.array([5,6,7,8])
print(a1+a2)

[ 6  8 10 12]

arr1=np.zeros((2,3))#2,3 is dimention
print(arr1)

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

arr2=np.ones((2,3))
print(arr2)

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

arr3=np.eye(3)#this works as identity matrix
print(arr3)

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

arr4=np.array([[3,4,5],[9,5,0]])
print(arr4)
print(np.ndim(arr4))#which gives the dimention of the matrix
print(np.shape(arr4))#gives the shape of array

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

```
arr5= np.array([6,7,8,9,9,4,2,1])
arr5=arr5.reshape(4,2)#reshape the array but original size is not
modified, if we want modified we have to save/assign it.
arr5
```

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

```
arr6= np.array([6,7,8,9,9,4,2,1])
arr6.resize(4,2)# use to modify the original size not need to a assign

arr6
```

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

```
arr7=np.arange(10,50)#arange act which bulid an array between the
numbers
print(arr7)
print(np.shape(arr7))
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
(40,)
```

```
arr7=np.arange(10,50).reshape(8,5)#this is act as bulid the array with
dimnetion
print(arr7)
print(np.shape(arr7))
```

```
[[10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]
 [25 26 27 28 29]
 [30 31 32 33 34]
 [35 36 37 38 39]
 [40 41 42 43 44]
 [45 46 47 48 49]]
(8, 5)
```

```
ar1=np.arange(8,1001,8)#start ,stop,steps
print(ar1)
print(type(ar1))
```

```
[   8   16   24   32   40   48   56   64   72   80   88   96  104  112
  120  128  136  144  152  160  168  176  184  192  200  208  216  224
  232  240  248  256  264  272  280  288  296  304  312  320  328  336
  344  352  360  368  376  384  392  400  408  416  424  432  440  448
  456  464  472  480  488  496  504  512  520  528  536  544  552  560
  568  576  584  592  600  608  616  624  632  640  648  656  664  672
  680  688  696  704  712  720  728  736  744  752  760  768  776  784
  792  800  808  816  824  832  840  848  856  864  872  880  888  896
  904  912  920  928  936  944  952  960  968  976  984  992 1000]
<class 'numpy.ndarray'>
```

```
ar2=np.arange(7,701,7)#which gives the 7 multiplication upto the 700
(strat 7 , upto 701, step/multiplication
print(ar2)
```

```
[  7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119
126
 133 140 147 154 161 168 175 182 189 196 203 210 217 224 231 238 245
252
 259 266 273 280 287 294 301 308 315 322 329 336 343 350 357 364 371
378
 385 392 399 406 413 420 427 434 441 448 455 462 469 476 483 490 497
504
 511 518 525 532 539 546 553 560 567 574 581 588 595 602 609 616 623
630
 637 644 651 658 665 672 679 686 693 700]
```

```
ar3=np.linspace(2,8,6)#2--to--8 inbetween it generate the numbers by
some relation with them
print(ar3)
```

```
[2.  3.2 4.4 5.6 6.8 8. ]
```

```
ar4=np.array([[[1,2,3],[6,7,8]],[[4,5,2],[3,6,0]]])#each row have the
two groups
print(ar4)
print(np.shape(ar4))#group,row,column
print(np.ndim(ar4))#dimention of matrix 3
```

```
[[[1 2 3]
  [6 7 8]]

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

Matrix

```
m1=np.array([9,4,6,7]).reshape(2,2)#matrix form
m2=np.array([1,2,3,4]).reshape(2,2)
```

```
print("matrix 1: \n",m1)
print("matrix 2: \n",m2)
```

```
matrix 1:
 [[9 4]
 [6 7]]
matrix 2:
 [[1 2]
 [3 4]]
```

```
print(m1*m2)#which give only the idex of value mutliplication
```

```
[[ 9  8]
 [18 28]]
```

```
print(m1@m2)#which gives the matrix multipliaction
```

```
[[21 34]
 [27 40]]
```

```
print(m1.dot(m2))#the function which act as matrix multiplication
```

```
[[21 34]
 [27 40]]
```

```
print(m2)
print(np.linalg.inv(m2))#inverse of the matrix
```

```
[[1 2]
 [3 4]]
[[-2.   1. ]
 [ 1.5 -0.5]]
```

statistics

```
r1=np.array([90,45,34,16,23,12])
print(np.mean(r1))#which gives the average value of array
```

```
36.666666666666664
```

```
print(np.median(r1))#by arranging array values in ascending order and then select the median
```

```
28.5
```

```
print(np.std(r1))#whish give the standard value
```

```
26.278423764669668
```

```
print(np.var(r1))#which give the varience
```

```
690.5555555555557
```

Trignometricz

```
print(np.pi)

3.141592653589793

rad=[90,30,45]
for i in rad:
    print(np.sin(i))#which gives the radian values of it

0.8939966636005579
-0.9880316240928618
0.8509035245341184

deg=[np.pi/4,np.pi/2,np.pi/3]
for i in deg:
    print(np.sin(i))

0.7071067811865476
1.0
0.8660254037844386

import numpy as np
print(np.hypot(6,8))

10.0
```

Arthmatic operation

```
a=np.array([8,9,1])
b=np.array([2,5,8])
print(np.sum((a,b))) #sum of both array

33

print(np.cumsum(a))

[ 8 17 18]

c=np.array([[1,2,3],[6,7,3],[9,1,6]])#sum of array works like
1,1+6,1+6+9 = 1,7,16 as we see in output
print(np.cumsum(c,axis=0))#column wise axis =0

[[ 1  2  3]
 [ 7  9  6]
 [16 10 12]]

c=np.array([[1,2,3],[6,7,3],[9,1,6]])#sum of array works 1,1+2,1+2+3=
1,3,6 as we see in output
print(np.cumsum(c,axis=1))#row wise axis =1
```

```
[[ 1  3  6]
 [ 6 13 16]
 [ 9 10 16]]
```

```
print(np.prod((a,b)))#similarly we can do for productive
```

```
5760
```

```
print(np.cumprod(c))#product of array in order
```

```
[    1     2     6    36   252   756  6804  6804 40824]
```

```
print(np.cumprod(c,axis=0))#product of array in colunm wise
```

```
[[ 1  2  3]
 [ 6 14  9]
 [54 14 54]]
```

```
print(np.cumprod(c,axis=1))#product of array in row wise
```

```
[[  1   2   6]
 [  6  42 126]
 [  9   9  54]]
```

```
s1=np.array([90,23,40,12])
s2=np.array([10,2,11,5])
print(np.mod(s1,s2))
```

```
[0 1 7 2]
```

```
print(np.divmod(s1,s2))#1st arrray = qoucitent 2nd array=remainder
```

```
(array([ 9, 11,  3,  2]), array([0, 1, 7, 2]))
```

UFunc #universal function

```
A=np.array([56,78,12,32,111,109])
print(max(A))#gives max
```

```
111
```

```
A=np.array([56,78,12,32,111,"like"])
print(max(A))#comparing with ASCII values
```

```
like
```

```
A=np.array([56,78,12,32,111,109])
print(min(A))#gives the min value
```

```
12
```

sorting

```
B=np.array([90,12,45,1,89,98])
B.sort()#which does effect the original array by sorting
print(B)
```

```
[ 1 12 45 89 90 98]
```

```
C=np.array([90,12,45,1,89,98])
D=sorted(C)#which doesnot effect the original array by sorting by assinging it
print(C)
print(D)
```

```
[90 12 45  1 89 98]
[1, 12, 45, 89, 90, 98]
```

rounding

```
s2=np.array([9.1,-7.8])
print(np.ceil(s2))#which give roundup of greatest
```

```
[10. -7.]
```

```
print(np.floor(s2))#which give roundup of smallest
```

```
[ 9. -8.]
```

random module

```
import numpy.random as rd#which generates the random numbers
```

```
ran1=rd.rand()#range between 0 to 1
print(ran1)
```

```
0.8802052740717733
```

```
ran=rd.randint(5)#range between 0 to 5
print(ran)
```

```
4
```

```
ran3=rd.randint(5,size=(6))#range between 0 to 5 with  limits size
print(ran3)
```

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

```
ran3=rd.randint(5,size=(6,2))#range between 0 to 5 with limit size (group ,row )
print(ran3)
```

```
[[2 3]
 [4 2]
 [4 3]
 [0 3]
 [3 2]
 [4 4]]
```

```
ran3=rd.randint(5,size=(6,2,3))#range between 0 to 5 with limit size 
(group ,row,column )
print(ran3)
```

```
[[[1 4 1]
  [3 0 0]]

 [[3 4 1]
  [1 2 1]]

 [[4 0 3]
  [0 1 4]]

 [[2 2 0]
  [1 1 0]]

 [[3 3 3]
  [4 3 0]]

 [[3 1 1]
  [0 1 2]]]
```

stack

```
ar1=np.array([[9,4,23],[3,4,5]])
ar2=np.array([[8,1,2],[33,42,51]])
print(ar1)
print("\n")
print(ar2)
```

```
[[ 9  4 23]
 [ 3  4  5]]


[[ 8  1  2]
 [33 42 51]]
```

```
a1=np.hstack((ar1,ar2))#side by side print
print(a1)
```

```
[[ 9  4 23  8  1  2]
 [ 3  4  5 33 42 51]]
```

```
a1=np.vstack((ar1,ar2))#one top of another print
print(a1)
```

```
[[ 9  4 23]
 [ 3  4  5]
 [ 8  1  2]
 [33 42 51]]
```

```
ar5=np.arange(1,13).reshape(3,2,2)
print(ar5)
```

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

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

```
ar6=np.dstack(ar5)#which select the index/dimnetion  wise make it as row
print(ar6)
```

```
[[[1 5]
  [2 6]]

 [[3 7]
  [4 8]]]
```

```
ar36=np.arange(1,9).reshape(2,2,2)
print(ar36)
```

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

 [[5 6]
  [7 8]]]
```

```
ar=np.dstack(ar36)
print(ar)
```

```
[[[1 5]
  [2 6]]

 [[3 7]
  [4 8]]]
```

```
n=81
m=99
o=73
print(np.sqrt(n))#gives the square root of it
print(np.lcm(n,m))# gives LCM but only for two numbers
AA=[45,67,89]#for more numbers to get LCM & GCD use lcm.reduce with array
```

```
print(np.lcm.reduce(AA))
print(np.gcd.reduce(AA))
print(np.gcd(n,m))#gcd only for two numbers
```

```
9.0
891
268335
1
9
```

```
ab=np.array([0,-5,7,23])
print(np.absolute(ab))#give modulus of number
```

```
[ 0  5  7 23]
```

logorithm

```
n=45
print(np.log(n))#gives log numbers --- logE x
print(np.log10(n))#---- log10 x
print(np.log2(n))#---- log2 x
```

```
3.8066624897703196
1.6532125137753437
5.491853096329675
```

set

```
s1=np.array([9,3,5,2,1])
s2=np.array([4,5,2,1,3])
print(s1,"\n")
print(s2,"\n")
print(np.union1d(s1,s2),"\n")#by removing duplicate element combines all
print(np.intersect1d(s1,s2),"\n")#by selecting the commmon element in both array
print(np.setdiff1d(s1,s2))#which gives s1-s2
```

```
[9 3 5 2 1]

[4 5 2 1 3]

[1 2 3 4 5 9]

[1 2 3 5]

[9]
```

Search

```
c1=np.array([44,33,12,67,19])
index=np.where(c1%2==0)
print(index)
```

```
(array([0, 2], dtype=int64),)
```

```
c2=np.array([45,33,21,50,60,15])
index1=np.where((c2%3==0) & (c2%5==0))
print(index1)
```

```
(array([0, 4, 5], dtype=int64),)
```