```
num1=5
num2=20
sum=num1+num2
print(sum)
```

```
25
```

```
print(type(num1))
```

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

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

```
<class 'str'>
```

```
list1=[20,45,61,52]
print(20 in list1)
```

```
True
```

```
def taxcla(s):
    if(s<=0):
        print("invalid")
    elif(s>0 and s<=10000):
        tax =0.05*s
        return tax
    elif(s>10000 and s<=50000):
        return 0.1*s
    elif(s>50000 and s<=200000):
        return 0.15*s
    elif(s>200000):
        return 0.20*s


taxcla(10000)
```

```
500.0
```

```
taxcla(300000)
```

```
60000.0
```

```
taxcla(90000)
```

```
13500.0
```

Loops

w=[67,45,23,50] h=[160,127,140,187] output:bmw=w/h^2

```
w=[67,45,23,50]      #direct value
h=[1.60,1.27,1.40,1.87]
for i,j in zip(w,h):
        bmi=(i/(j*j))
        print(bmi)
```

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

```
for i in range(len(w)):   #using index value -->0,1,2,3
    print(w[i]/(h[i]*h[i]))
```

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

Numpy

```
list1=[9,56,12,34]
list2=[78,45,55,67]
print(list1+list2)
```

```
[9, 56, 12, 34, 78, 45, 55, 67]
```

```
import numpy as np
arr1=np.array([90,56,12,34])
arr2=np.array([78,45,55,67])
print(arr1+arr2)
```

```
[168 101  67 101]
```

```
arr1=np.zeros((2,3))
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))
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))
print(np.shape(arr4))

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

arr5=np.array([6,7,8,9,9,4,2,1])

arr5=arr5.reshape(2,4)   #reshape array does not happen to verginal
array change occur in the saparate variable

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)   #resize it does change the verinal array

arr6

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

arr6=np.arange(10,50).reshape(8,5)
print(arr6)
print(np.shape(arr6))

[[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)

arr7=np.arange(8,1001,8)   #start,stop,step multiple of 8 upto 1000
print(arr7)
```

```
[   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]
```

```
print(type(arr7))
```

```
<class 'numpy.ndarray'>
```

```
arr8=np.arange(7,701,7)     #multiple of 7
print(arr8)
```

```
[  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]
```

```
import numpy as np
arr9=np.linspace(2,8,6)
print(arr9)                # 2, %,% ,%, %,8 #generate 6 evenly spaced
numbers from 2 to 8
```

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

matrix

```
mat1=np.array([9,4,6,7]).reshape(2,2)
mat2=np.array([1,2,3,4]).reshape(2,2)
print("marix 1:\n",mat1)
print("marix 2:\n",mat2)
```

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

```
arr10=np.array([[[1,2,3],[6,7,8]],[[4,5,2],[3,6,0]]])    # 3D array
print(arr10)
print(np.shape(arr10))     #ndim-dimension
print(np.ndim(arr10))      # group ,row,column
```

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

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

```
print(mat1*mat2)     #index wise multipliction occurs
```

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

```
print(mat1.dot(mat2))   #matrix multipliction
```

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

```
print(mat1@mat2);      #matrix multipliction
```

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

```
print(np.linalg.inv(mat1))     #inverse matrix
```

```
[[ 0.17948718 -0.1025641 ]
 [-0.15384615  0.23076923]]
```

```
print(np.linalg.inv(mat2))      # determinante |a|=ad-cb
```

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

statistics

```
arr1 = np.array([90,45,34,16,23,12])          #mean=average of totalnumbers
print(np.mean(arr1))                          #standerd deviation=sqrt(varince)

#varince= xi-u/n
```

```
36.666666666666664
```

```
print(np.std(arr1))        # standerd devaition=sqrt(varince)
```

```
26.278423764669668
```

```
print(np.var(arr1))        #varince =xi-u/n
```

```
690.5555555555557
```

```
print(np.pi)
```

```
3.141592653589793
```

trignometry

```
rad=[90,30,45]
for i in rad:
    print(np.sin(i))
```

```
0.8939966636005579
-0.9880316240928618
0.8509035245341184
```

```
import numpy as np
deg=[np.pi/4,np.pi/2,np.pi/3]
for i in deg:
     print(np.sin(1))
```

```
0.8414709848078965
0.8414709848078965
0.8414709848078965
```

```
print(np.hypot(6,8))
```

```
10.0
```

arthamatic operation

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

```
33
```

```
print(np.cumsum(a))
```

```
[ 8 17 18]
```

```
#axix=0 is column wise addition

#axis=1 is row wise addition
#this are the cunulative addition
```

```
#axis =2 is add the group wise addition
#product of means muliply the all numbers
#cumprod means multiply the one by one until last
#cumprod means multiply the row wise we give axis=0
#smilarly axis=1

c=np.array([[1,2,3],[6,7,3],[9,1,6]])
print(np.cumsum(c,axis=0))
```

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

```
print(np.cumsum(c,axis=1))
```

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

```
print(np.prod((a,b)))  #multiply the all the numbers
```

```
5760
```

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

```
[   8   72   72  144  720 5760]
```

```
print(np.cumprod(c))   #multiply the one by one then it will return in
single line
```

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

```
print(np.cumprod(c,axis=0))  #column
```

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

```
print(np.cumprod(c,axis=1))  #row
```

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

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

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

```
print(np.divmod(s1,s2))   #
```

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

```
A=np.array([10,27,200,111,109])
print(max(A))
```

```
200
```

```
h=np.array([10,27,200,111,109,"like"])   #because of the print asci values
print(max(h))
```

```
like
```

```
print(min(A))
```

```
10
```

sorting

```
#sort it will modify the veriginal array
#sorted array it will not affert on original array ,when the varible to the saparate then it will change

B=np.array([90,12,45,1,89,98])
B.sort()
print(B)
```

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

```
c=np.array([90,12,45,1,89,98])
D=sorted(c)
print(D)
print(c)
```

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

```
print(c)
```

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

rounding

```
s2=np.array([9.1,-7.8])   #ceil is taken the round fig by bigger value
print(np.ceil(s2))
```

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

```
print(np.floor(s2))      #floor is taken the smaller the value dependence on the value given
```

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

Random module

```
random number will generate b/w the 1 to 0
randint wil generate the random numbers  when the we will given the
limite b/w the the 0 to  limit
```

```
import numpy.random as rd
```

```
ran1=rd.rand(2)     # 0 to 1
print(ran1)
```

```
[0.87346139 0.41591613]
```

```
ran2=rd.randint(5)    # b/w 0 to 5 because limit
print(ran2)
```

```
3
```

```
rad3=rd.randint(5,size=(6))       # limit ,size
print(rad3)
```

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

```
rad4=rd.randint(5,size=(6,2,3))     #limit,size(g,r,c)
print(rad4)
```

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

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

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

 [[0 0 4]
  [2 0 2]]

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

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

stack

```
#hstack arranged in side by side
#vstack is the virticle array
#dstack is the change by the colume determine by the no of groups in
the array similarly transpose

arr1=np.array([[9,4,23],[3,4,6]])
arr2=np.array([[8,1,2],[33,42,51]])
print(arr1)
print("\n")
print(arr2)
```

```
[[ 9  4 23]
 [ 3  4  6]]


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

```
arr3=np.hstack((arr1,arr2))
print(arr3)
```

```
[[ 9  4 23  8  1  2]
 [ 3  4  6 33 42 51]]
```

```
arr4=np.vstack((arr1,arr2))
print(arr4)
```

```
[[ 9  4 23]
 [ 3  4  6]
 [ 8  1  2]
 [33 42 51]]
```

```
arr5=np.arange(1,13).reshape(3,2,2)
print(arr5)
```

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

 [[ 5  6]
  [ 7  8]]

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

```
arr6=np.dstack((arr5))
print(arr6)
```

```
[[[ 1  5  9]
  [ 2  6 10]]
```

```
 [[ 3  7 11]
  [ 4  8 12]]]
```

```
num1=81
num2=99
num3=78
print(np.sqrt(num1))
```

```
9.0
```

```
print(np.lcm(num1,num2))
```

```
891
```

```
print(np.gcd(num1,num2))
```

```
9
```

```
AA=[45, 67,89]
print(np.lcm.reduce(AA))   #reduce the lcm  it is common to all we apply to all
```

```
268335
```

```
print(np.gcd.reduce(AA))
```

```
1
```

```
AB=np.array([0,-5,20,-23])    # it will remove the negative values
print(np.absolute(AB))
```

```
[ 0  5 20 23]
```

logarithums

```
n=45
print(np.log(n))    #natural log
```

```
3.8066624897703196
```

```
print(np.log10(n))
```

```
1.6532125137753437
```

```
print(np.log2(n))
```

```
5.491853096329675
```

set function

```
s1=np.array([9,5,2,1,3])
s2=np.array([4,5,2,1,3])
print(s1 ,"\n")
print(s2)
```

```
[9 5 2 1 3]

[4 5 2 1 3]
```

```
print(np.union1d(s1,s2))
```

```
[1 2 3 4 5 9]
```

```
print(np.intersect1d(s1,s2))
```

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

```
print(np.setdiff1d(s1,s2))    # only take the uniqe elements  s1-s2
```

```
[9]
```

search

```
col1=np.array([44,33,67,12,53])
index=np.where(col1%2==0)
print(index)
```

```
(array([0, 3], dtype=int64),)
```

```
col1=np.array([45,33,67,12,60,15])
index=np.where((col1%5==0) & (col1%3==0))
print(index)
```

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