```
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
```

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

taxcal(50000, 10)
```

```
5000.0
```

```
list1=[]

def tax(S):
    if(S<10000 and S>0):
        tax=0.05*S
        return tax
    elif(S>=10000 and S<50000):
        tax=0.1*S
        return tax
    elif(S>=50000 and S<200000):
        tax=0.15*S
        return tax
    elif(S>=200000):
        tax=0.2*S
        return tax
    else:
        return "invalid"

tax(60000)
```

```
9000.0
```

```
tax(50000)
```

```
7500.0
```

```
tax(-1)
```

```
'invalid'
```

```
tax(0)
```

```
'invalid'
```

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

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

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

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

Numpy

```
l1=[220, 24,365,677]
l2=[35,665,89,0]
print(l1+l2)
```

```
[220, 24, 365, 677, 35, 665, 89, 0]
```

```
import numpy as np
a1=np.array([220, 24,365,677])
a2=np.array([35,665,89,0])
print(a1+a2)
```

```
[255 689 454 677]
```

```
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([[3,4,7,9],[1,9,3,6]])
print(arr4)
print(np.ndim(arr4))
```

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

```
print(np.shape(arr4))
```

```
(2, 4)
```

```
arr5=np.array([3,4,7,9,1,9,3,6])
arr5.reshape(4,2)
```

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

```
arr6=arr5.resize(2,4)
print(arr6)
```

```
None
```

```
arr7=np.arange(10,50).reshape(8,5)
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)
```

```
arr8=np.arange(8,1001,8)
print(arr8)
print(type(arr8))
```

```
[   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'>

seven=np.arange(7,701,7)
print(seven)
print(type(seven))

[  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]
<class 'numpy.ndarray'>

a=np.linspace(2,8,6)  #generates 6 values with same difference between
2 and 8
print(a)

[2.  3.2 4.4 5.6 6.8 8. ]

a1=np.array([ [[1,2,3],[4,5,6]],[[7,8,9],[1,7,4]] ])
print(a1)
print(np.shape(a1))   #group, row, column
print(np.ndim(a1))

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

 [[7 8 9]
  [1 7 4]]]
(2, 2, 3)
3

m1=np.array([9,4,6,7]).reshape(2,2)
m2=np.array([1,2,3,4]).reshape(2,2)
print("matrix1\n",m1)
print("matrix2\n",m2)
```

```
matrix1
 [[9 4]
 [6 7]]
matrix2
 [[1 2]
 [3 4]]
```

matrix operation

```
print(m1*m2)

[[ 9  8]
 [18 28]]

print(m1.dot(m2))

[[21 34]
 [27 40]]

print(m1@m2)

[[21 34]
 [27 40]]

print(np.linalg.inv(m1))

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

statistics

```
a1=np.array([23,45,78,90,43,21])
print(np.mean(a1))

50.0

print(np.median(a1))

44.0

print(np.std(a1))

25.9101009904117

print(np.var(a1))

671.3333333333334
```

trignometry

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

```
3.141592653589793
```

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

```
0.8939966636005579
-0.9880316240928618
0.8509035245341184
```

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

```
-0.4480736161291701
0.15425144988758405
0.5253219888177297
```

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

```
0.7071067811865476
1.0
0.8660254037844386
```

arith,etic operations

```
a=np.array([8,3,4])
b=np.array([2,1,4])
print(np.sum((a,b)))
```

```
22
```

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

```
[ 8 11 15]
```

```
c=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(np.cumsum(c,axis=0))   #column
```

```
[[ 1  2  3]
 [ 5  7  9]
 [12 15 18]]
```

```
c=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(np.cumsum(c,axis=1))   #row
```

```
[[ 1  3  6]
 [ 4  9 15]
 [ 7 15 24]]

print(np.prod((a,b)))

768

print(np.cumprod(c))

[     1      2      6     24    120    720   5040  40320 362880]

print(np.cumprod(c,axis=0))

[[  1   2   3]
 [  4  10  18]
 [ 28  80 162]]

print(np.cumprod(c,axis=1))

[[  1   2   6]
 [  4  20 120]
 [  7  56 504]]

a=np.array([81,32,74])
b=np.array([21,14,41])
print(np.mod(a,b))

[18  4 33]

print(np.divmod(a,b))

(array([3, 2, 1]), array([18,  4, 33]))

n1=54
n2=21
print(np.sqrt(n1))

7.3484692283495345

print(np.lcm(n1,n2))

378

print(np.gcd(n1,n2))

3

a=np.array([81,32,74])
print(np.lcm.reduce(a))

95904

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

```
1
```

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

```
3.8066624897703196
```

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

```
1.6532125137753437
```

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

```
5.491853096329675
```

UNIVERSAL FUN

```
a=np.array([12,35,87,643,3,6,8,1236])
print(max(a))
```

```
1236
```

```
print(min(a))
```

```
3
```

```
b=np.array([12,35,87,643,3,6,8,1236])
b.sort()
print(b)
```

```
[   3    6    8   12   35   87  643 1236]
```

```
c=np.array([12,35,87,643,3,6,8,1236])
d=sorted(c)
print(c)
print(d)
```

```
[  12   35   87  643    3    6    8 1236]
[3, 6, 8, 12, 35, 87, 643, 1236]
```

```
s2=np.array([1.23,-7.23])
print(np.ceil(s2))
```

```
[ 2. -7.]
```

```
s2=np.array([1.23,-7.23])
print(np.floor(s2))
```

```
[ 1. -8.]
```

random module

```
import numpy.random as rd

ran1=rd.rand(2)
print(ran1)
```

```
[0.71363097 0.20945292]
```

```
ran2=rd.randint(5)  #0 to 5 random numbers
print(ran2)
```

```
4
```

```
ran3=rd.randint(5,size=(6))  #limit ,size
print(ran3)
```

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

```
ran4=rd.randint(5,size=(6,2,3))  #limit ,size(g,r,c)
print(ran4)
```

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

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

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

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

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

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

stack

```
ar1=np.array([[1,2,34],[5,67,88]])
ar2=np.array([[15,32,39],[58,37,81]])
print(ar1)
print("\n")
print(ar2)
```

```
[[ 1  2 34]
 [ 5 67 88]]


[[15 32 39]
 [58 37 81]]
```

```
ar3=np.hstack((ar1,ar2))  #side by side
print(ar3)
```

```
[[ 1  2 34 15 32 39]
 [ 5 67 88 58 37 81]]
```

```
ar3=np.vstack((ar1,ar2))  #one top of another
print(ar3)
```

```
[[ 1  2 34]
 [ 5 67 88]
 [15 32 39]
 [58 37 81]]
```

```
ar4=np.arange(1,13).reshape(3,2,2)
print(ar4)
```

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

 [[ 5  6]
  [ 7  8]]

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

```
ar5=np.dstack(ar4)   #rows are arranged as colums from all group
print(ar5)
```

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

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

```
col1=np.array([44,33,12,83,55])
index=np.where(col1%2==0)
print(index)
```

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

```
col2=np.array([50,33,21,15,60,45])
index1=np.where(col2%3==0)
index2=np.where(col2%5==0)
print(index1)
print(index2)
```

```
(array([1, 2, 3, 4, 5], dtype=int64),)
(array([0, 3, 4, 5], dtype=int64),)
```

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

```
(array([3, 4, 5], dtype=int64),)
```