basic

```
a=60
b=6.5
print(type(a))
print(type(b))
```

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

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

```
66
```

```
num="23"
type(num)
```

```
str
```

```
13//2
```

```
6
```

```
13/2
```

```
6.5
```

```
def taxcl(s,t):
    tax=(t/100*s)
    return tax


taxcl(10000,10)
```

```
1000.0
```

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 200000,apply 15% tax 4.If salary is more than 2lakhs,apply 20% tax.

```
def taxcal(Sal):
    if (Sal>0 and Sal<10000):
        return 0.05*Sal
    elif (Sal<=10000 and Sal>50000):
        return 0.1*Sal
    elif (Sal<=50000 and Sal>200000):
        return 0.15*Sal
    elif (Sal>=200000):
        return 0.20*Sal
    else:
        return "INVALID"

taxcal(200000)
```

```
40000.0
```

```
taxcal(-40)
```

```
'INVALID'
```

Loops

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

```
w = [67,45,23,50]
h = [1.6,1.27,1.40,1.87]
for i,j in zip(w,h):
    print(i / (j*j))
```

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

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

```
26.171874999999996
27.900055800111602
11.734693877551022
14.298378563870855
```

Numpy

```
lst1 = [90,56,12,34]
lst2 = [78,45,55,67]
print(lst1+lst2)
```

```
[90, 56, 12, 34, 78, 45, 55, 67]
```

```python
import numpy as np
ar1 =  np.array([90,56,12,34])
ar2 = np.array([78,45,55,67])
print(ar1+ar2)
```

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

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

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

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

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

```python
arr3 = np.eye(3)
print(arr3)
```

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

```python
arr4 = np.array([[4,5,6],[9,5,0]])
print(arr4)
print(np.ndim(arr4))
print(np.shape(arr4))
```

```
[[4 5 6]
 [9 5 0]]
2
(2, 3)
```

```python
arr5 = np.array([9,4,5,6,9,7,4,3])
```

```python
arr5.reshape(4,2)
```

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

```python
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(7,701,7)
print(arr7)

[  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]

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'>

arr9 = np.array( [ [[1,2,3],[6,7,8]],[[4,5,2],[3,6,0]] ])
print(arr9)
print(np.shape(arr9))
print(np.ndim(arr9))

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

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

```
arr10 = np.linspace(2,8,6)
print(arr10)
```

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

Matrix operation

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

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

```
print(mat1*mat2)
```

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

```
print(mat1.dot(mat2))
```

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

```
print(mat1@mat2)
```

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

```
print(np.linalg.inv(mat2))
```

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

Statistics

```
ar1 = np.array([90,45,34,16,23,12])
print(np.mean(ar1))
```

```
36.666666666666664
```

```
print(np.median(ar1))
```

```
28.5
```

```
print(np.std(ar1))
```

```
26.278423764669668
```

```python
print(np.var(ar1))
```

```
690.5555555555557
```

Trigonometry

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

```
3.141592653589793
```

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

```
0.8939966636005579
-0.9880316240928618
0.8509035245341184
```

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

```
-0.4480736161291701
0.15425144988758405
0.5253219888177297
```

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

```
0.7071067811865476
1.0
0.8660254037844386
```

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

```
10.0
```

Arithmetic operation

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

```
33
```

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

```
[ 8 17 18]
```

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

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

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

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

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

```
5760
```

```
print(np.cumprod(c))
```

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

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

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

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

```
[[  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))
```

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

```
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))
```

```
268335
```

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

```
1
```

```
AB = np.array([0,-5,7,-23])
print(np.absolute(AB))
```

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

Logarithms

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

```
3.8066624897703196
```

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

```
1.6532125137753437
```

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

```
5.491853096329675
```

Universal function

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

```
111
```

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

```
12
```

Sorting

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

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

```
C = np.array([98,34,56,78,89])
D = sorted(C)
print(C)
print(D)

[98 34 56 78 89]
[34, 56, 78, 89, 98]
```

Rounding

```
s2 = np.array([ 9.1 , -7.8 ])
print(np.ceil(s2))

[10. -7.]

print(np.floor(s2))

[ 9. -8.]
```

Random module

```
import numpy.random as rd

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

[0.12143128 0.09445885]

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

2

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

[2 0 4 0 3 1]

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

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

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

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

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

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

Stack

```
Ar1 = np.array([[9,4,23],[3,4,5]])
Ar2 = np.array([[8,5,2],[33,42,51]])
print(Ar1)
print("\n")
print(Ar2)
```

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


[[ 8  5  2]
 [33 42 51]]
```

```
Ar3 = np.hstack((Ar1,Ar2))  #side by side
print(Ar3)                  #h=horizontal
```

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

```
Ar4 = np.vstack((Ar1,Ar2))   #one top another
print(Ar4)                   #v=vertical
```

```
[[ 9  4 23]
 [ 3  4  5]
 [ 8  5  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)
print(Ar6)
```

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

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

SET

```
S1 = np.array([9,3,5,2,1])
S2 = np.array([4,5,2,1,3])
print(S1,"\n")
print(S2)
```

```
[9 3 5 2 1]

[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))
```

```
[9]
```

Search

```
col1 = np.array([44,33,12,67,19])
index = np.where(col1%2 == 0)
print(index)
```

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

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

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