EDS LAB ASSIGNMENT 3 :

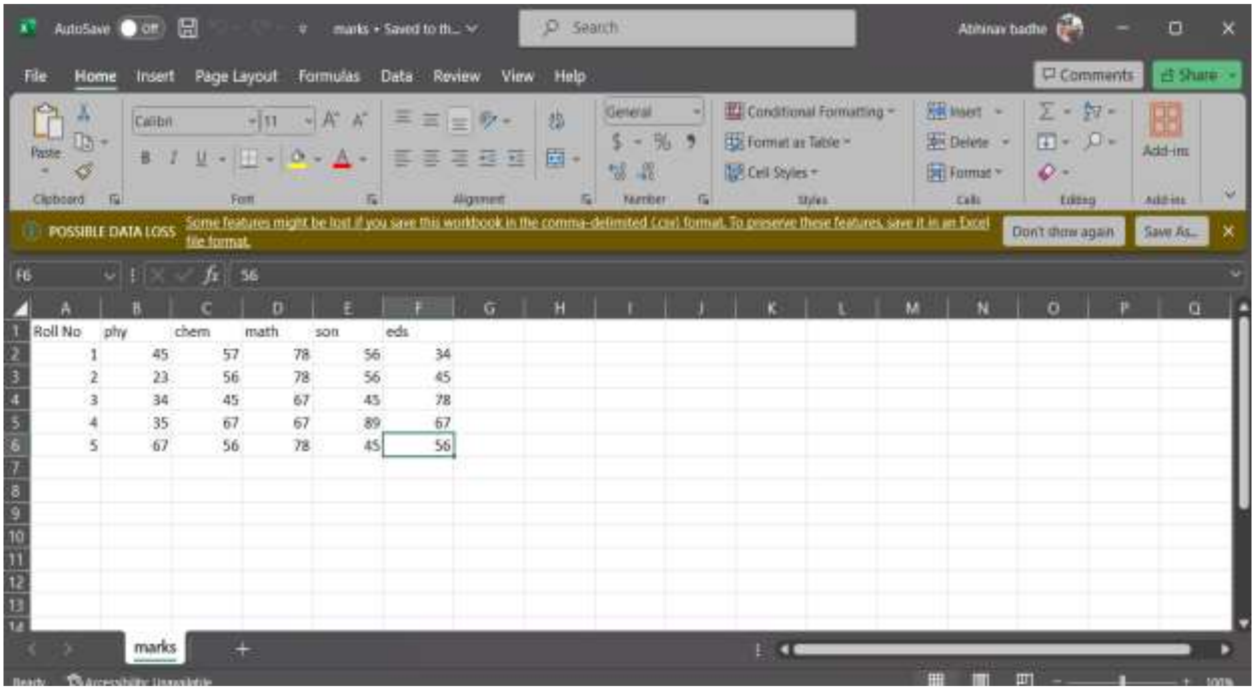
Name : OMKAR LONDHE

Roll No: C173 DIVISION : C1

Batch : C1 [D]

PRN NO. : 202301040027

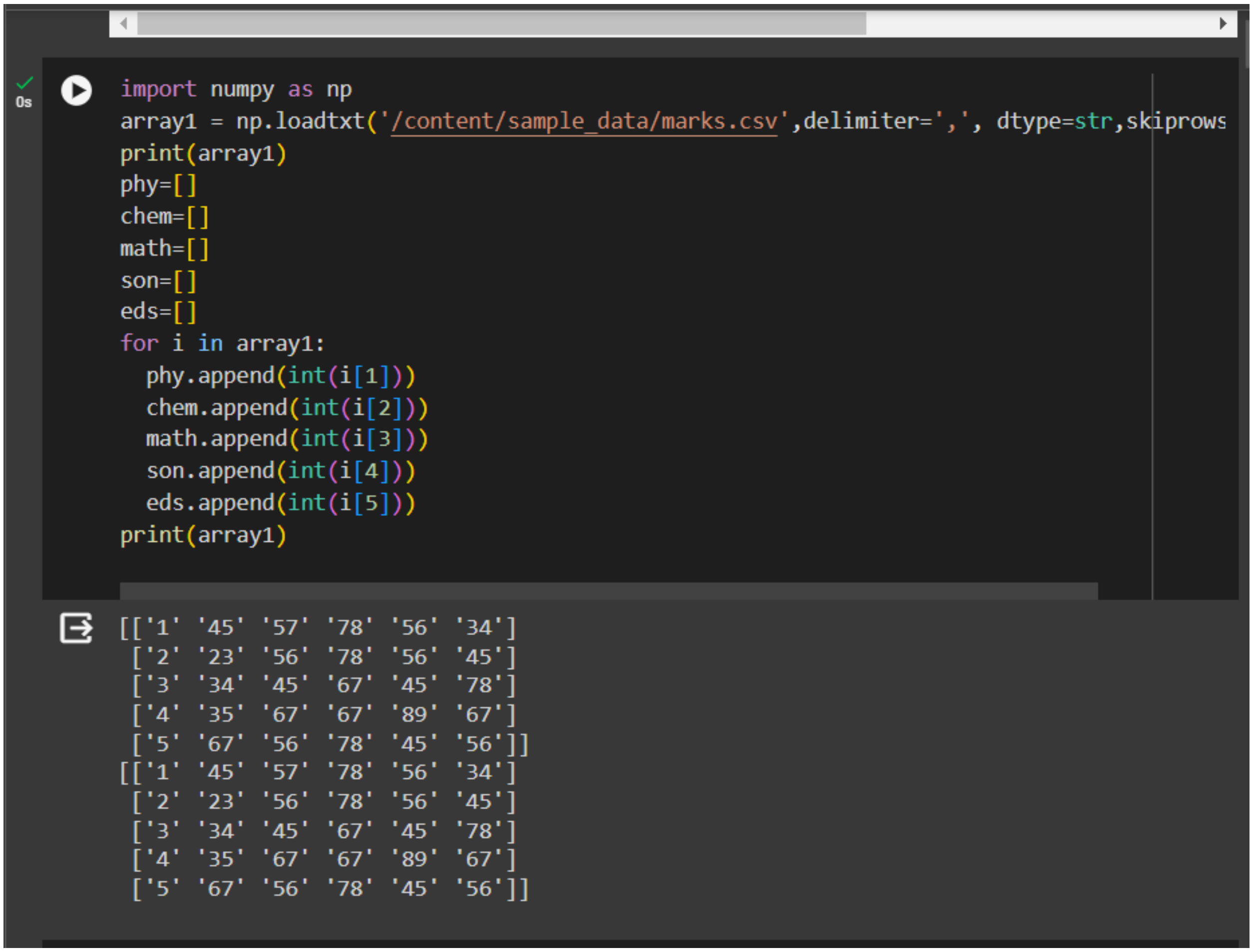
Dataset:



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

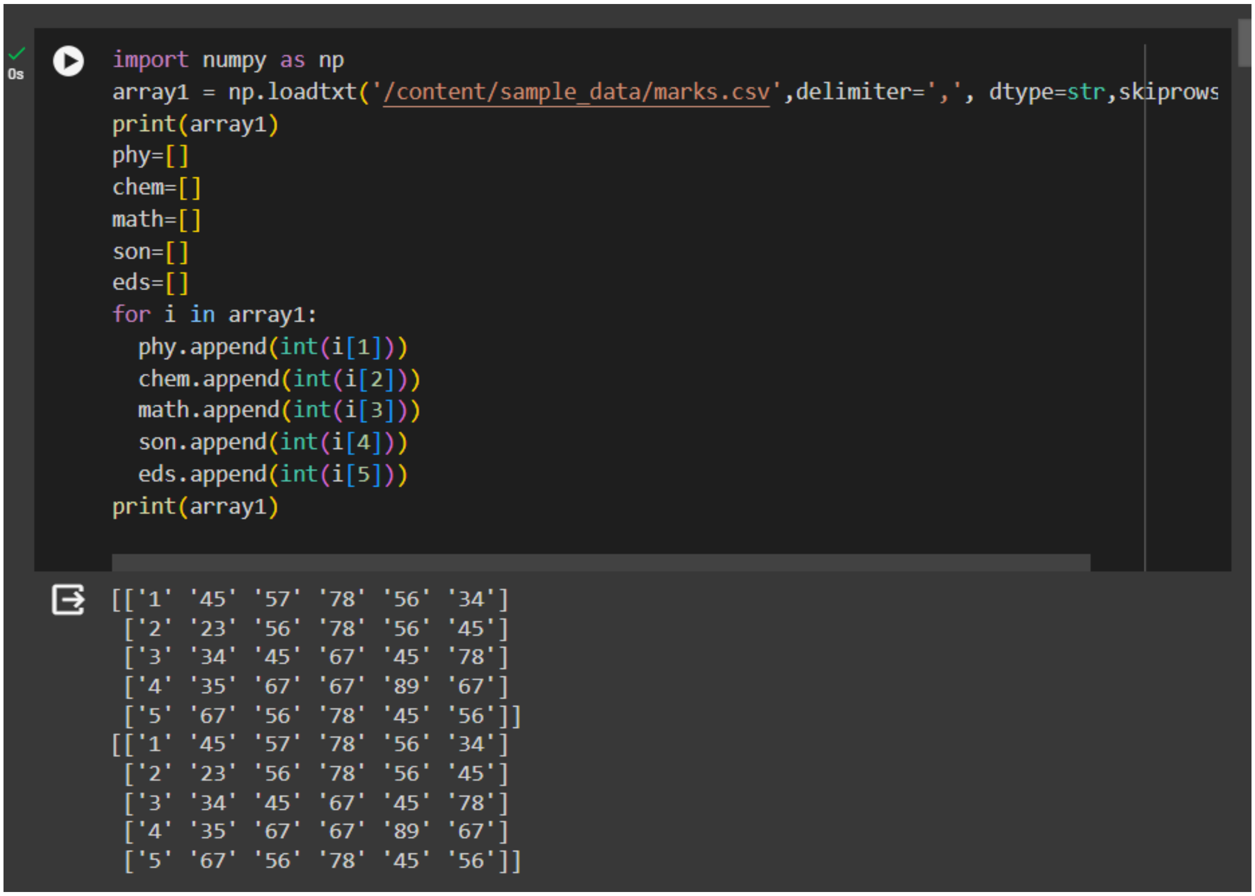
F6 56

Roll No	phy	chem	math	son	eds
1	45	57	78	56	34
2	23	56	78	56	45
3	34	45	67	45	78
4	35	67	67	89	67
5	67	56	78	45	56



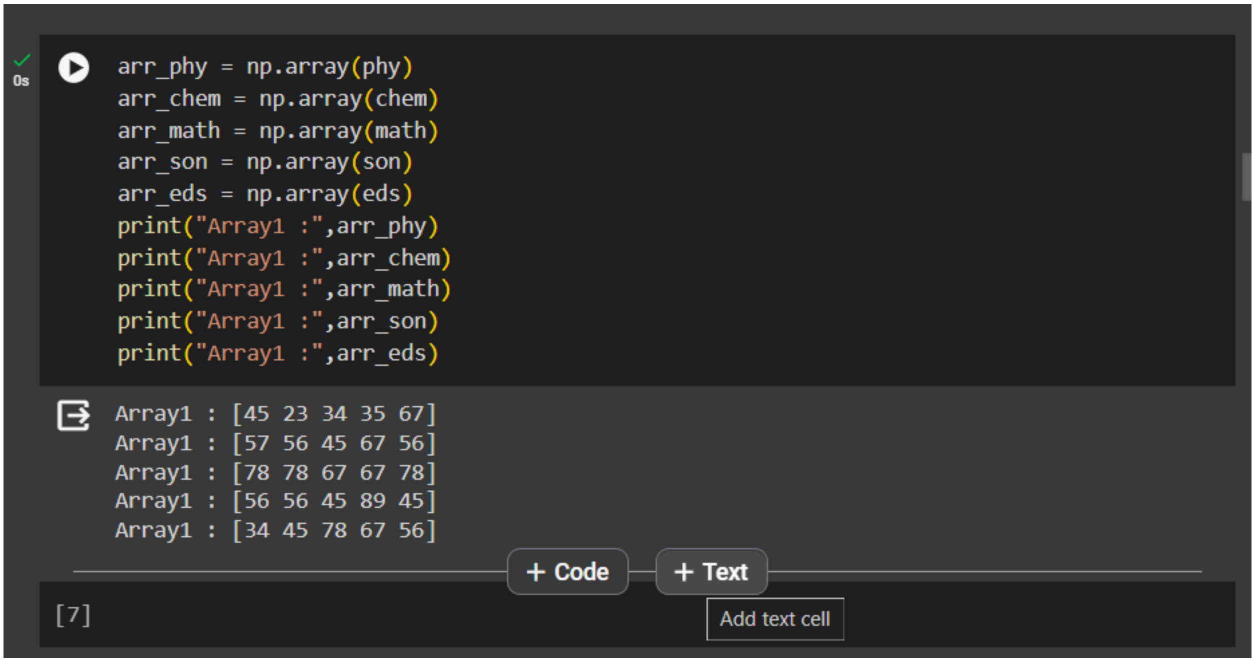
```python
import numpy as np
array1 = np.loadtxt('/content/sample_data/marks.csv',delimiter=',', dtype=str,skiprows
print(array1)
phy=[]
chem=[]
math=[]
son=[]
eds=[]
for i in array1:
  phy.append(int(i[1]))
  chem.append(int(i[2]))
  math.append(int(i[3]))
  son.append(int(i[4]))
  eds.append(int(i[5]))
print(array1)
```

```
[['1' '45' '57' '78' '56' '34']
 ['2' '23' '56' '78' '56' '45']
 ['3' '34' '45' '67' '45' '78']
 ['4' '35' '67' '67' '89' '67']
 ['5' '67' '56' '78' '45' '56']]
[['1' '45' '57' '78' '56' '34']
 ['2' '23' '56' '78' '56' '45']
 ['3' '34' '45' '67' '45' '78']
 ['4' '35' '67' '67' '89' '67']
 ['5' '67' '56' '78' '45' '56']]
```



```
import numpy as np
array1 = np.loadtxt('/content/sample_data/marks.csv',delimiter=',', dtype=str,skiprows
print(array1)
phy=[]
chem=[]
math=[]
son=[]
eds=[]
for i in array1:
  phy.append(int(i[1]))
  chem.append(int(i[2]))
  math.append(int(i[3]))
  son.append(int(i[4]))
  eds.append(int(i[5]))
print(array1)
```

```
[['1' '45' '57' '78' '56' '34']
 ['2' '23' '56' '78' '56' '45']
 ['3' '34' '45' '67' '45' '78']
 ['4' '35' '67' '67' '89' '67']
 ['5' '67' '56' '78' '45' '56']]
[['1' '45' '57' '78' '56' '34']
 ['2' '23' '56' '78' '56' '45']
 ['3' '34' '45' '67' '45' '78']
 ['4' '35' '67' '67' '89' '67']
 ['5' '67' '56' '78' '45' '56']]
```

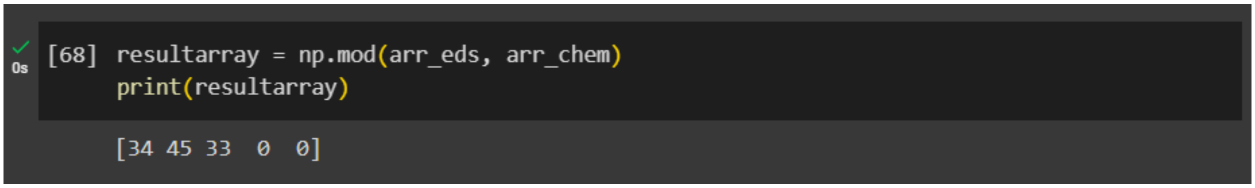


```
arr_phy = np.array(phy)
arr_chem = np.array(chem)
arr_math = np.array(math)
arr_son = np.array(son)
arr_eds = np.array(eds)
print("Array1 :",arr_phy)
print("Array1 :",arr_chem)
print("Array1 :",arr_math)
print("Array1 :",arr_son)
print("Array1 :",arr_eds)
```

```
Array1 : [45 23 34 35 67]
Array1 : [57 56 45 67 56]
Array1 : [78 78 67 67 78]
Array1 : [56 56 45 89 45]
Array1 : [34 45 78 67 56]
```

1] Perform All Matrix Operations :

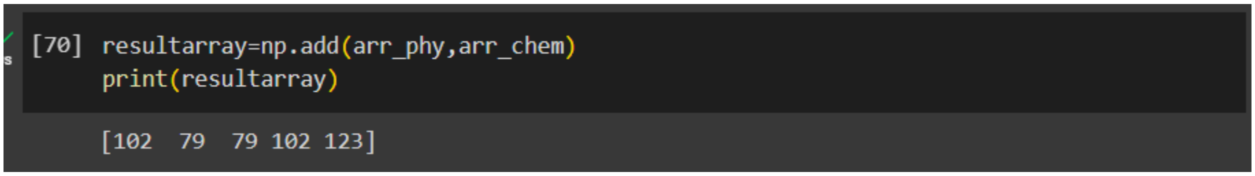
*Modulus:



```
resultarray = np.mod(arr_eds, arr_chem)
print(resultarray)
```

```
[34 45 33  0  0]
```

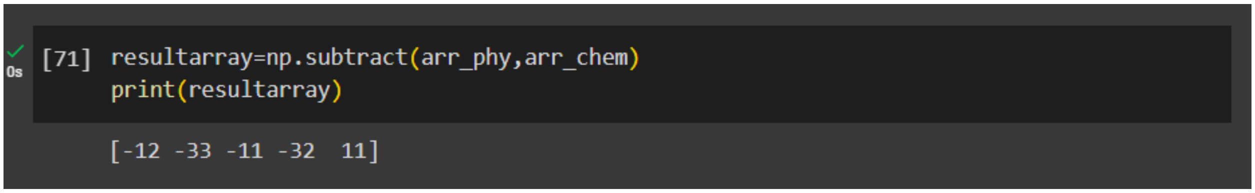
*Addition:



```
resultarray=np.add(arr_phy,arr_chem)
print(resultarray)
```

```
[102  79  79 102 123]
```

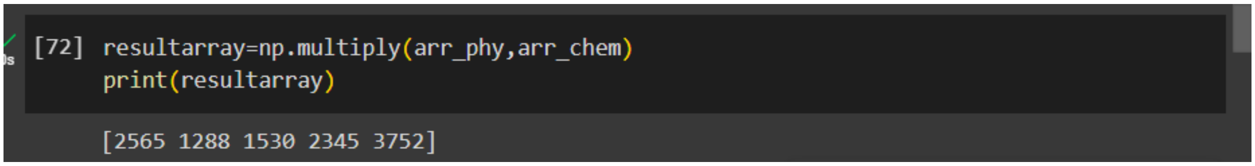
*Subtraction:



```
resultarray=np.subtract(arr_phy,arr_chem)
print(resultarray)
```

```
[-12 -33 -11 -32  11]
```

*Multiplication:

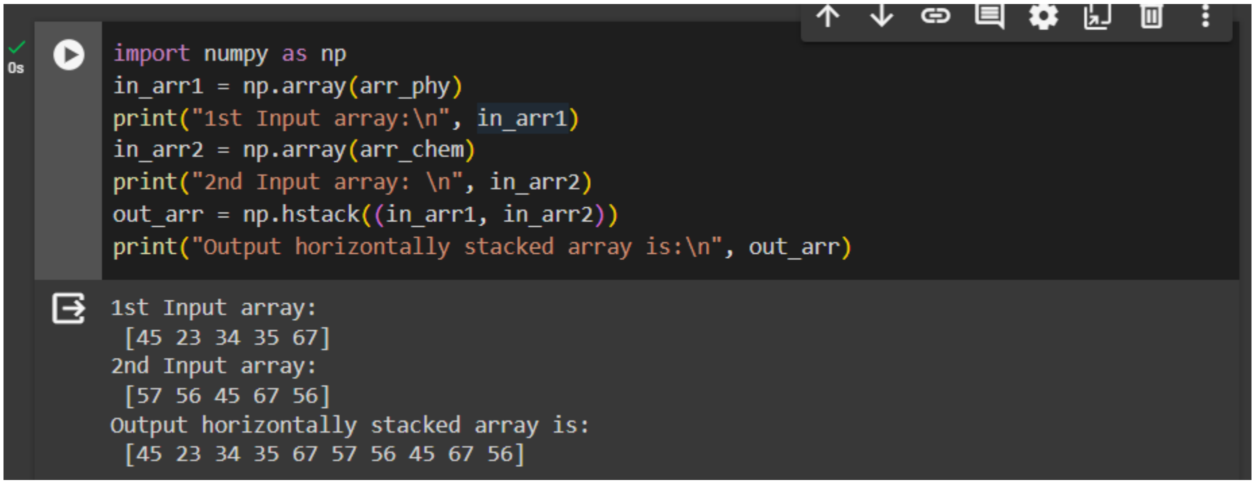


```
resultarray=np.multiply(arr_phy,arr_chem)
print(resultarray)
```

```
[2565 1288 1530 2345 3752]
```

2) Vertical and Horizontal stacking in numpy arrays:

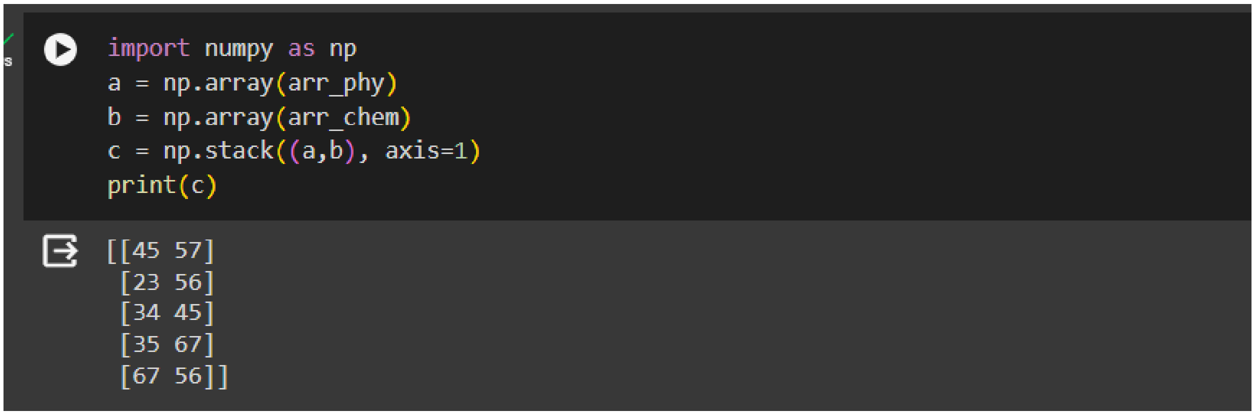
* Horizontal Stacking:



```
import numpy as np
in_arr1 = np.array(arr_phy)
print("1st Input array:\n", in_arr1)
in_arr2 = np.array(arr_chem)
print("2nd Input array: \n", in_arr2)
out_arr = np.hstack((in_arr1, in_arr2))
print("Output horizontally stacked array is:\n", out_arr)
```

```
1st Input array:
 [45 23 34 35 67]
2nd Input array:
 [57 56 45 67 56]
Output horizontally stacked array is:
 [45 23 34 35 67 57 56 45 67 56]
```

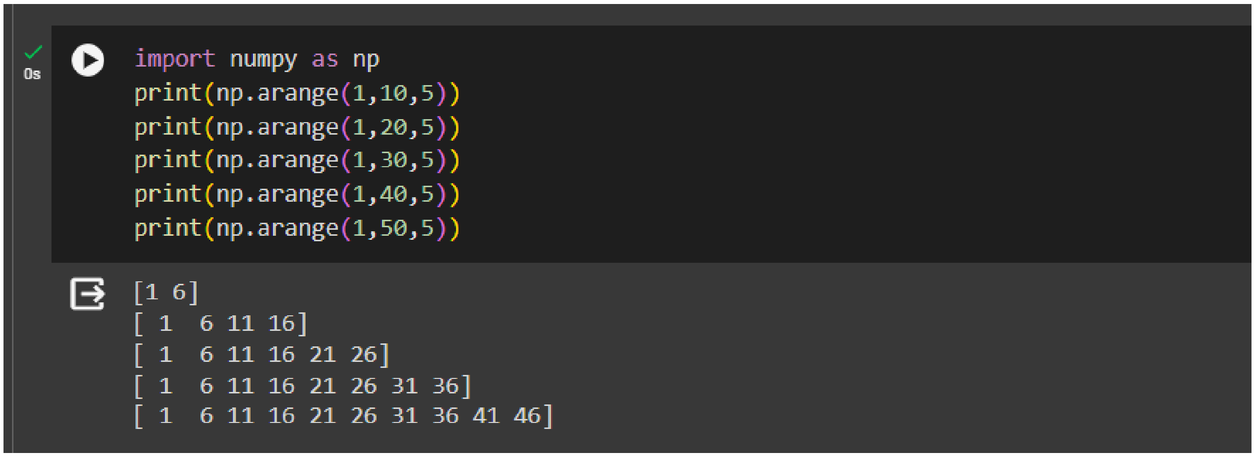
*Vertical Stacking:



```
import numpy as np
a = np.array(arr_phy)
b = np.array(arr_chem)
c = np.stack((a,b), axis=1)
print(c)
```

```
[[45 57]
 [23 56]
 [34 45]
 [35 67]
 [67 56]]
```

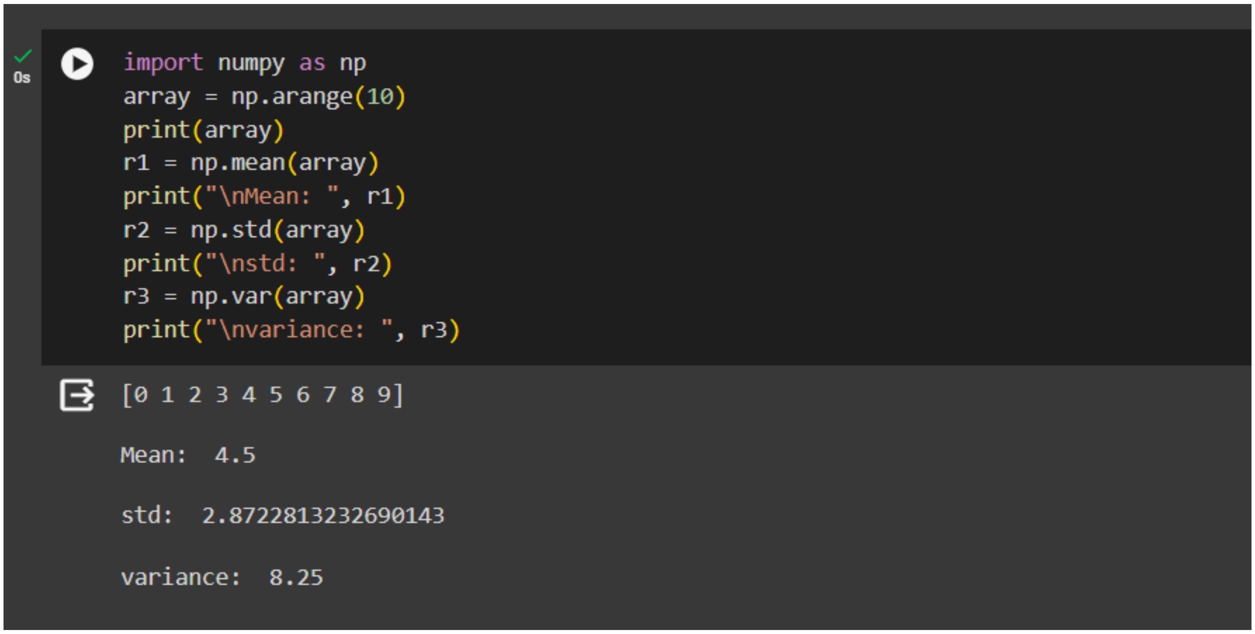
4) Custom Sequence Generation:



```
import numpy as np
print(np.arange(1,10,5))
print(np.arange(1,20,5))
print(np.arange(1,30,5))
print(np.arange(1,40,5))
print(np.arange(1,50,5))
```

```
[1 6]
[ 1  6 11 16]
[ 1  6 11 16 21 26]
[ 1  6 11 16 21 26 31 36]
[ 1  6 11 16 21 26 31 36 41 46]
```

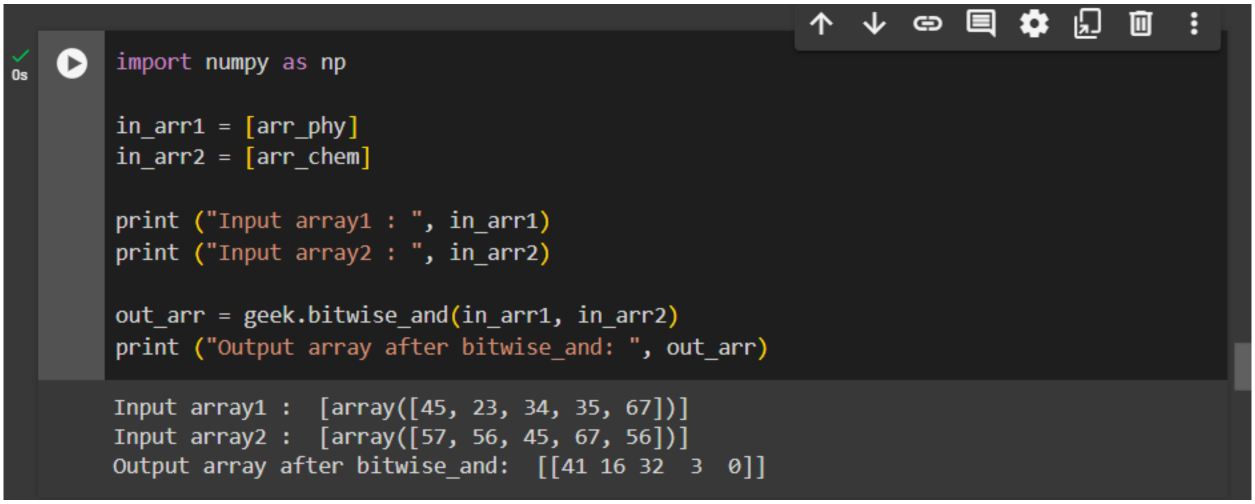
5) Arithematic and Statical Operations:



```
import numpy as np
array = np.arange(10)
print(array)
r1 = np.mean(array)
print("\nMean: ", r1)
r2 = np.std(array)
print("\nstd: ", r2)
r3 = np.var(array)
print("\nvariance: ", r3)
```

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

Mean:  4.5

std:  2.8722813232690143

variance:  8.25
```

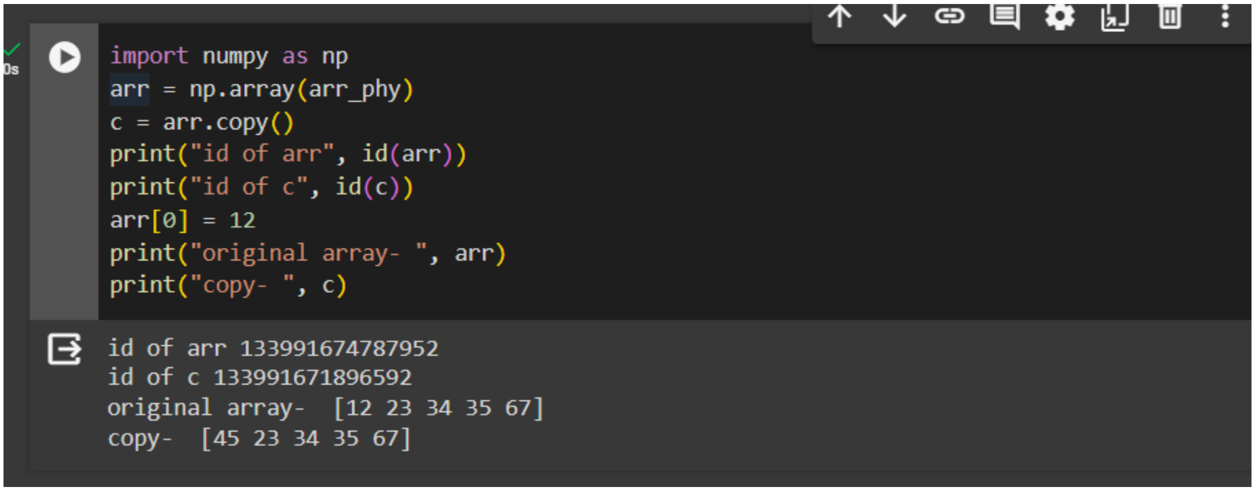
6) Mathematical And Bitwise Operations:



```
import numpy as np

in_arr1 = [arr_phy]
in_arr2 = [arr_chem]

print ("Input array1 : ", in_arr1)
print ("Input array2 : ", in_arr2)

out_arr = geek.bitwise_and(in_arr1, in_arr2)
print ("Output array after bitwise_and: ", out_arr)
```

```
Input array1 :  [array([45, 23, 34, 35, 67])]
Input array2 :  [array([57, 56, 45, 67, 56])]
Output array after bitwise_and:  [[41 16 32  3  0]]
```

7) Copying and Viewing Arrays:

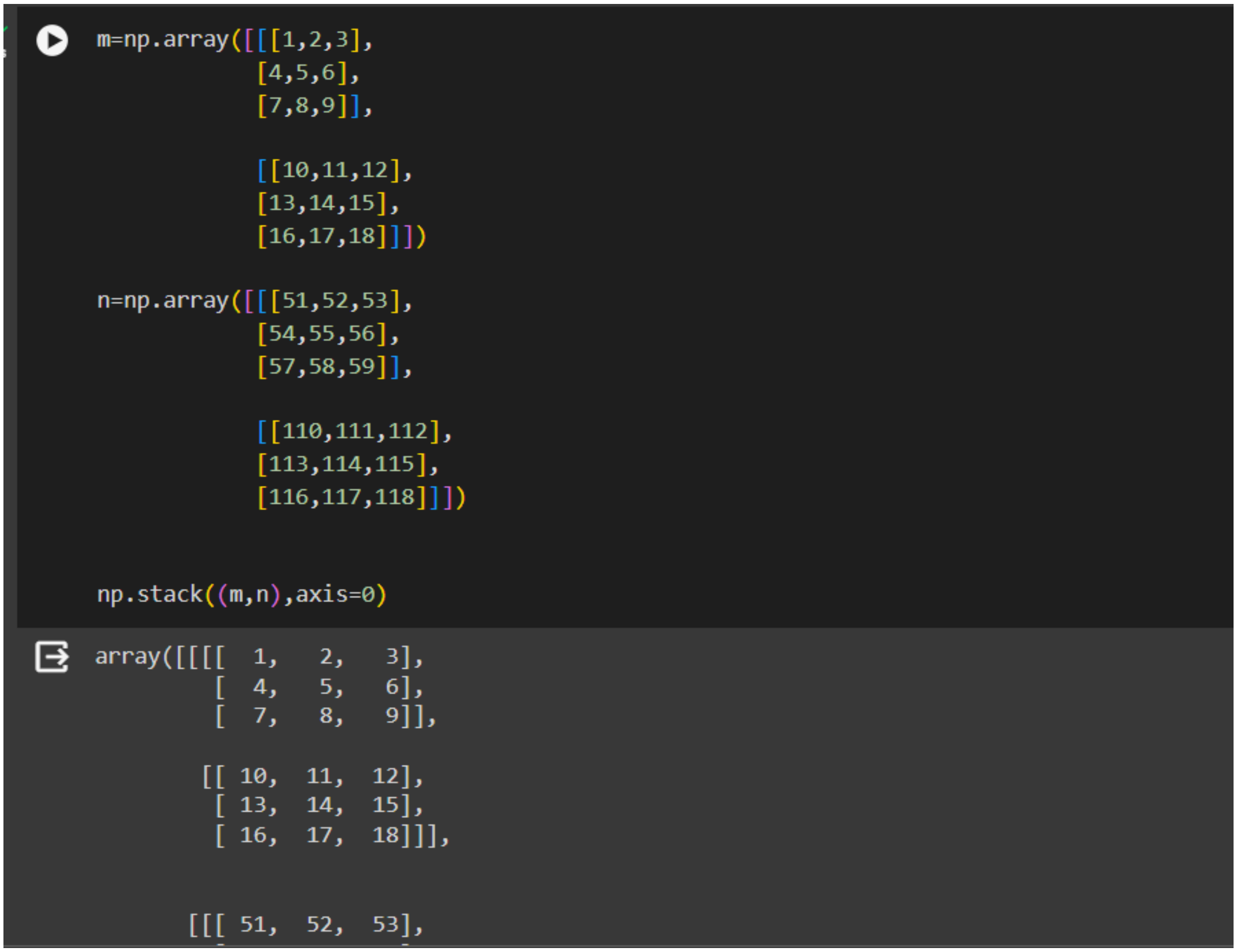


```
import numpy as np
arr = np.array(arr_phy)
c = arr.copy()
print("id of arr", id(arr))
print("id of c", id(c))
arr[0] = 12
print("original array- ", arr)
print("copy- ", c)
```

```
id of arr 133991674787952
id of c 133991671896592
original array-  [12 23 34 35 67]
copy-  [45 23 34 35 67]
```

8)Data Stacking, Searching, Sorting, Counting, Broadcasting:

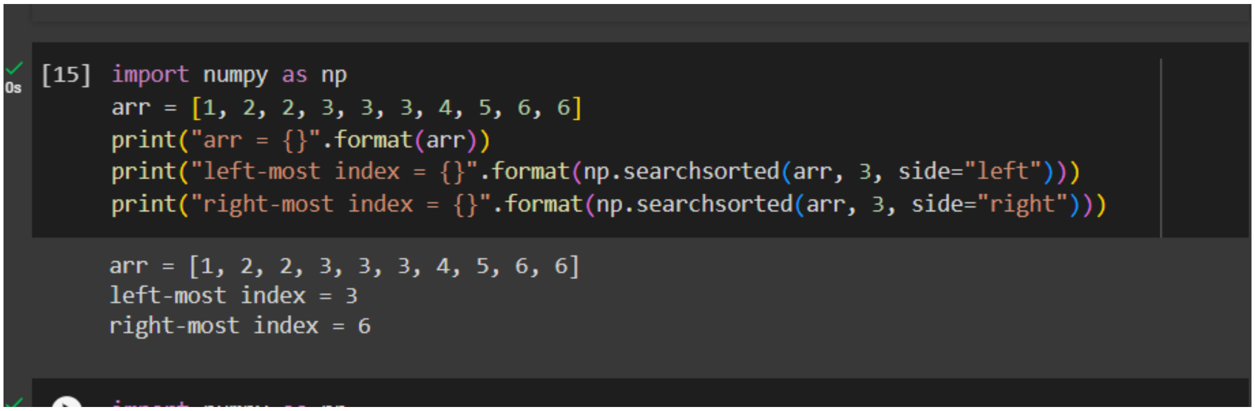
*Data Stacking:



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

            [[10,11,12],
            [13,14,15],
            [16,17,18]]])

n=np.array([[[51,52,53],
            [54,55,56],
            [57,58,59]],

            [[110,111,112],
            [113,114,115],
            [116,117,118]]])


np.stack((m,n),axis=0)
```

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

        [[ 10,  11,  12],
         [ 13,  14,  15],
         [ 16,  17,  18]]],


       [[[ 51,  52,  53],
```

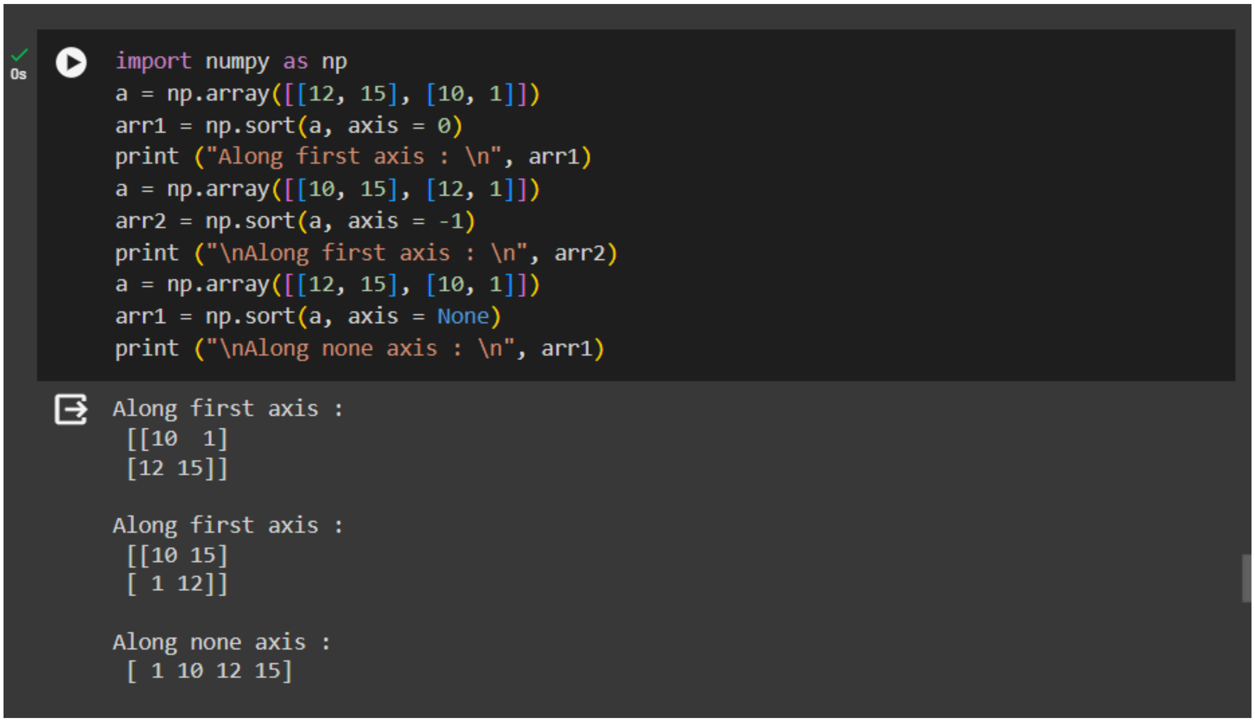
*Searching:



```
import numpy as np
arr = [1, 2, 2, 3, 3, 3, 4, 5, 6, 6]
print("arr = {}".format(arr))
print("left-most index = {}".format(np.searchsorted(arr, 3, side="left")))
print("right-most index = {}".format(np.searchsorted(arr, 3, side="right")))
```

```
arr = [1, 2, 2, 3, 3, 3, 4, 5, 6, 6]
left-most index = 3
right-most index = 6
```

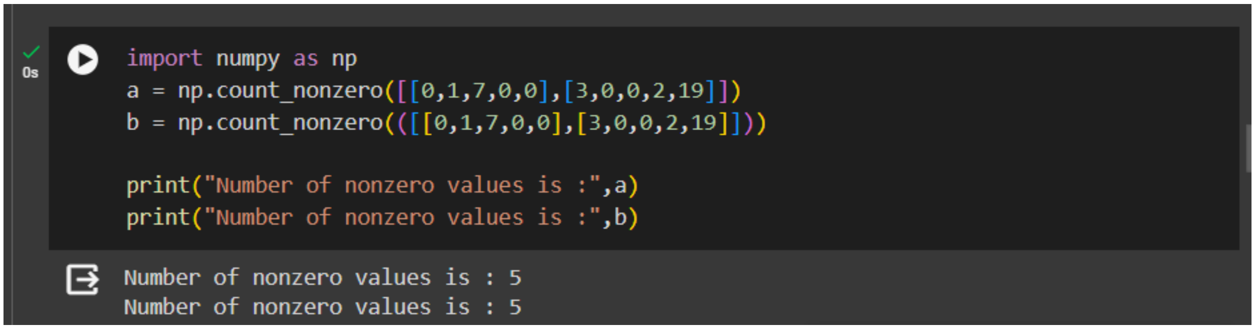
*Sorting:



```
import numpy as np
a = np.array([[12, 15], [10, 1]])
arr1 = np.sort(a, axis = 0)
print ("Along first axis : \n", arr1)
a = np.array([[10, 15], [12, 1]])
arr2 = np.sort(a, axis = -1)
print ("\nAlong first axis : \n", arr2)
a = np.array([[12, 15], [10, 1]])
arr1 = np.sort(a, axis = None)
print ("\nAlong none axis : \n", arr1)
```

```
Along first axis :
 [[10  1]
 [12 15]]

Along first axis :
 [[10 15]
 [ 1 12]]

Along none axis :
 [ 1 10 12 15]
```

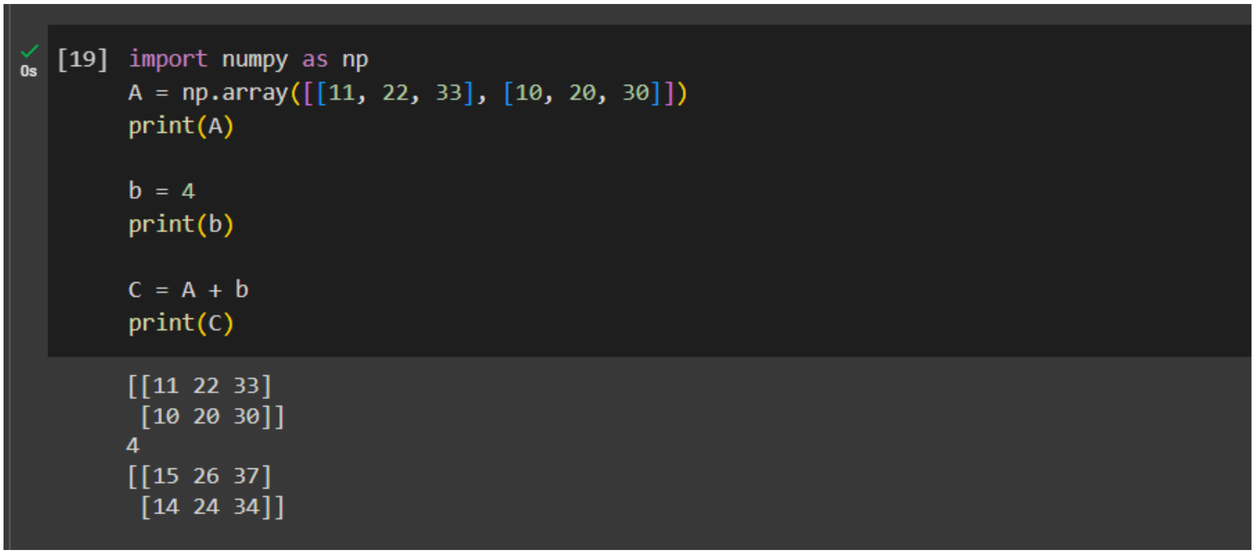
*Counting:



```
import numpy as np
a = np.count_nonzero([[0,1,7,0,0],[3,0,0,2,19]])
b = np.count_nonzero(([[0,1,7,0,0],[3,0,0,2,19]]))

print("Number of nonzero values is :",a)
print("Number of nonzero values is :",b)
```

```
Number of nonzero values is : 5
Number of nonzero values is : 5
```

*Broadcasting:



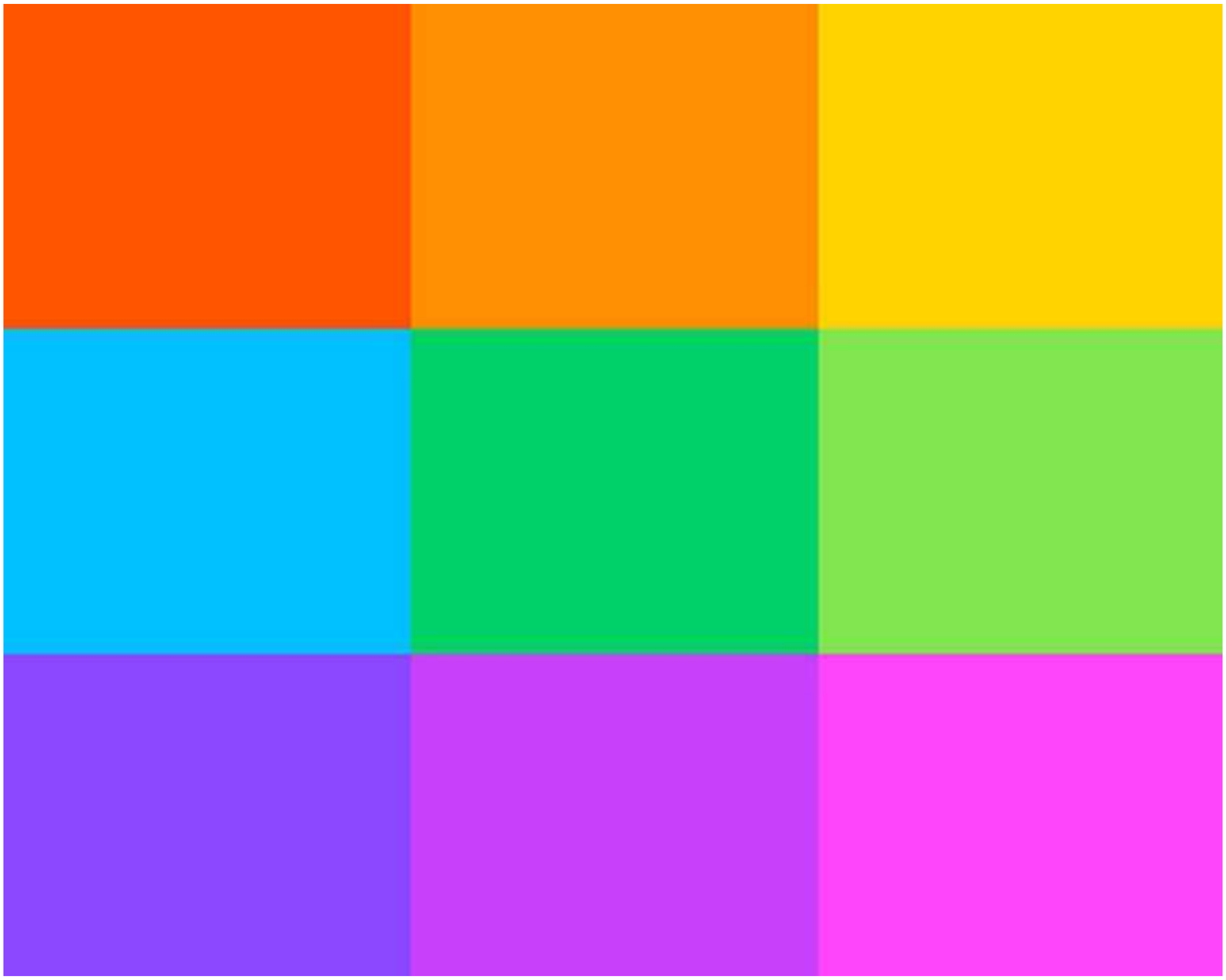
```
import numpy as np
A = np.array([[11, 22, 33], [10, 20, 30]])
print(A)

b = 4
print(b)

C = A + b
print(C)
```

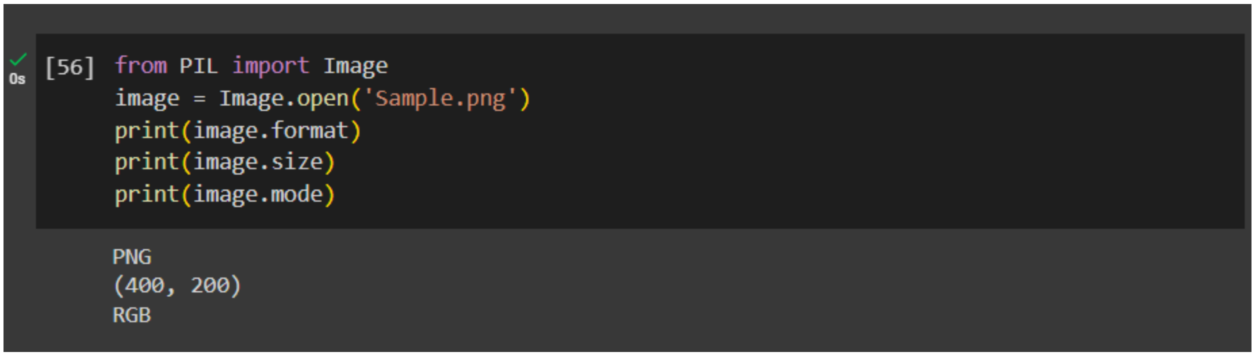
```
[[11 22 33]
 [10 20 30]]
4
[[15 26 37]
 [14 24 34]]
```

Self Study Assignment:

For any real life application, perform advanced data operations such as image as array and image manipulations

1) Image as Array:
   Image:



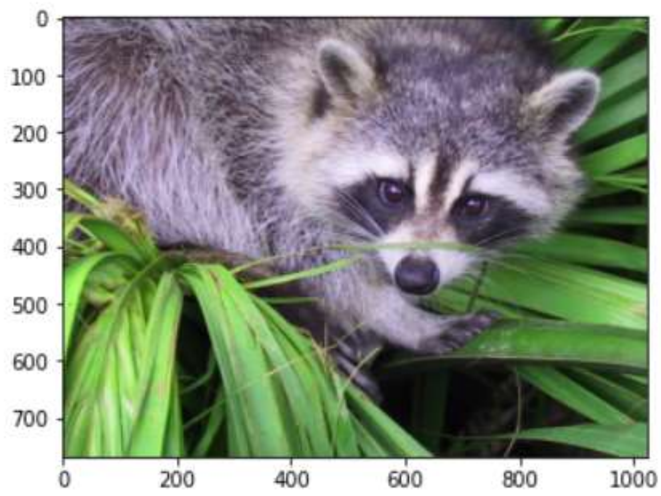


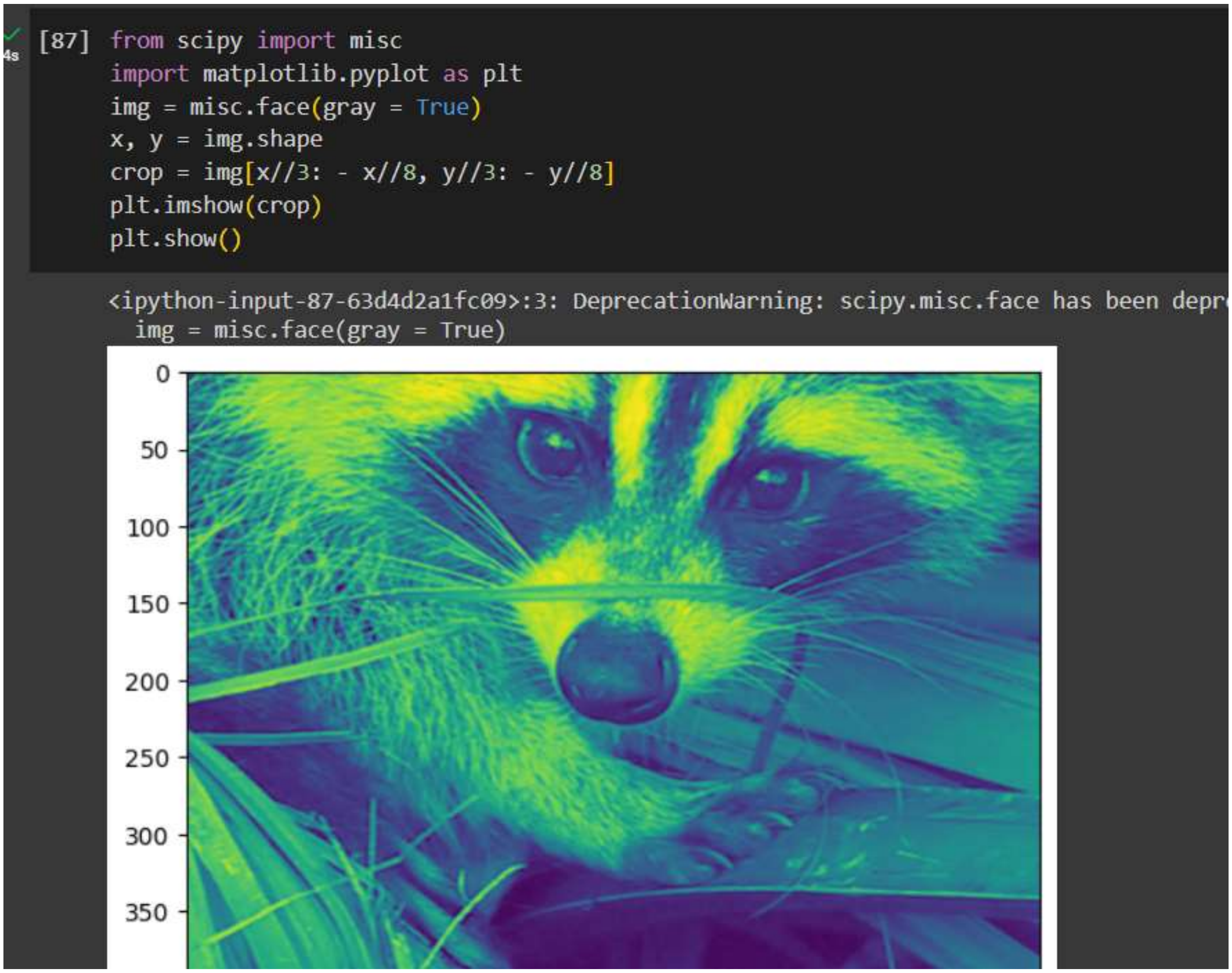
```
from PIL import Image
image = Image.open('Sample.png')
print(image.format)
print(image.size)
print(image.mode)
```

```
PNG
(400, 200)
RGB
```

2)Image Manipulation:

Sample Image:





```python
from scipy import misc
import matplotlib.pyplot as plt
img = misc.face(gray = True)
x, y = img.shape
crop = img[x//3: - x//8, y//3: - y//8]
plt.imshow(crop)
plt.show()
```

```
<ipython-input-87-63d4d2a1fc09>:3: DeprecationWarning: scipy.misc.face has been depr
  img = misc.face(gray = True)
```