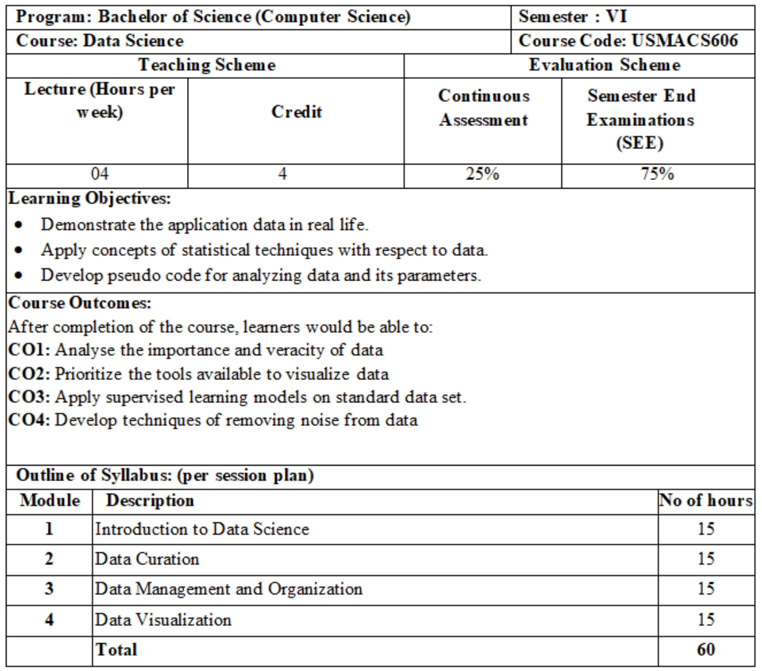


| Program: Bachelor of Science (Computer Science) | | | Semester : VI | |
|---|---|---|---|---|
| Course: Data Science | | | Course Code: USMACS606 | |
| Teaching Scheme | | Evaluation Scheme | | |
| Lecture (Hours per week) | Credit | Continuous Assessment | Semester End Examinations (SEE) | |
| 04 | 4 | 25% | 75% | |

**Learning Objectives:**

- Demonstrate the application data in real life.
- Apply concepts of statistical techniques with respect to data.
- Develop pseudo code for analyzing data and its parameters.

**Course Outcomes:**

After completion of the course, learners would be able to:

**CO1:** Analyse the importance and veracity of data

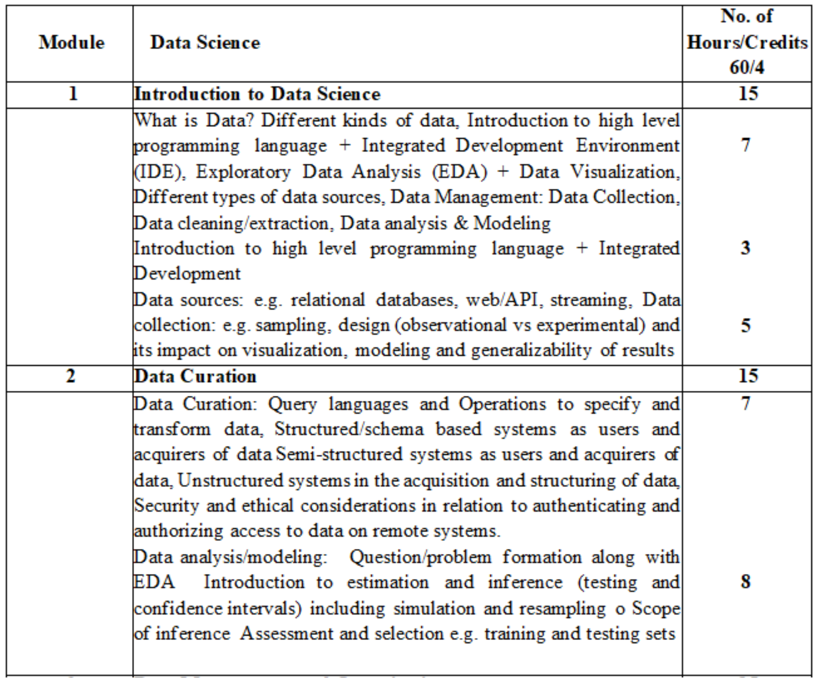
**CO2:** Prioritize the tools available to visualize data

**CO3:** Apply supervised learning models on standard data set.

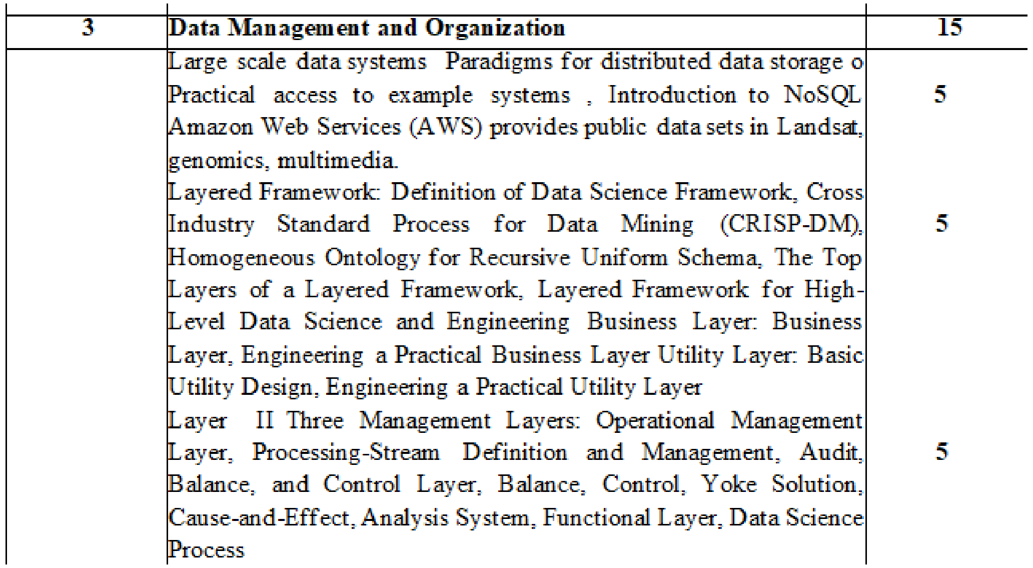
**CO4:** Develop techniques of removing noise from data

**Outline of Syllabus: (per session plan)**

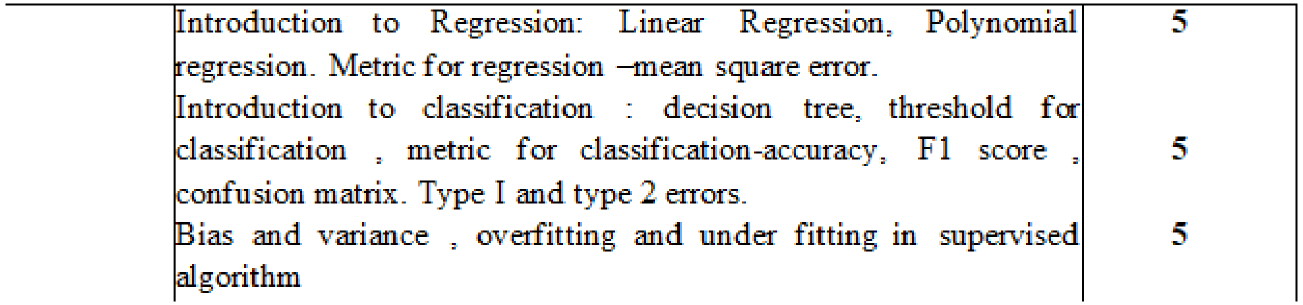
| Module | Description | No of hours |
|---|---|---|
| 1 | Introduction to Data Science | 15 |
| 2 | Data Curation | 15 |
| 3 | Data Management and Organization | 15 |
| 4 | Data Visualization | 15 |
| | **Total** | **60** |



| Module | Data Science | No. of Hours/Credits 60/4 |
|---|---|---|
| 1 | **Introduction to Data Science** | 15 |
| | What is Data? Different kinds of data, Introduction to high level programming language + Integrated Development Environment (IDE), Exploratory Data Analysis (EDA) + Data Visualization, Different types of data sources, Data Management: Data Collection, Data cleaning/extraction, Data analysis & Modeling | 7 |
| | Introduction to high level programming language + Integrated Development | 3 |
| | Data sources: e.g. relational databases, web/API, streaming, Data collection: e.g. sampling, design (observational vs experimental) and its impact on visualization, modeling and generalizability of results | 5 |
| 2 | **Data Curation** | 15 |
| | Data Curation: Query languages and Operations to specify and transform data, Structured/schema based systems as users and acquirers of data Semi-structured systems as users and acquirers of data, Unstructured systems in the acquisition and structuring of data, Security and ethical considerations in relation to authenticating and authorizing access to data on remote systems. | 7 |
| | Data analysis/modeling: Question/problem formation along with EDA Introduction to estimation and inference (testing and confidence intervals) including simulation and resampling o Scope of inference Assessment and selection e.g. training and testing sets | 8 |



| | | |
|---|---|---|
| 3 | **Data Management and Organization** | 15 |
| | Large scale data systems Paradigms for distributed data storage o Practical access to example systems , Introduction to NoSQL Amazon Web Services (AWS) provides public data sets in Landsat, genomics, multimedia. | 5 |
| | Layered Framework: Definition of Data Science Framework, Cross Industry Standard Process for Data Mining (CRISP-DM), Homogeneous Ontology for Recursive Uniform Schema, The Top Layers of a Layered Framework, Layered Framework for High-Level Data Science and Engineering Business Layer: Business Layer, Engineering a Practical Business Layer Utility Layer: Basic Utility Design, Engineering a Practical Utility Layer | 5 |
| | Layer II Three Management Layers: Operational Management Layer, Processing-Stream Definition and Management, Audit, Balance, and Control Layer, Balance, Control, Yoke Solution, Cause-and-Effect, Analysis System, Functional Layer, Data Science Process | 5 |



| | | |
|---|---|---|
| | Introduction to Regression: Linear Regression, Polynomial regression. Metric for regression –mean square error. | 5 |
| | Introduction to classification : decision tree, threshold for classification , metric for classification-accuracy, F1 score , confusion matrix. Type I and type 2 errors. | 5 |
| | Bias and variance , overfitting and under fitting in supervised algorithm | 5 |

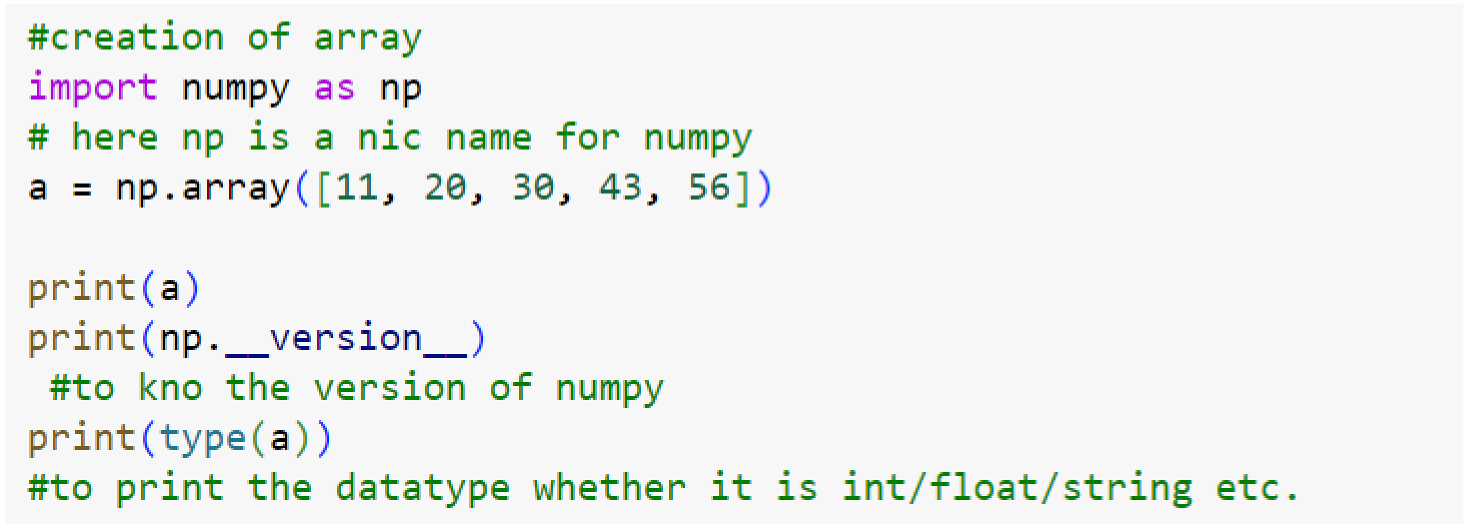
# Introduction to NumPy

## Features

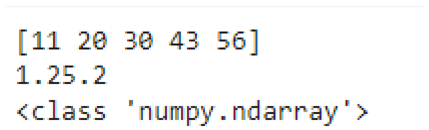
- Python library
- Handle arrays at ease.
- linear algebra, fourier transform, and matrices.
- lists that serve the purpose of arrays, but they are slow to process.

```
import numpy
```

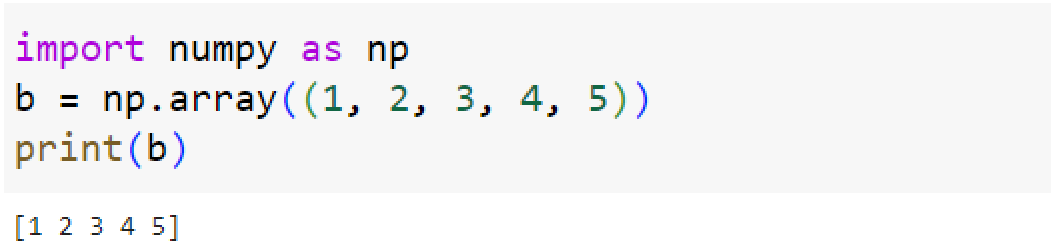
```
import numpy as np
```



```python
#creation of array
import numpy as np
# here np is a nic name for numpy
a = np.array([11, 20, 30, 43, 56])

print(a)
print(np.__version__)
 #to kno the version of numpy
print(type(a))
#to print the datatype whether it is int/float/string etc.
```

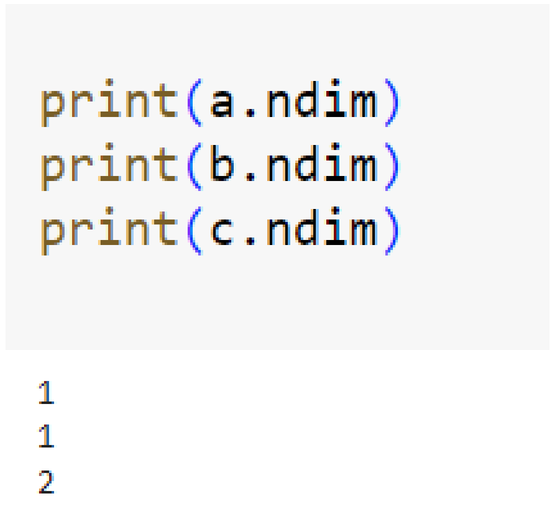


```
[11 20 30 43 56]
1.25.2
<class 'numpy.ndarray'>
```



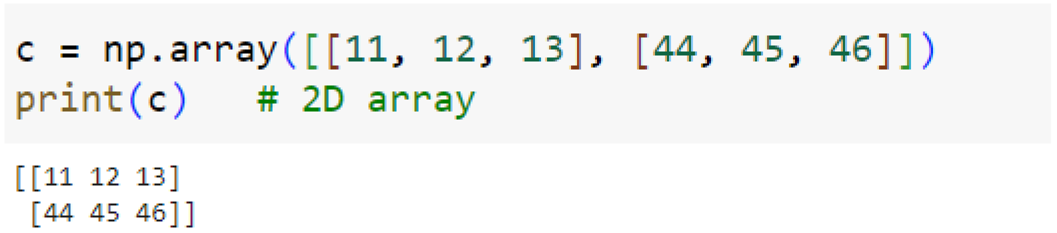
```python
import numpy as np
b = np.array((1, 2, 3, 4, 5))
print(b)
```

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



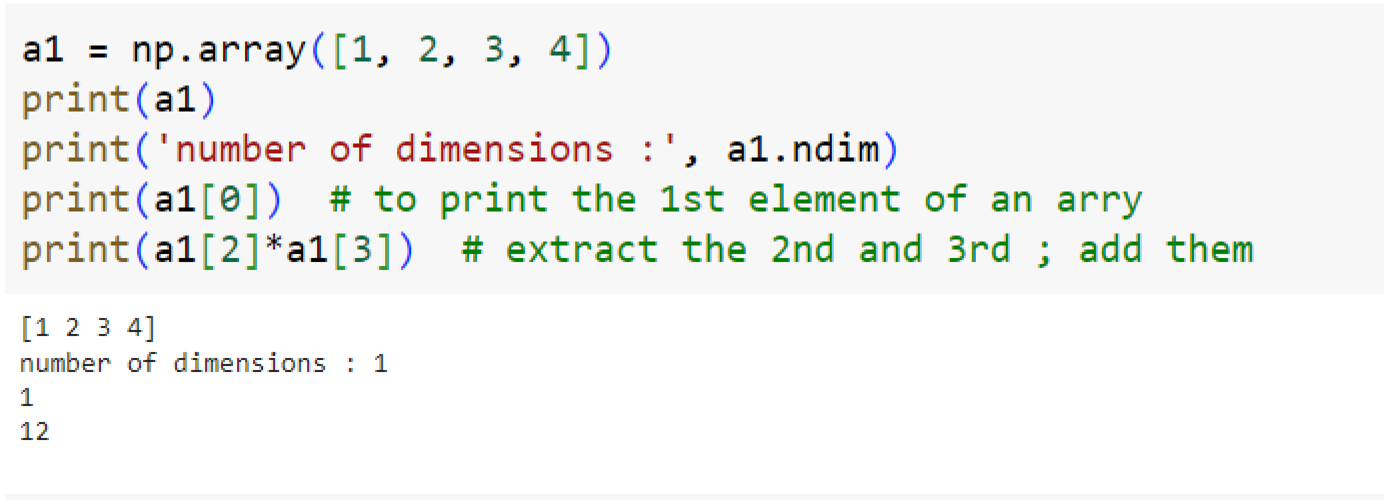
```python
print(a.ndim)
print(b.ndim)
print(c.ndim)
```

```
1
1
2
```



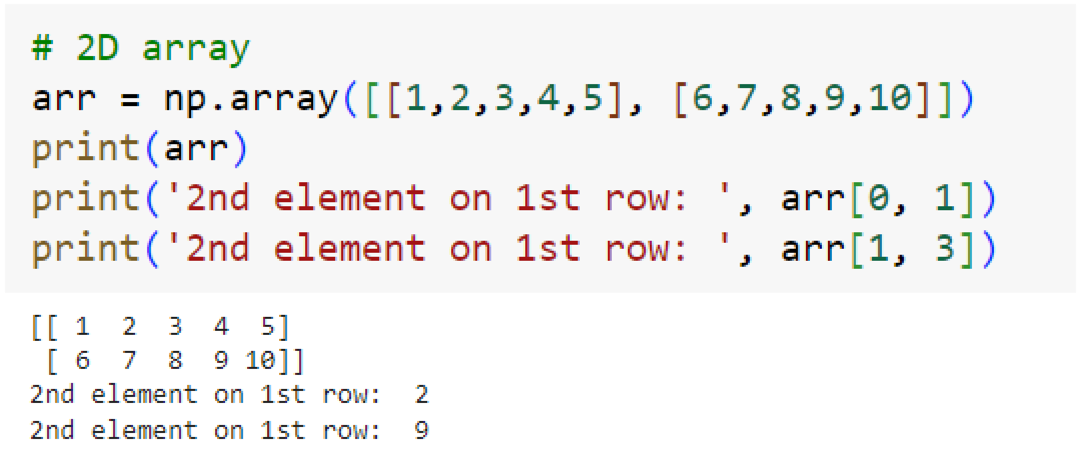
```python
c = np.array([[11, 12, 13], [44, 45, 46]])
print(c)   # 2D array
```

```
[[11 12 13]
 [44 45 46]]
```



```
a1 = np.array([1, 2, 3, 4])
print(a1)
print('number of dimensions :', a1.ndim)
print(a1[0])  # to print the 1st element of an arry
print(a1[2]*a1[3])  # extract the 2nd and 3rd ; add them
```

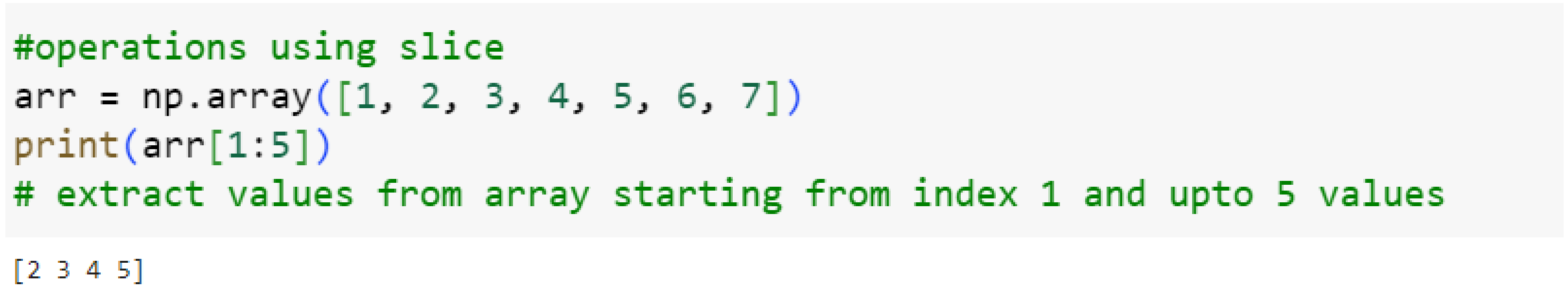
```
[1 2 3 4]
number of dimensions : 1
1
12
```



```
# 2D array
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print(arr)
print('2nd element on 1st row: ', arr[0, 1])
print('2nd element on 1st row: ', arr[1, 3])
```

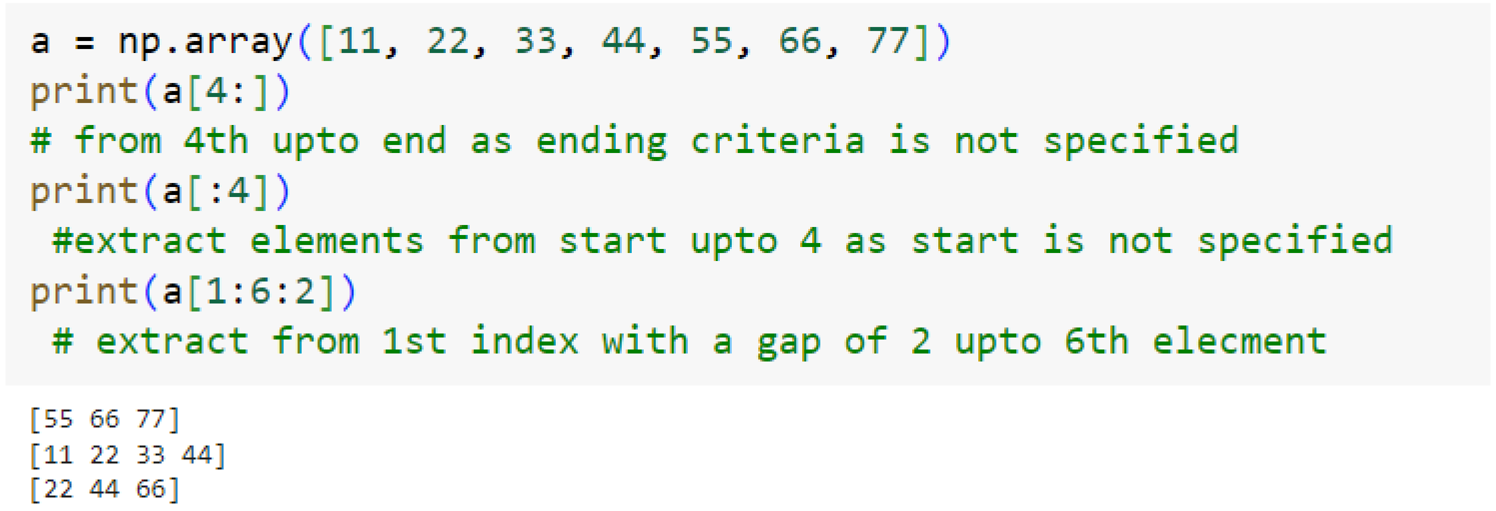
```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
2nd element on 1st row:  2
2nd element on 1st row:  9
```

Slicing ->> taking elements from one given index to another given index.
Syntax : [start:end]. or [start:end:step].
default value of start is 0 if not passed 0
If we don't pass end its considered length of array in that dimension
If we don't pass step its considered 1



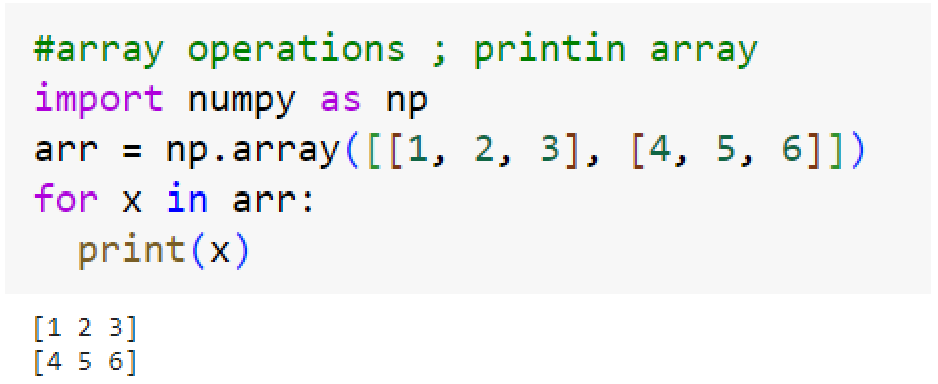
```
#operations using slice
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5])
# extract values from array starting from index 1 and upto 5 values
```

```
[2 3 4 5]
```



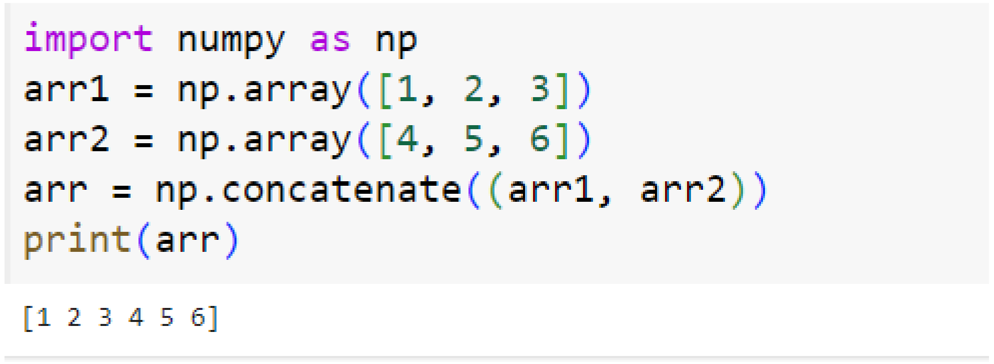
```
a = np.array([11, 22, 33, 44, 55, 66, 77])
print(a[4:])
# from 4th upto end as ending criteria is not specified
print(a[:4])
 #extract elements from start upto 4 as start is not specified
print(a[1:6:2])
 # extract from 1st index with a gap of 2 upto 6th elecment
```

```
[55 66 77]
[11 22 33 44]
[22 44 66]
```



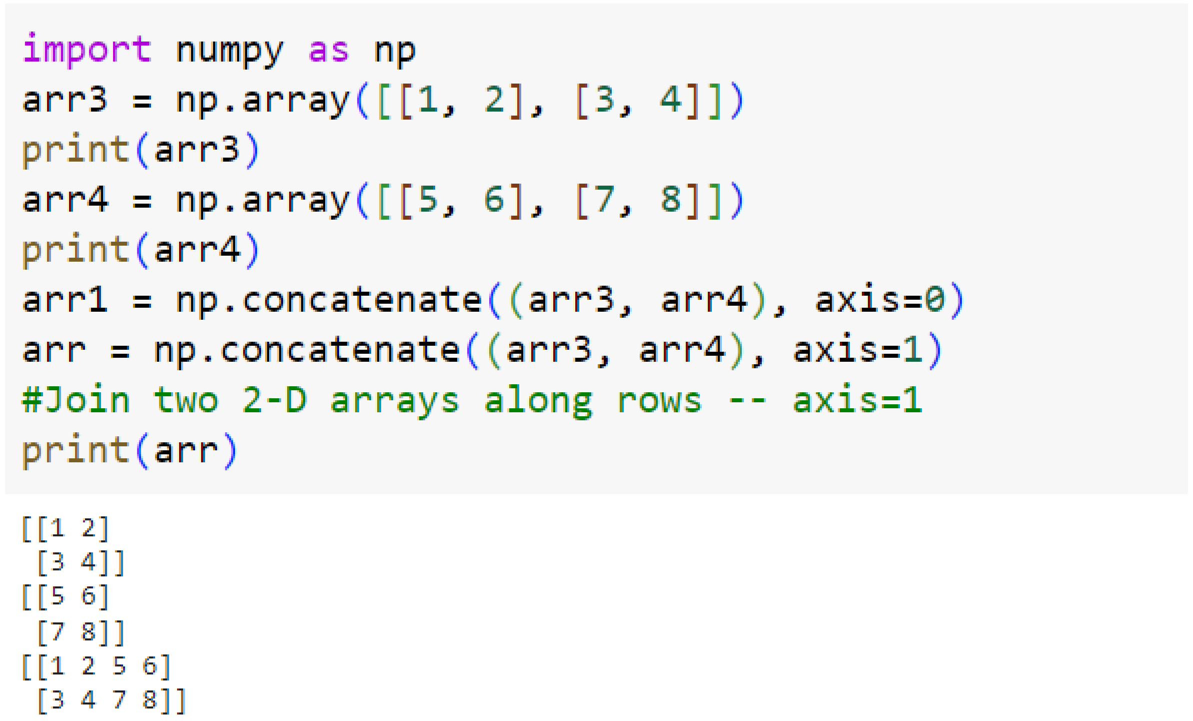
```
#array operations ; printin array
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
  print(x)
```

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



```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

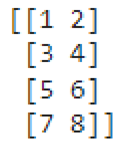
```
[1 2 3 4 5 6]
```



```
import numpy as np
arr3 = np.array([[1, 2], [3, 4]])
print(arr3)
arr4 = np.array([[5, 6], [7, 8]])
print(arr4)
arr1 = np.concatenate((arr3, arr4), axis=0)
arr = np.concatenate((arr3, arr4), axis=1)
#Join two 2-D arrays along rows -- axis=1
print(arr)
```

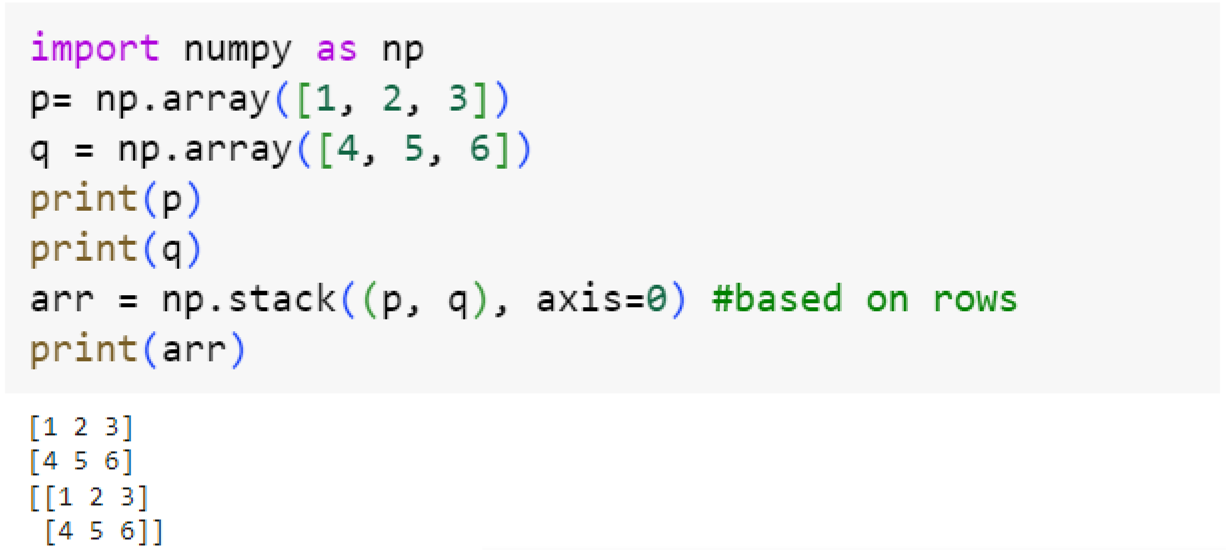
```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
[[1 2 5 6]
 [3 4 7 8]]
```

print(arr1)



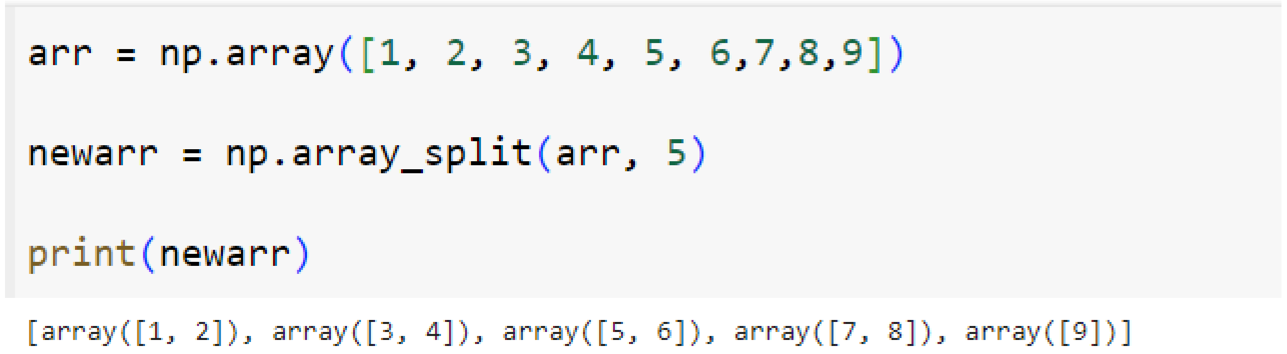
```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

Stacking == concatenation, the only difference is that stacking is done along a new axis.
We can concatenate two 1-D arrays along the second axis which would result in putting them one over the other, ie. stacking.
We pass a sequence of arrays that we want to join to the stack() method along with the axis. If axis is not explicitly passed it is taken as 0.



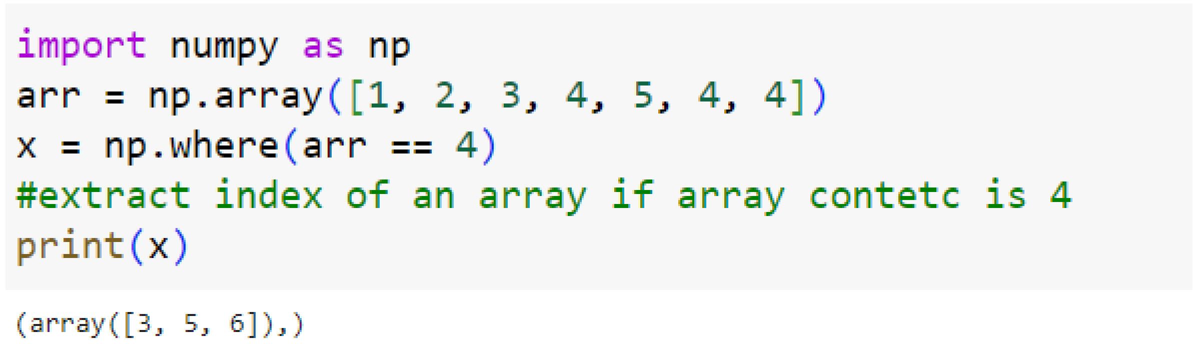
```python
import numpy as np
p= np.array([1, 2, 3])
q = np.array([4, 5, 6])
print(p)
print(q)
arr = np.stack((p, q), axis=0) #based on rows
print(arr)
```

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



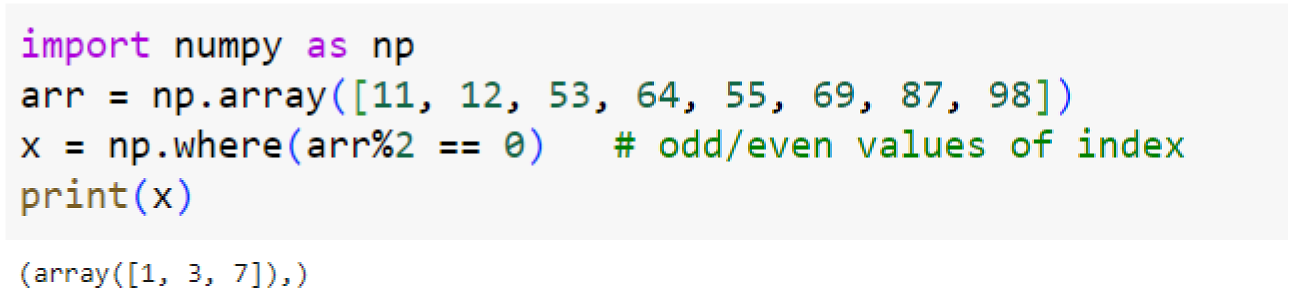
```python
arr = np.array([1, 2, 3, 4, 5, 6,7,8,9])

newarr = np.array_split(arr, 5)

print(newarr)
```

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



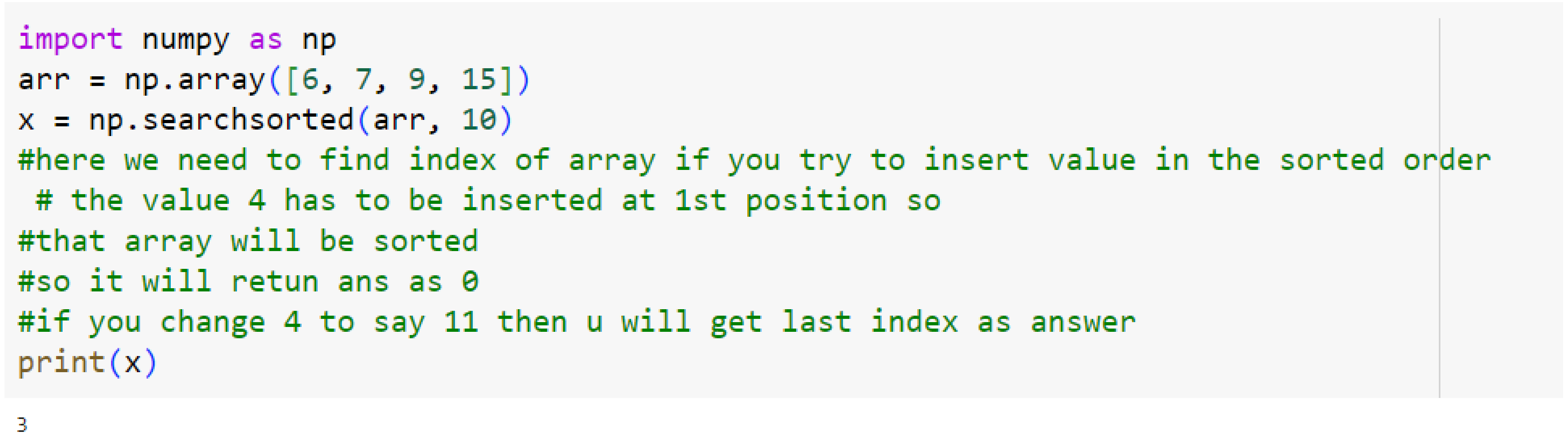
```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
#extract index of an array if array contetc is 4
print(x)
```

```
(array([3, 5, 6]),)
```



```python
import numpy as np
arr = np.array([11, 12, 53, 64, 55, 69, 87, 98])
x = np.where(arr%2 == 0)   # odd/even values of index
print(x)
```

```
(array([1, 3, 7]),)
```

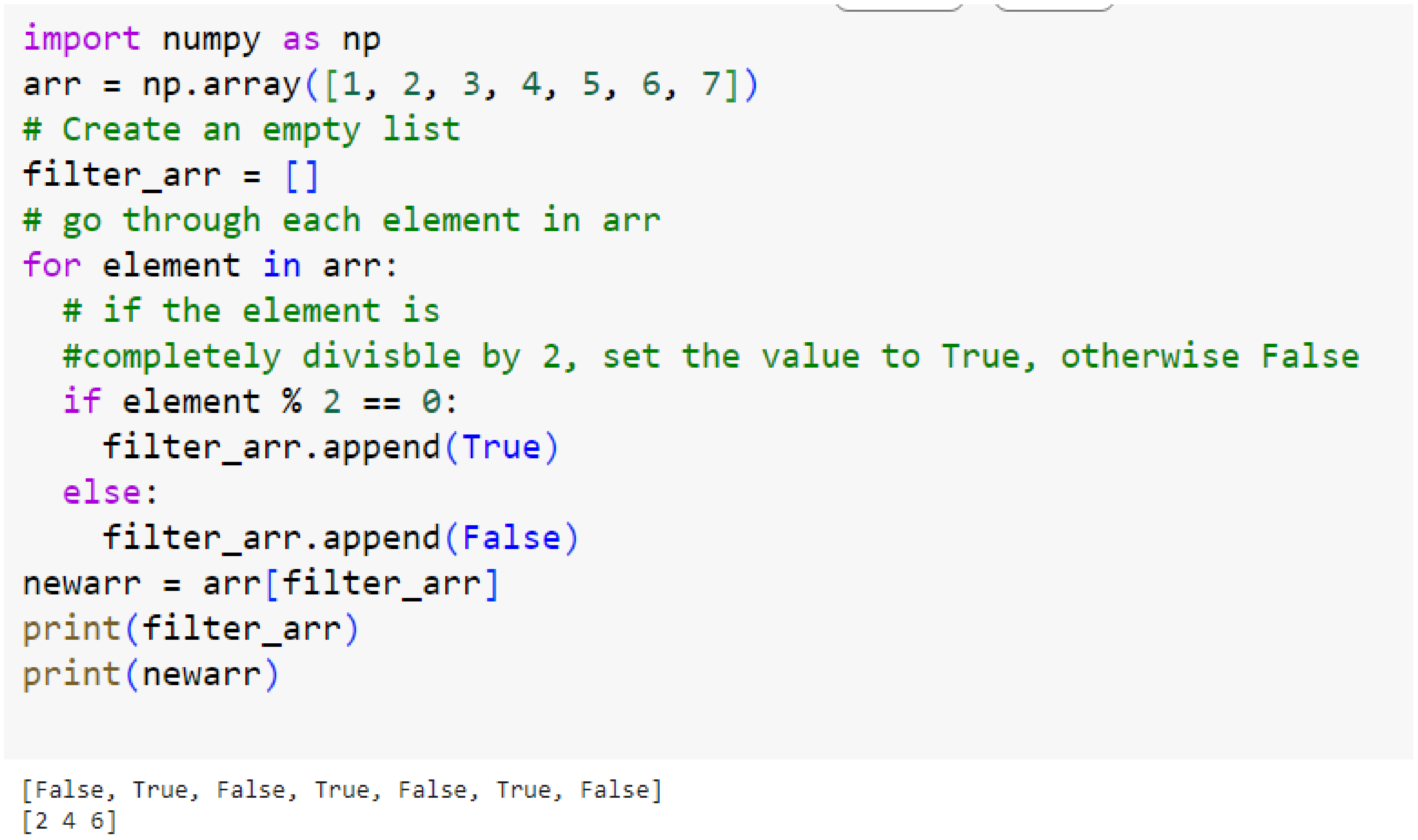


```python
import numpy as np
arr = np.array([6, 7, 9, 15])
x = np.searchsorted(arr, 10)
#here we need to find index of array if you try to insert value in the sorted order
 # the value 4 has to be inserted at 1st position so
#that array will be sorted
#so it will retun ans as 0
#if you change 4 to say 11 then u will get last index as answer
print(x)
```

```
3
```

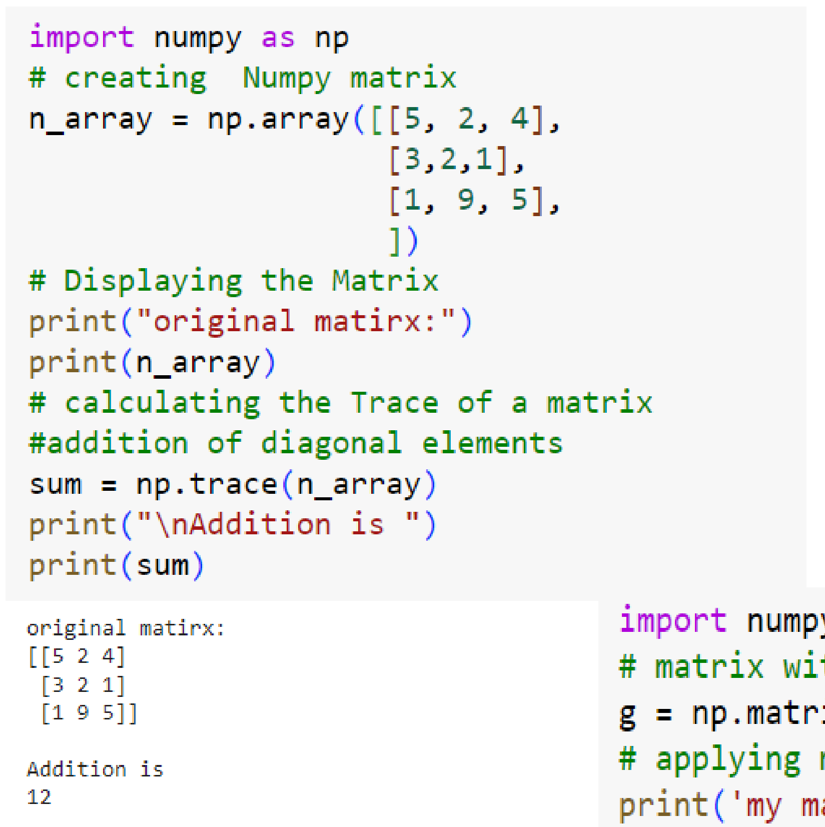
```python
#create array ; extract only elements with a condition
import numpy as np
arr = np.array([66, 87, 98, 91])
# Create an empty list
filter_arr = []
# go through each element in arr
for element in arr:
  # if the element is higher than 91,
  #set the value to True, otherwise False:
  if element > 90:
    filter_arr.append(True)
  else:
    filter_arr.append(False)
newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
[False, False, True, True]
[98 91]
```



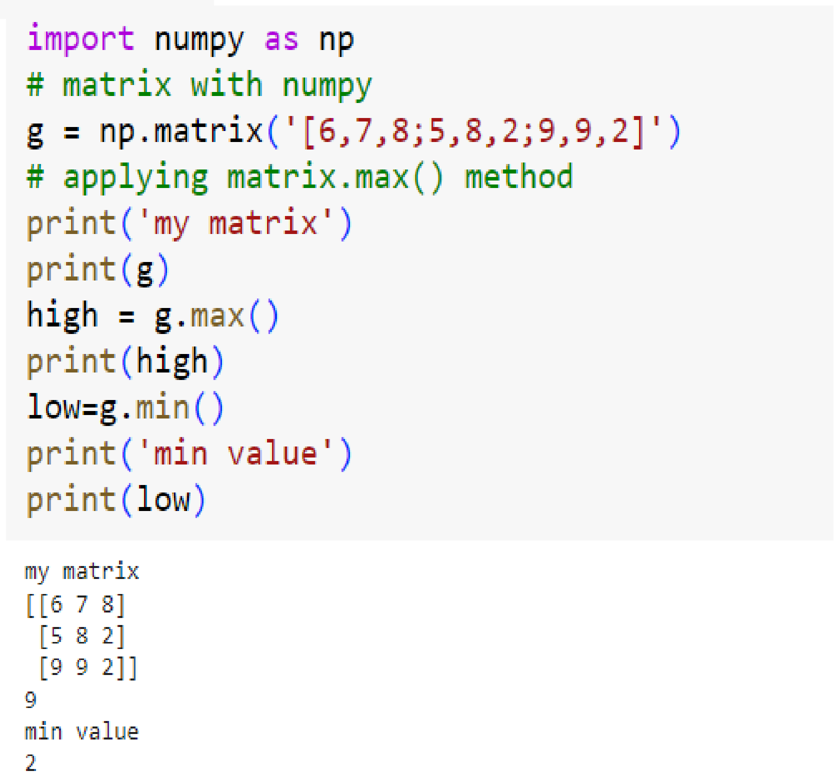
```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
# Create an empty list
filter_arr = []
# go through each element in arr
for element in arr:
  # if the element is
  #completely divisble by 2, set the value to True, otherwise False
  if element % 2 == 0:
    filter_arr.append(True)
  else:
    filter_arr.append(False)
newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```

```
[False, True, False, True, False, True, False]
[2 4 6]
```



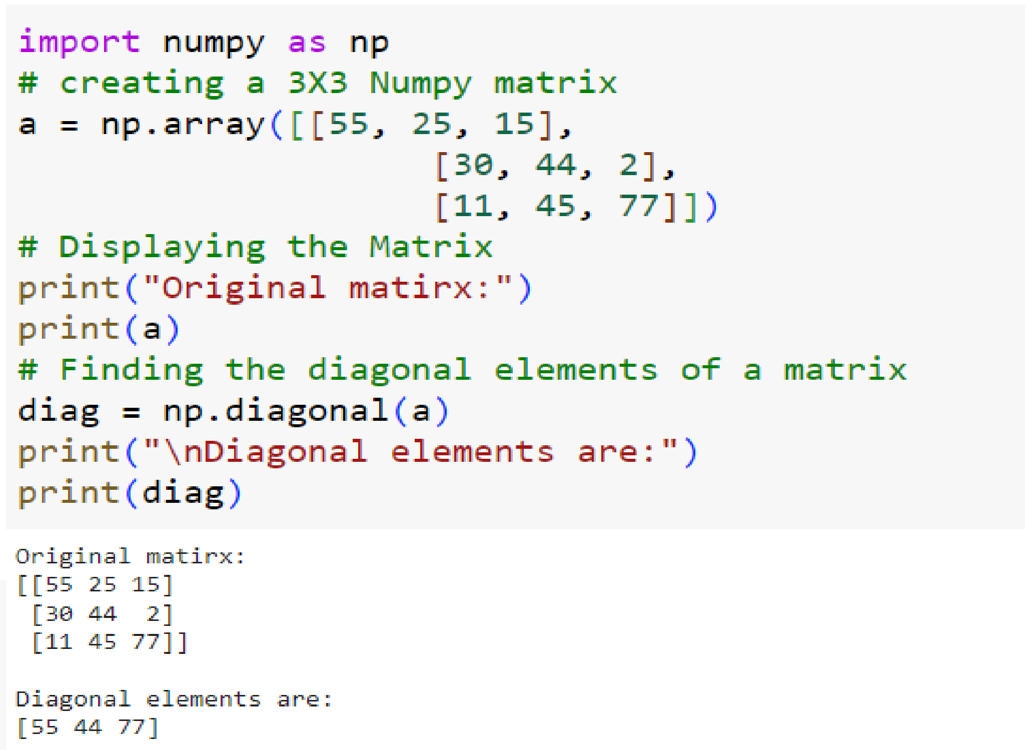
```
import numpy as np
# creating  Numpy matrix
n_array = np.array([[5, 2, 4],
                    [3,2,1],
                    [1, 9, 5],
                    ])
# Displaying the Matrix
print("original matirx:")
print(n_array)
# calculating the Trace of a matrix
#addition of diagonal elements
sum = np.trace(n_array)
print("\nAddition is ")
print(sum)
```

```
original matirx:
[[5 2 4]
 [3 2 1]
 [1 9 5]]

Addition is
12
```



```
import numpy as np
# matrix with numpy
g = np.matrix('[6,7,8;5,8,2;9,9,2]')
# applying matrix.max() method
print('my matrix')
print(g)
high = g.max()
print(high)
low=g.min()
print('min value')
print(low)
```

```
my matrix
[[6 7 8]
 [5 8 2]
 [9 9 2]]
9
min value
2
```



```
import numpy as np
# creating a 3X3 Numpy matrix
a = np.array([[55, 25, 15],
              [30, 44, 2],
              [11, 45, 77]])
# Displaying the Matrix
print("Original matirx:")
print(a)
# Finding the diagonal elements of a matrix
diag = np.diagonal(a)
print("\nDiagonal elements are:")
print(diag)
```

```
Original matirx:
[[55 25 15]
 [30 44  2]
 [11 45 77]]

Diagonal elements are:
[55 44 77]
```

```python
#Addition of Matrix and Substration of matrix
import numpy as np
# creating first matrix
A = np.array([[10, 20], [30, 40]])
# creating second matrix
B = np.array([[1, 1], [1, 1]])
print("matrix A")
print(A)
print(" matrix B")
print(B)
# adding two matrix
print("Addition of two matrix")
print(np.add(A, B))
print("Subtraction of two matrix")
print(np.subtract(A, B))
```

```
matrix A
[[10 20]
 [30 40]]
 matrix B
[[1 1]
 [1 1]]
Addition of two matrix
[[11 21]
 [31 41]]
Subtraction of two matrix
[[ 9 19]
 [29 39]]
```

```python
import numpy as np
x = np.random.randint(low=10, high=30, size=11)
print(x)
```

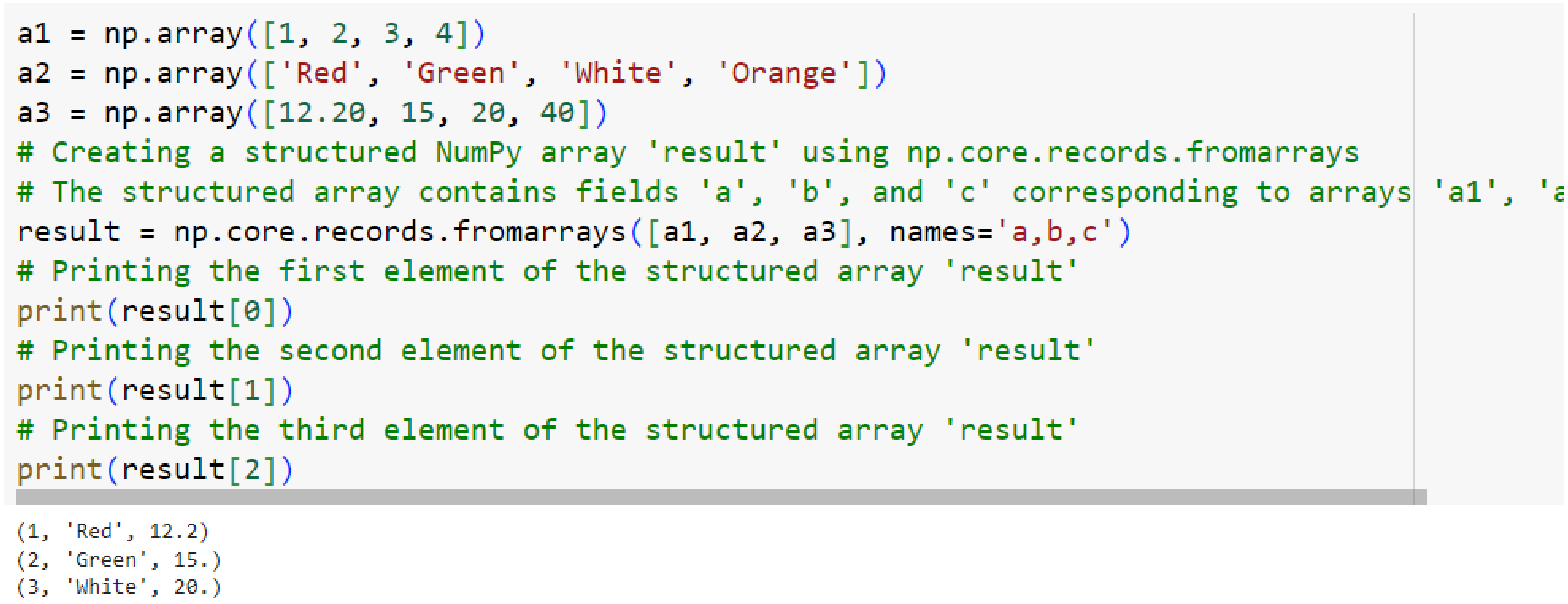
```
[19 11 13 13 14 13 21 27 20 20 12]
```

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

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

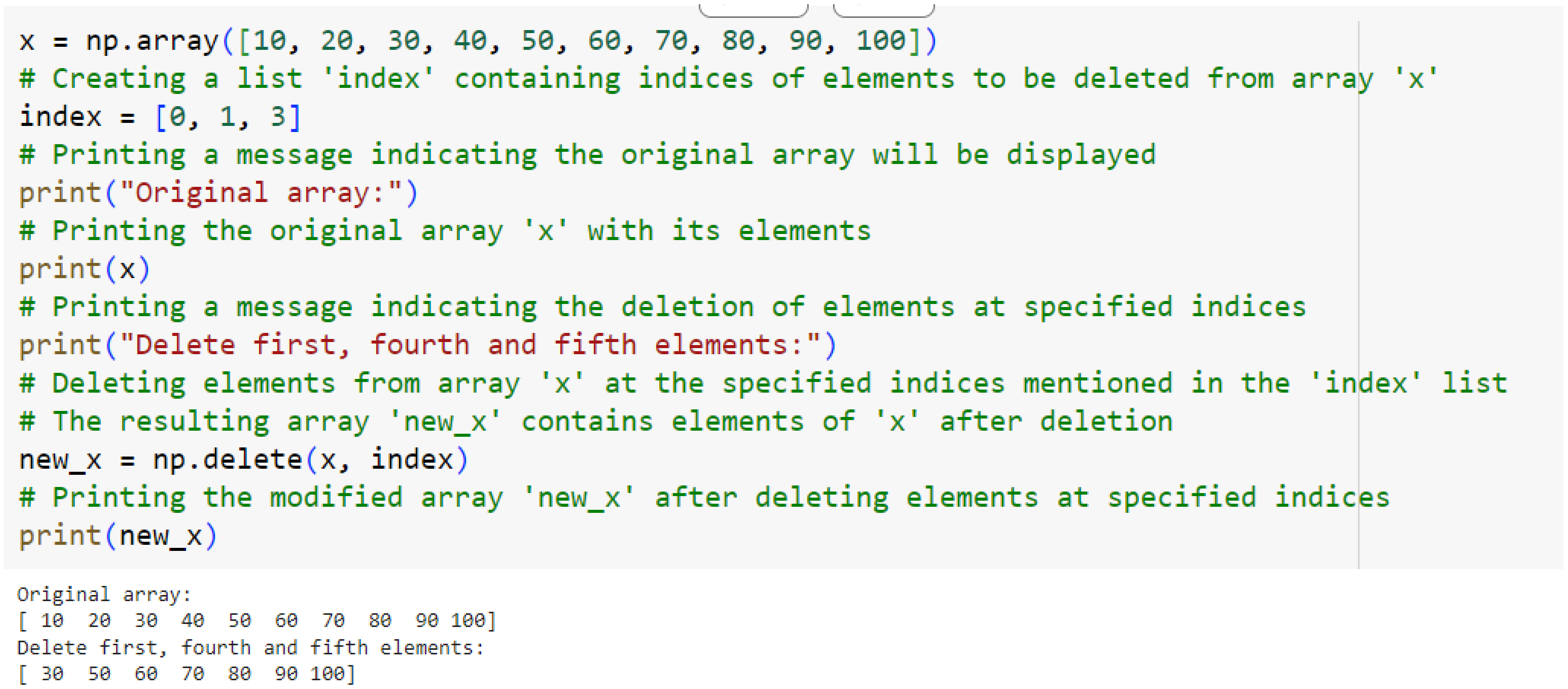
```python
x = np.array([1, 2, 3, 4, 5, 6])
# Displaying the original array
print("Original array: ", x)
# Finding the index of the maximum value in the array
print("Maximum Values: ", np.argmax(x))
# Finding the index of the minimum value in the array
print("Minimum Values: ", np.argmin(x))
print("Minimum Values: ", np.min(x))
print("Minimum Values: ", np.max(x))
```

```
Original array:  [1 2 3 4 5 6]
Maximum Values:  5
Minimum Values:  0
Minimum Values:  1
Minimum Values:  6
```



```
a1 = np.array([1, 2, 3, 4])
a2 = np.array(['Red', 'Green', 'White', 'Orange'])
a3 = np.array([12.20, 15, 20, 40])
# Creating a structured NumPy array 'result' using np.core.records.fromarrays
# The structured array contains fields 'a', 'b', and 'c' corresponding to arrays 'a1', 'a
result = np.core.records.fromarrays([a1, a2, a3], names='a,b,c')
# Printing the first element of the structured array 'result'
print(result[0])
# Printing the second element of the structured array 'result'
print(result[1])
# Printing the third element of the structured array 'result'
print(result[2])
```

```
(1, 'Red', 12.2)
(2, 'Green', 15.)
(3, 'White', 20.)
```

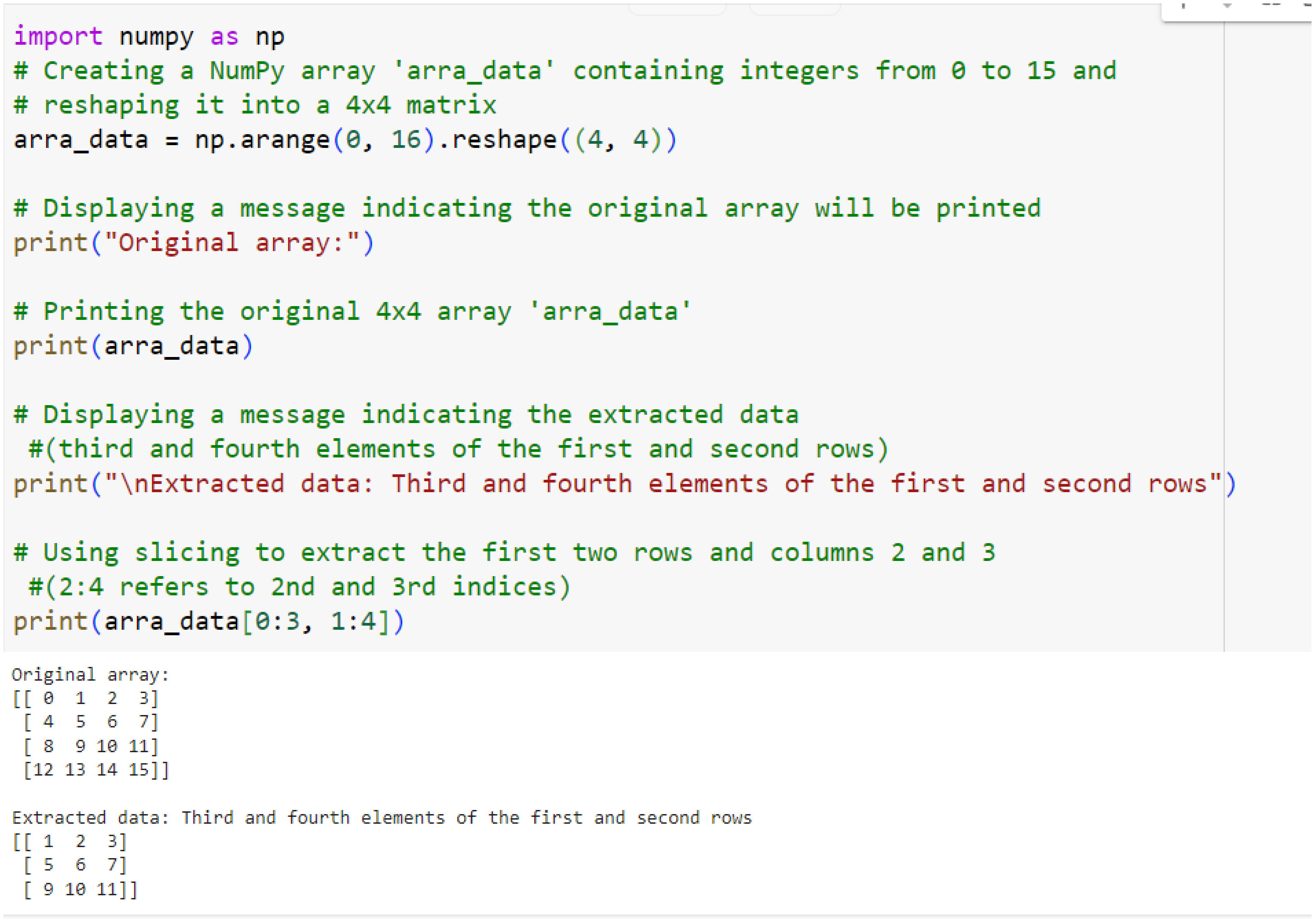


```python
x = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
# Creating a list 'index' containing indices of elements to be deleted from array 'x'
index = [0, 1, 3]
# Printing a message indicating the original array will be displayed
print("Original array:")
# Printing the original array 'x' with its elements
print(x)
# Printing a message indicating the deletion of elements at specified indices
print("Delete first, fourth and fifth elements:")
# Deleting elements from array 'x' at the specified indices mentioned in the 'index' list
# The resulting array 'new_x' contains elements of 'x' after deletion
new_x = np.delete(x, index)
# Printing the modified array 'new_x' after deleting elements at specified indices
print(new_x)
```

```
Original array:
[ 10  20  30  40  50  60  70  80  90 100]
Delete first, fourth and fifth elements:
[ 30  50  60  70  80  90 100]
```

```python
x = np.array([10, 20, 30])

# Printing a message indicating the original array will be displayed
print("Original array:")

# Printing the original array 'x' with its elements
print(x)

# Printing a message indicating the sum of the array elements will be displayed
print("Sum of the array elements:")

# Calculating and printing the sum of the elements in the array 'x' using 'sum()'
print(x.sum())

# Printing a message indicating the product of the array elements will be displayed
print("Product of the array elements:")

# Calculating and printing the product of the elements in the array 'x' using 'prod()'
print(x.prod())
```

```
Original array:
[10 20 30]
Sum of the array elements:
60
Product of the array elements:
6000
```



```
import numpy as np
# Creating a NumPy array 'arra_data' containing integers from 0 to 15 and
# reshaping it into a 4x4 matrix
arra_data = np.arange(0, 16).reshape((4, 4))

# Displaying a message indicating the original array will be printed
print("Original array:")

# Printing the original 4x4 array 'arra_data'
print(arra_data)

# Displaying a message indicating the extracted data
 #(third and fourth elements of the first and second rows)
print("\nExtracted data: Third and fourth elements of the first and second rows")

# Using slicing to extract the first two rows and columns 2 and 3
 #(2:4 refers to 2nd and 3rd indices)
print(arra_data[0:3, 1:4])
```

```
Original array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

Extracted data: Third and fourth elements of the first and second rows
[[ 1  2  3]
 [ 5  6  7]
 [ 9 10 11]]
```

```
#example 1
import numpy as np
a = np.array([1,2,3])
b=np.array(['cat','dog'])
print (a)
print (b)
```

```
[1 2 3] ['cat' 'dog']
```

```
#example 2
#usage of complex
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print (a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```

```
#example 3  ; usage of dtype
# using array-scalar type
import numpy as np
dt = np.dtype(np.int32)
print (dt)
```

```
int32
```

```
#create a structure of student with 3 parameters
import numpy as np

student = np.dtype([('name','S20'), ('age','i2'), ('marks','f4')])
#here s20 implies 20 charaters , i2->> 2 bytes of interger , f ->> float
a1=np.array([('raj', 21, 50),('rahul', 18, 75)], dtype = student)
print (a1)
```

```
[(b'raj', 21, 50.) (b'rahul', 18, 75.)]
```

```
#reshape  -> to print dimension of arry 2 X 3
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print (a.shape)
```

```
(2, 3)
```

```
# this resizes the ndarray
import numpy as np

a = np.array([[1,2,3],[4,5,6]])
print(a)
```

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

```
a.shape = (3,2)
print (a )
```

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

```
a.shape=(2,3)
print('2X3 matrix' ,a)
#will create a sequence from 0 to the
parameter specified
c=np.arange(20)
print(c)
d=np.arange(2,10,2)
print(d)
```

```
2X3 matrix [[1 2 3]
```

```
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]
```

```
[2 4 6 8]
```

```
# creating an array of only 0's. Default
dtype is float
import numpy as np
x = np.zeros(5)
print (x)
```

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

```
#  creating an array of only 1's. Default dtype is float
import numpy as np
x = np.ones(5)
print (x)
y=np.ones([2,2],dtype=int)
print(y)
```

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

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

```
#used to plot data from 10 to 20 with 8 values
import numpy as np
x = np.linspace(10,20,8)
print (x)
```

```
[10. 11.42857143 12.85714286 14.28571429
15.71428571 17.14285714 18.57142857 20.
```

```
 #used to cut array based on start - stop -
 step
 import numpy as np
 a = np.arange(10)
 s = slice(2,7,3)
 print(a[s])

 print('alternate way')

 a = np.arange(10)
 b = a[2:7:2]
 print (b)
```

```
[2 5]

alternate way

[2 4 6]
```

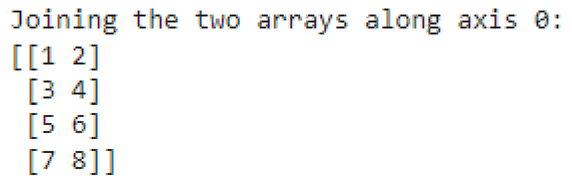
```
#to print trasnpose
import numpy as np
a = np.arange(12).reshape(3,4)

print('The original array is:' )
print (a)
print ('\n')

print ('The transposed array is:' )
print (np.transpose(a))
```

```
The original array is: [[ 0 1 2 3]
[ 4 5 6 7]
[ 8 9 10 11]]
 The transposed array is:
[[ 0 4 8]
[ 1 5 9]
[ 2 6 10]
 [ 3 7 11]]
```

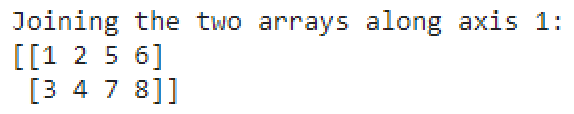
```
#concat of array
import numpy as np
a = np.array([[1,2],[3,4]])

print ('First array:' )
print (a)
print ('\n')
b = np.array([[5,6],[7,8]])

print ('Second array:' )
print (b)
print ('\n')
# both the arrays are of same dimensions

print ('Joining the two arrays along axis
0:' )
print (np.concatenate((a,b)) )
print ('\n')

print ('Joining the two arrays along axis
1:' )
print (np.concatenate((a,b),axis = 1))
```

```
First array:
[[1 2]
[3 4]]
```

```
Second array:
 [[5 6]
 [7 8]]
```

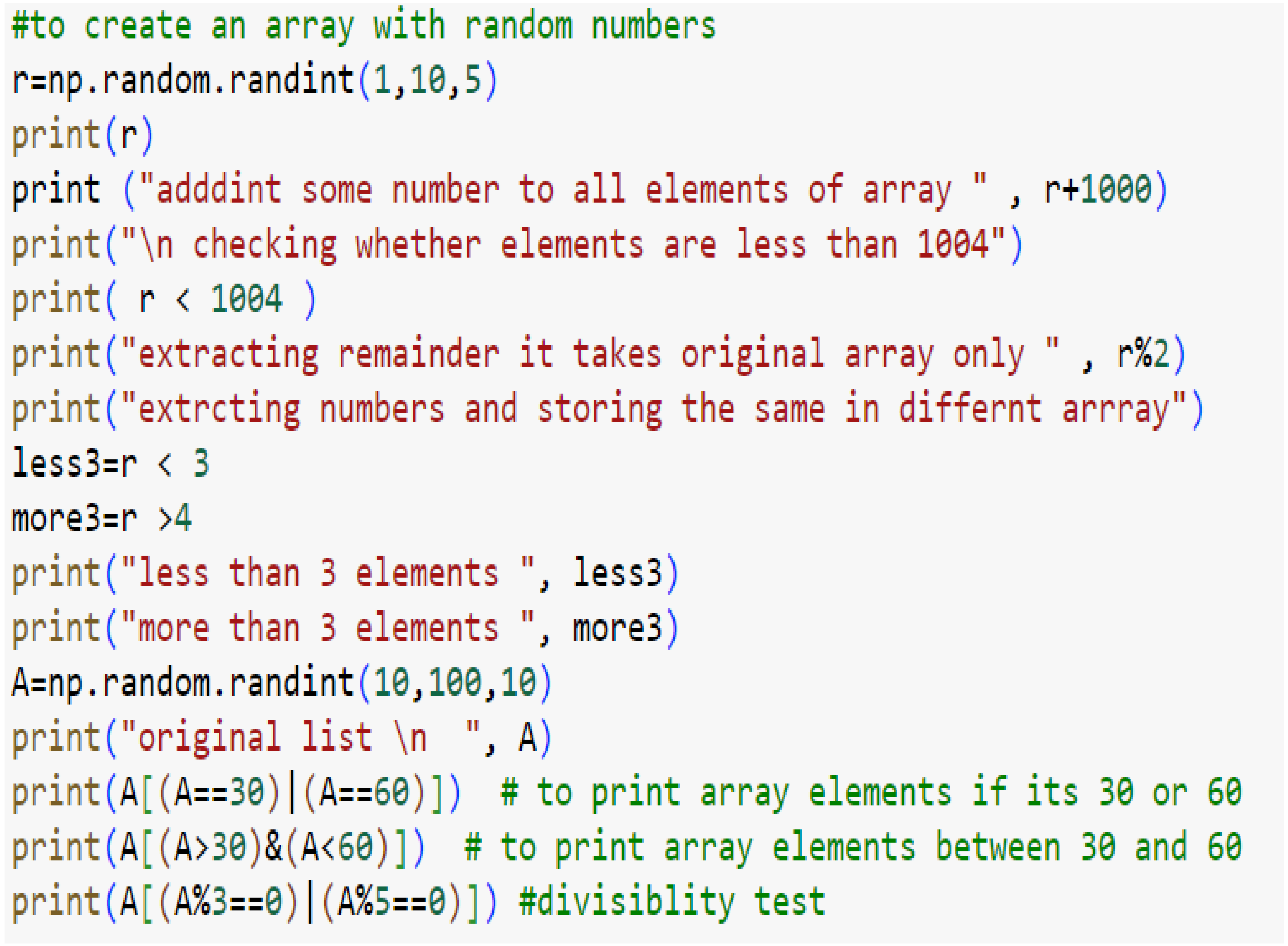


```
Joining the two arrays along axis 0:
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

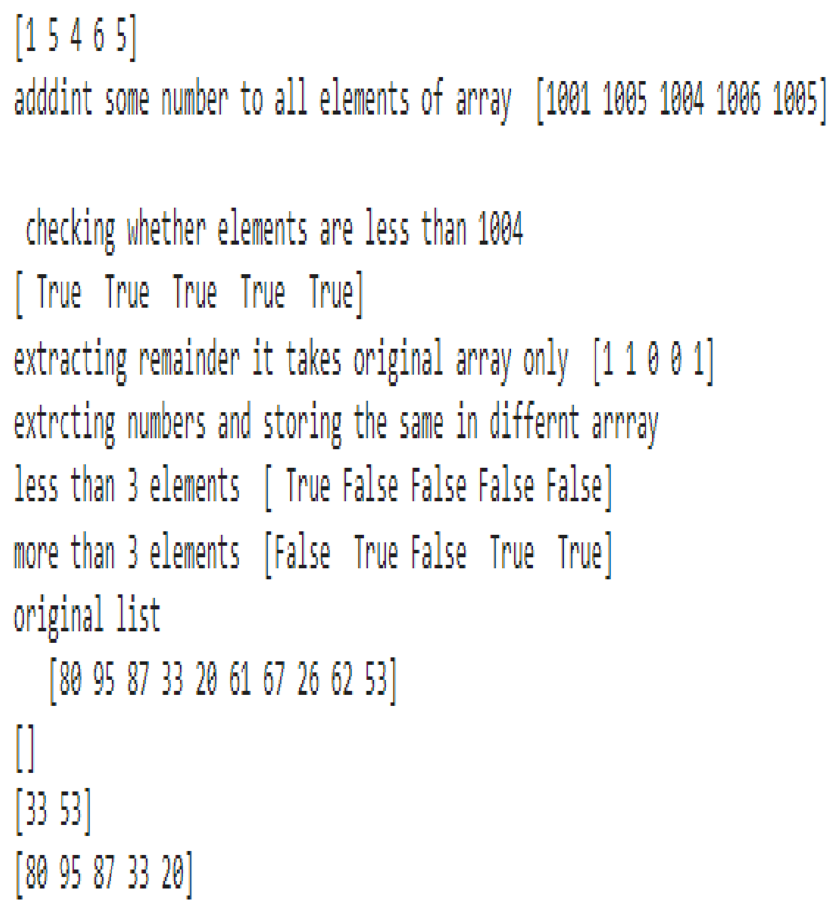


```
Joining the two arrays along axis 1:
[[1 2 5 6]
 [3 4 7 8]]
```

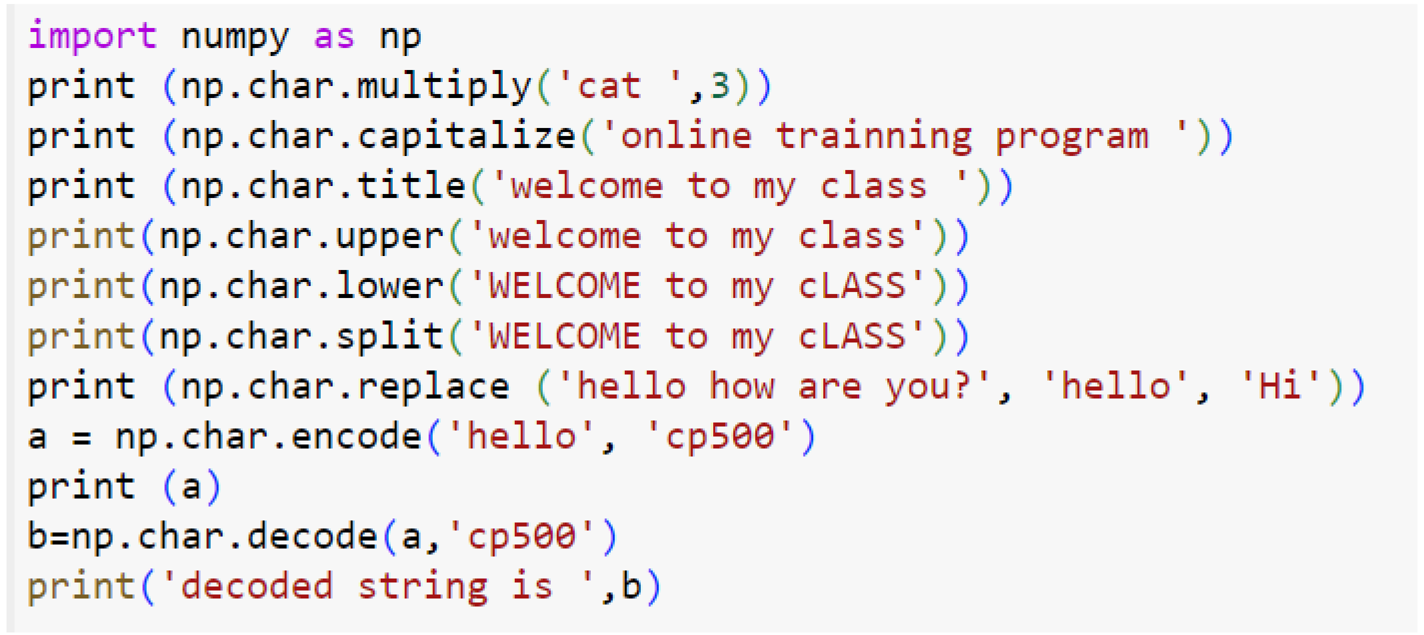
```python
import numpy as np
```



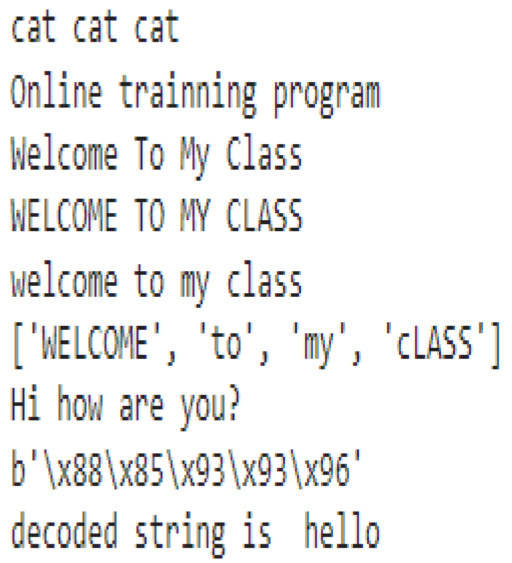
```python
#to create an array with random numbers
r=np.random.randint(1,10,5)
print(r)
print ("adddint some number to all elements of array " , r+1000)
print("\n checking whether elements are less than 1004")
print( r < 1004 )
print("extracting remainder it takes original array only " , r%2)
print("extrcting numbers and storing the same in differnt arrray")
less3=r < 3
more3=r >4
print("less than 3 elements ", less3)
print("more than 3 elements ", more3)
A=np.random.randint(10,100,10)
print("original list \n  ", A)
print(A[(A==30)|(A==60)])  # to print array elements if its 30 or 60
print(A[(A>30)&(A<60)])  # to print array elements between 30 and 60
print(A[(A%3==0)|(A%5==0)]) #divisiblity test
```



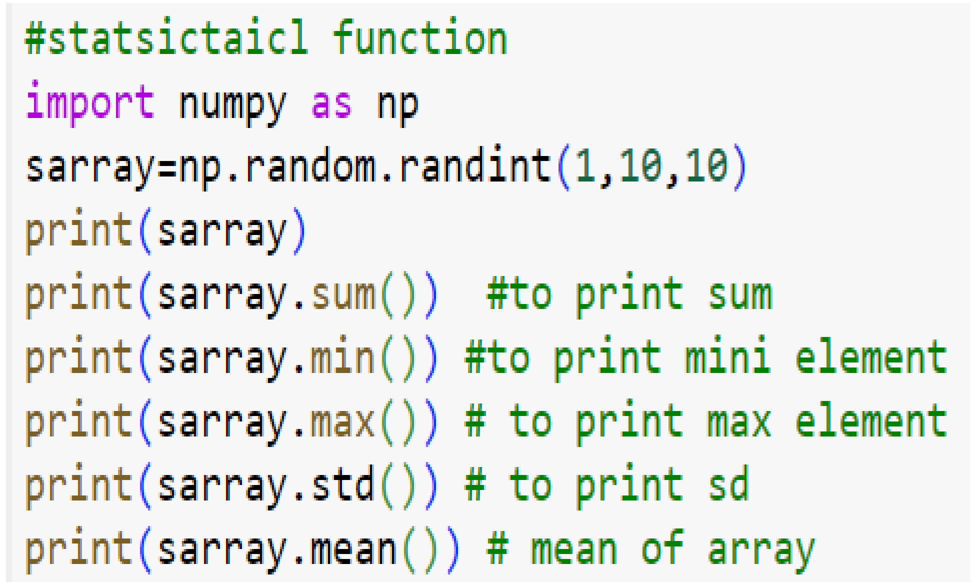
```
[1 5 4 6 5]
adddint some number to all elements of array  [1001 1005 1004 1006 1005]

 checking whether elements are less than 1004
[ True  True  True  True  True]
extracting remainder it takes original array only  [1 1 0 0 1]
extrcting numbers and storing the same in differnt arrray
less than 3 elements  [ True False False False False]
more than 3 elements  [False  True False  True  True]
original list
   [80 95 87 33 20 61 67 26 62 53]
[]
[33 53]
[80 95 87 33 20]
```



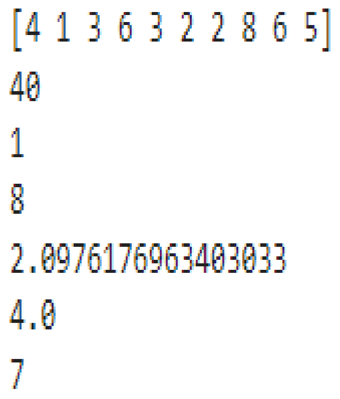
```python
import numpy as np
print (np.char.multiply('cat ',3))
print (np.char.capitalize('online trainning program '))
print (np.char.title('welcome to my class '))
print(np.char.upper('welcome to my class'))
print(np.char.lower('WELCOME to my cLASS'))
print(np.char.split('WELCOME to my cLASS'))
print (np.char.replace ('hello how are you?', 'hello', 'Hi'))
a = np.char.encode('hello', 'cp500')
print (a)
b=np.char.decode(a,'cp500')
print('decoded string is ',b)
```



```
cat cat cat
Online trainning program
Welcome To My Class
WELCOME TO MY CLASS
welcome to my class
['WELCOME', 'to', 'my', 'cLASS']
Hi how are you?
b'\x88\x85\x93\x93\x96'
decoded string is  hello
```



```python
#statsictaicl function
import numpy as np
sarray=np.random.randint(1,10,10)
print(sarray)
print(sarray.sum())  #to print sum
print(sarray.min()) #to print mini element
print(sarray.max()) # to print max element
print(sarray.std()) # to print sd
print(sarray.mean()) # mean of array
```



```
[4 1 3 6 3 2 2 8 6 5]
40
1
8
2.0976176963403033
4.0
7
```

```python
#demonstration of axis
#lets create matrix
import numpy as np
mat = np.random.randint(10,90,(4,4))
mat
```

```
array([[34, 51, 46, 83],
       [28, 77, 47, 41],
       [87, 52, 50, 40],
       [40, 45, 17, 65]])
```

```python
print(mat.max(axis=1)) # as axis=1 is for vertical so from the given matrix it will extraact
#max element from each row
print("extracting max element from horizontal/ rowwise position:", mat.max(axis=0))
```

```
[83 77 87 65]
extracting max element from horizontal position: [87 77 50 83]
```

```python
#working with linear algebra
import  numpy as np
#create an array 3X3 with random values from 1 to 15
B = np.random.randint(1,15,(3,3))
B
```

```
array([[ 5,  5, 13],
       [11,  9, 11],
       [12,  6, 12]])
```

```python
print("transpose - Row to col and col to row")
print(B.T)
```

```
transpose - Row to col and col to row
[[ 5 11 12]
 [ 5  9  6]
 [13 11 12]]
```

```
#lets crate another matrix with 3X4
mat = np.random.randint(10,90,(3,4))
mat
```

```
array([[15, 17, 79, 18],
       [52, 52, 69, 56],
       [63, 68, 61, 70]])
```

```
#working with linear algebra
import  numpy as np
#create an array 3X3 with random values from 1 to 15
B = np.random.randint(1,15,(3,3))
B
```

Created already!

```
#perform matrix multiplication
print(mat.shape, B.shape)  #just printing shape of matrix
```

```
B.dot(mat) # matrix multiplication
```

```
array([[1154, 1229, 1533, 1280],
       [1326, 1403, 2161, 1472],
       [1248, 1332, 2094, 1392]])
```

```
np.dot(B,mat) #another way!..
#need to check order as in the condition need for matrix multiplication
```

```
array([[1154, 1229, 1533, 1280],
       [1326, 1403, 2161, 1472],
       [1248, 1332, 2094, 1392]])
```

```
print(B)
print(B*B)  #elementwise multiplication ; like addition
```

```
[[ 5  5 13]
 [11  9 11]
 [12  6 12]]
[[ 25  25 169]
 [121  81 121]
 [144  36 144]]
```

```
np.linalg.det(B)  # B is a squar matrix
```

```
-336.0
```

```
print("sin " ,np.sin(B))
print("cos" , np.cos(B))
```

```
sin  [[-0.95892427 -0.95892427  0.42016704]
 [-0.99999021  0.41211849 -0.99999021]
 [-0.53657292 -0.2794155  -0.53657292]]
cos [[ 0.28366219  0.28366219  0.90744678]
 [ 0.0044257  -0.91113026  0.0044257 ]
 [ 0.84385396  0.96017029  0.84385396]]
```

```
print(B)
print(np.where(B%2==0, 'Even', 'Odd'))
```

```
[[ 5  5 13]
 [11  9 11]
 [12  6 12]]
[['Odd' 'Odd' 'Odd']
 ['Odd' 'Odd' 'Odd']
 ['Even' 'Even' 'Even']]
```

to extract common elements from 2 arrays

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
np.intersect1d(a,b)
```

```
array([2, 4])
```

Get the positions where elements of a and b match a = [1,2,3,2,3,4,3,4,5,6] b = [7,2,10,2,7,4,9,4,9,8]
here the index will start from 0 so on 1st position element 2 is prenset on A and B on 5th positiosn element 4 is present on A and B so it will return all index where A==B

```
a = np.array([10,20,3,2,3,14,3,4,5,6])
b = np.array([7,20,10,2,7,41,9,4,9,8])

np.where(a == b)
```

```
(array([1, 3, 7]),)
```

set difference a =1,2,3,4,5 b = 5,6,7,8,9
Return the unique values in a that are not in b.
o/p ->> [ 1,2,3,4 ]

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

np.setdiff1d(a,b)
```

```
array([1, 2, 4])
```

```
a = np.array([10,20,3,2,3,14,3,4,5,6])
print(a)
print('mean ')
print(np.mean(a))
print('median')
print(np.median(a))
print('std diviation')
print(np.std(a))
```

```
[10 20  3  2  3 14  3  4  5  6]
mean
7.0
median
4.5
std diviation
5.60357029044876
```

```
from numpy import random

x=random.randint(100, size=(18))

print(x)
```

```
[45 11 92 89 24 30 87 28 50 38 86  6  2 15 66 27 60 22]
```

```
np.zeros((3,5), dtype= )  #array with 0's
```

```
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0]])
```

```
np.ones((3,2))
```

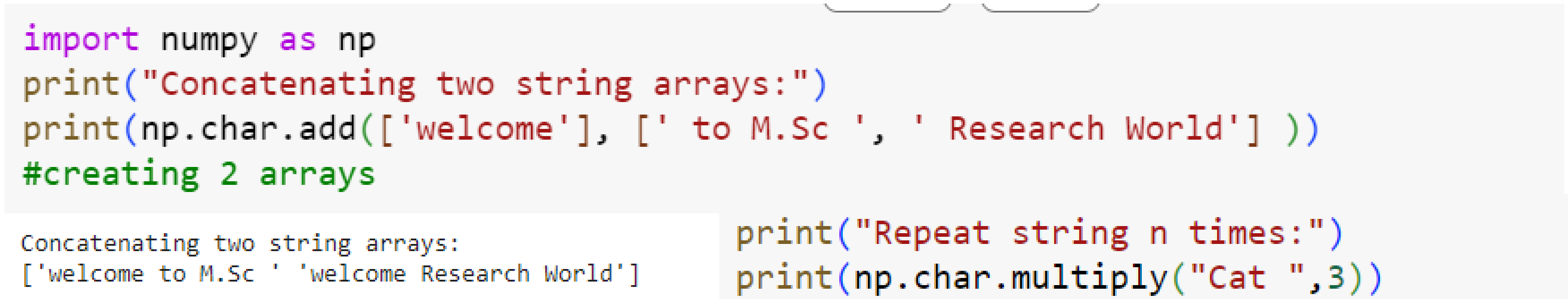
```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

```
np.random.randint(10,25,10) #random numbers between 10 to 25 ;
#10 numbers
```

```
array([22, 14, 13, 11, 23, 24, 16, 20, 14, 15])
```

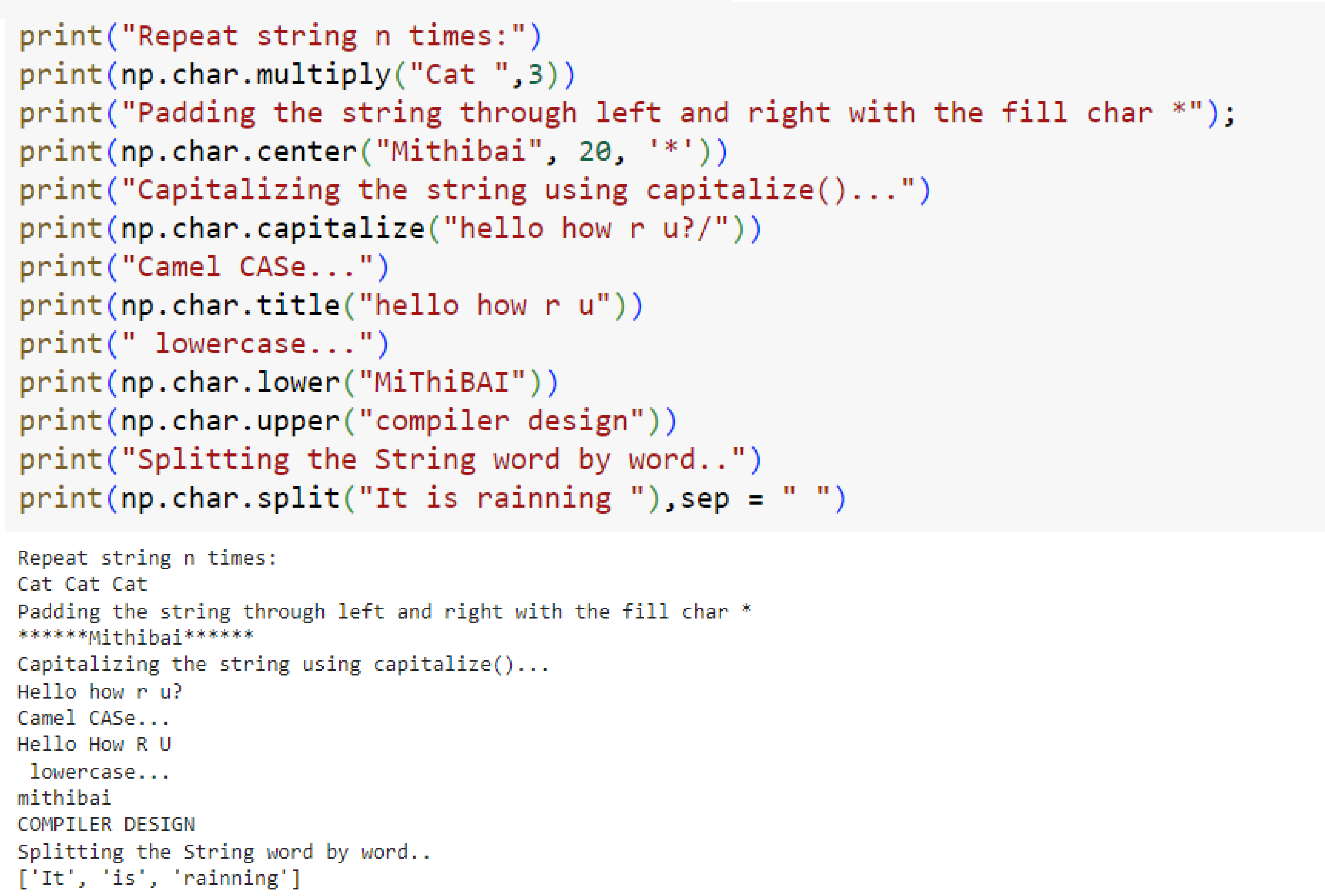
```
np.random.randint(10,25,(10,3))  # 10 arrays index 0 1 2 each
```

```
array([[16, 16, 11],
       [22, 22, 12],
       [13, 12, 13],
       [17, 14, 23],
       [16, 20, 23],
       [23, 18, 21],
       [19, 16, 24],
       [15, 17, 10],
       [10, 23, 15],
```



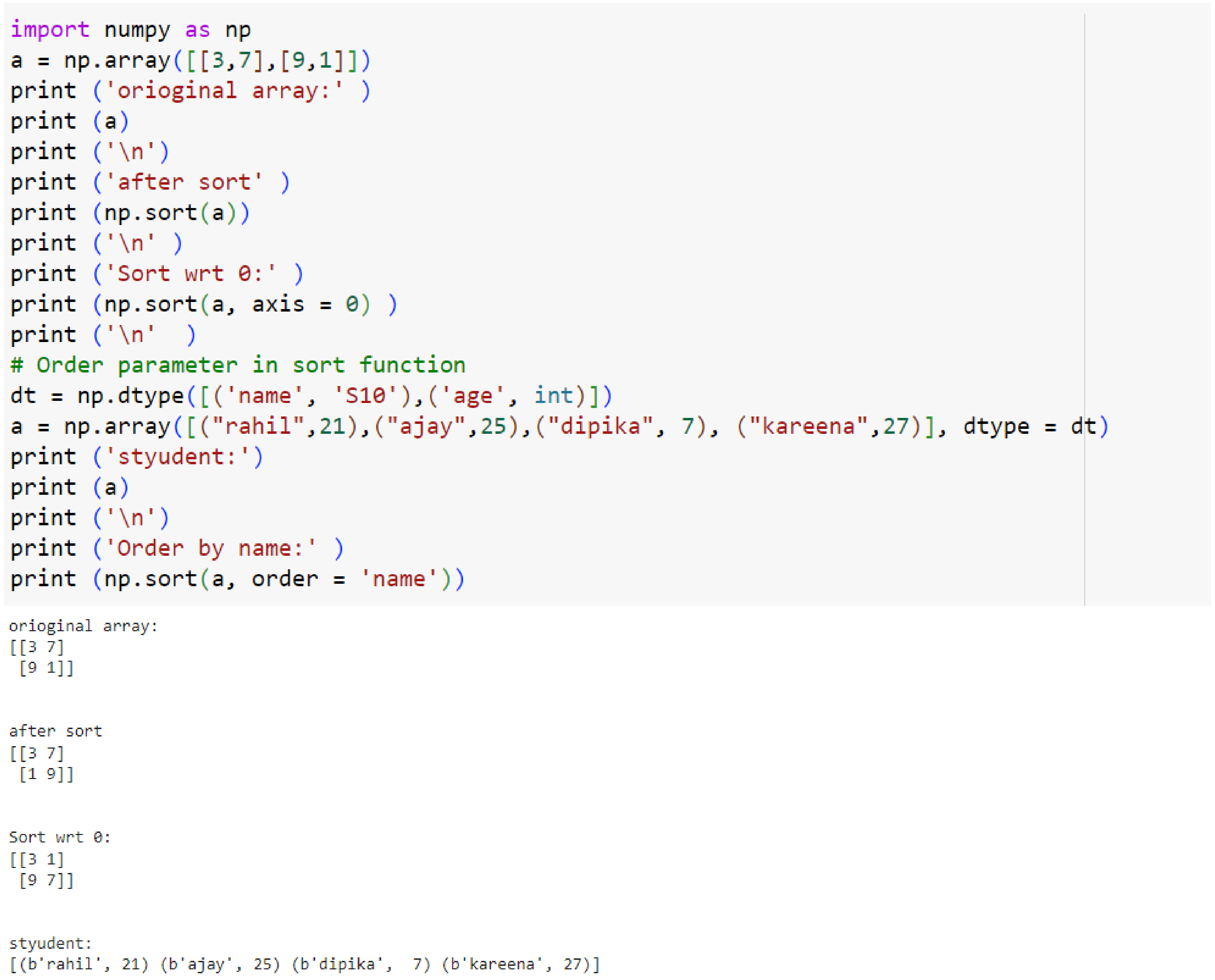
```python
import numpy as np
print("Concatenating two string arrays:")
print(np.char.add(['welcome'], [' to M.Sc ', ' Research World'] ))
#creating 2 arrays
```

```
Concatenating two string arrays:
['welcome to M.Sc ' 'welcome Research World']
```



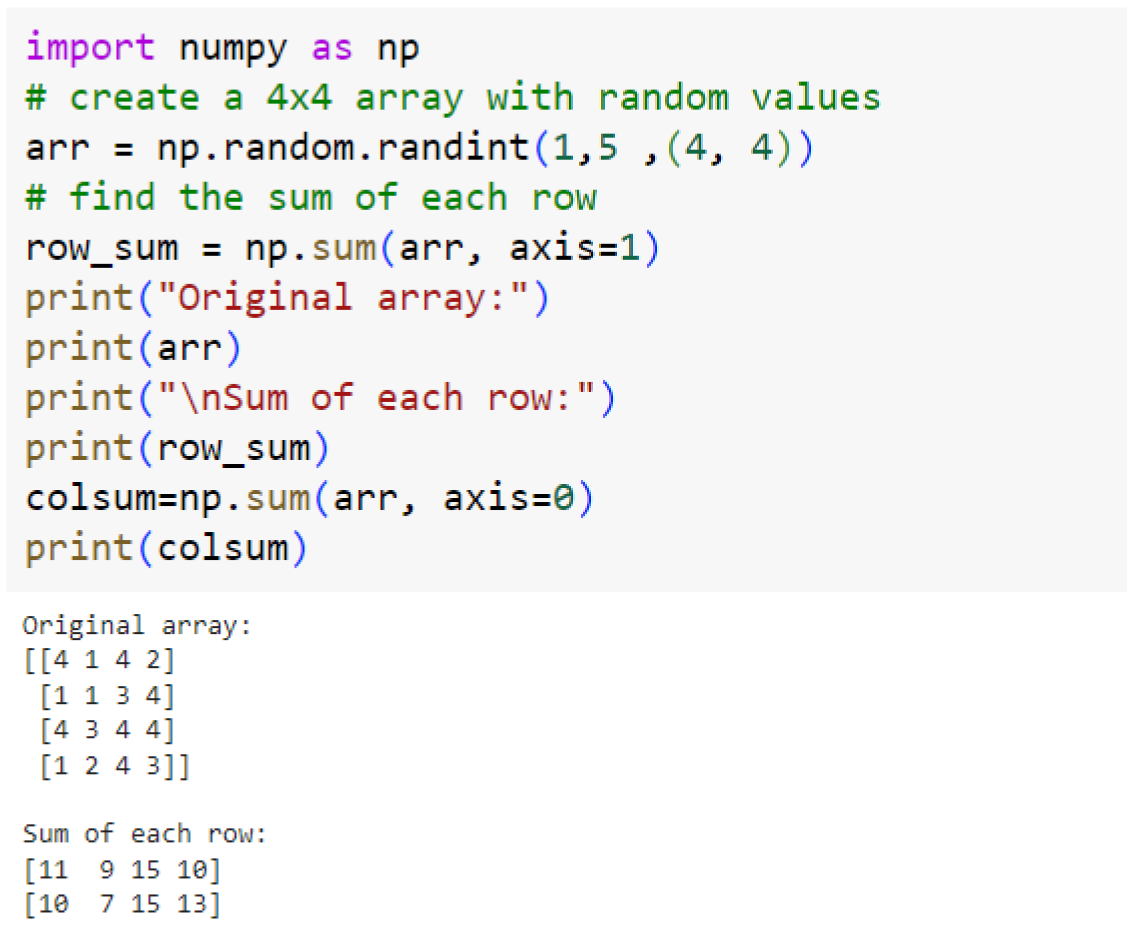
```python
print("Repeat string n times:")
print(np.char.multiply("Cat ",3))
print("Padding the string through left and right with the fill char *");
print(np.char.center("Mithibai", 20, '*'))
print("Capitalizing the string using capitalize()...")
print(np.char.capitalize("hello how r u?/"))
print("Camel CASe...")
print(np.char.title("hello how r u"))
print(" lowercase...")
print(np.char.lower("MiThiBAI"))
print(np.char.upper("compiler design"))
print("Splitting the String word by word..")
print(np.char.split("It is rainning "),sep = " ")
```

```
Repeat string n times:
Cat Cat Cat
Padding the string through left and right with the fill char *
******Mithibai******
Capitalizing the string using capitalize()...
Hello how r u?
Camel CASe...
Hello How R U
 lowercase...
mithibai
COMPILER DESIGN
Splitting the String word by word..
['It', 'is', 'rainning']
```



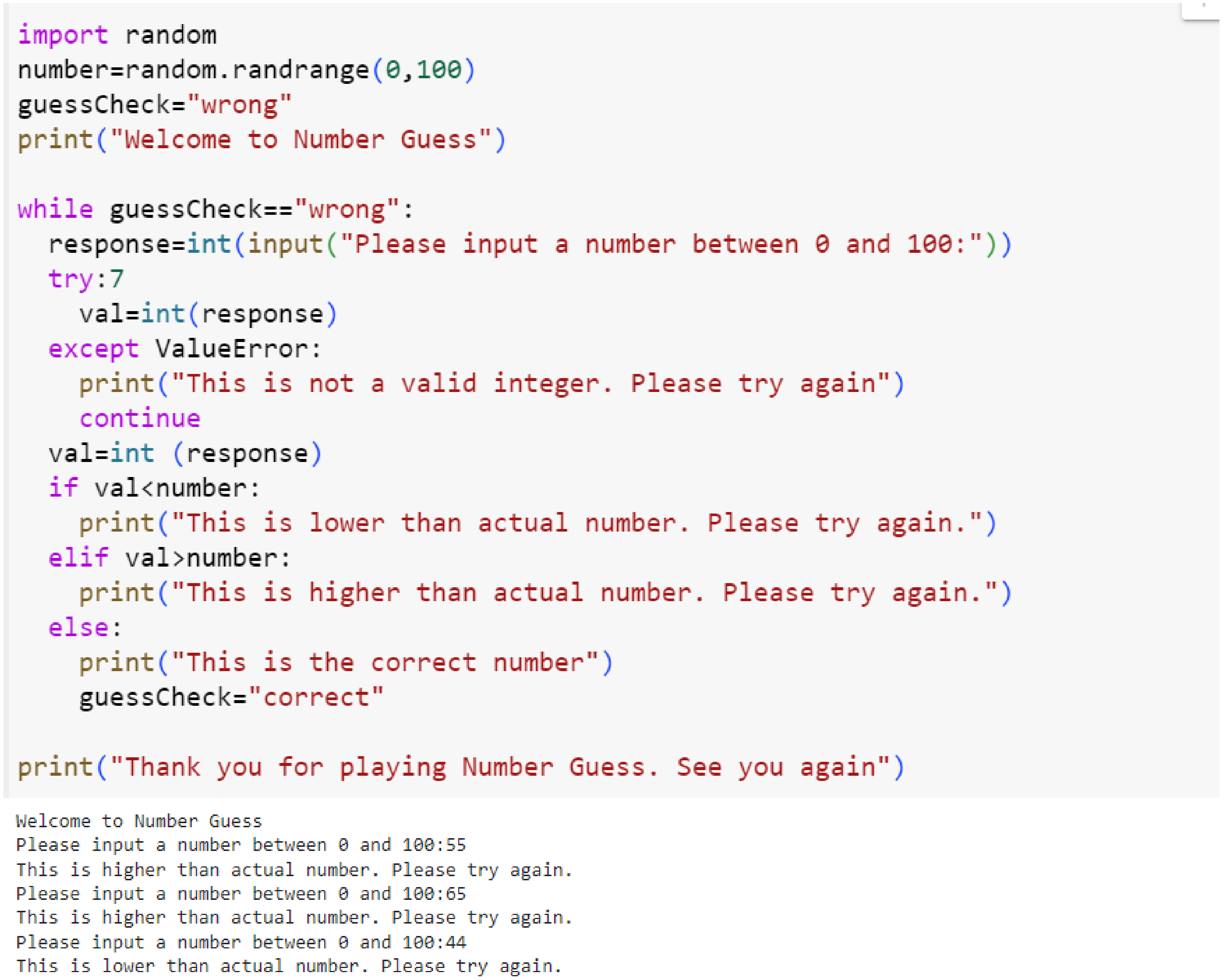
```python
import numpy as np
a = np.array([[3,7],[9,1]])
print ('orioginal array:' )
print (a)
print ('\n')
print ('after sort' )
print (np.sort(a))
print ('\n' )
print ('Sort wrt 0:' )
print (np.sort(a, axis = 0) )
print ('\n'  )
# Order parameter in sort function
dt = np.dtype([('name', 'S10'),('age', int)])
a = np.array([("rahil",21),("ajay",25),("dipika", 7), ("kareena",27)], dtype = dt)
print ('styudent:')
print (a)
print ('\n')
print ('Order by name:' )
print (np.sort(a, order = 'name'))
```

```
orioginal array:
[[3 7]
 [9 1]]


after sort
[[3 7]
 [1 9]]


Sort wrt 0:
[[3 1]
 [9 7]]


styudent:
[(b'rahil', 21) (b'ajay', 25) (b'dipika',  7) (b'kareena', 27)]
```



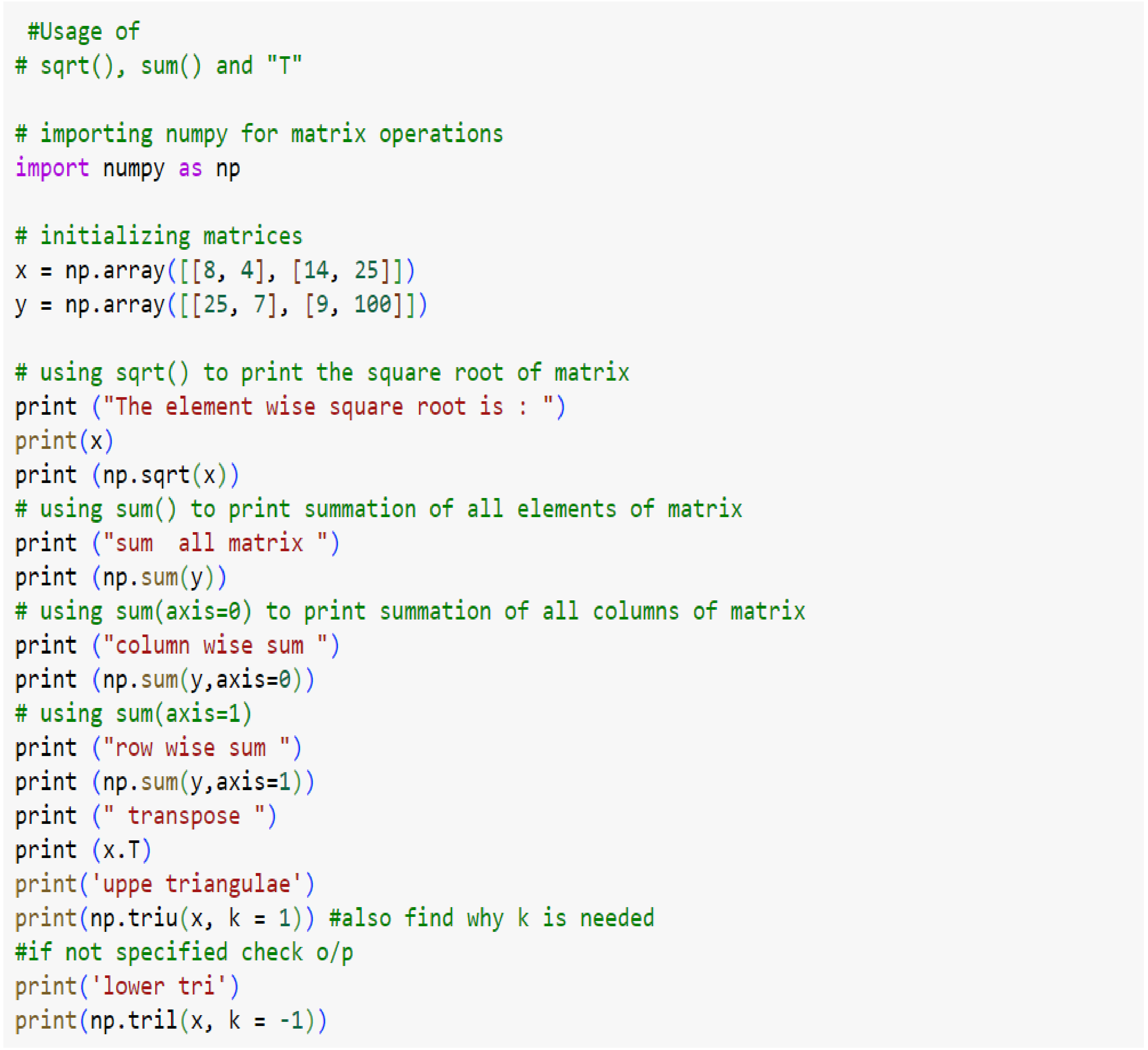
```python
import numpy as np
# create a 4x4 array with random values
arr = np.random.randint(1,5 ,(4, 4))
# find the sum of each row
row_sum = np.sum(arr, axis=1)
print("Original array:")
print(arr)
print("\nSum of each row:")
print(row_sum)
colsum=np.sum(arr, axis=0)
print(colsum)
```

```
Original array:
[[4 1 4 2]
 [1 1 3 4]
 [4 3 4 4]
 [1 2 4 3]]

Sum of each row:
[11  9 15 10]
[10  7 15 13]
```

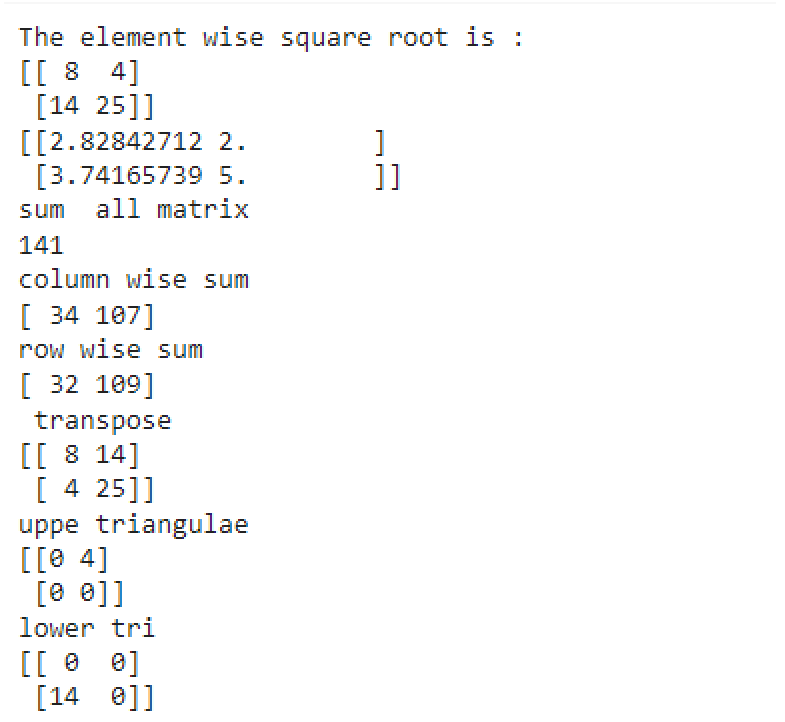


```python
import random
number=random.randrange(0,100)
guessCheck="wrong"
print("Welcome to Number Guess")

while guessCheck=="wrong":
  response=int(input("Please input a number between 0 and 100:"))
  try:7
    val=int(response)
  except ValueError:
    print("This is not a valid integer. Please try again")
    continue
  val=int (response)
  if val<number:
    print("This is lower than actual number. Please try again.")
  elif val>number:
    print("This is higher than actual number. Please try again.")
  else:
    print("This is the correct number")
    guessCheck="correct"

print("Thank you for playing Number Guess. See you again")
```

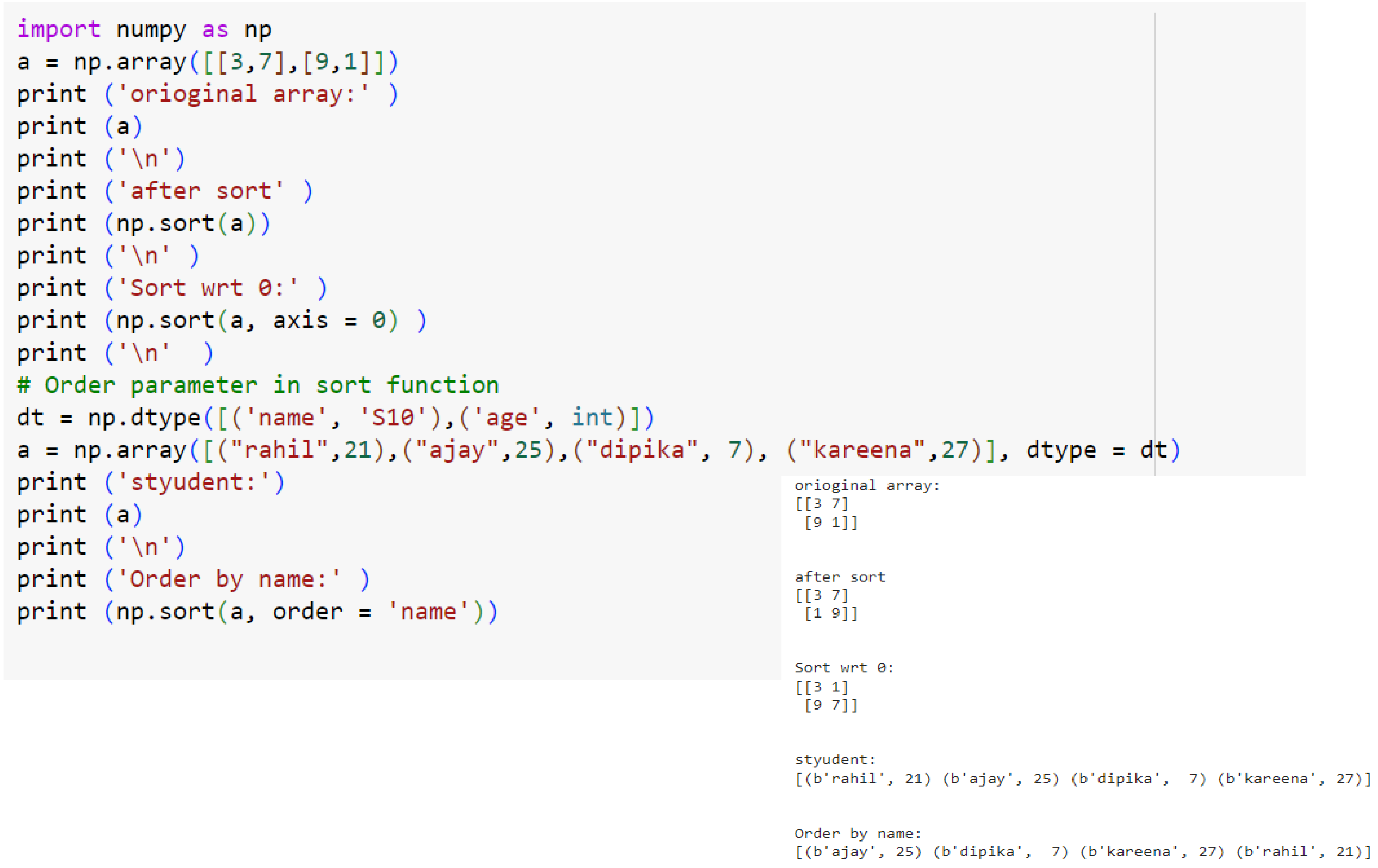
```
Welcome to Number Guess
Please input a number between 0 and 100:55
This is higher than actual number. Please try again.
Please input a number between 0 and 100:65
This is higher than actual number. Please try again.
Please input a number between 0 and 100:44
This is lower than actual number. Please try again.
```



```python
 #Usage of
# sqrt(), sum() and "T"

# importing numpy for matrix operations
import numpy as np

# initializing matrices
x = np.array([[8, 4], [14, 25]])
y = np.array([[25, 7], [9, 100]])

# using sqrt() to print the square root of matrix
print ("The element wise square root is : ")
print(x)
print (np.sqrt(x))
# using sum() to print summation of all elements of matrix
print ("sum  all matrix ")
print (np.sum(y))
# using sum(axis=0) to print summation of all columns of matrix
print ("column wise sum ")
print (np.sum(y,axis=0))
# using sum(axis=1)
print ("row wise sum ")
print (np.sum(y,axis=1))
print (" transpose ")
print (x.T)
print('uppe triangulae')
print(np.triu(x, k = 1)) #also find why k is needed
#if not specified check o/p
print('lower tri')
print(np.tril(x, k = -1))
```

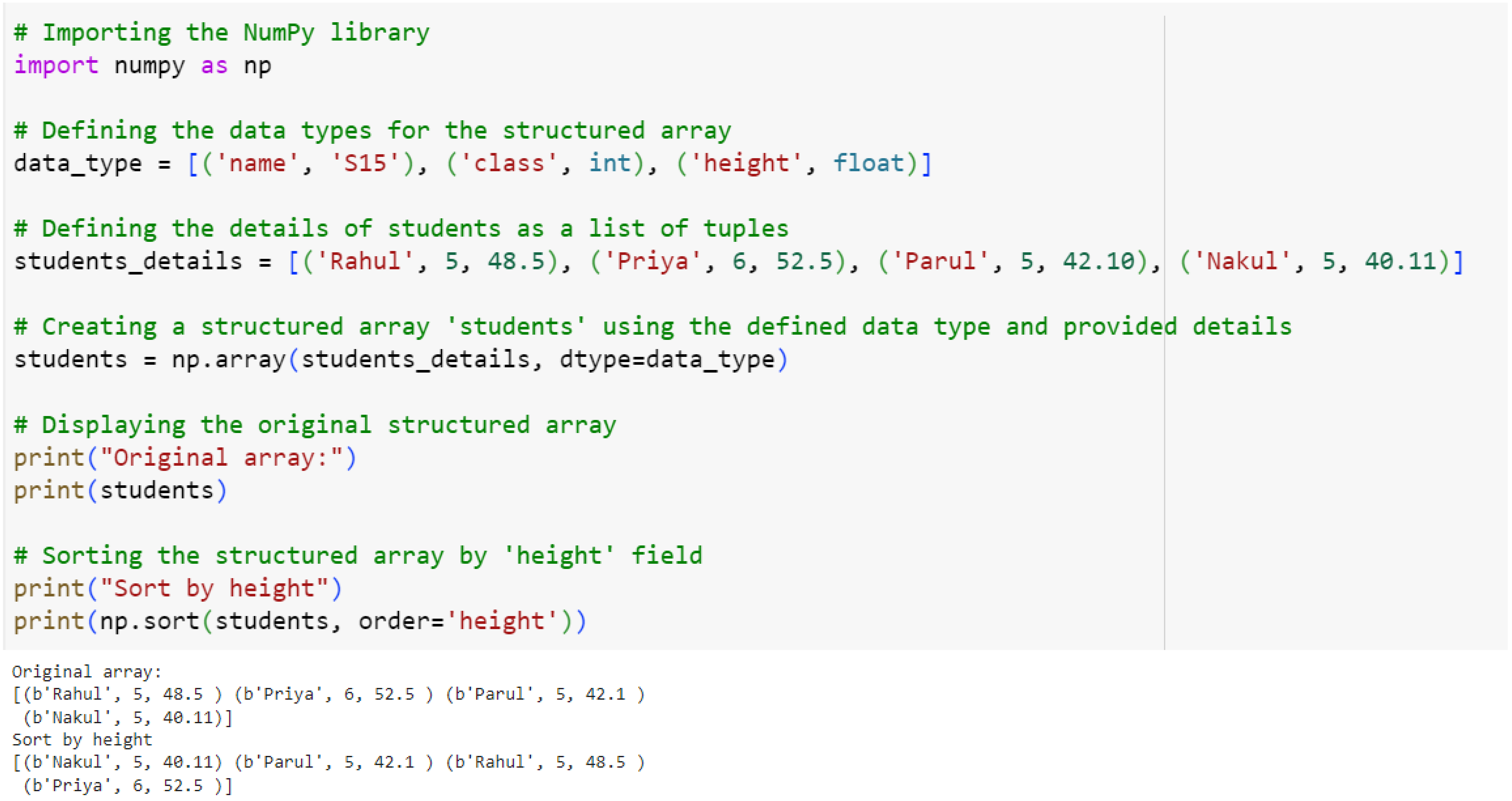


```
The element wise square root is :
[[ 8  4]
 [14 25]]
[[2.82842712 2.        ]
 [3.74165739 5.        ]]
sum  all matrix
141
column wise sum
[ 34 107]
row wise sum
[ 32 109]
 transpose
[[ 8 14]
 [ 4 25]]
uppe triangulae
[[0 4]
 [0 0]]
lower tri
[[ 0  0]
 [14  0]]
```



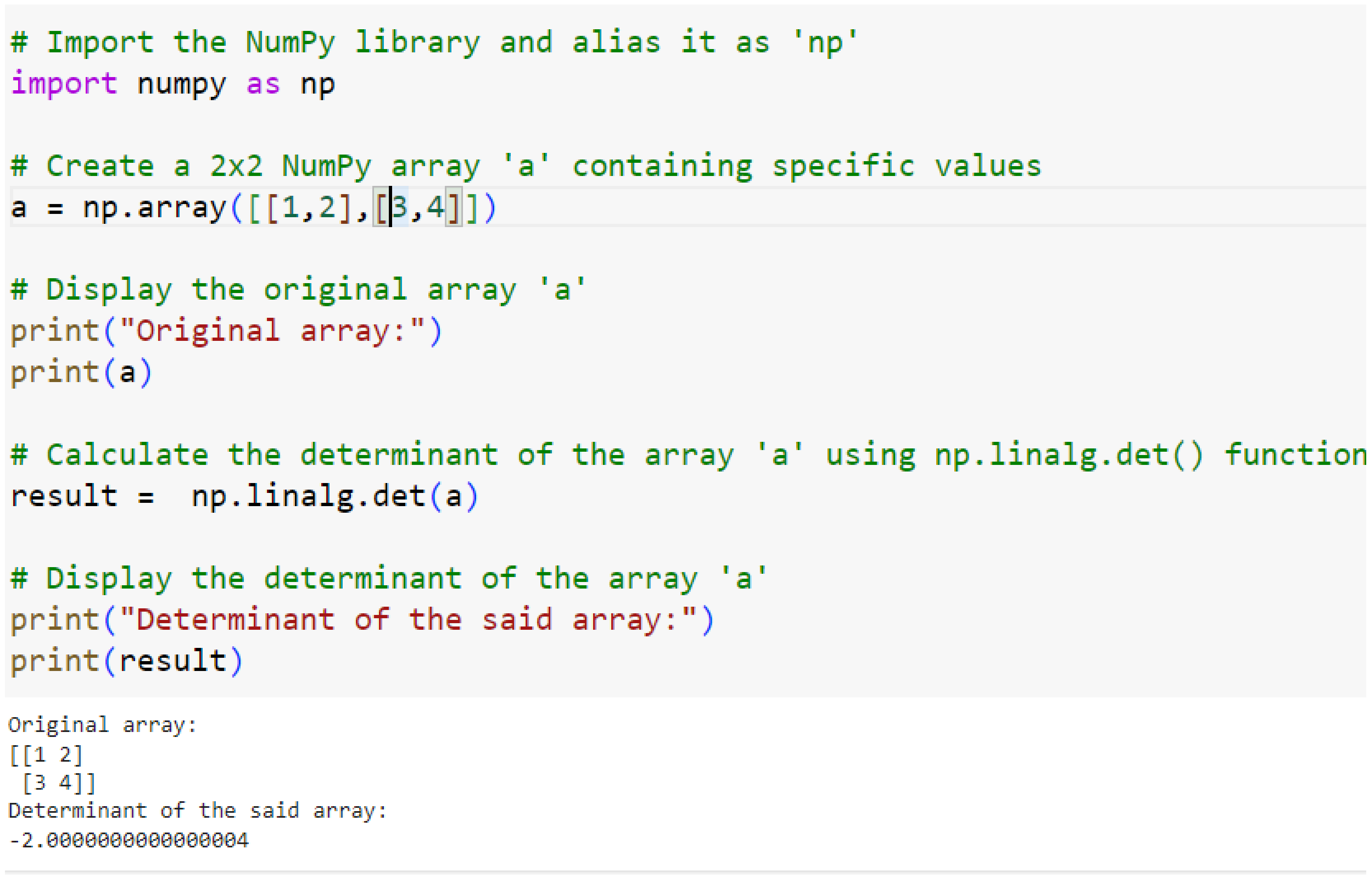
```python
import numpy as np
a = np.array([[3,7],[9,1]])
print ('orioginal array:' )
print (a)
print ('\n')
print ('after sort' )
print (np.sort(a))
print ('\n' )
print ('Sort wrt 0:' )
print (np.sort(a, axis = 0) )
print ('\n'  )
# Order parameter in sort function
dt = np.dtype([('name', 'S10'),('age', int)])
a = np.array([("rahil",21),("ajay",25),("dipika", 7), ("kareena",27)], dtype = dt)
print ('styudent:')
print (a)
print ('\n')
print ('Order by name:' )
print (np.sort(a, order = 'name'))
```

```
orioginal array:
[[3 7]
 [9 1]]


after sort
[[3 7]
 [1 9]]


Sort wrt 0:
[[3 1]
 [9 7]]


styudent:
[(b'rahil', 21) (b'ajay', 25) (b'dipika',  7) (b'kareena', 27)]


Order by name:
[(b'ajay', 25) (b'dipika',  7) (b'kareena', 27) (b'rahil', 21)]
```



```
# Importing the NumPy library
import numpy as np

# Defining the data types for the structured array
data_type = [('name', 'S15'), ('class', int), ('height', float)]

# Defining the details of students as a list of tuples
students_details = [('Rahul', 5, 48.5), ('Priya', 6, 52.5), ('Parul', 5, 42.10), ('Nakul', 5, 40.11)]

# Creating a structured array 'students' using the defined data type and provided details
students = np.array(students_details, dtype=data_type)

# Displaying the original structured array
print("Original array:")
print(students)

# Sorting the structured array by 'height' field
print("Sort by height")
print(np.sort(students, order='height'))
```

```
Original array:
[(b'Rahul', 5, 48.5 ) (b'Priya', 6, 52.5 ) (b'Parul', 5, 42.1 )
 (b'Nakul', 5, 40.11)]
Sort by height
[(b'Nakul', 5, 40.11) (b'Parul', 5, 42.1 ) (b'Rahul', 5, 48.5 )
 (b'Priya', 6, 52.5 )]
```



```python
# Import the NumPy library and alias it as 'np'
import numpy as np

# Create a 2x2 NumPy array 'a' containing specific values
a = np.array([[1,2],[3,4]])

# Display the original array 'a'
print("Original array:")
print(a)

# Calculate the determinant of the array 'a' using np.linalg.det() function
result =  np.linalg.det(a)

# Display the determinant of the array 'a'
print("Determinant of the said array:")
print(result)
```

```
Original array:
[[1 2]
 [3 4]]
Determinant of the said array:
-2.0000000000000004
```