National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Ma'am Neelma Naz **Date:** September 18, 2025

Semester: 7th **Group:** 02

CS471 Machine Learing

Lab 2: Data Structures, NumPy Arrays and SciPy Functions

		PLO4	PLO5	PLO5	PLO8	PLO9
		CLO4	CLO5	CLO5	CLO6	CLO7
Student Name	**Reg. No**	**Viva / Quiz / Demo** 5 Marks	**Analysis of Data in Report** 5 Marks	**Modern Tool Usage** 5 Marks	**Ethics** 5 Marks	**Individual and Teamwork** 5 Marks
Hanzla Sajjad	403214					
Irfa Farooq	412564					

Introduction

This laboratory exercise is focused on the introduction of data structures native in Python particularly Lists and Dictionaries which are very commonly used. This lab also introduces the importing of modules that are used for machine learning tasks. In this lab, the NumPy and SciPy libraries will be introduced which are very important to the field of Machine Learning.

Objectives

The following are the main objectives of this lab:

- Implement data structures such as lists and dictionaries in python
- Create, alter and loop through lists
- Use slicing to access range of items in a list
- Utilize various list methods such as append, insert, extend, remove, pop etc
- Create and implement a dictionary
- Create Numpy arrays and perform matrix operations and broadcasting
- Use Scipy for minimization, scarce matrices and iterpolation

Lab Conduct

- Respect faculty and peers through speech and actions
- The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
- In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
- Use the tab key to provide the indentation in python.
- When you provide the code in the report, keep the font size at 12

Theory

Data structures are an important part of python. The 4 main data structures are lists, tuples, sets and dictionaries. Lists and dictionaries are the most commonly used for machine learning tasks. The *import* keyword is used to load modules and libraries. In machine learning, there are many popular libraries. The most basic of these is the NumPy library which provides an optimized array implementation for very fast matrix computations necessary for machine learning. The SciPy library provides numerous functions for scientific computations.

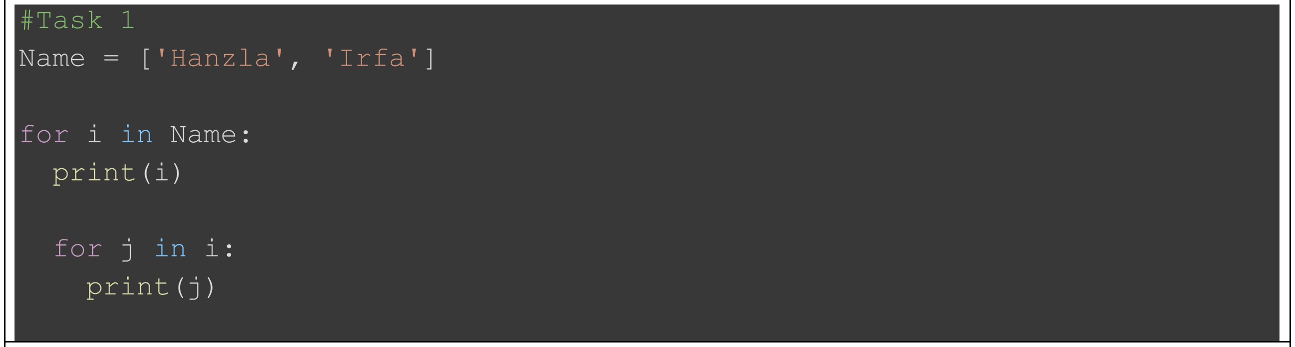
A brief summary of the list functions in python is provided below:

append(I)	append item I to the end of the list
insert(i, I)	insert item I at i position of the list
extend(L)	extend/concatenate a second list L
remove(I)	remove a specified item I from a list
pop(i)	remove item at specific index i in the list
count(I)	return total number of a specific item I from a list
index(I)	return index of first occurrence of a specific item I
reverse	reverse the items of the list

Lab Task 1 – Lists

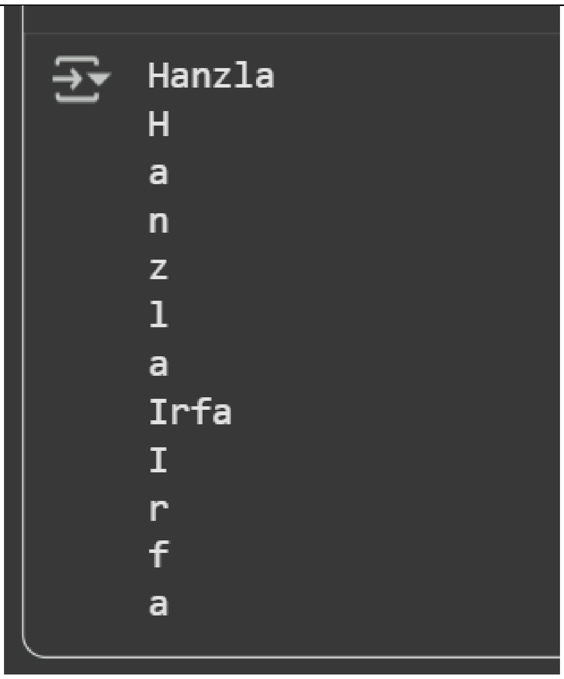
Create 1-D lists containing the characters of the names of all persons in your group. Loop through the lists and display each character on a new line.

Code



```
#Task 1
Name = ['Hanzla', 'Irfa']

for i in Name:
  print(i)

  for j in i:
    print(j)
```

Output Console



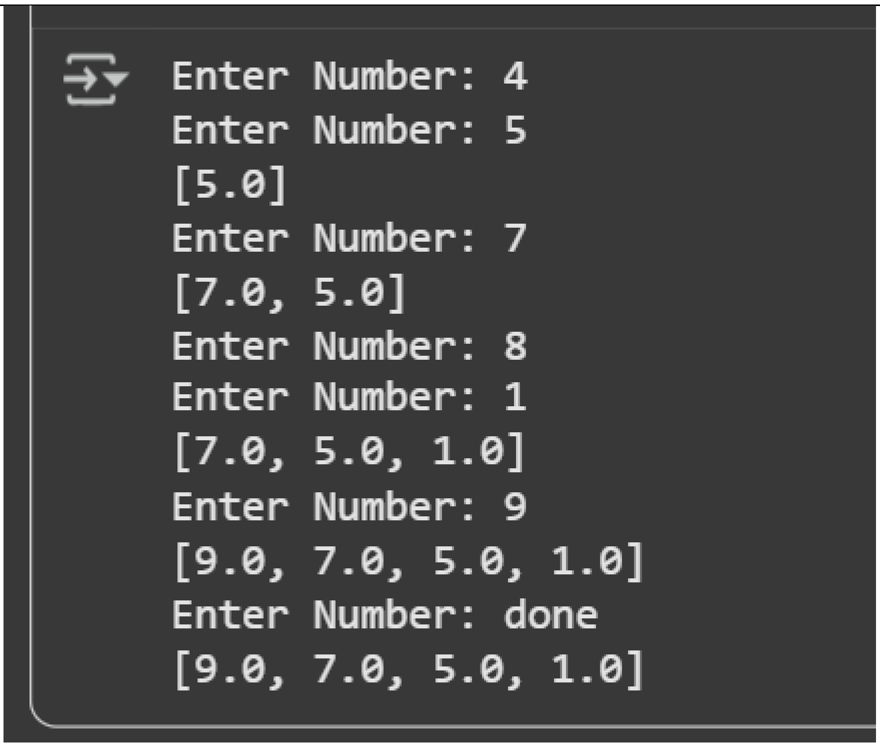
```
Hanzla
H
a
n
z
l
a
Irfa
I
r
f
a
```

Lab Task 2 – Sorting

Write a program that repeatedly prompts the user for input. The user will keep entering numbers (including float numbers) which are added to a list. If a user adds a multiple of 4, then it should be ignored and not added to the list. Each time a number (not a multiple of 4) is added to the list, it must be placed in such a way that the list items are always in descending order. Each time a number is given as input, the list is to be printed showing the newly added number. This continues until the word "done" is input at which point the prompts will stop. The final list is then displayed. Do NOT use any inbuilt sorting function for this task.

Code

```python
#Task 2
Num = []
List = []

while(1):
  Num = input("Enter Number: ")

  if(Num == 'done'):
    break

  if(float(Num) % 4 != 0):
    List.append(float(Num))

    for i in range(len(List)):
      for j in range(i + 1, len(List)):
        if (List[i] < List[j]):
          temp = List[i]
          List[i] = List[j]
          List[j] = temp
    print(List)

print(List)
```

Output Console	
	

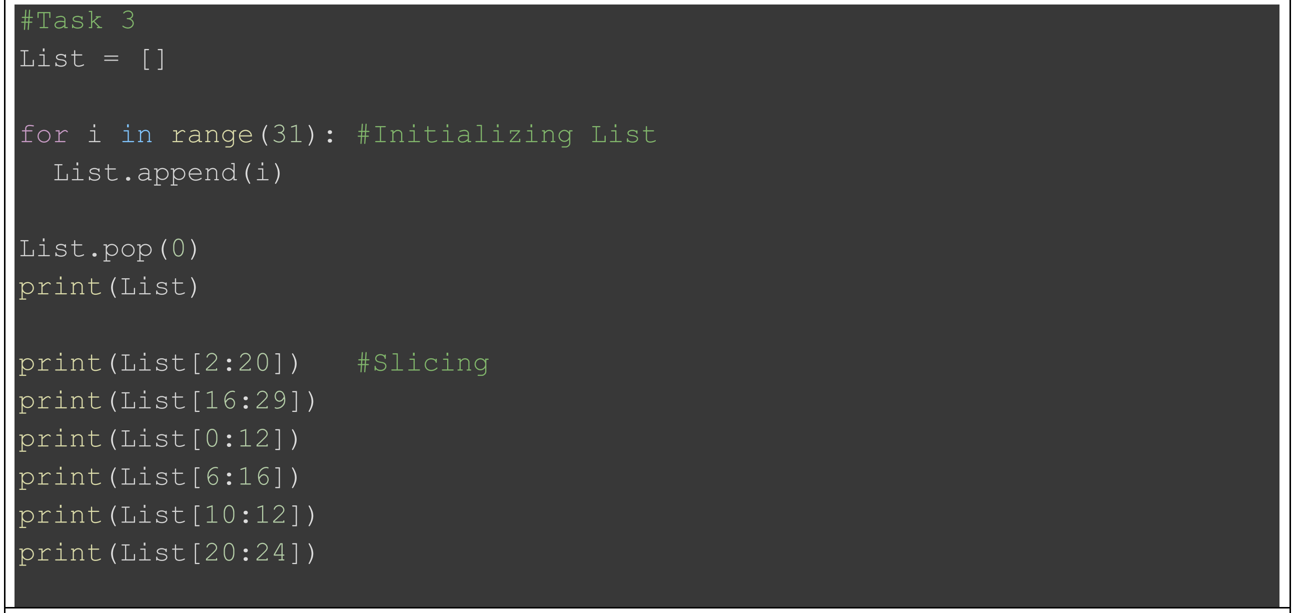
```
Enter Number: 4
Enter Number: 5
[5.0]
Enter Number: 7
[7.0, 5.0]
Enter Number: 8
Enter Number: 1
[7.0, 5.0, 1.0]
Enter Number: 9
[9.0, 7.0, 5.0, 1.0]
Enter Number: done
[9.0, 7.0, 5.0, 1.0]
```

Lab Task 3 – Slicing

Create a list with the sequence 1, 2, 3… 30. Then using the slice operation (:) on this list, print the following sub-lists:

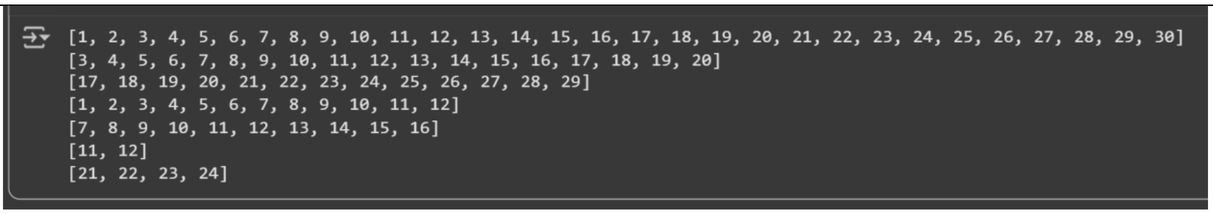
3, 4, 5, 6, 7… 20
17, 18, 19 … 29
1, 2, 3… 12
7, 8, 9 … 16
11, 12
21, 22, 23, 24

Code



```
#Task 3
List = []

for i in range(31): #Initializing List
  List.append(i)

List.pop(0)
print(List)

print(List[2:20])   #Slicing
print(List[16:29])
print(List[0:12])
print(List[6:16])
print(List[10:12])
print(List[20:24])
```

Output Console

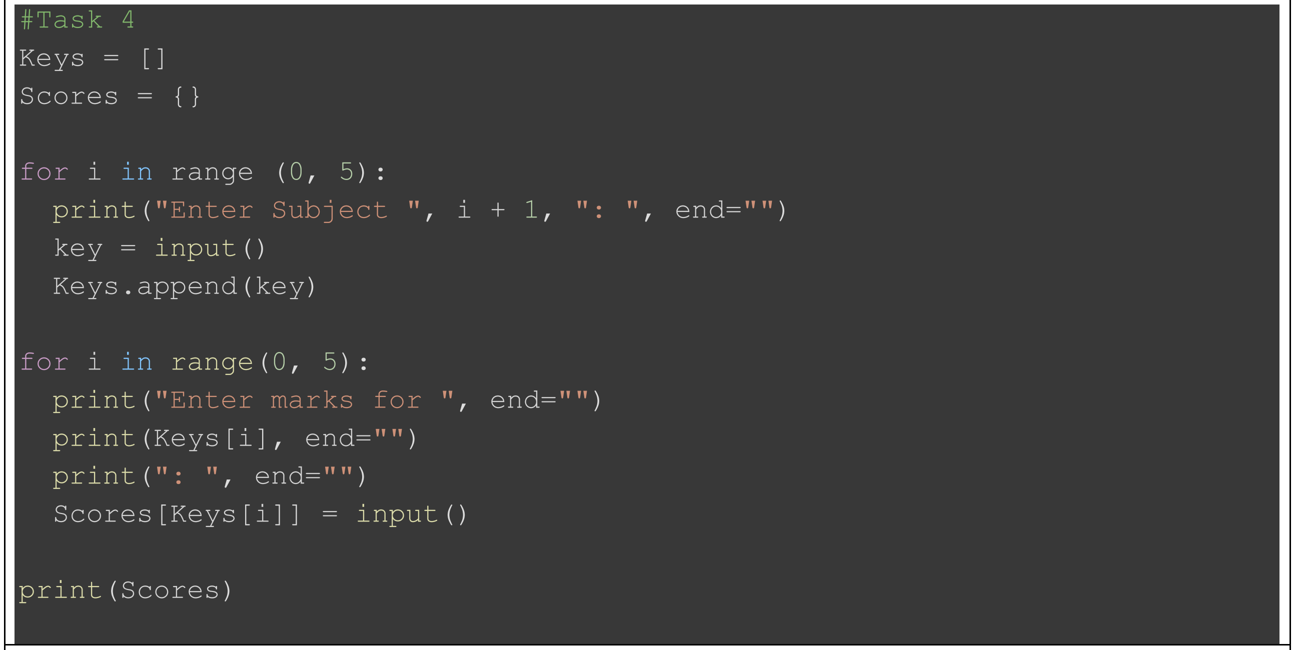


```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
[17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
[7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
[11, 12]
[21, 22, 23, 24]
```

Lab Task 4 – Dictionaries

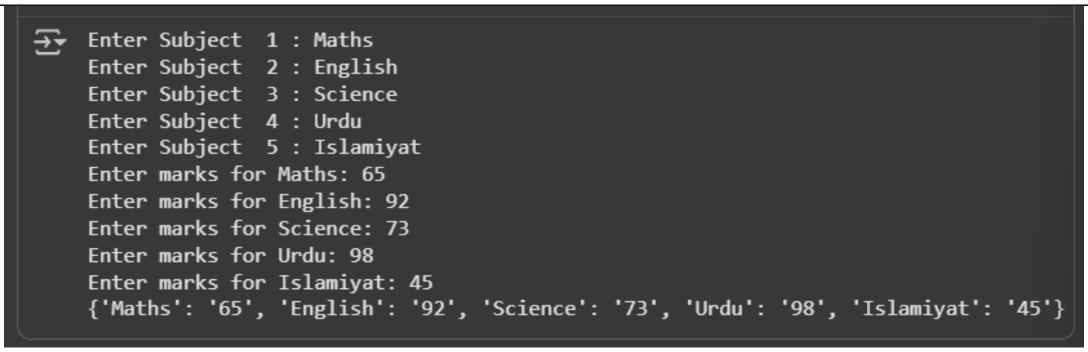
In this task, you will make use of dictionaries. Write a program that first prompts the user to input fifteen different strings which will be the keys of the dictionary. Then, the program must prompt the user to input the values (integers or float numbers) of the respective keys. When entering the values, the user must be shown the key whose value is being input. Once all values are entered, display the entire dictionary.

Code



```python
#Task 4
Keys = []
Scores = {}

for i in range (0, 5):
  print("Enter Subject ", i + 1, ": ", end="")
  key = input()
  Keys.append(key)

for i in range(0, 5):
  print("Enter marks for ", end="")
  print(Keys[i], end="")
  print(": ", end="")
  Scores[Keys[i]] = input()

print(Scores)
```

Output Console



```
Enter Subject  1 : Maths
Enter Subject  2 : English
Enter Subject  3 : Science
Enter Subject  4 : Urdu
Enter Subject  5 : Islamiyat
Enter marks for Maths: 65
Enter marks for English: 92
Enter marks for Science: 73
Enter marks for Urdu: 98
Enter marks for Islamiyat: 45
{'Maths': '65', 'English': '92', 'Science': '73', 'Urdu': '98', 'Islamiyat': '45'}
```

Lab Task 5 – NumPy Arrays

Import the NumPy and Random Library. Use the np.array function to define a 5x5 array with random integers. Ensure that the elements are numbers (not strings). Then, perform the following:

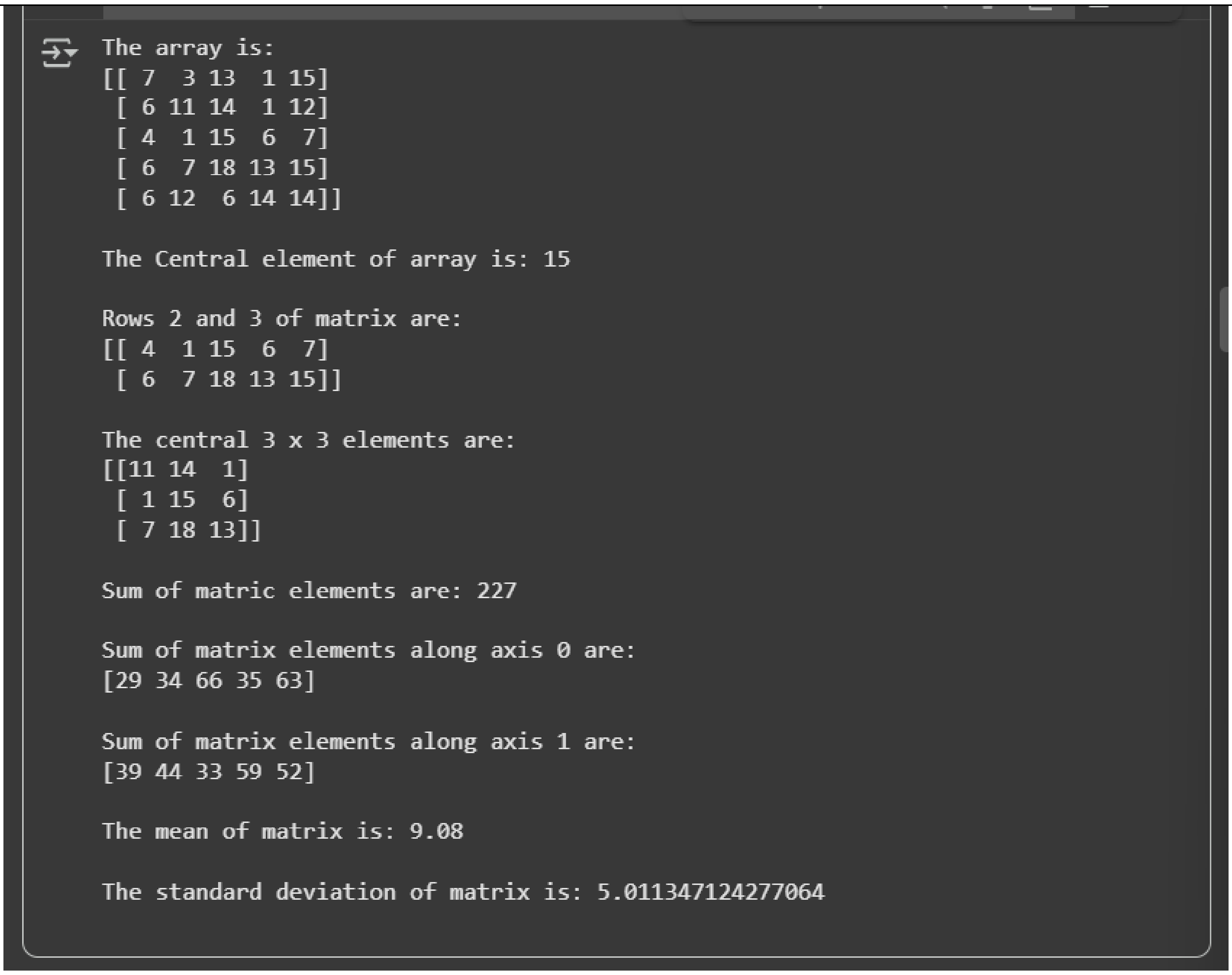
- Print the complete array
- Print the central element(3, 3)
- Print rows 2 and 3 via slicing
- Print the central 3x3 elements in a matrix
- Compute the sum of the matrix elements
- Compute the sum of the matrix elements along axis 0
- Compute the sum of the matrix elements along axis 1
- Compute the mean of the matrix elements
- Compute the standard deviation of the matrix elements

Code

```
#Task 5
import numpy as np

my_array = np.array(np.random.randint(1, 20, (5, 5)))
print("The array is: ")
print(my_array, "\n")                    #Print Complete array
print("The Central element of array is: ", end="")
print(my_array[2][2], "\n")              #Print central element (3, 3)
print("Rows 2 and 3 of matrix are: ")
print(my_array[2:4], "\n")               #Print rows 2 and 3 via slicing
print("The central 3 x 3 elements are: ")
print(my_array[1:4, 1:4], "\n")          #Print central 3 x 3 elements in a
matrix
print("Sum of matric elements are: ", end="")
print(np.sum(my_array), "\n")            #Compute sum of matrix elements
print("Sum of matrix elements along axis 0 are: ")
print(np.sum(my_array, axis = 0), "\n") #Compute sum of matrix elements
along axis 0
print("Sum of matrix elements along axis 1 are: ")
print(np.sum(my_array, axis = 1), "\n") #Compute sum of matric elements
along axis 1
```

```
print("The mean of matrix is: ", end="")
print(np.mean(my_array), "\n")            #Compute mean of matrix
print("The standard deviation of matrix is: ", end="")
print(np.std(my_array), "\n")             #Compute standard deviation of
matrix
```

Output Console



```
The array is:
[[ 7  3 13  1 15]
 [ 6 11 14  1 12]
 [ 4  1 15  6  7]
 [ 6  7 18 13 15]
 [ 6 12  6 14 14]]

The Central element of array is: 15

Rows 2 and 3 of matrix are:
[[ 4  1 15  6  7]
 [ 6  7 18 13 15]]

The central 3 x 3 elements are:
[[11 14  1]
 [ 1 15  6]
 [ 7 18 13]]

Sum of matric elements are: 227

Sum of matrix elements along axis 0 are:
[29 34 66 35 63]

Sum of matrix elements along axis 1 are:
[39 44 33 59 52]

The mean of matrix is: 9.08

The standard deviation of matrix is: 5.011347124277064
```

Lab Task 6 – Matrix Computations

Use the np.array function to define two different matrices of size 3x3. Place numerical elements of your choice in the matrices. Write code to perform the following:

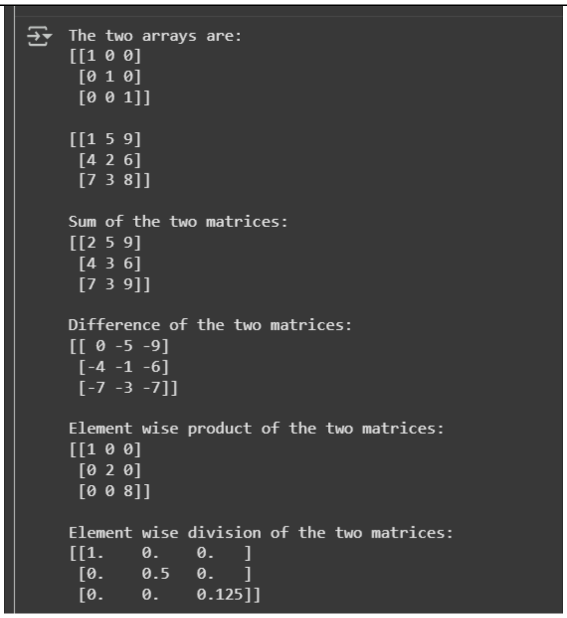
- Print the arrays
- Compute the sum of the matrices
- Compute the difference of the matrices
- Compute the element-wise product of the matrices
- Compute the element-wise division of the matrices
- Compute the transpose of both matrices
- Compute the matrix multiplication of the matrices

Code

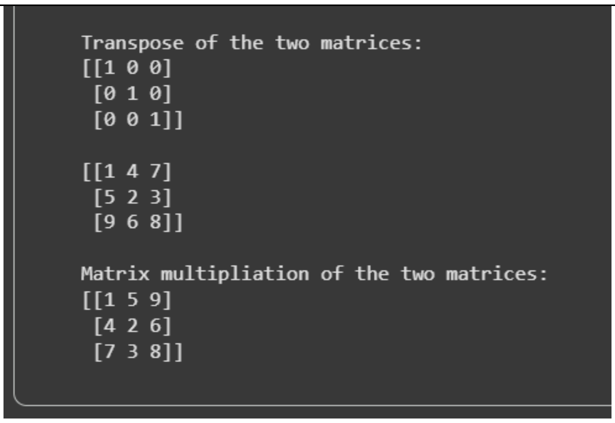
```
#Task 6
import numpy as np
array1 = np.array([[1, 0, 0], #Identity matrix
                   [0, 1, 0],
                   [0, 0, 1]])

array2 = np.array([[1, 5, 9], #Matrix 2
                   [4, 2, 6],
                   [7, 3, 8]])

print("The two arrays are: ")
print(array1, "\n")                    #Print the two arrays
print(array2, "\n")

array_sum = array1 + array2
print("Sum of the two matrices: ")
print(array_sum, "\n")                 #Sum of matrices

array_diff = array1 - array2
print("Difference of the two matrices: ")
print(array_diff, "\n")                #Difference of matrices
```

```
array_prod = array1 * array2
print("Element wise product of the two matrices: ")
print(array_prod, "\n")              #Product of arrays

array_div = array1 / array2
print("Element wise division of the two matrices: ")
print(array_div, "\n")               #division of matrices

array1_transpose = np.transpose(array1)
array2_transpose = np.transpose(array2)
print("Transpose of the two matrices: ")
print(array1_transpose, "\n")        #Transpose of matrices
print(array2_transpose, "\n")

array_multiplication = np.dot(array1, array2)
print("Matrix multipliation of the two matrices: ")
print(array_multiplication, "\n")   #Matrix Multiplication
```

Output Console



```
The two arrays are:
[[1 0 0]
 [0 1 0]
 [0 0 1]]

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

Sum of the two matrices:
[[2 5 9]
 [4 3 6]
 [7 3 9]]

Difference of the two matrices:
[[ 0 -5 -9]
 [-4 -1 -6]
 [-7 -3 -7]]

Element wise product of the two matrices:
[[1 0 0]
 [0 2 0]
 [0 0 8]]

Element wise division of the two matrices:
[[1.    0.    0.   ]
 [0.    0.5   0.   ]
 [0.    0.    0.125]]
```



```
Transpose of the two matrices:
[[1 0 0]
 [0 1 0]
 [0 0 1]]

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

Matrix multipliation of the two matrices:
[[1 5 9]
 [4 2 6]
 [7 3 8]]
```

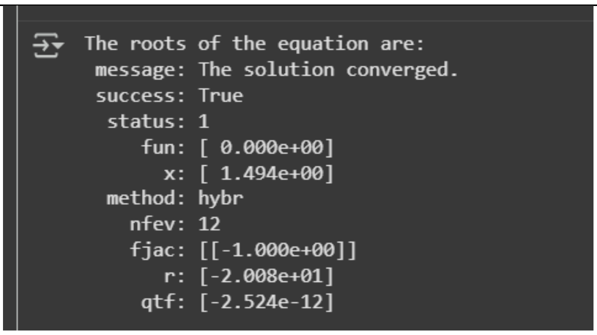
Lab Task 7 – SciPy Functions

In this task, you will use various functions of the SciPy library that are commonly used in machine learning. Import the various modules from SciPy Library:

```
from scipy.optimize import root
from scipy.optimize import minimize
from scipy.sparse import csr_matrix
from scipy.interpolate import interp1d
```

a) Use the root function to determine the roots of the equation $3x^2 + 2x - 10$.

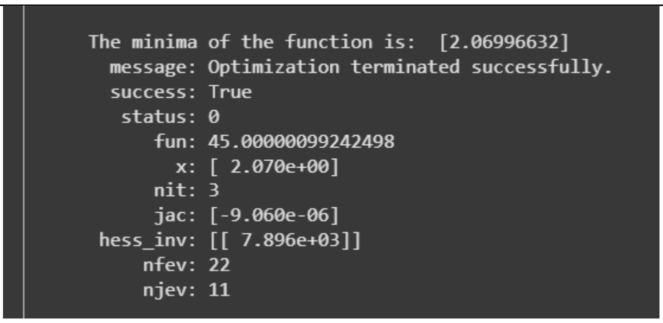
Output Console



```
The roots of the equation are:
 message: The solution converged.
 success: True
  status: 1
     fun: [ 0.000e+00]
       x: [ 1.494e+00]
  method: hybr
    nfev: 12
    fjac: [[-1.000e+00]]
       r: [-2.008e+01]
     qtf: [-2.524e-12]
```

b) In machine learning, it is very often required to find the argument that minimizes a complex equation with the given data. Use the minimize function to determine the minima of the equation $x^2 - 20x + 45$.

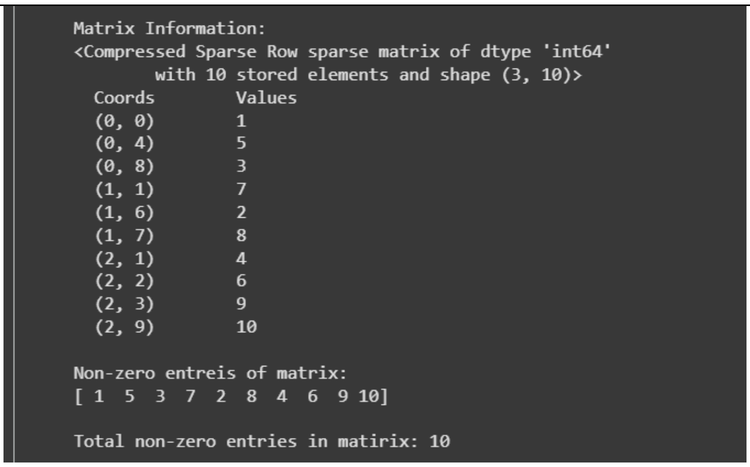
Output Console



```
The minima of the function is:  [2.06996632]
  message: Optimization terminated successfully.
  success: True
   status: 0
      fun: 45.00000099242498
        x: [ 2.070e+00]
      nit: 3
      jac: [-9.060e-06]
 hess_inv: [[ 7.896e+03]]
     nfev: 22
     njev: 11
```

c) In machine learning, sometimes there are matrices in which most of the elements are zero. In such cases, it is more convenient to store them as sparse matrices which hold information of the non-zero elements. In this task, create a 3x10 sparse matrix (A) with elements of your choice. Ensure that about 2/3 of the elements are zero. Then, print the matrix information using csr_matrix(A), csr_matrix(A).data and csr_matrix(A).count_nonzero().

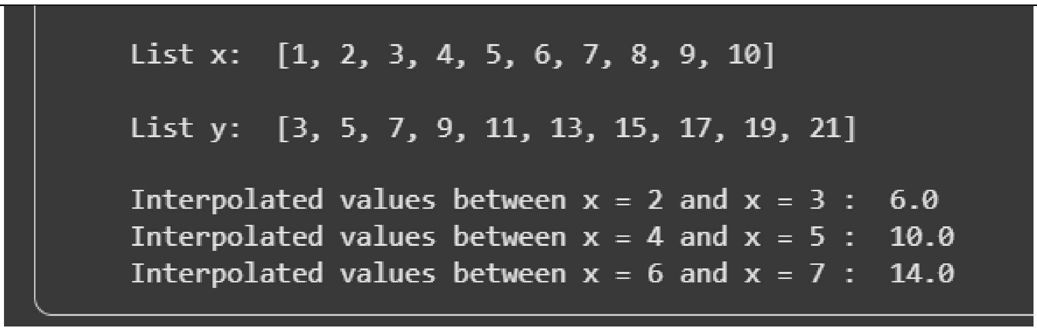
Output Console



```
Matrix Information:
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 10 stored elements and shape (3, 10)>
  Coords        Values
  (0, 0)        1
  (0, 4)        5
  (0, 8)        3
  (1, 1)        7
  (1, 6)        2
  (1, 7)        8
  (2, 1)        4
  (2, 2)        6
  (2, 3)        9
  (2, 9)        10

Non-zero entreis of matrix:
[ 1  5  3  7  2  8  4  6  9 10]

Total non-zero entries in matirix: 10
```

d) In this task, you will perform interpolation. Create two lists x and y. The list x contains elements 1,2,3… 10. The list y contains the elements for y = 2x + 1. Use the interp_func = interp1d(x, y) to get the interpolating function. Then, use interp_func(val) to get any 3 interpolated.

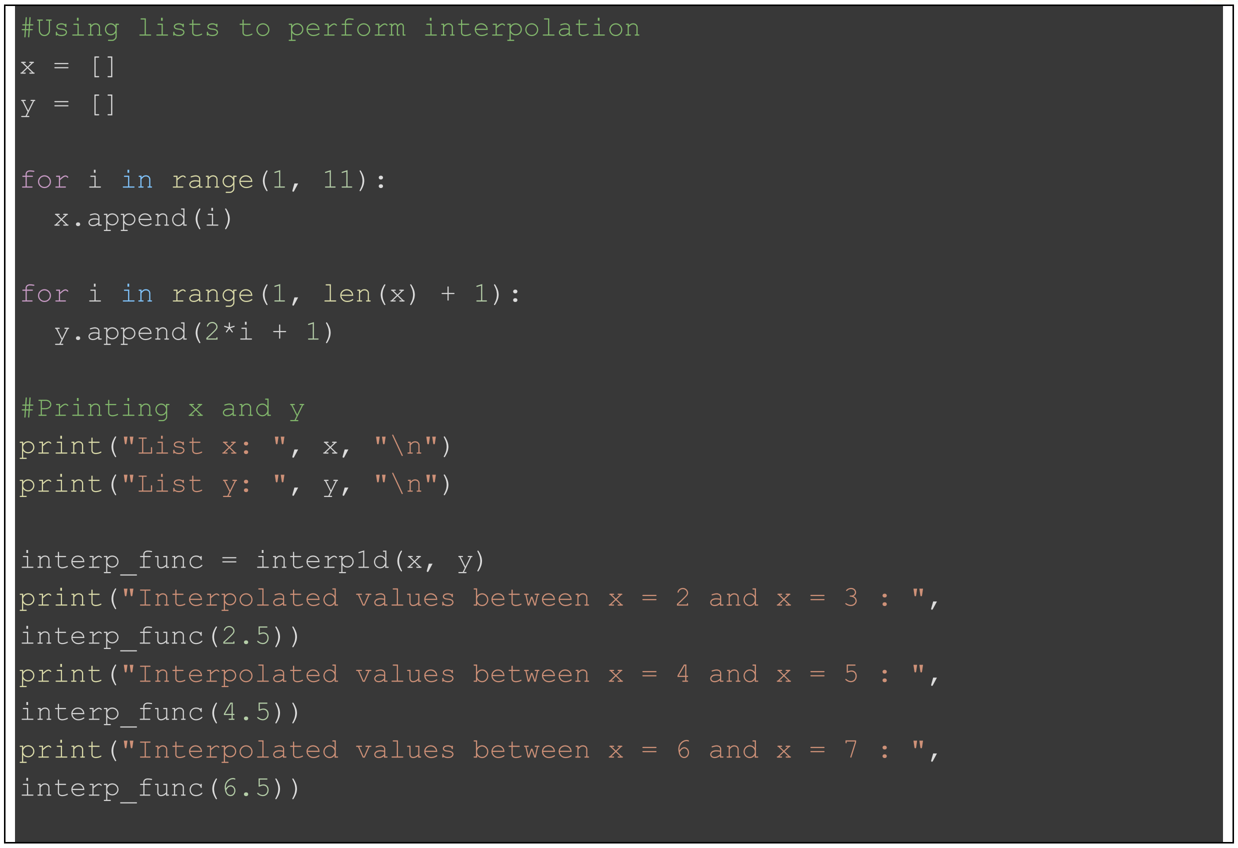
Output Console



```
List x:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

List y:  [3, 5, 7, 9, 11, 13, 15, 17, 19, 21]

Interpolated values between x = 2 and x = 3 :  6.0
Interpolated values between x = 4 and x = 5 :  10.0
Interpolated values between x = 6 and x = 7 :  14.0
```

Code

```python
#Task 7
import numpy as np
from scipy.optimize import root
from scipy.optimize import minimize
from scipy.sparse import csr_matrix
from scipy.interpolate import interp1d

#Using root function to determine roots of equation
def f_scalar(x):
  return 3*x** + 2*x - 10

result1 = root(f_scalar, 1)
print("The roots of the equation are: ")
print(result1, "\n")

#Using minimize function to determine minima
def minima(x):
  return x** - 20*x + 45

result2 = minimize(minima, 1)
print("The minima of the function is: ", result2.x)
print(result2, "\n")

#Using CSR and performing operations
A = np.array([[1, 0, 0, 0, 5, 0, 0, 0, 3, 0],
              [0, 7, 0, 0, 0, 0, 2, 8, 0, 0],
              [0, 4, 6, 9, 0, 0, 0, 0, 0, 10]])

print("Matrix Information: ")
print(csr_matrix(A), "\n")
print("Non-zero entreis of matrix: ")
print(csr_matrix(A).data, "\n")
print("Total non-zero entries in matirix: ", end="")
print(csr_matrix(A).count_nonzero(), "\n")
```



```python
#Using lists to perform interpolation
x = []
y = []

for i in range(1, 11):
  x.append(i)

for i in range(1, len(x) + 1):
  y.append(2*i + 1)

#Printing x and y
print("List x: ", x, "\n")
print("List y: ", y, "\n")

interp_func = interp1d(x, y)
print("Interpolated values between x = 2 and x = 3 : ",
interp_func(2.5))
print("Interpolated values between x = 4 and x = 5 : ",
interp_func(4.5))
print("Interpolated values between x = 6 and x = 7 : ",
interp_func(6.5))
```

Lab Task 8 – Broadcasting

In this task, you will explore the concept of broadcasting. Use the np.array function to define a 4x4 matrix and a 4x1 vector. Place numerical elements of your choice in the matrices. Write code to perform the following:

- Print the arrays
- Compute the sum of the matrix with the vector
- Compute the difference of the matrix with the vector
- Compute the transpose of both
- Compute the matrix multiplication of the matrix with the vector

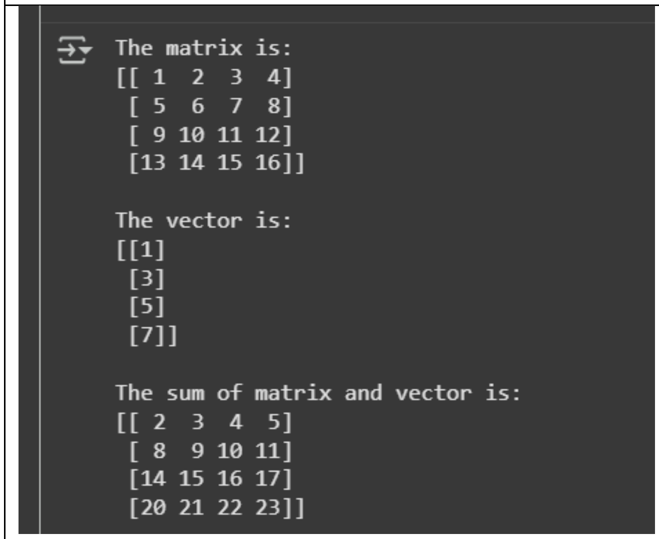
Explain what is happening in the sum and difference operations. Provide the code and all relevant screenshots.

Code

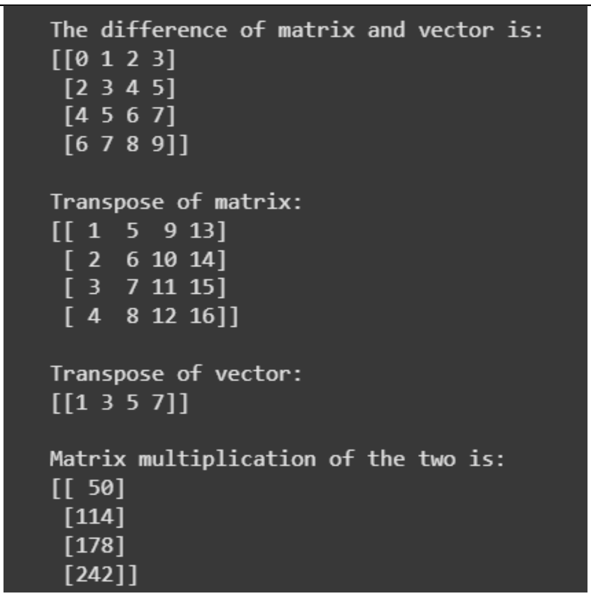
```
#Task 8
import numpy as np

my_matrix = np.array([[1, 2, 3, 4],                    #Matrix Initialization
                      [5, 6, 7, 8],
                      [9, 10, 11, 12],
                      [13, 14, 15, 16]])

my_vector = np.array([[1],                             #Vector Initialization
                      [3],
                      [5],
                      [7]])

print("The matrix is: ")
print(my_matrix, "\n")                                 #Printing matrix
print("The vector is: ")
print(my_vector, "\n")                                 #Printing vector

sum = my_matrix + my_vector
print("The sum of matrix and vector is: ")
print(sum, "\n")                                       #Sum of the two
```

```
diff = my_matrix - my_vector
print("The difference of matrix and vector is: ")
print(diff, "\n")                                    #Difference of the two

matrix_transpose = np.transpose(my_matrix)       #Transpose of the two
print("Transpose of matrix: ")
print(matrix_transpose, "\n")

vector_transpose = np.transpose(my_vector)
print("Transpose of vector: ")
print(vector_transpose, "\n")

multiplication = np.dot(my_matrix, my_vector)
print("Matrix multiplication of the two is: ")  #Matrix Multiplication of
the two
print(multiplication)
```

Output Console



```
The matrix is:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

The vector is:
[[1]
 [3]
 [5]
 [7]]

The sum of matrix and vector is:
[[ 2  3  4  5]
 [ 8  9 10 11]
 [14 15 16 17]
 [20 21 22 23]]
```



```
The difference of matrix and vector is:
[[0 1 2 3]
 [2 3 4 5]
 [4 5 6 7]
 [6 7 8 9]]

Transpose of matrix:
[[ 1  5  9 13]
 [ 2  6 10 14]
 [ 3  7 11 15]
 [ 4  8 12 16]]

Transpose of vector:
[[1 3 5 7]]

Matrix multiplication of the two is:
[[ 50]
 [114]
 [178]
 [242]]
```

In the sum and difference operations, the vector is added and subtracted with each column individually and the answers are shared in the form of the matrix.