Artificial Intelligence and Machine Learning

LABORATORY MANUAL

[R20A0588]

B.TECH III YEAR – I SEM

[A.Y:2022-2023]

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Prepared by

Ms P Anitha, Associate Professor

Ms K Bhavana, Assistant Professor

Ms R Sathish Kumar, Assistant Professor

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC
ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Electronics and Communication Engineering

Vision

- To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong which involves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students in to competent and confident engineers.

- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce internationally accepted competitive and world class professionals.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problemsincreasing their productivity.

PEO2 – TECHNICAL SKILLS

- To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1. Fundamentals and critical knowledge of the Computer System:- Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardwareaspects of it .

2. The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data,and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

1. **Lab Objectives:**

- Learning basic concepts of Python through illustrative examples and small exercises
- To prepare students to become Familiarity with the Python programming in AI environment.
- To provide student with an academic environment aware of various AI Algorithms.
- To train Students with python programming as to comprehend, analyze, design and create AI platforms and solutions for the real life problems.

2. Lab Outcomes:

Upon completion of the course, students will be able to

- Apply various AI search algorithms (uninformed, informed, heuristic, constraint satisfaction,)
- Understand the fundamentals of knowledge representation, inference.
- Understand the fundamentals of theorem proving using AI tools.
- Demonstrate working knowledge of reasoning in the presence of incomplete and/or uncertain information

3. Introduction about lab

Minimum System requirements:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor.
- Disk space: 1 GB.
- Operating systems: Windows* 7 or later, macOS, and Linux.
- **Python*** versions: 2.7.X, 3.6.X.,3.8.X

About lab:

Python is a general purpose, high-level programming language; other high level languages you might have heard of C++, PHP, Java and Python. Virtually all modern programming languages make us of an Integrated Development Environment (IDE), which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle).

Many modern languages use both processes. They are first compiled into a lower level language, called byte code, and then interpreted by a program called avirtual machine. Python uses both processes, but because of the way programmers interact with it, it is usually considered an interpreted language.

Practical aspects are the key to understanding and conceptual visualization of Theoretical aspects covered in the books. Also, this course is designed to review the concepts of Data Structure , studied in previous semester and implement the various algorithms related to different data structures.

4. Guidelines to students

A. Standard operating procedure

a) Explanation on today's experiment by the concerned faculty using PPT covering the following aspects:

1) Name of the experiment

2) Aim

b) Writing the python programs by the students

c) Commands for executing programs

Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

a) Name of the experiment

b) Aim

c) Writing the program

d) Viva-Voce Questions and Answers

e) Errors observed (if any) during compilation/execution

f) Signature of the Faculty

B. Guide Lines to Students in Lab

Disciplinary to be maintained by the students in the Lab

- ➢ Students are required to carry their lab observation book and record book with completed experiments while entering the lab.
- ➢ Students must use the equipment with care. Any damage is caused student is punishable
- ➢ Students are not allowed to use their cell phones/pen drives/ CDs in labs.
- ➢ Students need to be maintain proper dress code along with ID Card
- ➢ Students are supposed to occupy the computers allotted to them and are not supposed to talk or make noise in the lab.□Students, after completion of each experiment they need to be updated in observation notes and same to be updated in the record.
- ➢ Lab records need to be submitted after completion of experiment and get it corrected with the concerned lab faculty.
- ➢ If a student is absent for any lab, they need to be completed the same experiment in the free time before attending next lab.

Instructions to maintain the record

- Before start of the first lab they have to buy the record and bring the record to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks (5M) will be deducted.

C. General laboratory instructions

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab. c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

LIST OF PROGRAMS

PROGRAM-1

Aim:

Write a Program to Implement Breadth First Search.

Program:

```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '5')    # function calling
```

Output:

```
Following is the Breadth-First Search
5 3 7 2 4 8
```

Viva Questions:

1. **Differences between Informed and Uninformed Search.**
2. **What are the Properties Of Search Algorithms.**
3. **What is Breadth-First-Search.**

PROGRAM-2

Aim:

Write a Program to Implement Depth First Search.

Program:

```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}

visited = set() # Set to keep track of visited nodes of graph.

def dfs(visited, graph, node):  #function for dfs
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

# Driver Code
print("Following is the Depth-First Search")
```

```
dfs(visited, graph, '5')
```

Output:

```
Following is the Depth-First Search
5
3
2
4
8
7
```

Viva Questions:

1. **What is Depth-First- Search.**
2. **Differences between BFS and DFS.**
3. **Write the Applications of Stack and Queue.**

PROGRAM-3

Aim:

Write a program to implement Hill Climbing Algorithm

Program:

```
import random
def randomSolution(tsp):
    cities = list(range(len(tsp)))
    solution = []
    for i in range(len(tsp)):
        randomCity = cities[random.randint(0, len(cities) - 1)]
        solution.append(randomCity)
        cities.remove(randomCity)
    return solution
def routeLength(tsp, solution):
    routeLength = 0
    for i in range(len(solution)):
        routeLength += tsp[solution[i - 1]][solution[i]]
    return routeLength
def getNeighbours(solution):
    neighbours = []
    for i in range(len(solution)):
        for j in range(i + 1, len(solution)):
            neighbour = solution.copy()
            neighbour[i] = solution[j]
            neighbour[j] = solution[i]
            neighbours.append(neighbour)
    return neighbours
def getBestNeighbour(tsp, neighbours):
    bestRouteLength = routeLength(tsp, neighbours[0])
    bestNeighbour = neighbours[0]
    for neighbour in neighbours:
        currentRouteLength = routeLength(tsp, neighbour)
        if currentRouteLength < bestRouteLength:
            bestRouteLength = currentRouteLength
            bestNeighbour = neighbour
    return bestNeighbour, bestRouteLength

def hillClimbing(tsp):
    currentSolution = randomSolution(tsp)
    currentRouteLength = routeLength(tsp, currentSolution)
    neighbours = getNeighbours(currentSolution)
    bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)
    while bestNeighbourRouteLength < currentRouteLength:
        currentSolution = bestNeighbour
        currentRouteLength = bestNeighbourRouteLength
        neighbours = getNeighbours(currentSolution)
        bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)
    return currentSolution, currentRouteLength
```

```
def main():
    tsp = [
        [0, 400, 500, 300],
        [400, 0, 300, 500],
        [500, 300, 0, 400],
        [300, 500, 400, 0]
    ]

    print(hillClimbing(tsp))


if __name__ == "_main_":
    main()
```

Output:

```
([1, 0, 3, 2], 1400)
```

Viva Questions:

1. **What is Heuristic Search.**
2. **List the problems of Hill Climbing.**
3. **Define IDDFS .**

PROGRAM-4

Aim:
Write a program to implement A* Algorithm

Program:
```
from collections import deque

class Graph:
    def__init_(self, adjac_lis):
        self.adjac_lis = adjac_lis

    def get_neighbors(self, v):
        return self.adjac_lis[v]

    # This is heuristic function which is having equal values for all nodes
    def h(self, n):
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }

        return H[n]

    def a_star_algorithm(self, start, stop):
        # In this open_lst is a lisy of nodes which have been visited, but who's
        # neighbours haven't all been always inspected, It starts off with the start
  #node
        # And closed_lst is a list of nodes which have been visited
        # and who's neighbors have been always inspected
        open_lst = set([start])
        closed_lst = set([])

        # poo has present distances from start to all other nodes
        # the default value is +infinity
        poo = {}
        poo[start] = 0

        # par contains an adjac mapping of all nodes
        par = {}
        par[start] = start

        while len(open_lst) > 0:
            n = None
```

```
# it will find a node with the lowest value of f() -
for v in open_lst:
    if n == None or poo[v] + self.h(v) < poo[n] + self.h(n):
        n = v;

if n == None:
    print('Path does not exist!')
    return None

# if the current node is the stop
# then we start again from start
if n == stop:
    reconst_path = []

    while par[n] != n:
        reconst_path.append(n)
        n = par[n]

    reconst_path.append(start)

    reconst_path.reverse()

    print('Path found: {}'.format(reconst_path))
    return reconst_path

# for all the neighbors of the current node do
for (m, weight) in self.get_neighbors(n):
  # if the current node is not presentin both open_lst and closed_lst
    # add it to open_lst and note n as it's par
    if m not in open_lst and m not in closed_lst:
        open_lst.add(m)
        par[m] = n
        poo[m] = poo[n] + weight

    # otherwise, check if it's quicker to first visit n, then m
    # and if it is, update par data and poo data
    # and if the node was in the closed_lst, move it to open_lst
    else:
        if poo[m] > poo[n] + weight:
            poo[m] = poo[n] + weight
            par[m] = n

            if m in closed_lst:
                closed_lst.remove(m)
                open_lst.add(m)
            # remove n from the open_lst, and add it to closed_lst
            # because all of his neighbors were inspected
            open_lst.remove(n)
            closed_lst.add(n)

        print('Path does not exist!')
        return None
adjac_lis = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjac_lis)
graph1.a_star_algorithm('A', 'D')
```

Output:

```
Path found: ['A', 'B', 'D']
```

Viva Questions:

1. **What is Best First Search?**
2. **Explain Heuristic Function?**
3. **Explain A* Search.**

.

PROGRAM-5

Aim:
Write a program to implement Tic-Tac-Toe game.

Program:

```
import os
import time

board = [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ']
player = 1

########win Flags##########
Win = 1
Draw = -1
Running = 0
Stop = 1
###########################
Game = Running
Mark = 'X'

#This Function Draws Game Board
def DrawBoard():
    print(" %c | %c | %c " % (board[1],board[2],board[3]))
    print("___|___|___")
    print(" %c | %c | %c " % (board[4],board[5],board[6]))
    print("___|___|___")
    print(" %c | %c | %c " % (board[7],board[8],board[9]))
    print("   |   |   ")

#This Function Checks position is empty or not
def CheckPosition(x):
    if(board[x] == ' '):
        return True
    else:
        return False

#This Function Checks player has won or not
def CheckWin():
    global Game
    #Horizontal winning condition
    if(board[1] == board[2] and board[2] == board[3] and board[1] != ' '):
        Game = Win
    elif(board[4] == board[5] and board[5] == board[6] and board[4] != ' '):
        Game = Win
    elif(board[7] == board[8] and board[8] == board[9] and board[7] != ' '):
        Game = Win
    #Vertical Winning Condition
    elif(board[1] == board[4] and board[4] == board[7] and board[1] != ' '):
        Game = Win
    elif(board[2] == board[5] and board[5] == board[8] and board[2] != ' '):
        Game = Win
    elif(board[3] == board[6] and board[6] == board[9] and board[3] != ' '):
        Game=Win
    #Diagonal Winning Condition
    elif(board[1] == board[5] and board[5] == board[9] and board[5] != ' '):
        Game = Win
    elif(board[3] == board[5] and board[5] == board[7] and board[5] != ' '):
        Game=Win
    #Match Tie or Draw Condition
    elif(board[1]!=' ' and board[2]!=' ' and board[3]!=' ' and board[4]!=' ' and board[5]!=' ' and
board[6]!=' ' and board[7]!=' ' and board[8]!=' ' and board[9]!=' '):
        Game=Draw
    else:
        Game=Running
print("Tic-Tac-Toe Game Designed By Sourabh Somani")
print("Player 1 [X] --- Player 2 [O]\n")
print()
print()
print("Please Wait...")
time.sleep(3)
while(Game == Running):
    os.system('cls')
    DrawBoard()
    if(player % 2 != 0):
        print("Player 1's chance")
        Mark = 'X'
    else:
        print("Player 2's chance")
        Mark = 'O'
    choice = int(input("Enter the position between [1-9] where you want to mark : "))
    if(CheckPosition(choice)):
        board[choice] = Mark
        player+=1
        CheckWin()

os.system('cls')
DrawBoard()
if(Game==Draw):
    print("Game Draw")
elif(Game==Win):
    player-=1
    if(player%2!=0):
        print("Player 1 Won")
    else:
        print("Player 2 Won")
```

Output:

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 0]
 [0 1 0]
 [0 0 0]]
Board after 2 move
[[0 0 0]
 [0 1 0]
 [0 2 0]]
Board after 3 move
[[0 0 0]
 [0 1 0]
 [1 2 0]]
Board after 4 move
[[0 0 2]
 [0 1 0]
 [1 2 0]]
Board after 5 move
[[0 0 2]
 [0 1 0]
 [1 2 1]]
Board after 6 move
[[0 0 2]
 [0 1 2]
 [1 2 1]]
Board after 7 move
[[0 0 2]
 [1 1 2]
 [1 2 1]]
Board after 8 move
[[2 0 2]
 [1 1 2]
 [1 2 1]]
Board after 9 move
[[2 1 2]
 [1 1 2]
 [1 2 1]]
Winner is: -1
```

Viva Questions:

1. **What is Stochastic Search?**
2. **What is MIN-MAX Search?**
3. **What is Alpha-Beta Pruning.**

NumPy is an open source library available in Python that aids in mathematical, scientific, engineering, and data science programming. NumPy is an incredible library to perform mathematical and statistical operations. It works perfectly well for multi-dimensional arrays and matrices multiplication

For any scientific project, NumPy is the tool to know. It has been built to work with the N- dimensional array, linear algebra, random number, Fourier transform, etc. It can be integrated toC/C++ and Fortran.

NumPy is a programming language that deals with multi-dimensional arrays and matrices. On top ofthe arrays and matrices, NumPy supports a large number of mathematical operations.

NumPy is memory efficiency, meaning it can handle the vast amount of data more accessible than anyother library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, TensorFlow and Scikit learn to use NumPy arrayto compute the matrix multiplication in the back end.

- **Arrays in NumPy:** NumPy's main object is the homogeneous multidimensional array.

  - It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positiveintegers.
  - In NumPy dimensions are called axes. The number of axes is rank.
  - NumPy's array class is called **ndarray**. It is also known by the alias **array**.

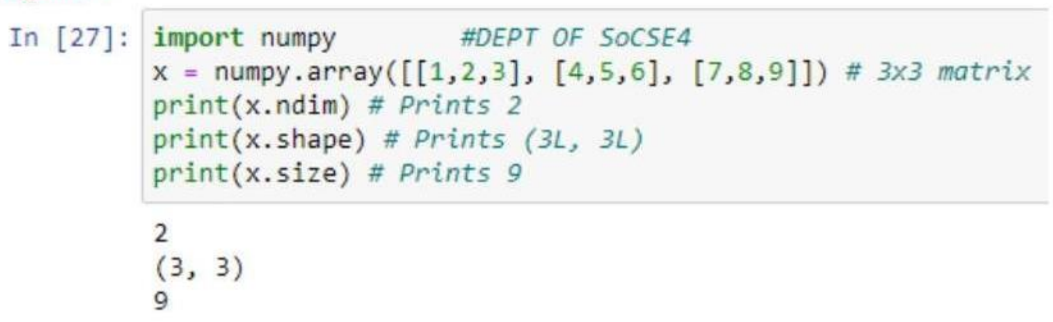
  We use python numpy array instead of a list because of the below three reasons:

  1. Less Memory
  2. Fast
  3. Convenient

- **Numpy Functions**

  Numpy arrays carry attributes around with them. The most important ones are:
  ndim: The number of axes or rank of the array
  shape: A tuple containing the length in each dimension
  size: The total number of elements

'rogram-1

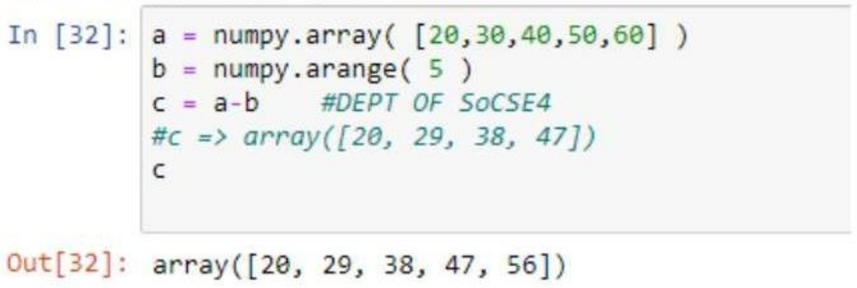


```
In [27]: import numpy         #DEPT OF SoCSE4
         x = numpy.array([[1,2,3], [4,5,6], [7,8,9]]) # 3x3 matrix
         print(x.ndim) # Prints 2
         print(x.shape) # Prints (3L, 3L)
         print(x.size) # Prints 9

         2
         (3, 3)
         9
```

'an be used just like Python lists
:[1] will access the second element
:[-1] will access the last element

'rogram-2

\rithmetic operations apply element wise

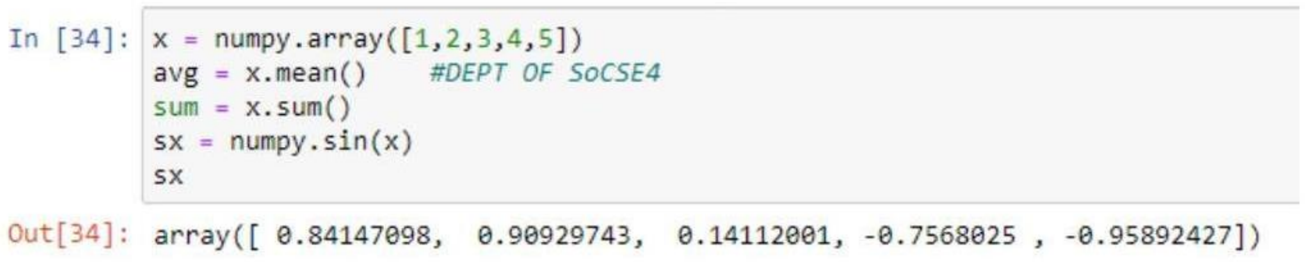


```
In [32]: a = numpy.array( [20,30,40,50,60] )
         b = numpy.arange( 5 )
         c = a-b     #DEPT OF SoCSE4
         #c => array([20, 29, 38, 47])
         c

Out[32]: array([20, 29, 38, 47, 56])
```

- **Built-in Methods**

  Many standard numerical functions are available as methods out of the box:

'rogram-3



```
In [34]: x = numpy.array([1,2,3,4,5])
         avg = x.mean()     #DEPT OF SoCSE4
         sum = x.sum()
         sx = numpy.sin(x)
         sx

Out[34]: array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.95892427])
```

- **Python Scipy Library**

  SciPy is an Open Source Python-based library, which is used in mathematics, scientific computing, Engineering, and technical computing. SciPy also pronounced as "Sigh Pi."

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

Numpy VS SciPyNumpy:

1. Numpy is written in C and use for mathematical or numeric calculation.
2. It is faster than other Python Libraries
3. Numpy is the most useful library for Data Science to perform basic calculations.
4. Numpy contains nothing but array data type which performs the most basic operation like sorting,shaping, indexing, etc.

SciPy:

1. SciPy is built in top of the NumPy
2. SciPy is a fully-featured version of Linear Algebra while Numpy contains only a few features.
3. Most new Data Science features are available in Scipy rather than Numpy.

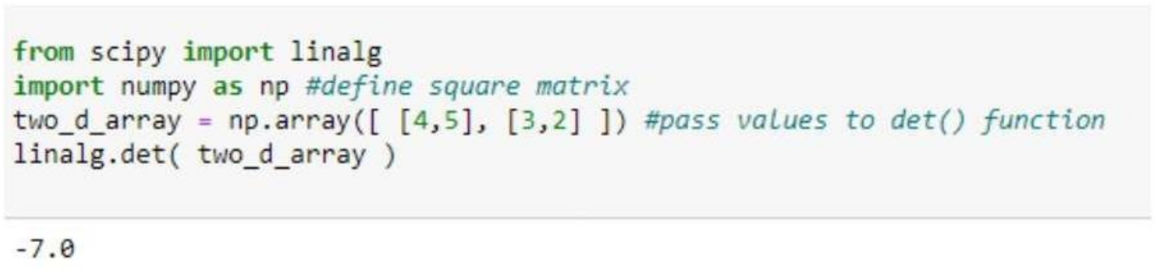
Linear Algebra with SciPy

1. Linear Algebra of SciPy is an implementation of BLAS and ATLAS LAPACK libraries.
2. Performance of Linear Algebra is very fast compared to BLAS and LAPACK.
3. Linear algebra routine accepts two-dimensional array object and output is also a two-dimensional array.

Now let's do some test with **scipy.linalg,**

Calculating **determinant** of a two-dimensional matrix,

Program-1



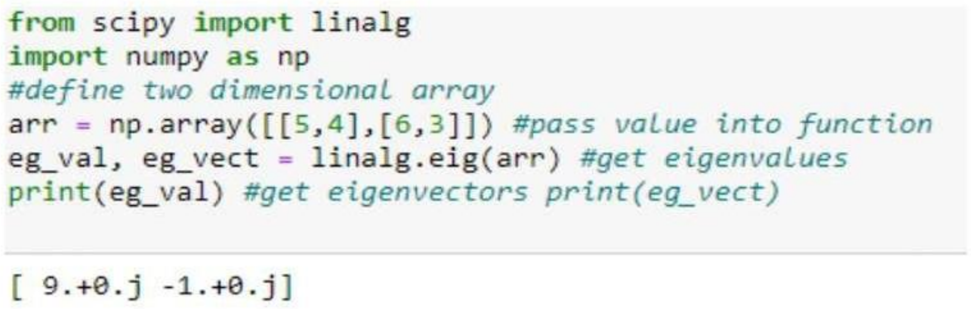
```
from scipy import linalg
import numpy as np #define square matrix
two_d_array = np.array([ [4,5], [3,2] ]) #pass values to det() function
linalg.det( two_d_array )
```

```
-7.0
```

Eigenvalues and Eigenvector – scipy.linalg.eig()

- The most common problem in linear algebra is eigenvalues and eigenvector which can beeasily solved using **eig()** function.
- Now lets we find the Eigenvalue of (**X**) and correspond eigenvector of a two-dimensionalsquare matrix.

Program-2



```
from scipy import linalg
import numpy as np
#define two dimensional array
arr = np.array([[5,4],[6,3]]) #pass value into function
eg_val, eg_vect = linalg.eig(arr) #get eigenvalues
print(eg_val) #get eigenvectors print(eg_vect)
```

```
[ 9.+0.j -1.+0.j]
```

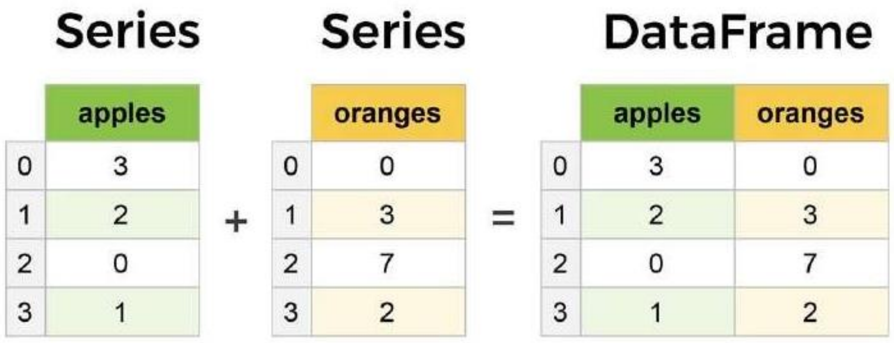
Exercise programs:

1. consider a list datatype then reshape it into 2d,3d matrix using numpy
2. genrate random matrices using numpy
3. find the determinant of a matrix using scipy
4. find eigenvalue and eigenvector of a matrix using scipy

b) Implementation of Python Libraries for ML application such as Pandas and Matplotlib.

- **Pandas Library**

The primary two components of pandas are the Series and DataFrame.
A Series is essentially a column, and a DataFrame is a multi-dimensional table made up of acollection of Series.
DataFrames and Series are quite similar in that many operations that you can do with one you can do with the other, such as filling in null values and calculating the mean.



Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

- **Reading data from CSVs**

With CSV files all you need is a single line to load in the data:

```
df =
pd.read_csv('purchases.csv')df
```

Let's load in the IMDB movies dataset to begin:

```
movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="Title")
```

We're loading this dataset from a CSV and designating the movie titles to be our index.

- **Viewing your data**

The first thing to do when opening a new dataset is print out a few rows to keep as a visualreference. We accomplish this with .head():

```
movies_df.head()
```

Another fast and useful attribute is .shape, which outputs just a tuple of (rows, columns):

```
movies_df.shape
```

Note that .shape has no parentheses and is a simple tuple of format (rows, columns). So we have1000 rows and 11 columns in our movies DataFrame.
You'll be going to .shape a lot when cleaning and transforming data. For example, you might filtersome rows based on some criteria and then want to know quickly how many rows were removed.

rogram-1

```
import pandas as pd
S = pd.Series([11, 28, 72, 3, 5, 8])
S
```

```
0    11
1    28
2    72
3     3
4     5
5     8
dtype: int64
```

Ve haven't defined an index in our example, but we see two columns in our output: The right column ontains our data, whereas the left column contains the index. Pandas created a default index starting with 0 oing to 5, which is the length of the data minus 1.

rogram-2

Ve can directly access the index and the values of our Series S:

```
print(S.index)
print(S.values)
```

```
RangeIndex(start=0, stop=6, step=1)
[11 28 72  3  5  8]
```

rogram-3

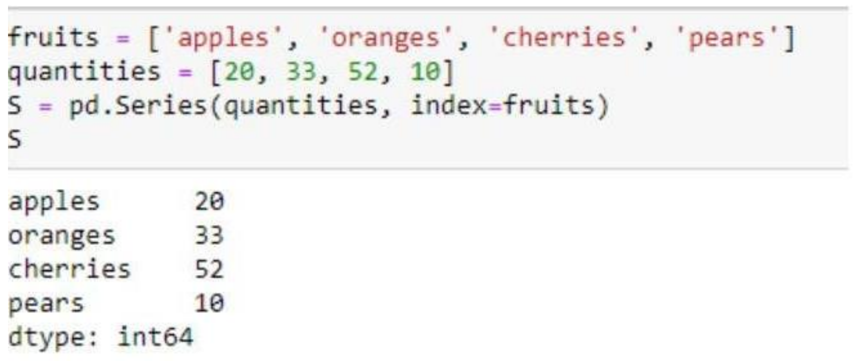
f we compare this to creating an array in numpy, we will find lots of similarities:

```
import numpy as np
X = np.array([11, 28, 72, 3, 5, 8])
print(X)
print(S.values)
# both are the same type:
print(type(S.values), type(X))
```

```
[11 28 72  3  5  8]
[11 28 72  3  5  8]
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

So far our Series have not been very different to ndarrays of Numpy. This changes, as soon as we start lefining Series objects with individual indices:

Program-4



```
fruits = ['apples', 'oranges', 'cherries', 'pears']
quantities = [20, 33, 52, 10]
S = pd.Series(quantities, index=fruits)
S
```

```
apples      20
oranges     33
cherries    52
pears       10
dtype: int64
```

Program-5

A big advantage to NumPy arrays is obvious from the previous example: We can use arbitrary indices.
If we add two series with the same indices, we get a new series with the same index and the correponding values will be added:

```
fruits = ['apples', 'oranges', 'cherries', 'pears']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits)
print(S + S2)
print("sum of S: ", sum(S))
```

OUTPUT:

```
    apples     37
    oranges     46
    cherries    83
    pears      42
    dtype: int64
    sum of S:  115
```

Program-6

The indices do not have to be the same for the Series addition. The index will be the "union" of both indices.
If an index doesn't occur in both Series, the value for this Series will be NaN:

```
fruits = ['peaches', 'oranges', 'cherries', 'pears']
fruits2 = ['raspberries', 'oranges', 'cherries', 'pears']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits2)
print(S + S2)
```

OUTPUT:

```
    cherries      83.0
    oranges       46.0
```

```
peaches        NaN
pears        42.0
raspberries    NaN
dtype: float64
```

Program-7

In principle, the indices can be completely different, as in the following example. We have two indices. One is the Turkish translation of the English fruit names:

```
fruits = ['apples', 'oranges', 'cherries', 'pears']

fruits_tr = ['elma', 'portakal', 'kiraz', 'armut']

S = pd.Series([20, 33, 52, 10], index=fruits)
S2 = pd.Series([17, 13, 31, 32], index=fruits_tr)
print(S + S2)
```

OUTPUT:

```
apples    NaN
armut     NaN
cherries   NaN
elma      NaN
kiraz     NaN
oranges   NaN
pears     NaN
portakal  NaN
dtype: float64
```

Program-8

Indexing

It's possible to access single values of a Series.

```
print(S['apples'])
```

OUTPUT:

```
20
```

- **Matplotlib Library**

  Pyplot is a module of Matplotlib which provides simple functions to add plot elements like lines, images, text, etc. to the current axes in the current figure.

 Make a simple plot
import matplotlib.pyplot as pltimport
numpy as np

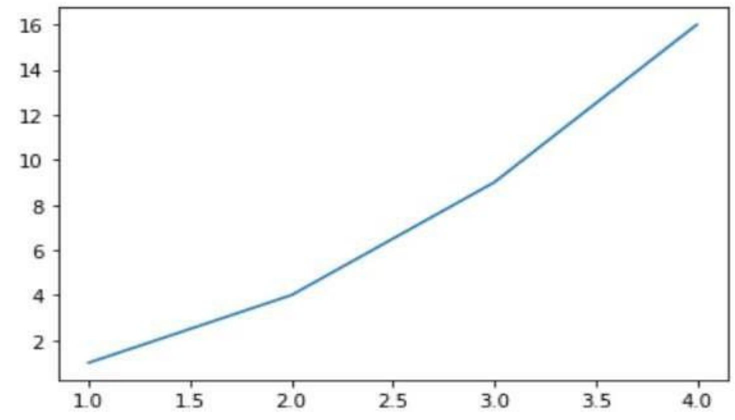
List of all the methods as they appeared.

 plot(x-axis values, y-axis values) — plots a simple line graph with x-axis values against y-axis values
 show() — displays the graph
 title(–string‖) — set the title of the plot as specified by the string
 xlabel(–string‖) — set the label for x-axis as specified by the string
 ylabel(–string‖) — set the label for y-axis as specified by the string
 figure() — used to control a figure level attributes
 subplot(nrows, ncols, index) — Add a subplot to the current figure
 suptitle(–string‖) — It adds a common title to the figure specified by the string
 subplots(nrows, ncols, figsize) — a convenient way to create subplots, in a single call. It returns a tuple of a figure and number of axes.
 set_title(–string‖) — an axes level method used to set the title of subplots in a figure
 bar(categorical variables, values, color) — used to create vertical bar graphs
 barh(categorical variables, values, color) — used to create horizontal bar graphs
 legend(loc) — used to make legend of the graph
 xticks(index, categorical variables) — Get or set the current tick locations and labels of the x-axis
 pie(value, categorical variables) — used to create a pie chart
 hist(values, number of bins) — used to create a histogram
 xlim(start value, end value) — used to set the limit of values of the x-axis
 ylim(start value, end value) — used to set the limit of values of the y-axis
 scatter(x-axis values, y-axis values) — plots a scatter plot with x-axis values against y-axis values
 axes() — adds an axes to the current figure
 set_xlabel(–string‖) — axes level method used to set the x-label of the plot specified as a string
 set_ylabel(–string‖) — axes level method used to set the y-label of the plot specified as a string
 scatter3D(x-axis values, y-axis values) — plots a three-dimensional scatter plot with x-axis values against y-axis values
 plot3D(x-axis values, y-axis values) — plots a three-dimensional line graph with x-axis values against y-axis values

Here we import Matplotlib‘s Pyplot module and Numpy library as most of the data that we will be working with will be in the form of arrays only.

Program-1

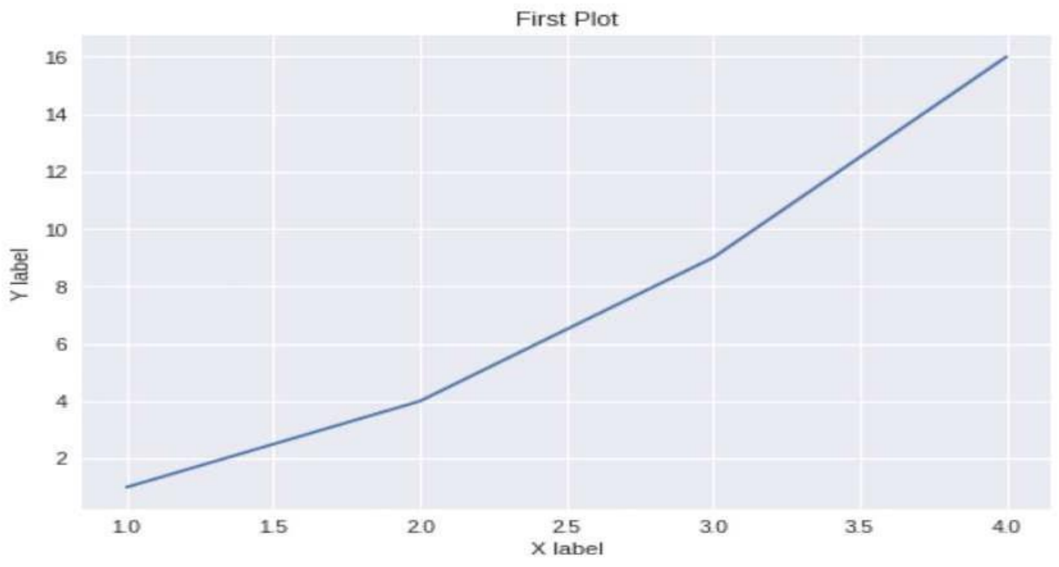
```
import matplotlib.pyplot as plt
import numpy as np
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



We pass two arrays as our input arguments to Pyplot's plot() method and use show() method to invoke the required plot. Here note that the first array appears on the x-axis andsecond array appears on the y-axis of the plot. Now that our first plot is ready, let us add the title, and name x-axis and y-axis using methods title(), xlabel() and ylabel() respectively.

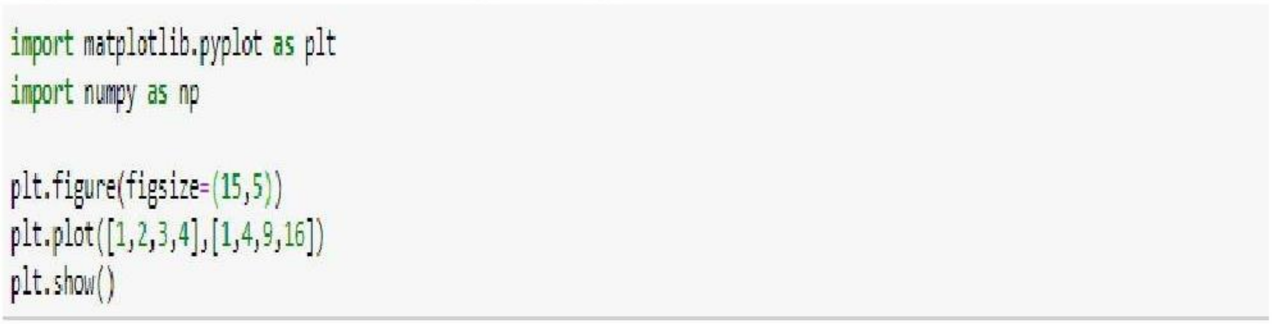
Program-2

```
plt.plot([1,2,3,4],[1,4,9,16])
plt.title("First Plot")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```

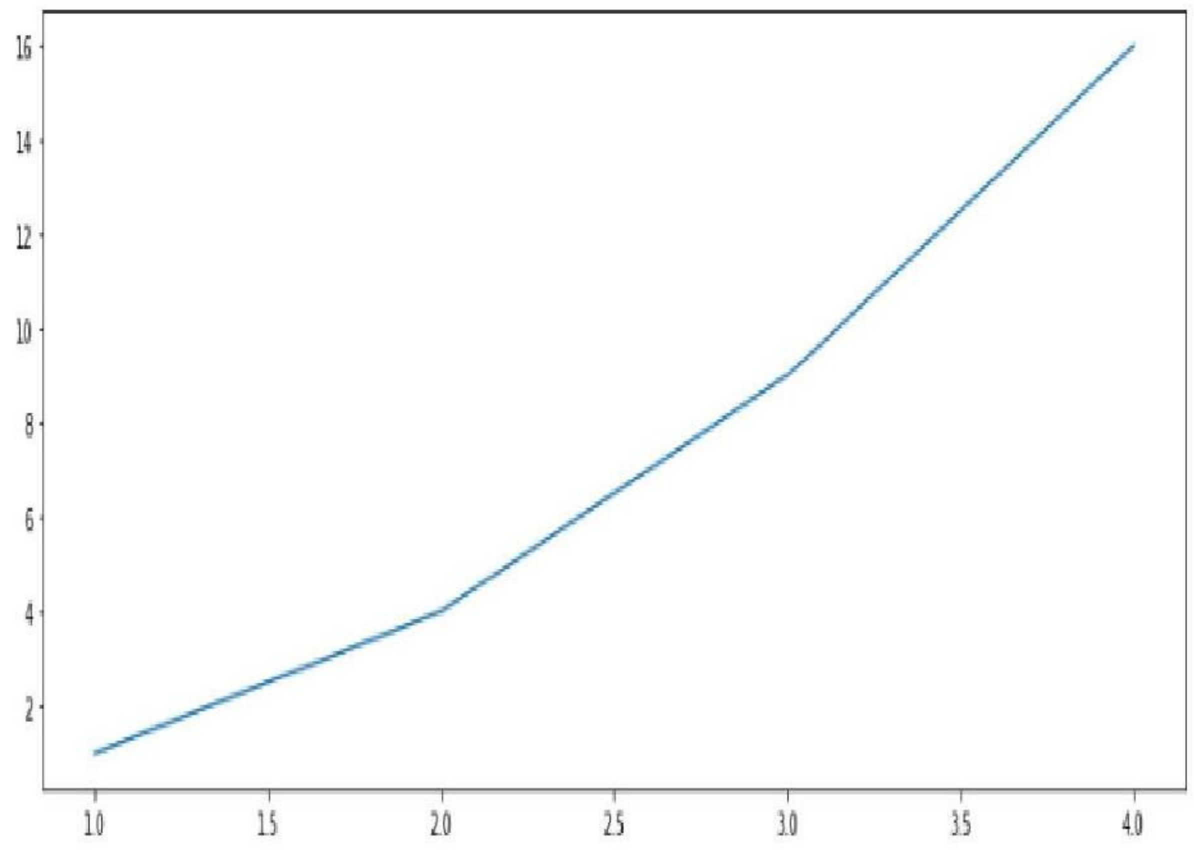


Program-3

We can also specify the size of the figure using method figure()and passing the valuesas a tuple of the length of rows and columns to the argument figsize



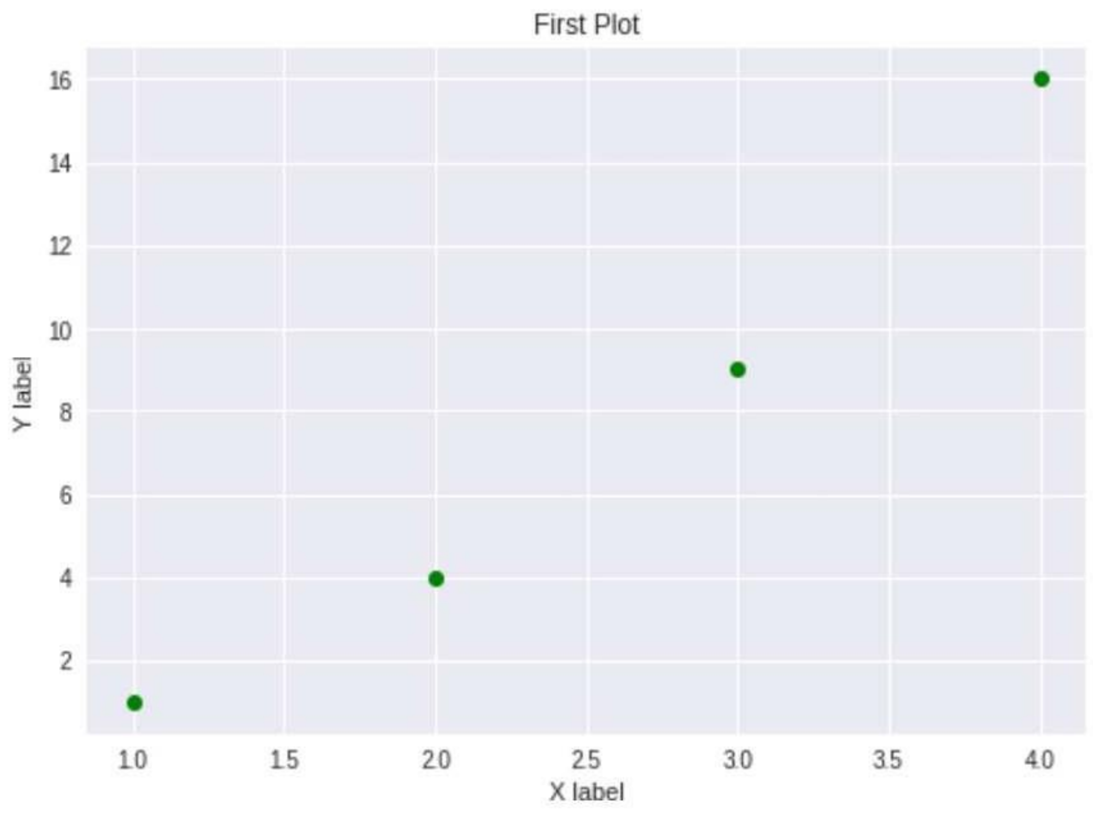
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(15,5))
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```



Program-4

With every X and Y argument, you can also pass an optional third argument in the formof a string which indicates the colour and line type of the plot. The default format is **b-** which means a solid blue line. In the figure below we use **go** which means green circles.Likewise, we can make many such combinations to format our plot.

```
plt.plot([1,2,3,4],[1,4,9,16],"go")
plt.title("First Plot")
plt.xlabel("X label")
plt.ylabel("Y label")
plt.show()
```

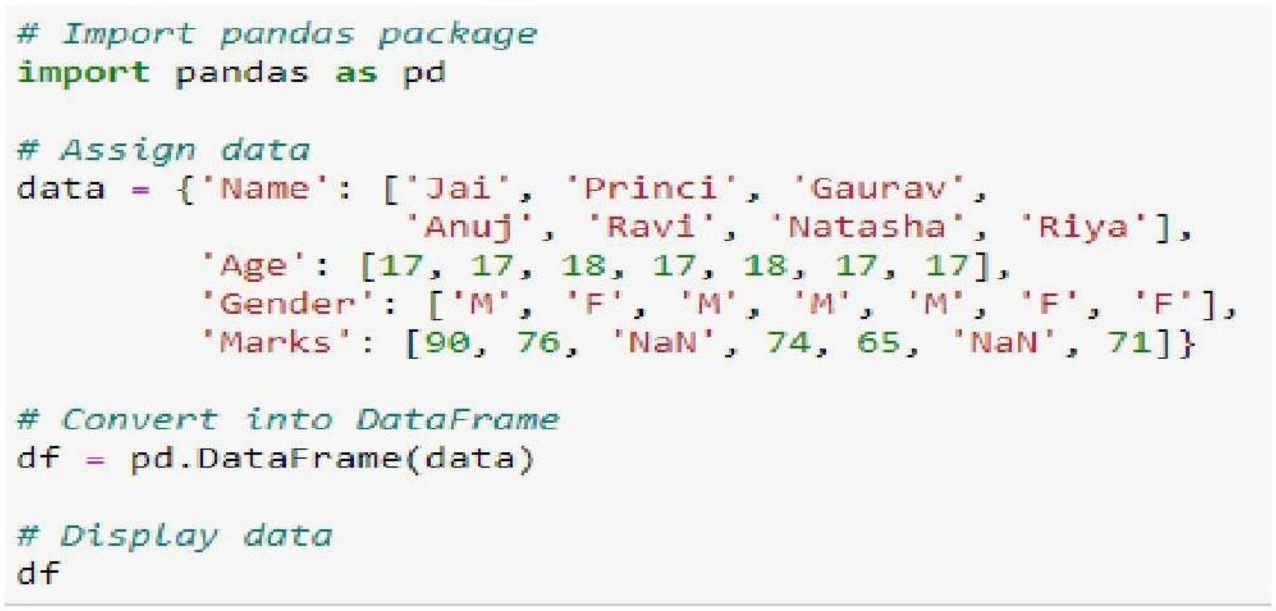


WEEK-2

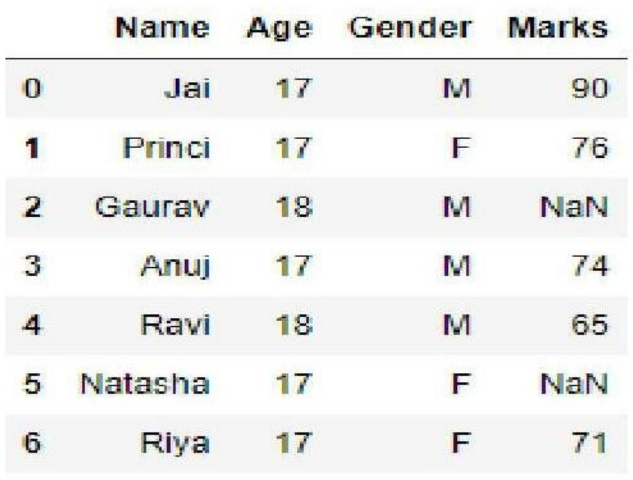
a) Creation and Loading different datasets in Python

Program-1

Method-I



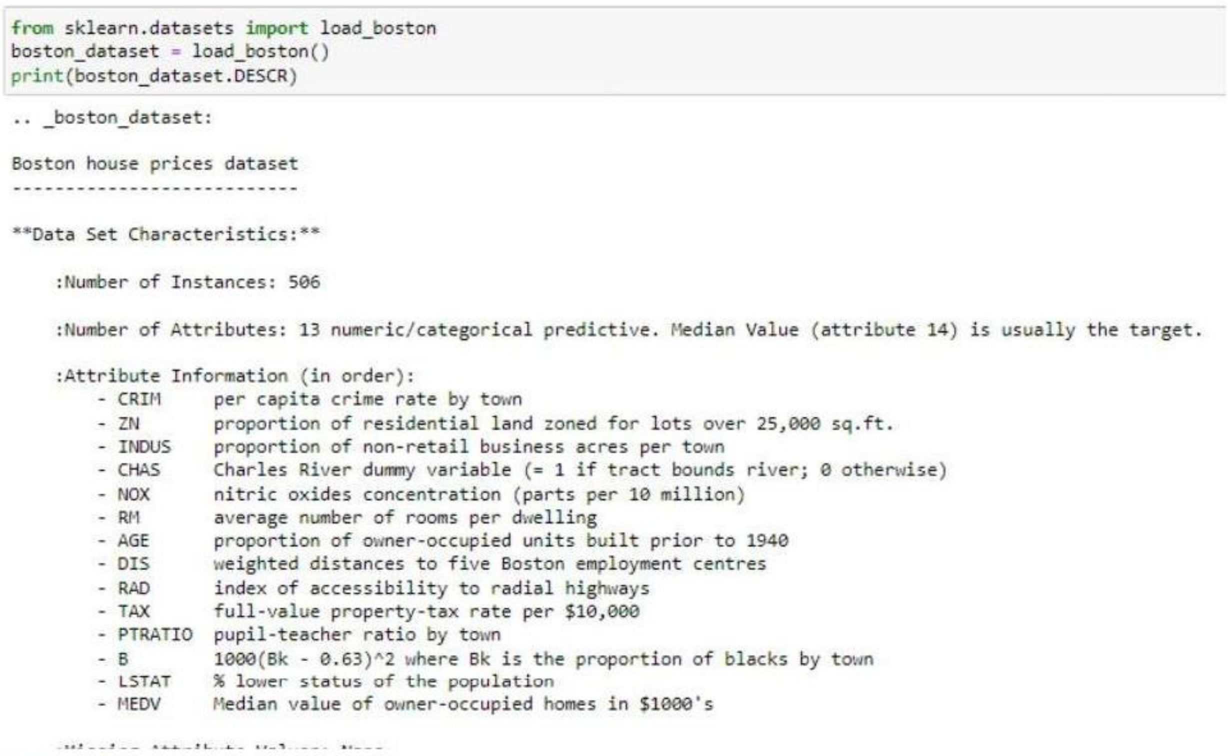
```python
# Import pandas package
import pandas as pd

# Assign data
data = {'Name': ['Jai', 'Princi', 'Gaurav',
                 'Anuj', 'Ravi', 'Natasha', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```



```
      Name  Age  Gender  Marks
0      Jai   17       M     90
1   Princi   17       F     76
2   Gaurav   18       M    NaN
3     Anuj   17       M     74
4     Ravi   18       M     65
5  Natasha   17       F    NaN
6     Riya   17       F     71
```

Program-2

Method-II:

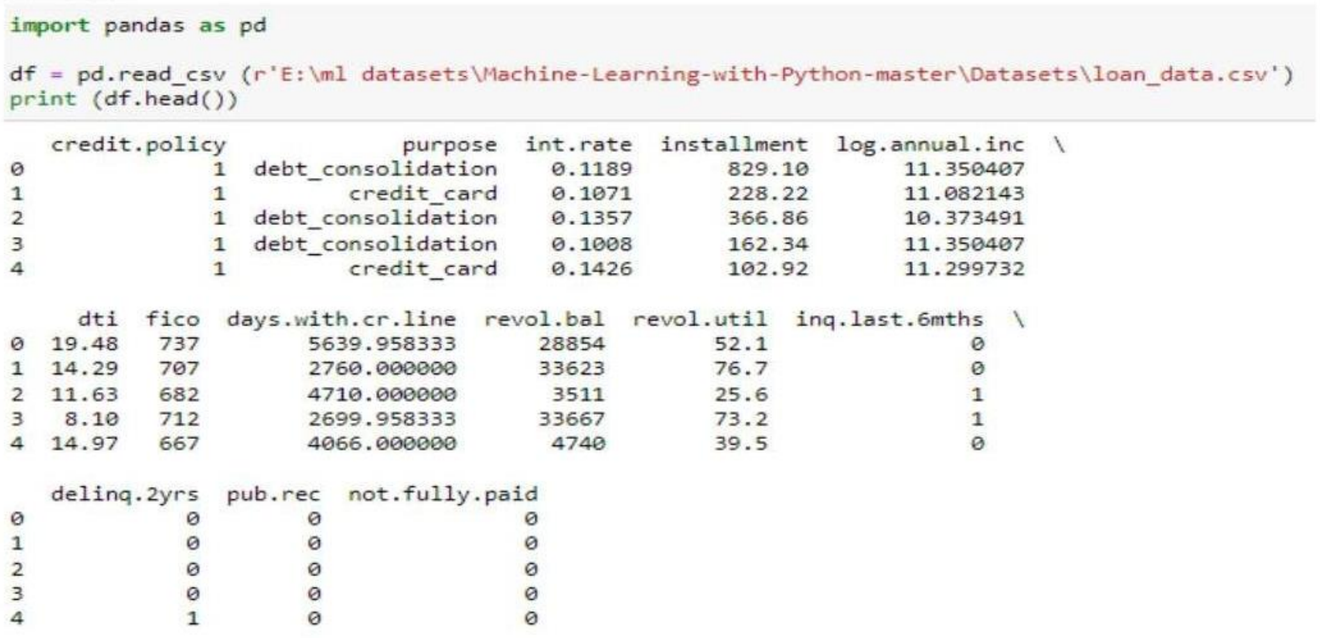


```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
print(boston_dataset.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's
```

Program-3 **Uploading csv file:**

Method-III:



```
import pandas as pd

df = pd.read_csv (r'E:\ml datasets\Machine-Learning-with-Python-master\Datasets\loan_data.csv')
print (df.head())
```

```
   credit.policy             purpose  int.rate  installment  log.annual.inc  \
0              1  debt_consolidation    0.1189       829.10       11.350407
1              1         credit_card    0.1071       228.22       11.082143
2              1  debt_consolidation    0.1357       366.86       10.373491
3              1  debt_consolidation    0.1008       162.34       11.350407
4              1         credit_card    0.1426       102.92       11.299732

     dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  \
0  19.48   737        5639.958333      28854        52.1               0
1  14.29   707        2760.000000      33623        76.7               0
2  11.63   682        4710.000000       3511        25.6               1
3   8.10   712        2699.958333      33667        73.2               1
4  14.97   667        4066.000000       4740        39.5               0

   delinq.2yrs  pub.rec  not.fully.paid
0            0        0               0
1            0        0               0
2            0        0               0
3            0        0               0
4            1        0               0
```

b) Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using Datasets

- **Python Statistics library**

  This module provides functions for calculating mathematical statistics of numeric (Real-valued) data. The statistics module comes with very useful functions like: Mean, median, mode, standard deviation,and variance.

  The four functions we'll use in this post are common in statistics:

  1. mean - average value
  2. median - middle value
  3. mode - most often value
  4. standard deviation - spread of values

- **Averages and measures of central location**

  These functions calculate an average or typical value from a population or

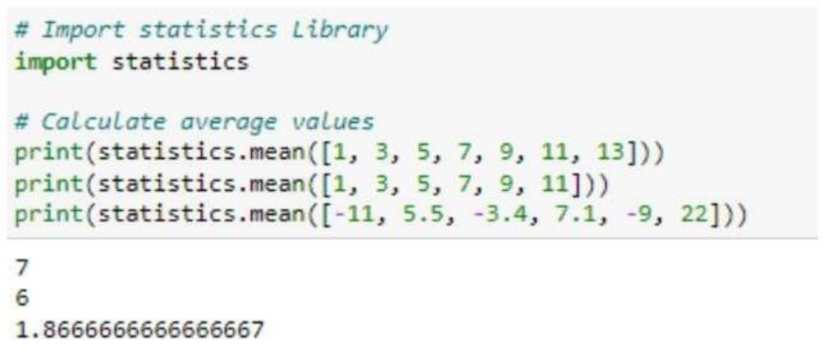
sample.mean()	Arithmetic mean (–average‖) of data.
harmonic_mean()	Harmonic mean of data.
median()	Median (middle value) of
data.median_low()	Low median of data.
median_high()	High median of data.
median_grouped()	Median, or 50th percentile, of grouped
data.mode()	Mode (most common value) of discrete
data.	

- **Measures of spread**

  These functions calculate a measure of how much the population or sample tends to deviate fromthe typical or average values.

pstdev()	Population standard deviation of data.
pvariance()	Population variance of data.
stdev()	Sample standard deviation of data.
variance()	Sample variance of data.

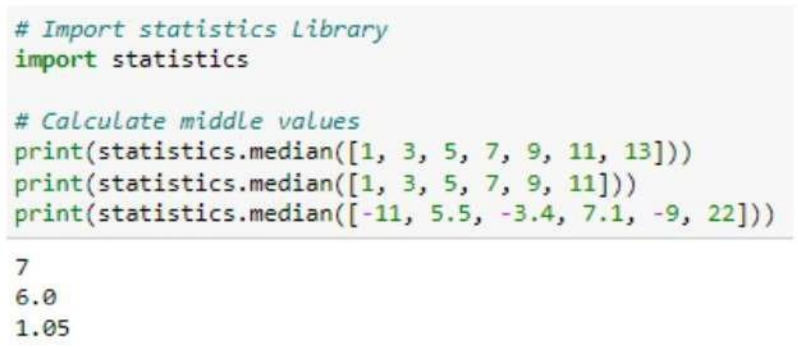
Program-1



```
# Import statistics Library
import statistics

# Calculate average values
print(statistics.mean([1, 3, 5, 7, 9, 11, 13]))
print(statistics.mean([1, 3, 5, 7, 9, 11]))
print(statistics.mean([-11, 5.5, -3.4, 7.1, -9, 22]))
```

```
7
6
1.8666666666666667
```

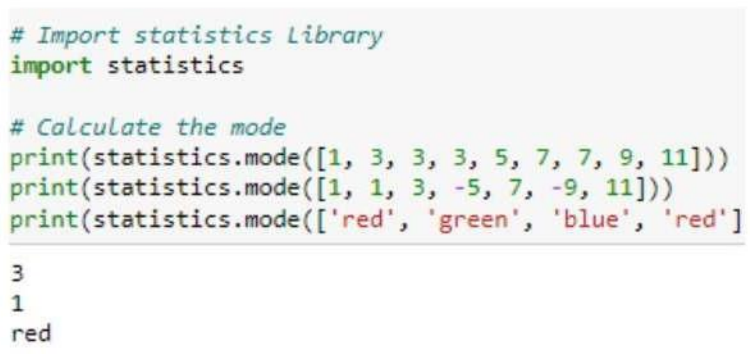
Program-2



```
# Import statistics Library
import statistics

# Calculate middle values
print(statistics.median([1, 3, 5, 7, 9, 11, 13]))
print(statistics.median([1, 3, 5, 7, 9, 11]))
print(statistics.median([-11, 5.5, -3.4, 7.1, -9, 22]))
```

```
7
6.0
1.05
```

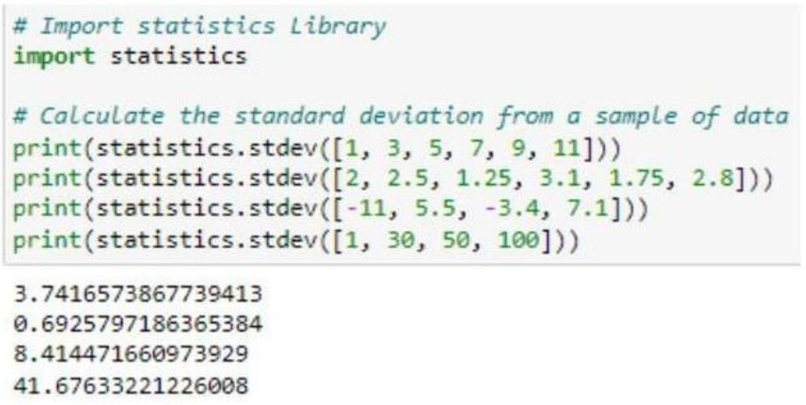
Program-3



```
# Import statistics Library
import statistics

# Calculate the mode
print(statistics.mode([1, 3, 3, 3, 5, 7, 7, 9, 11]))
print(statistics.mode([1, 1, 3, -5, 7, -9, 11]))
print(statistics.mode(['red', 'green', 'blue', 'red']
```

```
3
1
red
```

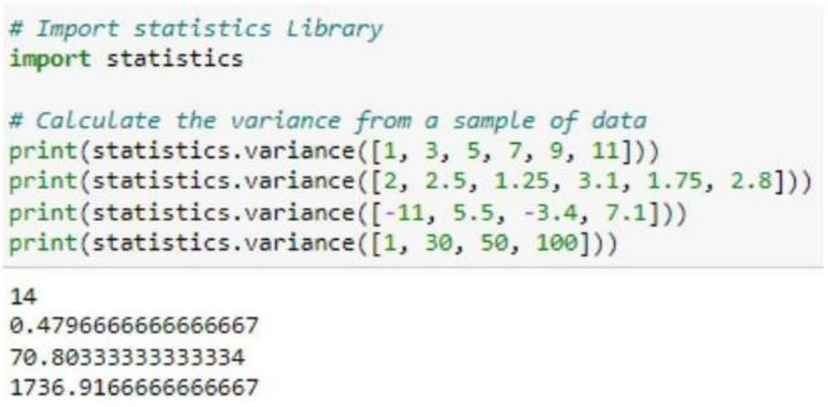
Program-4



```python
# Import statistics Library
import statistics

# Calculate the standard deviation from a sample of data
print(statistics.stdev([1, 3, 5, 7, 9, 11]))
print(statistics.stdev([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.stdev([-11, 5.5, -3.4, 7.1]))
print(statistics.stdev([1, 30, 50, 100]))
```

```
3.7416573867739413
0.6925797186365384
8.414471660973929
41.67633221226008
```

Program-5



```python
# Import statistics Library
import statistics

# Calculate the variance from a sample of data
print(statistics.variance([1, 3, 5, 7, 9, 11]))
print(statistics.variance([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.variance([-11, 5.5, -3.4, 7.1]))
print(statistics.variance([1, 30, 50, 100]))
```

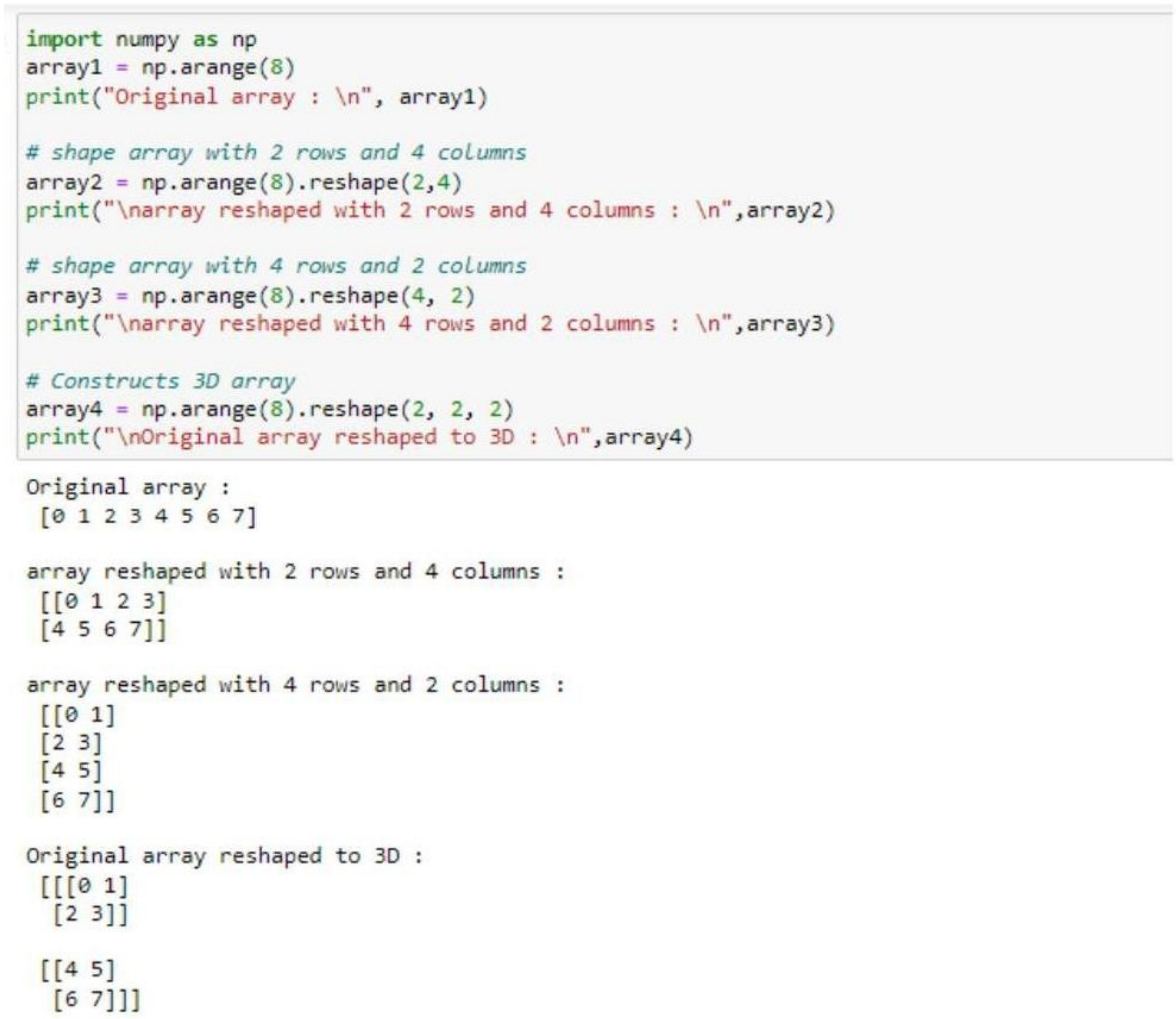
```
14
0.4796666666666667
70.80333333333334
1736.9166666666667
```

c) Write a python program to compute reshaping the data, Filtering the data , merging the data and handling the missing values in datasets.

Program-1

Reshaping the data:

Method-I

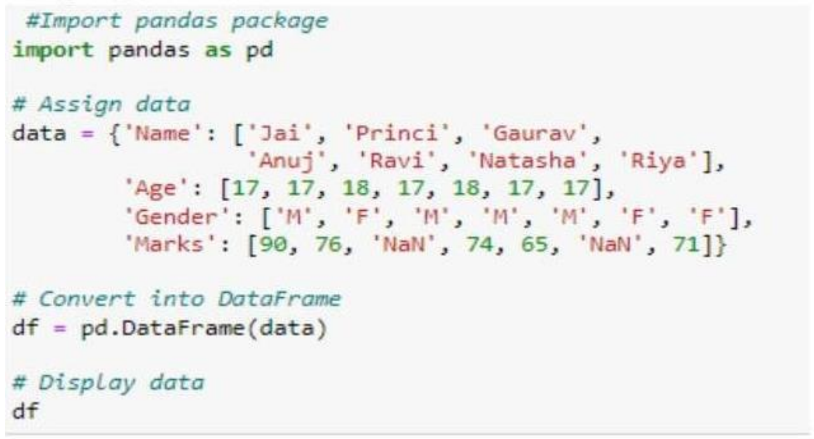


```python
import numpy as np
array1 = np.arange(8)
print("Original array : \n", array1)

# shape array with 2 rows and 4 columns
array2 = np.arange(8).reshape(2,4)
print("\narray reshaped with 2 rows and 4 columns : \n",array2)

# shape array with 4 rows and 2 columns
array3 = np.arange(8).reshape(4, 2)
print("\narray reshaped with 4 rows and 2 columns : \n",array3)

# Constructs 3D array
array4 = np.arange(8).reshape(2, 2, 2)
print("\nOriginal array reshaped to 3D : \n",array4)
```

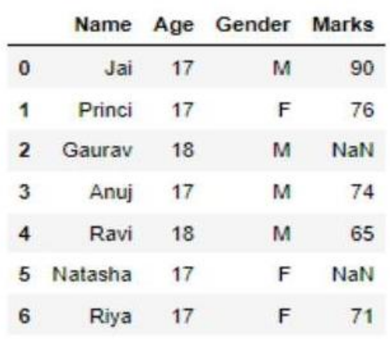
```
Original array :
 [0 1 2 3 4 5 6 7]

array reshaped with 2 rows and 4 columns :
 [[0 1 2 3]
 [4 5 6 7]]

array reshaped with 4 rows and 2 columns :
 [[0 1]
 [2 3]
 [4 5]
 [6 7]]

Original array reshaped to 3D :
 [[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]
```

Program-2

Method:II
Assigning the data:

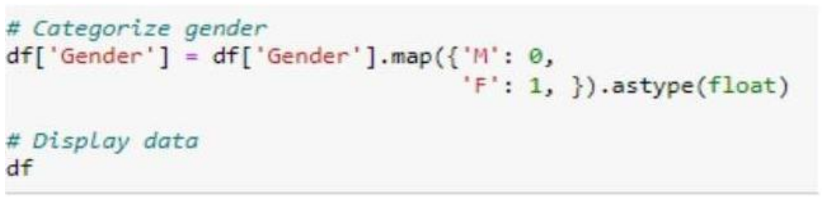


```
 #Import pandas package
import pandas as pd

# Assign data
data = {'Name': ['Jai', 'Princi', 'Gaurav',
                 'Anuj', 'Ravi', 'Natasha', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```

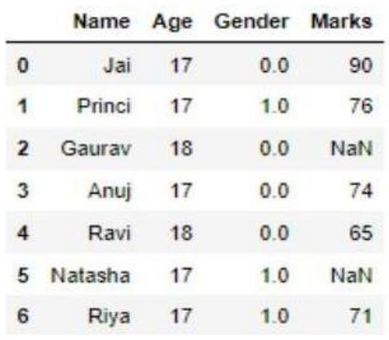


	Name	Age	Gender	Marks
0	Jai	17	M	90
1	Princi	17	F	76
2	Gaurav	18	M	NaN
3	Anuj	17	M	74
4	Ravi	18	M	65
5	Natasha	17	F	NaN
6	Riya	17	F	71

Program-3



```
# Categorize gender
df['Gender'] = df['Gender'].map({'M': 0,
                                 'F': 1, }).astype(float)

# Display data
df
```



	Name	Age	Gender	Marks
0	Jai	17	0.0	90
1	Princi	17	1.0	76
2	Gaurav	18	0.0	NaN
3	Anuj	17	0.0	74
4	Ravi	18	0.0	65
5	Natasha	17	1.0	NaN
6	Riya	17	1.0	71