PERIYAR UNIVERSITY

(NAAC 'A++' Grade with CGPA 3.61 (Cycle - 3)
State University - NIRF Rank 56 - State Public University Rank 25
SALEM - 636 011

CENTRE FOR DISTANCE AND ONLINE EDUCATION

(CDOE)

MASTER OF COMPUTER APPLICATION

SEMESTER - I

CORE V: PYTHON PROGRAMMING LAB

(Candidates admitted from 2024 onwards)

PERIYAR UNIVERSITY

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

M.C.A 2024 admission onwards

CORE – V
Python Programming Lab

Prepared by:

Centre for Distance and Online Education (CDOE)
Periyar University
Salem - 636011

LIST OF CONTENTS

PROGRAM USING ELEMENTARY DATA ITEMS, LIST, DICTIONARIES AND TUPLES

ALGORITHM FOR LIST

Step 1: Start the Program

Step 2: Initialize an empty list list_1.

Step 3: Prompt the user to enter the total number of list items and store the input in the variable num.

Step 4: Loop num times to get list items from the user.

- Step 4.1: Within the loop, prompt the user to enter an item.
- Step 4.2: Append the item to list_1.

Step 5: Print the current list elements.

Step 6: Prompt the user if they want to add another item (yes or no).

- Step 6.1: If the user chooses 'yes', prompt the user to enter an item.
- Step 6.2: Append the item to list_1.
- Step 6.3: Print the updated list elements.

Step 7: Prompt the user if they want to remove an item (yes or no).

- Step 7.1: If the user chooses 'yes', prompt the user to enter the index value of the item to delete.
- Step 7.2: Remove the item at the specified index from list_1.
- Step 7.3: Print the updated list elements.

Step 8: Prompt the user if they want to reverse the list (yes or no).

- Step 8.1: If the user chooses 'yes', reverse the order of elements in list_1.
- Step 8.2: Print the reversed list elements.

Step 9: Prompt the user if they want to sort the list (yes or no).

- Step 9.1: If the user chooses 'yes', sort list_1 alphanumerically.
- Step 9.2: Print the sorted list elements.

Step 10: Prompt the user if they want to clear the list (yes or no).

- Step 10.1: If the user chooses 'yes', clear all elements from list_1.
- Step 10.2: Print the empty list.
- Step 11: End the Program

SOURCE CODE FOR LIST

```
print("{{{{{{{{{...LIST...}}}}}}}}}")
list_1=[]
num=int(input("Enter the total list item : "))for i in
range(num):
        item=str(input("Enter item to present in list : "))
        list_1.append(item)
print("List element are : ",list_1)
choice=input("Do you want to add another element ? (y / n) :")if
choice.casefold()=='y':
        item=str(input("Enter item to present in list : "))
        list_1.append(item)
        print("List element are : ",list_1)
choice=input("Do you want to remove element ? (y / n) :")if
choice.casefold()=='y':
        print("Enter the index value to delete")
        index=int(input())
        list_1.pop(index)
        print("List element are :",list_1)
choice=input("Do you want to reverse the element ? (y / n) :")if
choice.casefold()=='y':
        list_1.reverse()
        print("Reverseing the order of the list :",list_1) choice=input("Do
you want to Sort the element ? (y / n) :")
if choice.casefold()=='y':
        list_1.sort()
        print("Sorting the list alphanumerically :",list_1) choice=input("Do
you want to Clear the element ? (y / n) :")if choice.casefold()=='y':
        list_1.clear()
        print(list_1)
if choice.casefold()=='n':pass
```

OUTPUT

```
{{{{{{{{...LIST...}}}}}}}}
Enter the total list item : 3
Enter item to present in list : Dhanush
Enter item to present in list : Hari
Enter item to present in list : Lachu
List element are :  ['Dhanush', 'Hari', 'Lachu']
Do you want to add another element ? (y / n) :y
Enter item to present in list : Sanjai
List element are :  ['Dhanush', 'Hari', 'Lachu', 'Sanjai']
Do you want to remove element ? (y / n) :y
Enter the index value to delete
2
List element are : ['Dhanush', 'Hari', 'Sanjai']
Do you want to reverse the element ? (y / n) :y
Reverseing the order of the list : ['Sanjai', 'Hari', 'Dhanush']
Do you want to Sort the element ? (y / n) :y
Sorting the list alphanumerically : ['Dhanush', 'Hari', 'Sanjai']
Do you want to Clear the element ? (y / n) :y
[]
```

ALGORITHM FOR TUPLE

Step 1: Start the Program

Step 2: Initialize an empty list Tuple_1.

Step 3: Prompt the user to enter the total number of tuple items and store the input in the variable numT.

Step 4: Loop from 1 to numT (inclusive) to get tuple items from the user.

- Step 4.1: Within the loop, prompt the user to enter an item.
- Step 4.2: Convert the list to a tuple and append the item to Tuple_1.

Step 5: Print the current tuple elements.

Step 6: Prompt the user if they want to see the position of an element (yes or no).

- Step 6.1: If the user chooses 'yes', prompt the user to enter an element.
- Step 6.2: Print the index (position) of the specified element in Tuple_1.

Step 7: Prompt the user if they want to see the count for a specified element (yes or no).

- Step 7.1: If the user chooses 'yes', prompt the user to enter an element.
- Step 7.2: Print the count of the specified element in Tuple_1.

Step 8: End the Program

SOURCE CODE FOR TUPLE

```
print("{{{{{{{{{...TUPLE...}}}}}}}}}")
Tuple_1=[]
numT=int(input("Enter the total Tuple item : "))for i in
range(1,numT+1):
    value=int(input("Enter tuple item : "))
    Tuple_1+=(value,)
print("Tuple element are: ",Tuple_1)
choice=input("Do you want to see the position of element ? (y / n) :")if
choice.casefold()=='y':
    a=int(input("Enter a element to see a position : "))
    print(Tuple_1.index(a))
choice=input("Do you want to see the count for specified element ?(y / n):")
if choice.casefold()=='y':
    a=int(input("Enter a element to see a count : "))
    print(Tuple_1.count(a))
if choice.casefold()=='n':pass
```

OUTPUT

```
{{{{{{{{...TUPLE...}}}}}}}}
Enter the total Tuple item : 3
Enter tuple item : 26
Enter tuple item : 27
Enter tuple item : 14
Tuple element are:  [26, 27, 14]
Do you want to see the position of element ? (y / n) :y
Enter a element to see a position : 14
2
```

ALGORITHM FOR DICTIONARY

Step 1: Start the Program

Step 2: Initialize two empty dictionaries worker_1 and worker_2.

Step 3: Prompt the user to enter the total number of items and store the input in the variable numD.

Step 4: Loop numD times to get worker details from the user.

Step 4.1: Within the loop, prompt the user to enter the name.

Step 4.2: Prompt the user to enter the salary.

Step 4.3: Add the name and salary as a key-value pair to worker_1.

Step 5: Print the contents of worker_1.

Step 6: Prompt the user if they want to copy the dictionary (yes or no).

Step 6.1: If the user chooses 'yes', copy worker_1 to worker_2.

Step 6.2: Print the contents of worker_2.

Step 7: Prompt the user if they want to clear the dictionary (yes or no).

Step 7.1: If the user chooses 'yes', clear all elements from worker_1.

Step 7.2: Print the contents of worker_1 and worker_2.

Step 8: End the Program

SOURCE CODE FOR DICTIONARY

```
print("{{{{{{{{{...DICTIONARY...}}}}}}}}}")
worker_1={}
worker_2={}
numD=int(input("Enter the total number of items : "))for i in
range(numD):
    name=input("Enter Name : ")
    salary=input("Enter Salary : ")
    worker_1[name]=salary
print("Workers_1 :",(worker_1))
choice=input("Do you want to copy the Dictionary ? (y / n) :")if
choice.casefold()=='y':
    worker_2=worker_1.copy()
    print("Worker_2 : ",(worker_2))
choice=input("Do you want to clear the Dictionary ? (y / n) :")if
choice.casefold()=='y':
    worker_1.clear() print("Worker_1 :
    ",(worker_1))print("Worker_2 :
    ",(worker_2))
if choice.casefold()=='n':pass
```

OUTPUT

```
{{{{{{{{...DICTIONARY...}}}}}}}}
Enter the total number of items : 2
Enter Name : Dhaunush
Enter Salary : 1000
Enter Name : Hari
Enter Salary : 7000
Workers_1 : {'Dhaunush': '1000', 'Hari': '7000'}
Do you want to copy the Dictionary ? (y / n) :y
Worker_2 :  {'Dhaunush': '1000', 'Hari': '7000'}
Do you want to clear the Dictionary ? (y / n) :y
Worker_1 :  {}
Worker_2 :  {'Dhaunush': '1000', 'Hari': '7000'}
```

RESULT

Thus the above program has been executed successfully.

PROGRAM USING CONDITIONALBRANCHES AND LOOPS

Step 1: Start the Program

Step 2: Define four functions for basic arithmetic operations:

- Step 2.1: Define add(num1, num2) to return the sum of num1 and num2.
- Step 2.2: Define sub(num1, num2) to return the difference of num1 and num2.
- Step 2.3: Define mul(num1, num2) to return the product of num1 and num2.
- Step 2.4: Define div(num1, num2) to return the quotient of num1 and num2.

Step 3: Print the available operations:

- Print "1. Addition"
- Print "2. Subtraction"
- Print "3. Multiplication"
- Print "4. Division"

Step 4: Enter a loop to continually prompt the user for input until they decide to quit.

- Step 4.1: Prompt the user to enter their choice (1/2/3/4) and store it in the variable choice.

Step 5: Check if the user's choice is valid (i.e., one of '1', '2', '3', '4').

- Step 5.1: If the choice is valid, prompt the user to enter two numbers, num1 and num2.

Step 6: Perform the corresponding arithmetic operation based on the user's choice.

- Step 6.1: If choice is '1', call the add function and print the result.
- Step 6.2: If choice is '2', call the sub function and print the result.
- Step 6.3: If choice is '3', call the mul function and print the result.
- Step 6.4: If choice is '4', call the div function and print the result.

Step 7: Prompt the user to enter 'Q' to quit the calculator.

- Step 7.1: If the user inputs 'Q' (case insensitive), break the loop and end the program.
- Step 7.2: If the user inputs anything else, continue the loop.

Step 8: If the user's choice is not valid, print "Invalid Input".

Step 9: End the Program

SOURCE CODE

```
def add(num1, num2):
    return num1 + num2
def sub(num1, num2):
return num1 - num2 def
mul(num1, num2):return
num1 * num2
def div(num1, num2):
    return num1 / num2
print("{{{{...Select Operation to Perform...}}}}")
print("1.Addition")
print("2.Subtraction")
print("3.Multiplication")
print("4.Division") while True:
    choice=input("Enter your choice (1/2/3/4) :")if choice in
    ('1','2','3','4'):
        num1=float(input("Enter the First number :")) num2=float(input("Enter the Second
        number :"))
    if choice =='1': print(num1,"+",num2,"=",add(num1,num2))
    elif choice =='2':
        print(num1,"-",num2,"=",sub(num1,num2))elif choice
    =='3':
        print(num1,"*",num2,"=",mul(num1,num2))elif choice
    =='4':
        print(num1,"/",num2,"=",div(num1,num2))quit=input("Enter Q
    to quit calculator:")
    if quit.casefold()=="q":break
    else:
         print("Invalid Input")
```

OUTPUT

```
{{{{...Select Operation to Perform...}}}}
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice (1/2/3/4) :1
Enter the First number :10
Enter the Second number :55
10.0 + 55.0 = 65.0
Enter Q to quit calculator:q
```

RESULT

Thus the above program has been executed successfully.

PROGRAM USING FUNCTIONS

Step 1: Start the Program

Step 2: Prompt the user to enter any number and store the input in the variable num.

Step 3: Define the function cal_factorial(num) to calculate the factorial of num:

- Step 3.1: Initialize a variable fact to 1.
- Step 3.2: Check if num is 0 or 1.
    - Step 3.2.1: If num is 0 or 1, return 1 (since 0! and 1! are both 1).
- Step 3.3: Loop from 1 to num (inclusive) to calculate the factorial.
    - Step 3.3.1: Within the loop, multiply fact by the current loop index i.
- Step 3.4: Return the final value of fact.

Step 4: Call the function cal_factorial(num) with the input number and store the result in the variable output.

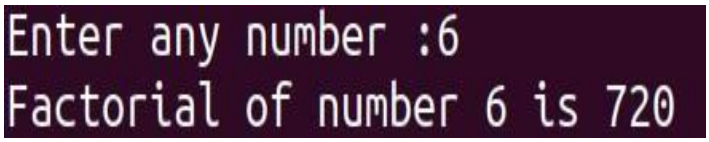
Step 5: Print the factorial of the number using the value of output.

Step 6: End the Program

SOURCE CODE

```
num=int(input("Enter any number :"))def
cal_factorial(num):
    fact=1
    if num==0 or num==1:
        return 1
    for i in range(1,num+1):
        fact=fact*i
    return fact
output=cal_factorial(num)
print("Factorial of number",num,"is",output)
```

OUTPUT



RESULT

Thus the above program has been executed successfully.

PROGRAM USING CLASSES AND OBJECTS

Step 1: Start the Program

Step 2: Define the Bank_Account class.

Step 3: Define the __init__ method to initialize the balance to 0 and print a welcome message.

- Step 3.1: Initialize self.balance to 0.
- Step 3.2: Print "Welcome to the Deposit & Withdrawal Machine".

Step 4: Define the deposit method to deposit an amount into the account.

- Step 4.1: Prompt the user to enter the amount to be deposited.
- Step 4.2: Add the entered amount to self.balance.
- Step 4.3: Print the deposited amount.

Step 5: Define the withdraw method to withdraw an amount from the account.

- Step 5.1: Prompt the user to enter the amount to be withdrawn.
- Step 5.2: Check if self.balance is greater than or equal to the entered amount.
    - Step 5.2.1: If true, subtract the amount from self.balance and print the withdrawn amount.
    - Step 5.2.2: If false, print "Insufficient balance".

Step 6: Define the display method to show the current balance.

- Step 6.1: Print the current balance (self.balance).

Step 7: Create an instance of Bank_Account named s.

Step 8: Call the deposit method on the instance s.

Step 9: Call the withdraw method on the instance s.

Step 10: Call the display method on the instance s.

Step 11: End the Program

SOURCE CODE

```
class Bank_Account:def
    _____init___(self):
        self.balance=0
        print("Welcome to the Deposit & Withdrawal Machine")

    def deposit(s):
        amount=float(input("\n Enter amount to be Deposited: "))s.balance +=
        amount
        print(" Amount Deposited:",amount)

    def withdraw(self):
        amount = float(input("\n Enter amount to be Withdrawn: "))if
        self.balance>=amount:
            self.balance-=amount
            print(" You Withdrew:", amount)else:
            print("\n Insufficient balance  ")

    def display(self):
        print("\n Net Available Balance=",self.balance)
s = Bank_Account()
s.deposit() s.withdraw()
s.display()
```

OUTPUT

```
Welcome to the Deposit & Withdrawal Machine

 Enter amount to be Deposited: 10000
 Amount Deposited: 10000.0

 Enter amount to be Withdrawn: 500
 You Withdrew: 500.0

 Net Available Balance= 9500.0
```

RESULT

Thus the above program has been executed successfully.

PROGRAM USING INHERITANCE

Step 1: Start the Program

Step 2: Define the Employee class.

- Step 2.1: Define the getEmployee method to input employee details.
  - Prompt the user to enter the employee ID, name, and basic salary.
  - Store these values in self.id, self.name, and self.salary, respectively.
- Step 2.2: Define the printEmployeeDetails method to print employee details.
  - Print the employee ID, name, and basic salary.
- Step 2.3: Define the getSalary method to return the basic salary.

Step 3: Define the Perks class, inheriting from Employee.

- Step 3.1: Define the calcPerks method to calculate perks.
  - Call the getEmployee method to input employee details.
  - Get the basic salary using getSalary.
  - Calculate Dearness Allowance (DA) as 35% of the basic salary.
  - Calculate House Rent Allowance (HRA) as 17% of the basic salary.
  - Calculate Provident Fund (PF) as 12% of the basic salary.
- Step 3.2: Define the putPerks method to print the perks details.
  - Call printEmployeeDetails to print employee details.
  - Print the calculated DA, HRA, and PF.
- Step 3.3: Define the totalPerks method to calculate the total perks.
  - Compute the total perks as the sum of DA and HRA minus PF.
  - Return the total perks.

Step 4: Define the NetSalary class, inheriting from Perks.

- Step 4.1: Define the getTotal method to calculate the net salary.
  - Call calcPerks to calculate perks.
  - Calculate the net salary as the sum of the basic salary and total perks.
- Step 4.2: Define the showTotal method to display the total salary.
  - Call putPerks to print employee details and perks.
  - Print the net salary.

Step 5: Create an instance of NetSalary named empSalary.

Step 6: Call the getTotal method on empSalary to calculate the total salary.

Step 7: Call the showTotal method on empSalary to display the total salary.

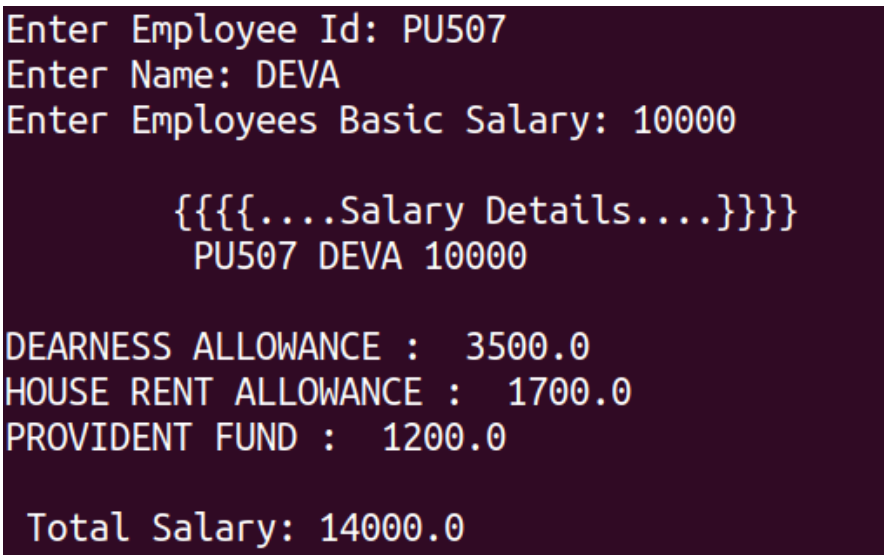
Step 8: End the Program

SOURCE CODE

```
class Employee:
    def getEmployee(self):
        self.id = input("Enter Employee Id: ")
        self.name = str(input("Enter Name: "))
        self.salary = int(input("Enter Employees Basic Salary: "))
    def printEmployeeDetails(self):
        print("\n\t{{{{....Salary Details. ..... }}}}")
        print("\t",self.id,self.name,self.salary)
    def getSalary(self):
        return(self.salary)
class Perks(Employee):def
    calcPerks(self):
        self.getEmployee()
        sal=self.getSalary()
        self.da=sal*35/100 self.hra
        = sal * 17 / 100
        self.pf=sal*12/100
    def putPerks(self): self.printEmployeeDetails()
        print("\nDEARNESS ALLOWANCE : ",self.da) print("HOUSE
        RENT ALLOWANCE : ",self.hra)print("PROVIDENT FUND :
        ",self.pf)
    def totalPerks(self):
        t=self.da+self.hra-self.pf
        return (t)
 class NetSalary(Perks):def
          getTotal(self):
         self.calcPerks()
             self.ns=self.getSalary()+self.totalPerks()
```

```
 def showTotal(self):
        self.putPerks()
        print("\n Total Salary:",self.ns)
empSalary = NetSalary()
empSalary.getTotal()
empSalary.showTotal()
```

OUTPUT:


Enter Employee Id: PU507
Enter Name: DEVA
Enter Employees Basic Salary: 10000
{{{{....Salary Details....}}}}
PU507 DEVA 10000
DEARNESS ALLOWANCE : 3500.0
HOUSE RENT ALLOWANCE : 1700.0
PROVIDENT FUND : 1200.0
Total Salary: 14000.0

RESULT

Thus the above program has been executed successfully.

PROGRAM USING POLYMORPHISM

Step 1: Start the Program

Step 2: Import the pi constant from the math module.

Step 3: Define the Square class.

- Step 3.1: Define the __init__ method to initialize the side length (l).
- Step 3.2: Define the perimeter method to calculate and return the perimeter of the square.
- Step 3.3: Define the area method to calculate and return the area of the square.

Step 4: Define the Circle class.

- Step 4.1: Define the __init__ method to initialize the radius (r).
- Step 4.2: Define the perimeter method to calculate and return the perimeter of the circle.
- Step 4.3: Define the area method to calculate and return the area of the circle.

Step 5: Define the Rectangle class.

- Step 5.1: Define the __init__ method to initialize the width (w) and length (l).
- Step 5.2: Define the perimeter method to calculate and return the perimeter of the rectangle.
- Step 5.3: Define the area method to calculate and return the area of the rectangle.

Step 6: Prompt the user to enter the side length for the square.

- Step 6.1: Create an instance of the Square class with the given side length.

Step 7: Prompt the user to enter the radius for the circle.

- Step 7.1: Create an instance of the Circle class with the given radius.

Step 8: Prompt the user to enter the length and width for the rectangle.

- Step 8.1: Create an instance of the Rectangle class with the given length and width.

Step 9: Calculate and print the perimeter and area of the square.

- Step 9.1: Call the perimeter method on the square instance and print the result.
- Step 9.2: Call the area method on the square instance and print the result.

Step 10: Calculate and print the perimeter and area of the circle.

- Step 10.1: Call the perimeter method on the circle instance and print the result.
- Step 10.2: Call the area method on the circle instance and print the result.

Step 11: Calculate and print the perimeter and area of the rectangle.

- Step 11.1: Call the perimeter method on the rectangle instance and print the result.
- Step 11.2: Call the area method on the rectangle instance and print the result.

Step 12: End the Program

SOURCE CODE:

```
from math import pi

class square:
    def___init___(self, length):
        self.l = length
    def perimeter(self):
        return 4 * (self.l)
    def area(self):
      return self.l * self.l

class Circle:
    def_init_(self, radius):self.r =
         radius
    def perimeter(self): return 2 *
         pi * self.r
    def area(self):
       return pi * self.r**2

class rectangle:
    def___init___(self,width,length):self.l
        = length
        self.w=width def
    perimeter(self):
        return 2 * (self.l+self.w)def
    area(self):
      return self.l * self.w

s=int(input("Enter the length to find Perimeter and Area of Square : "))sqr = square(s)
c=int(input("Enter the Radius to find Perimeter and Area of Circle : "))c1 = Circle(c)
l, w = map(int, input("Enter the length and width to find Perimeter andArea of
                      rectangle : ").split())
```

```
rec = rectangle(l,w)

print("\nPerimeter computed for square: ", sqr.perimeter()) print("Area
computed for square: ", sqr.area()) print("Perimeter computed for Circle:
", c1.perimeter()) print("Area computed for Circle: ", c1.area())
print("Perimeter computed for Rectangle: ", rec.perimeter())print("Area
computed for Rectangle: ", rec.area())
```

OUTPUT

```
Enter the length to find Perimeter and Area of Square : 10
Enter the Radius to find Perimeter and Area of Circle : 8
Enter the length and width to find Perimeter and Area of rectangle : 58 56

Perimeter computed for square:  40
Area computed for square:  100
Perimeter computed for Circle:  50.26548245743669
Area computed for Circle:  201.06192982974676
Perimeter computed for Rectangle:  228
Area computed for Rectangle:  3248
```

RESULT

Thus the above program has been executed successfully.

PROGRAM USING NUMPY

Step 1 : Input Rows and Columns: Prompt the user to input the number of rows and columns for the matrix.

Step 2: Input Matrix Entries: Prompt the user to input the matrix entries in a single line separated by spaces.

Step 3 : Create Matrix: Convert the input entries into a matrix of the specified size using NumPy.

Step 4: Create Arrays: Create different types of arrays using NumPy functions and the matrix.

Step 5 : Display Results: Print out the arrays created along with their content.

Step 6 : End Program.

SOURCE CODE

```
import numpy as np
R = int(input("Enter the number of rows:"))
C = int(input("Enter the number of columns:"))
print("Enter the entries in a single line (separated by space): ")entries =
list(map(int, input().split()))
matrix = np.array(entries).reshape(R, C)a
= np.array(matrix, dtype = 'U')
print ("Array created using passed list:\n", a)b =
np.array((matrix))
print ("\nArray created using passed tuple:\n", b)c =
np.zeros((3, 2))
print ("\nAn array initialized with all zeros:\n", c)d =
np.full((3, 3), 6, dtype = 'complex')
print ("\nAn array initialized with all 6s. Array type is complex:\n", d)e =
np.random.random((2, 2))
print ("\nA random array:\n", e)f
= np.arange(0, 30, 5)
print ("\nA sequential array with steps of 5:\n", f)g =
np.linspace(0, 5, 10)
print ("\nA sequential array with 10 values between 0 and 5:\n", g)
slc=matrix[:,:-1]
print("\nSlicing the array:\n",slc)
srt=np.sort(matrix,axis=None)
print("\nSorting the array:\n",srt)
arr = np.array(matrix)
newarr = arr.reshape(3,2)
print ("\nOriginal array:\n", arr) print
("Reshaped array:\n", newarr)flarr =
a.flatten()
print ("\nOriginal array:\n", a)
print ("Fattened array:\n", flarr)
```

OUTPUT

```
Enter the number of rows:2
Enter the number of columns:3
Enter the entries in a single line (separated by space):
1 4 2 5 3 6
Array created using passed list:
 [['1' '4' '2']
 ['5' '3' '6']]

Array created using passed tuple:
 [[1 4 2]
 [5 3 6]]

An array initialized with all zeros:
 [[0. 0.]
 [0. 0.]
 [0. 0.]]
An array initialized with all 6s. Array type is complex:
 [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]

A random array:
 [[0.68710609 0.38827712]
 [0.20152964 0.8367944 ]]

A sequential array with steps of 5:
 [ 0  5 10 15 20 25]

A sequential array with 10 values between 0 and 5:
 [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
 3.33333333 3.88888889 4.44444444 5.        ]

Slicing the array:
 [[1 4]
 [5 3]]

Sorting the array:
 [1 2 3 4 5 6]

Original array:
 [[1 4 2]
 [5 3 6]]
Reshaped array:
 [[1 4]
 [2 5]
 [3 6]]

Original array:
 [['1' '4' '2']
 ['5' '3' '6']]
Fattened array:
 ['1' '4' '2' '5' '3' '6']
```

RESULT

Thus the above program has been executed successfully

PROGRAM USING PANDAS

Step 1 : Import **pandas** and **matplotlib.pyplot** libraries.

Step 2 : Use **pd.read_csv()** to read data from the CSV file into a DataFrame.

Step 3 : Extract 'total_profit' and 'month_number' columns as lists.

Step 4 : Use **plt.plot()** to create a line plot with specified attributes. Customize labels, legend, title, ticks, and grid using appropriate **plt** functions.

Step 5 : Use **plt.show()** to display the plot window.

SOURCE CODE

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("~/Desktop/python/company_sales_data.csv")
profitList = df ['total_profit'].tolist()
monthList = df ['month_number'].tolist()
plt.plot(monthList, profitList, label = 'Profit data of last year',
color='r', marker='o', markerfacecolor='k', linestyle='--', linewidth=3)
plt.xlabel('Month Number')
plt.ylabel('Profit in dollar')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.grid()
plt.show()
```

OUTPUT

.csv file (company_sales_data.csv):

	A	B	C	D	E	F	G	H	I
1	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	total_profit
2	1	2500	1500	5200	9200	1200	1500	21100	211000
3	2	2630	1200	5100	6100	2100	1200	18330	183300
4	3	2140	1340	4550	9550	3550	1340	22470	224700
5	4	3400	1130	5870	8870	1870	1130	22270	222700
6	5	3600	1740	4560	7760	1560	1740	20960	209600
7	6	2760	1555	4890	7490	1890	1555	20140	201400
8	7	2980	1120	4780	8980	1780	1120	29550	295500
9	8	3700	1400	5860	9960	2860	1400	36140	361400
10	9	3540	1780	6100	8100	2100	1780	23400	234000
11	10	1990	1890	8300	10300	2300	1890	26670	266700
12	11	2340	2100	7300	13300	2400	2100	41280	412800
13	12	2900	1760	7400	14400	1800	1760	30020	300200

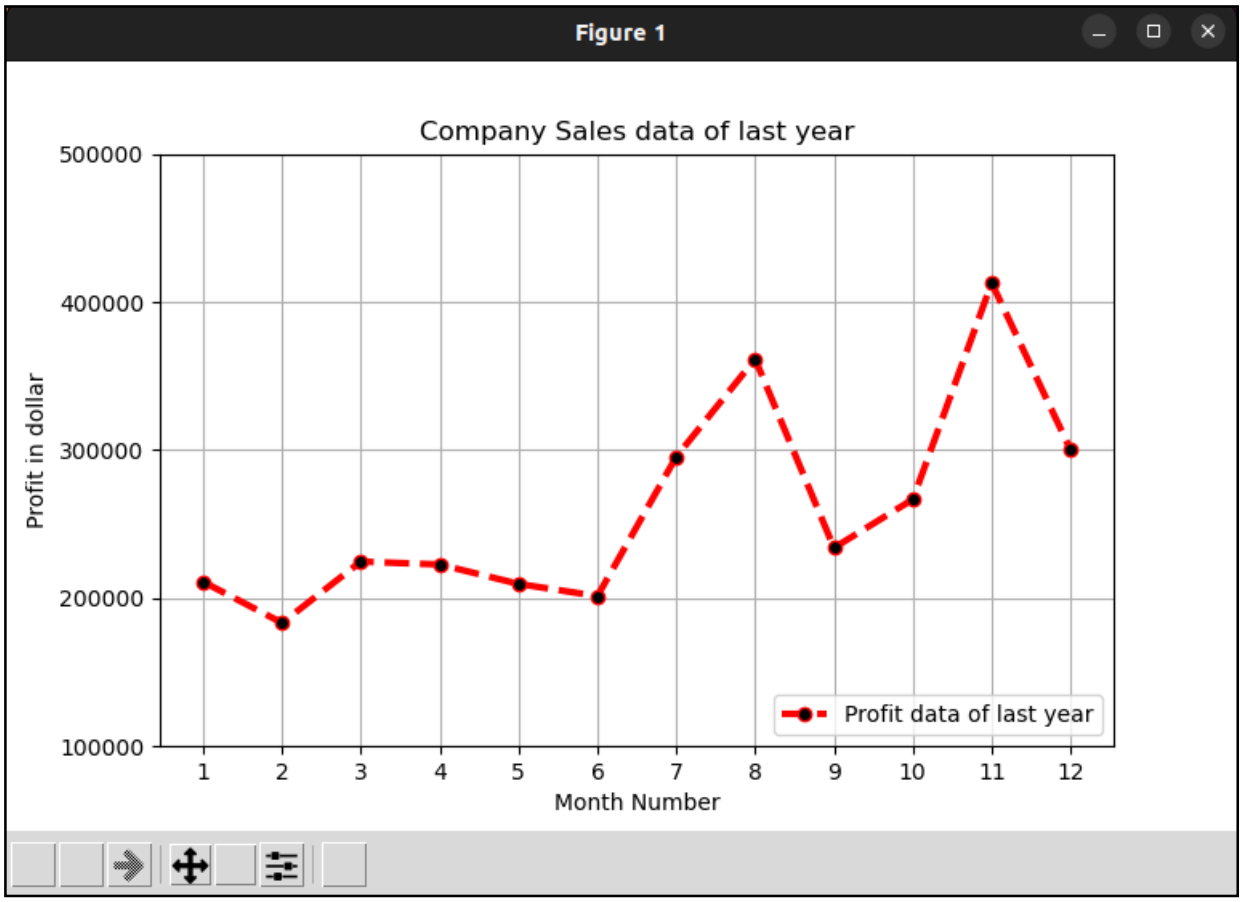


Figure 1
Company Sales data of last year
Profit in dollar
500000
400000
300000
200000
100000
1
2
3
4
5
6
7
8
9
10
11
12
Month Number
Profit data of last year

RESULT

Thus the above program has been executed successfully.

PROGRAM USING MATPLOTLIB

Step 1: Import Required Libraries

- Import the pandas library with the alias pd for data manipulation.
- Import the pyplot module from the matplotlib library with the alias plt for data visualization.

Step 2: Read Data from CSV File

Step 2 : Read CSV File:

- Use the pd.read_csv() function to read the data from the CSV file located at the specified path (~/Desktop/python/company_sales_data.csv).
- Store the data in a DataFrame variable named df.

Step 3: Prepare Data for Plotting

- Extract the 'total_profit' and 'month_number' columns from the DataFrame df.
- Convert these columns to lists named profitList and monthList, respectively, using the tolist() method.

Step 4: Create and Customize the Plot

Step 5 : Plot the Data:

- Use plt.plot() to create a line plot with the 'month_number' on the x-axis and 'total_profit' on the y-axis.
- Customize the plot appearance:
  - Set the label, color, marker style, marker color, linestyle, and linewidth.
- Add x-label, y-label, legend, and title to the plot.
- Customize tick marks on both axes using plt.xticks() and plt.yticks().
- Add a grid to the plot using plt.grid().

Step 6: Display the Plot

Step 7 : Show the Plot:

- Use plt.show() to display the plot window with the customized plot.

Step 8 : End of Algorithm

SOURCE CODE

```
from matplotlib import pyplot as pltimport
numpy as np
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.plot(x, y)
plt.title("Matplotlib Multiple Points")
plt.legend(["X and Y Multiple Points"])
plt.show()
plt.plot(x, y) plt.title("Matplotlib
Grid")
plt.legend(["X and Y Multiple Points"])plt.grid()
plt.show()
plt.plot(y, linewidth = '20.5',color = 'r')
plt.title("Matplotlib Line Width") plt.legend(["
Line Width"])
plt.show()
plt.title("Matplotlib Bar Chart")plt.bar(x,
y,color = "hotpink") plt.legend(["Gold
Rate"]) plt.show()
plt.title("Matplotlib Histogram")z=
np.random.normal(y) plt.hist(z)
plt.show()
plt.title("Matplotlib Pie Chart")w=
np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]plt.pie(w,
labels = mylabels)
plt.legend(loc='lower right')
plt.show()
plt.scatter(x, y) plt.title("Matplotlib Scatter
Plot") plt.xlabel("Time (hr)")
plt.ylabel("Position (Km)") plt.legend(['X
and Y Scatter Polt'])plt.show()
```

```
plt.title("Matplotlib Linespace")y=
np.linspace(0, 5, 100) plt.plot(y,
np.cos(y))
plt.plot(y, np.sin(y))
plt.legend(['Cos','Sine'])
plt.show()
```

OUTPUT:

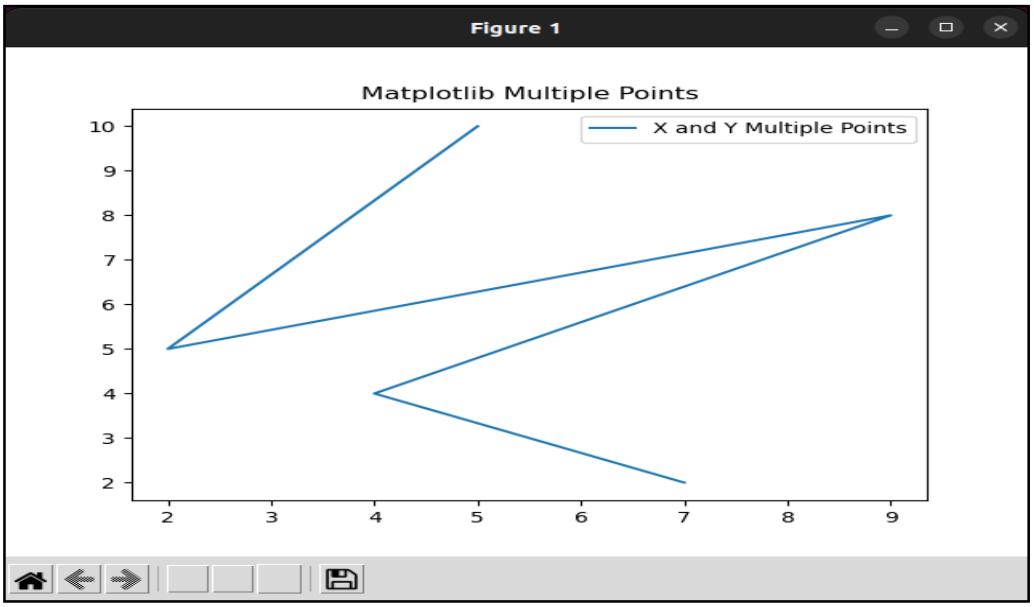


Figure 1
Matplotlib Multiple Points
X and Y Multiple Points
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

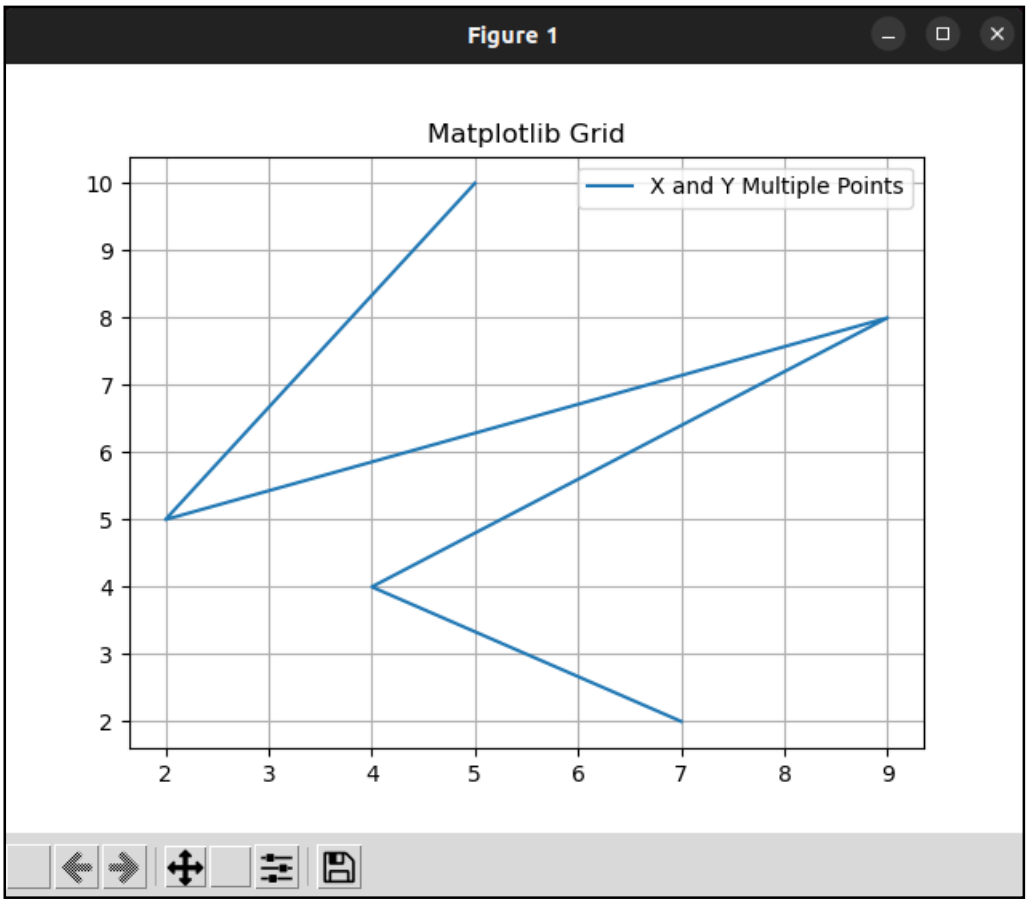


Figure 1
Matplotlib Grid
X and Y Multiple Points
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

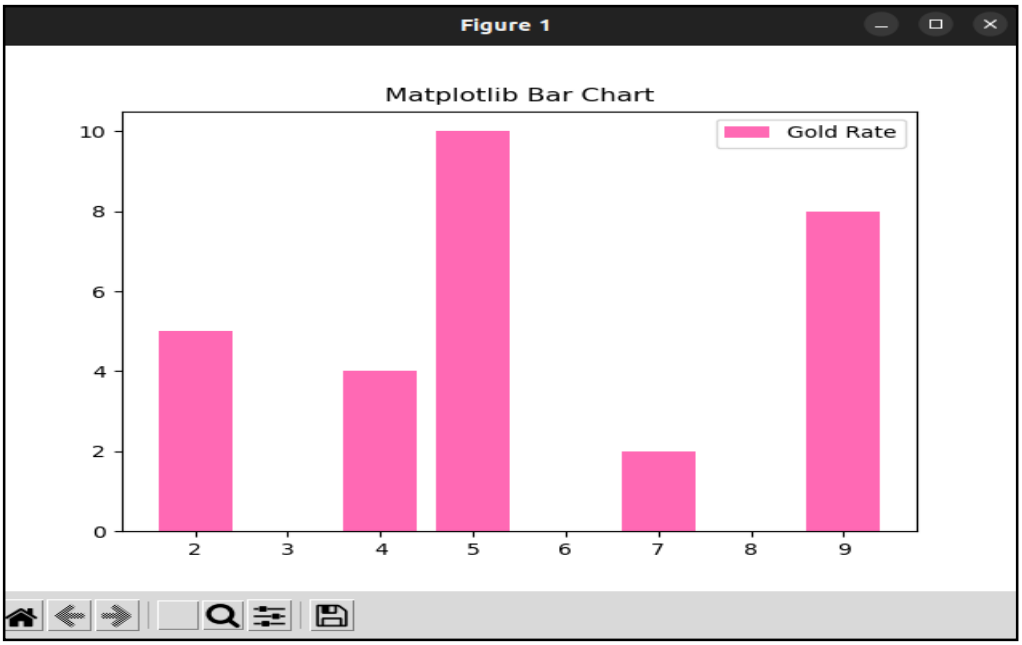


Figure 1
Matplotlib Bar Chart
Gold Rate
10
8
6
4
2
0
2
3
4
5
6
7
8
9

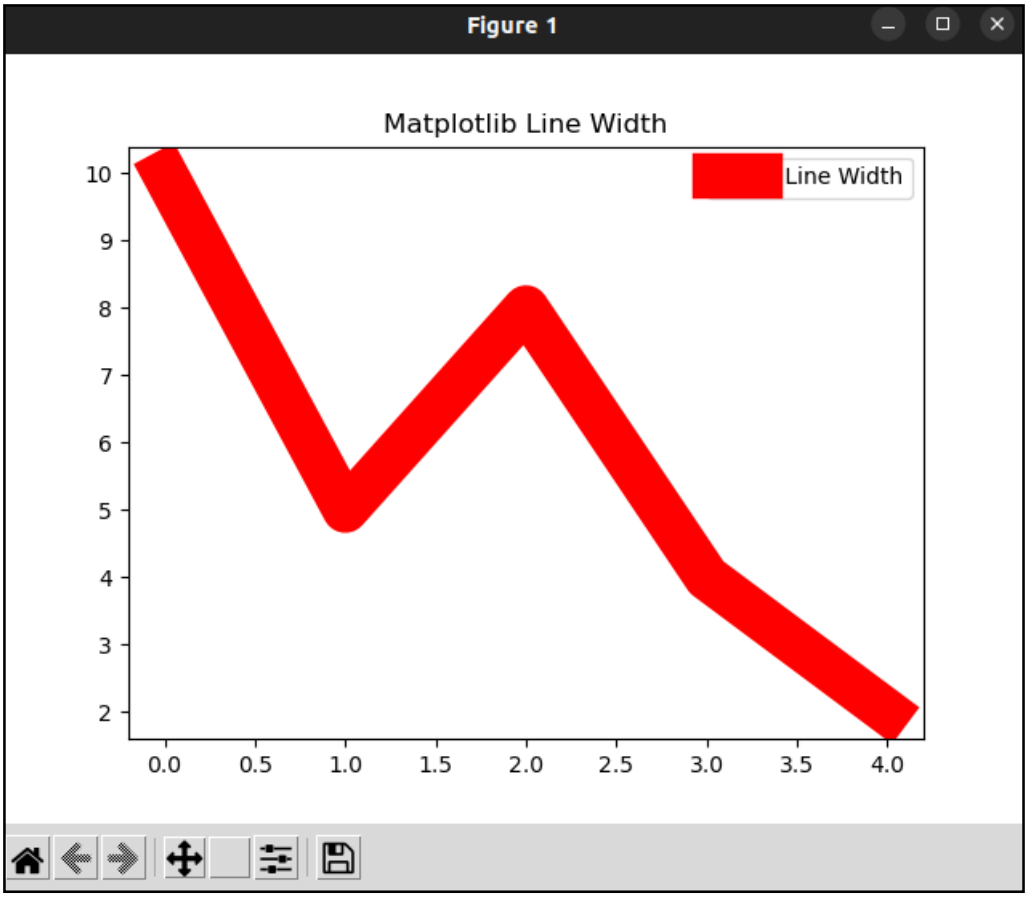


Figure 1
Matplotlib Line Width
Line Width
10
9
8
7
6
5
4
3
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

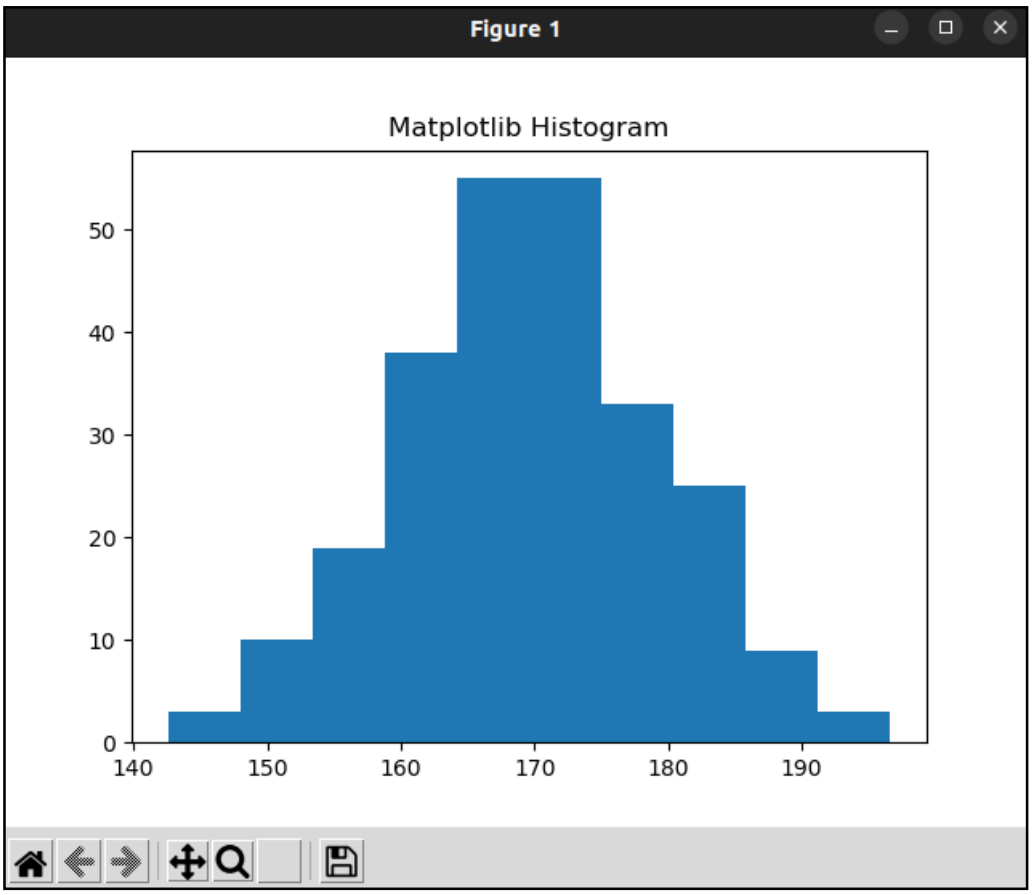


Figure 1
Matplotlib Histogram
50
40
30
20
10
0
140
150
160
170
180
190

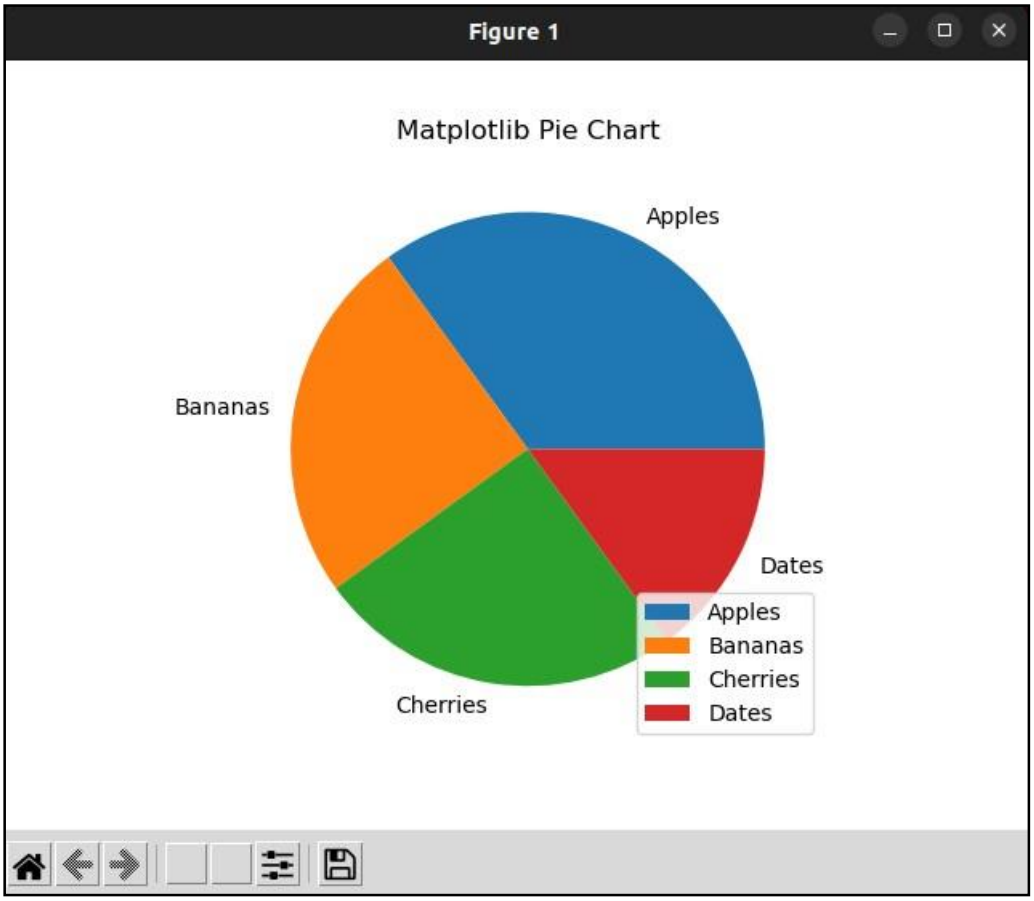


Figure 1
Matplotlib Pie Chart
Apples
Bananas
Dates
Cherries
Apples
Bananas
Cherries
Dates

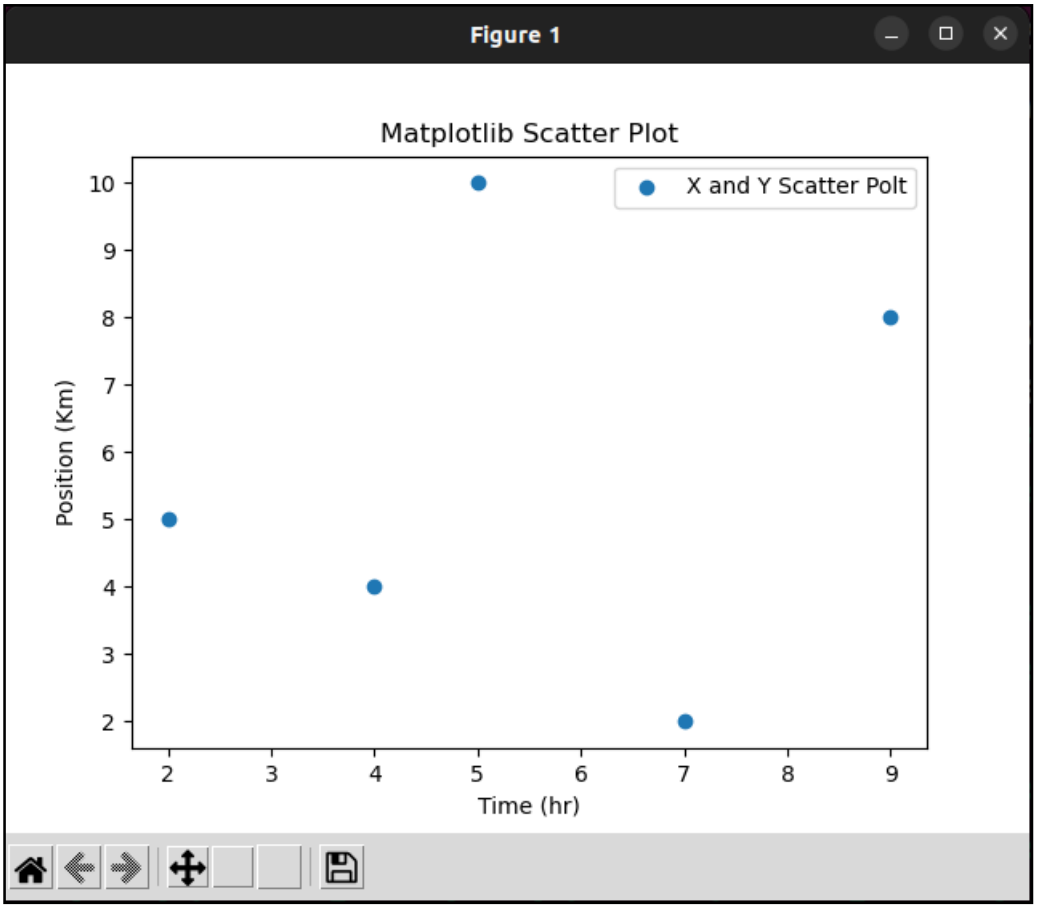


Figure 1
Matplotlib Scatter Plot
X and Y Scatter Polt
Position (Km)
Time (hr)
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

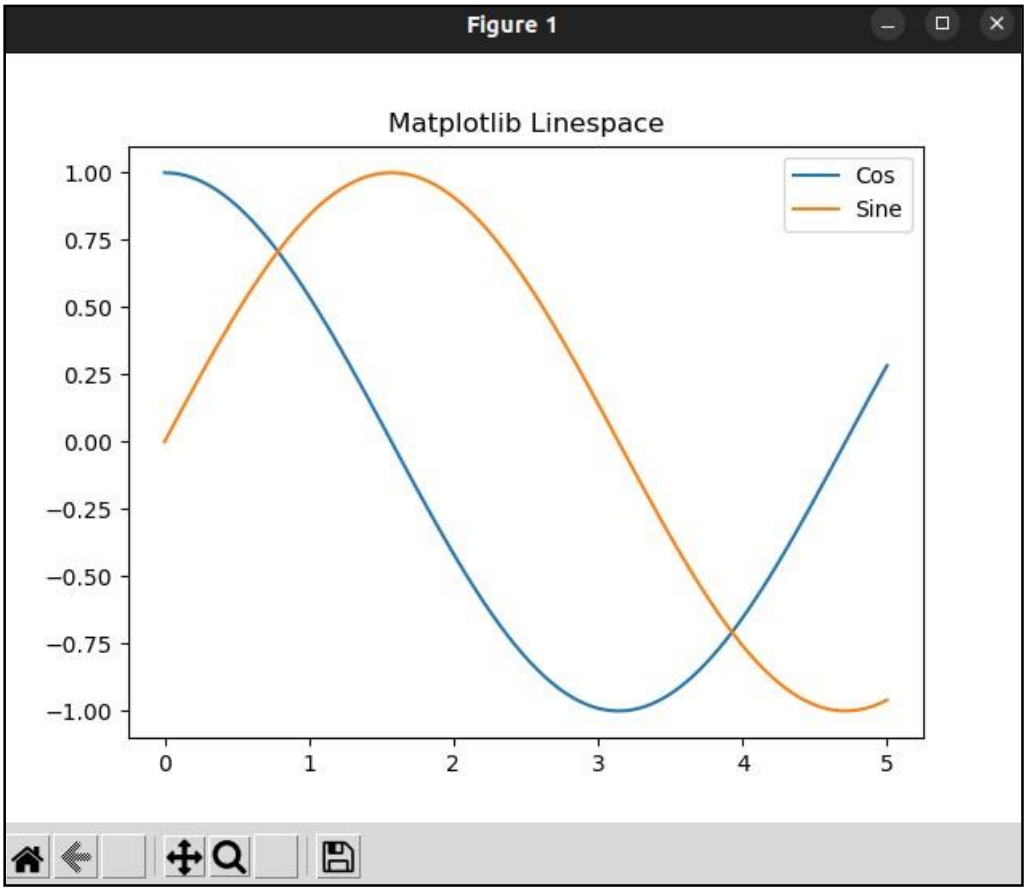


Figure 1
Matplotlib Linespace
Cos
Sine
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5

CREATING DYNAMIC AND INTERACTIVE WEB PAGES USING FORMS

Step 1 : Import Required Modules

1. Import Django Modules:
    - Import necessary modules from Django such as render, HttpResponse, loader, forms, and InputForm from .forms.

Step 2: Define Views

2. Define Home View:
    - Define a function home_view(request) that renders a form to the user.
    - Inside the function:
        - Create an empty dictionary context.
        - Add the form instance InputForm() to the context dictionary.
        - Render the forms.html template with the provided context.

Step 3 : Define Data Store View:

- Define a function data_store(request) to handle form submission and data storage.
- Check if the request method is POST:
    - If POST, initialize the form with the data from the request.
    - If the form is valid, save the form data to the database.
    - Render the forms.html template.
- If the request method is not POST, render the form without data.

Step 4: Define Models (Not Provided)

Define Models:

- Models are not provided in the snippet, but the InputForm likely corresponds to a model form generated from a Django model named sample_model in formsapp.models.

Step 5: Define URL Patterns

Define URL Patterns in urls.py:

- Define URL patterns in urls.py to map views to URLs.
- The URL pattern '' (empty string) is mapped to the home_view function.

Step 6: Create HTML Template

Create HTML Template (forms.html):

- Define an HTML form with required input fields.
- Use Django template language to include CSRF token and render form fields.
- Add styling using inline CSS or CSS in <style> tags.

Step 7: End of Algorithm

SOURCE CODE

A) forms.py

```
from ssl import Options
from click import Option, optionfrom
django import forms
from .models import sample_model
from phonenumber_field.modelfields import PhoneNumberField

class InputForm(forms.ModelForm):class
    Meta:
        model = sample_model
        fields = ["first_name", "last_name",'mobile','email',] labels =
        {"first_name":"First Name:", "last_name":"Last
        Name:",'mobile':"Mobile No:",'email':"Mail Id:",}
```

B) models.py

```
from django.db import models
from phonenumber_field.modelfields import PhoneNumberField

class sample_model(models.Model):
        first_name = models.CharField(max_length=20)last_name
        = models.CharField(max_length=20)mobile =
        PhoneNumberField(max_length=10) email =
        models.EmailField(max_length=20)
```

C) views.py

```
from django.shortcuts import render from
django.http import HttpResponsefrom
django.template import loader
from formsapp.models import sample_modelfrom
django import forms
from django.shortcuts import renderfrom
.forms import InputForm
def home_view(request):
        context ={}
        context['form']= InputForm()
        return render(request, "formsapp/forms.html", context)def
data_store(request):
        if request.method == "POST":
                form = InputForm(request.POST )if
                form.is_valid():
                        forms=form.save()
                        return render(request,'formsapp/forms.html')
                else:
                        form = InputForm()
                return render(request, 'formsapp/forms.html', {'form':form})
```

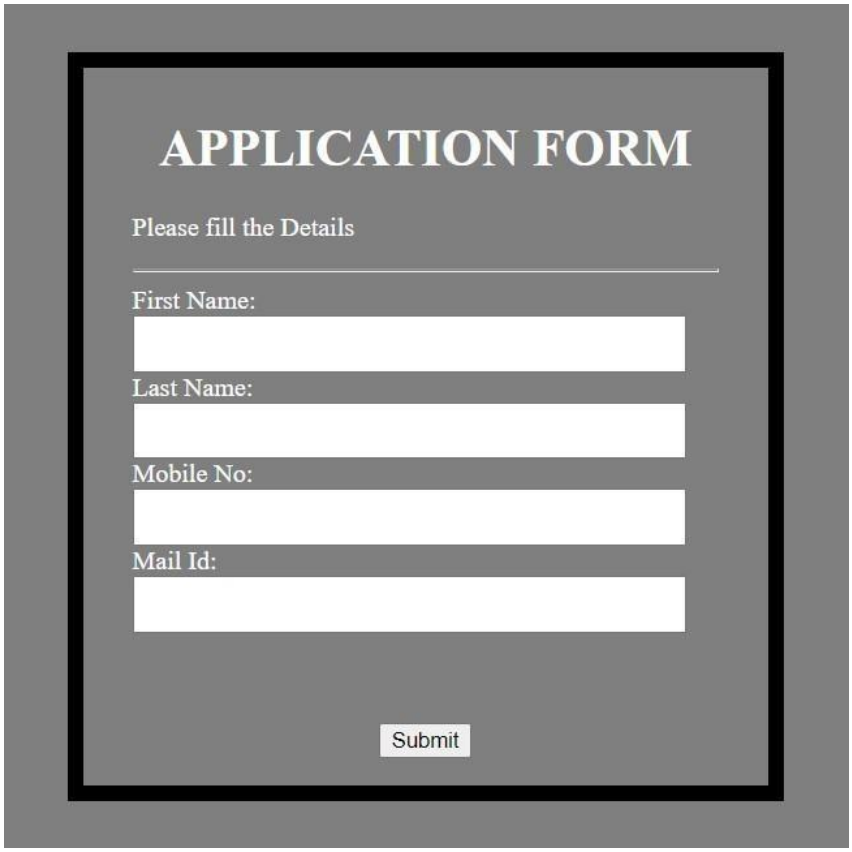
D) urls.py

```
from django.contrib import adminfrom
django.urls import path from formsapp
import views
urlpatterns = [
                path("", views.home_view, name='data_store'),
]
```

E) forms.html

```
<html>
<head>
<style>
body {
        margin: 130px auto;
        font-family: 'Times New Roman';border:
        10px solid rgb(0, 0, 0); width: 360px;
        height: 380px;
        padding: 30px;
        background-color: rgba(0,0,0,.5);
        color:white; }
input {
        width:      340px;
        font-size:12pt;
        height:35px;}
select {
        width: 350px;
        height: 40px; font-
        size: 12pt;}
</style>
</head>
<body>
<form  id="specform"          method="POST">
{% csrf_token %}
<center><h1>APPLICATION FORM</h1></center>
<p>Please fill the Deatils</p><hr>
{{ form }} <br></br><div class="container">
<br></br>
<center><button type="submit" class="btn btn-
primary">Submit</button></center>
</div>
</form>
</body>
</html>
```

OUTPUT:


APPLICATION FORM
Please fill the Details
First Name:
Last Name:
Mobile No:
Mail Id:
Submit

RESULT

Thus the above program has been executed successfully.