# PYTHON PROGRAMMING LAB

**(Course Code: 22UPCSC1C05)**

**A laboratory record submitted to Periyar University, Salem**

**In partial fulfillment of the requirements for the degree of**

**MASTER OF COMPUTER APPLICATIONS**

**By**

**ELANCHEZHIAN M**

**[Reg. No: U22PG507CAP006]**


**DEPARTMENT OF COMPUTER SCIENCE**

**PERIYAR UNIVERSITY**

**(NACC `A++` Grade with CGPA 3.61) – NIRF RANK 59 – ARIIA RANK 10**

**PERIYAR PALKALAI NAGAR,**

**SALEM – 636 011.**

**(NOVEMBER - 2022)**

# CERTIFICATE

This is to certify that the Programming Laboratory entitled "**PYTHON PROGRAMMING LAB (22UPCSC1C05)**" is a bonafide record work done by Mr. / Ms._____
Register No: _____ in partial fulfillment of the requirements for the degree of Master of Computer Applications, in the Department of Computer Science, Periyar University, Salem, during the Academic Year 2022-2023.

Staff In-charge Head of the Department

Submitted for the practical examination held on……

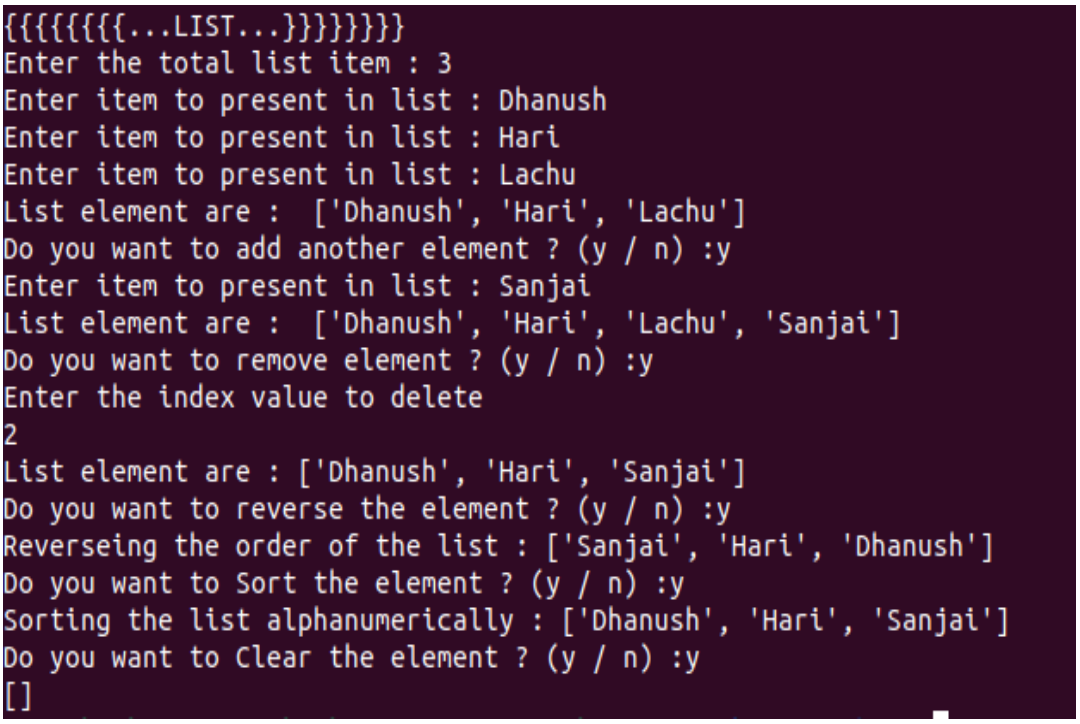
Internal Examiner External Examiner

# CONTENTS

**SOURCE CODE:**

```
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
[]

**SOURCE CODE:**

```
    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
```

**SOURCE CODE:**

```
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'}
```

**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")
```

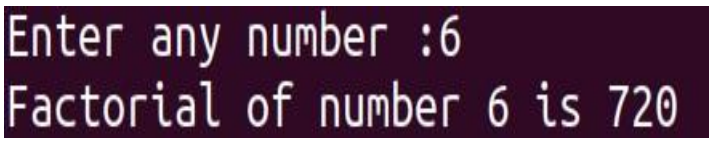
**OUTPUT:**

```
{{{{...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
```

**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:**



**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
```

**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
```

**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
and Area 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
```

**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']
```

**SOURCE CODE:**

```
from matplotlib import pyplot as plt
import 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:

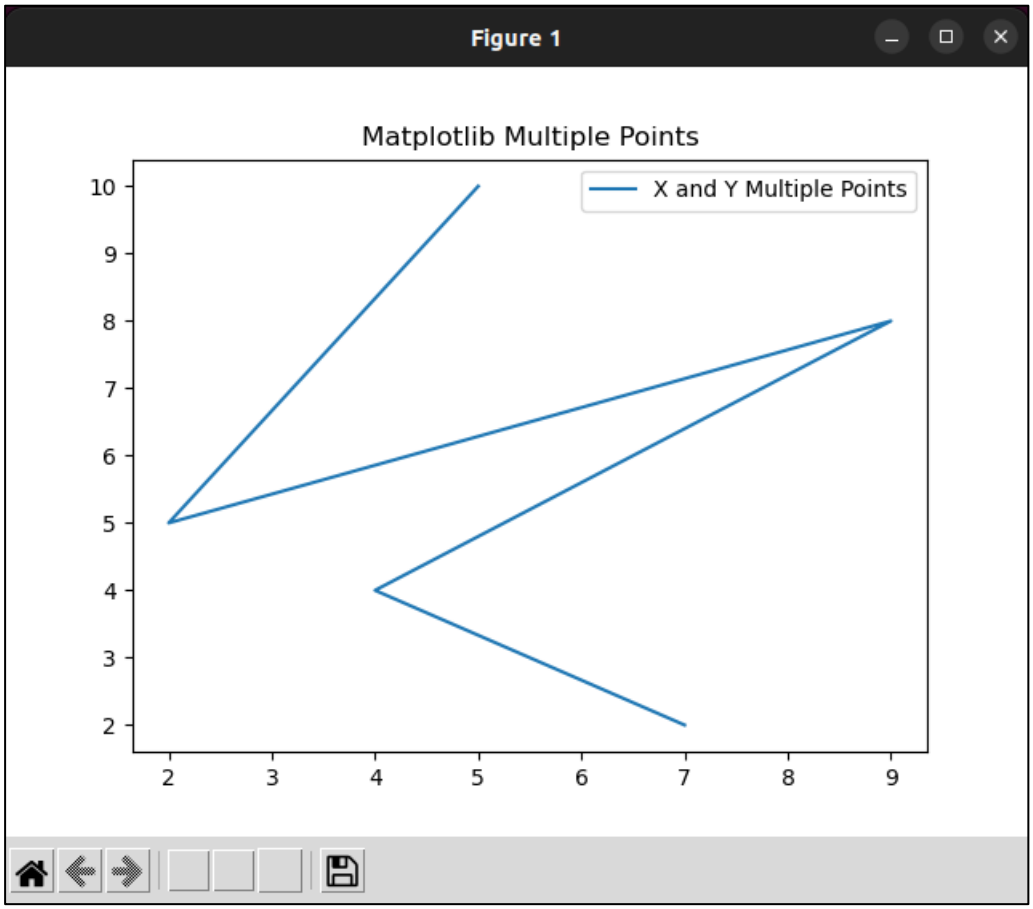


Figure 1
Matplotlib Multiple Points
X and Y Multiple Points
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

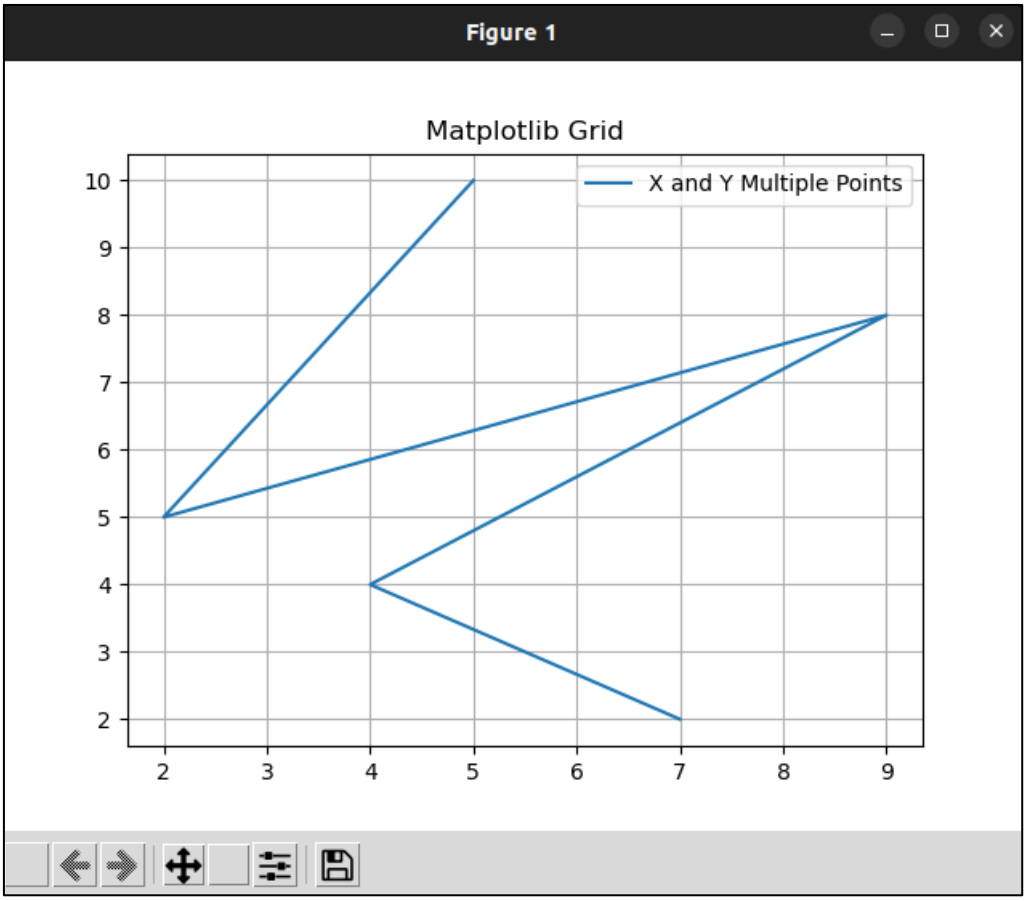


Figure 1
Matplotlib Grid
X and Y Multiple Points
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

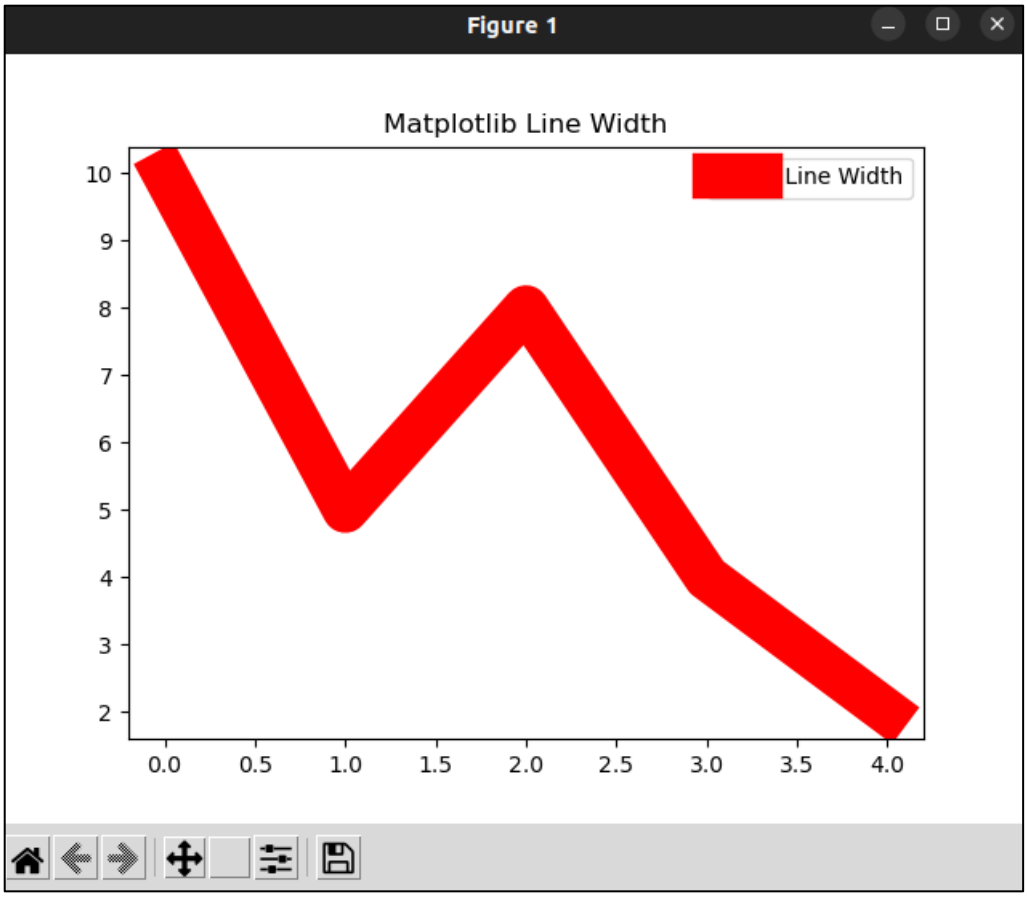


Figure 1
Matplotlib Line Width
Line Width
10
9
8
7
6
5
4
3
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

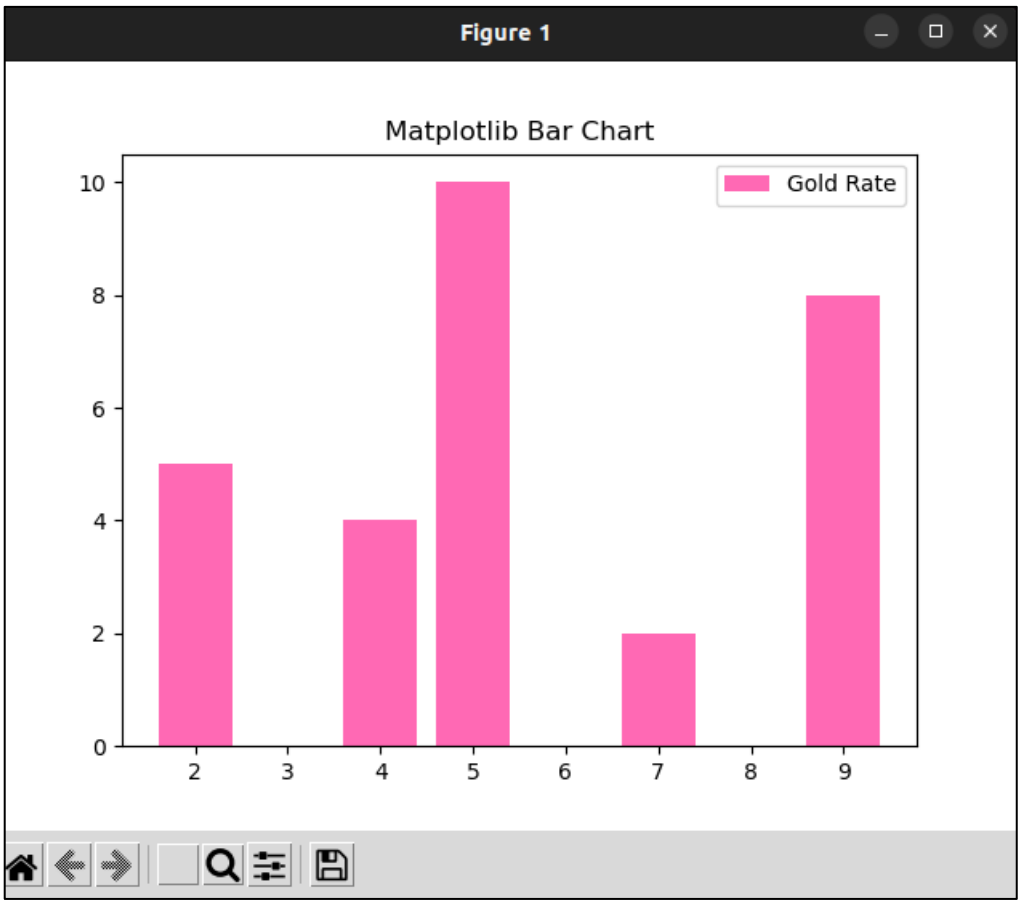


Figure 1
Matplotlib Bar Chart
Gold Rate
10
8
6
4
2
0
2
3
4
5
6
7
8
9

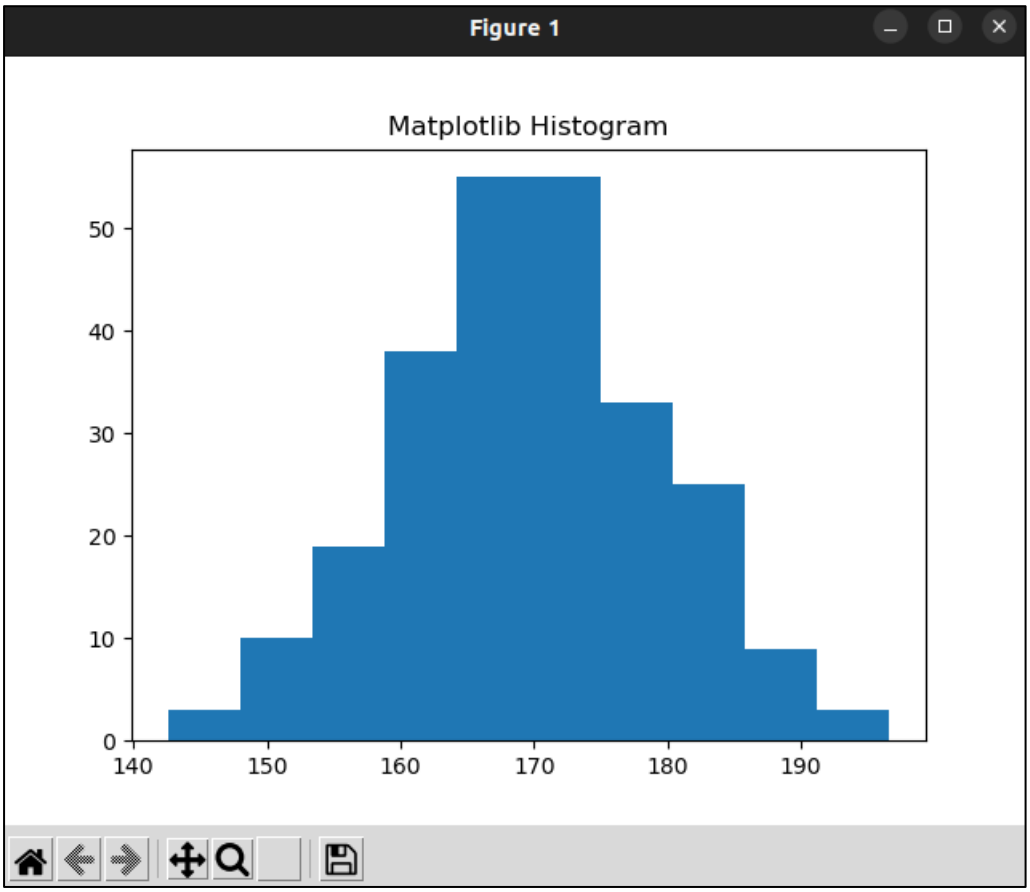


Figure 1
Matplotlib Histogram
50
40
30
20
10
0
140
150
160
170
180
190

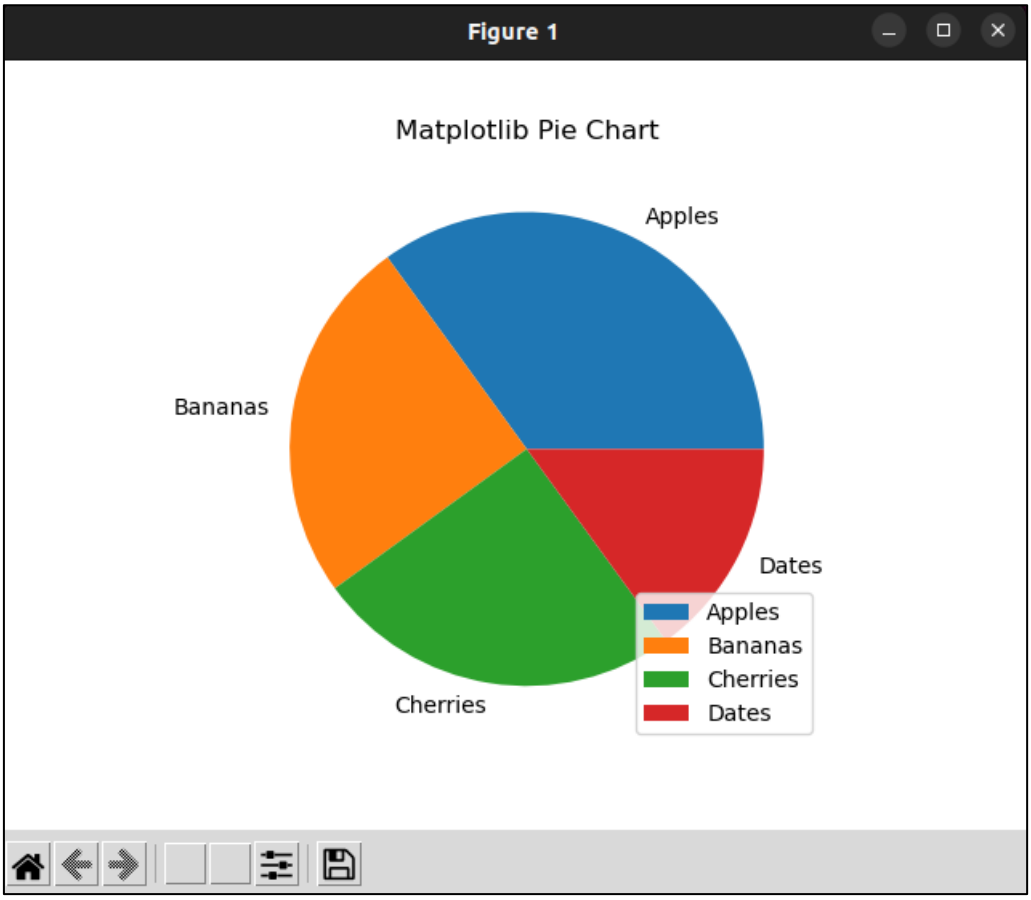


Figure 1
Matplotlib Pie Chart
Apples
Bananas
Cherries
Dates
Apples
Bananas
Cherries
Dates

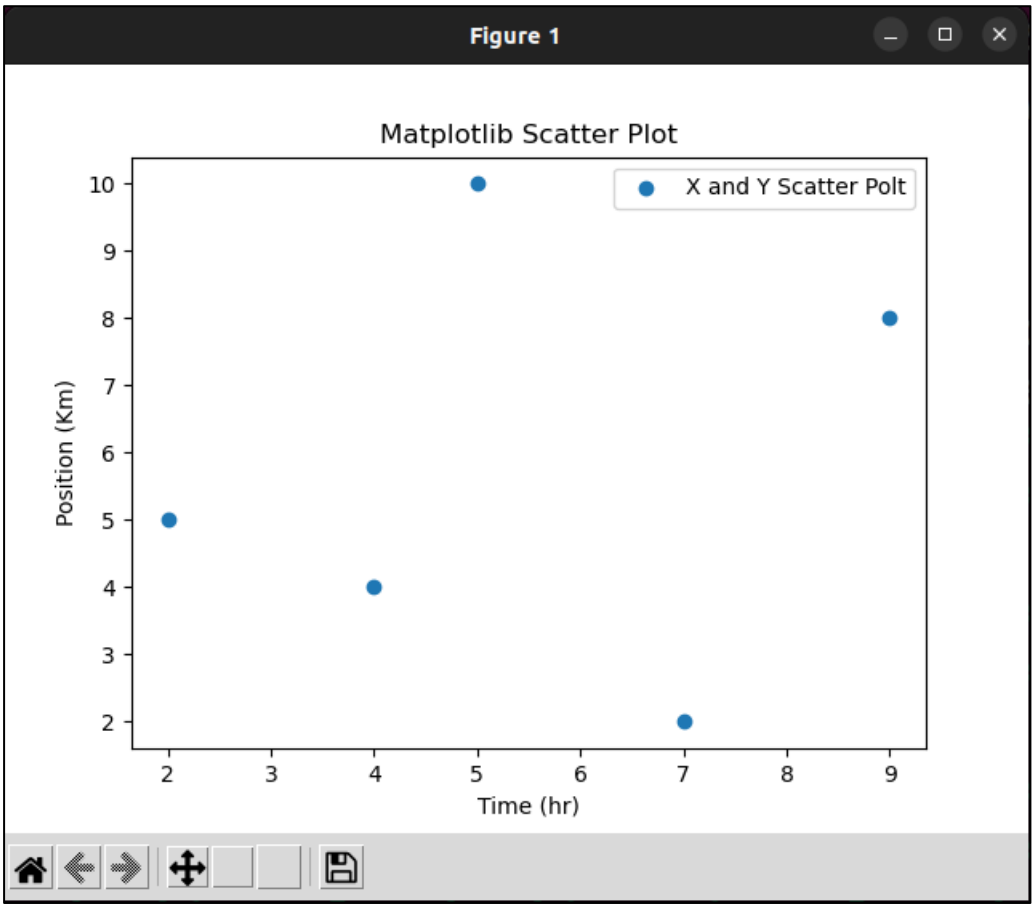


Figure 1
Matplotlib Scatter Plot
X and Y Scatter Polt
Position (Km)
Time (hr)
10
9
8
7
6
5
4
3
2
2
3
4
5
6
7
8
9

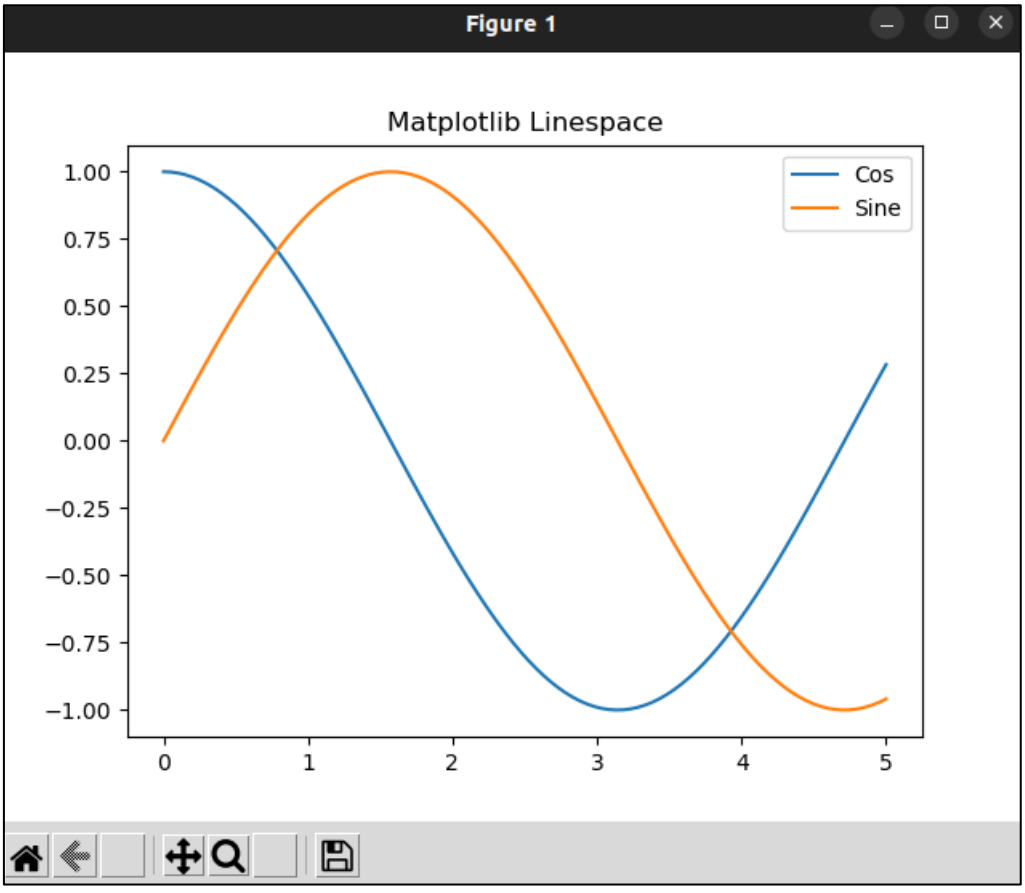


Figure 1
Matplotlib Linespace
Cos
Sine
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5

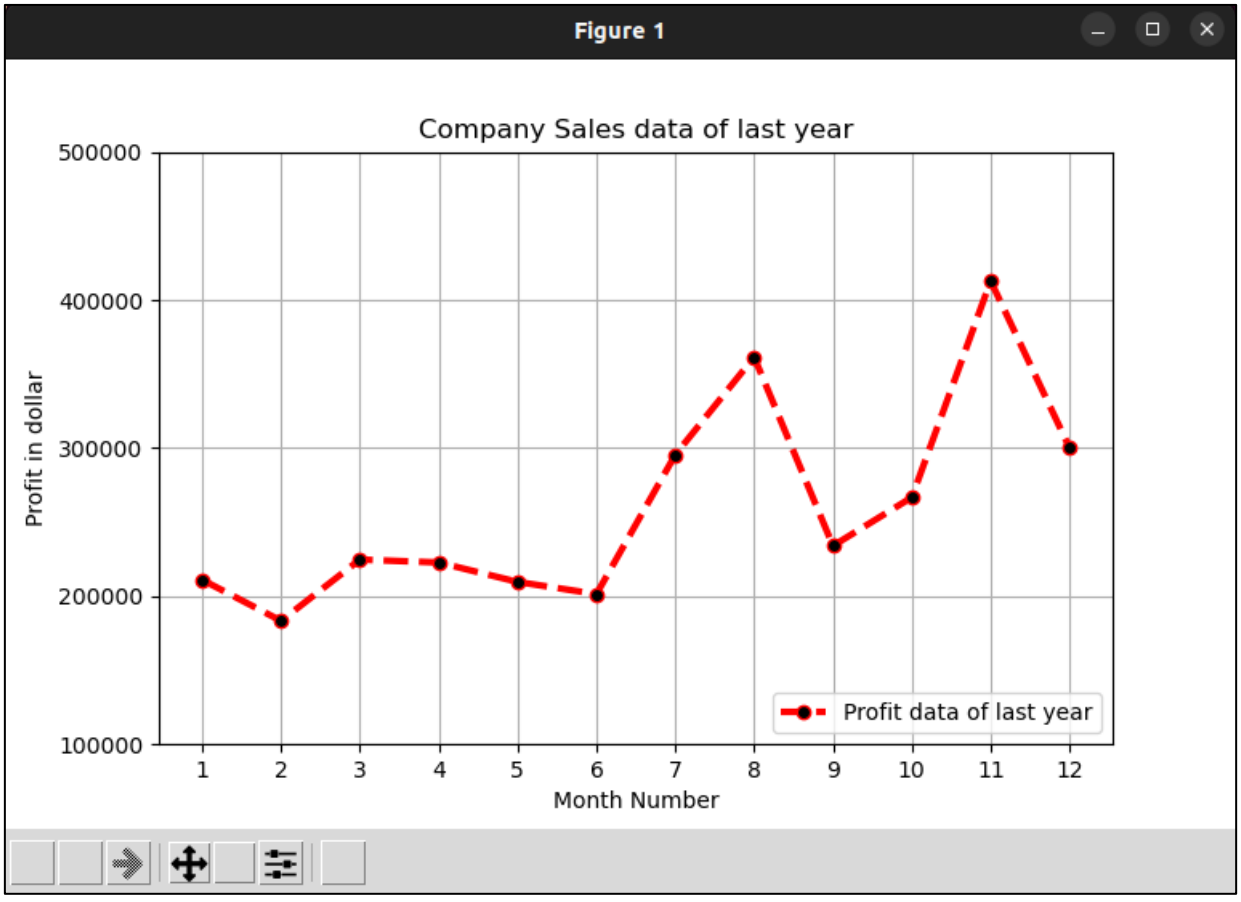
**SOURCE CODE:**

```
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()
```

## .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 |

## OUTPUT:



**SOURCE CODE:**

A) forms.py

```
from ssl import Options
from click import Option, option
from 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 .forms import InputForm

def home_view(request):
        context ={}
        context['form']= InputForm()
        return render(request, "formsapp/forms.html", context)
```

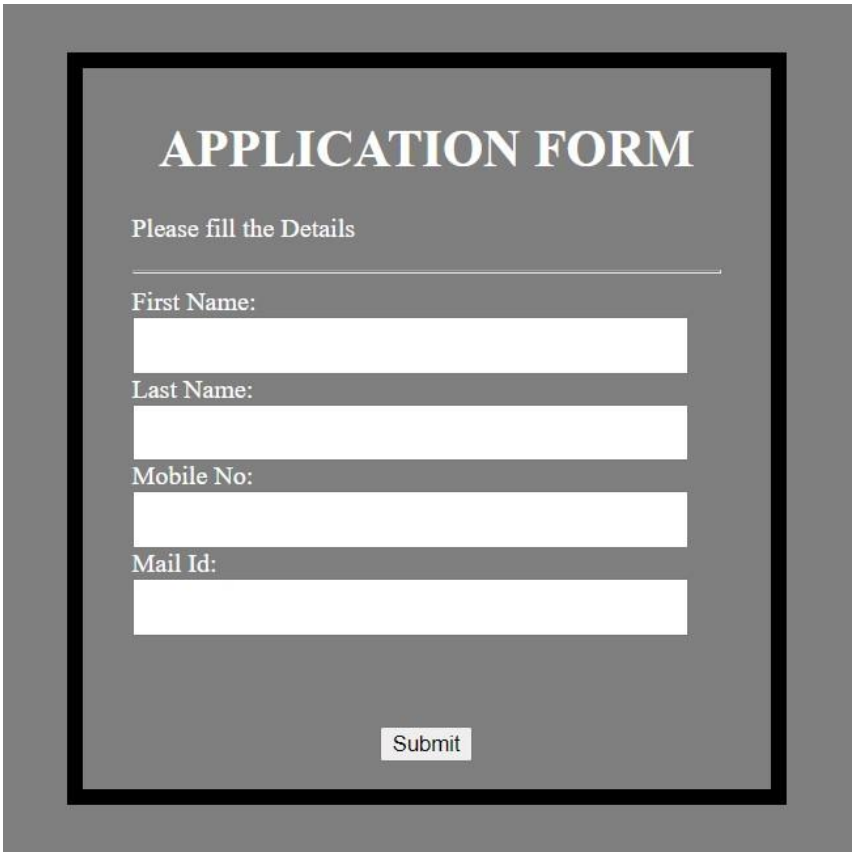
D)urls.py

```
from django.contrib import admin
from 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