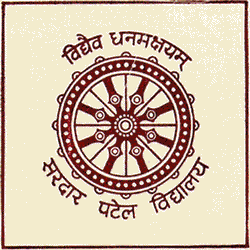
**Project Report**

**on**

**Graphs With Matplotlib**



**Submitted By**

Vinayak Bector

Hari R. Kartha

Class: XII D

**Under the Guidance of**

Ms. Angel Panesar

Department of Computer Science

Sardar Patel Vidyalaya

Lodhi Estate, New Delhi 110003

**CERTIFICATE**

This is to certify that Hari R. Kartha and Vinayak Bector Of Class XII D have prepared the report on the Project entitled “Graphs With Matplotlib”. The report is the result of their efforts & endeavors. The report is found worthy of acceptance as final project report for the subject Computer Science of Class XII. They have prepared the report under my guidance.

Ms. Angel Panesar

Department of Computer Science

Sardar Patel Vidyalaya

Lodhi Estate, New Delhi 110003

**DECLARATION**

We hereby declare that the project work entitled “Graphs With Matplotlib”, submitted to Department of Computer Science, Sardar Patel Vidyalaya, Lodhi Estate, New Delhi 110003 is prepared by us. The project work is result of our personal efforts.

Hari R. Kartha

Vinayak Bector

Class: XII D

**ACKNOWLEDGEMENT**

I would like to express my gratitude towards my teacher, Ms. Angel Panesar who guided me and helped in clearing all my queries and fixing errors in my code for the interactive card game Poker. The final outcome and completion of my project required a lot of guidance where she provided the necessary assistance for me to complete my project.

Vinayak Bector

Hari R. Kartha

Class: XII D

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| S. No. | Topic | Page No. |
| 1. | Libraries used | 5 |
| 2. | Working Description | 7 |
| 3. | Project Code | 11 |
| 4. | Output Screens | 28 |
| 5. | Conclusion | 34 |
| 6. | Bibliography | 35 |

**LIBRARIES USED**

**Python Standard Library**

**Matplotlib**

It is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

**Python Numpy**

Numpyis a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.  
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data

**Tkinter - Standard GUI Library**

Tkinter is the python's standard GUI (Graphical User Interface) package. It is the most commonly used GUI toolkit. This module provides a number of functions that you can use to display appropriate messages and text boxes, and create buttons. It provides a convenient, volatile and user friendly method to input data into a program, which can later be for various purposes. Once the main window is created, any number of widgets can be inserted into it.

.

**MySQL Connector**

It is a library used to connect to a MySQL database from a python script. This is useful in cases which require storage of large amounts of data in a program. By establishing a connection and creating connection objects and cursor objects, one can traverse data present in a database and fetch the required data to be used in the program. it provides an efficient method of data manipulation.

**WORKING DESCRIPTION**

The program begins by opening the main window, where the user can either register as a new user or log in. Buttons created using Tkinter are used for this purpose. When a new user registers, they are taken to a new window where they are asked to input data into text boxes, which is later fetched. a file is created for each new user using the os module, which saves their password and username, After this, all of the fetched data is stored in a database, for which the connection is established is using MySQL connector library. If a user is logging in, their password is first confirmed using the file, after which a log is made onto a table in the database recording the username, date, time and login status.

After logging in, they are taken to a new window where they are given 3 options to choose from- lines, conics or trigonometry. buttons for these were created using Tkinter, using the Button() function and upon clicking them are taken to the respective windows.

the windows for straight lines and conics contain text boxes, where the user inputs the values of the constants in the mathematical equations of the curve into text boxes made using Tkinter and the values were fetched using get()function, assigned variables. (Separate functions were made for plotting for making the code more compact). The windows finally contain two buttons- calculate and back to homescreen. upon clicking on calculate, the variables are plotted using pyplot and a new window displays the graph of the given equation. On clicking back to homescreen, it deletes the current window by using the destroy() function and takes you back to the original window.

For TrigoFunctions, buttons were created for all the trigonometric functions. Upon clicking any of them, the graph for the function is plotted using pyplot directly and requires no further input.

\*In both the cases multiple graphs can be plotted in the same window.

We used mysql.connector and MySQL to store information in a database. a connector object was created, and two tables were created in the database. One table was used to store the data regarding the registration of users, i.e, their username, password, date of birth, gender and age

the other table is used to store the data about the users logging in. After the user enters the username and password into the text boxes and they are confirmed to be registered users, it makes a record on the table LOGIN\_INFO. here their username, the exact date and time when they logged in and their login status is recorded. to store the date and time, the datetime module was used to get an accurate value.

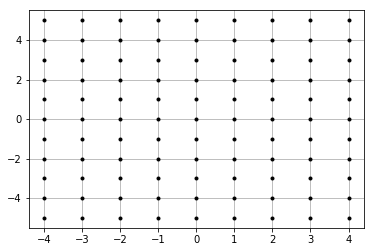
all of the inputs entered into the text boxes are fetched using get() and stored in variables. the variables are stored into the table using a cursor object when the register button is clicked.

We have used many modules in our code, a brief description of each is provided as follows :

**NumpyMeshgrid function**

The numpy.meshgrid function is used to create a rectangular grid out of two given one-dimensional arrays representing the Cartesian indexing or Matrix indexing. Meshgrid function is somewhat inspired from MATLAB.

Consider the above figure with X-axis ranging from -4 to 4 and Y-axis ranging from -5 to 5. So there are a total of (9 \* 11) = 99 points marked in the figure each with a X-coordinate and a Y-coordinate. For any line parallel to the X-axis, the X-coordinates of the marked points respectively are -4, -3, -2, -1, 0, 1, 2, 3, 4. On the other hand, for any line parallel to the Y-axis, the Y-coordinates of the marked points from bottom to top are -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5. The numpy.meshgrid function returns two 2-Dimensional arrays representing the X and Y coordinates of all the points.



**NumpyLinspace**

The NumPylinspace function (sometimes called np.linspace) is a tool in Python for creating numeric sequences.

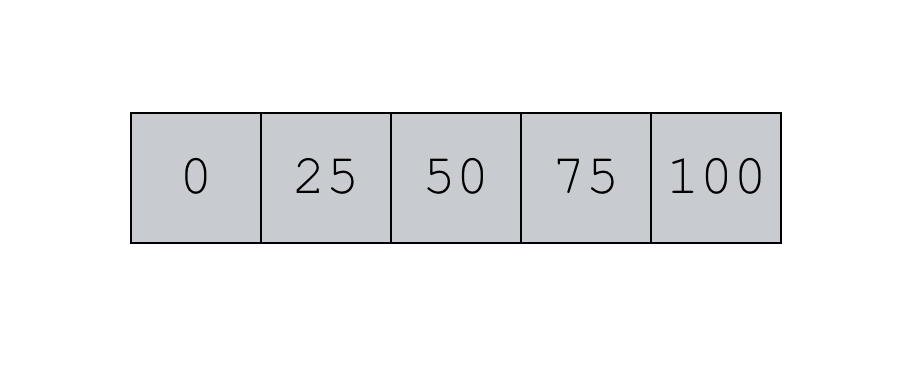
The NumPylinspace function creates sequences of evenly spaced values within a defined interval.

Essentially, you specify a starting point and an ending point of an interval, and then specify the total number of breakpoints you want within that interval (*including* the start and end points). The np.linspace function will return a sequence of evenly spaced values on that interval.

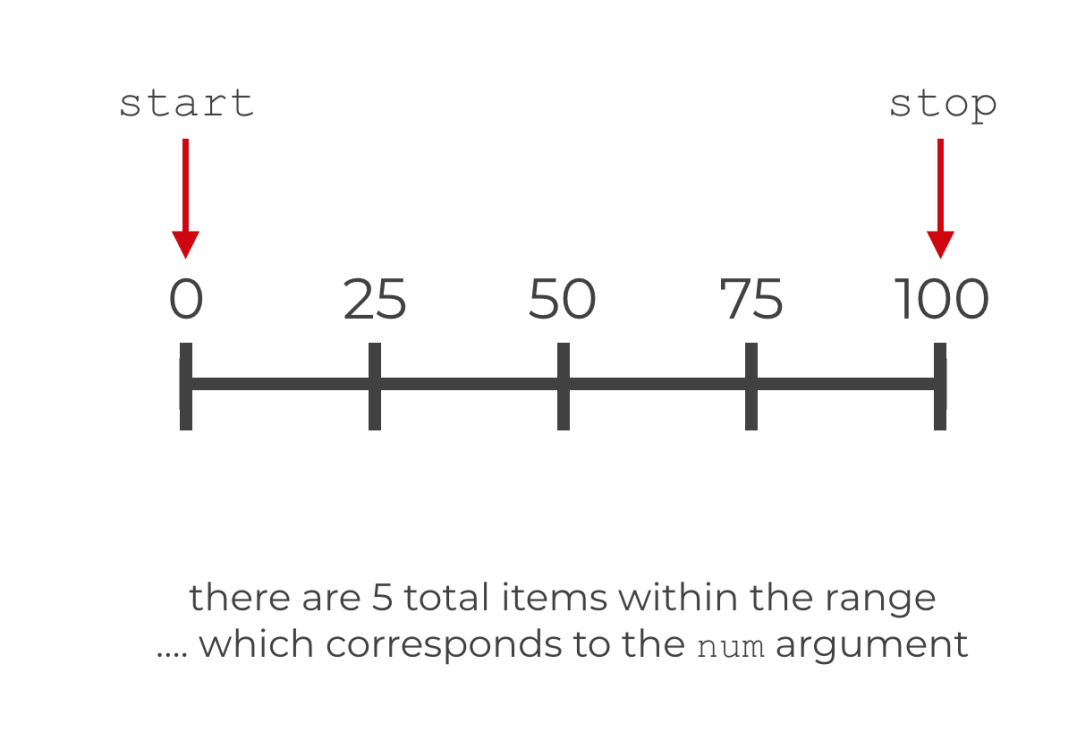
To illustrate this, here’s a quick example. (We’ll look at more examples later, but this is a quick one just to show you what np.linspace does.)

np.linspace(start =0, stop =100,num=5)

This code [produces a NumPy array](https://www.sharpsightlabs.com/blog/numpy-array-python/) that looks like the following:



That’s the ndarray that the code produces, but we can also visualize the output like this:



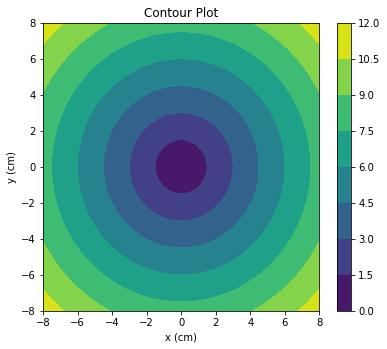
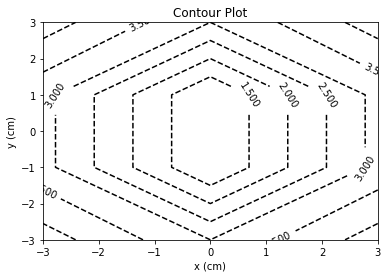
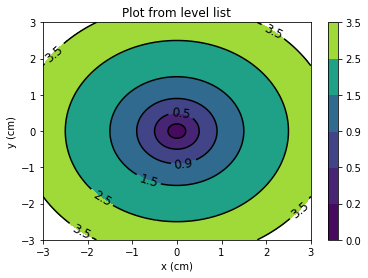
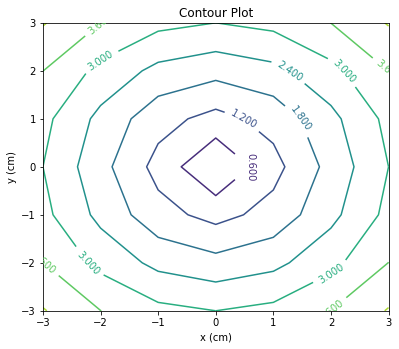
**Contour:**

It is usually used to represent a 3-d array on a 2-d platform.

In the code you’ll observe that I used the meshgrid X and Y coordinates and in place of Z coordinates, the equation of conic. The Z coordinate array, in this case the equation of conic, has a parameter [Value, to satisfy the equation] to make the equation of conics mathematically sound.

Contour has many benefits, first and foremost being the ease with which you can print complex graphs, along with features like adding colors and label lines with different altitudes/ heights (called contour).

This will also help us in the future to add multiple conics on the same graphs, with different altitudes to add a parallax effect, which will help us, differentiate these graphs with ease. Some contour output is as follows:



**os module**

This module helps to create, read or write in text files and provides a portable way of using operating system dependent functionality. It provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

This module has been used to store the username and password of users. After every successful registration, it creates a text file named after the username. During login, the text file of the username entered is traversed to confirm the password.

**datetime module**

The datetime module supplies classes for manipulating dates and times. We used the function datetime.now() which returns the year, month, day, hour, minute, second, and microsecond. This data was saved in a variable and stored into the database.

**PROJECT CODE**

from Tkinter import \*

#importing Tkinter for the gui

import os

#importing os so as to access the file, as we have to read and write usernames and passwords

from datetime import datetime

import mysql.connector as sql

con1 = sql.connect(host = 'localhost', username='root', passwd='Hk@191202', database='csproj')

cur1 = con1.cursor()

def delete1():

screen1.destroy()

#screen1.deiconify()

#screen1.quit()

def delete2():

screen3.destroy()

def delete3():

screen4.destroy()

def delete4():

screen5.destroy()

def login\_sucess():

import graphcs

def password\_not\_recognised():

global screen4

screen4 = Toplevel(screen)

screen4.title("Success")

screen4.geometry("150x100")

Label(screen4, text = "Password Error").pack()

Button(screen4, text = "OK", command =delete3).pack()

def user\_not\_found():

global screen5

screen5 = Toplevel(screen)

screen5.title("Success")

screen5.geometry("150x100")

Label(screen5, text = "User Not Found").pack()

Button(screen5, text = "OK", command =delete4).pack()

def register\_user():

print("working")

username\_info = username.get()

password\_info = password.get()

dob\_info = dob.get()

gen\_info = gen.get()

age\_info = age.get()

st1 = 'insert into REGISTRATION values(%s,%s,%s,%s,%s)'

b11 = username\_info

b12 = password\_info

b13 = dob\_info

b14 = gen\_info

b15 = age\_info

val1 = [(b11,b12,b13,b14,b15)]

cur1.executemany(st1,val1)

con1.commit()

file=open(username\_info, "w")

file.write(username\_info+"\n")

#to add a new line in the file

file.write(password\_info)

file.close()

username\_entry.delete(0, END)

password\_entry.delete(0, END)

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Registration Sucess", fg = "green" ,font = ("calibri", 11)).pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Button(screen1, text = "Okay, take to login page ", command=lambda:[login(),delete1()]).pack()

#command lamda fn is used to execute multiple fns at the same time :)

def login\_verify():

username1 = username\_verify.get()

password1 = password\_verify.get()

username\_entry1.delete(0, END)

password\_entry1.delete(0, END)

list\_of\_files = os.listdir()

if username1 in list\_of\_files:

file1 = open(username1, "r")

verify = file1.read().splitlines()

if password1 == verify[1]:

login\_sucess()

l1 = username1

l2 = datetime.now()

l3 = 'Login Success'

vl1 = [(l1,l2,l3)]

lg1 = 'insert into LOGIN\_INFO values(%s,%s,%s)'

cur1.executemany(lg1,vl1)

con1.commit()

else:

password\_not\_recognised()

else:

user\_not\_found()

##get password \*

def register():

global screen1

screen1 = Toplevel(screen)

screen1.geometry("600x600")

screen1['bg'] = '#A55D35'

Label(screen1, text = "", bg = "#A55D35").pack()

#making the variables global so that we can access them in resigter verfiy function

#photo1 = PhotoImage(file = r"logo\_new2.png")

#labelphoto1 = Label(screen1, image = photo1)

#labelphoto1.pack()

global username

global password

global dob

global gen

global age

global username\_entry

global password\_entry

global dob\_entry

global gen\_entry

global age\_entry

username = StringVar()

password = StringVar()

dob = StringVar()

gen = StringVar()

age = StringVar()

Label(screen1, text = "Please enter details below").pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Username \* ").pack()

username\_entry = Entry(screen1, textvariable = username)

username\_entry.pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Password \* ").pack()

password\_entry = Entry(screen1, textvariable = password)

password\_entry.pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Date of birth \* ").pack()

dob\_entry = Entry(screen1, textvariable = dob)

dob\_entry.pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Gender \* ").pack()

gen\_entry = Entry(screen1, textvariable = gen)

gen\_entry.pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Label(screen1, text = "Age \* ").pack()

age\_entry = Entry(screen1, textvariable = age)

age\_entry.pack()

Label(screen1, text = "", bg = "#A55D35").pack()

Button(screen1, text = "Register", width = 10, height = 1, command = register\_user).pack()

def login():

global screen2

print('login Working ')

screen2 = Toplevel(screen)

#photo2 = PhotoImage(file = r"logo\_new1")

#labelphoto = Label(screen2, image = photo2)

#labelphoto.pack()

screen2.title("Login")

screen2['bg'] = '#A55D35'

screen2.geometry("600x600")

Label(screen2, text = "", bg = "#A55D35").pack()

Label(screen2, text = "Please enter details below to login").pack()

Label(screen2,text = "", bg = "#A55D35").pack()

global username\_verify

global password\_verify

username\_verify = StringVar()

password\_verify = StringVar()

# user and pass are used as string var, since we are not using input fn, doing the same is conisdered a good progtamming practice

global username\_entry1

global password\_entry1

Label(screen2, text = "Username \* ").pack()

username\_entry1 = Entry(screen2, textvariable = username\_verify)

username\_entry1.pack()

Label(screen2, text = "", bg = "#A55D35").pack()

Label(screen2, text = "Password \* ").pack()

password\_entry1 = Entry(screen2, textvariable = password\_verify)

password\_entry1.pack()

Label(screen2, text = "", bg = "#A55D35").pack()

Button(screen2, text = "Login", width = 10, height = 1, command = login\_verify).pack()

Label(screen2, text = "\* Means Required ",fg = "red" ,font = ("calibri", 11)).pack()

Label(screen2, text = "", bg = "#A55D35").pack()

def main\_screen():

global screen

print('main\_screen Working')

screen = Tk()

photo = PhotoImage(file = r"logo\_new1.png")

labelphoto = Label(screen, image = photo)

labelphoto.pack()

screen.geometry("600x600")

screen.title("Login page")

screen['bg'] = '#A55D35'

Label(text = "Conics with Matpltlib", bg = "grey", width = "300", height = "2", font = ("Calibri", 13)).pack()

Label(text = "", bg = "#A55D35").pack()

Button(text = "Login", height = "2", width = "30", command = login).pack()

Label(text = "", bg = "#A55D35").pack()

Button(text = "Register",height = "2", width = "30", command = register).pack()

screen.mainloop()

main\_screen()

from Tkinter import \*

import matplotlib.pyplot as plt

import numpy as np

def StraightBut():

global screenstraight

print('StraightBut Working')

screenstraight = Toplevel(graphscreen)

screenstraight.title("Straightlines")

screenstraight.geometry("1080x720")

Label(screenstraight, text = "Graphs with Matpltlib -- Straightlines", bg = "grey", width = "300", height = "2", font = ("Calibri", 13)).pack()

global straightlineX

global straightlineY

global straightlineK

def delete2():

screenstraight.destroy()

def CalculateST():

x = np.linspace(-60,61, 200)

def axes():

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

a = straightlineX.get()

b = straightlineY.get()

c = straightlineK.get()

y = (a/(-b))\*x+(c/(-b))

eqn = str(a) + " x " +"+"+ str(b) + " y " +"+"+ str(c)+ " = 0"

#Printing Vertical line, Y axis

plt.plot(x, y, '-r', label=eqn)

plt.title('Graph of ' + eqn)

plt.xlabel('x', color='#1C2833')

plt.ylabel('y', color='#1C2833')

plt.legend(loc='upper left')

plt.plot(0,c)

plt.axis('equal')

axes()

plt.show()

straightlineX = IntVar()

straightlineY = IntVar()

straightlineK = IntVar()

Label(screenstraight, text = "Please enter details below").pack()

Label(screenstraight, text = "Given equation ax + by + c= 0").pack()

Label(screenstraight, text = "").pack()

Label(screenstraight, text = "Coefficient of X, i.e a ").pack()

straightlineX\_entry = Entry(screenstraight, textvariable = straightlineX)

straightlineX\_entry.pack()

Label(screenstraight, text = "").pack()

Label(screenstraight, text = "Coefficient of Y, i.e b").pack()

straightlineY\_entry = Entry(screenstraight, textvariable = straightlineY)

straightlineY\_entry.pack()

Label(screenstraight, text = "Constant, i.e c").pack()

straightlineK\_entry = Entry(screenstraight, textvariable = straightlineK)

straightlineK\_entry.pack()

Label(screenstraight, text = "").pack()

Button(screenstraight, text = "Calculate", height = "2", width = "30", command = CalculateST).pack()

Button(screenstraight, text = "back to homescreen",height = "2", width = "30", command = delete2).pack()

def Conics():

global screenconics

print('StraightBut Working')

screenconics = Toplevel(graphscreen)

screenconics.title("Conics")

screenconics.geometry("1080x720")

Label(screenconics, text = "Graphs with Matpltlib -- Straightlines", bg = "grey", width = "300", height = "2", font = ("Calibri", 13)).pack()

def CalculateConics():

x = np.linspace(-60,61, 200)

y = np.linspace(-50, 51, 200)

#meshgrid makes an array, this is useful in defining functions

x, y = np.meshgrid(x, y)

def axes():

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

plt.axis('equal')

a = aco.get()

b = bco.get()

c = cco.get()

f = fco.get()

g = gco.get()

h = hco.get()

eqn = str(a)+'\*x\*\*2'+'+' + str(h)+'\*x\*y'+'+' + str(b)+'\*y\*\*2' + '+' + str(g)+'\*x'+'+' + str(f)+'\*y' + str(c)

#Printing Vertical line, Y axis

plt.title('Graph of ' + eqn)

plt.xlabel('x', color='#1C2833')

plt.ylabel('y', color='#1C2833')

plt.legend(loc='upper left')

plt.contour(x, y,(a\*x\*\*2 + h\*x\*y + b\*y\*\*2 + g\*x + f\*y + c), [0], colors='k')

plt.axis('equal')

axes()

plt.show()

global aco

global bco

global cco

global fco

global gco

global hco

aco = IntVar ()

bco = IntVar ()

cco = IntVar ()

fco = IntVar ()

gco = IntVar ()

hco = IntVar ()

def delete2():

screenconics.destroy()

Label(screenconics, text = "Please enter details below").pack()

Label(screenconics, text = "Given equation a\*x\*\*2 + h\*x\*y + b\*y\*\*2 + g\*x + f\*y + c").pack()

Label(screenconics, text = "").pack()

Label(screenconics, text = "Coefficient of X^2, i.e a ").pack()

conicsX2\_entry = Entry(screenconics, textvariable = aco)

conicsX2\_entry.pack()

Label(screenconics, text = "").pack()

Label(screenconics, text = "Coefficient of Y^2, i.e b ").pack()

conicsY2\_entry = Entry(screenconics, textvariable = bco)

conicsY2\_entry.pack()

Label(screenconics, text = "").pack()

Label(screenconics, text = "Coefficient of Y, i.e f").pack()

straightlineY\_entry = Entry(screenconics, textvariable = fco)

straightlineY\_entry.pack()

Label(screenconics, text = "Coefficient of X, i.e g").pack()

straightlineY\_entry = Entry(screenconics, textvariable = gco)

straightlineY\_entry.pack()

Label(screenconics, text = "Coefficient of XY, i.e h").pack()

straightlineK\_entry = Entry(screenconics, textvariable = hco)

straightlineK\_entry.pack()

Label(screenconics, text = "").pack()

Label(screenconics, text = "Constant, i.e c").pack()

straightlineK\_entry = Entry(screenconics, textvariable = cco)

straightlineK\_entry.pack()

Label(screenconics, text = "").pack()

Button(screenconics, text = "Calculate", height = "2", width = "30", command = CalculateConics).pack()

Button(screenconics, text = "back to homescreen",height = "2", width = "30", command = delete2).pack()

def Trigo():

def delete2():

screentrigo.destroy()

def axes():

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Vertical line, Y axis

plt.axis('equal')

for i in (-1\*np.pi,1\*np.pi,np.pi/2,-1\*np.pi/2):

plt.plot(i,0,'.')

plt.axhline(1, alpha= .2, linewidth= 2, color='g' )

plt.axhline(-1, alpha= .2, linewidth= 2, color='g' )

plt.xlabel('Angle-Input')

plt.ylabel('Integer-Output')

def sin():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.sin(x))

plt.ylim(-5, 5)

axes()

plt.show()

def cos():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.cos(x))

plt.ylim(-5, 5)

axes()

plt.show()

def tan():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.tan(x))

plt.ylim(-5, 5)

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

plt.show()

def cot():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

while True:

try:

plt.plot(x, 1/np.tan(x))

plt.ylim(-5, 5)

except:

continue

finally:

break

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

plt.show()

def cosec():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

while True:

try:

plt.plot(x, 1/np.sin(x))

plt.ylim(-5, 5)

except:

continue

finally:

break

plt.axis('equal')

axes()

plt.show()

def sec():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 70)

while True:

try:

plt.plot(x, 1/np.cos(x))

plt.ylim(-5, 5)

except:

continue

finally:

break

plt.axis('equal')

axes()

plt.show()

def tan2x():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.tan(2\*x))

plt.ylim(-5, 5)

plt.grid()

plt.axhline(0, alpha= .2, linewidth= 2, color='k' )

#Printing Horizontal line, X axis

plt.axvline(0, alpha= .2, linewidth= 2, color='k' )

plt.show()

def sin2x():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.sin(2\*x))

plt.ylim(-5, 5)

axes()

plt.show()

def cos2x():

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 1000)

plt.plot(x, np.cos(2\*x))

plt.ylim(-5, 5)

axes()

plt.show()

global screentrigo

screentrigo = Toplevel(graphscreen)

screentrigo.title('TrigoFunctions')

screentrigo.geometry("1080x720")

Label(screentrigo, text = "Graphs with Matpltlib -- TrigoFunctions", bg = "grey", width = "300", height = "2", font = ("Calibri", 13)).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='sine',height = "2", width = "30", command = sin).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Cos',height = "2", width = "30", command = cos).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Tan',height = "2", width = "30", command = tan).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Cot',height = "2", width = "30", command = cot).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='cosec',height = "2", width = "30", command = cosec).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='sec',height = "2", width = "30", command = sec).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='sin2x',height = "2", width = "30", command = sin2x).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Cos2x',height = "2", width = "30", command = cos2x ).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Tan2x',height = "2", width = "30", command = tan2x ).pack()

Label(screentrigo, text = "").pack()

Button(screentrigo, text ='Back to Homescreen',height = "2", width = "30",command= delete2).pack()

return True

def graphmain():

global graphscreen

graphscreen = Tk()

graphscreen.title("Main")

graphscreen.geometry("600x600")

graphscreen['bg'] = '#A55D35'

Label(graphscreen,text = "Conics with Matpltlib", bg = "grey", width = "300", height = "2", font = ("Calibri", 13)).pack()

Label(graphscreen,text = "", bg = "#A55D35").pack()

Button(graphscreen,text = "Straight Lines", height = "2", width = "30", command = StraightBut).pack()

Label(graphscreen,text = "", bg = "#A55D35").pack()

Button(graphscreen,text = "Conics", height = "2", width = "30", command = Conics).pack()

Label(graphscreen,text = "", bg = "#A55D35").pack()

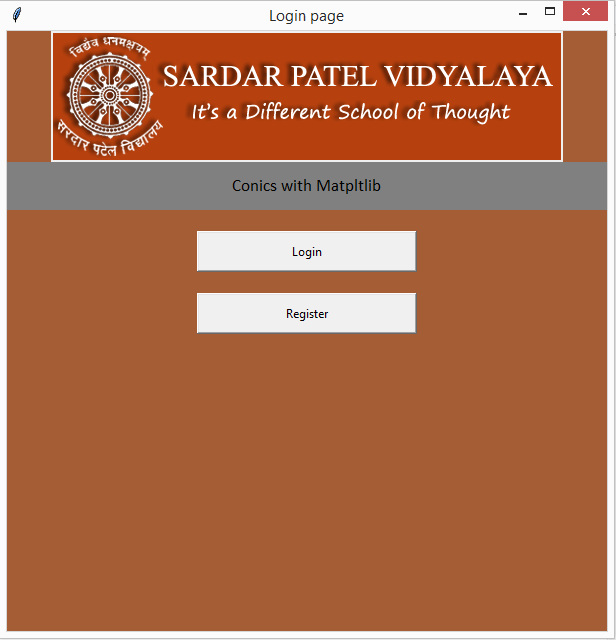
Button(graphscreen,text = "TrigoFunctions", height = "2", width = "30", command = Trigo).pack()

Label(graphscreen,text = "", bg = "#A55D35").pack()

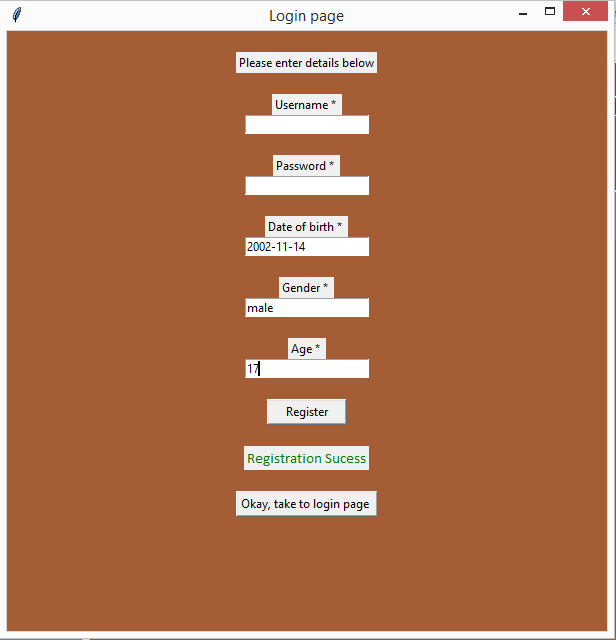
graphmain()

**OUTPUT SCREENS**

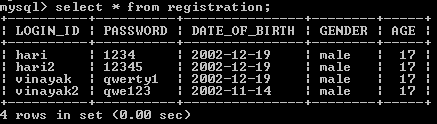
**Main Window**

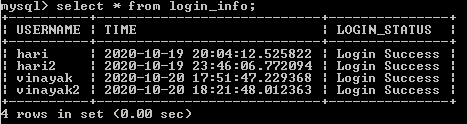


**Registration**

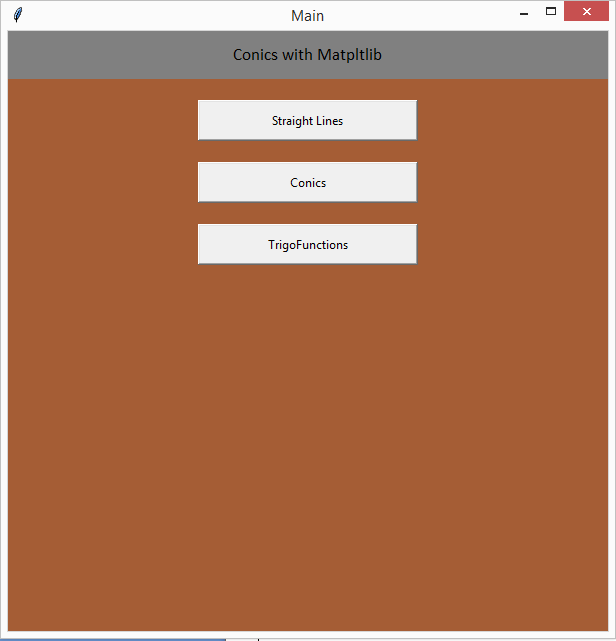


**SQL tables (also used datetime module)**

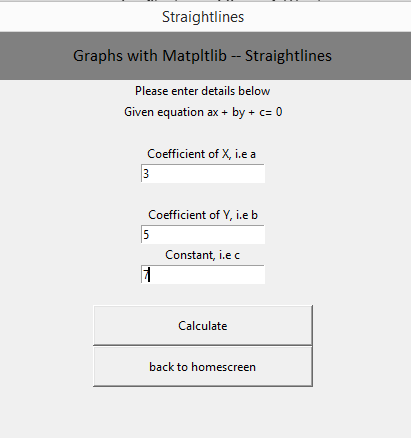




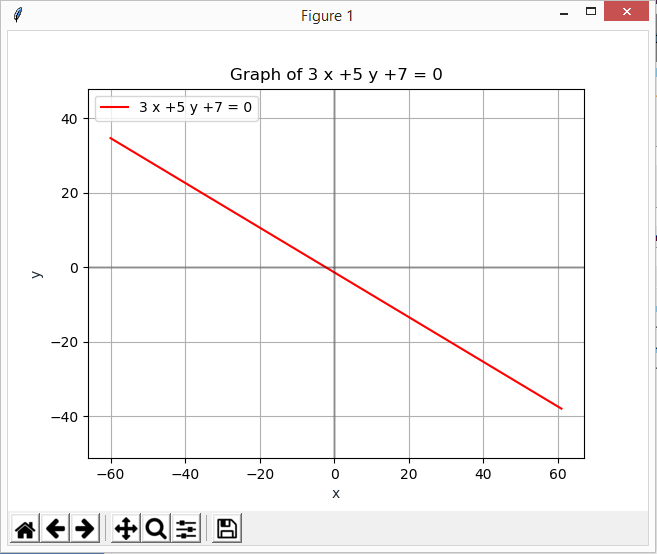
**Conics options window**



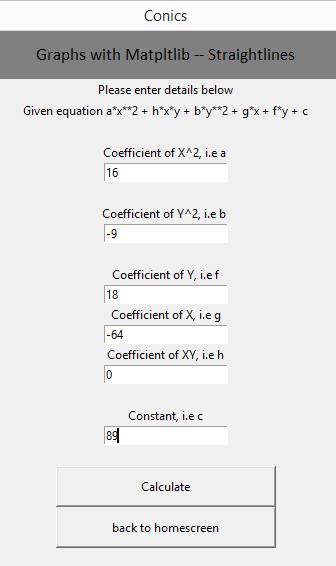
**Straight Line**



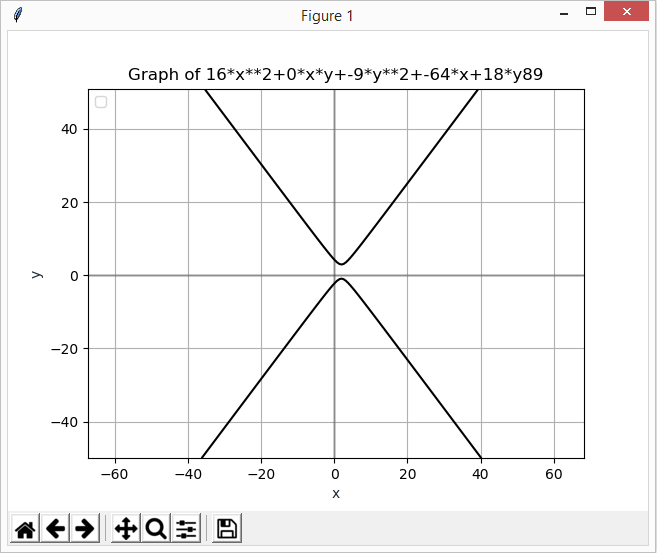
**Graph of Line**



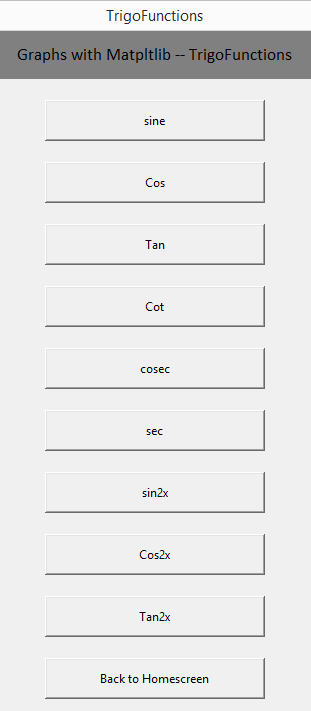
**Conics**



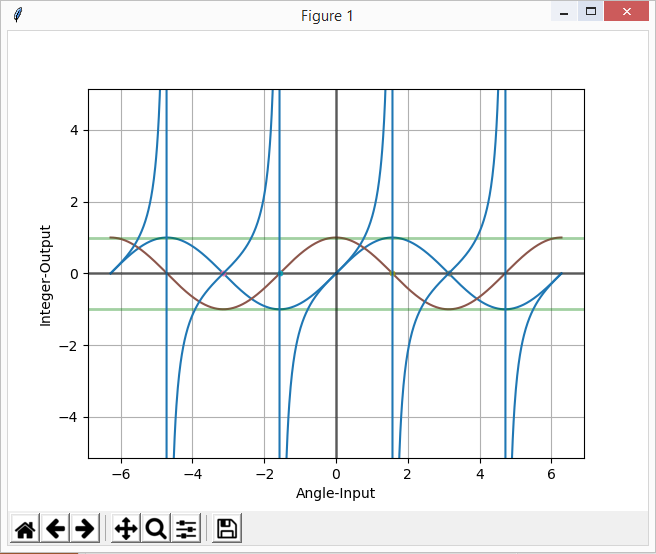
**Graph of Conics**



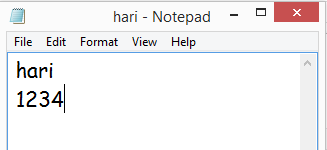
**TrigoFunctions**



**Graph Of TrigoFunctions**



**Text files (os module)**





**CONCLUSION**

Our project is an application that provides the user to observe and plot the graphs of specific mathematical functions- straight lines, conics and trigonometric functions. It is a user friendly program that allows the user to input values of constants in the mathematical expressions and plots the graph of the conic/straight line thus formed. Matplotlib and NumPy together make mainpulation and plotting of complex data much easier. We also used the Contour function, which is helpful in adding features like colours, labels and multiple outputs on a screen. After researching on how to integrate the different functions and modules in an efficient manner, we were able to produce the desired output without errors. We tried out numerous ways to code to achieve the output, and were able to settle on one we found highly optimisable.

The GUI that we created for this program using Tkinter made is user friendly and very easy to use. All inputs are taken in text boxes or buttons with instructions, therefore it's usage is not time consuming. We had to do a lot of research on the different functions Tkinter provided to create a GUI that could be integrated with our code.

We had the opportunity to learn the functioning and efficient integration of multiple modules and libraries thanks to this project.

**BIBLIOGRAPHY**

1. Sumita Arora, *Computer Science with Python*, 2020
2. https://www.geeksforgeeks.org/os-module-python-examples/
3. https://www.geeksforgeeks.org/working-with-images-in-python-using-matplotlib/
4. https://www.geeksforgeeks.org/python-gui-tkinter/
5. <https://www.python-course.eu/matplotlib_contour_plot.php>
6. <https://mmas.github.io/conics-matplotlib>
7. <https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/>
8. https://www.geeksforgeeks.org/python-creating-a-button-in-tkinter/