AMITY INTERNATIONAL SCHOOL

COMPUTER SCIENCE PROJECT

An Interactive GRAPH PLOTTER

Shantanu, Nikunj, Abhinav, Raghav

ACKNOWLEDGEMENT

I acknowledge the valuable contribution of our computer teacher Ms. Deepshikha Sethi in providing me the proper guidance to complete the computer science project. The programs would not have been completed without her support and kind help. I would also like to thank the lab assistants for their help.

Name of Student

CERTIFICATE

AMITY INTERNATIONAL SCHOOL, MAYUR VIHAR

This is to certify that Shantanu Sharma of class twelve, Amity International School, Mayur Vihar has successfully completed his project in computer practicals for the AISSCE as prescribed by CBSE in the year 2015-2016.

Date :

Registration No. :

Signature of Internal Signature of
External

Examiner Examiner

__________________ __________________

CONTENTS

AIM

This project is aimed to design and demonstrate an interactive graph plotter which can plot the following graphs and figures:

- GRAPH OF TRIGONOMETRIC FUNCTIONS
- CIRCLE
- CYCLOID
- REGULAR POLYGON
- HYPOCYCLOID
- EPICYCLOID
- POLYNOMIALS

INTRODUCTION

A graph is a planned drawing, consisting of lines and relating numbers to one another. Any student studying math or doing analysis of any kind, needs to make use of structure. This is done by using and studying a graph. Graphing is used daily has to one day face the problem of analyzing graphs. Inspired by the android hit app calculator++ our team decided to come with a model of our own making.

The project uses one of the most popular computer application Turtle, which introduced us to programming in 4th standard and with some advanced tweaks is rigorously used in our program.

The program is a menu driven, user friendly graph plotter with well defined layout which is based on python and turtle graphics and draws graph of 7 fundamental functions including trigonometric functions, polynomial functions in python. It also plots curves like cycloid, epicycloid, hypocycloid which are used in designing and architecture.

Our Graphical User Interface (GUI), which is created using turtle and tkinter for the underlying graphics, provides a rich environment in which information can be exchanged between a user and the computer, as GUI's are not limited to simply displaying text and reading text from the keyboard. Our GUI enables users to control the behavior of the plotter by performing actions such as plotting graph according to the desired coordinates, shifting the graph left or right along the axis according to the need of the user and also returning to the previous graph by interacting with the well defined menu with elaborate options. Our interface makes using the graph plotter much more intuitive and easier to learn since it provides users with immediate visual feedback that shows the effects of their actions.

We have made use of list, classes, functions, os module, tkinter module, pickle module, time module, turtle module and graphics, oop concepts, datafile handling.

MODULES USED

TIME: The time module of python standard library provides the time class. Time values are represented with the time class. Times have attributes for hour, minute, second and microsecond. The time type represents a time, independent of the date.

MATH: Python has a math module that provides various mathematical functions.

OS: Python os module is used to perform tasks such as finding the name of present working directory, changing current working directory, checking if certain files or directories exist at a location, creating new directories, deleting existing files or directories, walking through a directory and performing operations on every file in the directory that satisfies some user defined criteria etc.

PICKLE: The pickle module implements a fundamental, but also a powerful algorithm for serializing and de- serializing a python object structure. The object structure could be a variable, instance of a class, or a list, dictionary, or tuple. Picking is just serialization i.e., putting data into a form that can be stored in a file and retrieved later.

TKFONT: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface.

TURTLE: The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. “Turtle” is a python feature like a drawing board, which lets you command a turtle to draw all over it. it uses Tkinter for the underlying graphics

CLASSES & FUNCTIONS USED

1. <u>CLASS trigo</u>: This class is responsible for representing trigonometry functions graphically

Functions used in class trigo:

1. trigo_main(q):

This function makes the trigonometric graph.

2. trigo_axis_make():

This function makes the axes of the graph.

3. trigo_mark_axis():

This function marks the coordinates of the function.

4. trigo_shift(b,q):

This function is responsible for shifting the graph left or right about the axes.

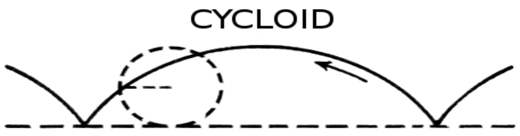
2. <u>CLASS CIRCLE</u>: This class is responsible for creating a circle

Functions used in class Circle: 1. drawCircle(centerpoint,radius):

This function gets the user to input the centre and radius if the circle. The radius is multiplied by 10 so that the graphic illustrations are clear and evident

3. <u>CLASS Cycloid</u>: This class responsible for creating a cycloid.

Cycloid: A cycloid is the curve traced by a point on the rim of a circular wheel as the wheel



rolls along a straight line without slippage

Functions used in class Cycloid:1.def cycloid():

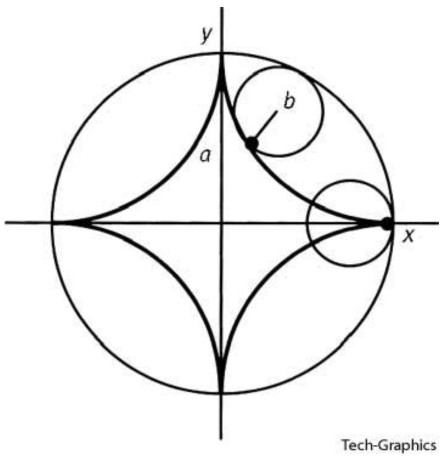
Creates a cycloid using the equation of the cycloid:
turtle.goto(15*(i/10-math.sin(i/10))-300,15*(1-math.cos(i/10)))

4. CLASS regular_polygon: This class responsible for creating a regular polygon

Functions used in class Cycloid: 1. def polygon():

Asks the user to input the no. of sides. The angle of each vertice of a regular polygon is 360 divided by the number of sides. turtle.left rotates turtle at its position by degrees given in argument.

5. CLASS hypocycloid: This class responsible for creating a hypocycloid.



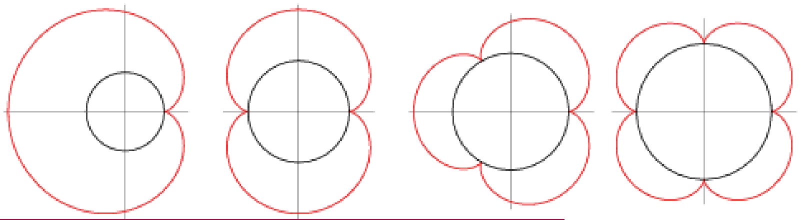
Hypocycloid: The curve produced by fixed point P on the circumference of a small circle of radius b rolling around the inside of a large circle of radius $a > b$.

Functions used in class hypocycloid: 1. def hypo(k):

This function forms a hypocycloid. It takes no of cusps where cusps is a sharp point on the curve. The formula is:
turtle.goto(r*(k-1)*math.cos(i)+r*math.cos((k-1)*i),r*(k-1)*math.sin(i)-r*math.sin((k-1)*i))

6. CLASS epicycloid: This class responsible for creating a epicycloid.

Epicycloid: The path traced out by a point P on the edge of a circle of radius b rolling on the outside of a circle of radius a



Functions used in class hypocycloid: 1. def epicycloid(k):

This function forms an epicycloid. It takes no of cusps where cusps is a sharp point on the curve. The formula is:

turtle.goto(r*(k+1)*math.cos(i)-r*math.cos((k+1)*i),r*(k+1)*math.sin(i)-r*math.sin((k+1)*i))

6. CLASS polynomial: This class responsible for representing polynomial functions graphically

Functions used in class hypocycloid:

1. def new_polynomial():

This function enters a list of coefficients for the function.

2. def poly_make_axis():

This function makes the axes.

3. def poly_main(s):

This function plots the graph of the function.

SOURCE CODE

```
import time, math, os, pickle

import tkFont

import turtle


class trigo:


    @staticmethod

    def trigo_main(q):

        #gives colour in rgb mode with scale 255

        turtle.colormode(255)

        turtle.pencolor((197, 83, 253))

        turtle.width(1.4)


        turtle.penup()

        turtle.goto(-250,0)

        turtle.pendown()


# we have multiplied by 10 in x and y coordinate so that graph looks bigger and readable and shifted actual axis by
250 pixles to make whole graph visible

        for i in range (50):

            if q=="cos":

                turtle.goto((i*10-250),10*math.cos(i))

            if q=="sin":

                 turtle.goto((i*10-250),10*math.sin(i))

        turtle.penup()


        b = raw_input("Enter where you want to shift graph\n\t1.Nowhere\n\t2.Right\n\t3.Left\n")

        trigo.trigo_shift(b,q)


    @staticmethod
```

```
def trigo_axis_make():
    turtle.goto(0,0)
    turtle.pendown()
    turtle.pencolor("black")
    #making X coordinates
    turtle.goto(-300,0)
    turtle.goto(300,0)
    turtle.goto(0,0)
    #making Y coordinates
    turtle.goto(0,20)
    turtle.goto(0,-20)
    turtle.goto(0,0)

    turtle.penup()

@staticmethod
def trigo_mark_axis():
    turtle.pencolor("blue")
    turtle.width(1)
    #marking Y coordinates
    #marks 1 on yaxis
    turtle.penup()
    turtle.goto(-5,10)
    turtle.pendown()
    turtle.write("1",font=("Arial",6,"normal"))

    #marks -1 on yaxis
    turtle.penup()
    turtle.goto(-7,-20)
    turtle.pendown()
    turtle.write("-1",font=("Arial",6,"normal"))
```

```
        #marking x coordinate

        turtle.colormode(255)

        #let us obtain color in rgb color codes which has code range of each color of 255

        turtle.pencolor((103, 150, 8))

        turtle.width(1)

        for i in range (16):

            if i==8:

            # to write for 0 and other as it will print 0pi then it is i=8 bcoz i have shifted graph toward left by 250 pixles to
make whole graph come in window and 8*pi*10 will give value around 251

                turtle.penup()

                turtle.goto(2,-10)

                turtle.pendown()

                turtle.write("0",font=("Arial",6,"bold"))

            else:

                #marks a kink in graph

                turtle.penup()

                turtle.goto((math.pi*i*10)-(250),+4)

                turtle.pendown()

                turtle.goto((math.pi*i*10)-(250),-4)

                turtle.penup()


                #writes the value

                turtle.goto((math.pi*i*10)-(250),-20)

                turtle.pendown()

                turtle.write((str(i-8)+("pi")) ,font=("Arial",6,"bold"))


    @staticmethod

    def trigo_shift(b,q):


        turtle.colormode(255)


        turtle.pencolor((240, 169, 108))
turtle.width(1)


c = input("Enter the value till you want to shift if you want to shift\nright\nleft\nnowhere")

d = c*10


if b.lower() in ["right"]:

    #to get graph lying on same axis

    for i in range (-60,60):

         if -251<(i*10-250+d)<251:

             if q=="cos":

                turtle.goto((i*10-250+d),10*math.cos(i))

             if q=="sin":

                 turtle.goto((i*10-250+d),10*math.sin(i))

             turtle.pendown()


elif b.lower() in ["left"]:

    for i in range (-60,60):

      if -251<(i*10-250-d)<251:

        if q=="cos":

            turtle.goto((i*10-250-d),10*math.cos(i))

        if q=="sin":

            turtle.goto((i*10-250-d),10*math.sin(i))

        turtle.pendown()


elif b.lower() in ["nowhere", "no"]:

    turtle.penup()

    turtle.goto(0,0)


else:

    turtle.exitonclick()

    exit()
```

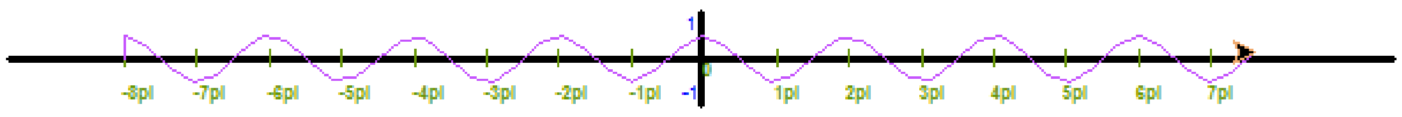
```
class circle:


    turtle.colormode(255)

    turtle.pencolor((240, 169, 108))

    turtle.width(3.2)


    @staticmethod

    def drawCircle (centerpoint, radius):

        (x,y) = centerpoint

        turtle.up()

        turtle.setpos(x,y)

        turtle.down()

        #r*10 so that circle looks bigger

        turtle.circle(radius*10)


class cycloid:


    @staticmethod

    def cycloid():

        turtle.penup()

        turtle.goto(-300,0)

        turtle.pendown()

        turtle.colormode(255)

        turtle.pencolor((240, 169, 108))

        turtle.width(3.2)

        turtle.speed(1000)


        for i in range (400):

            #this is equation of cycloid

            turtle.goto(15*(i/10-math.sin(i/10))-300,15*(1-math.cos(i/10)))
```

```
class regular_polygon:


    @staticmethod
    def polygon(a):
        if a>2:
            turtle.pendown()
            for i in range(a+1):
                turtle.forward(50)
    # the angle of each vertice of a regular polygon is 360 divided by the number of sides
    # turtle.left rotates turtle at its position by dgrees given in argument
                turtle.left(360/a)
        else:
            print "invalid input"
            exit()



class hypocycloid:


  @staticmethod
  def hypo(k):
    r=50
    #k is no. of cusps.........Cusp- A sharp point on a curve.........If k is a rational number, say k = p/q expressed in simplest terms, then the curve has p cusps.
    #If k is an irrational number, then the curve never closes, and fills the space between the larger circle and a circle of radius R ? 2r


    turtle.colormode(255)
    turtle.pencolor((74, 185, 240))
    turtle.width(7)


    #have taken radius divided by 10 as in circle class have multiplied incoming radius into 10 to make circle bigger
    circle.drawCircle((0,-r*k),r*k/10)
    turtle.penup()

    turtle.goto(r*k,0)

    turtle.pendown()


    turtle.speed(120)

    turtle.colormode(255)

    turtle.pencolor((75, 201, 142))

    turtle.width(.8)

    for  i in range(0,1000):

        turtle.goto(r*(k-1)*math.cos(i)+r*math.cos((k-1)*i),r*(k-1)*math.sin(i)-r*math.sin((k-1)*i))


    turtle.penup()

    turtle.goto(-300,-r*k-30)

    turtle.pendown()

    turtle.pencolor("black")

    turtle.width(2)

    turtle.write("Quick Fact: \nfor k=2 the curve is a straight line and the circles are called Cardano circles.\nThey are used in high-speed printing",font=("Arial",8,"normal"))



class epicycloid:


  @staticmethod

  def epicycloid(k):

    r=50

    turtle.colormode(255)

    turtle.pencolor((74, 185, 240))

    turtle.width(7)

    circle.drawCircle((0,-r*k),r*k/10)


    turtle.penup()

    #as at i=0--in equation of epicycloid,turtle is at (r*k,0)
    turtle.goto(r*k,0)
    turtle.pendown()

    turtle.speed(120)
    turtle.colormode(255)
    turtle.pencolor((75, 201, 142))
    turtle.width(.8)
    for  i in range(0,1000):
        turtle.goto(r*(k+1)*math.cos(i)-r*math.cos((k+1)*i),r*(k+1)*math.sin(i)-r*math.sin((k+1)*i))



class polynomial:

    @staticmethod
    def new_polynomial():

        s=[]
        a = input("Degree of polynomial\n")
        for i in range (a+1):
            b = input("Enter coefficient of power "+ str(i)+ "\t")
            s+=[b]
        return  s

    @staticmethod
    def poly_make_axis():
        turtle.colormode(255)
      #let us obtain color in rgb color codes which has code range of each color of 255
        turtle.pencolor((103, 150, 8))
        turtle.width(1)
        #x axis
        turtle.goto(-300,0)
        turtle.goto(300,0)
        turtle.goto(0,0)


        #y axis

        turtle.pendown()

        turtle.goto(0,250)

        turtle.goto(0,-250)

        turtle.penup()

        turtle.goto(0,0)


    @staticmethod

    def poly_main(s):


        turtle.colormode(255)

        turtle.pencolor((240, 169, 108))

        turtle.width(1.75)


        for j in range(-40,40):

            p=0


       #s(list)is of form {2,1,2,-3}.......meaning coefficient of x^0 is 2,x^1 is 1,x^2 is 2,x^3 is -3

            for i in range(len(s)):

                p+=s[i]*(j**i)

                #here p becomes the polynomial...  -3(i^3)+2(i^2)+1(i^1)+2(i^0)

                #in which i is varied to make graph



            #to contain graph in window

            if -290<p<290:

                turtle.goto(j*10,p)

                turtle.pendown()


            if p==0:
```

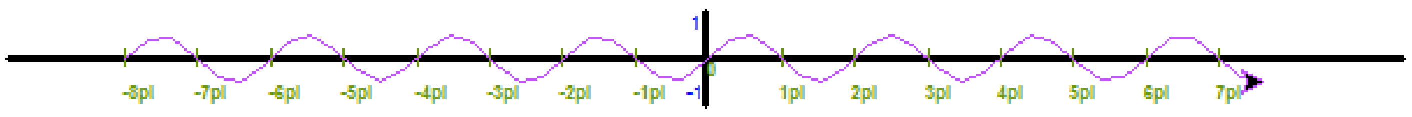
```
            #if by chance, a root comes integer in -40 to 40...it is marked

            turtle.penup()

            turtle.goto(j*10,0)

            turtle.pendown()

            #make kink

            turtle.goto(j*10,-4)

            turtle.goto(j*10,+4)

            turtle.penup()

            #write the integer root....try drawing (x-2)^3={-8,12,-6,1}..(values to be put according to order when
program is run

            turtle.goto(j*10,-15)

            turtle.pendown()

            turtle.write(j)

            turtle.penup()

            turtle.goto(j*10,0)

            turtle.pendown()

            #and graph is continued from this pt again


def new():

  y=raw_input("What do you want to draw -\n1.Cos\n2.Sin\n3.Circle\n4.Regular
polygon\n5.Polynomial\n6.Cycloid\n7.Hypocycloid\n8.Epicycloid\n")


  if y.lower() in ["1", "cos"]:

      trigo.trigo_axis_make()

      trigo.trigo_mark_axis()

      trigo.trigo_main("cos")

      var = 0


  if y.lower() in ["2", "sin"]:

      trigo.trigo_axis_make()

      trigo.trigo_mark_axis()

      trigo.trigo_main("sin")

      var = 0
```

```
if y.lower() in ["3", "circle"]:

    a = input("Enter x coordinate of circle\t")

    b = input("Enter y coordinate of circle\t")

    r = input("Enter radius\t")

    circle.drawCircle((a,b),r)

    var = (a,b,r)


if y.lower() in ["4", "regular polygon"]:

    a = input("Enter number of sides of polygon\t")

    regular_polygon.polygon(a)

    var = a



if y.lower() in ["5", "polynomial"]:

    a = polynomial.new_polynomial()

    polynomial.poly_make_axis()

    polynomial.poly_main(a)

    var = a


if y.lower() in ["6", "cycloid"]:

    cycloid.cycloid()

    var = 0


if y.lower() in ["7", "hypocycloid"]:

    k = input("Enter no. of cusps\t")

    #to make cardano line--k=2

    #to make deltoid---k=3

    #to make asteroid--k=4

    #to make star---k=5

    hypocycloid.hypo(k)

    var = k
    if y.lower() in ["8", "epicycloid"]:

        k = input("Enter no. of cusps\t")

        #for best result---k=2 or 3 or 5

        epicycloid.epicycloid(k)

        var = k


    return (y,var)


def prev(y, var):

    if y.lower() in ["1", "cos"]:

        trigo.trigo_axis_make()

        trigo.trigo_mark_axis()

        trigo.trigo_main("cos")


    if y.lower() in ["2", "sin"]:

        trigo.trigo_axis_make()

        trigo.trigo_mark_axis()

        trigo.trigo_main("sin")


    if y.lower() in ["3", "circle"]:

        (a,b,r) = var

        circle.drawCircle((a,b),r)


    if y.lower() in ["4", "regular polygon"]:

        a = var

        regular_polygon.polygon(a)


    if y.lower() in ["5", "polynomial"]:

        a = var

        polynomial.poly_make_axis()

        polynomial.poly_main(a)
    if y.lower() in ["6", "cycloid"]:
        cycloid.cycloid()


    if y.lower() in ["7", "hypocycloid"]:
        k = var
        hypocycloid.hypo(k)


    if y.lower() in ["8", "epicycloid"]:
        k = var
        epicycloid.epicycloid(k)
while True:
    # Asking to continue with the program
    v = raw_input("Do you want to plot new graph or from last function plotted -\n1.New Graph\n2.Previous
Graph\n3.Exit\n")
    turtle.clear()
    turtle.penup()
    turtle.goto(0,0)
    turtle.pendown()
    # Allowing multiple possible inputs to be accepted
    if v.lower() in ["new graph", "new", "1"]:
        # y,var are the variables with the information regarding the graph needed to reproduce it
        (y,var) = new()
        # Checking if Graphs folder already there
        if not os.path.exists(".\\Graphs"):
            # Making said folder if not found
            os.mkdir(".\\Graphs")
        # Using a generator function to get a list of files in Graphs folder
        (root,dir,files) = os.walk(".\\Graphs").next()
        # Calculating the name of the new file
        filenum = len(files)
        # Opening a new file with .graph extension which is self created but can still be opened in binary mode
        graph = file(".\\Graphs\\"+str(filenum+1)+".graph", "wb")
        # Filling the file with required data
        pickle.dump((y,var), graph)
        # Remember to close to make sure everything functions smoothly!
        graph.close()
    # Again allowing multitudes of possible inputs
    elif v.lower() in ["2", "previous", "previous graph"]:
        # Getting the filelist
        (root, dir, files) = os.walk(".\\Graphs").next()
        # Sorting so as to be able to determine the last file
        files.sort()
        # Opening the file
        previous = file(".\\Graphs\\" + files[-1], "rb")
        # Getting the critical variables required to make graph
        (y,var) = pickle.load(previous)
        # Closing to ensure smooth functioning
        previous.close()
        # Deleting opened graph
        os.remove(".\\Graphs\\"+files[-1])
        # Sending acquired graph information for processing
        prev(y, var)


    elif v.lower() in ["3", "exit"]:
        turtle.exitonclick()
        exit()
    else:
        print "Make an appropriate choice"
        print "The program now closes"
        time.sleep(1)
        exit()
turtle.exitonclick
```

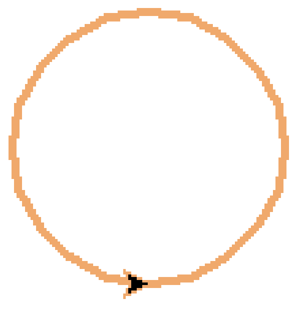
OUTPUT



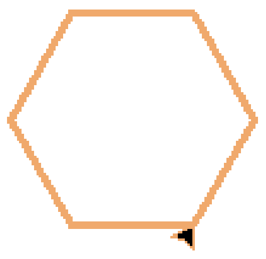
Graph of Cos function



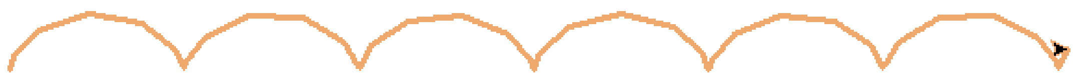
Graph of Sin Function



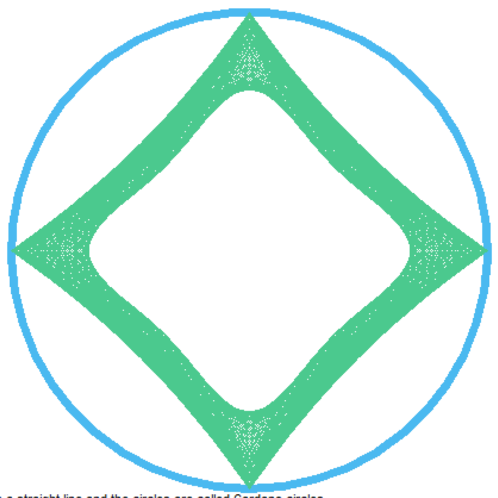
Circle



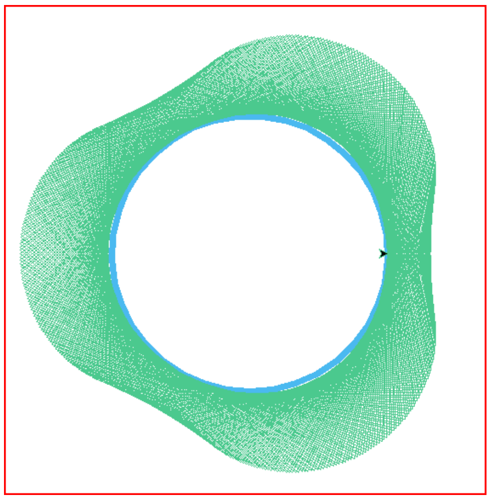
Hexagon



Cycloid



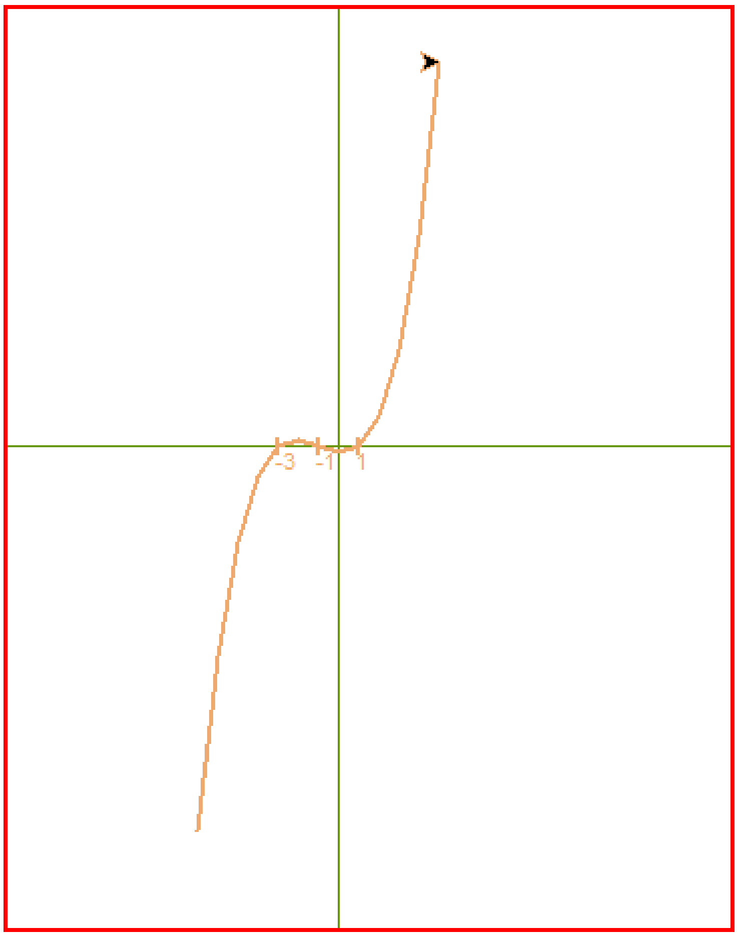
Quick Fact:
for k=2 the curve is a straight line and the circles are called Cardano circles.
They are used in high-speed printing



Hypocycloid with cusps 4

Epicycloid with cusps3

Graph of a cubic polynomial



LIMITATIONS AND SCOPE FOR FUTURE EXPANSION

1. Function with asymptote (like tangent, sec, cosec) can’t be drawn
2. Combination of function can’t be drawn and the library of function is limited to 7
3. The graph can’t be resized according to the size of graph window
4. Graph can’t be resized according to the needs
5. Domain can’t be taken as input from user.

BIBLIOGRAPHY

1. Computer science with Python – Sumita Arora
2. Together with computer science (Python)
3. www.Wikipedia.org
4. docs.python.org
5. www.interactivepython.org