```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 28 09:33:30 2023

@author: holmesfamily
"""
import numpy as np
import turtle as t
import turtle
from math import sqrt,cos,pi

totalLength = 0

#setup Turtles for graphic, text, and title
t2 = turtle.Turtle()
t2.hideturtle()
t.title("12 inch LP Record")
t.reset()
t.setup (width=1200, height=1200)


#set up measurements
inch = 72 #72dpi
t.tracer(50,1)
radius = 6 * inch
spindleRadius = 10

labelsize = 1.875 * inch
track1 = round(5.75 * inch, 0)
track2 = round(5.45 * inch, 0)
track3 = round(5.15 * inch, 0)
track4 = round(4.75 * inch, 0)
track5 = round(4.45 * inch, 0)
track6 = round(4.0 * inch, 0)
track7 = round(3.25 * inch, 0)
leadout = round(2.75 * inch, 0)
print("track1 ", track1)
print("track2 ", track2)
print("track3 ", track3)
print("track4 ", track4)
print("track5 ", track5)
print("track6 ", track6)
print("track7 ", track7)

"""Archimedean spiral analysis"""
def opposite(a,b,C):
    """Returns the side opposite the given angle in
       a triangle using the Law of Cosines
       Enter side, side, angle"""
    c = sqrt(a**2 + b**2 - 2*a*b*cos(C))
```

```
    return c


def spiral(r,a,b,step=0.0001):
    """Returns length of spiral r from
       a to b taking given step size"""
    length = 0
    theta = a
    while theta < b:
        length += opposite(r(theta),r(theta+step),step)
        theta += step
    return length

def r(theta):
    return (2.75 + 0.12892*theta)

"""label text mouse click"""
def clearFXN():
    t2.clear()

def fxn(x, y):
    r = x/72
    circumference = round(2*np.pi*r, 2)
    perInch = circumference * 0.555555555555556

    t2.penup()
    t2.clear()
    t2.home()
    t2.pencolor("gold1")
    t2.sety(200)
    t2.setx(0)
    str1 = ("At this groove the needle travels %g inches per rotation" %
     (circumference))
    str2 = ("The needle travels %g inches per second" % (perInch))
    t2.write(str(str1), align="center",
                 font=("Arial",
                       24, "normal"))

    t2.sety(170)
    t2.write(str(str2), align="center",
                 font=("Arial",
                       24, "normal"))
    turtle.ontimer(clearFXN, t = 5000)

""" Record label stats """
def recordStats():
    t.penup()
    t.home()
    t.sety(labelsize* + 0.15)
    t.pencolor("black")
    str3 = ("Total turns = %g" % (counter2))
```

```
    str4 = ("Groove length = %g inches" % (totalLength))
    str5 = ("%g feet" % (totalLength / 12))
    str6 = ("%g yards" % (totalLength / 36))
    str7 = ("%g miles" % (totalLength / 63360))
    t.write(str(str3), align="center",
                  font=("Arial",
                        18, "normal"))
    t.home()
    t.sety(labelsize* - 0.30)
    t.write(str(str4), align="center",
                  font=("Arial",
                        18, "normal"))
    t.sety(labelsize* - 0.45)
    t.write(str(str5), align="center",
                  font=("Arial",
                        18, "normal"))
    t.sety(labelsize* - 0.60)
    t.write(str(str6), align="center",
                  font=("Arial",
                        18, "normal"))
    t.sety(labelsize* - 0.75)
    t.write(str(str7), align="center",
                  font=("Arial",
                        18, "normal"))
    t.penup()

# set turtlr screen
sc = t.Screen()

# call method on screen click
t.onscreenclick(fxn)



"""make the grid lines and measurements"""
#make x grid
t.home()
for i in range(-7, 8):
    t.penup()
    t.setx(i * inch)
    t.sety(7 * inch)
    t.pencolor("red")
    if i != -7:
        t.pendown()
        t.sety(-6.5 * inch)
    t.penup()
    t.sety(-7 * inch)
    t.pendown()
    t.write(str(i), align="center",
              font=("Arial",
                    24, "normal"))
```

```
    t.penup()

#make y grid
t.home()
for i in range(-7, 8):
    t.penup()
    t.sety(i * inch)
    t.setx(7 * inch)
    t.pencolor("red")
    if i != -7:
        t.pendown()
        t.setx(-6.5 * inch)
        t.penup()
        t.setx(-7 * inch)
        t.sety(i * inch -12)
        t.write(str(i), align="center",
              font=("Arial",
                    24, "normal"))
        t.penup()


#make record speed statement
t.penup()
t.home()
t.pencolor("gray0")
t.sety(7.05*inch)
t.setx(-0.5 * inch)
t.write(str("At 331/3 RPM a record makes one full rotation every 1.80000018...
 seconds"), align="center",
              font=("Arial",
                    24, "normal"))



#make record
t.penup()
t.home()
t.pencolor("gray0")
t.sety(-6 *inch)
t.pensize(6)
t.pendown()
t.fillcolor("black")
t.begin_fill()
t.circle(6 * inch, 360)
t.end_fill()

t.penup()
t.home()
t.sety(labelsize*.75)
t.pencolor("black")
t.write(str("Record Statistics"), align="center",
```

```
              font=("Arial",
                    24, "normal"))
t.penup()
t.home()
t.sety(-labelsize)
t.fillcolor("gold1")
t.begin_fill()
t.circle(labelsize, 360)
t.end_fill()
t.penup()
t.home()
t.sety(labelsize*.5)
t.pencolor("black")
t.write(str("Record Statistics"), align="center",
              font=("Arial",
                    24, "normal"))
t.penup()

#make Spindle hole
t.penup()
t.home()
t.pensize(1)
t.pencolor("black")
t.fillcolor("white")
t.begin_fill()
t.sety(-spindleRadius)

t.pendown()
t.circle(spindleRadius, 360)
t.end_fill()

t.penup()
t.pensize(1)
t.home()
t.pencolor("red")
t.setx(-spindleRadius)
t.pendown()
t.setx(spindleRadius)
t.penup()
t.home()
t.sety(-spindleRadius)
t.pendown()
t.sety(spindleRadius)


#make record grooves
t.penup()
t.home()
t.pencolor("gray90")
t.sety(-6 * inch)
t.pendown()
```

```
counter = 0
counter2 = 0
i = 6 * inch
n = .3
while i > 2.0 * inch:
 counter += 20
 if counter % 360 == 0:
    counter2 += 1
    #print("rotations = ", counter2)
 t.circle(i, 20)
 x = round(i,0)
 #lead in
 if x == round(5.8 * inch, 1):
     n =0.3

#first track
 if x == track1:
     n = 0.1
#track 2 and lead in
 if x == track2 + 5:
     n = 0.25
 if x == track2:
     n = 0.1
 #track 3 and lead in
 if x == track3 + 5:
     n = 0.25
 if x == track3:
     n = 0.1
#track 4 and lead in
 if x == round(track4 + 5, 1):
     n = 0.25
 if x == round(track4, 1):
     n = 0.1
#track 5 and lead in
 if x == round(track5 + 5, 1):
     n = 0.25
 if x == round(track5, 1):
     n = 0.1
 #track 6 and lead in
 if x == round(track6 + 5, 1):
     n = 0.25
 if x == round(track6, 1):
     n = 0.1
#track 7 and lead in
 if x == round(track7 + 5, 1):
     n = 0.25
 if x == round(track7, 1):
     n = 0.1
#leadout and end loop
 if x == round(leadout, 1):
```

```
    n = 0.5
 i -= n

t.circle(i, 560)

#find total length of groove
concenticRadius = np.linspace(5.7, 2.75, counter2)

for i in range(0,len(concenticRadius)):
        totalLength = totalLength + (concenticRadius[i]*2*np.pi)

recordStats()

grooveLenght = spiral(r,0,2*pi*(counter2 + 1))

programmingNotes = ("""
***----------------------programming notes-----------------------------***
The total distance of the groove was calculated in a number of ways.
I decided to use the Concentric Radius method for label information.

An average record is 800 turns and about 0.335 miles in groove length
 (according
to some sources on the internet).  And my concentric radius module corroborates
those findings.

I consistently overshot that mark with my graphic depiction of a record
using an Archimedean spiral analysis.

The archimedean spiral analysis is %g inches.
Concentric Radius analysis is %g inches.
***--------------------------------------------------------------------***
                    """ % (grooveLenght, totalLength))
print(programmingNotes)
print("count total ",totalLength)



print("counter2 =", counter2)
print(spiral(r,0,2*pi*(counter2)))

#t.update()
t.done()
```