```
from tkinter import *
from tkinter import Tk, Button
from tkinter import messagebox
from
tkinter import font
import pygame
import tkinter as tk
from PIL import Image,
ImageTk
import tkinter.font as font
import math

window = Tk()
myfont = font.Font(family =
'Monaco', weight = 'bold')
img = Image.open(r'C:\Users\savya\Downloads\bg1.png')
bg =
ImageTk.PhotoImage(img)
label = Label(window, image=bg)
label.place(x = 0,y =
0)
pygame.mixer.init()

class mywindow:
    def __init__(self,win):
        self.win =
win
        self.win.title("Mable Game")
        self.frame = tk.Frame(win)

 self.back_img = tk.PhotoImage(file=r'C:\Users\savya\Downloads\bg1.png')

self.background_label = tk.Label(self.win, image=self.back_img)

self.background_label.place(relwidth=1, relheight=1)
        self.b1 = Button(win,text =
"Start ", bg = "black", fg = "white", command = self.dum)

 self.b2 = Button(win,text = " Quit ", bg = "black", fg =
"white", command = window.quit)
        self.b3 = Button(win,text = "Music:ON
", bg = "black", fg = "white", command = self.play_sound)

self.b4 = Button(win,text = "Music:OFF", bg = "black", fg = 'white',
command = self.stop_sound)
        self.b5 = Button(win,text = "Instructions", bg =
"black", fg = "white", command = self.display_instructions)

self.b1.place(x = 155,y = 150)
        self.b2.place(x = 155,y = 200)
        self.b3.place(x
= 60 ,y = 250)
        self.b4.place(x = 200,y = 250)
        self.b5.place(x = 135,y =
300)
        self.b1['font'] = myfont
        self.b2['font'] = myfont

self.b3['font'] = myfont
        self.b4['font'] = myfont
        self.b5['font'] = myfont

  def play_sound(self):

pygame.mixer.music.load(r"C:\Users\savya\Downloads\stranger-things-124008.mp3")

       pygame.mixer.music.play()
    def stop_sound(self):

pygame.mixer.music.stop()
    def display_instructions(self):

messagebox.showinfo("Instructions", "(1) Ensure that number of marbles are
always less than number of marble places \n (2) Ensure to update the skip length value of
```

```
marbles before going to next move. \n (3) The number of marbles left gives you the final
answer")
    def dum(self):
        self.win.title("Input")

self.frame = tk.Frame(self.win)
        self.back_img =
Image.open(r"C:\Users\savya\OneDrive\Pictures\hVyvpN1.jpg")
        self.bg =
ImageTk.PhotoImage(self.back_img)
        label = Label(window, image=self.bg)

label.place(x = 0,y = 0)
        start_window = tk.Toplevel(self.win)

start(start_window)

class start:
    def __init__(self, win):
        self.win = win


        lbl = Label(window, text="Enter marble data.\n E.g 3 5 1 2 3 5 2 1 \n Where
the first number 3 represents the number of marbles \n the second number 5 represents the
number of cells \n 1, 3, 2 are the initial places of marbles 1, 2, 3 respectivly, \n and 2, 5,
1 are the skip length of the respecitive marbles.", fg="white",
bg="black", font="Helvetica")
        lbl.place(x = 500, y = 250)

lbl1 = Label(window, text="Enter marble information",
fg="white",bg="black", font=("Helvetica",15))

lbl1.place(x=500, y=450)
        self.ip = tk.StringVar()
        self.t1 = Entry(window,
bg="white", fg="black", bd=5, textvariable=self.ip)

self.t1.place(x = 750, y = 450)
        bt1 = Button(window, text="Run Game",
command = self.ip_board)
        bt1.place(x=750, y = 500)
        self.marbles = []

self.bt2=Button()



    def ip_board(self):

self.win.title("Marble Game")
        ip = self.ip.get()
        print(type(ip))

      image_path = r"C:\Users\savya\OneDrive\Pictures\neon wallpaper 1.jpg"

image = Image.open(image_path)
        background_image = ImageTk.PhotoImage(image)

canvas = tk.Canvas(self.win, width=1550, height=1000, bg="white")

canvas.pack()
        canvas.create_image(0, 0, image=background_image, anchor=tk.NW)


        ip = ip.split()
        nm = int(ip[0])
        num_parts = int(ip[1])

   parts_list = []
        m = []
        sl = []

        for i in range(0, nm):

   m.append(i+1)
```

```
        for i in range(2, len(ip), 2):

parts_list.append(int(ip[i]))
        d = dict(zip(m,parts_list))
        print(d)

for i in range(3, len(ip), 2):
            sl.append(int(ip[i]))
        print(sl)

print(parts_list)

        #3 5 2 1 5 3 4 3

        center_x, center_y =
750, 375
        inner_radius = 150
        outer_radius = 350

        #Outer circle

       canvas.create_oval(center_x - outer_radius, center_y - outer_radius,

         center_x + outer_radius, center_y + outer_radius, outline="black",
width=5)
        #Inner circle
        canvas.create_oval(center_x - inner_radius, center_y -
inner_radius,
                           center_x + inner_radius, center_y + inner_radius,
outline="black", width=5)
        angle_increment = 360/ num_parts
        ifont =
font.Font(size = 16)
        for i in range(num_parts):
            angle_rad =
math.radians(i * angle_increment)
            x1 = center_x + inner_radius *
math.cos(angle_rad)
            y1 = center_y - inner_radius * math.sin(angle_rad)

 x2 = center_x + outer_radius * math.cos(angle_rad)
            y2 = center_y - outer_radius *
math.sin(angle_rad)
            #divide section between concentric circles

canvas.create_line(x1, y1, x2, y2, fill="black", width=5)
            #placing
numbers at the midpoints of each section
            mid_angle_rad = math.radians((i + 1.5) *
angle_increment)
            mid_radius = (outer_radius + inner_radius) / 2
            x =
center_x + (outer_radius - 10) * math.cos(mid_angle_rad)
            y = center_y -
(outer_radius - 10) * math.sin(mid_angle_rad)
            canvas.create_text(x, y, text=str(i
+ 1),font = ifont, fill="black")

        mfont = font.Font(size = 16,weight =
"bold")
        angle_increment = 360 / num_parts
        for part in parts_list:

           desired_part = part % num_parts
            if desired_part < 0 or desired_part
>= num_parts:
                continue
            start_angle = desired_part *
angle_increment
            end_angle = (desired_part + 1) * angle_increment

midpoint_angle = (start_angle + end_angle) / 2
            angle_rad =
math.radians(midpoint_angle)
            x = center_x + (outer_radius + inner_radius) * 0.5 *
```

```
math.cos(angle_rad)
            y = center_y - (outer_radius + inner_radius) * 0.5 *
math.sin(angle_rad)
            marble_radius = 25
            canvas.create_oval(x -
marble_radius, y - marble_radius,
                                    x + marble_radius, y +
marble_radius, fill='blue')
            canvas.create_text(x, y,
text=(parts_list.index(part)+1),font = mfont, fill = "white")

#initialvalue:skiplength
        angle = 0
        marble_radius = 15

self.bt2 = Button(self.win, text="Output", command=self.out)

self.bt2.place(x=300, y=350)

    def out(self):
        self.win.title("Marble
Game")
        self.frame = tk.Frame(self.win)
        self.back_img =
Image.open(r"C:\Users\savya\OneDrive\Pictures\neon wallpaper 1.jpg")
        self.bg
= ImageTk.PhotoImage(self.back_img)
        label = Label(window, image=self.bg)

label.place(x = 0,y = 0)
        output_window = tk.Toplevel(self.win)

output_instance = output_class(output_window, self.ip.get())

class output_class(start):

    def __init__(self,win,ip):
        super().__init__(win)
        self.win = win

  self.win.title("Marble Game")
        image_path =
r"C:\Users\savya\OneDrive\Pictures\neon wallpaper 1.jpg"
        image =
Image.open(image_path)
        background_image = ImageTk.PhotoImage(image)
        canvas =
tk.Canvas(self.win, width=1550, height=1000, bg="white")
        canvas.pack()

   canvas.create_image(0, 0, image=background_image, anchor=tk.NW)



center_x, center_y = 750 , 375
        inner_radius = 150
        outer_radius = 350

  ip = ip.split()
        nm = int(ip[0])
        num_parts = int(ip[1])    #it is used to
take the input from the user and use it to place the marbles
        parts_list = []

m = []
        sl = []
        #
        for i in range(0, nm):
            m.append(i+1)

       for i in range(2, len(ip), 2):
            parts_list.append(int(ip[i]))
        d =
dict(zip(m,parts_list))
        print(d)
```

```
        for i in range(3, len(ip), 2):

sl.append(int(ip[i]))
        print(sl)
        print(parts_list)
        #Outer circle

    canvas.create_oval(center_x - outer_radius, center_y - outer_radius,

      center_x + outer_radius, center_y + outer_radius, outline="black", width=5)

     #Inner circle
        canvas.create_oval(center_x - inner_radius, center_y -
inner_radius,
                           center_x + inner_radius, center_y + inner_radius,
outline="black", width=5)
        angle_increment = 360/ num_parts

        for i
in range(num_parts):
            angle_rad = math.radians(i * angle_increment)
            x1
= center_x + inner_radius * math.cos(angle_rad)
            y1 = center_y - inner_radius *
math.sin(angle_rad)
            x2 = center_x + outer_radius * math.cos(angle_rad)

 y2 = center_y - outer_radius * math.sin(angle_rad)
            #divide section between
concentric circles
            canvas.create_line(x1, y1, x2, y2, fill="black",
width=5)
            #placing numbers at the midpoints of each section

mid_angle_rad = math.radians((i + 1.5) * angle_increment)
            mid_radius =
(outer_radius + inner_radius) / 2
            x = center_x + (outer_radius - 10) *
math.cos(mid_angle_rad)
            y = center_y - (outer_radius - 10) *
math.sin(mid_angle_rad)
            canvas.create_text(x, y, text=str(i + 1),
fill="black", font=5)

        for i in range(1, nm+1):
            x = d.get(i)

           d[i] = x + sl[i-1]
            if d[i] > num_parts:
                d[i] = d[i]
- num_parts

        lt = list(d.values())
        x = max(lt, key = lt.count)


  while x in lt:
            lt.remove(x)
        lt1 = list(d.values())
        op =
[len(lt)]
        marbles=[]

        for i in range(0, len(lt)):
            x =
lt1.index(lt[i])
            op.append(x + 1)
            marbles.append(x+1)
        output
= ' '.join(map(str, op))
        print("Output: \n", output)
        print(op)

    print(marbles)
        mfont = font.Font(size = 16,weight = "bold")
```

```
angle_increment = 360 / num_parts
        for part in marbles:
            desired_part =
part % num_parts
            if desired_part < 0 or desired_part >= num_parts:

       continue
            start_angle = desired_part * angle_increment

end_angle = (desired_part + 1) * angle_increment
            midpoint_angle = (start_angle +
end_angle) / 2
            angle_rad = math.radians(midpoint_angle)
            x =
center_x + (outer_radius + inner_radius) * 0.5 * math.cos(angle_rad)
            y = center_y
- (outer_radius + inner_radius) * 0.5 * math.sin(angle_rad)
            marble_radius = 25

         canvas.create_oval(x - marble_radius, y - marble_radius,

      x + marble_radius, y + marble_radius, fill='blue')
            canvas.create_text(x, y,
text=(part),font = mfont, fill = "white")


mywin =
mywindow(window)
window.title("Marble
Game")
window.geometry("425x600+30+30")
window.mainloop()
```