**CSE(C) GROUPS for PYTHON PROJECT**

|  |  |
| --- | --- |
| **ROLL NUMBERS** | **QUESTION NUMBERS** |
| C1-D0 | 1,3 |
| D1-E0 | 7,6 |
| E1-F0 | 5,4 |
| F1-G0 | 2,1 |
| G1-H0 | 4,3 |
| H1-J0 | 6,5 |

Project Lists and their outputs example:

**1. Draw Colorful Spiral Web Using Turtle Graphics in Python.**

**2. Draw Spiralling Circles Using Turtle Graphics in Python.**

**3. Create a Snake-Game using Turtle in Python.**

**4. Draw Rainbow using Turtle Graphics in Python**

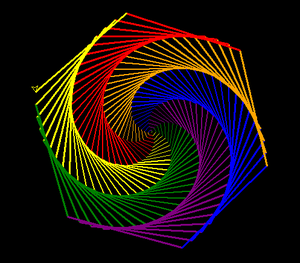
**5.BMI Calculator**

6. **Fidget Spinner Game**

**7. Write a python code to create GUI calendar**

|  |
| --- |
| 1. Draw Colorful Spiral Web Using Turtle Graphics in Python. # import turtle  import turtle    # defining colors  colors = ['red', 'yellow', 'green', 'purple', 'blue', 'orange']    # setup turtle pen  t= turtle.Pen()    # changes the speed of the turtle  t.speed(10)    # changes the background color  turtle.bgcolor("black")    # make spiral\_web  for x in range(200):      t.pencolor(colors[x%6]) # setting color      t.width(x/100 + 1) # setting width      t.forward(x) # moving forward      t.left(59) # moving left    turtle.done()  t.speed(10)    turtle.bgcolor("black") # changes the background color    # make spiral\_web  for x in range(200):      t.pencolor(colors[x%6]) # setting color      t.width(x/100 + 1) # setting width      t.forward(x) # moving forward      t.left(59) # moving left    turtle.done() |

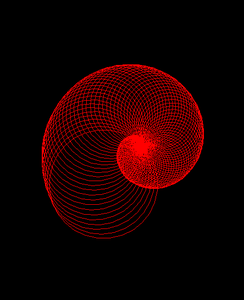
**Output:**



# 2. Draw Spiralling Circles Using Turtle Graphics in Python.

|  |
| --- |
| # importing turtle  import turtle    # initialise the turtle instance  animation = turtle.Turtle()    #creating animation  # changes speed of turtle  animation.speed(0)    # hiding turtle  animation.hideturtle()    # changing background color  animation.getscreen().bgcolor("black")    # color of the animation  animation.color("red")    for i in range(100):        # drawing circle using circle function      # by passing radius i      animation.circle(i)        # changing turtle face by 5 degree from it's      # previous position after drawing a circle      animation.\_rotate(5) |

**Output:**



# 3. Create a Snake-Game using Turtle in Python.

|  |
| --- |
| # import required modules  import turtle  import time  import random    delay = 0.1  score = 0  high\_score = 0      # Creating a window screen  wn = turtle.Screen()  wn.title("Snake Game")  wn.bgcolor("blue")  # the width and height can be put as user's choice  wn.setup(width=600, height=600)  wn.tracer(0)    # head of the snake  head = turtle.Turtle()  head.shape("square")  head.color("white")  head.penup()  head.goto(0, 0)  head.direction = "Stop"    # food in the game  food = turtle.Turtle()  colors = random.choice(['red', 'green', 'black'])  shapes = random.choice(['square', 'triangle', 'circle'])  food.speed(0)  food.shape(shapes)  food.color(colors)  food.penup()  food.goto(0, 100)    pen = turtle.Turtle()  pen.speed(0)  pen.shape("square")  pen.color("white")  pen.penup()  pen.hideturtle()  pen.goto(0, 250)  pen.write("Score : 0  High Score : 0", align="center",            font=("candara", 24, "bold"))      # assigning key directions  def group():      if head.direction != "down":          head.direction = "up"      def godown():      if head.direction != "up":          head.direction = "down"      def goleft():      if head.direction != "right":          head.direction = "left"      def goright():      if head.direction != "left":          head.direction = "right"      def move():      if head.direction == "up":          y = head.ycor()          head.sety(y+20)      if head.direction == "down":          y = head.ycor()          head.sety(y-20)      if head.direction == "left":          x = head.xcor()          head.setx(x-20)      if head.direction == "right":          x = head.xcor()          head.setx(x+20)      wn.listen()  wn.onkeypress(group, "w")  wn.onkeypress(godown, "s")  wn.onkeypress(goleft, "a")  wn.onkeypress(goright, "d")    segments = []      # Main Gameplay  while True:      wn.update()      if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -290:          time.sleep(1)          head.goto(0, 0)          head.direction = "Stop"          colors = random.choice(['red', 'blue', 'green'])          shapes = random.choice(['square', 'circle'])          for segment in segments:              segment.goto(1000, 1000)          segments.clear()          score = 0          delay = 0.1          pen.clear()          pen.write("Score : {} High Score : {} ".format(              score, high\_score), align="center", font=("candara", 24, "bold"))      if head.distance(food) < 20:          x = random.randint(-270, 270)          y = random.randint(-270, 270)          food.goto(x, y)            # Adding segment          new\_segment = turtle.Turtle()          new\_segment.speed(0)          new\_segment.shape("square")          new\_segment.color("orange")  # tail colour          new\_segment.penup()          segments.append(new\_segment)          delay -= 0.001          score += 10          if score > high\_score:              high\_score = score          pen.clear()          pen.write("Score : {} High Score : {} ".format(              score, high\_score), align="center", font=("candara", 24, "bold"))      # Checking for head collisions with body segments      for index in range(len(segments)-1, 0, -1):          x = segments[index-1].xcor()          y = segments[index-1].ycor()          segments[index].goto(x, y)      if len(segments) > 0:          x = head.xcor()          y = head.ycor()          segments[0].goto(x, y)      move()      for segment in segments:          if segment.distance(head) < 20:              time.sleep(1)              head.goto(0, 0)              head.direction = "stop"              colors = random.choice(['red', 'blue', 'green'])              shapes = random.choice(['square', 'circle'])              for segment in segments:                  segment.goto(1000, 1000)              segments.clear()                score = 0              delay = 0.1              pen.clear()              pen.write("Score : {} High Score : {} ".format(                  score, high\_score), align="center", font=("candara", 24, "bold"))      time.sleep(delay)    wn.mainloop() |

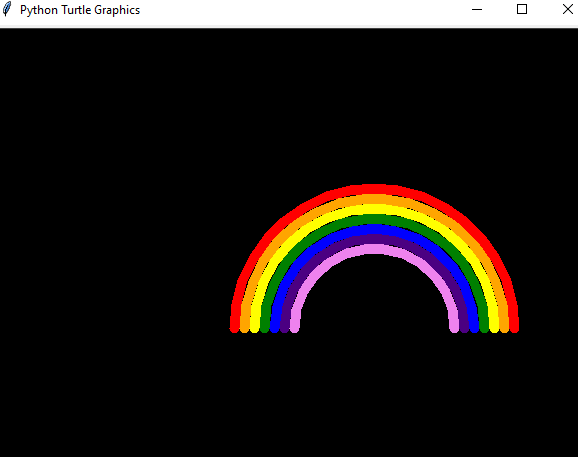
**Output:**



# 4. Draw Rainbow using Turtle Graphics in Python

|  |
| --- |
| # Import turtle package  import turtle    # Creating a turtle screen object  sc = turtle.Screen()    # Creating a turtle object(pen)  pen = turtle.Turtle()    # Defining a method to form a semicircle  # with a dynamic radius and color  def semi\_circle(col, rad, val):        # Set the fill color of the semicircle      pen.color(col)        # Draw a circle      pen.circle(rad, -180)        # Move the turtle to air      pen.up()        # Move the turtle to a given position      pen.setpos(val, 0)        # Move the turtle to the ground      pen.down()        pen.right(180)      # Set the colors for drawing  col = ['violet', 'indigo', 'blue',         'green', 'yellow', 'orange', 'red']    # Setup the screen features  sc.setup(600, 600)    # Set the screen color to black  sc.bgcolor('black')    # Setup the turtle features  pen.right(90)  pen.width(10)  pen.speed(7)    # Loop to draw 7 semicircles  for i in range(7):      semi\_circle(col[i], 10\*(        i + 8), -10\*(i + 1))    # Hide the turtle  pen.hideturtle() |

**Output:**



5.BMI Calculator

Height=float(input("Enter your height in centimeters: "))

Weight=float(input("Enter your Weight in Kg: "))

Height = Height/100

BMI=Weight/(Height\*Height)

print("your Body Mass Index is: ",BMI)

if(BMI>0):

if(BMI<=16):

print("you are severely underweight")

elif(BMI<=18.5):

print("you are underweight")

elif(BMI<=25):

print("you are Healthy")

elif(BMI<=30):

print("you are overweight")

else: print("you are severely overweight")

else:("enter valid details")

OUTPUT:

Enter your height in centimeters: 162

Enter your Weight in Kg: 55

your Body Mass Index is: 20.957171162932475

you are Healthy

6. **Fidget Spinner Game**

from turtle import \*

state = {'turn': 0}

def spinner():

clear()

angle = state['turn']/10

right(angle)

forward(100)

dot(120, 'red')

back(100)

right(120)

forward(100)

dot(120, 'green')

back(100)

right(120)

forward(100)

dot(120, 'blue')

back(100)

right(120)

update()

def animate():

if state['turn']>0:

state['turn']-=1

spinner()

ontimer(animate, 20)

def flick():

state['turn']+=10

setup(420, 420, 370, 0)

hideturtle()

tracer(False)

width(20)

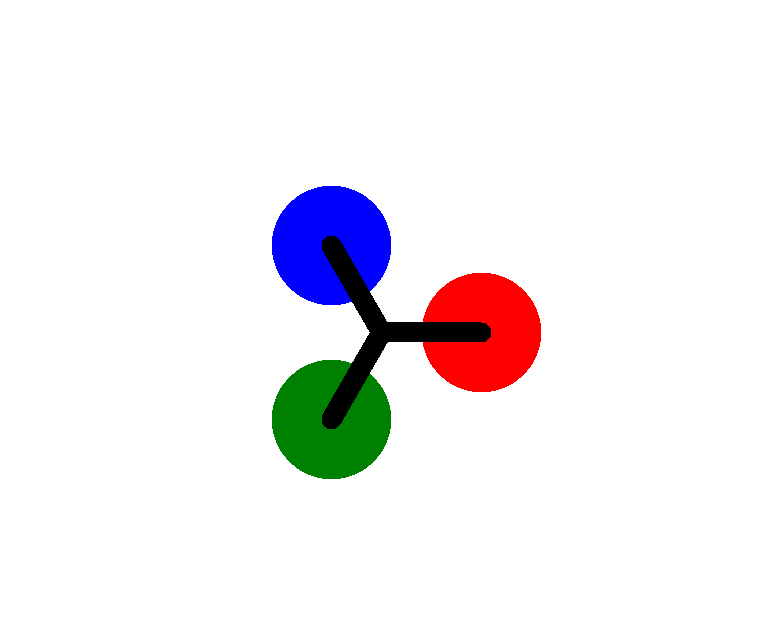
onkey(flick, 'space')

listen()

animate()

done()

**OUTPUT:**



**7. Write a python code to create GUI calendar**

#Importing tkinter module

from tkinter import \*

#importing calendar module

import calendar

#function to show calendar of the given year

def showCalender():

    gui = Tk()

    gui.config(background='grey')

    gui.title("Calender for the year")

    gui.geometry("550x600")

    year = int(year\_field.get())

    gui\_content= calendar.calendar(year)

    calYear = Label(gui, text= gui\_content, font= "Consolas 10 bold")

    calYear.grid(row=5, column=1,padx=20)

    gui.mainloop()

new = Tk()

new.config(background='grey')

new.title("Calender")

new.geometry("250x140")

cal = Label(new, text="Calender",bg='grey',font=("times", 28, "bold"))

year = Label(new, text="Enter year", bg='dark grey')

year\_field=Entry(new)

button = Button(new, text='Show Calender',fg='Black',bg='Blue',command=showCalender)

#putting widgets in position

cal.grid(row=1, column=1)

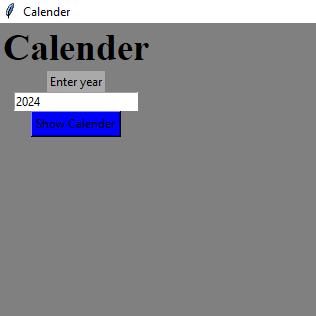
year.grid(row=2, column=1)

year\_field.grid(row=3, column=1)

button.grid(row=4, column=1)

new.mainloop()

**Output:**

****

****