**CHAPTER PARTICULARS PAGE NO**

1. INTRODUCTION 03
2. OBJECTIVE 04
3. LANGUAGE SPECIFICATION 05
   1. TURTLE PROGRAMMING

IN PYTHON

* 1. TURTLE AND CROSSPONDING

FUNCTION

1. SYSTEM SPECIFICATION 11
2. IMPLEMENTATION 12
3. RESULT 17
4. CONCLUSION 20
5. REFERENCES 21

CHAPTER 1

INTRODUCTION

Python is a popular programming language. It was created by guido van Rossum, and released in 1991.

It is used for:

* web development
* software development
* mathematics
* system scripting

Python works on different platforms. It has a simplex to the English language . python has syntax that allows developers to write programs with fewer lines than some other programming language.

Pythons runs on an interpreter system, meaning that code can be executed as soon as it is written . This means that prototyping can be very quick. Pyton can be treated in a procedural way, an object-oriented way or a functional way.

Python was designed for readability, and has some similarities to the English language with influences from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parenthesis.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

CHAPTER 2

OBJECTIVES

The main objective is to understand the basics concepts of Python, build basic programs using fundamental programming constructs like variables, conditional logic, looping ,and functions. Work with user input to create fun and interactive programs.

CHAPTER 3

LANGUAGE SPECIFICATION

**3.1 TURTLE PROGRAMMING IN PYTHON**

“Turtle” is a python feature like a drawing board, which lets us command a turtle to draw all over it. To make use of the turtle methods and functionalities, we need to import turtle .”turtle” comes packed with the standard python packages and need not to be installed externally. The roadmap for executing a turtle program follows four steps:

1. Import the turtle module
2. Create a turtle to control
3. Draw around using the turtle methods
4. Run turtle done().

we can use the function like turtle.forward(…) and turtle.right(…)which can move the turtle around. Commonly used turtle methods are:

* Turtle() : creates and returns a new turtle object.
* Forward() : forward by the specified amount .
* Backward() : moves the turtle backward by the specified amount.
* Right() : turns the turtle clockwise
* Left() : turns the turtle counter clockwise
* Penup() : picks up the turtle pen.
* Pendown() : puts down the turtle pen.
* Up() : picks up the turtle pen.
* Down() : puts down the turtle pen.
* Color() : changes the color of the turtle pen
* Fillcolor() : the turtle will use the fill a polygon.
* Heading() : returns the current heading.
* Position() : returns the current position.
* Goto() : move the turtle to the position x,y.
* begin\_fill() : remember the starting point for a filled polygon.
* Dot() : leaves the dot at the current position.
* Stamp() : leaves an impression of a turtle shape at the current location.
* Shape() : should be ‘arrow’ ,‘classic’,’turtle’ or ‘circle’

3.2 Turtle and corresponding functions

Most of the examples in this section refer to a Turtle instance called turtle.

### Turtle motion

turtle. **forward**(*distance*)

Move the turtle forward by the specified *distance*, in the direction the turtle is headed.

**Parameters: distance** – a number (integer or float)

Example

**>>>** turtle.position()

(0.00,0.00)

**>>>** turtle.forward(25)

**>>>** turtle.position()

(25.00,0.00)

**>>>** turtle.forward(-75)

**>>>** turtle.position()

(-50.00,0.00)

turtle.**right**(*angle*)

turtle.**rt**(*angle*)

**Parameters: angle** – a number (integer or float)

Turn turtle right by *angle* units.(Unit can be [degrees()](https://docs.python.org/3/library/turtle.html#turtle.degrees) and [radians()](https://docs.python.org/3/library/turtle.html#turtle.radians) functions.) Angle orientation depends on the turtle mode, see [mode()](https://docs.python.org/3/library/turtle.html#turtle.mode).

Example

**>>>** turtle.heading()

22.0

**>>>** turtle.right(45)

**>>>** turtle.heading()

337.0

turtle.**goto**(*x*, *y=None*)

turtle.**setpos**(*x*, *y=None*)

turtle.**setposition**(*x*, *y=None*)

**Parameters: x** – a number or a pair/vector of numbers

**y** – a number or None

If *y* is None, *x* must be a pair of coordinates or a [Vec2D](https://docs.python.org/3/library/turtle.html#turtle.Vec2D) (e.g. as returned by [pos()](https://docs.python.org/3/library/turtle.html" \l "turtle.pos)).

Move turtle to an absolute position. If the pen is down, draw line. Do not change the turtle’s orientation.

Example

>>> tp = turtle.pos()

>>> tp

(0.00,0.00)

>>> turtle.setpos(60,30)

>>> turtle.pos()

(60.00,30.00)

>>> turtle.setpos((20,80))

>>> turtle.pos()

(20.00,80.00)

>>> turtle.setpos(tp)

>>> turtle.pos()

(0.00,0.00)

turtle.**setx**(*x*)

**Parameter: x** – a number (integer or float)

Set the turtle’s first coordinate to *x*, leave second coordinate unchanged.

> Example>>

**>>>** turtle.position()

(0.00,240.00)

**>>>** turtle.setx(10)

**>>>** turtle.position()

(10.00,240.00)

### Turtle’s state

turtle.**position**()

Return the turtle’s current location (x,y) (as a [Vec2D](https://docs.python.org/3/library/turtle.html#turtle.Vec2D) vector).>>>

Example

**>>>** turtle.pos()

(440.00,-0.00)

turtle.**xcor**()

Return the turtle’s x coordinate.

Example>>>

**>>>** turtle.home()

**>>>** turtle.left(50)

**>>>** turtle.forward(100)

**>>>** turtle.pos()

(64.28,76.60)

**>>>** print(round(turtle.xcor(), 5))

64.27876

### Pen control

turtle.**pendown**()

Pull the pen down – drawing when moving.

turtle.**penup**()

Pull the pen up – no drawing when moving.

turtle.**color**(*\*args*)

Return or set pencolor and fillcolor.

Several input formats are allowed. They use 0 to 3 arguments as follows:

color()

color(colorstring), color((r,g,b)), color(r,g,b)

Example

>>> turtle.color("red", "green")

>>> turtle.color()('red', 'green')

>>> color("#285078", "#a0c8f0")

>>> color() ((40.0, 80.0, 120.0),(160.0, 200.0, 240.0))

#### Filling

turtle. **begin\_fill**()

To be called just before drawing a shape to be filled.

turtle.**end\_fill**()

Fill the shape drawn after the last call to [begin\_fill()](https://docs.python.org/3/library/turtle.html" \l "turtle.begin_fill).

Whether or not overlap regions for self-intersecting polygons or multiple shapes are filled depends on the operating system graphics, type of overlap, and number of overlaps. For example, the Turtle star above may be either all yellow or have some white regions.

Example >>>

**>>>** turtle.color("black", "red")

**>>>** turtle.begin\_fill()

**>>>** turtle.circle(80)

**>>>** turtle.end\_fill()

#### Appearance

turtle.**shape**(*name=None*)

**Parameters: name** – a string which is a valid shape name

### Shape with *name* must exist in the TurtleScreen’s shape dictionary.

### Example

**>>>** turtle.shape("turtle")

**>>>** turtle.shape()

'turtle'

turtle.**shapesize**(*stretch\_wid=None*, *stretch\_len=None*, *outline=None*)

turtle.**turtlesize**(*stretch\_wid=None*, *stretch\_len=None*, *outline=None*)

**Parameters: stretch\_wid** – positive number

**stretch\_len** – positive number

**outline** – positive number

Example

**>>>** turtle.shapesize()

(1.0, 1.0, 1)

**>>>** turtle.resizemode("user")

**>>>** turtle.shapesize(5, 5, 12)

**>>>** turtle.shapesize()

(5, 5, 12)

CHAPTER 4

SYSTEM SPECIFICATION

**SOFTWARE REQUIREMENTS**

Operating system : Windows

Language : Python

**HARDWARE REQUIREMENTS**

Processor : Intel®Core™i5

Hard disk : 500GB

RAM : 2GB

System Type : 64Bit operating system

CHAPTER 5

IMPLEMENTATION

from turtle import \*

from random import \*

import turtle

import time

setup(800, 500)

title("Turtle Race")

bgcolor("forestgreen")

speed(0)

penup()

goto(-100, 205)

color("white")

write("TURTLE RACE", font=("Arial", 20, "bold"))

goto(-350, 200)

pendown()

color("chocolate")

begin\_fill()

for i in range(2):

forward(700)

right(90)

forward(400)

right(90)

end\_fill()

gap\_size = 20

shape("square")

penup()

color("white")

for i in range(10):

goto(250, (170 - (i \* gap\_size \* 2)))

stamp()

# WHITE SQUARES ROW 2

for i in range(10):

goto(250 + gap\_size, ((210 - gap\_size) - (i \* gap\_size \* 2)))

stamp()

# BLACK SQUARES ROW 1

color("black")

for i in range(10):

goto(250, (190 - (i \* gap\_size \* 2)))

stamp()

# BLACK SQUARES ROW 2

for i in range(10):

goto(251 + gap\_size, ((190 - gap\_size) - (i \* gap\_size \* 2)))

stamp()

# TURTLE 1 - BLUE

blue\_turtle = Turtle()

blue\_turtle.color("cyan")

blue\_turtle.shape("turtle")

blue\_turtle.shapesize(1.5)

blue\_turtle.penup()

blue\_turtle.goto(-300, 150)

blue\_turtle.pendown()

# TURTLE 2 - PINK

pink\_turtle = Turtle()

pink\_turtle.color("magenta")

pink\_turtle.shape("turtle")

pink\_turtle.shapesize(1.5)

pink\_turtle.penup()

pink\_turtle.goto(-300, 50)

pink\_turtle.pendown()

# TURTLE 3 - YELLOW

yellow\_turtle = Turtle()

yellow\_turtle.color("yellow")

yellow\_turtle.shape("turtle")

yellow\_turtle.shapesize(1.5)

yellow\_turtle.penup()

yellow\_turtle.goto(-300, -50)

yellow\_turtle.pendown()

# TURTLE 4 - GREEN

green\_turtle = Turtle()

green\_turtle.color("lime")

green\_turtle.shape("turtle")

green\_turtle.shapesize(1.5)

green\_turtle.penup()

green\_turtle.goto(-300, -150)

green\_turtle.pendown()

# PAUSE FOR 1 SECOND BEFORE RACING

time.sleep(1)

# move the turtles

while blue\_turtle.xcor() <= 230 and pink\_turtle.xcor() <= 230 and yellow\_turtle.xcor() <= 230 and green\_turtle.xcor() <= 230:

blue\_turtle.forward(randint(1, 10))

pink\_turtle.forward(randint(1, 10))

yellow\_turtle.forward(randint(1, 10))

green\_turtle.forward(randint(1, 10))

# celebrate the winner

# blue turtle wins

if blue\_turtle.xcor() > pink\_turtle.xcor() and blue\_turtle.xcor() > yellow\_turtle.xcor() and blue\_turtle.xcor() > green\_turtle.xcor():

print("blue turtle wins the race!!!")

for i in range(72):

blue\_turtle.right(5)

blue\_turtle.shapesize(2.5)

# pink turtle wins

elif pink\_turtle.xcor() > blue\_turtle.xcor() and pink\_turtle.xcor() > yellow\_turtle.xcor() and pink\_turtle.xcor() > green\_turtle.xcor():

print("pink turtle wins the race!!!")

for i in range(72):

pink\_turtle.right(5)

pink\_turtle.shapesize(2.5)

# yellow turtle wins

elif yellow\_turtle.xcor() > blue\_turtle.xcor() and yellow\_turtle.xcor() > pink\_turtle.xcor() and yellow\_turtle.xcor() > green\_turtle.xcor():

print("yellow turtle wins the race!!!")

for i in range(72):

yellow\_turtle.right(5)

yellow\_turtle.shapesize(2.5)

else:

print("green turtle wins the race!!!")

for i in range(72):

green\_turtle.right(5)

green\_turtle.shapesize(2.5)

CHAPTER 6

RESULTS

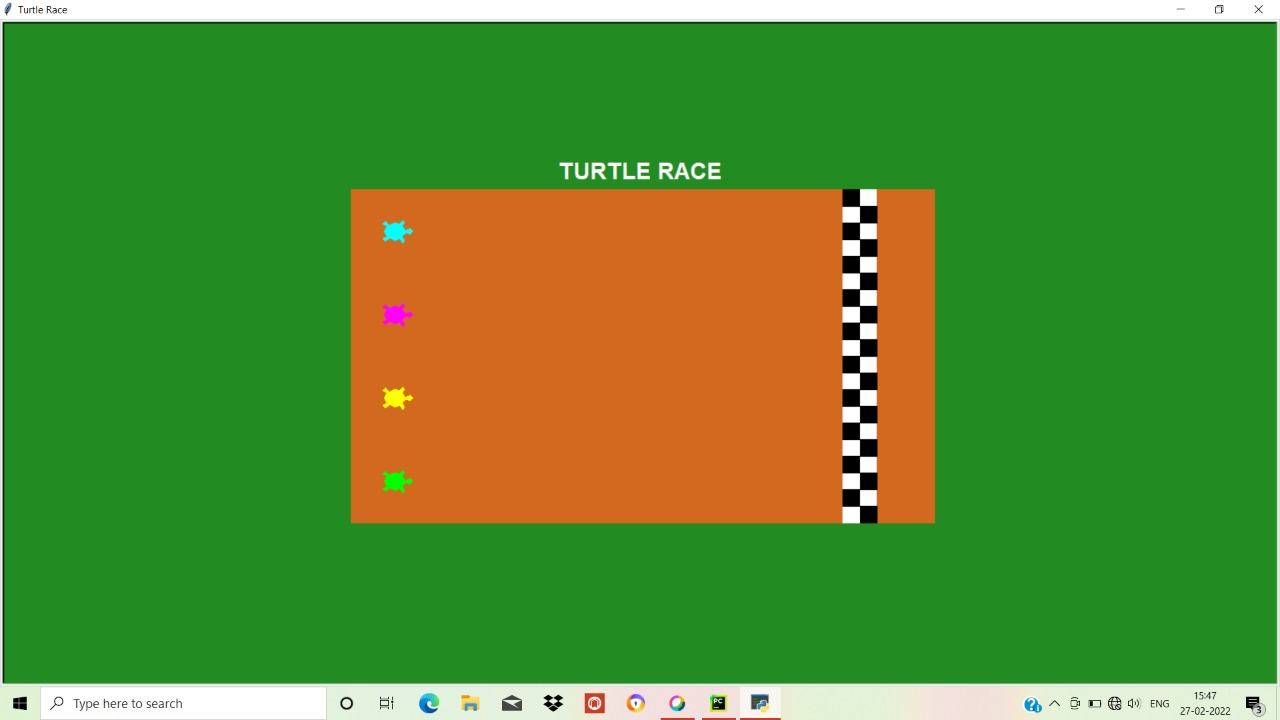
FIG 1.1: Background colour is Green. Title of the game is “Turtle Race”. The Ground colour is given as orange. There are four turtle present in the ground for the race. The four turtle are at the starting position 

FIG 1.2: The Second output figure shows that the race has begin between between four turtle ‘s which have different colours (Green, yellow, Pink, Blue). They have to reach the final line which have been displayed in combination of black and white boxes.

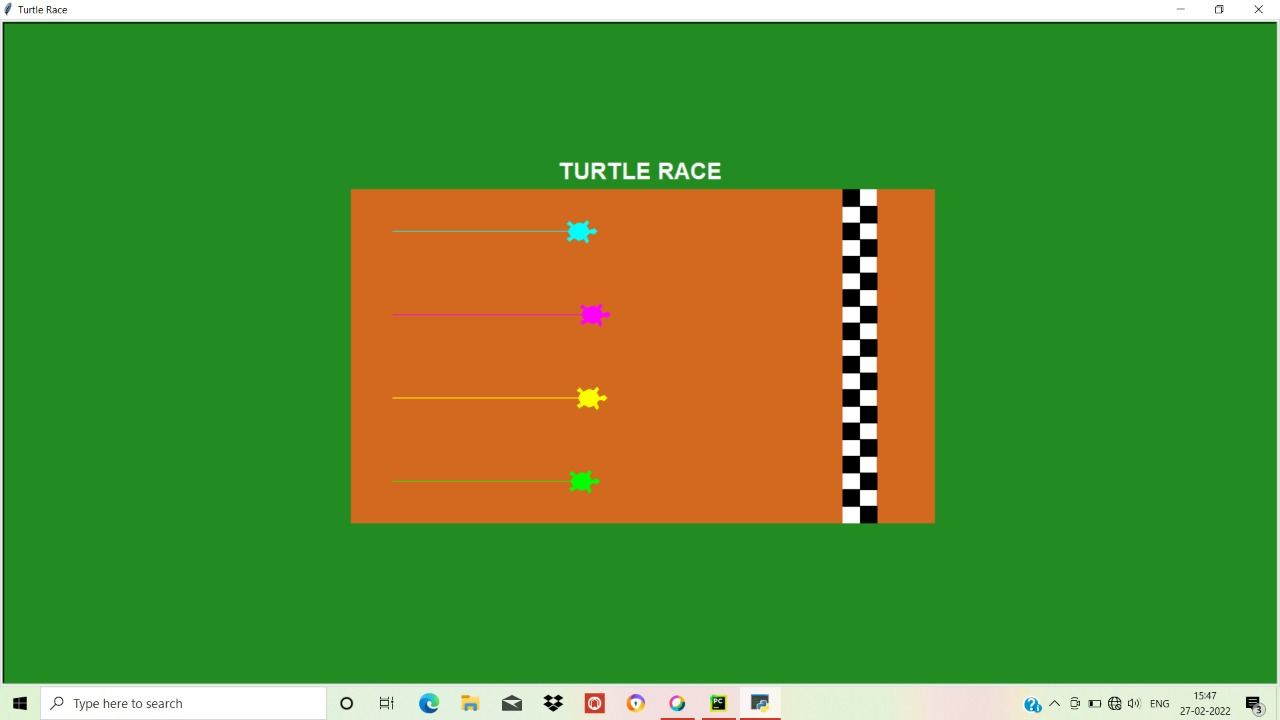


FIG 1.3: The third output figure shows that

Blue turtle has win the race .It is highlighted like wise any turtle can win the race

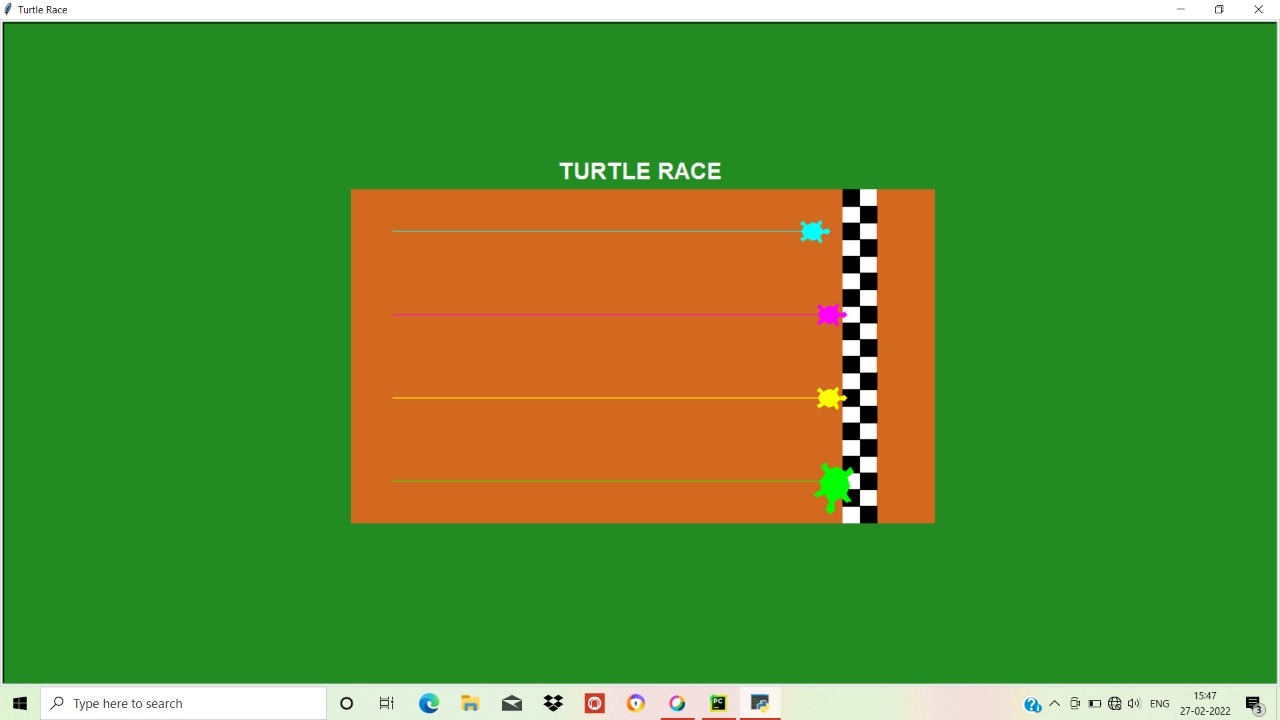
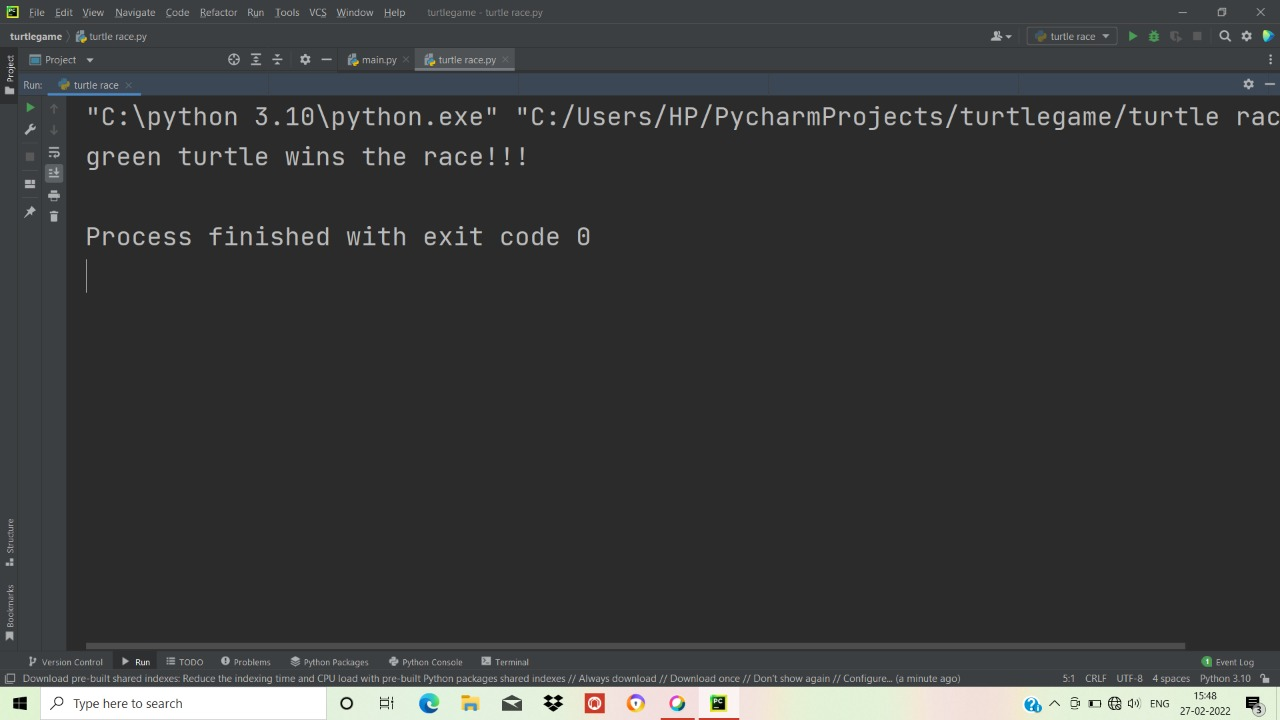


FIG 1.4: The fourth output fig:

Displays the message “Blue turtle has win the race!”



CHAPTER 6

CONCLUSION

We have tried to implement the Turtle Race game using turtle package (Graphics). Which can perform basic operation of Python , providing simple implementation.

In future we would like to implement other type of games using the Python language.

CHAPTER 7

REFERENCES

[1][www.google.com](http://www.google.com)

[2] wikipedia.com

[3] <https://docs.python.org/3/library/turtle.html>

[4] [www.w3schools.com](http://www.w3schools.com)