Bangladesh University of Business & Technology

B.Sc. Eng. Report

Submitted to
Department of Computer Science & Engineering
(In partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science & Engineering)

A Report on Arcade Video Game (Snake Game)

Md.Tajwar Ali (19202103183)
Tasmia Binte Munir Maisha (19202103195)
Parvas Hossain Piash (19202103200)
Sadman Sakib (19202103167)

Contents

Acknowledgement

We would like to pay our gratitude to the Almighty Allah who created us with all the abilities to understand analysis and develop the process with patience. We are thankful to our report supervisor Jubayer Al Mahmud, Assistant Professor, Computer Science and Engineering Department, Bangladesh University of Business and Technology for his professional guidance and motivation during the work of this thesis which is a major part of it. Without his valuable support and guidance, this report could not reach this level of development from our point of view.

Abstract

We will need to be aware of the necessary tools before you can begin learning about game creation. Python may not always be the most widely used language in this industry. Instead, there is a compelling argument to be made for the use of low-level languages like C++. Yet, many gaming firms use Python extensively in their development processes, and a number of well-known titles, such as Eve Online, Civilization IV, and Frets on Fire, were created entirely in this language. We can also start creating our own games using a ton of Python-based tools.

Chapter 1

Introduction

Snake game is one of the most popular arcade games of all time. In this game, the main objective of the player is to catch the maximum number of fruits without hitting the wall or itself. Creating a snake game can be taken as a challenge while learning Python or Pygame.

The "snake" that the player controls in the snake game has a set beginning point and advances continuously while getting longer. To avoid running into walls and either snake's body, it must be maneuvered left, right, up, and down. The winner is the person who lasts the longest. One or more snakes are controlled by AI in a single-player version of the idea, similar to the light cycles section of the arcade game.

Python is used in game development to write short scripts that do automation. In video games, this is comparable to the accumulation of points or the dashboard that shows the locations of territories acquired on a map while playing. The Python gaming framework enables the creation of games for the Windows, Mac, Linux, iOS, and Android operating systems. The language is frequently employed to create video games that can be played on both mobile and Computer platforms.

This one language makes it simple to make whole video games. People who already own a game can improve and modify its versioning by adding scripting engines. Python is an accessible programming language that anyone can use to create video games rapidly.

Chapter 2

Methodology

2.1 Requirement Analysis

In the action video game subgenre of "Snake Game!," the player controls the tip of a snake-like line that is constantly growing. In order to build this game we have used the Python programming language. The turtle library played a major role in building this game. A virtual canvas is made available to users through the pre-installed Python module called turtle, which allows them to draw images and shapes. The library's name derives from the on-screen pen that you use to sketch, which is referred to as the turtle.

We have also used the time function which returns the number of seconds since the epoch (the point where time begins). The Python time module offers a variety of coding representations for time, including objects, integers, and strings. Together with functions other than time representation, it also allows you to measure the effectiveness of our code and wait while it executes.

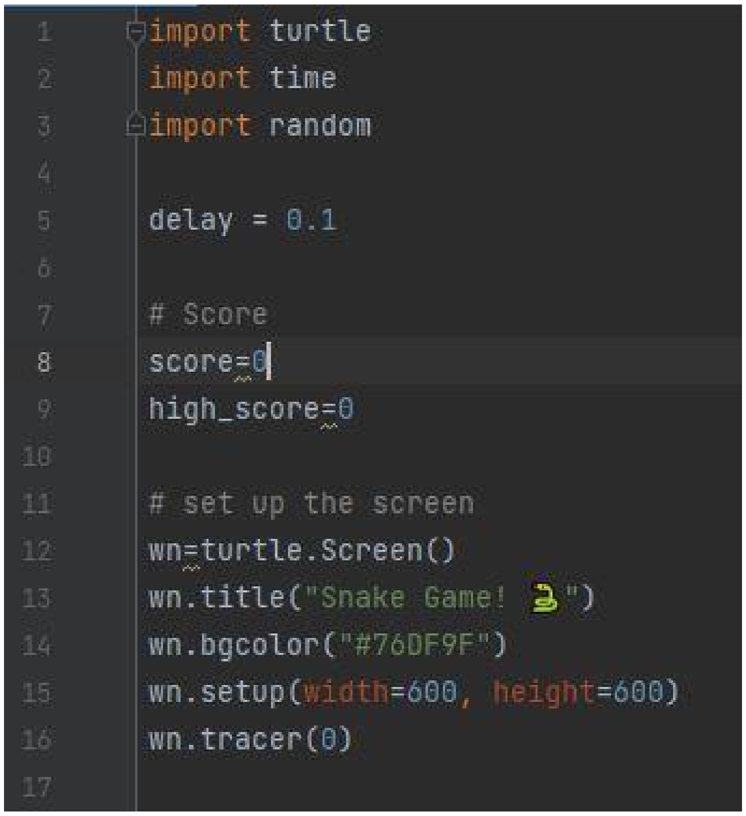
Also the import random function. The Python import random module in Python defines a series of functions for generating or manipulating random integers. Python random() is a pseudo-random number generator function that generates a random float number between 0.0 and 1.0, is used by functions in the random module.

2.1.1 Software Requirement

- Pycharm or Google colab.
- turtle library.
- graphics.py library (optional).

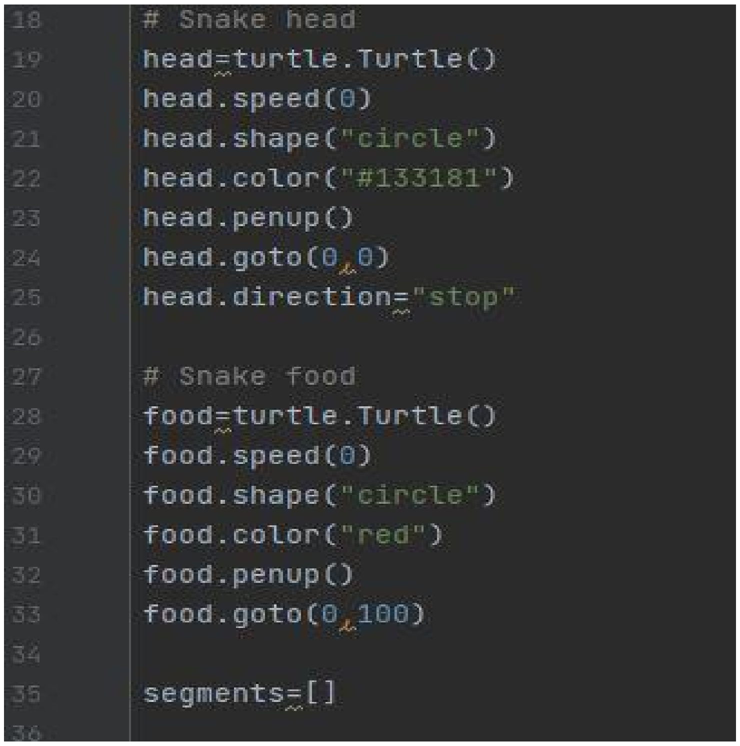
2.2 Implementation

Screenshots of code:



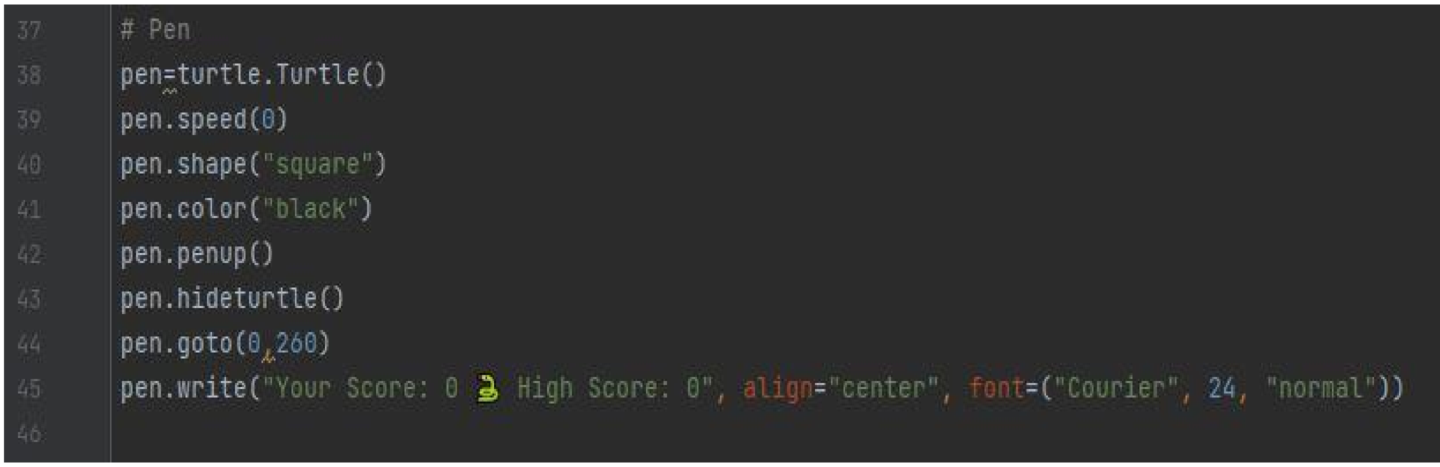
```
import turtle
import time
import random

delay = 0.1

# Score
score=0
high_score=0

# set up the screen
wn=turtle.Screen()
wn.title("Snake Game! 🐍")
wn.bgcolor("#76DF9F")
wn.setup(width=600, height=600)
wn.tracer(0)

```

Fig.2.2.1: Importing the library functions and set up the window screen.



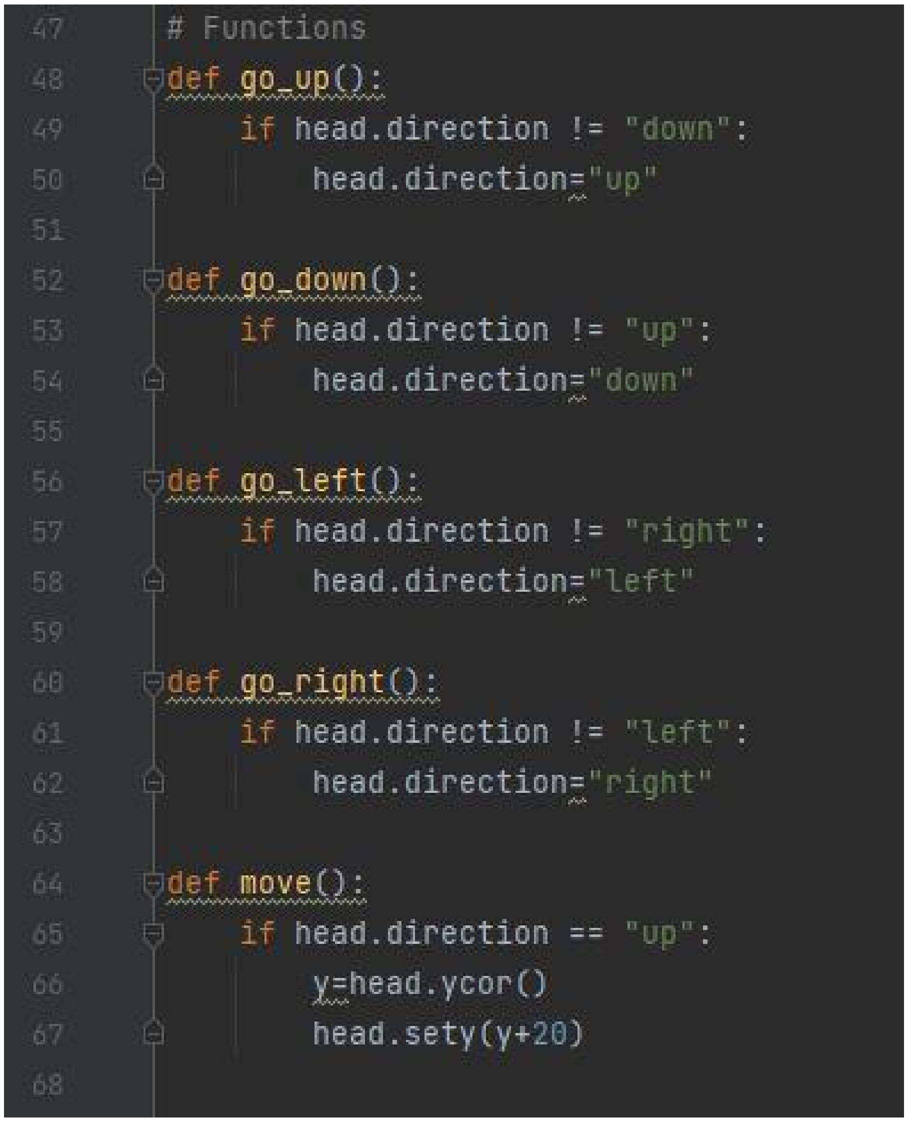
```
# Snake head
head=turtle.Turtle()
head.speed(0)
head.shape("circle")
head.color("#133181")
head.penup()
head.goto(0,0)
head.direction="stop"

# Snake food
food=turtle.Turtle()
food.speed(0)
food.shape("circle")
food.color("red")
food.penup()
food.goto(0,100)

segments=[]

```

Fig.2.2.2: Creating the snake head and snake food.



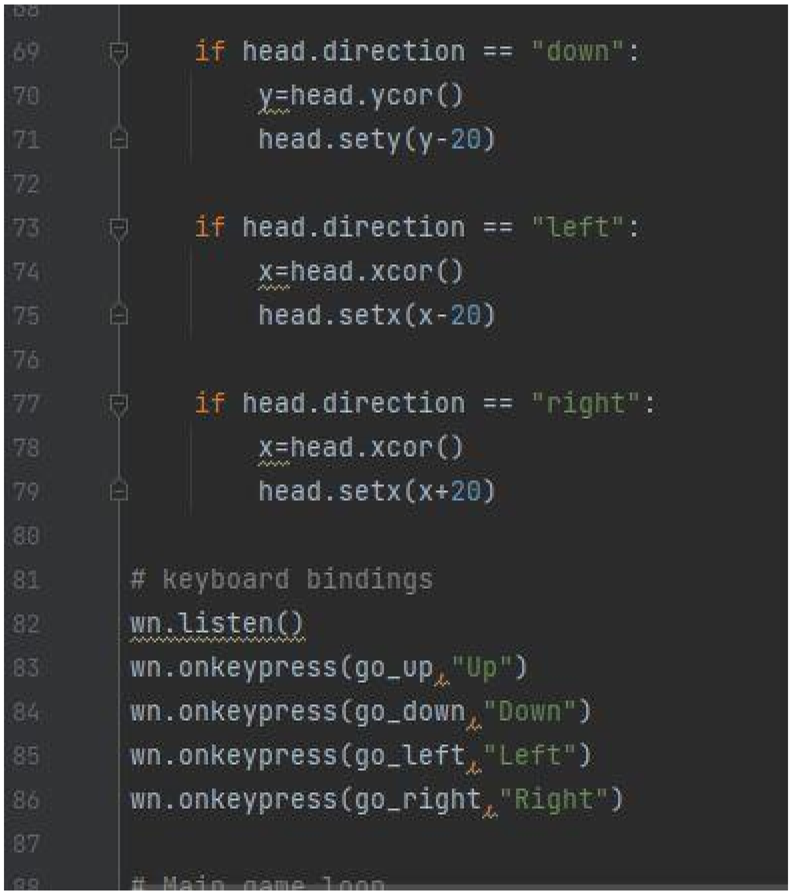
```python
# Pen
pen=turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("black")
pen.penup()
pen.hideturtle()
pen.goto(0,260)
pen.write("Your Score: 0 🐍 High Score: 0", align="center", font=("Courier", 24, "normal"))

```

Fig.2.2.3: Creating the pen part.



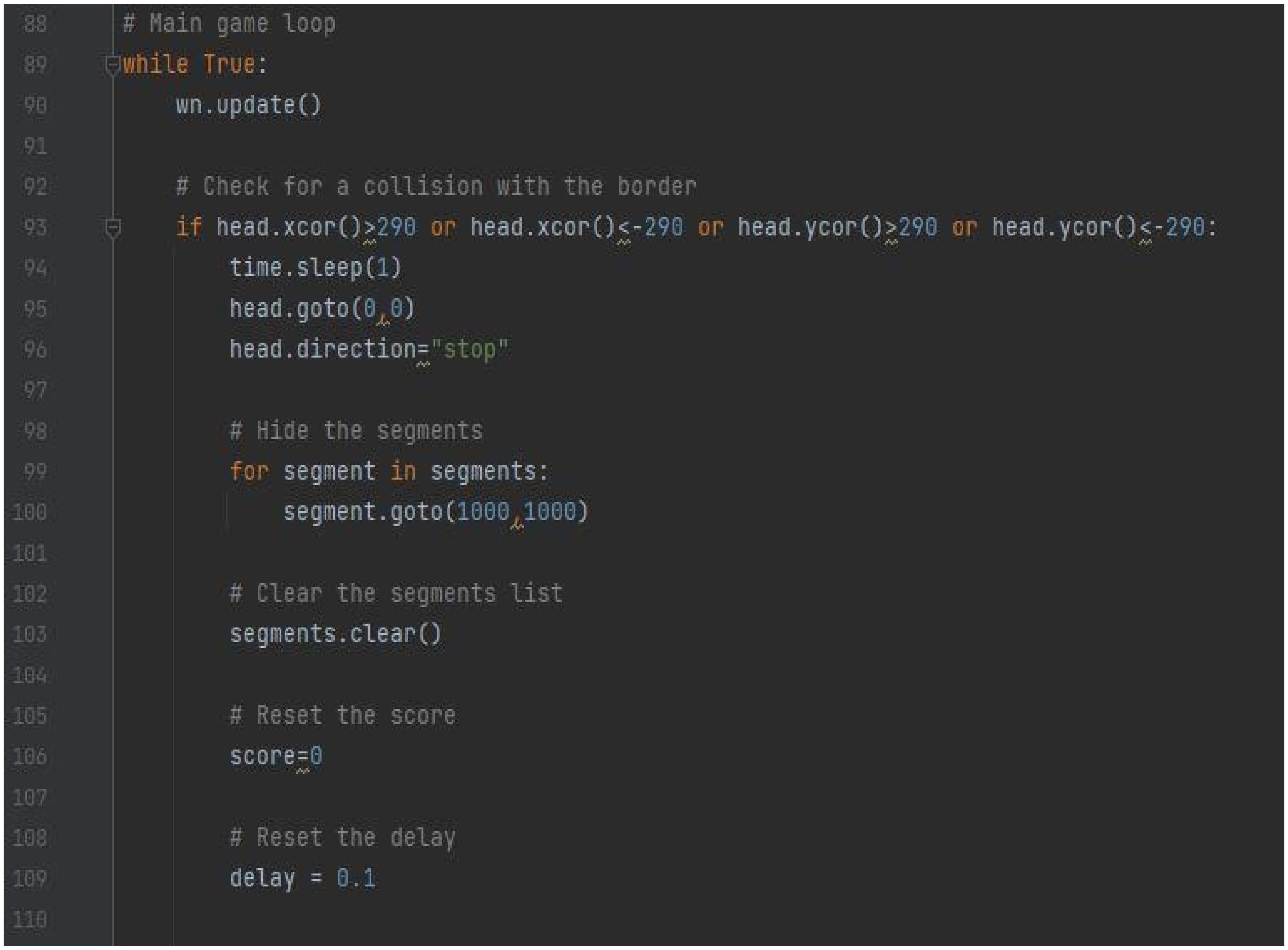
```python
# Functions
def go_up():
    if head.direction != "down":
        head.direction="up"

def go_down():
    if head.direction != "up":
        head.direction="down"

def go_left():
    if head.direction != "right":
        head.direction="left"

def go_right():
    if head.direction != "left":
        head.direction="right"

def move():
    if head.direction == "up":
        y=head.ycor()
        head.sety(y+20)

```

Fig.2.2.4: Creating the functionalities.



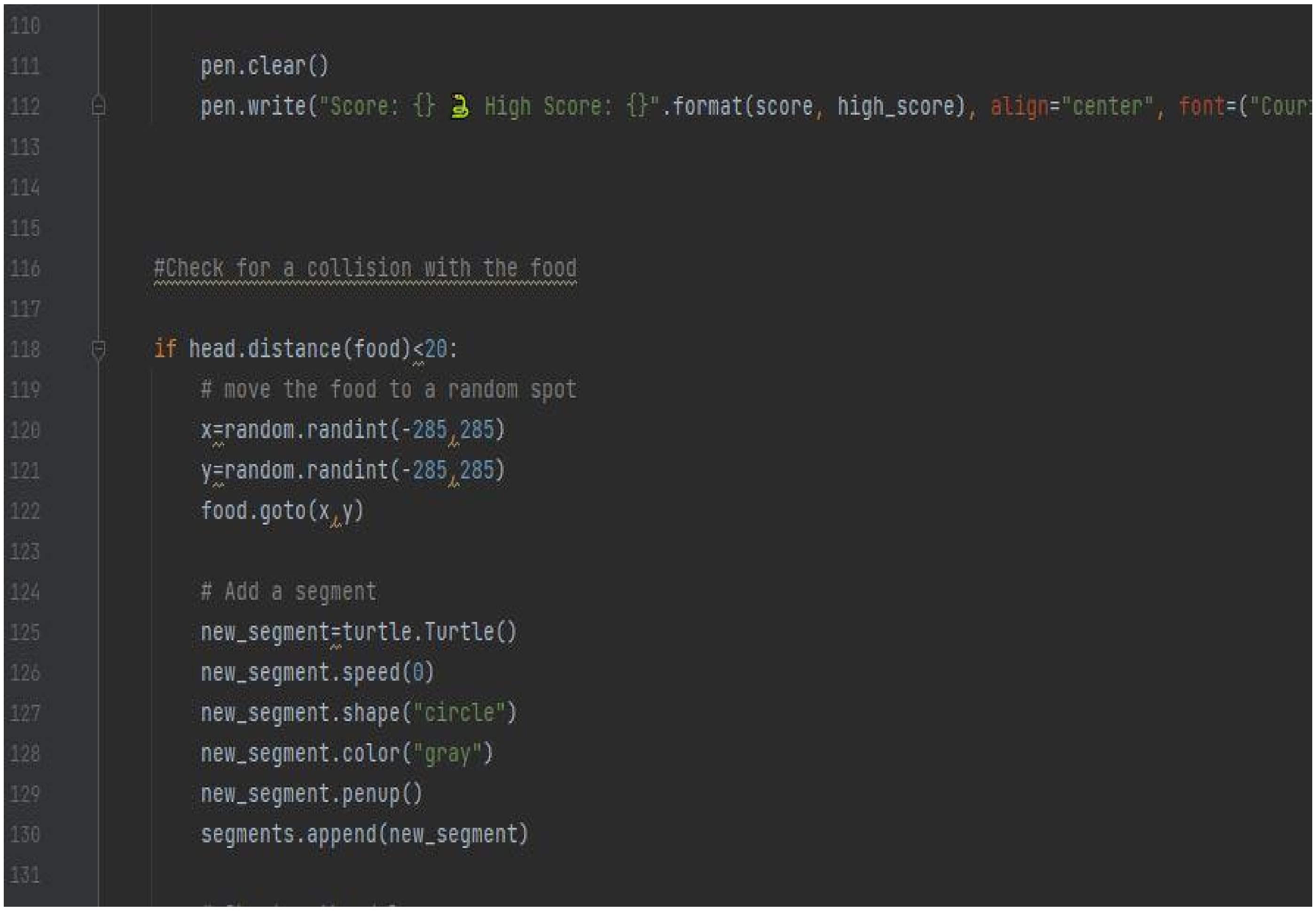
```
    if head.direction == "down":
        y=head.ycor()
        head.sety(y-20)

    if head.direction == "left":
        x=head.xcor()
        head.setx(x-20)

    if head.direction == "right":
        x=head.xcor()
        head.setx(x+20)

# keyboard bindings
wn.listen()
wn.onkeypress(go_up,"Up")
wn.onkeypress(go_down,"Down")
wn.onkeypress(go_left,"Left")
wn.onkeypress(go_right,"Right")

# Main game loop
```

Fig.2.2.5: Creating the keyword bindings.



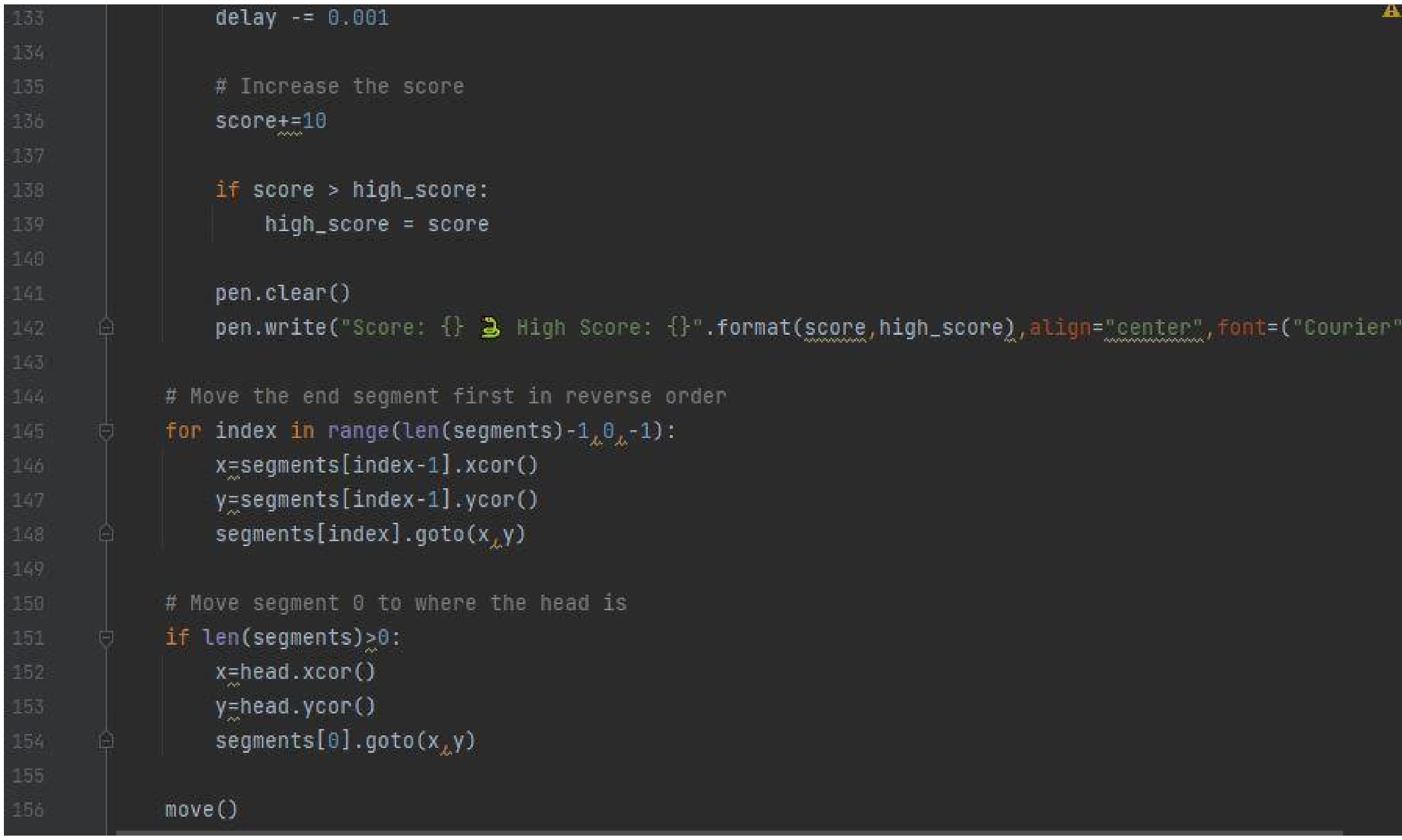
```
# Main game loop
while True:
    wn.update()

    # Check for a collision with the border
    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"

        # Hide the segments
        for segment in segments:
            segment.goto(1000,1000)

        # Clear the segments list
        segments.clear()

        # Reset the score
        score=0

        # Reset the delay
        delay = 0.1

```

Fig.2.2.6: Creating the snake head and snake food.

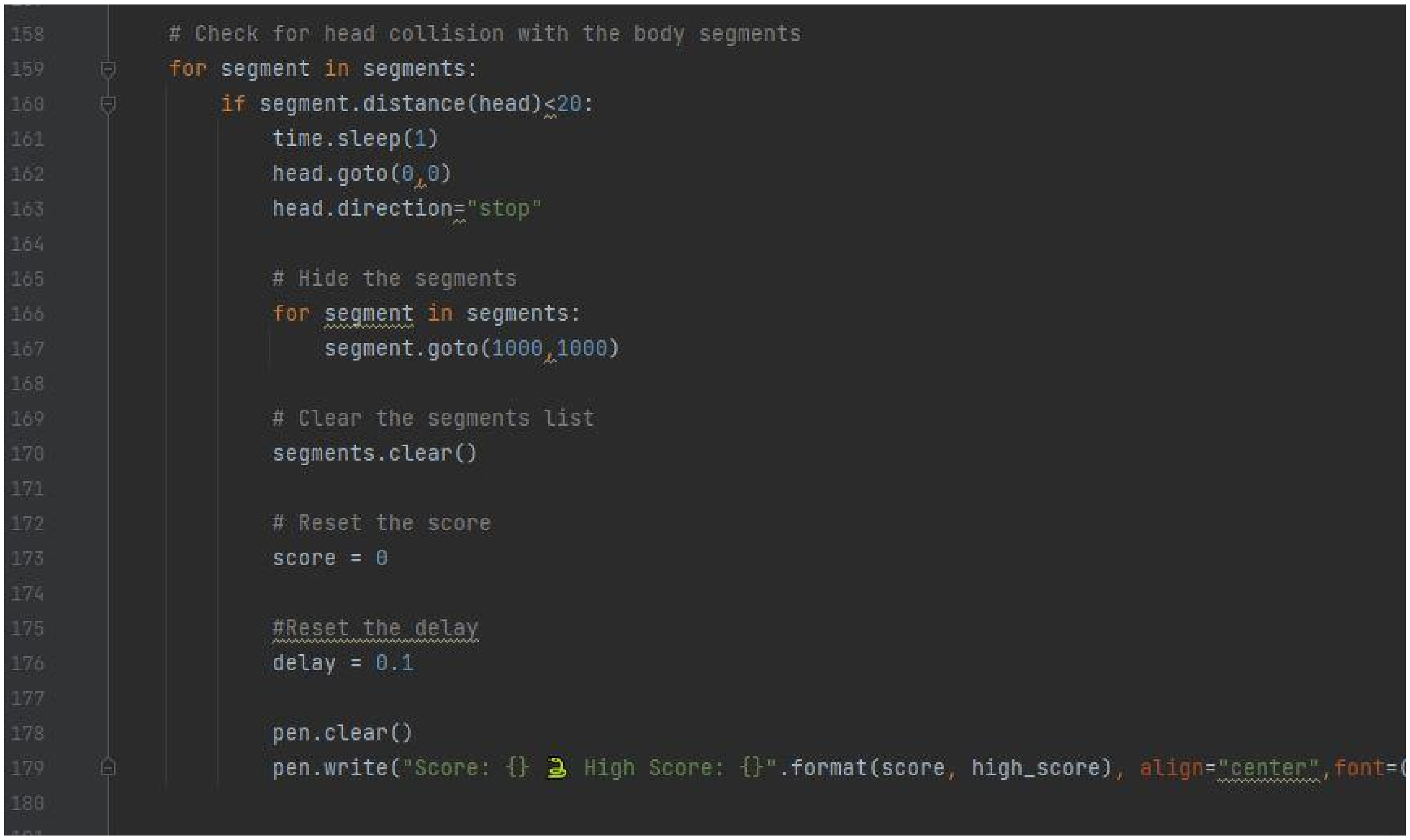


```

        pen.clear()
        pen.write("Score: {} 🐍 High Score: {}".format(score, high_score), align="center", font=("Couri



    #Check for a collision with the food

    if head.distance(food)<20:
        # move the food to a random spot
        x=random.randint(-285,285)
        y=random.randint(-285,285)
        food.goto(x,y)

        # Add a segment
        new_segment=turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("circle")
        new_segment.color("gray")
        new_segment.penup()
        segments.append(new_segment)

```

Fig.2.2.7: Creating the collision withe the food part.

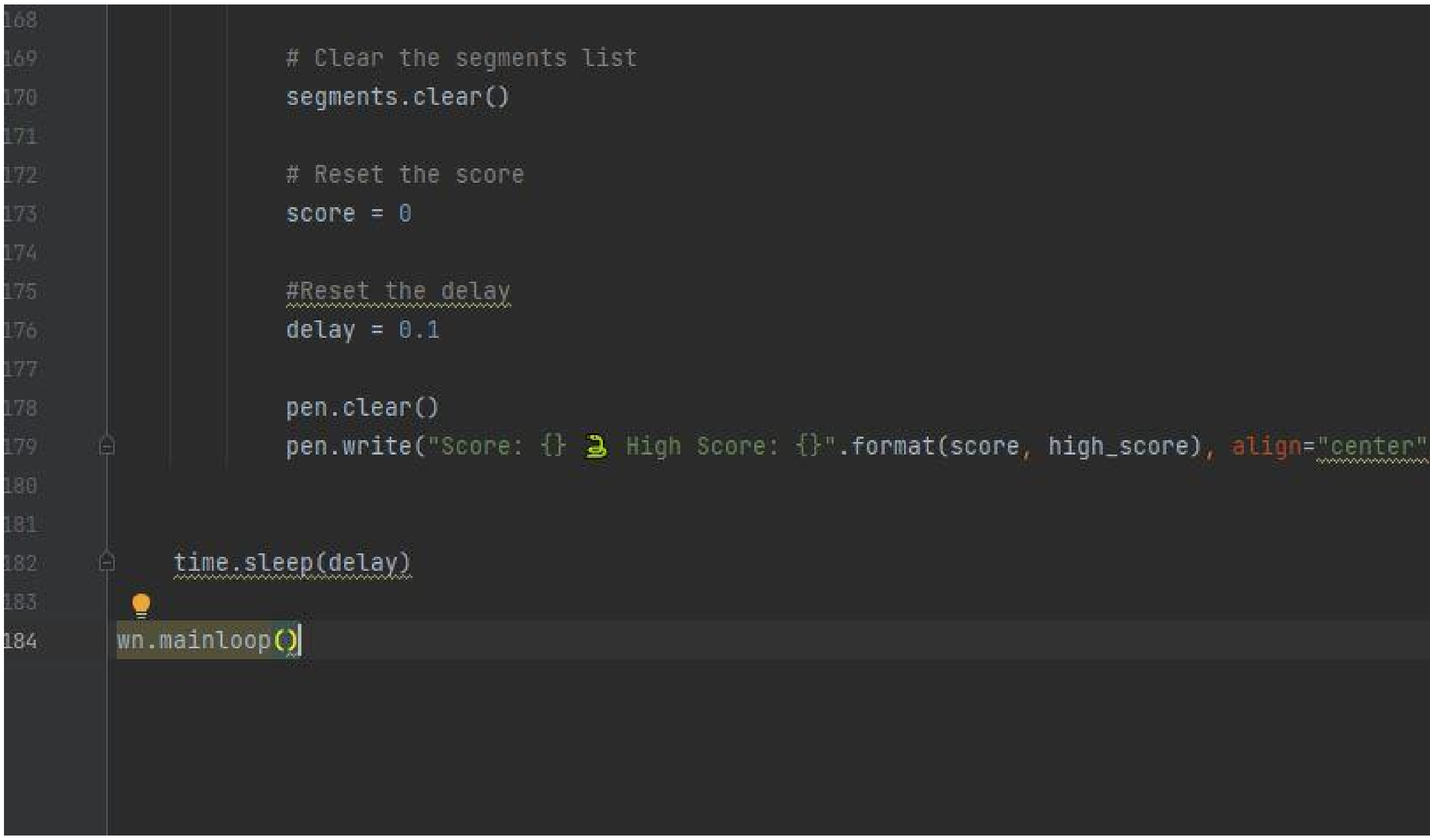


```
        delay -= 0.001

        # Increase the score
        score+=10

        if score > high_score:
            high_score = score

        pen.clear()
        pen.write("Score: {} 🐍 High Score: {}".format(score,high_score),align="center",font=("Courier"

    # Move the end segment first in reverse order
    for index in range(len(segments)-1,0,-1):
        x=segments[index-1].xcor()
        y=segments[index-1].ycor()
        segments[index].goto(x,y)

    # Move segment 0 to where the head is
    if len(segments)>0:
        x=head.xcor()
        y=head.ycor()
        segments[0].goto(x,y)

    move()
```



```
    # Check for head collision with the body segments
    for segment in segments:
        if segment.distance(head)<20:
            time.sleep(1)
            head.goto(0,0)
            head.direction="stop"

            # Hide the segments
            for segment in segments:
                segment.goto(1000,1000)

            # Clear the segments list
            segments.clear()

            # Reset the score
            score = 0

            #Reset the delay
            delay = 0.1

            pen.clear()
            pen.write("Score: {} 🐍 High Score: {}".format(score, high_score), align="center",font=(
```

Fig.2.2.8: Creating the snake head collision with the body segments.



```
                # Clear the segments list
                segments.clear()

                # Reset the score
                score = 0

                #Reset the delay
                delay = 0.1

                pen.clear()
                pen.write("Score: {} 🐍 High Score: {}".format(score, high_score), align="center"


        time.sleep(delay)

    wn.mainloop()
```

Fig.2.2.9: Finish creating the game.

Chapter 3

Experimental Results

Screenshots of Outputs:

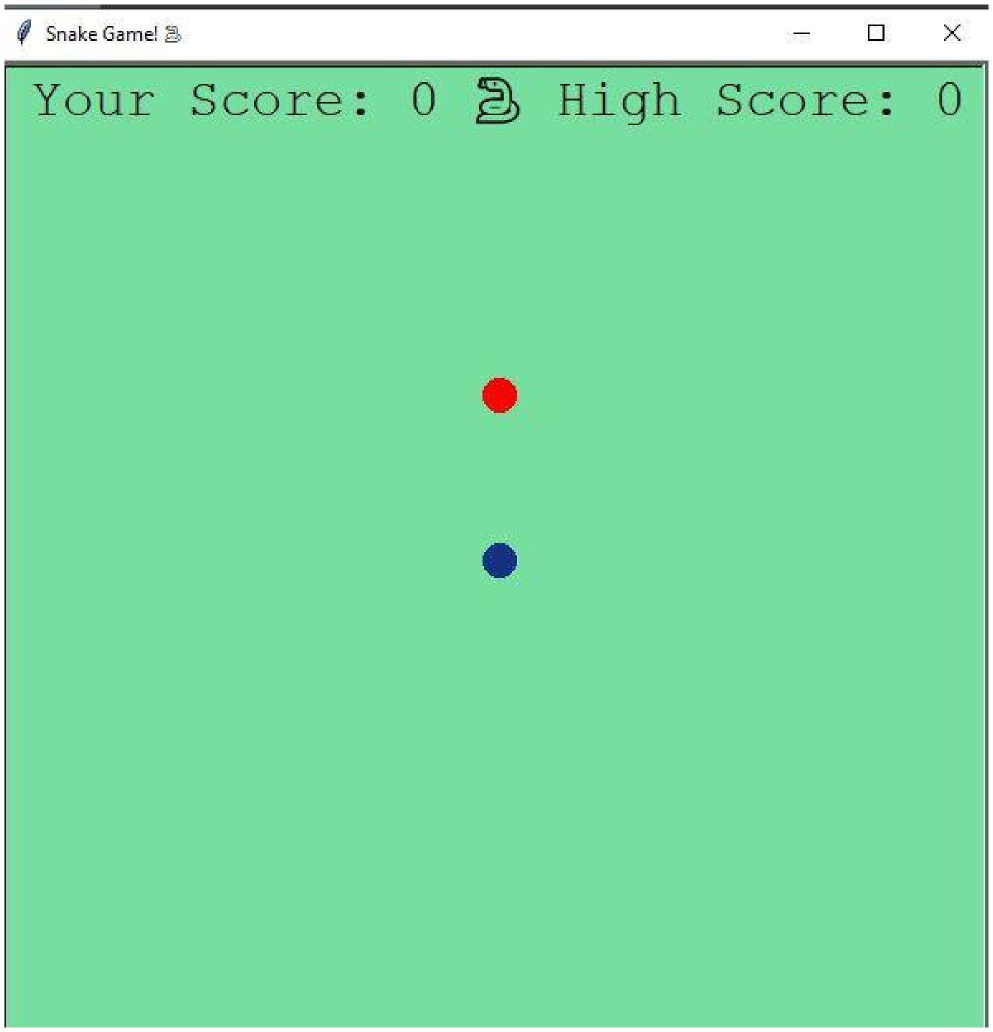


Fig.3.1: Output-1.

Observation:

- At start the score remains 0.
- Food defines the red color and the snake head is defined with blue color.

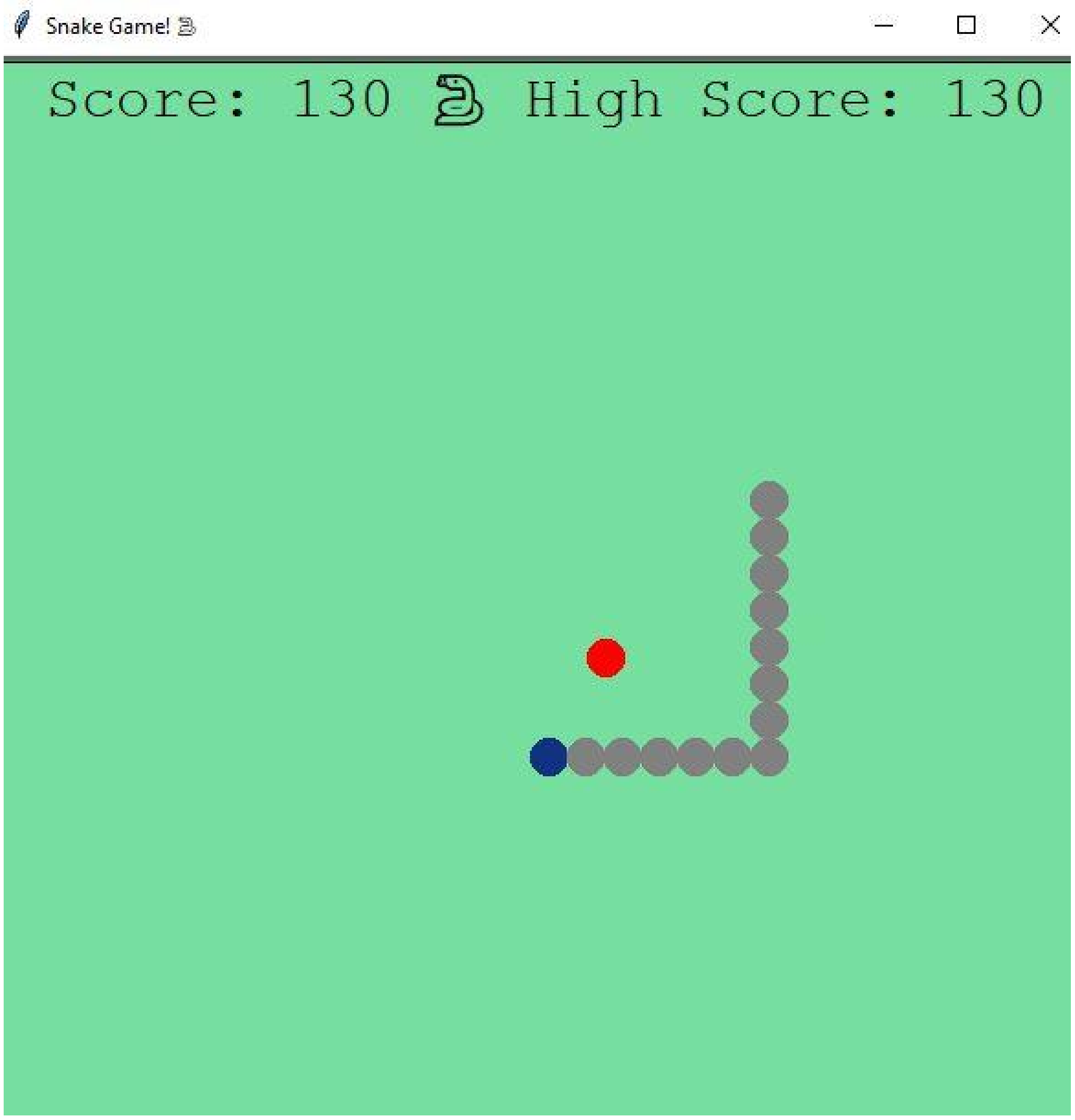


Fig.3.2: Output-2.

Observation:

- Snake body segment increases as it eats the food.
- The score increases at the same time also.

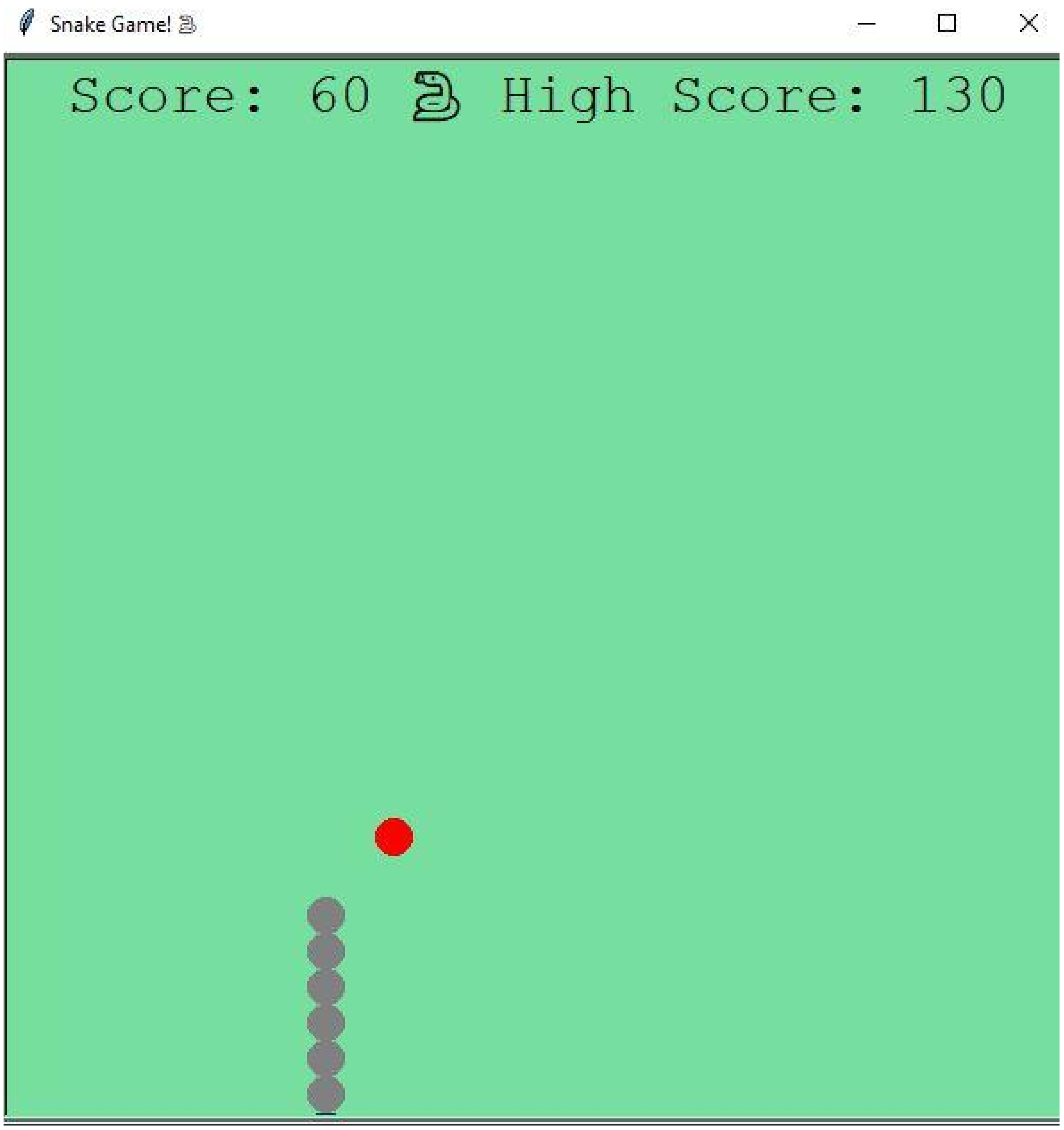


Fig.3.3: Output-3.

Observation:

- The game restarts if the snake crashes to the wall or bites its own body.
- The previous score becomes the highest score and the player has to start all over again to build-up new high score.

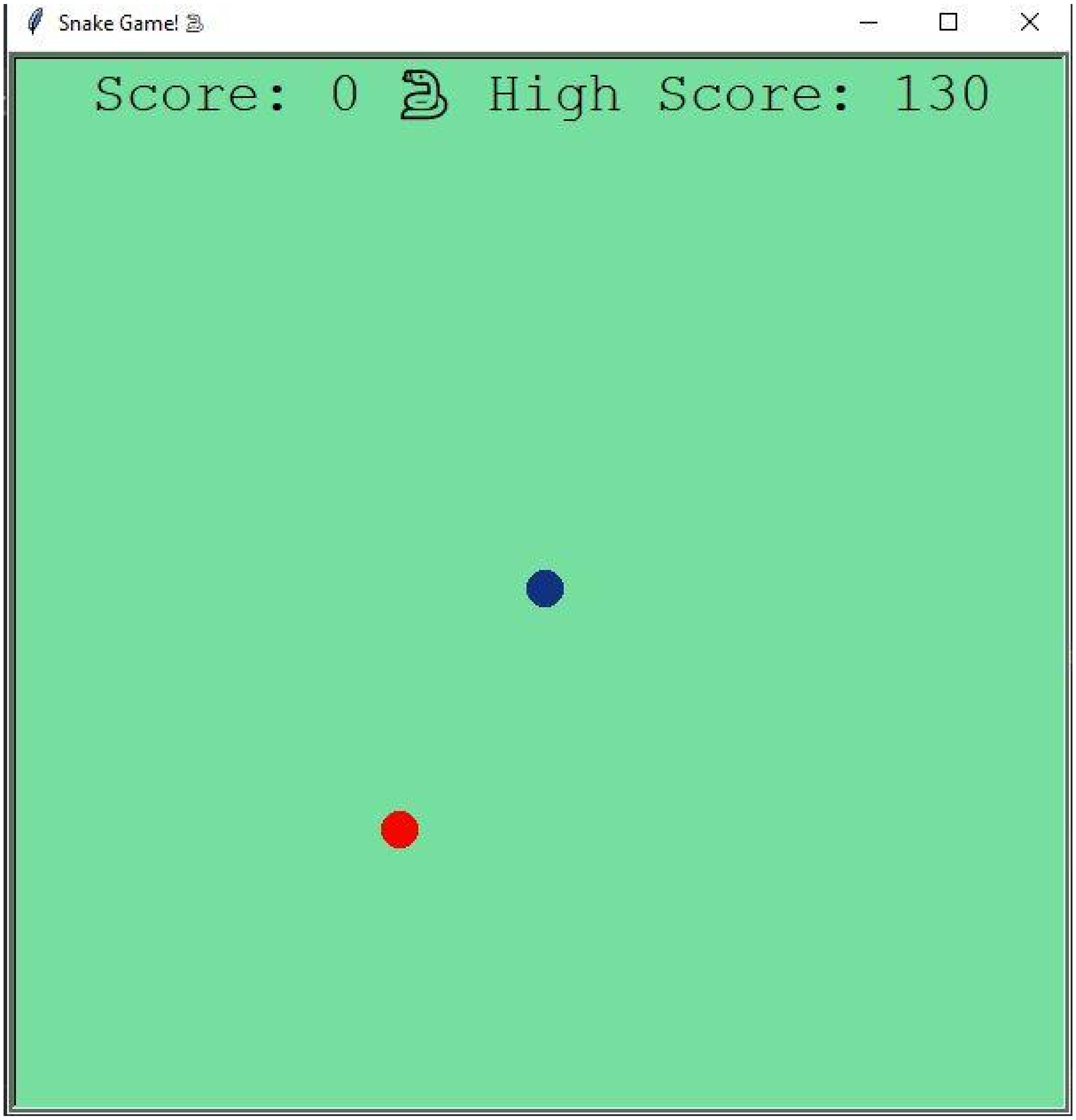


Fig.3.4: Output-4.

Observation:

- Previous highest score still remains.
- The player has to start a new game all over again!

Chapter 4

Conclusion

In brief, the Python turtle library offers a fun and engaging approach for new programmers to experience what it's like to work with Python. The primary purpose of turtle is to acquaint people with the world of computers. It's an easy method to grasp Python's ideas while also being flexible.

The most crucial life lesson that the traditional snake game imparts to its players is control over their regular activities. In addition, players discover the dual nature of life and the value of concentrating on a certain objective.