PROJECT REPORT On

**“Snake Game In Python”**

***Submitted By***

**Lucky K. Bhure**

***Guided By:-***

Mr. Ratnesh K. Choudhary



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## S. B. JAIN INSTITUTE OF TECHNOLOGY

**MANAGEMENT AND RESEARCH, NAGPUR.**

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

### 2021-2022

#### © S.B.J.I.T.M.R Nagpur 2022

i

### S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

SESSION 2021-2022

# CERTIFICATE

This is to certify that the Project titled **“Snake Game In Python”** is a bonafide work of **Lucky Krushnarao Bhure** carried out for the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in **Computer Science & Engineering**.

|  |  |  |
| --- | --- | --- |
| **Mr. Ratnesh K. Choudhary**  Assistant Professor |  | **Mr. Animesh Tayal**  Head of Department |

# INDEX

[CERTIFICATE i](#_Toc23564)

[INDEX ii](#_Toc23565)

[LIST OF FIGURES iii](#_Toc23566)

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 METHODOLOGY 2

CHAPTER 3 TOOLS/PLATFORMS 3

CHAPTER 4 DESIGN & IMPLEMENTATION 5-12

4.1 ALGORITHM

4.2 FLOWCHART

4.3 SOURCE CODE

CHAPTER 5 RESULT & DISCUSSION 13-16

5.1 OUTPUT

5.2 DISCUSSION

5.3 APPLICATION

CHAPTER 6 CONCLUSION 17

REFERENCES 18

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **TITLE OF FIGURE** | **PAGE NO.** |
| 4.2 | Snake game flowchart | 6 |
| 5.1.1 | Initial player interface | 13 |
| 5.1.2 | In-game screen | 14 |
| 5.1.4 | After the game is over | 15 |

#### CHAPTER 1

The Snake Game using python

The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head. The player loses when the snake runs into the screen border, other obstacle, or itself. We will create the snake game using pygame. **Pygame** is totally dependent on Python. We hope Python is already available on your system, and if you do not have it, first install the latest version of Python. **Pygame** is a crossplatform set of Python modules designed for composing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.tail or the edges of the playing area. Each time the snake eats a piece of food, its tail grows longer, making the game increasingly difficult.

**CHAPTER 2**

### METHODOLOGY

#### 1. REQUIREMENT ANALYSIS: -

At this stage we will review what users need basic features in a snake game like start, end, moving in all four directions, food to grow the size of snake, sounds for engagement, score, etc.

#### 2. PLANNING: -

In the planning stage, we should first try to explore out the features that the snake game can have. Next, we will eliminate the features that users feel no really useful or low cost-effective.

Finally, each feature’s is prioritized and assigned to an iteration.

#### 3. DESIGN: -

The design stage is prepared according to the requirements of users. Since there are many details and problems encountered during development to be considered for each feature. Therefore, we will discuss and formulate solutions and test strategies to verify the product at this stage.

#### 4. IMPLEMENTATION: -

During the development phase, we will iteratively implement each of the features listed during the planning phase. At this stage, there will be many setbacks and obstacle, we need to constantly overcome these obstacles. Moreover, we will prioritize the most important features and need to make intelligent trade-offs between the depth of completeness of a single feature and the breadth of implementation of multiple features.

#### 5. TESTING: -

In this stage, we will test the performance of each feature in order to check whether it meets the requirements of users. For example, we will test whether the application can be properly run, and check whether any errors occur in the running process and each feature is up to standard.

**CHAPTER 3**

### TOOLS/PLATFORMS

**3.1 SOFTWARE REQUIREMENT:**

1. **IDE / FRAMEWORK:** Pycharm
2. **LIBRARIES:**, pygame ,random , OS
3. **OPERATING SYSTEM:** Windows 11
4. **Language:** *Python*, Version: - 3.10.4

#### 1. Installing pygame and importing it in the project

Pygame - Pygame uses the [Simple DirectMedia Layer](https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer) (SDL) library, with the intention of allowing [real-time](https://en.wikipedia.org/wiki/Real-time_computer_graphics) [computer game](https://en.wikipedia.org/wiki/Computer_game) development without the [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) mechanics of the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)) and its derivatives. This is based on the assumption that the most [expensive](https://en.wikipedia.org/wiki/Computationally_expensive) functions inside games can be abstracted from the game logic, making it possible to use a [high-level programming language,](https://en.wikipedia.org/wiki/High-level_programming_language) such as Python, to structure the game.

Other features that SDL does have include vector math, [collision detection,](https://en.wikipedia.org/wiki/Collision_detection) 2D sprite [scene graph](https://en.wikipedia.org/wiki/Scene_graph) management, [MIDI](https://en.wikipedia.org/wiki/MIDI) support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing.

Applications using Pygame can run on Android phones and tablets with the use of Pygame Subset for Android (pgs4a). Sound, vibration, keyboard, and accelerometer are supported on Android.

##### 2. Importing random module in project

Random module - This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available. The functions supplied by this module are actually bound methods of a hidden instance of the [random.Random](https://docs.python.org/3/library/random.html#random.Random) class. You can instantiate your own instances of [Random](https://docs.python.org/3/library/random.html#random.Random) to get generators that don’t share state.

###### 3. Importing os module

Os module – It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

You first need to import the os module to interact with the underlying operating system. So, import it using the import os statement before using its functions.

4. User defined functions used in project

Functions that we define ourselves to do certain specific task are referred as user-defined functions.

The way in which we define and call functions in Python are already discussed.

Functions that readily come with Python are called built-in functions. If we use functions written by others in the form of library, it can be termed as library functions.

All the other functions that we write on our own fall under user-defined functions. So, our user defined function could be a library function to someone else. In python, we define the user-

defined function using def keyword, followed by the function name.

**3.2 HARDWARE REQUIREMENT:**

* 1. **PROCESSOR:** P IV or above.
  2. **RAM:** 1 GB RAM.
  3. **HARDISK :**120 GB

**CHAPTER 4**

### DESIGN & IMPLEMENTATION

**4.1 ALGORITHM**

**Step1.** Start

**Step2.** Import the necessary module that are require for the snake game e.g pygame ,random, Os. **Step 3.** Then initializing the pygame functions and the pygame.mixer functions for further uses in project.

**Step 4.** Defining a colors, screen size of the game.

**Step 5.** Defining the background and the images used in the project, and the game title.

**Step 6.** Defining all the functions that are being used in the main game loop.

**Step** **7**. The defined functions being used in the game loop are text\_screen, plot\_snake and welcome.

**Step** **8**. Entering the main game loop.

**Step** **9**.Execution of the random function to generate the food at random places.

**Step** **10**. Increment of the score when snake eat a food item.

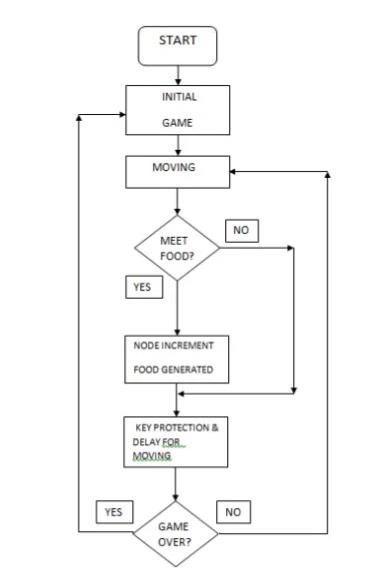
**Step** **11**. If snake coincide with the wall or overlap itself game\_over = true.

**Step** **12.** Displaying the exit or replying sreen .

**Step** **13.** Getting out of the main game loop.

**Step** **14**. STOP.

**4.2 FLOWCHART**



#### Fig.4.2: Flowchart for snake game

**4.3 SOURCE CODE**

# 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)

segment.clear()

score = 0

delay = 0.1

pen.clear()

pen.write("Score : {} High Score : {} ".format(

score, high\_score), align="center", font=("candara", 24, "bold"))

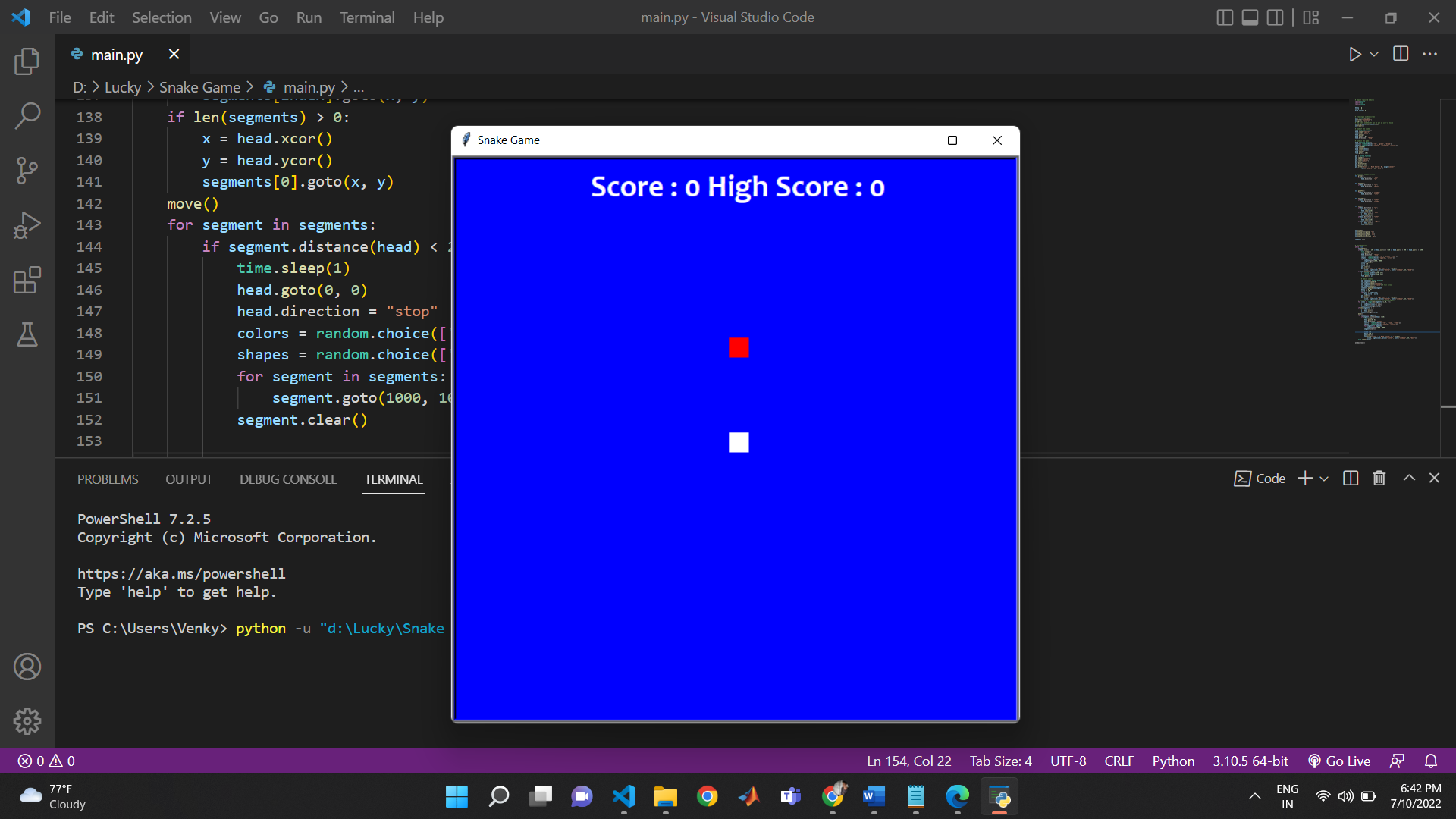
time.sleep(delay)

wn.mainloop()

**CHAPTER 5**

### RESULT & DISCUSSION

**5.1 OUTPUT**



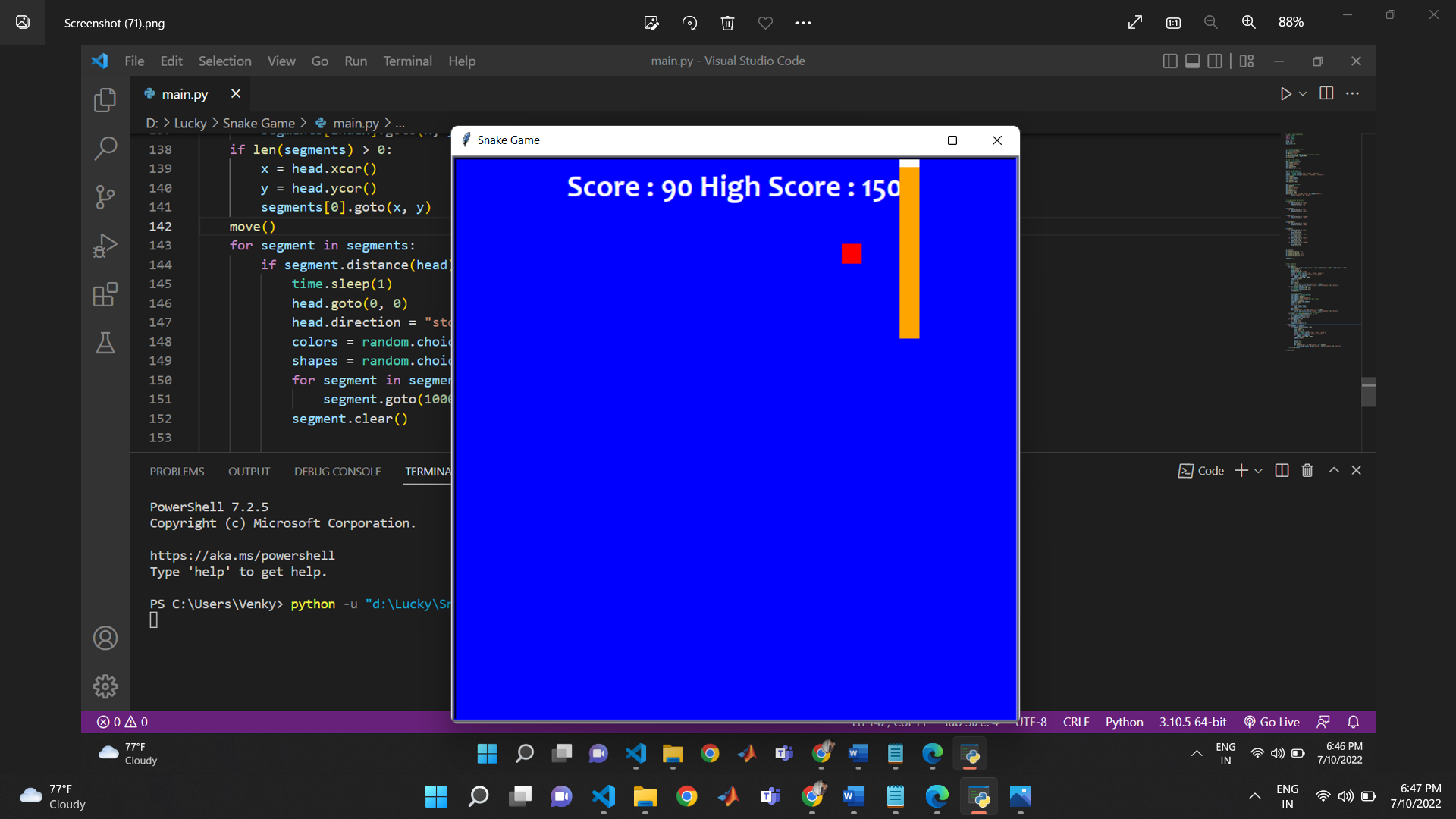
#### Fig 5.1.1: Initial player interface

**Fig 5.1.1:** It shows the Welcome screen to the user as soon as the program starts. And provides the information to press the space bar to play the game.

#### 

#### Fig 5.1.2: In-game screen

**Fig 5.1.2 :** A brief look of how the interface looks while playing the game. Here, the user or player can have look on the score and hi-score which has been made and see the position of snake and food item.

****

#### Fig 5.1.3: After the game is over

Fig 5.1.3: When the game is over this screen shows new game, then user can play new game and high score show at top.

**5.2 DISCUSSION**

First To create an snake game using the Python programming language to play the classic snake which is well-known foe it’s simple and playful interface. The game must have a simple and easyto-use interface. The game must have every common features that a classis snake game has such as Increment of length, food item, movement of snake with the arrow buttons in all the four directions for easy movement and to collect more food and increase the score of user. So here is the discussion about the music player that how it works.

First, we take a review what users need basic features in snake game After the review, we are clear that what we have to create

* So, I decided to use and pycharm ide.
* Then have imported required library’s like (pygame, random, OS).
* Then we have create a class named game\_loop in that we have define various functions to perform functionality like (text\_screen, plot\_snake, welcome).
* Then create a if condition to if the game\_over condition is true or not if it is true the game over .
* After writing the whole code, I run the program and a snake game exe appeared.
* Then the snake game ran without any error with full functionality.
* The whole program is properly working without any error**.**

**5.3 APPLICATION**

Snake is a classic game that requires players to assess their surroundings and find the quickest or safest route to a point. This is an excellent opportunity to learn about spatial awareness and plan ahead to your next move.

**CHAPTER 6**

### CONCLUSION

Here I have designed and developed Snake game by applying engineering knowledge which provides an approach in learning or building, interesting and different, apps and games. Have identified and analyzed problem while building the game as it was a whole new experience of studying new modules like pygame, random and OS and develop a snake game with it. I have used modern tools like PyCharm to implement this project. During the development of the project I have applied professional ethics and we understood the importance of time management through the whole process of developing the project. While showcasing our project, enhanced my communication skills and displayed professional ethics which results in lifelong learning.

Snake in its basic form is a series of blocks representing a snake moving around a grid, with the player controlling the direction. It also has simple rules – when the snake touches the sides of the screen, it dies. If the snake crosses itself, it also dies. If the snake eats some food (a different type of block), it grows by 1 block. The food then re-appears at another random place on the screen.

Pygame uses the [Simple DirectMedia Layer](https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer) (SDL) library, with the intention of allowing [realhttps://en.wikipedia.org/wiki/Real-time\_computer\_graphicstime](https://en.wikipedia.org/wiki/Real-time_computer_graphics) [computer game](https://en.wikipedia.org/wiki/Computer_game) development without the [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) mechanics of the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)) and its derivatives. This is based on the assumption that the most [expensive](https://en.wikipedia.org/wiki/Computationally_expensive) functions inside games can be abstracted from the game logic, making it possible to use a [high-level programming language,](https://en.wikipedia.org/wiki/High-level_programming_language) such as Python, to structure the game.

Other features that SDL does have include vector math, [collision detection,](https://en.wikipedia.org/wiki/Collision_detection) 2D sprite [scene graph](https://en.wikipedia.org/wiki/Scene_graph) management, [MIDI](https://en.wikipedia.org/wiki/MIDI) support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing.

Applications using Pygame can run on Android phones and tablets with the use of Pygame Subset for Android (pgs4a).[[13]](https://en.wikipedia.org/wiki/Pygame#cite_note-14) Sound, vibration, keyboard, and accelerometer are supported on Android.[[14]](https://en.wikipedia.org/wiki/Pygame#cite_note-15)

In conclusion, a successful project was built in which songs will play Which you want to play after selecting and the entire playlist will be there for you.

### REFERENCE

Websites

* [https://www.codewithharry.com/videos/python-game- development](https://www.codewithharry.com/videos/python-game-%20%20%20%20%20%20%20%20%20development)
* https://www.wikipedia.org/