

Snake Game using Python

A project report submitted in partial fulfilment for the Award

Of

BACHELOR OF TECHNOLOGY

IN

Computer Science & Engineering

By

Varshith Reddy Surakanti

SECTION-K23CH

Reg.No: 12301407

TO



LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

INDIA

Acknowledgement

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. I consider myself privileged to express gratitude and respect towards all those who guided us through the completion of this project.

I convey thanks to my project guide **Mr. Aman Kumar Sir** of **Computer Science and Engineering** Department for providing encouragement, constant support and guidance which was of a great help to complete this project successfully.

Last but not the least, we wish to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn engineering. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

Abstract

This project focuses on developing a classic Snake game using Python and the pygame library. The Snake game is a popular arcade game where the player controls a growing snake to consume food items while avoiding collisions with the walls and the snake's own body. The primary objective is to achieve the highest possible score by maneuvering the snake efficiently.

Our game is designed with an intuitive and visually appealing interface that enhances the user experience. It features a checkered background, smooth snake movements, and responsive controls. The game includes sound effects for food consumption and game over scenarios, adding to the engaging gameplay experience.

The project is built on modular code principles, ensuring ease of maintenance and scalability. Future iterations may include advanced features such as different game levels, powerups, and more complex obstacles to increase the game's challenge and excitement.

This project demonstrates the practical application of Python programming and game development principles, offering an enjoyable and educational experience for players and developers alike.

Abbreviations

- GUI: Graphical User Interface

Contents

1. Introduction
2. Objective of the Project
3. Requirement Specification
4. GUI of the Project
5. Conclusion
6. References

1. Introduction

In this section, introduce the project and provide a brief overview of what the Snake game is. Mention the programming language used and the motivation behind creating the game.

Example: The Snake game is a classic arcade game developed using Python's turtle module. This project involves creating a simple yet engaging game where the player controls a snake that grows in length as it consumes food, while avoiding collisions with the walls and itself. The motivation behind this project is to demonstrate the use of Python for game development and to provide a fun and interactive way to learn programming concepts.

2. Objective of Project

Outline the main goals of the project. What are you aiming to achieve by creating this game?

Example: The primary objective of this project is to develop an interactive and user-friendly Snake game using Python. The project aims to:

- Implement fundamental game development concepts.
- Enhance programming skills through practical application.
- Provide a fun and educational tool for learning Python.
- Create an engaging user interface with intuitive controls

3. Requirements

List the software and hardware requirements needed to develop and run the game. Include any external libraries or tools used.

Example:

Software Requirements:

- Python 3.6 or higher
- turtle module (included in Python Standard Library)
- random module (included in Python Standard Library)
- time module (included in Python Standard Library)

Hardware Requirements:

- A computer with at least 2 GB of RAM
- A monitor with a resolution of 1024x768 or higher
- Keyboard for user input

4. GUI of the project :

Describe the graphical user interface of the game. Include information about the layout, design elements, and controls.

Example: The graphical user interface of the Snake game includes the following elements:

- **Game Screen:** The main playing area where the snake moves and consumes food.
- **Score Display:** A text element at the top of the screen displaying the current score.
- **Start Button:** A button to initiate the game.
- **Game Over Screen:** A message displayed when the game ends, with an option to restart the game.

Code:-

```
CSE216-Project.py ×
1  import turtle
2  import random
3  import time
4  import tkinter as tk
5
6  # Initialize global variables for the snake and game elements
7  snake = None
8  fruit = None
9  old_fruit = []
10 scoring = None
11 screen = None
12 score = 0
13 delay = 0.1
14 difficulty = "Medium"
15 paused = False # Track if the game is paused
16 pause_message = None # Variable for the "Paused" message
17
```

```
CSE216-Project.py ×
18 # Function to start the game and show difficulty selection
19 def start_game(): 1usage
20     # Hide the Tkinter main menu
21     root.withdraw()
22
23     # Show the difficulty selection screen
24     show_difficulty_screen()
25
26 # Function to show the difficulty selection screen
27 def show_difficulty_screen(): 1usage
28     # Create a new window for difficulty selection
29     difficulty_window = tk.Toplevel()
30     difficulty_window.title("Select Difficulty")
31     difficulty_window.geometry("400x300")
32     difficulty_window.configure(bg="lightblue") # Set background to lightblue
33
34     # Label for difficulty selection
35     label = tk.Label(difficulty_window, text="Select Difficulty Level", font=("Courier", 18, "bold"), bg="lightblue")
36     label.pack(pady=20)
```

```
CSE216-Project.py x
27 def show_difficulty_screen(): 1 usage
37
38 # Easy button
39 easy_button = tk.Button(difficulty_window, text="Easy", command=lambda: set_difficulty(level: "Easy", difficulty_window), width=20, height=20)
40 easy_button.pack(pady=10)
41
42 # Medium button
43 medium_button = tk.Button(difficulty_window, text="Medium", command=lambda: set_difficulty(level: "Medium", difficulty_window), width=20, height=20)
44 medium_button.pack(pady=10)
45
46 # Hard button
47 hard_button = tk.Button(difficulty_window, text="Hard", command=lambda: set_difficulty(level: "Hard", difficulty_window), width=20, height=20)
48 hard_button.pack(pady=10)
49
50 function to set the difficulty and start the game
51 set_difficulty(level, window): 3 usages
52 global difficulty, delay
53 difficulty = level
54 window.destroy() # Close the difficulty selection window
55
```

```
CSE216-Project.py x
51 def set_difficulty(level, window): 3 usages
52 # Adjust the game delay based on the difficulty
53 if difficulty == "Easy":
54     delay = 0.15
55 elif difficulty == "Medium":
56     delay = 0.1
57 elif difficulty == "Hard":
58     delay = 0.05
59
60 # Start the game with the selected difficulty
61 start_game_with_difficulty()
62
63 function to start the game with the chosen difficulty
64 start_game_with_difficulty(): 1 usage
65 global snake, fruit, old_fruit, scoring, screen, score, delay, paused, pause_message
66
67 # Initialize the turtle screen for the game
68 global screen
69 screen = turtle.Screen()
70 screen.title('SNAKE GAME')
71 screen.setup(width=700, height=700)
72 screen.bgcolor("lightblue") # Set the game background color to lightblue
73 screen.tracer(0)
74
75 # Create the game border
76 create_border()
77
78
```



```
68     def start_game_with_difficulty(): 1 usage
69
70     # Initialize score and snake setup
71     score = 0
72     snake = turtle.Turtle()
73     snake.speed(0)
74     snake.shape('square')
75     snake.color("black")
76     snake.penup()
77     snake.goto(x: 0, y: 0)
78     snake.direction = 'stop'
79
80     # Fruit setup
81     fruit = turtle.Turtle()
82     fruit.speed(0)
83     fruit.shape('circle')
84     fruit.color('red')
85     fruit.penup()
86     fruit.goto(x: 30, y: 30)
87
88     # List to store the snake's body
89     old_fruit = []
90
91     # Scoring display
92     scoring = turtle.Turtle()
93     scoring.speed(0)
94     scoring.color("black")
95     scoring.penup()
96     scoring.hideturtle()
97     scoring.goto(x: 0, y: 300)
98     scoring.write(arg: "Score :", align="center", font=("Courier", 24, "bold"))
99
```

```
CSE216-Project.py ×
68 def start_game_with_difficulty(): 1 usage
112 # Pause message setup
113 pause_message = turtle.Turtle()
114 pause_message.speed(0)
115 pause_message.hideturtle()
116
117 # Keyboard bindings
118 screen.listen()
119 screen.onkeypress(snake_go_up, key: "Up")
120 screen.onkeypress(snake_go_down, key: "Down")
121 screen.onkeypress(snake_go_left, key: "Left")
122 screen.onkeypress(snake_go_right, key: "Right")
123
124 # Pause button binding
125 screen.onkeypress(toggle_pause, key: "space") # Press 'Space' to toggle pause
```

```
68 def start_game_with_difficulty(): 1 usage
127 # Main game loop
128 game_loop()
129
130 unction to create the game border
131 create_border(): 1 usage
132 turtle.speed(5)
133 turtle.pensize(4)
134 turtle.penup()
135 turtle.goto(-310, y: 250)
136 turtle.pendown()
137 turtle.color('black')
138 turtle.forward(600)
139 turtle.right(90)
140 turtle.forward(500)
141 turtle.right(90)
142 turtle.forward(600)
143 turtle.right(90)
144 turtle.forward(500)
145 turtle.penup()
146 turtle.hideturtle()
147
148 unctions to control snake movement
149 snake_go_up(): 1 usage
150 if snake.direction != "down":
151     snake.direction = "up"
152
153 snake_go_down(): 1 usage
154 if snake.direction != "up":
155     snake.direction = "down"
```

```
156
157 snake_go_left(): 1 usage
158 if snake.direction != "right":
159     snake.direction = "left"
160
161 snake_go_right(): 1 usage
162 if snake.direction != "left":
163     snake.direction = "right"
164
165 snake_move(): 1 usage
166 if snake.direction == "up":
167     y = snake.ycor()
168     snake.sety(y + 20)
169
170 if snake.direction == "down":
171     y = snake.ycor()
172     snake.sety(y - 20)
173
174 if snake.direction == "left":
175     x = snake.xcor()
176     snake.setx(x - 20)
177
178 if snake.direction == "right":
179     x = snake.xcor()
180     snake.setx(x + 20)
181
```

```

CSE216-Project.py ×
182 toggle_pause
183 toggle_pause(): 1 usage
184 global paused
185 paused = not paused
186
187 if paused:
188     # Display the "Paused" message in bold below the score with dark red color
189     pause_message.clear() # Clear any previous message
190     pause_message.penup()
191     pause_message.goto(0, 260) # Position it directly below the score
192     pause_message.color("darkred") # Dark red color
193     pause_message.write("Paused", align="center", font=("Courier", 24, "bold"))
194 else:
195     # Hide the "Paused" message when unpaused
196     pause_message.clear()

```

```

CSE216-Project.py ×
197
198 game over function
199 game_over(score): 2 usages
200 screen.clear()
201 screen.bgcolor('lightblue') # Set background to lightblue on game over
202 scoring.goto(0, 0)
203 scoring.write(f"GAME OVER\nYour Score: {score}", align="center", font=("Courier", 30, "bold"))
204
205 main game loop
206 game_loop(): 1 usage
207 global score, delay, old_fruit, paused
208
209 while True:
210     screen.update()
211
212     if paused:
213         time.sleep(0.1) # Pause the game
214         continue
215
216     # Snake and food collision detection
217     if snake.distance(fruit) < 20:
218         x = random.randint(-290, 270)
219         y = random.randint(-240, 240)
220         fruit.goto(x, y)
221         score += 1
222         scoring.clear()
223         scoring.write(f"Score: {score}", align="center", font=("Courier", 24, "bold"))
224         delay -= 0.001
225

```

CSE216-Project.py ×

```
206 def game_loop(): 1 usage
226     # Add new segment to the snake's body
227     new_fruit = turtle.Turtle()
228     new_fruit.speed(0)
229     new_fruit.shape('square')
230     new_fruit.color('red')
231     new_fruit.penup()
232     old_fruit.append(new_fruit)
233
234     # Move the snake's body
235     for index in range(len(old_fruit) - 1, 0, -1):
236         a = old_fruit[index - 1].xcor()
237         b = old_fruit[index - 1].ycor()
238         old_fruit[index].goto(a, b)
```

CSE216-Project.py ×

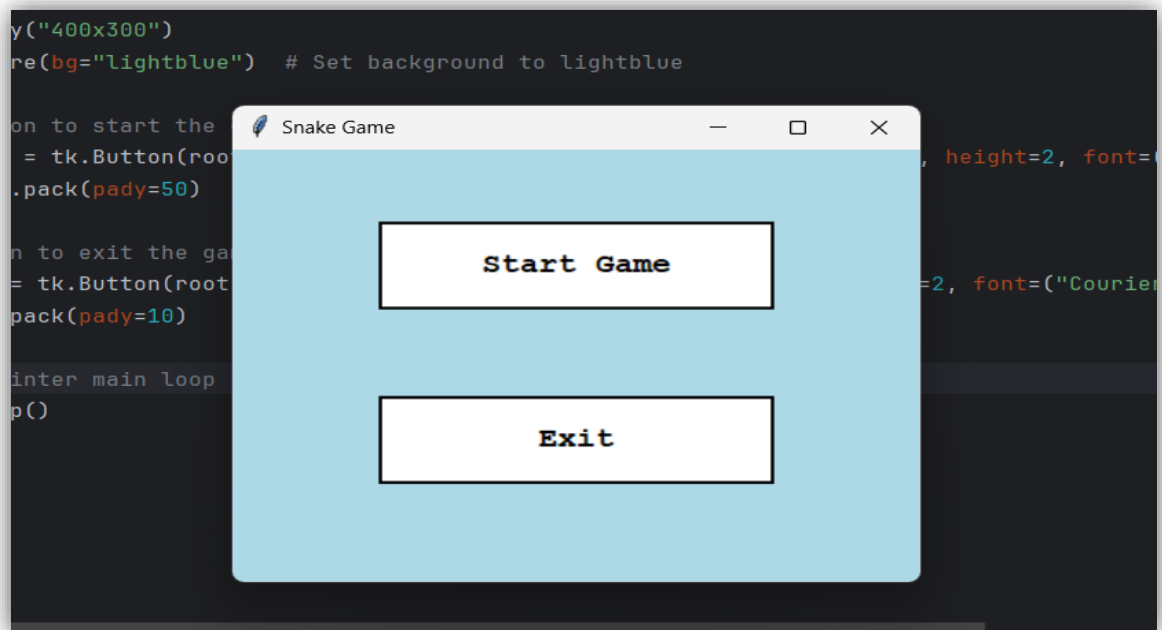
```
206 def game_loop(): 1 usage
238     old_fruit[index].goto(a, b)
239
240     if len(old_fruit) > 0:
241         a = snake.xcor()
242         b = snake.ycor()
243         old_fruit[0].goto(a, b)
244
245     snake_move()
246
247     # Check for snake collisions with borders
248     if snake.xcor() > 280 or snake.xcor() < -300 or snake.ycor() > 240 or snake.ycor() < -240:
249         game_over(score)
250         break
251
252     # Check for collision with itself
253     for segment in old_fruit:
254         if segment.distance(snake) < 20:
255             game_over(score)
256             break
257
258     time.sleep(delay)
```

```
CSE216-Project.py x
206 def game_loop(): 1 usage
252     # Check for collision with itself
253     for segment in old_fruit:
254         if segment.distance(snake) < 20:
255             game_over(score)
256             break
257
258     time.sleep(delay)
259
260 kinter window setup
261 t = tk.Tk()
262 t.title("Snake Game")
263 t.geometry("400x300")
264 t.configure(bg="lightblue") # Set background to lightblue
265
266 start button to start the game
267 rt_button = tk.Button(root, text="Start Game", command=start_game, width=20, height=2, font=("Courier", 14, "bold"), bg="white", fg="black", r
268 rt_button.pack(pady=50)
269
270 exit button to exit the game
271 t_button = tk.Button(root, text="Exit", command=root.quit, width=20, height=2, font=("Courier", 14, "bold"), bg="white", fg="black", r
272 t_button.pack(pady=10)
273
274 on the Tkinter main loop
275 t.mainloop()
```

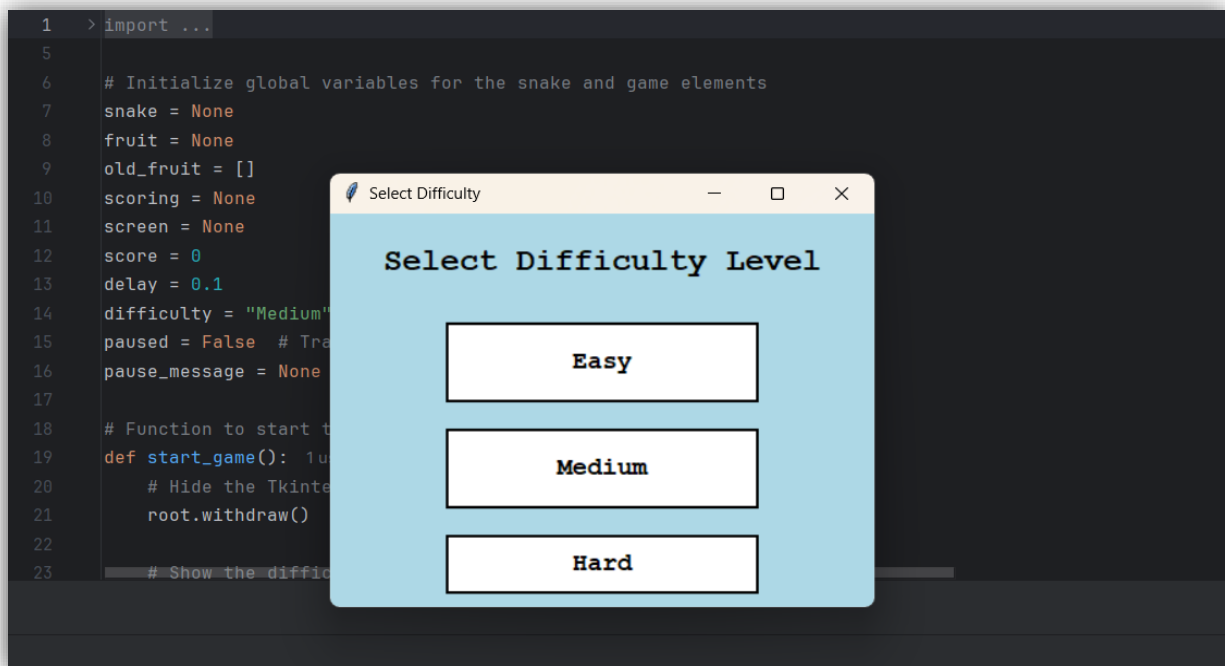
Outputs:-

Snake Game interface:

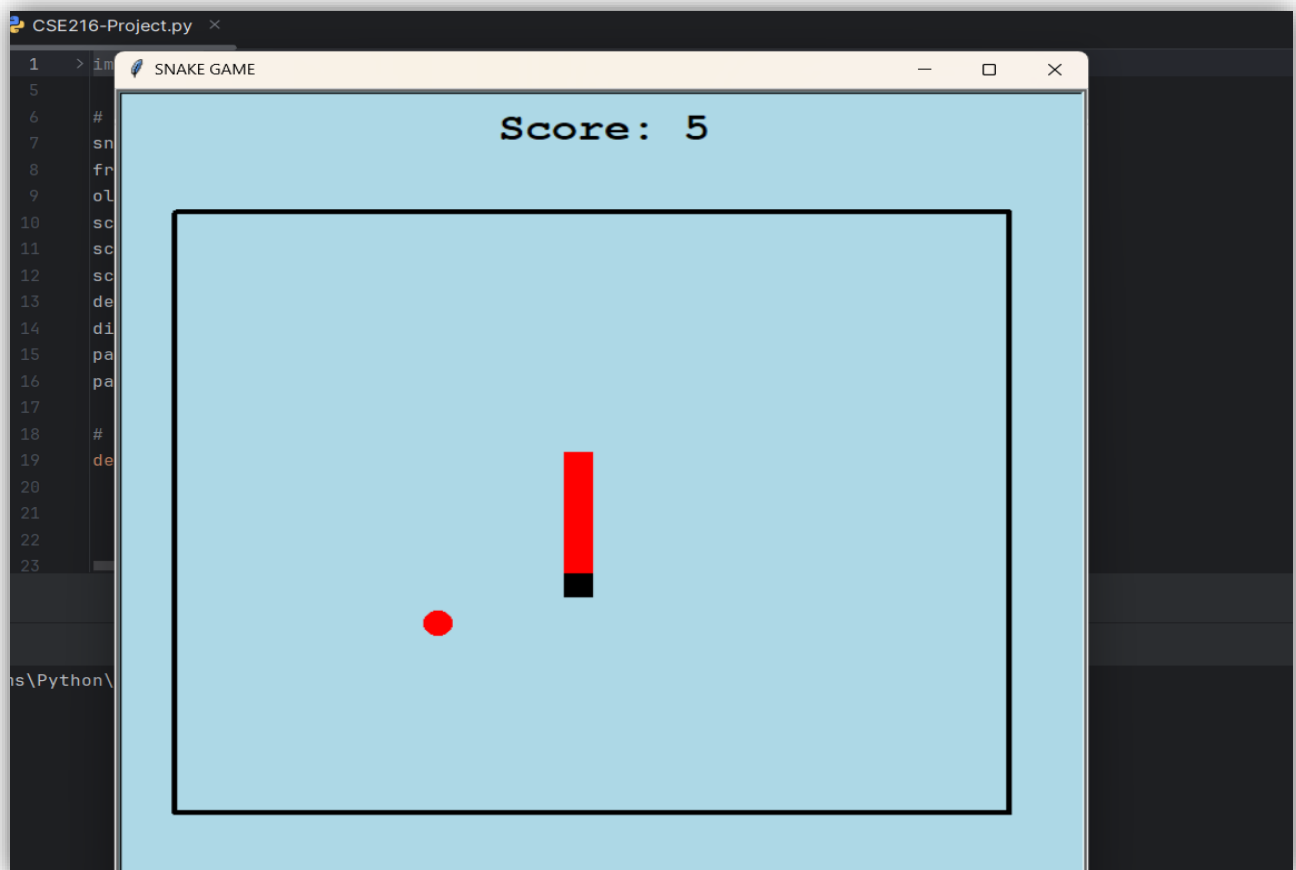
1.



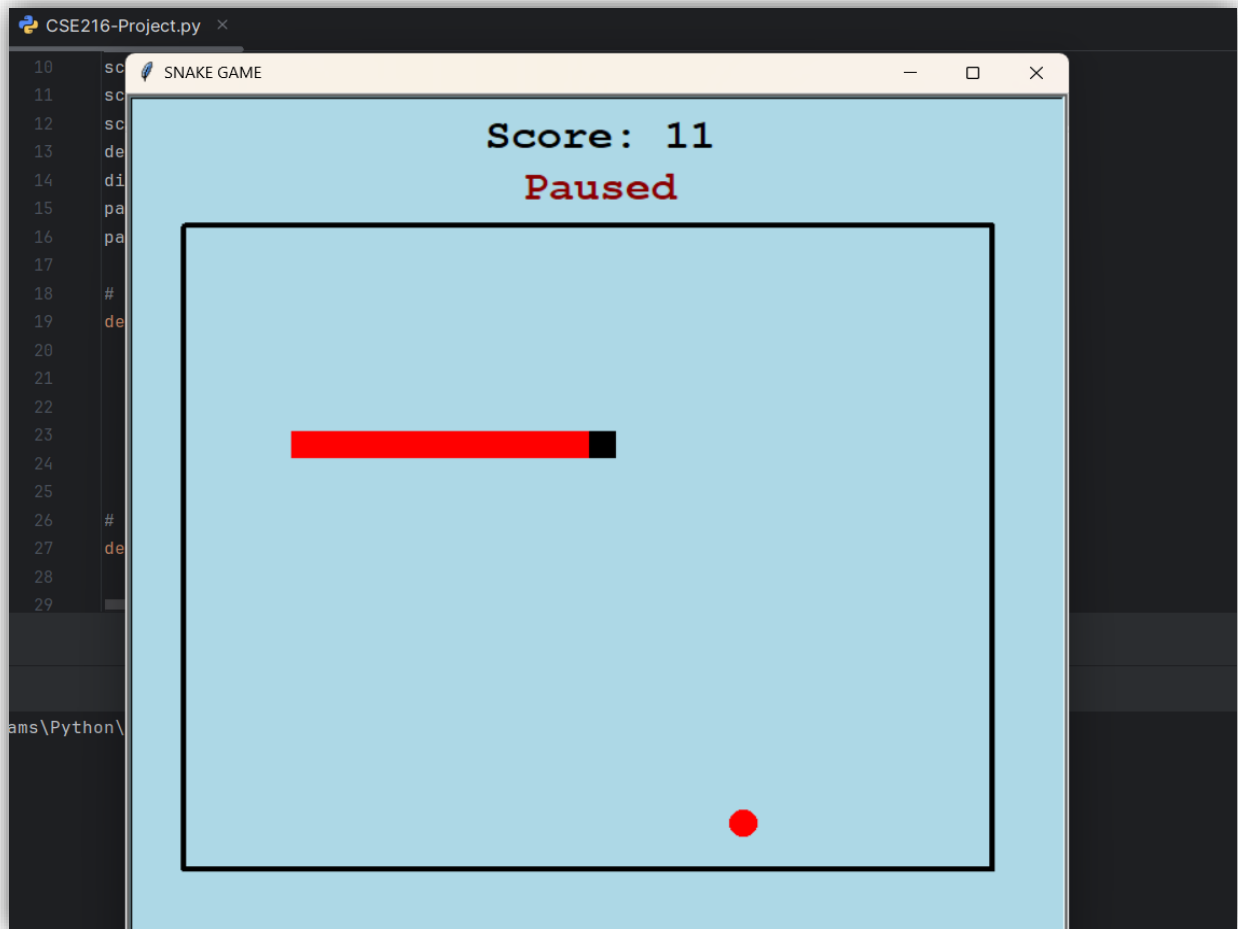
2.



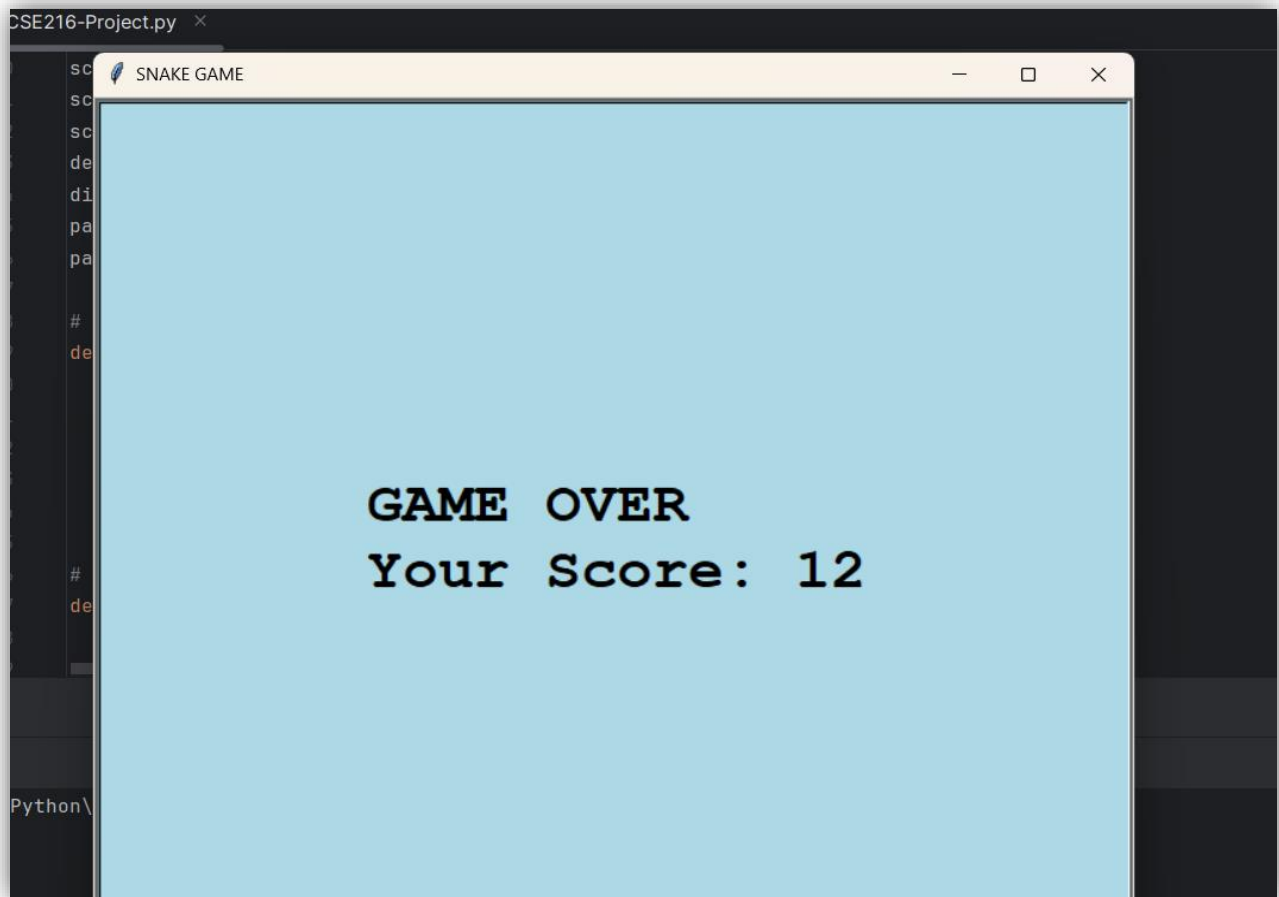
3.



4.



5.



5. Conclusion :

Summarize the project's achievements and its potential impact. Discuss any future improvements or extensions that could be made.

Example: The Snake game project successfully demonstrates the use of Python for creating interactive games. It provides an enjoyable way to learn programming concepts and enhances problem-solving skills. Future improvements could include adding sound effects, different levels of difficulty, and more sophisticated graphics.

6. References

List any references or resources you used during the project, such as tutorials, documentation, or books.

Example:

- Python Official Documentation: <https://docs.python.org/3/>
- turtle module documentation: <https://docs.python.org/3/library/turtle.html>
- Stack Overflow: Various threads and discussions