**Final Report**

SNAKE GAME USING PYTHON



In partial fulfilment for the requirements of the award of the degree of

“**INT213: Python programming**”

“**School of Computer Science**”

Lovely Professional University Jalandhar, Punjab

India

**Submitted by**

Adarsh Agrawal

Reg no. 12003841

Jatin Khandelwal

Reg no. 12013591

ABSTRACT

Now a days, the technology is developing at a very fast pace. Various new techs are releasing day by day. In this growing time, computer games have also shown a good potential in this competitive era.

Earlier, it was very hard to make simple 2D games. But due to development in programming languages, it is now easy.

With python 3.9, we can make 2D games in just some lines of codes without even need of external files.

This project is based on a famous 2D game which we all are familiar with. It is the snake game which we saw on small keypad phones.

ACKNOWLEDGEMENT

I would like to express my gratitude to my teacher Md. Imran Hussain sir who gave me the opportunity to do this wonderful project. The project helped me learn how to do proper Research and I learned about many new things while doing the project.

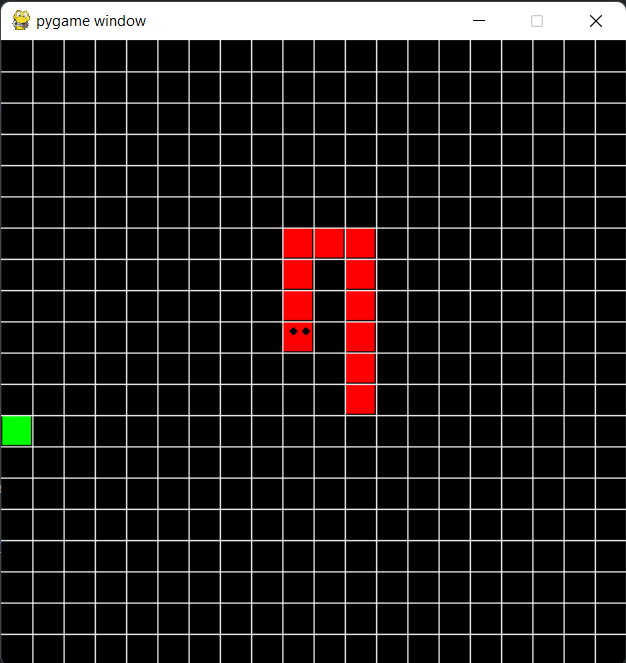
TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| SL.no. | Content | Page no. |
| 1 | Abstract | 2 |
| 2 | Acknowledgement | 2 |
| 3 | Introduction | 3 |
| 4 | Making of project | 4 |
| 5 | Screenshots | 12 |
| 6 | Resources | 13 |
| 7 | Reference | 13 |

INTRODUCTION

We all have heard about the Snake game. Yes, it’s a 2D snake game whose aim is simple; the snake has to eat the available fruit to grow larger and larger without eating itself, the score rises as the snake gets larger. The first snake which is given in the starting of the game is a simple square box and for each fruit he eats, one box gets added to his body.

Here is a screenshot of the actual game which I made.



The snake can be controlled using the arrow keys in the keyboard. So, the game is interesting, isn’t it? We have made this game using python.

To make this game using python, we used three assets of python –

PyGame library, random module and Tkinter.

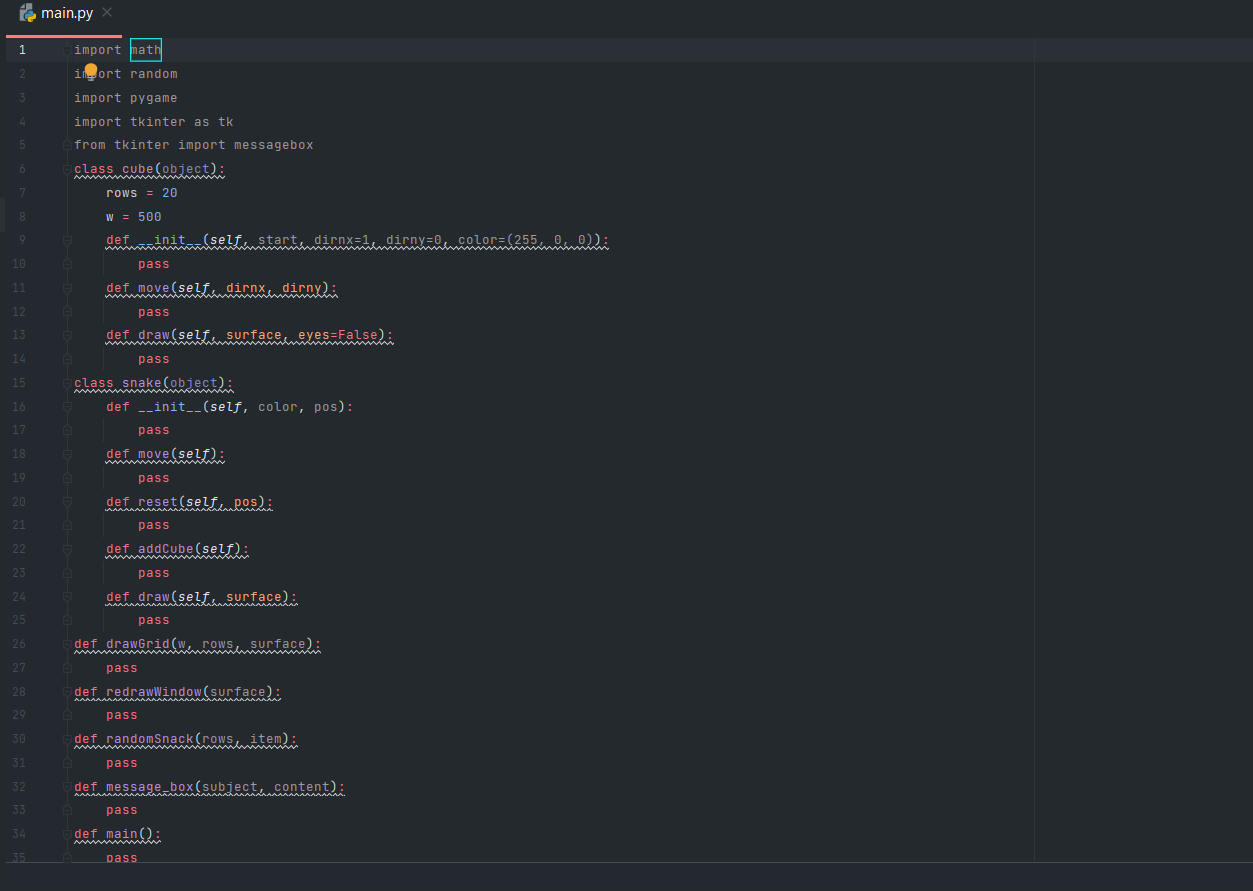
The pygame library is a cross-platform set of python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the python programming language.

The random module is a built-in module to generate the pseudo random variables. It can be used to get a random number, selecting random elements from a list, shuffle elements randomly, etc.

Tkinter is a python binding to the tk GUI toolkit. It is the standard python interface to the Tk GUI toolkit.

MAKING THE PROJECT

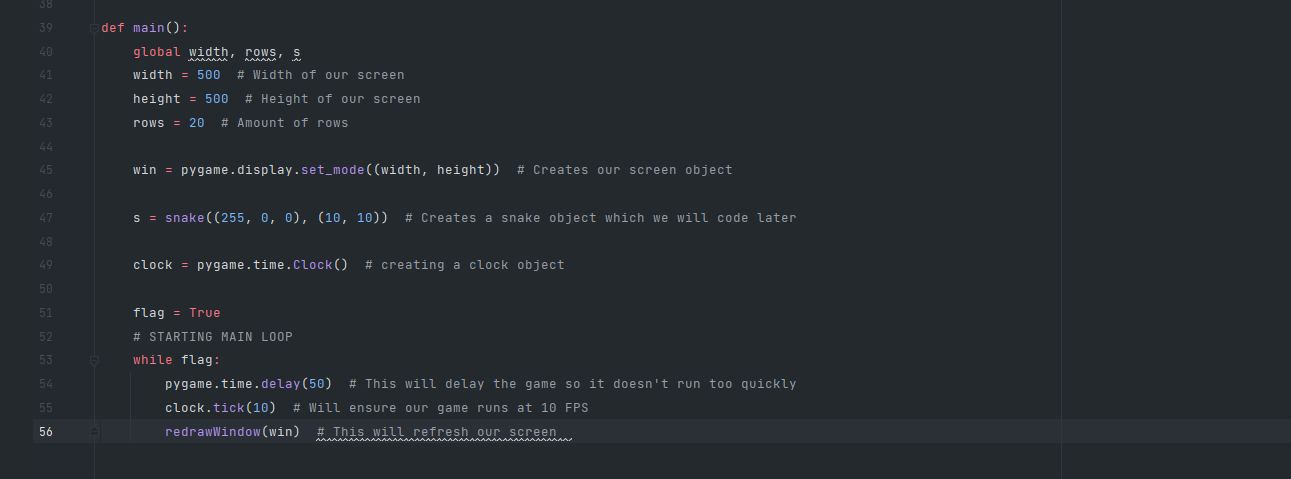
So, first we have already set up the main structure for our game. We used two main classes (snake and cube).



**Creating our Game loop**

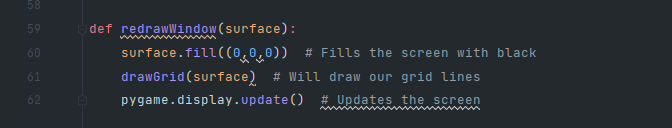
In our game, there is a “main loop” or “game loop”. This loop runs continuously until the game is exited. It is mainly responsible for checking for events and calling functions and methods based on those events.

We coded the game loop inside the main() function. We declared some variables at the top of the function then move into our while loop which represent our game loop.



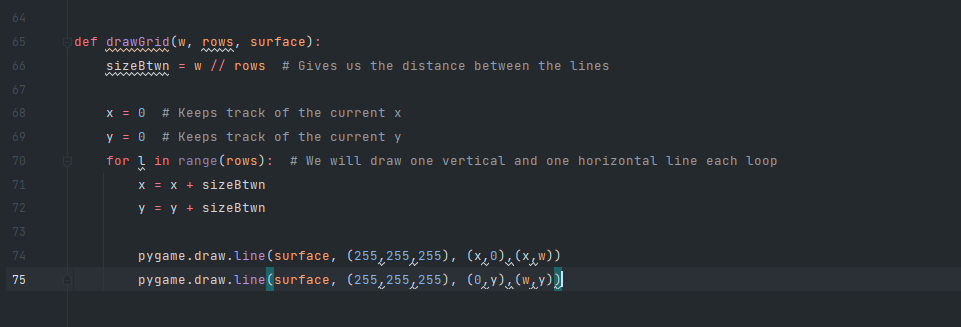
**Updating the Screen**

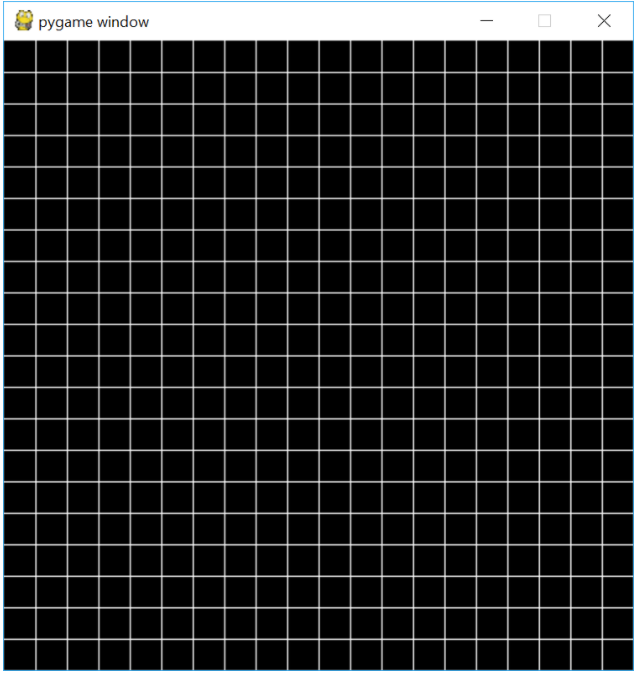
We used **redrawWindow** function to update the display. We called this function once a frame from out game loop.



**Drawing the Grid**

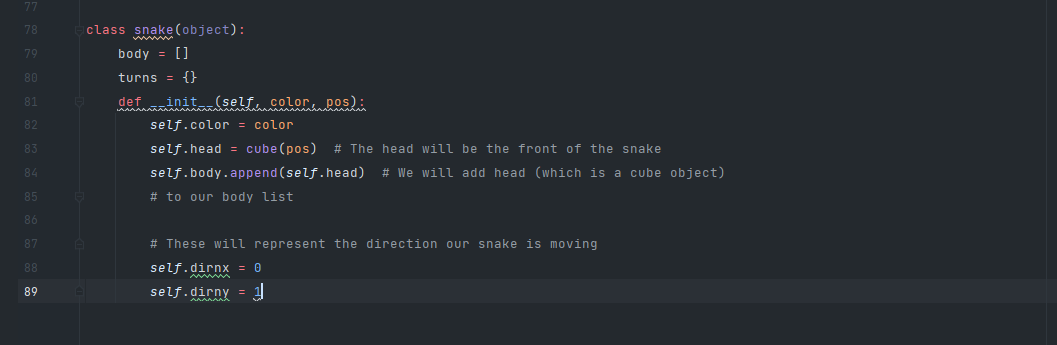
We made a grid that contains lines to represent 20\*20 grid. We did this in **drawGrid** function.





**Starting the Snake Class**

Our snake object contains a list of cubes which represent the snake body. We stored these cubes in a list called body which is a class variable. We also have class variable called turns.

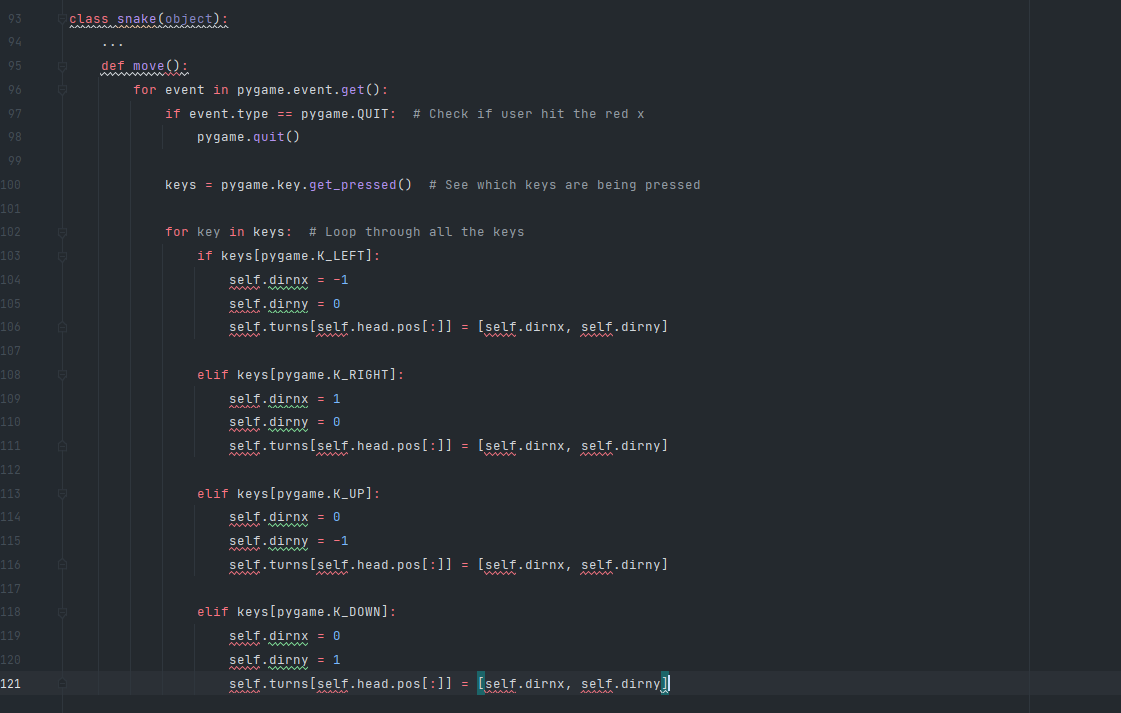


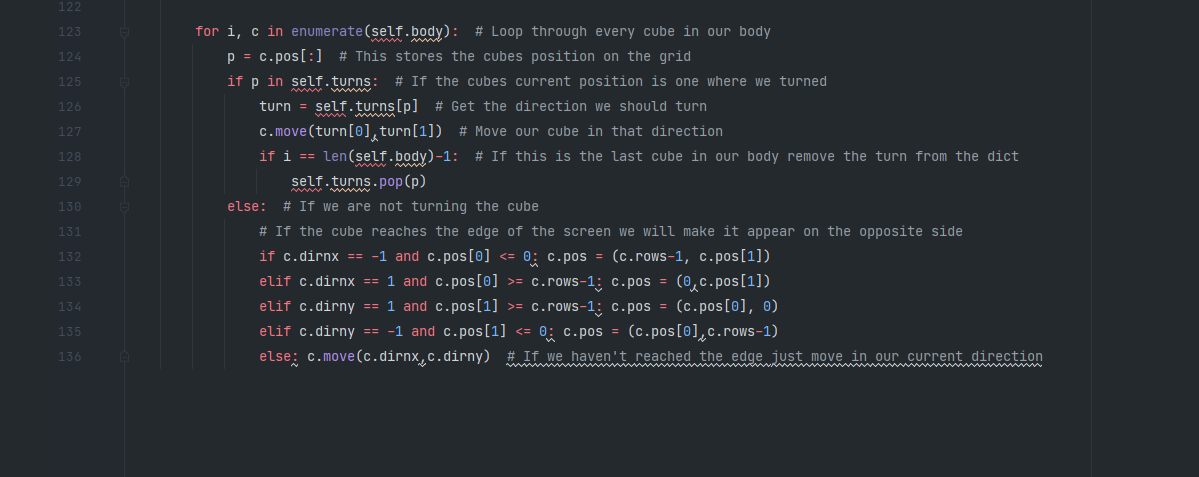
**Snake class – Move method**

We coded the snake class by completing the **move()** method.

This method checks for any events.

Turning the snake is little bit complicated but possible. We have to keep track where the snake needs to turn so that the new cube forms up in that space. Whenever we needed to turn, we added the position of our head to a turn dictionary where the value is the direction we turned. This way the other cubes reach this position we know which way to turn them.

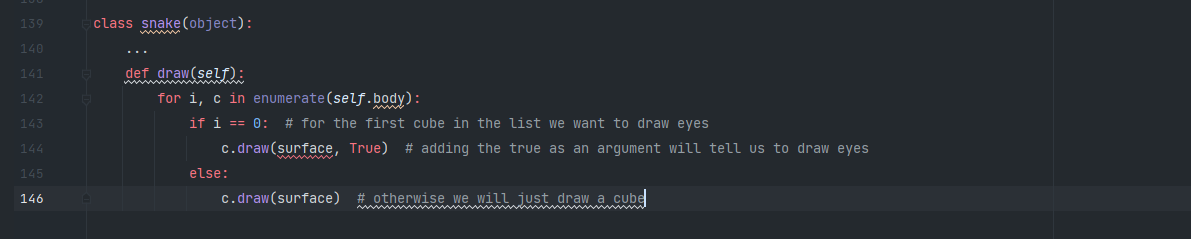




**Drawing the Snake**

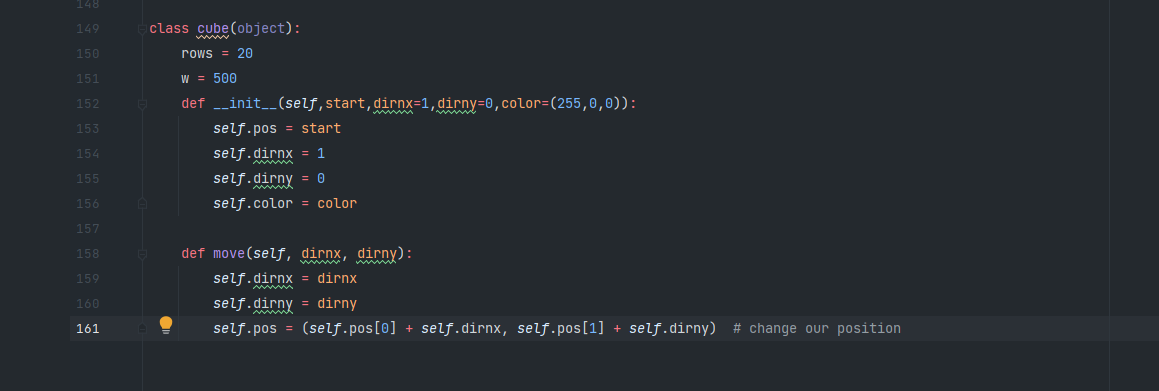
To draw the snake, we draw each cube object in the body.

We did this in the **draw()** method.



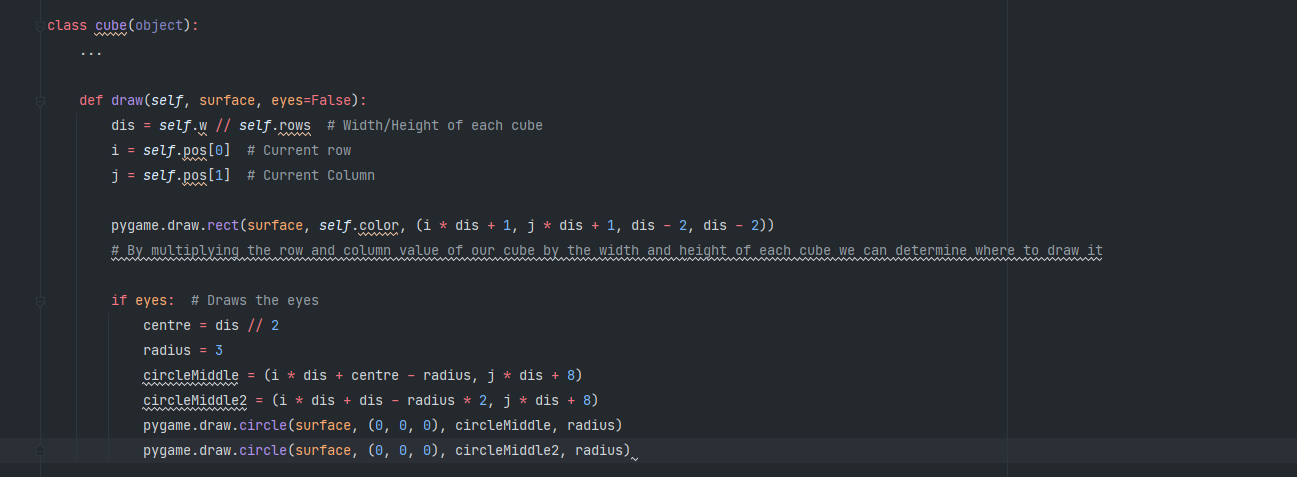
**Starting the Cube Class**

We started the cube class by coding in **move() and \_\_init\_\_()** methods.

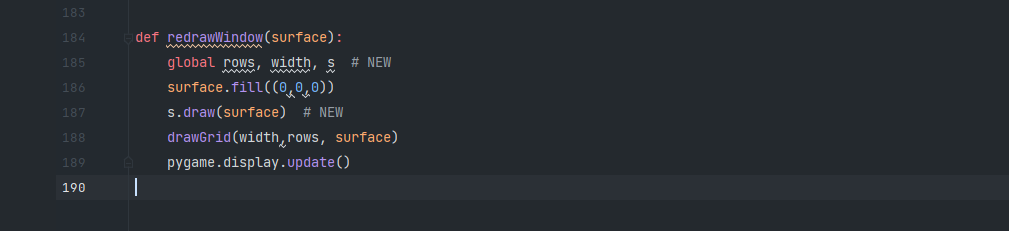


**Cube Class – Draw method**

Now we drew our cubes inside of the **draw()** method. This involves a bit of math as we have 20 positions in our and a screen with dimensions of 500\*500. We needed to determine where to draw each cube so that it appears within the proper space on the grid.

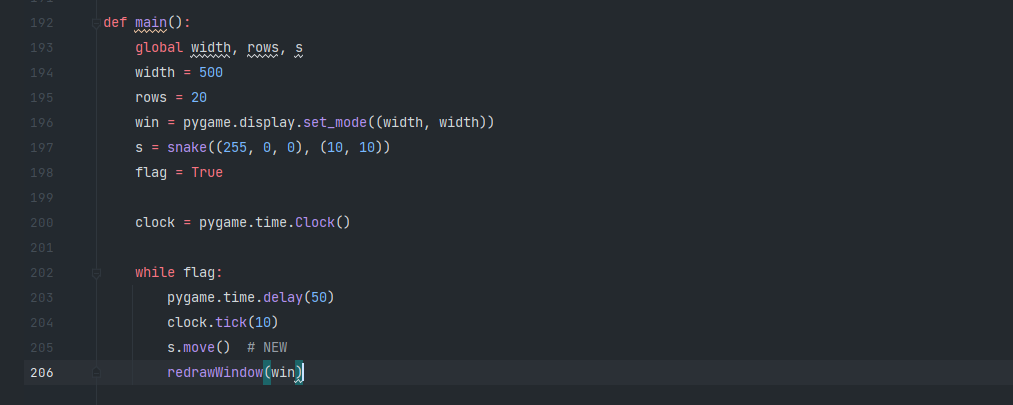


Now, we actually see anything in the screen, we add something to the **redrawWindow** function.



**Move Our Snake**

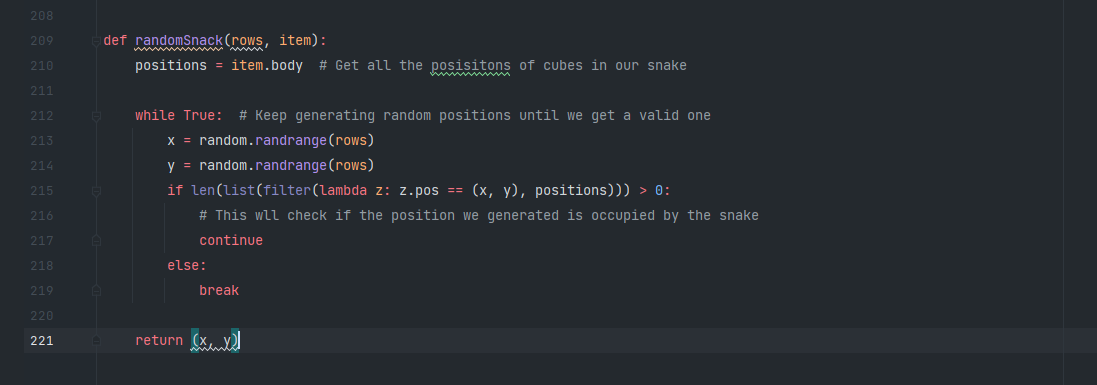
To allow the snake to move, we called **s.move().**



**Adding Cubes**

We need to make something to eat for the snake. Every time we collide with one of these objects, we add a new cube to the end of the snake.

The first step is to generate a position for the snake. We did this inside a **randomSnack()** function.



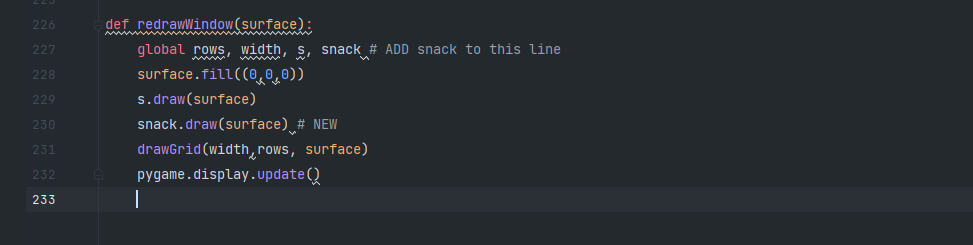
Now that we have a function for creating a random snack, we should use it! We added the following line into the **main()** function before the while loop.



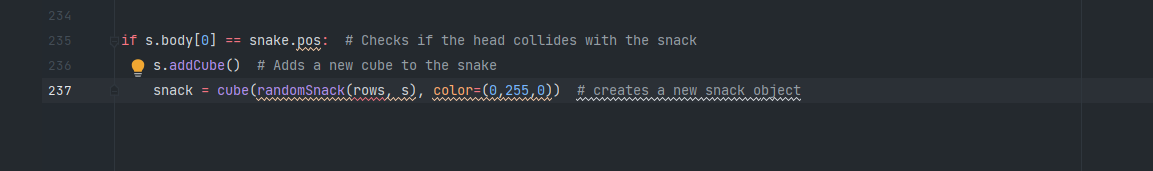
**Collision With Snack**

We have created the snack; we needed to draw it and see if the head of the snake collides with it.

Our **redrawWindow()** function should now look like this:

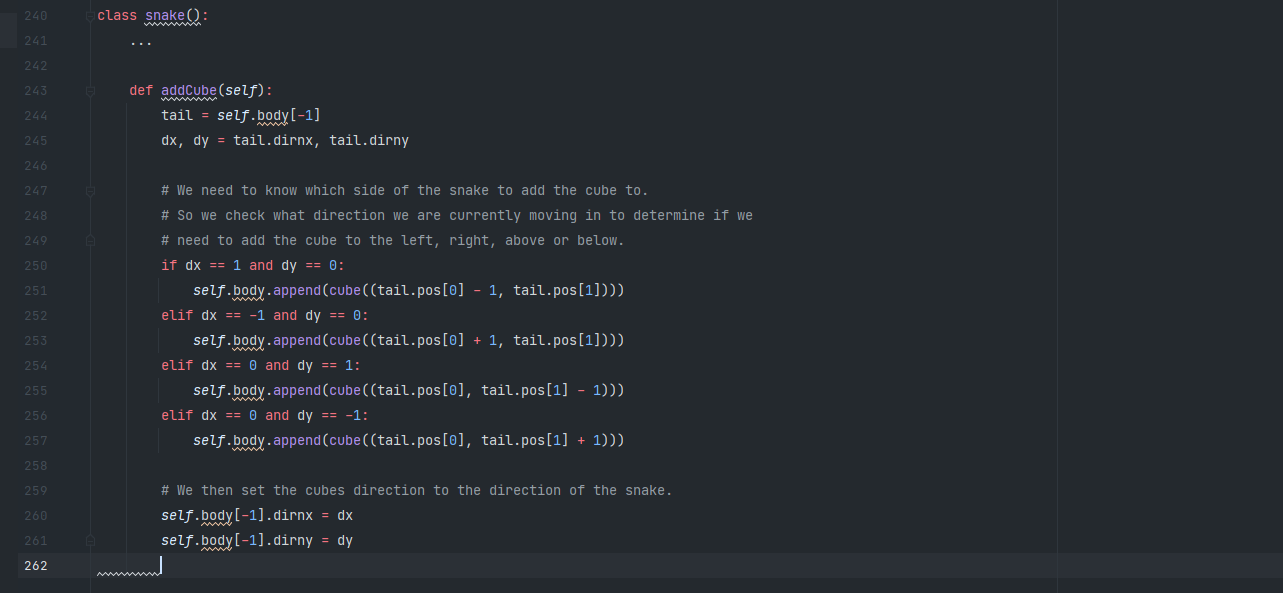


To check for collision, we added the following lines into our game loop within the main() function.



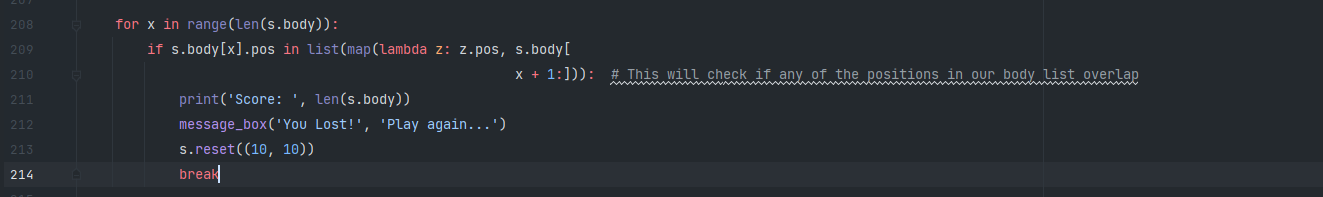
**Snake Class – addCube() Method**

The last step to add a cube is to add it in the end of our snake object. We did this using **addCube()** method.



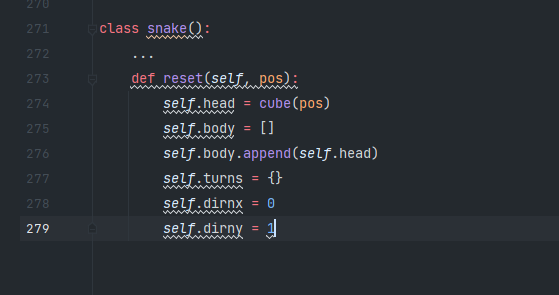
**Ending the Game**

We lose the game when our object collides with itself. To check this, we added the following line to our main() function inside the game loop.



**Snake Class – reset() Method**

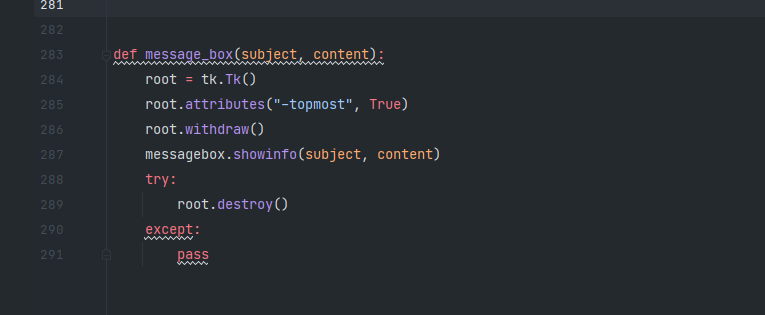
Now we coded the reset() method from within the snake class. This helps to reset the game so we can play it again after we lose.



**Displaying a Message Box**

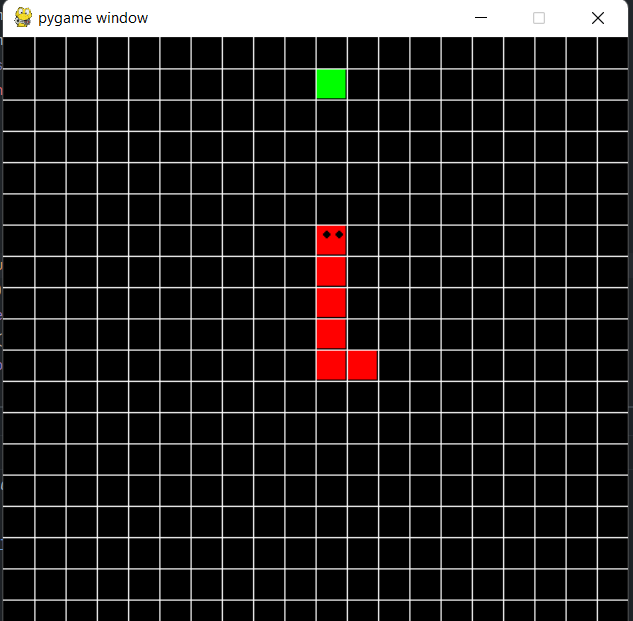
When the user looses, we want to display a message box telling them they lost.

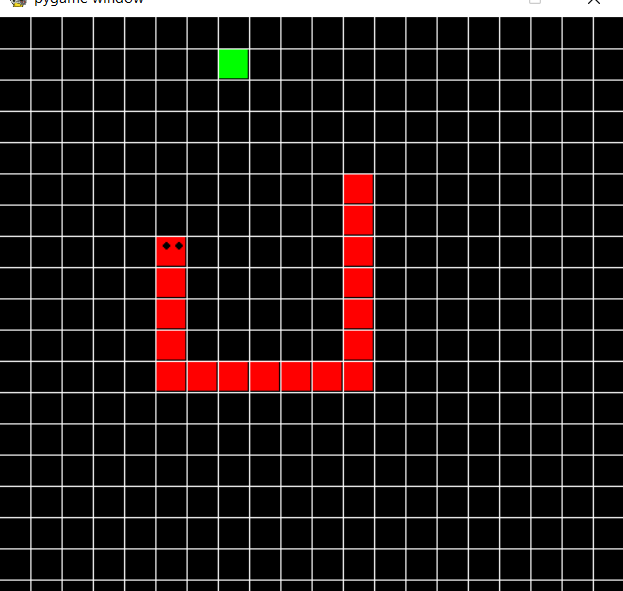
We used tkinter and inside of the message\_box() function.



So, finally the game is finished.

Here are some **screenshots of the game**.





CONCLUSION

The snake is a 2D fun to play game. We have made this game using python 3.9. This game can be played on any configuration and only 200 lines of code are used to make this game.

We have learned many things making this project.

We leant the use of various modules in pygame.

We used random module.

We learnt how to use Tkinter methods.

**Resources**

Python 3.9

Pygame Library

Random module

Tkinter

Pycharm IDE

**Reference**

<https://stackoverflow.com/>

<https://www.techwithtim.net/>

<https://www.pygame.org/>

<https://docs.python.org/3/library/tkinter.html>