**PYTHON PROJECT REPORT**

**Snake Game Development Using Pygame**

**As a project work for course-**

**“Python Programming (INT213)”**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**



**Team Size – 02**

**Team Member 01-**

**Name – Aaditya Raj**

**Registration No.- 12004802**

**Team Member 02-**

**Name – Nisha Bisht**

**Registration No.- 12001718**

**Table of Content**

* **Acknowledgement**
* **Introduction to the Project**
* **Context**
* **Objective of the project**
* **Libraries used in the project**
* **Working with the project**

**ACKNOWLEDGEMENT**

The success and outcome of this project required a lot of guidance and assistance from many people. All that we have done is only due to such supervision and assistance and we will not forget to thank them.

We respect and thank our course teacher **Prof. P. Raja Sir** for providing us with an opportunity to do the Python Project and giving us all the support and guidance, which made us complete the project.

We would also like to underscore dynamic efforts of the teamwork and their advice, feedback and input/ contributions on this project.

We came to know about so many new things, and we are thankful to them.

* Aaditya Raj (12004802)
* Nisha Bisht (12001718)

**Introduction to the Project**

* + **Context: -**

This project has been done as part of our course Python Programming (INT213) for the completion of degree BTech CSE at Lovely Professional University. Supervised by Prof. P. Raja Sir, we had almost two months to fulfil the requirements to succeed in the module.

* + **Objective of the Project: -**

The main aim of this project is to develop a snake game using pygame module. Snake game is one of the most popular arcade games of all time. The snake in this Snake game is controlled using the four direction buttons relative to the direction it is headed in. In this game, the main objective of the player is to catch the maximum number of fruits or food without hitting the wall or itself.

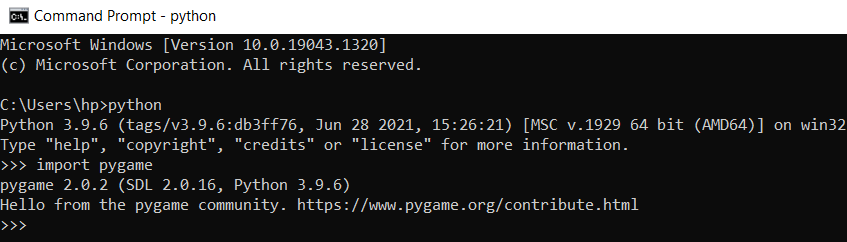
* **Libraries/Modules Used: -**
* **Pygame-**

1. Pygame is a cross-platform set of Python modules which is used to create video games.
2. It consists of computer graphics and sound libraries designed to be used with the Python programming language.

* Installing Pygame-

**Step-1:** Type “pip install pygame” in your windows PowerShell or terminal and your “pygame” module will be installed.

**Step-2:** Type “python” to check which python version you are using.



**Step-3:** Type “import pygame” to import pygame.

* **Random-**

Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. This module can be used to perform random actions such as generating random numbers, printing random value for a list or string, etc.

**Working with the Project**

* **Creating The Game Window:**
* **import pygame**

It will import pygame library along with its all-existing modules.

* **pygame.init()**

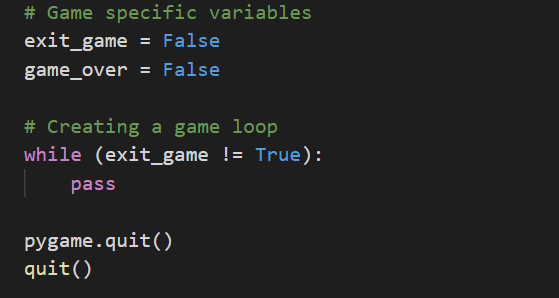
Initializing all modules of pygame.

* **screen\_height = 500**

**screen\_width = 900**   **gameWindow = pygame.display.set\_mode((screen\_width, screen\_height))**

gameWindow is a variable and set\_mode is a function in pygame which is taking the size of window as arguments (as a tuple)

* **Setting the caption name of pygame window:**
* **display.set\_caption("Snake Game")**
* **Creating the Game Loop:**

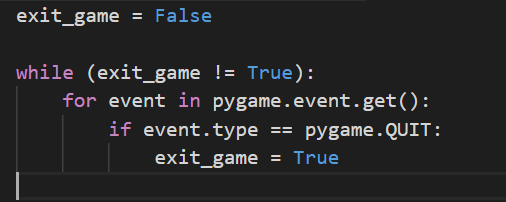


Here we are running the **exit\_game = False** loop which means this code will run until exit\_game variable become true. Here pygame.quit() is written so that the pygame will be quitted. After that quit() will exit the program.

* **Exit from game loop/ Quitting the game window:**

**KEYUP in Pygame:** When the keyboard buttons are released.

**KEYDOWN in Pygame:** When the keyboard buttons are pressed.



In this we first used an “if” statement to check if event’s type is “QUIT” and if it’s “QUIT” then exit\_game will become **True**. “exit\_game” is running the while loop, as soon as exit\_game will become True, it will terminate the while loop.

* + **Creating Ground for our Snake Game:**
  + **Background Color Variables:**

We will make a tuple and fill RGB (Red, Green, Blue) values in it. RGB values are out of 255.

**white = (255, 255, 255)**  **red = (255, 0, 0)**  **black = (0, 0, 0)**

**blue = (0, 0, 255)**

* **fill():**

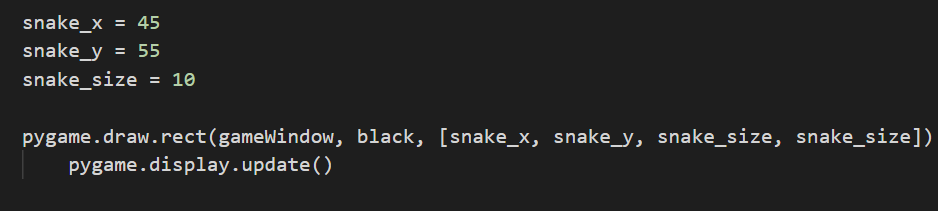
Fill function is used to fill the screen with some color. It takes color defined in RGB form as an argument.

**For example:**

**gameWindow.fill(white)**

**pyame.display.update() #** It will update our display to show changes.

* + **Creating the head of snake:**



**snake\_x and snake\_y** are the a and y coordinates of our snake’s head in the **gameWindow** and **snake\_size** is the length and breadth of **snake’s head.**

**The rect() function takes three arguments.**

The first one is the surface.

Second is the color of snake’s head.

Third is a list in which we told where the rectangle should be placed(co-ordinates) and length and breadth of the rectangle.

* + **Setting Game FPS:**

Fps means frame per second.

**clock = pygame.time.Clock()**

For framerate we will make a variable “fps”.

**fps = 45**

Code to make tick:

**clock.tick(fps)**

The tick function takes framerate as an argument. It updates the frame according to the frames we want per second. When we click on a key, we want our snake to move continuously with a defined speed, clock does that.

* + **Giving velocity to our snake:**

Speed means it is covering distance. In our game distance is in co-ordinates (x and y) which are snake\_x and snake\_y. So, it just means that we have to keep increasing values of co-ordinates like it’s in a loop. We already are in a **while loop,** so we just have to increment values.

velocity\_x = 4

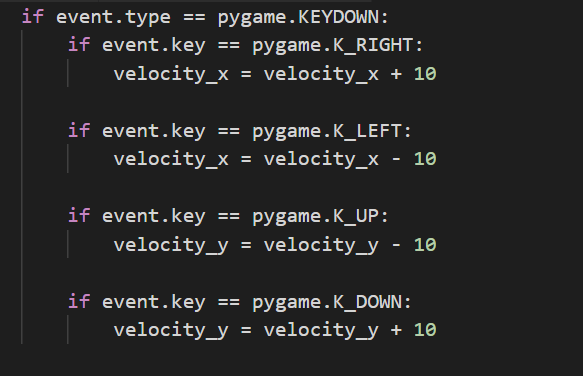
velocity\_y = 4

snake\_x = snake\_x + velocity\_x

snake\_y = snake\_y + velocity\_y

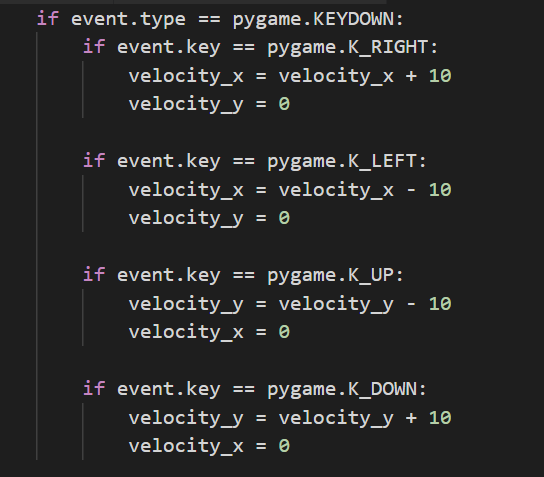
* + **Making Snake Move in Arrow Key Press:**
* **Making Controls for All Directions:**

For every key there is an event which verifies that which key is pressed. We simply checked which key is pressed and added/subtracted in x/y co-ordinates according to that.



* **Making It Move Straight:**

In order make sure that the snake should not move diagonally just null the other velocity when one is in play.



* + **Creating Food for Snake:**
* **Getting Random Number:**

To get a random number we will use random module’s “randint” function. “rand” stands for random and “int” means integer. It takes a range (start and end) as argument and returns a random number between that range.

In our game we want to place food in the game window so it should choose random number between 0 and game width and height. We made 2 variables for game width and game height, **screen\_width and screen\_height.**

food\_x = random.randint(0, screen\_width)

food\_y = random.randint(0, screen\_height)

* **Making Food:**

Making food is the same as creating the head of snake.

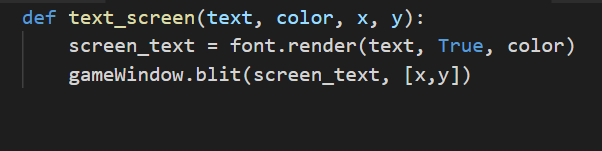
**pygame.draw.rect(gameWindow, red, [food\_x, food\_y, snake\_size, snake\_size])**

* + **Displaying Score in Game Window:**
* **For Font:**

font = pygame.font.SysFont(None, 45)

It takes two arguments, the first one is the name of font and the second is font size. We have given “None” as font name for default font.

* **Adding Score:**

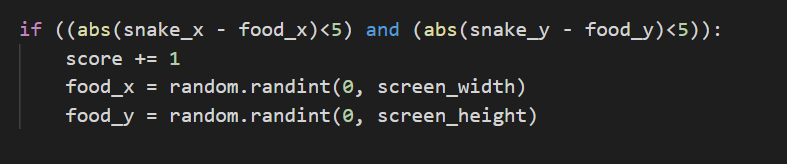


We are taking **text, color and co-ordinates** (of where text should be placed) as argument and then we are using the render function which takes text, color and antialias(bool) as argument. Then we are using **blit**, blit means **a logical operation in which a block of data is rapidly moved or copied in memory.**

**And finally,**



* + **Replotting Food:**

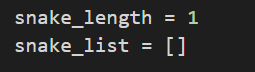


The food will be replotted as soon as the snake almost overlaps (according to the **if** **condition**) with the food.

* + **Increasing Snake Length:**

We made two variables, **snake\_length** and **snake\_list.**

**snake\_length** will have an integer value (with an initial value of 1) and we will increment it’s value every time our snake eats food.

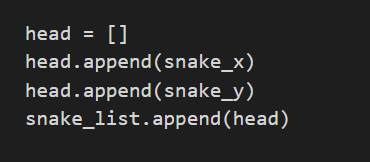


And **snake\_list** is a list of lists. It will have co-ordinates of the snake’s rectangles.

**Incrementing value:**

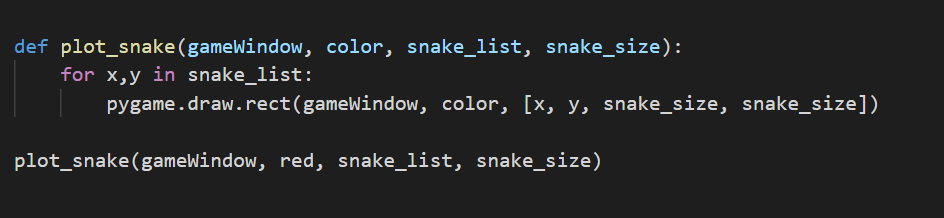
snake\_length += 5

* + **Appending in Snake List:**



Here we are just appending **snake\_x and snake\_y** in head and then appending head in snake\_list.

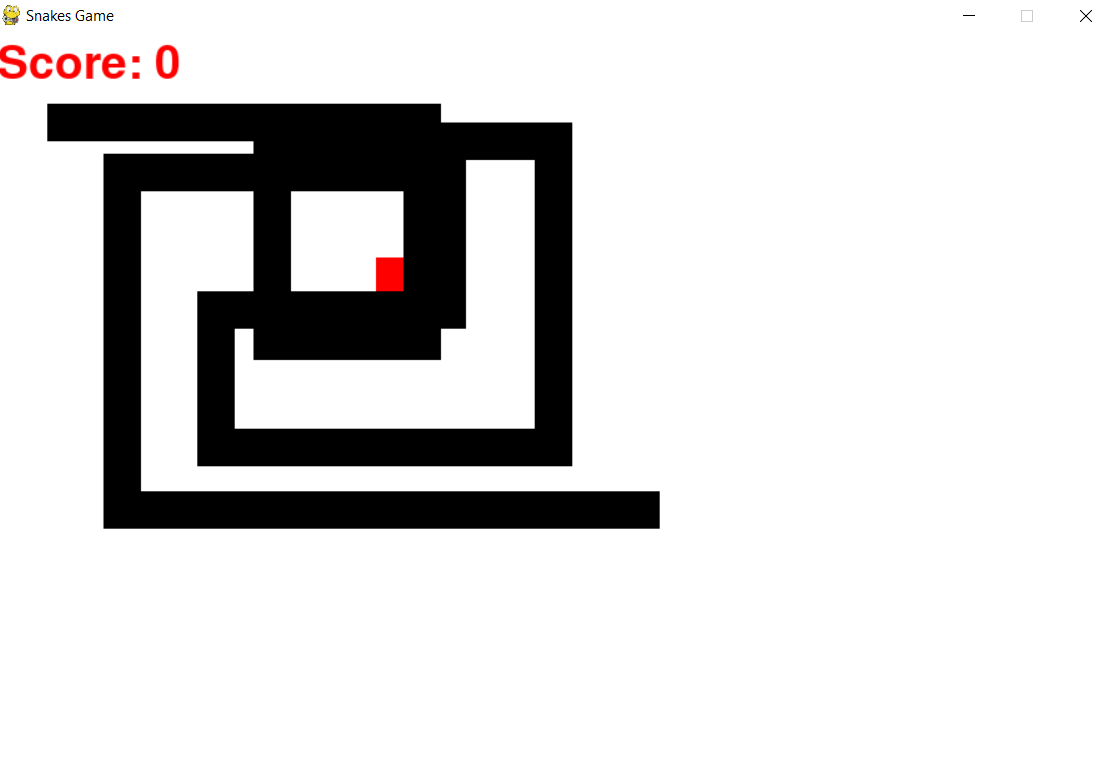
* + **Plot Snake:**



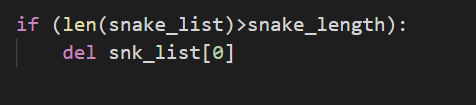
Here we are taking 4 arguments which are required to make a rectangle, iterating **snake\_list** and simply just putting those values in **rect().**

* **Deleting old rectangles:**

If we won’t delete the old rectangles, our program will look like this.



To delete old rectangles, we just have to keep deleting 0th item of **snake\_list** whenever length of **snake\_list** becomes greater than **snake\_length**.



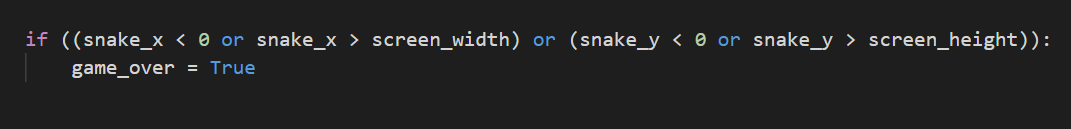
* + **Handling Game Over and Collision:**
  + **Game Over:**

There are only two reasons for game over:

1. Hit wall
2. Collision

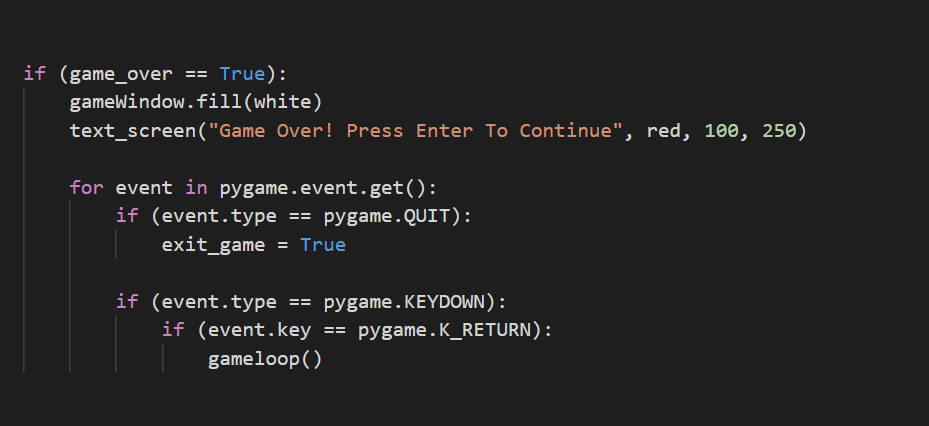
* **Hit Wall:**

There are 4 walls so 4 possibilities to hit the wall. So, we will make an **if** statement for this and the **if** statement will have 4 conditions (one for each wall).



Our game screen is between 0 to screen\_width and screen\_height so if snake is on any other co-ordinate (less than 0 or greater than screen\_width/screen\_height) then it’s out of screen. Hence, Game Over.

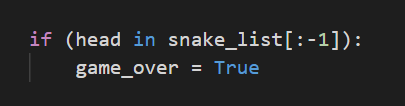
After the game is over, we will ask the user if he wants to quit the game or play again.



Here, **gameloop()** is a function for the main loop of the game.

* **Collision:**

Here, collision means head of snake will hit any part of snake.



So, we are simply seeing if snake’s head and snake’s any other body part is on the same co-ordinate except **head.**