Project Proposal: Snake Game in Python

Submitted to:
Lecturer Faria Sajjad

Group Members	Nutech ID
Syed Hamid Haider	F20605034
Usman Malik	F20605053
Muhammad Irbaz Anwar	F20605037
Ahmed Ali Haider	F20605013

Start here

Table of Content

Chap 1: Overview

1.1 Introduction

In this project we aim to create a snake game in python. This specific project was chosen to be done in this programming language due to its versatility and it is within the best interest of our future. We aim to own our skills so that it may help us to make a successful career out of this experience.

This project allows us to have an insight on Indie game development. As in indie game development the task of creating a fully-fledged game taken up by only a few handful of people. Game development is a field that requires a person to have mastery of a programming language to constantly be innovative with ideas so that they can compete with other game studios.

1.2 Technology Required

The project requires the following tech.

1.2.1 Hardware

For the making of this project the basic requirement include an average personal computer. Minimum requirements are as under

Processor: Intel core 2 Duo

RAM: 1G DDR1

Storage: 10GB HDD of free space

OS: Windows Xp

Drivers: Direct X11

1.2.2 Software

The software requirements include an IDE (Integrated Development Environment) that allows us to program in the required language and specific modules and drivers that allows us to generate the executable file for the game.

1.3 Reasons for Choosing this Project

The simplicity of this project makes it a challenge itself as the game to be made user friendly but at the same time challenging and fun so that it can draw attention. On the coding end it has to be made with simplistic, so that in the case if modification are required they can be done even by a third party without the presence of the original source code producer. The benefits include

- Introduction to OOP based languages
- Understanding sprite formation
- Understanding the IDE.
- Learning the limitations of the programming language.
- Community research
- UI/UX interface introduction

Chap 2: Making the Game

2.1 General Idea For The Game

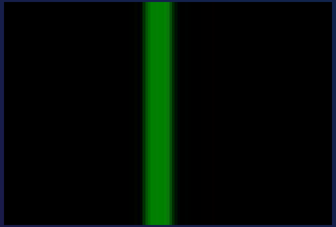
The goal was to keep the game as simple as possible. This meant the game was based on the idea to be as user friendly as possible. The snake, fruit and the core game mechanism.

The project was to be entirely software based. This meant all the project were coding intensive and perfectly suitable for our field. The most noticeable differences included sprite formation, rendering the movement of the main sprite, mathematical calculations, platform formation, UI/UX interface and sprite manipulation.

2.2 Structure of The Game

❖ Game Border

The game border is the designated area within which the game operates. It has only two functions to act as a barrier for the game and to be a kill zone upon interaction. This means that it would be an immediate game over if the snake touches the border.



❖ Fruit

The fruit is the main objective for the player. It is counted as the score. The more the player collects, the higher the score he can achieve. The colour was picked to be red so that it was easy for the human eye to perceive.

❖Snake Head

The snake head is the part where most of the interaction takes place. It must interact with the fruit in order for the game to progress. It is also the area where movement is coding.

❖ Snake Body

After eating the fruit it is important for the game to progress and the best way to show progress is to increase the size of the snake itself and to make the game harder.

Chap 3: Setup and Game Mechanics

3.1 Requirements

The modules required are as under

Random: Module will be used to generate random numbers

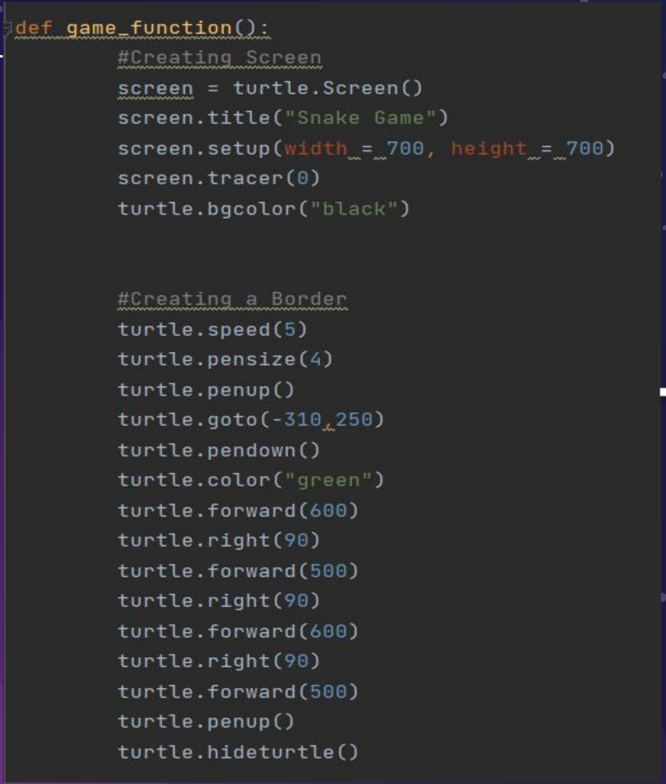
Time: Module is an inbuilt module in python. It provides the functionality of time.

Turtle: Module gives us a feature to draw on a drawing board.

```
from turtle import *
color('red', 'yellow')
begin_fill()
    while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill() done()
```

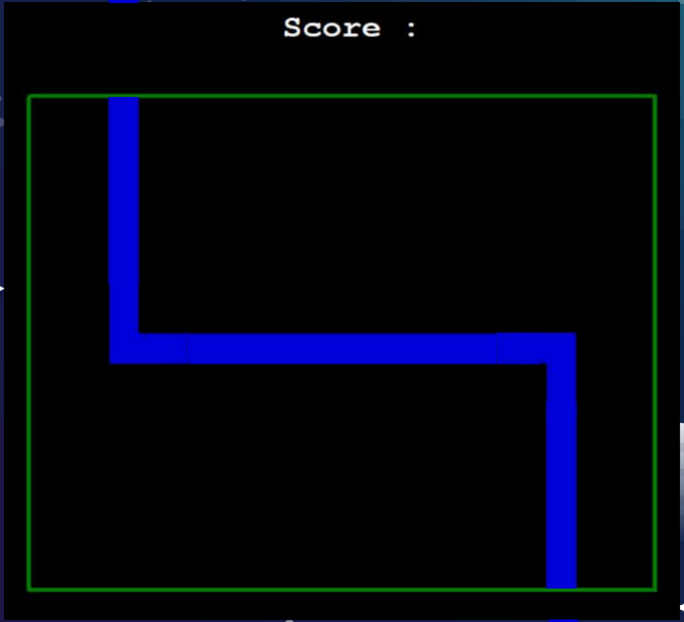
3.2 Creation

❖Game Area

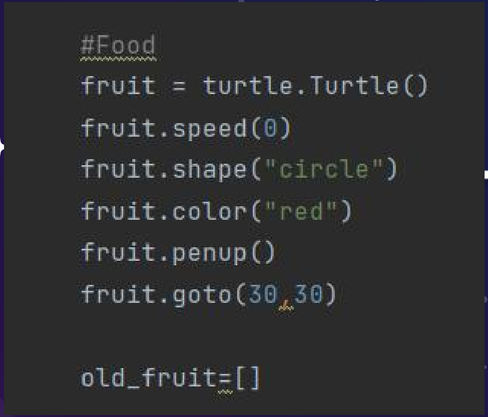


```
def game_function():
        #Creating Screen
        screen = turtle.Screen()
        screen.title("Snake Game")
        screen.setup(width = 700, height = 700)
        screen.tracer(0)
        turtle.bgcolor("black")


        #Creating a Border
        turtle.speed(5)
        turtle.pensize(4)
        turtle.penup()
        turtle.goto(-310,250)
        turtle.pendown()
        turtle.color("green")
        turtle.forward(600)
        turtle.right(90)
        turtle.forward(500)
        turtle.right(90)
        turtle.forward(600)
        turtle.right(90)
        turtle.forward(500)
        turtle.penup()
        turtle.hideturtle()
```

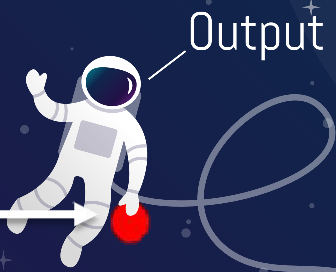
Output



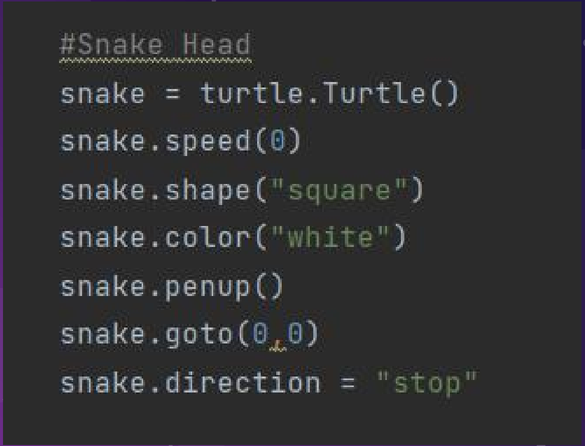
❖ Fruit



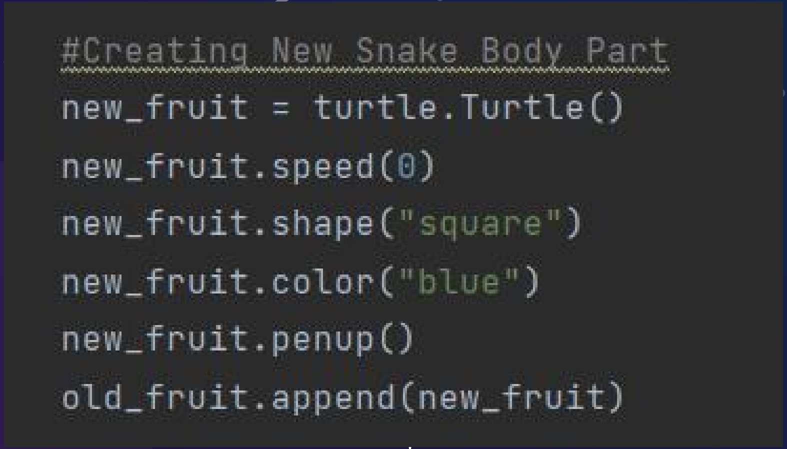
```
#Food
fruit = turtle.Turtle()
fruit.speed(0)
fruit.shape("circle")
fruit.color("red")
fruit.penup()
fruit.goto(30,30)

old_fruit=[]
```



❖Snake Head and Body



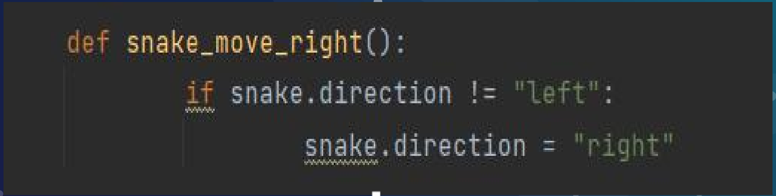
```
#Snake Head
snake = turtle.Turtle()
snake.speed(0)
snake.shape("square")
snake.color("white")
snake.penup()
snake.goto(0,0)
snake.direction = "stop"
```



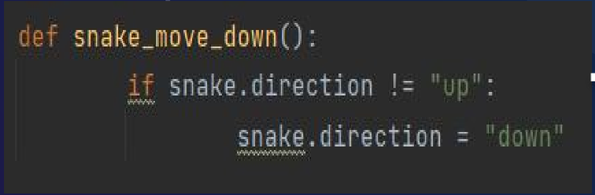
```
#Creating New Snake Body Part
new_fruit = turtle.Turtle()
new_fruit.speed(0)
new_fruit.shape("square")
new_fruit.color("blue")
new_fruit.penup()
old_fruit.append(new_fruit)
```

3.2 Snake and Environment

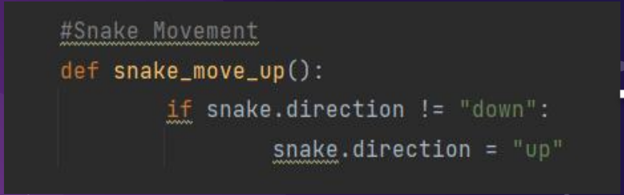
❖ Key Configuration



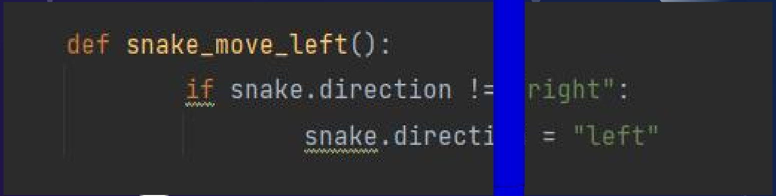
```
def snake_move_right():
        if snake.direction != "left":
                snake.direction = "right"
```



```
def snake_move_down():
        if snake.direction != "up":
                snake.direction = "down"
```



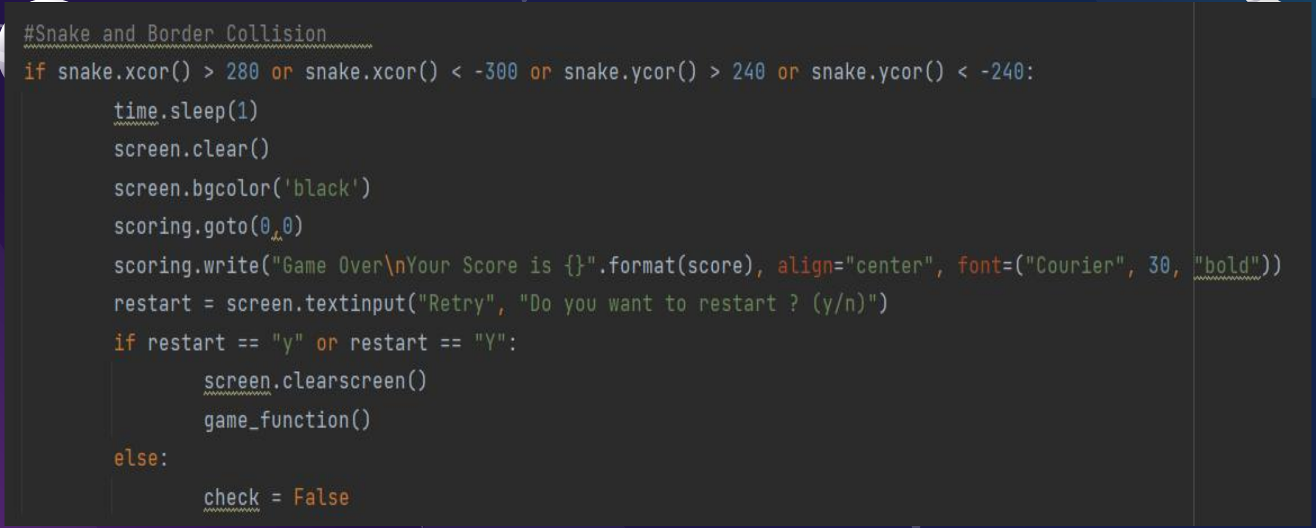
```
#Snake Movement
def snake_move_up():
        if snake.direction != "down":
                snake.direction = "up"
```



```
def snake_move_left():
        if snake.direction != right":
                snake.directi  = "left"
```

❖ Snake and Border Collision

One of the most important mechanics of the game as it determine if the player has lost the game of not. It is important to set boundaries to the game area in the case that the player does not go beyond the area that is being displayed

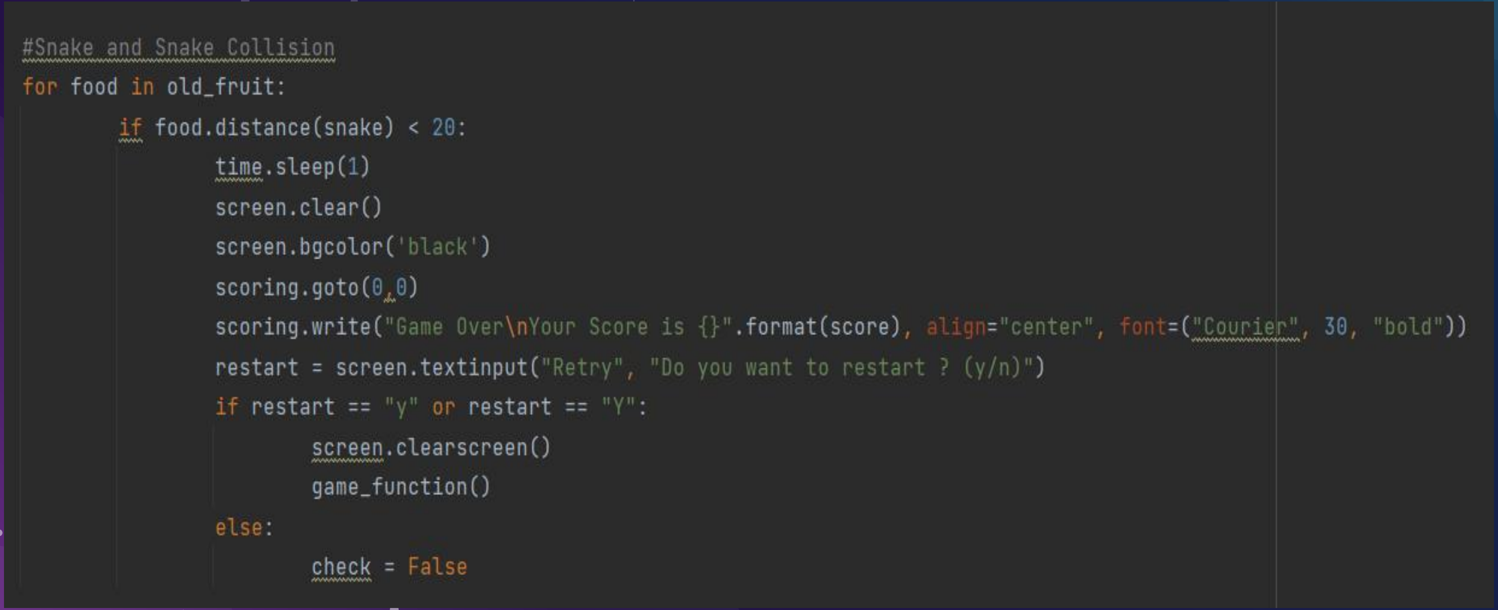


```
#Snake and Border Collision
if snake.xcor() > 280 or snake.xcor() < -300 or snake.ycor() > 240 or snake.ycor() < -240:
        time.sleep(1)
        screen.clear()
        screen.bgcolor('black')
        scoring.goto(0,0)
        scoring.write("Game Over\nYour Score is {}".format(score), align="center", font=("Courier", 30, "bold"))
        restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
        if restart == "y" or restart == "Y":
                screen.clearscreen()
                game_function()
        else:
                check = False
```

It is important to have this mechanism in order to avoid making an endless experience.

❖ Snake and Snake Collision

Also one of the most important mechanics of the game as it determine if the player has lost the game of not. It is important to make the body of the snake and object as in if the head touches the body it result in a game over.

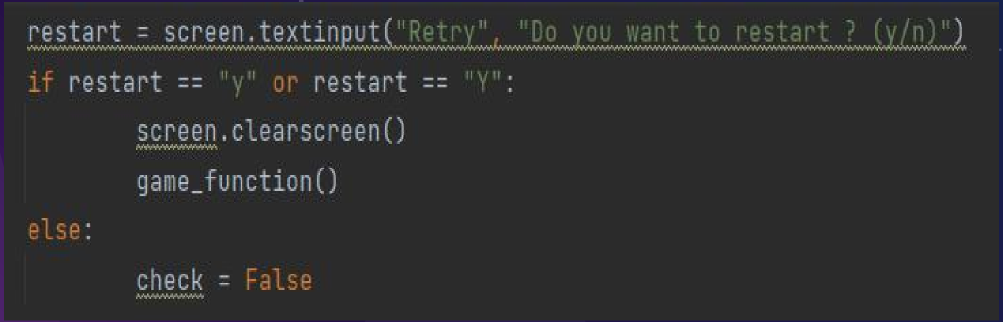


```
#Snake and Snake Collision
for food in old_fruit:
        if food.distance(snake) < 20:
                time.sleep(1)
                screen.clear()
                screen.bgcolor('black')
                scoring.goto(0,0)
                scoring.write("Game Over\nYour Score is {}".format(score), align="center", font=("Courier", 30, "bold"))
                restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
                if restart == "y" or restart == "Y":
                        screen.clearscreen()
                        game_function()
                else:
                        check = False
```

❖ Game over

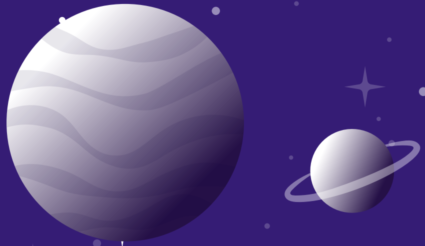
The game over screen the prompted either when the player has touched the border or the snake head collides with the snake body. It is important for this to exist in order to indicate that the user has lost and they cannot continue any further.

The restart option of given to the player in the case if he want to try again in order to obtain a better score or would like to experience the game again.



```
restart = screen.textinput("Retry", "Do you want to restart ? (y/n)")
if restart == "y" or restart == "Y":
        screen.clearscreen()
        game_function()
else:
        check = False
```

This code allows us to show a prompt asking the player if he would like to try again. If the user enter the letter “*Y*“(capitalized or not) the score of the game is reset and begins anew.



Thanks You

Any Questions?