

|  |  |
| --- | --- |
| Name | Adarsh Khatri |
| Registration No | 12020971 |
| Section | K20BG |
| Roll No | RK20BGA24 |
| Subject | Snake Game |

Text

Description automatically generated

*Submitted to:*

**Dr. Sagar Pande**

INT213: Python Programming

Snake Game

Submitted By:

Adarsh Khatri

Registration Number:

12020971

This was done as the work of mine during the third semester known as Continuous Assessment (CA1).

Contents:

* Getting Started
* Snake Head
* Snake Food
* Snake Body
* Border Collisions
* Body Collisions
* Scoring

**1.) Getting Started**

**Turtle Module:** The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways.

**Turtle Screen:** The Turtle Screen class defines graphics windows as a playground for the drawing turtles.

**Turtle** -Turtle is a python feature like a drawing board, which lets you command a turtle to draw all over it.

Some of the turtle functions used in the project are:

a. Turtle Methods

1)**Turtle Motion** (Move and Draw):

->goto() #for setting position

->setx() and sety() #for setting the x and y coordinates

->speed() #for speeding of animation

2)**Turtle’s State**:

->xcor() and ycor() #tells the x and y coordinates

->distance() #tells us the distance between two turtles

3)**Drawing State:**

->penup() #there is no drawing when we move

4)**Color Control:**

->color() #turtle’s object color

5)**Appearance:**

->shape() #turtle’s object shape

6)**Background Image:**

**->bgpic(“pic.gif”)**#for adding background image cause turtle doesn’t supports normal picture format.

b. Methods of Turtle Screen

1)**Window control**

->bgcolor() # for setting the background color

2)**Animation Control**

->tracer() #for turning off the animation on screen and

Turning off the screen object

->update() #for updating the screen every time

3)**Using screen events:**

->listen() #for collecting the key events

->onkeypress() #for assigning the keys to a function

->mainloop() #main loop function

4)**Special Methods:**

->turtles() #for returning the list of turtles on the screen.

5)**Methods specific to Screen:**

->setup() #for setting the dimensions of the screen

->title()#for assigning the title for the screen

C. **Time Module:**

->Generally, Python has a module named time to handle time-related tasks.

Some of the time module functions used in the project are:

->**Time Functions**

1)time.sleep() #suspends the current running thread for the given number of seconds

d.**Random Module**

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

Some of the random functions used in the project are:

->randint() #Returns a random integer between the specified integers.

**Snake Head**

The snake head function does the following:

#head=turtle.Turtle #creating a turtle object for the snake head

head.speed(0) #this controls the speed of the animation

head.shape(“square”) #gives the shape

head.color(“yellow”) #for choosing the color

head.goto(0,0) #this function will put the head at the middle

head.direction = “stop”

->**move function**

We also have a move function which will move the snake head.

->if the head.direction is up then we change the y coordinate, we do that with all four directions up, down, right, and left.

->**key bindings**

We also have binded the keys on the keyboard for the respective directions by using the onkeypress().

For moving the snake I have done the normal up,down,left and right keys.

Text, application, chat or text message

Description automatically generated

**Snake Food**

The snake food function does the following:

food=turtle.Turtle #creating another turtle object for the food

The code is similar to that of the snake head.

We change the shape and color and also assign its position at (0,0).

To make the snake eat the food we use an if statement:

->if the distance between two turtles (i.e., the food and head) is less than 20 (which the length and breadth of each pixel in turtle

we put the food at random spots by using the randint() imported by the random module.

**Snake body**

Here we are adding segments whenever the snake eats the food. i.e., adding multiple objects that will help snake body to grow.

1. At the starting of the game snake consist of just head and no segments will be added unless snake gets food.

Therefore, we create an empty list which consist of segments.

i.e., segments= [].

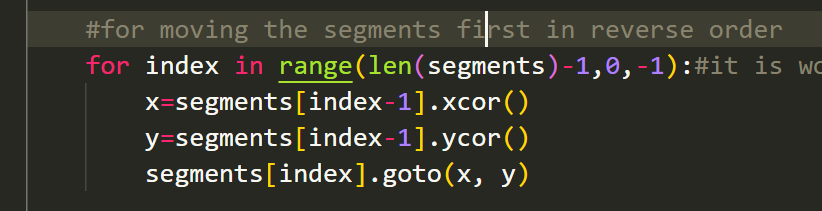
1. In the main game loop:

#addition of the segment i.e. the growing body of the snake below head.

Text

Description automatically generated

1. Move end segments in reverse order: Following the above statements won’t help to add segments hence, in the main loop:



1. Now we have to add these created segments at the back of the snake head

# Move segment 0 to where the head is:

Text, chat or text message

Description automatically generated

**Border collisions**

1. In this part we have to check for a border collision i.e., if the snake hits the border of the window (i.e., screen) the game should reset, and the segments added at the back of the head gets deleted.

Graphical user interface, text, application

Description automatically generated

1. Now we have to hide the segments after the snake collides with the borders of the screen.

A screenshot of a computer

Description automatically generated with low confidence

#segments behind the head

goes off the screen as python turtle module does not supports deleting the segment from the screen.

1. Clearing the segments list in order to get the head back at the initial position.

segments.clear()

**Body collisions**

1. In this part we have to check if the snake collides with its own body then the game should reset i.e. the head goes to its initial co-ordinates and the segments behind the head should disappear.

# Check for head collision with the body segments

for segment in segments:

if segment.distance(head) < 20: #collision with the segment.

time.sleep(1) #time delay(pause)

head.goto(0,0) #head to the initial co-ordinates.

head.direction = "stop" #no movement of the head.

# Hide the segments

for segment in segments:

segment.goto(1000, 1000) #segments go off the screen

# Clear the segments list

segments.clear() #clearing the segments list from the code for head appearance.

**Scoring**

In this we will add the scoreboard in the main screen of the game. Whenever snake eats the food, the segments goes on increasing and simultaneously the score increases accordingly. After the snake collides with the border or with its own body the score should reset to the initial value and high score by the previous game played.

1. Defining the main scoreboard on the screen:

score=0

high\_score=0

pen = turtle.Turtle()

pen.speed(0) #animation speed.

pen.shape("square") #shape of the scoreboard.

pen.color("white") #colour of the scoreboard

pen.penup()

pen.hideturtle()

pen.goto(0, 260) #position of the scoreboard.

pen.write("Score: 0 High Score: 0", align="center", font=("Courier", 24, "normal")) #winitializing the values.

1. In the main loop.

# Increase the score

score += 10 #here we increase the score

if score > high\_score:

high\_score = score # he we save high score of the previous game.

pen.clear()

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal")) #formatting

1. When body hits the border:

# Reset the score

score = 0

# Reset the delay

delay = 0.1 #resetting the speed of the snake

pen.clear() #clears the main initializing of the default score

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

1. When the snake hits the segments (i.e., its own body): these statements are to be written at the loops of the border collisions and body collisions with proper indentation

# Reset the score

score = 0

# Reset the delay

delay = 0.1 #resetting the speed of the snake.

# Update the score display

pen.clear() #clears the main initializing of the default score

pen.write("Score: {} High Score: {}".format(score, high\_score), align="center", font=("Courier", 24, "normal"))

A picture containing text, grass

Description automatically generated

So, in this way I created a snake game.

**References:**

Stack Overflow

Reddit