This first solution will be based on refracting the Snake Game to improve its efficiency using Python. The high-level pseudocode for this algorithm is as follows:

1. Game Initialization:

- Set up the game environment, including the score, frame rate, and segment size.

- Create the Snake and Food entities.

2. Main Game Loop:

while the Snake is alive:

a. User Input Handling:

- Respond to arrow key input to change the snake's direction.

- Check for quit events.

b. Update Game State:

- Move the snake, handle collisions, and update its body segments.

- Check for collisions with food, increase the score, and spawn new food.

c. Draw Game:

- Clear the screen.

- Render the snake, its body, and the food on the screen.

- Update the display.

d. Adjust Frame Rate:

- Increase the frame rate periodically for faster gameplay.

3. Game Over:

Display the final score and clean up resources.

I have provided screenshots of pseudocode for critical functions in the Snake Game:

Snake Movement: A black background with white text

Description automatically generated

Collision Check:

A screenshot of a computer

Description automatically generated

Since I uploaded the base code from Rosetta Code, the first thing I noticed was the problem with the initial positioning of the snake and food as sometimes neither the snake nor food would appear within the visible area, e.g. the photo below shows the food but no snake.

A screen shot of a computer

Description automatically generated

I corrected this by modifying the initial random placement of the “SnakeFood”.

and the screen size in the “Game” class

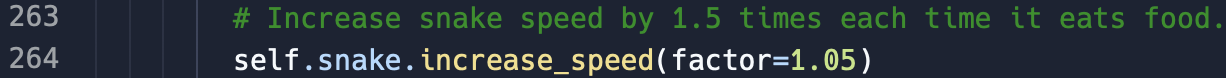
A black background with white text

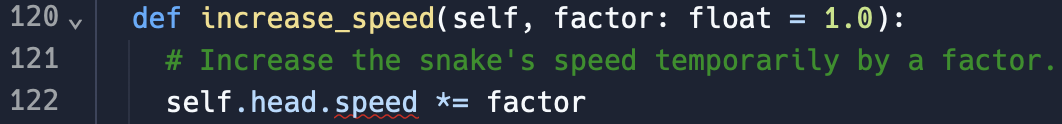
Description automatically generated

A black background with text and numbers

Description automatically generated with medium confidence

The first step I took to modify this Snake Game is making it gradually harder by increasing the speed by x1.05 each time the snake eats the food. I had to add this method to the main(self) method of my ‘Game’ class and modify the ‘increase\_speed’ method under Snake(RenderUpdates) in my ‘Snake’ class.





There is also a High Score feature that has been implemented and displayed at the end of each game. A file was created with 3 scores initialized to zero and the ‘Game’ class was modified to introduce methods to save, load, update and display the high scores. The Current score is also printed at the end of the game.

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated



Finally, to make this game more unique, a Student ID needs to be inputted before the game starts, with some validation in place to make sure that it is 8 digits only.

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer code

Description automatically generated

**TESTING**

Here are the tests I ran once the system was working efficiently:

1. Basic Gameplay

A screenshot of a computer with Marfa lights in the background

Description automatically generated

A screenshot of a game

Description automatically generated

1. Speed Increase

A screen shot of a computer

Description automatically generated

Snake segments start separating slightly as speed gets faster.

1. High Score

A screenshot of a computer

Description automatically generated

1. Input Validation of Student ID

A screenshot of a computer

Description automatically generated

1. Screen Boundary Test

A screen shot of a computer

Description automatically generated