******

**School of Mechanical & Manufacturing Engineering (SMME),**

**National University of Science and Technology (NUST),**

**Sector H-12, Islamabad**

|  |
| --- |
| **Fundamentals of Programming II**  Program: BE-Aerospace Section: AE-01  Session: Fall 2024 Semester: 3rd  Name: Affan Suhail CMS: 477560  Subject Code: CS-223 |

**Introduction**

The interactive **space shooter** is a command-line game written in Python, designed to offer a fun and educational experience for players and students while teaching foundational concepts of programming such as loops, conditional statements, and keyboard input handling. This program simulates a simple space battle scenario where the player controls a spaceship (X) to avoid and destroy falling asteroids (**O** for big and **o** for small) using bullets (**|**).

This project is ideal for beginners and intermediate programmers who want to practice Python by building an interactive game with real-time updates.

**About the Project**

**The program is a command-line game built using:**

* Python's standard library for randomness, timing, and screen management.
* The keyboard library to capture real-time keyboard inputs.

**The game allows the player to:**

* Move the spaceship left or right.
* Shoot bullets to destroy asteroids.
* Earn points for hitting asteroids.
* Continue playing until an asteroid collides with the spaceship.

Asteroids fall from the top of the screen, and the player must react quickly to avoid them and score points.

**Theory**

The project utilizes several programming concepts and tools, which are detailed below:

**Game Mechanics**

* The spaceship (X) moves horizontally along the bottom row of the screen.
* Bullets (I) shoot upward from the spaceship's position.
* Asteroids (O and o) fall from the top of the screen at random positions.

**Key Concepts**

**Global Variables:** Used for tracking game state, such as the position of the spaceship, bullets, and asteroids.

**Keyboard Input:** Captured using the keyboard library to move the spaceship and shoot bullets.

**Collision Detection:** Implemented to check if bullets hit asteroids or if asteroids collide with the spaceship.

**Dynamic Updates:** The game field is updated in real-time to reflect changes in positions of objects.

**Scoring System:** Players earn points for destroying asteroids, with larger asteroids offering more points.

**Key Python Features Used**

**Random Module:** Used to randomly generate asteroids at different positions and sizes.

**Time Module:** Used to control the game loop’s speed and pacing.

**Loops and Conditionals:** Form the core logic for game updates and interactions.

**Lists:** Used to store and manage bullets and asteroids.

**Application**

This project demonstrates several applications of programming concepts:

**Interactive Command-Line Games:** Showcases how to build engaging games without graphical interfaces.

**Real-Time Input Handling:** Demonstrates capturing and reacting to user input in real-time.

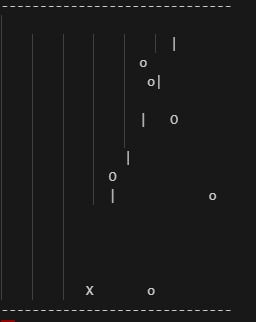
**Game Design Basics:** Introduces core mechanics such as collision detection, scoring, and game-over conditions.

**Practical Learning:** Helps learners understand concepts such as state management and dynamic updates in a fun and interactive way.

**Output**

Expected Output During Gameplay

The game displays:



The above picture shows a dynamic game field with:

The spaceship (X) at the bottom.

Bullets (**|**) moving upward.

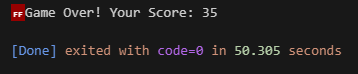
Asteroids (O and o) falling from the top.

A real-time score displayed at the top of the screen.

Boundaries (-) to denote the game area.

Game Over Output

When the spaceship collides with an asteroid, the game ends, and the final score is displayed:



**User-Defined Functions**

The program uses several user-defined functions to structure the logic:

**clear\_screen()**

* Clears the terminal screen for real-time updates.

**draw\_game()**

* Creates and displays the game field.
* Places the spaceship, bullets, and asteroids on the screen based on their positions.

**move\_spaceship()**

* Handles keyboard input (left and right arrow keys) to move the spaceship horizontally.

**shoot\_bullet()**

* Allows the player to shoot bullets upward by pressing the space key.
* Adds a bullet to the bullets list.

**update\_bullets()**

* Moves bullets upward each frame. Frames are updated by the time function
* Checks for collisions with asteroids.
* Removes bullets and asteroids upon collision.

**spawn\_asteroids()**

* Randomly spawns new asteroids (O or o) at the top of the screen.

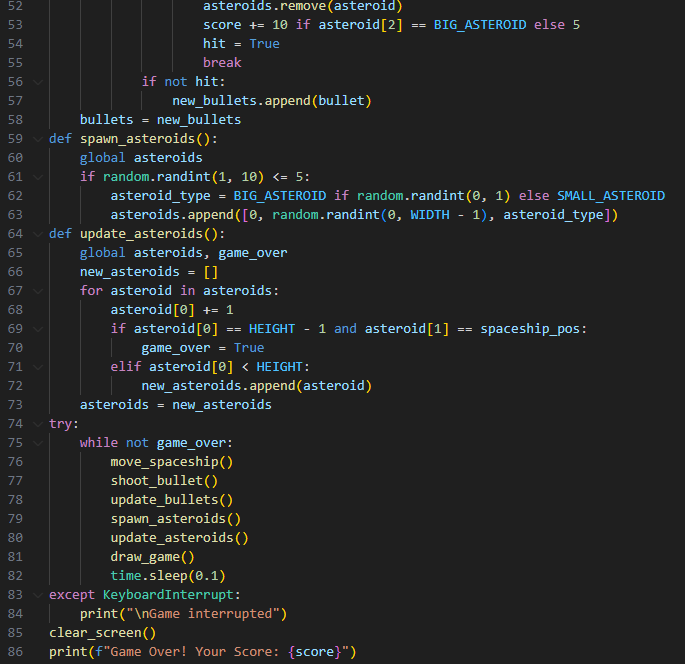
**update\_asteroids()**

* Moves asteroids downward each frame.
* Checks for collisions with the spaceship to determine if the game should end.

**Main Game Loop**

* Combines the above functions into a continuous loop to handle real-time gameplay.

**Code:**



**Conclusion**

The interactive space shooter program is an excellent way to learn programming by creating a simple yet dynamic game. It introduces key programming concepts like real-time updates, user input handling, and structured code design. The project is both educational and entertaining for a student who is a novice in programing.