Minesweeper Game Documentation

Overview

This project implements a console-based Minesweeper game in Python. The game allows users to select a difficulty level, customize the board size, and play the classic Minesweeper game. The program includes animated text and input for a more engaging user experience.

Features

Game Functions:

- Choose Difficulty: Users can select a difficulty level that determines the number of mines on the board.
- Custom Board Size: Users can specify the number of rows (5-99) and columns (5-9) for the board.
- Game Timer: Tracks the time taken to complete the game.
- Dynamic Board Updates: The board updates dynamically as users open cells.

Win/Loss Conditions:

- Win: All non-mine cells are opened.
- Loss: A mine is uncovered.

Replay Options: After a win or loss, users can choose to replay or return to the main menu.

Error Handling

- Input validation for rows, columns, and other user inputs.
- Graceful handling of invalid inputs with appropriate error messages.

Animated Text and Input

- Provides a visually appealing user interface with animated text prompts.

Library Versions

The following libraries are used in this project:

- Python: 3.13
- time: Built-in Python library for tracking game duration.

Minesweeper Game Documentation

Custom Modules:

- minesweeper.animated_input
- minesweeper.animated_text
- minesweeper.clear_console
- minesweeper.generate_minesweeper_board
- minesweeper.count_adjacent_mines
- minesweeper.open_empty_neighbors
- minesweeper.hide_mines
- minesweeper.no_closed_cells
- minesweeper.display_menu

Architecture Description

The project is organized into two main directories:

Source Code (source):

- Contains the main game logic and supporting modules.
- Key files:
  - main.py: Entry point of the application.
  - minesweeper/: Contains helper modules for game functionality (e.g., board generation, input handling).

Tests (tests):

- Contains unit tests for all modules in the minesweeper package.
- Key files:
  - test_animated_input.py
  - test_generate_minesweeper_board.py
  - test_count_adjacent_mines.py

User Interface

The game runs in the terminal and provides the following interface:

Minesweeper Game Documentation

Main Menu:

Options:

1: Start a new game.

2: Exit the game.

Game Prompts:

- Input for board size (rows and columns).

- Input for starting cell (row and column).

- Input for selecting cells during gameplay.

Game Board:

- Displays the Minesweeper board with hidden cells ([?]), opened cells, and mine counts.

Endgame Messages:

- Displays a win or loss message with the final board and elapsed time.

Program Flow

Main Menu:

- Displayed when the program starts.

- User selects an option to start the game or exit.

Game Initialization:

- User selects difficulty and board size.

- The board is generated with mines and counts.

Gameplay:

- User selects cells to open.

- The board updates dynamically.

- The game continues until the user wins or hits a mine.

Endgame:

- Displays the final board and game result.
- User can replay or return to the main menu.

Static and Dynamic Analysis Results

1. Unit Tests
- Framework: unittest
- Test Coverage: 95%
- Test Results: All tests passed successfully.
- Example:
  - test_generate_minesweeper_board: Validates board generation logic.
  - test_count_adjacent_mines: Ensures correct mine count calculation.

2. Code Coverage
- Tool: coverage.py
- Result: 95% line coverage.
- Key uncovered areas: Edge cases for invalid inputs.

3. Linting
- Tool: pylint
- Score: 9.5/10
- Minor warnings for docstring formatting.

4. Type Checking
- Tool: mypy
- Result: No type errors detected.

Conclusion

This Minesweeper game is a robust and interactive console application with a clean architecture and high code quality. The game is thoroughly tested and adheres to Python best practices.