Text Editor – Algorithm Explanation

Objective

The purpose of this program is to implement a simple text editor in Python that allows users to interact with and manipulate the contents of a text file. Users can read from a file, view statistics, modify the text, and save changes through a menu-driven interface.

General Algorithm Description

The program starts by reading a .txt file and storing its content in a global string variable named text. A loop then presents a menu of editing options to the user. Each option corresponds to a function that performs a specific text manipulation task. The loop continues until the user chooses to exit.

Function Overview

1. **AllWordCount**
   This function uses Python's Counter from the collections module to count the frequency of each word in the text. It then displays the top 5 most common words.

2. **SingleWordCount**
   The user is prompted to enter a word. The function counts how many times that word appears in the text, using a case-insensitive comparison.

3. **ReplaceWord**
   Prompts the user for a target word and a replacement word. It replaces all case-insensitive occurrences of the target with the replacement and reports how many replacements were made.

4. **AddText**
   Appends new user-provided text to the current text stored in memory.

5. **DeleteText**
   Removes the first exact occurrence of a user-specified substring from the text.

6. **HighLight**
   Asks the user for a word and prints the text with all exact matches of that word surrounded by **, for emphasis.

7. **saveTextFile**
   Saves the current state of the text variable back to the original file (text.txt), overwriting its content.

8. **readTextFile**
   Reloads the contents of the text file into memory, replacing any unsaved changes in the text variable.

9. **Main Menu Loop**
   A while True loop prints the menu, collects the user's input, and calls the appropriate function. The loop ends when the user selects the exit option.

Implementation Notes

- global is used inside functions that need to update the text variable.
- All word matching is performed in a case-insensitive manner for consistency.
- The menu design allows users to test each function interactively.
- Counter is used for efficient word counting and frequency tracking.
- The program structure makes it easy to expand with additional features if needed.

Summary

This project demonstrates fundamental programming concepts in Python, including file I/O, loops, conditionals, string manipulation, dictionaries, and modular function design. It simulates basic text editing functionality through a console interface and provides a foundation for more advanced file-based applications.