Title: Text Editor Algorithm

Author: Shaun Clarke

Goal: This program mimics some of the basic functionalities of a text editor.

Steps:

1. Import *string* module.
2. Import *re* module
3. Instantiate *file_path* global variable to hold .txt file location
4. Define a function close_file(file):
    a. This function uses file.close to close the text file after reading or writing.
    b. This function takes one parameter, *file*.
    c. This function returns nothing.
5. Define a function read_file(path, no_case=False):
    a. This function reads in the content of the .txt file and converts the content to lowercase.
        i. Opens the file
        ii. Reads the content of the file.
        iii. If case flag is false make text lowercase, else leave as is.
    b. This function takes two parameters:
        i. path, which is the location for the text file
        ii. no_case, a switch that is used to turn lowercase on or off
    c. This function can return a few different things
        i. If all goes well, it should return the contents of the text file.
        ii. It will also return false if the file was not found
6. Define a function write_to_file(path, mode, content):
    a. This function will be used throughout to write output to the text file and displays the updated output.
        i. If mode is a append content to file.
        ii. If mode is set to w, overwrite with new content.
    b. This function takes three parameters
        i. path, which is the location for the text file
        ii. mode, which is used to (a) append or (w) overwrite the file.
        iii. content, this is what we want to write to the file.
    c. This function returns two parameters.
        i. False, if file want to write to was not found.
        ii. The updated file content.
7. Define a function, remove_punctuations(file_content, split=True):

    a. This function is used to remove data from the text file when needed.
        i. Remove punctuations from file content.
        ii. If split flag is true, split file content string into list. If not leave as string.
    b. This function takes two parameters:
        i. file_content, which is the content of the text file.
        ii. split=False, this is a default parameter used as a switch. If set to True it will split into a list.
    c. This function returns the file content as a list or a string depending on the split flag.
8. Define a function all_word_count(path):
    a. This function counts the 5 most common words and returns them.
        i. Uses read_file to import the contents of the text file.
        ii. Uses remove_punctuation to strip punctuations from the file content.
    b. This function takes one parameter:
        i. path, which is the location for the text file.
    c. This function returns:
        i. Returns False if the file was not found.
        ii. Returns a list of tuples with the 5 most common words and the total times they appeared.
9. Define a function single_word_count(path, word):
    a. This function finds out how many times a word appeared in the text file.
        i. Uses read_file to import the contents of the text file.
        ii. Uses remove_punctuation to strip punctuations from the file content.
        iii. Uses .count() to count how many times the word occurred if the word exists.
    b. This function takes one parameter.
        i. word, this is input for the word the user wants to count.
        ii. path, the path to the text file.
    c. This function returns.
        i. If all goes well, this function returns and integer which is the count.
        ii. Returns 0 if the word submitted doesn't exist.
        iii. Returns file not found if the file doesn't exist.
10. Define a function replace_item(path, item_to_replace, new_item, skip=False):
    a. This function replaces text in the file, the replace feature is also used to delete items from the file
        i. Checks if the file exists, if not return 0.
        ii. Uses read_file to import the contents of the text file.
        iii. Uses remove_punctuation to strip punctuations from the file content.

iv. Use regex to the complete word and replace it.
v. Then write the updated content back to the text file with a “w” to overwrite.
vi. Check if the updated word exists.

b. This function takes four parameters:
i. path, the path to the text file.
ii. word, this is input for the word the user wants to replace.
iii. New_item, the new word that will be added to the text file.
iv. skip=False, default parameter that we set to True if using this function to delete instead of replace.

c. This function returns:
i. Returns 0 if the word is not in the file.
ii. Returns not updated if the file was not updated.
iii. Returns False if file was not found.

11. Define a function add_text(path, mode, content):
a. This function appends text to the text file:
i. Uses the write_file function to append text to the text file.

b. This function takes three parameters.
i. path, which is the location for the text file
ii. mode, which is used to (a) append or (w) overwrite the file.
iii. content, this is what we want to write to the file.

c. This will return:
i. False, if the file was not found.
ii. The updated file content.

12. Define a function delete_text(path, item_to_replace):
a. This function deletes text from the text file.
i. Uses the read_file function to read the text file
ii. Check If the word(item_to_replace) the user submitted exists in the file.
iii. Uses the replace_item function with the skip flag set to true to replace the word with empty quotes “”.
iv. Check if item was deleted.

b. This takes two parameters.
i. path, which is the location for the text file
ii. item_to_replace, the text to delete.

c. This function returns
i. The number of items that were deleted
ii. “not found” if the item to delete does not exist.

iii. False, if the file was not found.

13. Define a function highlight_text(path, item_to_highlight):
    a. This function highlights text entered by the user.
        i. Using the read_file to get the contents of the file.
        ii. Confirm if the word to highlight is in the file.
        iii. Update the file with the highlighted word.
        iv. Check if the file was updated.
    b. This function takes two parameters.
        i. path, which is the location for the text file
        ii. item_to_highlight, the text to highlight.
    c. This function returns.
        i. The number of occurrences of the word the was deleted.
        ii. Not found if the word doesn't exist
        iii. False if the file was not found
        iv. False if the file was not updated
        v. Exists if the word to highlight is already highlighted.
14. Define a function get_user_input(input_message, path, new=False):
    a. This function gets all user inputs except menu inputs.
        i. While loop to loop until user enters correct input.
            1. Ask user for input
            2. Read in the content from the file.
            3. If the input is not in the text file, prompt user to try again.
    b. This function takes three parameters
        i. Input message that tells the user what kind of input to enter.
        ii. Path to the text file
        iii. New, which is a flag set to true or false. This allows us to accept words that are not in the file.
    c. This function returns the user input.
15. Define a function user_menu_input():
    a. While loop that keeps looping until user enters correct input.
        i. Displays the user menu to accept input.
        ii. Check if the input is valid, if not prompt the user to try again.
    b. This function takes no parameters.
    c. This function returns the user input.
16. Define a function main(path):
    a. This function is were the program is defined
        i. Read in file content and store it in a separate variable so the original content can be restored.

ii. While loop to present user with the menu after a menu item was completed.
    1. Call the user menu function asking the user to choose an option.
        a. Return the user input
    2. Using match to compare the user menu output to the action to perform.
        a. If menu item 1 was entered:
            i. Call the all_word_count function and display top 5 words and their totals.
        b. If menu item 2 was entered:
            i. Call get_user_input
                1. Return the user input
            ii. Call remove_punctuations with the user input and split=False parameters.
                1. Return the string without punctuations.
            iii. Call single_word_count
                1. Return the single word count
            iv. Display the word count to the user.
        c. If menu item 3 was entered:
            i. Call get_user_input
                1. Return the user input
            ii. Call remove_punctuations
                1. Return a string without punctuations.
            iii. Call replace_item
                1. Return the count of how many items were replaced.
            iv. Display the output to the user
        d. If menu item 4 was entered:
            i. Call get_user_input
                1. Return the user input for text they want to add
            ii. Call add_text and passing in the user input
                1. Return output
            iii. Display updated text to the user.
        e. If menu item 5 was entered:
            i. Call get_user_input

1. Return the user input the text they want to delete

ii. Call remove_punctuations
1. Return a string without punctuations.

iii. Call delete_text with the remove punctuation output.
1. Return output of how many items were deleted.

iv. Display how many items were deleted to user.

f. If menu item 6 was entered:
i. Call get_user_input
1. Return the user input the text they want to highlight.

ii. Call remove_punctuations
1. Return a string without punctuations.

iii. Call highlight_text
1. Return the output

iv. If text is already highlighted, let user know
v. If not dislay updated content to the user.

g. If menu item 7 was entered:
i. Call write_to_file and pass in the original content that was stored before any changes.
ii. Display the updated file to the user.

h. If menu item 8 was entered:
i. Tell the user good bye.
ii. Exit the program.

17. Call the main function if the script is executed directly.