Anita Pal
Faculty of Arts and Humanities
Linnaeus University
ap224pu@student.lnu.se

Introduction

The following Python implementation, hosted on GitHub and Google Colab[1], contains functions for reading a text file and performing actions on its contents, such as extracting the longest sentence, retrieving the longest word, and calculating the top five frequencies. To carry out these actions, the implementation helper functions further process the text input.

Problem Definition

Similar to the first assignment, the implementation uses Python's standard methods, such as *max()*, *strip()*, and lambdas, to produce the desired outputs. For example, the Python *max* method takes a list as an argument, with the key parameter specifying how to compare values (e.g., in line 63[2], the strings in the sentences lists are compared by length) (GeeksforGeeks, 2025a).

Meanwhile, the *strip*() method removes unwanted spaces (e.g., line 64), tabs, and similar characters from a string, helping clean up text for further processing (Shah, 2025). Python's *lower*() method, which converts all uppercase characters in a string to lowercase, is also used as a means to clean the text for this assignment (Programiz).

As with the previous assignment, the implementation utilises Python's *split*() method to split a string using a separator (e.g., line 22) or whitespace (e.g., line 30), breaking the file's contents into list elements (Hunner, 2024); these methods are mainly used in helper functions to break up the text contents into separate sentences (e.g. lines 21 to 22) tor words (e.g. line 75).

One of the most prominent methods used throughout this assignment is Python's *with* keyword for opening files, which ensures that file resources are closed when they are no longer in use, even in the event of an error (Dataquest, 2025). It is used in helper functions, such as *create_stop_word_list_from_file* (i.e., lines 26 to 31), to read files smoothly. It is also used elsewhere, such as on lines 73-76. One of the biggest challenges for the assignment was with files that required a different encoding (i.e., line 27), as otherwise an error would be thrown (StackOverflow).

Methodology Approach

The *find_longest_sentence* function (i.e. lines 59-64) takes a file as input, proceeds to open the file and then reads its content. It then splits the file contents using the helper function *split_text_into_sentences* on lines 21-22. Taking the text content as input, this function uses the *split()* method to split the items into a list, using the full stop as the separator (cf. Hunner, 2024).

Once the output of this helper function has been retrieved (i.e. line 62), the function uses the *max()* method (cf.

Systech Group) to get the longest sentence from a list of sentences. Afterwards, it returns the longest sentence (i.e. line 64), removing any unwanted spaces using Python's *strip()* method (cf. Shah, 2025).

The *find_longest_word* function (i.e. lines 72-76) works similarly: it takes a file as input, reads it, and splits the text into a list (i.e. line 75), without requiring the full stop this time, as the emphasis is placed on individual words (cf. Hunner, 2024). In line 76, the *max()* method is used to retrieve the largest word, which is then further stripped of any undesirable elements, such as trailing spaces (cf. Hunner, 2024).

In lines 45 to 51, the *process_text_for_frequency* function takes in a text file as input before using the helper function *convert_words_to_lowercase* (lines 35 - 39) to convert its content to lowercase. In that very function, the text content is used as input, where, after being read, the Python methods lower() and split() are used to, respectively, convert its contents to lowercase and then split into separate items (i.e. line 38) to ensure that the function returns a list of lowercase strings as output (i.e. line 39).

Once the text has been converted to lowercase, the *process_text_for_frequency* function in lines 48 to 50 removes all stop words from the text content, using an empty list (declared in line 46) to append to it all words from the *lower_case_words_list* that do not contain a stopword. The actions described above use a *for* loop, and the keyword *not* to ensure that elements that do not match a specific pattern (cf. TutorialsPoint) are ignored (or, in this case, not appended to a list). The stop words are retrieved using the *create_stop_word_list_from_file* (i.e., lines 26 to 31) helper function that first reads a file (i.e. line 29) and then, after splitting the list of stop words into separate list items (i.e. line 30), returns a list of stop words (i.e. line 31).

In lines 85 to 93, the *get_top_five_frequencies* function first uses the helper function *get_all_frequencies* (i.e., lines 97 to 101) to retrieve all frequencies from a list and stores them in an empty dictionary (i.e., line 98) (GeeksforGeeks, 2025b). A dictionary stores items as key-value pairs while ensuring they are ordered and unique (W3Schools). In lines 99-100, a for loop iterates over all words, setting each word to 0 if it does not exist and incrementing the count if it occurs more than once (GeeksforGeeks, 2025b).

Line 101 returns all *word frequencies*, which, as a variable in line 91, are used by a lambda function to sort the list (LabEx). A lambda function is a function without a name (GeeksforGeeks, 2025c) that enables cleaner condition checking. More specifically, it serves as the key to sort the elements by using the second element in the tuple to count the frequencies each word has (GeeksforGeeks, d). Finally, line 93 returns the top five frequencies using array slicing, which allows one to retrieve the top five values with an end index (Megida, 2022).

Analysis of Results

According to the output produced by lines 110 to 117, the code produces the correct output, as the following things are printed on the screen:
- The longest sentence in the text.
- The longest word in the text.
- The top five frequencies.

For most outputs, the *get_pretty_output* helper (i.e., lines 16-17) is used to create pretty output with f-strings. F-strings make print statements easier, as variables can be embedded directly (IONOS Redaktion, 2024). They were not used for the top five frequencies because there was no real reason to use them there.

Conclusions and Reflections

In many ways, this assignment was fun and challenging, mainly because it taught me how to handle text files with different encodings (cf. StackOverflow).

(939 words, including in-text references but excluding titles).

AI Acknowledgements

For the sake of catching embarrassing typos and awkward phrasing (for the most part), the report used Grammarly. The code was written by me, with credit given where it is due.

References

Dataquest. (2025, April 7). *Tutorial: How to easily read files in Python (text, CSV, JSON)*. https://www.dataquest.io/blog/read-file-python/

Hunner, T. (2024, September 27). *The string split method in Python*. Python Morsels. https://www.pythonmorsels.com/string-split-method/

IONOS Redaktion. (2024, February 8). *Python f-Strings: Einfache String-Formatierung*. IONOS Digital Guide. https://www.ionos.at/digitalguide/websites/web-entwicklung/python-f-strings/

LabEx. (n.d.). *How to use set to count element frequencies in a Python list*. https://labex.io/tutorials/python-how-to-use-set-to-count-element-frequencies-in-a-python-list-398089

GeeksforGeeks. (2025a, July 15). *Python – max() function*. https://www.geeksforgeeks.org/python/python-max-function/

GeeksforGeeks. (2025b, October 25). Counting the frequencies in a list using dictionary in Python. https://www.geeksforgeeks.org/python/counting-the-frequencies-in-a-list-using-dictionary-in-python/

GeeksforGeeks. (2025c, November 11). *Python lambda functions*. https://www.geeksforgeeks.org/python/python-lambda-anonymous-functions-filter-map-reduce/

GeeksforGeeks. (d). *Python program to sort the list according to the column using lambda*. https://www.geeksforgeeks.org/python/python-program-to-sort-the-list-according-to-the-column-using-lambda/

Megida, D. (2022, December 8). *Python slicing – How to slice an array and what does [::-1] mean?* FreeCodeCamp. https://www.freecodecamp.org/news/python-slicing-how-to-slice-an-array/

Programiz. (n.d.). *Python string lower()*. https://www.programiz.com/python-programming/methods/string/lower

Shah, A. (2025, April 2). *How to strip characters from a Python string*. Real Python. https://realpython.com/python-strip/

StackOverflow. (n.d.). *Comment on "UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf in position 0"* [Online forum comment]. https://stackoverflow.com/questions/42339876/error-unicodedecodeerror-utf-8-codec-cant-decode-byte-0xff-in-position-0-in#comment79848426_42340744

Systech Group. (n.d.). *Python program to find the longest word in a string*. https://systechgroup.in/blog-python-program-to-find-the-longest-word/

TutorialsPoint. (n.d.). *Python not keyword*. https://www.tutorialspoint.com/python/python_not_keyword.htm

W3Schools. (n.d.). *Python dictionaries*. https://www.w3schools.com/python/python_dictionaries.asp