University of Dundee
Programming Languages for Data Engineering

Student: Dhanashree Nangre(2544403)

Github Link: https://github.com/dhanashree-nangre/Python_2544403

Python assignment(abb.py):

Objective:
For this assignment, the goal was to process a file containing phrases and generate three-letter abbreviations for each while adhering to specific rules. The implementation involves various functions, making use of pandas for streamlined code organization.

Imports and Setup

The necessary libraries (pandas, warnings, os) are imported.
Warnings are reset to their default behavior.

Functions

load_letter_values(): Loads letter values from a file.
weight_of_character(): Calculates the weight of a character based on its position and rarity value.
get_word_at_index(): Gets the word at a specific character index in a sentence.
calculate_score(): Calculates the score of an abbreviation.
get_possible_combinations_of_abbreviations(): Generates possible combinations of abbreviations for a name.

Code Flow:

1.The user is prompted to provide details of the input file via the command prompt.
2.A DataFrame is utilized to store information about abbreviations, including the name, abbreviation, and its corresponding score.
3.The code reads each line from the input file in a loop.
4.Within the loop, possible abbreviations for the phrase are collected, ensuring exclusion of non-alphanumeric characters.
5.The first letter of the abbreviation is set as the first letter of the phrase.
6.Checks are implemented to avoid repetition of the same abbreviation.
7.Scores for each abbreviation are calculated by considering factors such as the starting index of the word and the index of letters.
8.The scoring involves checking if the letter is at the beginning of the word (score: 0), at the end of the word (score: 5, or 20 if it's 'E'), or anywhere else (based on index and predefined values).
9.The details are stored in a DataFrame.

10. Duplicate abbreviations that occur in other words are eliminated.
11. Rows with the minimum score for abbreviations are selected and merged.
12. The results are saved in both CSV and text files.

Conclusion
The code successfully achieves its objective of generating three-letter abbreviations, calculating scores, and producing output files. It is well-organized and includes user-friendly prompts. The output files provide both detailed information (CSV) and a concise summary (text) of the results.

Sample input file:
Trees.py

```
Alder
Crab Apple
Common Ash
Silver Birch
Downy Birch
European Beech
Box
```

Sample Output file:
Nangre_trees_abbrevs.txt

```
Alder
Crab Apple CBP
Common Ash CAS
Silver Birch SVB
Downy Birch DYB
European Beech EBH
Box BOX
```

The code working as supposed to as for ‘Alder’ generated abbreviations was common with the abbreviations for ‘Alder Buckthorn ‘ so it is blank.