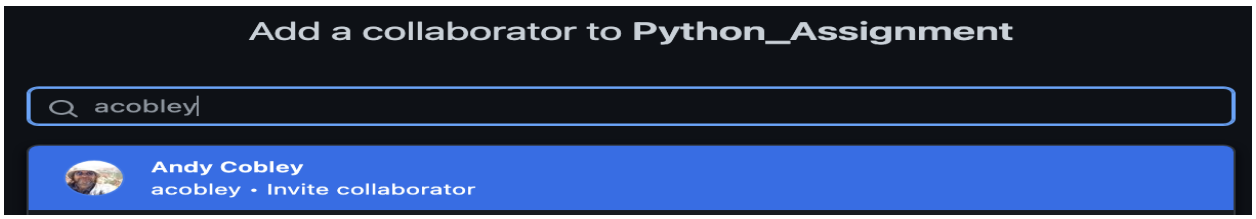
https://github.com/Ebuk-a/Python_Assignment
I have granted you(acobley) repo access, containing the input data, the code and the output.



Design

1. **Input/ Cleaning -> *accepts filepath from user***
    a. Clean each word: Accept input path from user, eparate compounded names (having special characters), remove special characters, some spaces, new lines, and capitalize words.
    b. Store the result as a list, cleaned_names.

2. **nameAbbreviator():** accepts a **name** and returns the **abbreviation** and **score**. It runs through all words in a name, checks letters with least score and returns the least-score abbreviation in order.
    It depends on two functions:
    a. **word_least_letter checker()**: accepts a word, and returns the **letter with least score** and the **score** of the letter

    b. **least_score_checker_updated()**: It depends on word_least_letter checker().
    It accepts a name and returns **two dictionaries**
        i. least_letter_tracker: Containing the each word and least letter (ignoring first letter) e.g for WONDER MAN {'WONDER': 'R', 'MAN': 'N'}

        ii. least_score_tracker: Containing the each word and least letter score (ignoring first letter) e.g for WONDER MAN {'WONDER': 5, 'MAN': 5}

3. **Bringing it all Together:**
    a. Call nameAbbreviator() function on each name in the cleaned_name list. E.g [Alder , Crab Apple, Common Ash, Silver Birch]

    b. nameAbbreviator() runs through combinations and calls the least_score_checker_updated() and returns the abbreviation and score for each.

    c. Store the abbreviations in a list. E.g [ADR, CBA, CNA, SRB]

    d. Zip the abbreviation list to the original(uncleaned) names and create a dictionary
    ```
    name_and_abb_dic = dict(zip(names, abbreviatons_only))
    ```
    {‘Alder’ : ‘ADR’ , ‘Crab Apple’: ‘CBA’, ‘Common Ash’: ‘CNA’, ‘Silver Birch’: ‘SRB’}

    e. For easy writing to a .txt file as new line items, iterate through the key & value and store in a list
    [‘Alder’, ‘ADR’ , ‘Crab Apple, ‘CBA’, ‘Common Ash’, ‘CNA’, ‘Silver Birch’, ‘SRB’]
4. **Output -> *output/akwiwu-uzoma_trees_abbrevs.txt***
    a. Create the output filename as surname +'_'+ input_filename + '_abbrevs.txt'
    b. Write each item of the list as a newline into a .txt file and store in the **output folder**

Evidence of Testing

Provided values.txt= {'Q': 1, 'Z': 1, 'J': 3, 'X': 3, 'K': 6, 'F': 7, 'H': 7, 'V': 7, 'W': 7, 'Y': 7, 'B': 8, 'C': 8, 'M': 8, 'P': 8, 'D': 9, 'G': 9, 'L': 15, 'N': 15, 'R': 15, 'S': 15, 'T': 15, 'O': 20, 'U': 20, 'A': 25, 'I': 25, 'E': 35}

Input File: trees.txt file (with some additional complex words to test performance)
https://github.com/Ebuk-a/Python_Assignment/blob/main/resources/trees.txt

Main Program file:
https://github.com/Ebuk-a/Python_Assignment/blob/main/akwiwu-uzoma_word_abbreviator.py

Output File: file can be found on github here:
https://github.com/Ebuk-a/Python_Assignment/blob/main/akwiwu-uzoma_trees_abbrevs.txt

Testing Calculations and Results (on spippet of data)

Word	Expected Abbreviation	Resulting Abbrevation	Individual Element & Scores	Total Score (Least Score)
Alder	ADR	ADR	A:first_letter(0) D: value(9) + index(2) R: last_letter(5)	16
Crab Apple	CBA	CBA	C:first_letter(0) B: last(5) A: first_letter(0)	5
Common Ash	CNA	CNA	C: first_letter(0) N: last_letter(5) A: first_letter(0)	5
Smooth-leaved Elm	SLE	SLE	S: first_letter(0) L: first_letter(0) E: first_letter(0)	0
He	‘ ’	‘ ‘	Not Applicable	Not Applicable

Snippets: