AC50002 Programming Languages for Data Engineering
Python Assignment
Name: Salar Junaid Qureshi
Student ID: 2432682

GitHub Link: https://github.com/salarjunaid/Python_Assignment_2432682

Code Explanation:

Three functions were defined:

1: load_data():

This function reads the file qureshi_values.txt file and splits each line through split() method on every whitespace character (default). It is stored in a list called temp which is then stored in a dictionary cost_map_new where the first value of the list is the key and the second value of the list is the value for the corresponding key. Qureshi_values.txt file consists of letters with their common/uncommon score.

```
# Function defined to load the common/uncommon score for each letter
def load_data():
    file_new = open("qureshi_values.txt", "r")
    while True:
        line = file_new.readline()
        if len(line) == 0:
            break
        else:
            temp = list(map(str, line.split()))
            cost_map_new[temp[0]] = int(temp[1])
    file_new.close()
```

2: calculate_score():

This function calculates score for each abbreviation generated for each name in the input file i.e. qureshi_trees.txt.

Three variables were initialized: first, last and score. These variables keeps a track on the first letter of each word, last letter of each word and the total score respectively.

All last letters are appended to last variable and all first letters were appended to first variable.

```
    first, last, score = "", "", 0
    for i in name_list:
        if len(i) > 1:
            last += i[-1]
        first += i[0]
```

Through a for loop scores are calculated for each abbreviation.

If a character is in the first variable, the score is increased by 0.

If the character E is in the last variable, the score is increased by 20. If there is any other character in last variable, the score is increased by 5.

If a letter is not in both first and last variable score is increased by the sum of positional score of a letter and the common/uncommon score of the letter.

The total score is returned by the function.

```
    for j in abb[1:]:
        if j in first:
            score += 0
            first = first[first.index(j):]

        elif j in last:
            if j == "E":
                score += 20
            else:
                score += 5
            last = last[last.index(j):]
        else:
            l = []
            for i in name_list:
                if j in i:
                    l.append(min(i.index(j), 3))
            score += min(l) + cost_map_new[j]
    return score
```

Whole function:

```
# Function defined to calculate scores for each abbreviation
def calculate_score(abb, name_list):
    first, last, score = "", "", 0
    for i in name_list:
        if len(i) > 1:
            last += i[-1]
        first += i[0]

    for j in abb[1:]:
        if j in first:
            score += 0
            first = first[first.index(j):]

        elif j in last:
            if j == "E":
                score += 20
            else:
                score += 5
            last = last[last.index(j):]
        else:
            l = []
            for i in name_list:
                if j in i:
                    l.append(min(i.index(j), 3))
            score += min(l) + cost_map_new[j]
    return score
```

3: generate_abbreviations():

This function generates abbreviations of length 3 for each name in the input file i.e. qureshi_trees.txt.

Common abbreviations among names are ignored.

calculate_score() function is called to calculate scores of each abbreviation.

```
# Function defined to generate abbreveations of length 3 for each name
def generate_abbreviations(name_list):
    name = "".join(name_list)
    n = len(name)
    abb = dict()
    for i in range(1, n):
        for j in range(i + 1, n):
            temp = name[0] + name[i] + name[j]
            if temp in common_new:
                continue
            else:
                abb[temp.upper()] = calculate_score(temp, name_list)
    return abb
```

Main Function:

In the main function input file i.e., qureshi_trees.txt is read and defined functions were called to get the desired output.

All of the common abbreviations among all of the names in the input file are stored in a list called common_new.

```
# common abbreviations are stored in common_new list
for i in range(len(mylist_new)):
    for j in range(i + 1, len(mylist_new)):
        temp = mylist_new[i].intersection(mylist_new[j])
        if len(temp) > 0:
            common_new += temp
```

The abbreviations with the smallest score are written in an output file i.e., qureshi_trees_abbrevs.txt.

```
# The abbreviations with the smallest score are written in an output file i.e., qureshi_trees_abbrevs.txt.
out_new = open("qureshi_trees_abbrevs.txt", "w")

for i in range(len(data_new)):
    temp = generate_abbreviations(data_new[i])
    out_new.write(" ".join(data1_new[i]) + "\n")

    if len(temp) > 0:
        m = min(temp.items(), key=lambda x: x[1])[1]
        for j in temp.keys():
            if temp[j] == m:
                out_new.write(j + " ")
    out_new.write("\n")
out_new.close()
file_new.close()
print("Success")
```