**A SUMMARY REPORT**

**BACKGROUND**

The program is designed to generate three-letter abbreviations for names listed in a file, adhering to specific rules, and assign scores to these abbreviations based on the position and value of the letters within the words. The following is a brief description of how the problem was approached and solved:

**PROBLEM OVERVIEW:**

* The assignment was to read names from a file, create three-letter abbreviations for each name, and then grade each name according to position and letter value.
* Rules had to be followed while using abbreviations; the first letter of the name had to be followed by two more letters in that order.

Troubleshooting Process: To expedite the debugging and analysis of the work, a Python script (**oge.py**) and numerous output files (**ike\_trees\_abbrevs.txt, ikee\_trees\_abbrevs.txt, ikeee\_trees\_abbrevs.txt, abbreviations\_output.txt, trees\_abbrevs.txt**) were used during the troubleshooting process. This multimodal method with several output files for analysis made a substantial contribution to the thorough debugging process, allowing for iterative improvements, code adjustments, and effective result analysis.

**Steps in the Solution:**

1. Reading Files: Two input files, **trees.txt** and **values.txt**, containing names and values, were read by the programme.txt with values for the letters.

After reading **values.txt**, the **read\_letter\_values** function builds a dictionary that associates each letter with its corresponding value.

2. Generating Abbreviations: For every name, the **generate\_abbreviations** function creates an acronym:

Split names into words, capitalises words, and ignores characters that are not letters.

It generates abbreviations beginning with the first three letters for words consisting of three or more characters.

Make sure that the initial letter of the word appears first in every abbreviation, then the next two letters in sequence.

3. Scoring Abbreviations: According to a set of principles, the **compute\_score** function determines scores for letters contained in abbreviations.

A word's initial letter receives a score of 0.

Accordance to predetermined criteria, assigns scores to letters depending on their values and places inside the word.

4. Choosing the Best Abbreviations: For every distinct combination of the second and third letters, the **choosing\_best\_abbreviations** function chooses the best abbreviation with the lowest score.

It guarantees that a single abbreviation is kept for each distinct letter combination.

5. Generating Output: The original names and the best abbreviations for them are written to an output file (ikeee\_trees\_abbrevs.txt) using the **generate\_output** function:

The names are read from **trees.txt**, abbreviations are created, the best ones are chosen, and both the original names and the abbreviations are written to the output file.

6. Main Function:

The program's entry point is the **main** function:

By contacting other functions, it creates the output file, reads letter values, and defines the input files.

In summary, the program follows certain scoring and abbreviation construction principles while processing input files quickly to produce three-letter abbreviations for names.

It makes sure that every abbreviation begins with the initial letter of the name and proceeds sequentially through the next two letters.

Lastly, it adds the original names to the output file for future reference or analysis, along with their best abbreviations, according to the specified rules.

**REFERENCES**:

* Automate the Boring Stuff with Python by Al Sweigart
* Fluent Python by Luciano Ramalho
* Python Crash Course" by Eric Matthes, 3RD Edition, A Hands on, Project Based