Geowox – data engineer assignment

By: Kunal Chaturvedi

Contents

[Assumptions: 2](#_Toc62244985)

[Architecture: 2](#_Toc62244986)

[Code Structure and Clarity: 2](#_Toc62244987)

[Execution: 2](#_Toc62244988)

[Testing: 3](#_Toc62244989)

Assumptions: I have assumed that since there can be multiple tragedies in the play that need to be converted to tragedy, it would become tedious for the user to enter it manually. It would be better that the tragedies are stored in a **text file** and name and location of the file be given by the user.

The format of the text file will be as defined in the problem; however, I have included some exception handling pertaining to erroneous entries in the file.

## Architecture:

* I have written the code in Python language where Python 3.7.3 version is used.
* IDE used is Spyder.
* The Entire code is kept in Github repository to track the progress and changes made.

Code Structure and Clarity: I have used **modular programming** where different functions have been made for specific tasks and processes. Following this approach has helped me to debug errors and adding new test cases for successful compilation of the program.

Standard naming conventions for variables are used for comprehending the code easily by anyone.

Execution: The entire functionality of achieving the desired task is present in functions:

1. **primary\_logic()**
2. **rev()**

The exception handling scenarios are present in functions:

1. **check\_file\_dimensions()**
2. **check\_file\_legality()**
3. **check\_file\_dirty\_lines()**

## Testing:

|  |  |  |  |
| --- | --- | --- | --- |
| S no. | Test Case Description | Pass/Fail | Function in code |
| 1. | Checking that the number of lines defined in the first line of the input file matches the actual number of the lines with 2 numbers separated by space. | PASS | check\_file\_dimensions() |
| 2. | Checking the input file has no non-integer character. (expect wide-space and next line) | PASS | check\_file\_legality() |
| 3. | Checking that the input file has all lines after the first one in the desired format of two numbers separated by space. | PASS | check\_file\_dirty\_lines() |
| 4. | Checking that there is no empty line in between any two correctly formatted lines provided in the input file, when the line count provided in the first line **matches** the actual number of lines present in the file. | PASS | check\_file\_dirty\_lines() |
| 5. | Checking that there is no empty line in between any two correctly formatted lines provided in the input file, when the line count provided in the first line **does not match** the actual number of lines present in the file. | PASS | check\_file\_dimensions() |