SENG265, FALL 2020
ASSIGNMENT 2
UNIVERSITY OF VICTORIA

Due: Oct 27, 2020 by 10:00 PM, by "git push".
(Late submissions **not** accepted)

1 Assignment Overview

This assignment involves writing a program to solve the same problem as Assignments 1 except using Python 3. As a reminder, your programs will read lines of text from a given file, compute the frequency of words of certain length from the file and print these frequencies to standard output.

The overall goal of this assignment is to familiarize yourself with programming in Python.

Section 2 contains the `Specifications` for the program. Section 3 describes the (new) `Constraints` you should consider in your implementation, and Section 4 describes the `Testing` component. `What you should Submit` is outlined in Section 5 and the `Evaluation` scheme is given in Section 6.

Your code is expected to run without warnings **in the course virtual machine *Senjhalla*** using **Python 3.9.6**.

2 Implementation

The assignment involves writing a program to solve the same problem as Assignments 1 except using **Python 3**. It includes optional arguments to sort and print the words of the same length, exactly as in Assignment 1.

```
$ python word_count.py --sort --print-words --infile <input_file>
```

Recall that there are four possible options to run the program with these arguments:

`$ python word_count.py --infile <input_file>` or,
`$ python word_count.py --sort --infile <input_file>` or,
`$ python word_count.py --print-words --infile <input_file>` or,
`$ python word_count.py --sort --print-words --infile <input_file>`

You may assume that if `<input file>` exists, it is passed immediately after `--infile`. You **cannot** assume that the arguments will be run in this order.

The same test files will be used during the evaluation so please review the expected output formatting.

3 Constraints

You may only use python3 modules that are installed using the *Vagrantfile* for your VM *Senjhalla*. If a python3 module is not installed, you may not use that module in your code.

4 Test Inputs

You should test all of your programs with a variety of test inputs, covering as many different use cases as possible, not just the test input provided. You should ensure that your programs handle error cases (such as files which do not exist) appropriately and do not produce errors on valid inputs. Since thorough testing is an integral part of the software engineering process, you will be provided test input.

Provided For You

There will a `assign2.zip` package available on Connex. Download and extract to the `a2` folder in your git repository. The folder will consist of: `cases`, `test`, `run_all.sh`, and `src/word_count.py`. **Do not move word_count.py from the `src` directory**.

Like with Assignment 1, do **not** modify the `main()` function description. It is required for the test framework to "hook" into your solution.

For this assignment, you will be using the same test cases as from Assignment 1. You are also provided a test framework to test your code. We will be using the python `unittest` module to run tests of the code, in the same manner as in Assignment 1. These test files are located in the `test` folder. There are three test files `test_A.py`, `test_B.py`, and `test_C.py`, corresponding to the parts A,B,C from Assignment 1.

There will be no tests provided for argument handling but your solution is expected to work as described in Section 2.

Your solution is expected to work for all 4 permutations of `--sort`, `--print-words` and `--infile`.

To run an each part (A,B,or C) individually:

```
python test/test_A.py
python test/test_B.py
python test/test_C.py
```

A `BASH` script has been provided for you in order to run all the tests:

```
bash run_all.sh
```

Deciphering test output

In Fig. 1, we see an example of the results for `python test/test_B.py`. The summary shows that the solution passed the first 3 three tests, `ok`. The `FAIL` indicates that the solution ran without error, but produced incorrect output. An `AssertionError` is returned by the test framework indicated the two strings (expected and actual output) do not match. The `ERROR` indicates that the solution could not run to completion, and exited due to an error (exception). The test framework will show the error message and code line number where the error occurred.

```
test_t01_empty_file (base_test.WordCountTest) ... ok
test_t02_single_word (base_test.WordCountTest) ... ok
test_t03_long_word (base_test.WordCountTest) ... ok
test_t04_multi_words (base_test.WordCountTest) ... FAIL
test_t05_multi_line (base_test.WordCountTest) ... ERROR
test_t06_special_characters (base_test.WordCountTest) ... ERROR
test_t07_extra_spaces (base_test.WordCountTest) ... ERROR
test_t08_long_paragraph (base_test.WordCountTest) ... skipped 'during evaluation marking only'
test_t09_long_paragraphs (base_test.WordCountTest) ... skipped 'during evaluation marking only'
test_t10_super_long (base_test.WordCountTest) ... skipped 'during evaluation marking only'
```

Figure 1: Test framework summary output

```
======================================================================
FAIL: test_t05_multi_line (base_test.WordCountTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/memet/Projects/uvic/seng265/seng265_labs_mz/202009/Assignments/assign2/solution/test/base_test.py", line 71, in t
est_t05_multi_line
    self._test_case(self.arguments + ["cases/t05.txt"], f"cases/{self.test_part}/c05.txt")
  File "/home/memet/Projects/uvic/seng265/seng265_labs_mz/202009/Assignments/assign2/solution/test/base_test.py", line 60, in _
test_case
    self.assertEqual(expected, buf.getvalue(), msg=f"--> {solution}")
AssertionError: 'Count[03]=05;\nCount[02]=04;\nCount[04]=04;\nCou[63 chars].0\n' != 'Count[02]=04;\nCount[03]=05;\nCount[04]=04
;\nCou[63 chars].5\n'
+ Count[02]=04;
  Count[03]=05;
- Count[02]=04;
  Count[04]=04;
  Count[06]=02;
  Count[07]=01;
  Count[11]=01;
- Median word length: 5.0
?                     --
+ Median word length: 3.5
?                     ++
 : --> cases/B/c05.txt
```

Figure 2: Test framework failed test output

Fig. 2 shows the result of a `FAIL` test. The test framework will show you the "diff output" between the expected and actual results.

In the "diff output", you may see these symbols: "-", a "+", "?", or a "^".

- `-` (minus) : this line was expected but not produced by your output (you need to add it)
- `+` (plus) : this line was produced by your output, but not expected (you need to remove it)
- `?` : the test framework has a suggestion (you're almost right)
- `^` : indicate you need to change a character

In Fig. 2 we can see that for `test_t05_multi_line` we've added the bin `Count[02]` in the wrong spot and the median calculation is incorrect.

(HINT: You can also use `diff` to compare your results with the provided expected output)

5 What you must submit

- Python source-code name `word_count.py` which contains your solution for assignment #2.

- Ensure your work is **committed** to your local repository in the provided **a2** folder **and pushed** to the remote **before the due date/time**. (You may keep extra files used during development within the repository.)

6 Evaluation

The teaching staff will primarily mark solutions based on the input files provided for this assignment, as well as additional solution files not provided. Students must adhere to the command execution and output formatting outlined in this assignment. For each assignment, some students will be randomly selected to demo their code to the course markers. Each student will have this opportunity to demo at least one assignment during the semester. Sign-up procedure will be discussed in class.

In addition to automated testing, your code will be evaluated based on:

- Proper handling of different command line argument passing
- Good coding practices (i.e. good variable/function naming, use of functions when appropriate, limited globals, etc.)

Our grading scheme is relatively simple.

- "A+" grade: A submission completing ALL Parts and all requirements of the assignment and all tests pass. The `word_count.py` programs runs without any problems.
- "A/A-" grade: A submission completing ALL Parts and all requirements of the assignment and test 1-7 pass. Most of the 8-10 tests pass. The `word_count.py` programs runs without any problems.
- "B+" grade: A submission that completes ALL parts of the assignments and tests 1-7 pass. Some of the 8-10 tests pass. The `word_count.py` programs runs without any problems.
- "B-/B" grade: A submission that completes part A & part B of the assignment and all tests (1-7) pass. Most or some of the tests 8-10 pass. The `word_count.py` programs runs without any problems.
- "C" grade: A submission that completes part A of the assignment and passes all of the tests (1-7). The `word_count.py` programs runs with some problems.
- "D" grade: A serious attempt at completing requirements for the assignment. The `word_count.py` program compiles and runs with some problems.
- "F" grade: Either no submission given (or did not attend demo); submission represents very little work or understanding of the assignment.