CITS5501 Software Testing and Quality Assurance
Semester 1, 2018
Workshop 2

- Document version number: 0.0.1
- Date: 2018-03-15

This workshop is worth 5% of your final grade. It is due on Friday 23 March, 5pm, and should be submitted via `cssubmit`. All work is to be done individually.

You are expected to have read and understood the University's guidelines on academic conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort.

You must submit your project before the submission deadline above. The penalties for late submission are described in the University's guidelines on assessment: For late submissions a penalty of 10% of the total mark allocated to the assessment item must be deducted per day for the first 7 days (including weekends and public holidays). After 7 days the assigned work will not be accepted and will receive a mark of zero (unless an application for mitigation is approved).

Overview

This workshop requires you to:

- develop simple Python functions, according to a specification
- write unit tests and `doctest` tests for the functions
- describe your rationale for choosing the unit tests
- identify some of the errors which your tests identify

Your code should be able to be run using a Python 3.6 interpreter. The Thonny IDE comes with Python 3.6 built in, so you are welcome to use that to check that your scripts can be run successfully.

Tasks

1. Write a python script in a file named `deduplicate.py`, which contains two functions: `deduplicate_line()` and `count_distinct_words()`.

   `deduplicate_line(string)`:

   - This function takes a string as input which contains words separated by spaces, and returns a string containing the same words, separated by spaces – but only the *first occurrence* of each distinct word. (Subsequent, duplicate occurrences are dropped.) The words which are returned should be in the same order as their first appearance in the input string.

     A "word" is defined as any sequence of contiguous, non-whitespace characters.

     For example:

     ```
     >>> myString = 'the the quality of mercy'
     >>> deduplicate_line(myString)
     'the quality of mercy'
     ```

   `count_distinct_words(strings)`:

   - This function takes a list of strings as input, each of which contains spaces separated by words (as described for `deduplicate_line(string)`. It should return the number of distinct words contained in the list of strings.

     For example:

     ```
     >>> myList = ['The quick brown fox', 'The lazy dog']
     >>> count_distinct_words(myList)
     6
     ```

   You should write doctests (and documentation) for each function, and code to ensure those doctests are executed when `deduplicate.py` is run as a script. The documentation is not required to be extensive, but if you need suggestions, you may refer to http://docs.python-guide.org/en/latest/writing/documentation/#writing-docstrings.

2. Write a python script named `test_deduplicate.py` containing unit tests for your `deduplicate.py` script.

   It should contain a class called `TestDeduplicateFunctions` that inherits from `unittest.TestCase` and contains test cases for your `deduplicate_line` and `count_distinct_words` functions.

   It should include code that executes the tests when `test_deduplicate.py` is run as a script.

3. Submit a text file named `test_rationale.txt`, which documents briefly (500 words or less) your rationale for choosing the unit tests in `test_deduplicate.py` that you

did, and which describes some of the errors that are detected by those unit tests. (You can give fragments of sample, erroneous code to illustrate this, if you like.)

Validation

A script, `check.py` is provided which runs a few basic tests on your Python files (checking that they have the correct names, can be successfully compiled, contain functions or classes with the specified names, and so on).

You are welcome to use this to see if your submission satisfies at least these minimal requirements, but the checks it performs are not comprehensive – the fact that `check.py` executes without error does not guarantee your submission meets all the requirements.

Submission

Submit your three files, `deduplicate.py`, `test_deduplicate.py` and `test_rationale.txt` via `cssubmit`.

Assessment

The submission will be marked out of 5.

- 1 mark: `deduplicate.py` contains functions which meet the specifications.
- 1 mark: `deduplicate.py` contains appropriate doctests and documentation.
- 1 mark: `test_deduplicate.py` contains appropriate unit tests.
- 2 marks: `test_rationale.txt` contains a rationale which meets the requirements above and is clear and logically structured.

To be awarded any marks, a Python file must compile without error using Python 3.6, and must execute appropriate tests when run – if it does not meet those requirements, it will be awarded 0 marks.