Scripting, Testing and Version Control

2810ICT— Software Technologies

School of ICT

Griffith University

Trimester 2, 2018

Due at the start of Week 7, Monday 27th August, 9am

Assignment Description

This assignment is a small group based assignment that requires a Python program to be completed that demonstrates understanding of the scripting languages taught in the course. As part of completing the program, existing functionality will need to be tested to ensure it is working correctly and new functions will be added to the program. A version control system shall be used to build the project and create and manage changes to source code over time.

It is important to note that submission of this assignment is a requirement for passing the course. Late submissions will be marked according to Griffith University's assessment policy. 10% of the overall mark will be deducted for each business day late. After 5 days, no submissions will be accepted.

Group Requirements

This assignment should be completed in pairs (groups of 2). If you are unable to work in a pair or have any specific issues, you should discuss this with your campus convenor. Make sure you include the name and student number of both group members in your documentation.

Submission Requirements

This assignment must be submitted online via L@G under the assessment page. Only 1 submission per group is needed. Your submission should include;

- A word document (preferred), PDF Is also acceptable, other formats are not allowed. The provided template shall be used.
- A .py script file containing the final version of the code.
- A .py file containing unit test code and test cases.

Problem Statement

The goal of a ladder-gram is to transform a source word into the target word in the least number of steps. During each step, you must replace one letter in the previous word so that a new word is formed, but without changing the positions of the other letters. All words must exist in the supplied dictionary (dictionary.txt). For example, we can achieve the alchemist's dream of changing “lead” to “gold” in 3 steps (lead->load->goad->gold), or “hide” to “seek” in 6 steps (hide->side->site->sits->sies->sees->seek).

You have been supplied with a Python program (word_ladder.py) for this problem that was written by an obviously brilliant Python programmer who unfortunately did not believe in documentation nor that users would make any mistakes. Even worse, this code has been modified by a somewhat less talented programmer and it no longer works as efficiently as it used to.

Your assignment is to:

- Produce a software design document based on the sample template for this program. In particular you must document each function within the program stating clearly what they do and how they do it.
- You must modify the Python code so that:
    - ALL possible user input errors are handled correctly.
    - The program performs as it did before the less talented programmer got his hands on it. In particular it must go from “lead” to “gold” in 3 steps and “hide” to “seek” in 6 steps.
- You must use the Python unittest module to test the functions of your program and provide test cases for each function. Note that to facilitate the testing, you can rewrite the code to make it testable.
- You must make all program changes using a configuration management tool (BitBucket recommended) such that each change can be tracked and easily undone.
- For full marks, you must add the following functionality to the program:
    - The user can supply a list of words that are not allowed to be used within the steps from a start word to a target word.
    - The user has the option of asking for the shortest path from a start word to a target word.
    - ***For 7810ICT students***: The user has the option of asking for a list of all possible unique paths (i.e. no common words) of a certain length from the start word to the target word.

Marking

This assignment is worth 20% of your final grade. The assignment will be marked out of 100 and marks will be allocated as follows:

- Documentation of software and testing: 30 marks
- Fixing ladder-gram functionality: 20 marks
- Correct use of the unittest module and correct acceptance testing: 20 marks
- Use of version control: 10 marks
- Implementation of additional features including testing: 20 marks

Additionally, marks will be deducted for incorrect:

- Naming conventions
- Poor English/grammar in the documentation
- Lack of comments in the code

Please note that all submitted assignments will be analysed by a plagiarism detector that is specifically designed for assignment submissions containing program source code.