Marking Sheet of Assignment 1 of BCPR301

Student Name/ ID: Simon Broekman

Other group members: Simon England,

# You MUST supply (i.e., ZERO mark if not)

1. A design-level class diagram of your proposed program. If the input of your program is a diagram in textual modeling language/ notation, you have to submit the current textual version of the class diagram required as well. And

1. A help file details the commands provided by your line-oriented command interpreter and the lecturer must approve these before you start the coding for this assessment. And

1. Your program must be able to do all the tasks mentioned in the section of Problem domain. And

1. Your code MUST comply with the Python style (i.e., being able to pass PEP8 check). And

1. A document to list (for each component claimed for marks in your program): a) the ownership (i.e., done by you or someone else?); b) self-reflection on robustness[[1]](#footnote-1); and c) self-reflection on the completeness and implementation. And

1. You must carry out version control in a repository during your development process. And

1. A filled self-marking sheet.

Your Online Repository Link

https://github.com/SimonJB07/Python---301-Project

# The textual modeling language/ notation used in your solution Marking guide (max 60 marks in total)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Features | | Used by your peers (2  mark) | Robustness (2 mark) | Complete and well implemented, i.e., “What is clever about this?” (2 mark) | | Marks |
| 1 | Support command‐line arguments | | 0 | 0 | 0 | | 0 |
| 2 | Has a line‐oriented command interpreter based on cmd | | 0 | 0 | 0 | | 0 |
| 3 | Display command line help of available commands | | 0 | 0 | 1 | | 1 |
| 4 | Change commands and  options | | 0 | 0 | 0 | |  |
| 5 | Extract data | | 0 | 2 | 2 | | 4 |
| 6 | Validate data | | 0 | 1 | 1 | | 2 |
| 7 | Provides object‐persistence / object serialization using either pickle or shelve | | 0 | 1 | 0 | | 1 |
| 8 | Can load data from a file | | 0 | 2 | 2 | | 4 |
| 9 | Can raise exceptions and provide exception handling | | 0 | 1 | 1 | | 2 |
| 10 | Amount of error trapping & handling | | 0 | 1 | 1 | | 2 |
| 11 | Provide doctests | | 0 | 2 | 2 | | 4 |
| 12 | Provide unittests | | 0 | 2 | 2 | | 4 |
| 13 | Breadth of test coverage | | 0 | 2 | 2 | | 4 |
| 14 | Can deal with directories and file locations | | 0 | 0 | 0 | | 0 |
| 15 | Pretty print, i.e., displaying data in bar chart, pie chart, UML diagram component, etc. | | 0 | 0 | 0 | | 0 |
| 16 | Can save and read data from a database | | 0 | 0 | 0 | | 0 |
|  | Total | |  | 14 | 14 | | 28 |
|  | |  | | **Marks** | |  | |
|  | | 0 | | 1 | | 2 | |
| **Used by peers** | | Not used by any peer | | Half of the team members use | | All team members use | |
| **Robustness** | | Not be able to run during demonstration | | Encounter some exceptions during demonstration | | Encounter ZERO exception during  demonstration | |
| **“Complete and well implemented” marking rubric except for doctest, unittest and test coverage tasks** | | Not complete | | Complete, but not very Pythonic | | Complete and very Pythonic | |
| **“Complete and well implemented” marking rubric for doctests task** | | No doctest | | < 5 different doctests | | >= 5 different doctests | |
| **“Complete and well implemented” marking rubric for unittests task** | | No unittest | | < 5 different unittests | | >= 5 different  unittests | |
| **“Complete and well implemented” marking rubric for “breadth of test coverage” task** | | < 18 different doctests and/or unittests | | >= 18 different doctests and/or unittests | | >= 36 different doctests and/or unittests | |
| **“Robustness” marking rubric for doctest, unittest and test coverage tasks** | | No testing case pass | | Half of the testing cases pass | | All testing cases pass | |

1. **Robustness**. The degree to which a system continues to function in the presence of invalid inputs or stressful environmental conditions. [↑](#footnote-ref-1)