Marking Sheet of BCDE321 Assessment 2

Student Name/ ID Harry Lo / 99127713 .....

Other group members Matthew Gordon & John Quiamco .....

Requirements for Submission

Every student MUST submit the followings as a single .zip file into the dropbox on course Moodle site by the deadline indicated; otherwise, ZERO mark will be given.

1 A class diagram of your proposed program. And

2 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

3 Your program must be able to do all the tasks mentioned in the section of Problem Domain. Please note that here displaying data does not mean simply outputting the data as a 2D table. And

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

5 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]; and c) self-reflection on the completeness and implementation. And

6 You must carry out version control in an online repository during your development process. The URL link of your online repository needs to be provided in your self-marking sheet. And

7 A filled self-marking sheet.

URL Link of Your Online Repository

[1] **Robustness**. The degree to which a system continues to function in the presence of invalid inputs or stressful environmental conditions.

Marking guide (max 60 marks in total)

	Component	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	2	2	2	6
2	Has a line-oriented command interpreter based on cmd or similar package	2	2	2	6
3	Display command line help of available commands	2	2	2	6
4	Change commands and options	2	2	2	6
5	Extract data	2	2	2	6
6	Validate data	2	2	2	6
7	Provides object-persistence / object serialization using either pickle or shelve	0	0	0	0
8	Can load data from a file	2	2	2	6
9	Can deal with file directory	2	2	2	6
10	Can raise exceptions and provide exception handling	2	2	2	6
11	Amount of checking for pre- and post- conditions of methods	2	2	2	6
12	Provide doctests	2	2	1	5
13	Provide unittests	2	2	1	5
14	Pretty print, i.e., displaying data in chart/ diagram, e.g., bar chart, pie chart, UML diagram, etc.	2	2	2	6
15	Can save and read data from a database, e.g., SQLite, MySQL and MongoDB				
	Total				

	Marks		
	0	1	2
Used by peers	Not used by any peer	Half of the team members use	All team members use
Marking Rubric for Robustness			
Provide doctests Provide unittests	No testing case passes	Half of the testing cases pass	All testing cases pass
other components	Not be able to run during demonstration	Encounter some unhandled exceptions during demonstration	Encounter ZERO unhandled exception during demonstration
Complete and well implemented			
Provide doctests	No doctest	<= 10 different doctests	>= 20 different doctests
Provide unittests	No unittest	<= 10 different unittests	>= 20 different unittests
Other components	Not complete	Complete, but not very Pythonic	Complete and very Pythonic