CP1200 SP7 2014 Assignment 2 - Great Golf Game

Task:

You are to plan and then code (in Python 3) the next version of the golf game from assignment 1, as described in the following information and sample output. This assignment has two parts, each of which is the whole program, but the second is a version using object oriented programming.
Part 2 is worth 12.5% of the assignment marks (i.e. 6 out of 48 marks) and you should only attempt this after completing part 1.

Program Features:

The program is similar to assignment 1, but with significant modifications. Ensure that your program has the following features, as demonstrated in the sample output below:

- the program asks the user for their name at the beginning, ensuring it is between 1 and 27 characters
- there is a high scores file system - when a round (hole) is finished, the user has the option of saving their score to a text file called scores.txt. This should simply append the score and name to the end of the file in the same format as the existing scores (see text file provided)
- there is a menu option for displaying the scores, which displays the scores in lowest-to-highest order with the names and scores lining up (name fits in 27 characters, score fits in 2).
- the play game option asks for how many rounds (between 1 and 9) and then plays that many rounds. This input must be error-checked and invalid inputs should not crash the program.
- there are four clubs, details of which must be stored in a dictionary with the club initial letter as the key and a tuple of name and length as the value

Please carefully check sample output to determine how the program should run, and notice things like the error-checking and outputs (including re-printing the club choices when the user hits an air swing) and the fact that the save prompt uses the player's first name only.

Playing the game (each hole) is the same as the first assignment, except for the clubs and their average distances, which are demonstrated in the sample output. Note that the use of a dictionary will change how you use these details. A good way to consider your approach is to write the code so that adding a new club would ONLY require adding a new key-value pair to the dictionary. If your code would require any more changes than that, then it is not done well enough and you should re-think it.

An example scores file (as used for the sample output below) is provided for you on LearnJCU and you must use this format. Each line contains a score then a comma then a space then a name.
You can assume that the file exists and its contents are valid, so you do not need to do any error checking for the file.

Planning:

Write up the algorithm in pseudocode – first! Please do this in a docstring (comment) at the top of your code file after your name, date and brief program details. For each function, you need separate pseudocode as well as a function header that shows any parameters. Follow the guide to good pseudocode and examples presented in the subject to ensure this is done to a high standard.
Your pseudocode should be accurate and up-to-date, so if your coding changes after you have written your planning, then update your pseudocode to match.
You may show this part of the assignment to your tutor during practical time to get comments or suggestions. It is important to note that you can only get help from staff in practical time after your prac work is finished, and only if you show your planning.
No help will be given with code without seeing your planning first.

Coding Requirements:

Use the techniques and patterns that you have learned and seen demonstrated in class.

- Use functions appropriately to implement top-down design. Carefully think about the inputs and outputs for functions and ensure that they are useful and reusable, as discussed in lectures.
- Start with one function for each menu option (except quit and instructions), plus use others as appropriate. E.g. a function like `getValidInt` might be suitable. Don't put user input or print statements inside a function unless that is what the function is for. Focus on reusability.
- Do not use global variables. Global CONSTANTS are appropriate, and you should use a constant for the clubs dictionary.
- Include meaningful docstrings for all of your functions as according to the Python style guide: See: http://legacy.python.org/dev/peps/pep-0008/ and http://legacy.python.org/dev/peps/pep-0257/
- Use inline comments (start with # and put them on the line above the code they are referring to) for anything that could benefit from some extra explanation, but do not use unhelpful ("noisy") comments.
- Use exceptions (`try: except:`) for error-checking number inputs as shown in class.
- Close any files you open, at the appropriate spots in your program.
- Use a default parameter (scores.txt) for the filename in the functions that use it. The user is not asked for it, but having it as a default parameter to the functions rather than hard-coding it inside the functions means you could extend the program to use different filenames without having to rewrite those functions at all (this is an example of extensibility – thinking ahead).
- Note that dictionaries are unsorted. The order that clubs are displayed in the sample output is indefinite. Your program may display the clubs in any order. If you would like an extra (unmarked) challenge, then you may get the clubs to print in distance order.
- Learn from the marks and feedback from your first assignment. Don't repeat any mistakes.

For part 2, complete the assignment to the exact same requirements, but by writing and using a Club class (each club will be an object of type Club). Store the class in a separate file called club.py. This class only needs basic init, get and set methods.

Output Requirements:

Sample output from the program is provided below. Ensure that your program matches the sample output exactly (except for your name and the date, and the random values). Do not add or remove anything from the program even if you think it would make it better. Aim to have your work look just like the sample output including spaces, spelling – everything! Think of this as a helpful way to guide and verify your development as well as training you to be particular and pay attention to detail.

Submission:

Hand in one Python 3 (.py) code file containing your pseudocode at the top.
Please name the file like: **FirstnameLastnameA2.py**
e.g. if your name were Miles Davis, the filename would be MilesDavisA2.py
If you chose to complete parts 1 and 2, then you should submit one single zip file containing three Python files: club.py, FirstnameLastnameA2.py (as above) and FirstnameLastnameA2part2.py
Submit your single file (either one .py or one .zip) by uploading it on LearnJCU under Assessment (click on the title of the assignment to access the submission facility).

Due:

Submit your assignment by the date and time specified on LearnJCU.
Submissions received after this date will incur late penalties as described in the subject outline.

Integrity:

The work you submit for this assignment must be your own. You are allowed to discuss the assignment with other students and get assistance from your peers, but you may not do anyone else's work for them and you may not get anyone else to do any part of your work. Programs that are detected to be too similar to another student's work will be dealt with promptly according to University procedures for handling plagiarism.
If you require assistance with the assignment, please ask **general** questions on the LearnJCU discussion board, or get **specific** assistance with your own work by talking with your lecturer or tutor.

Sample Output:

Bold text below shows user input for this example.

```
Let's play great golf, CP1200 style!
Written by Lindsay Ward, July 2014
Please enter your name:       <blank user input here…>
Your name can not be blank and must be <= 27 characters
Please enter your name: Joe is having fun playing your golf game :)
Your name can not be blank and must be <= 27 characters
Please enter your name: Joe Satriani

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> r
Invalid menu choice.

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> I
Instructions: It's golf on your console. Each shot will vary in distance around
its average.
Club selection:
7 for 7 Iron (30)
D for Driver (100)
P for Putter (10)
3 for 3 Iron (60)

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> v
Nuno Bettencourt             3
Jimi Hendrix                 4
Lincoln Brewster             5
Eric Gales                  12

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> P
How many rounds would you like to play? (1-9) 10
Invalid number of rounds.
How many rounds would you like to play? (1-9) Why can't I enter text?
Invalid number of rounds.
```

```
How many rounds would you like to play? (1-9) 0
Invalid number of rounds.
How many rounds would you like to play? (1-9) 2

Round 1
This hole is a 230m par 5
Club selection:
7 for 7 Iron (30)
D for Driver (100)
P for Putter (10)
3 for 3 Iron (60)
You are 230m from the hole, after 0 shot/s
Choose club: d
Your shot went 114m
You are 116m from the hole, after 1 shot/s
Choose club: D
Your shot went 82m
You are 34m from the hole, after 2 shot/s
Choose club: i
Invalid club selection = air swing :(
Club selection:
7 for 7 Iron (30)
D for Driver (100)
P for Putter (10)
3 for 3 Iron (60)
Your shot went 0m
You are 34m from the hole, after 3 shot/s
Choose club: 7
Your shot went 27m
You are 7m from the hole, after 4 shot/s
Choose club: p
Your shot went 8m
You are 1m from the hole, after 5 shot/s
Choose club: P
Your shot went 1m
Clunk... After 6 hits, the ball is in the hole!
Disappointing. You are 1 over par.
Would you like to save your score, Joe? (Y/N) no
Please enter Y or N
Would you like to save your score, Joe? (Y/N) y
Score saved. New high scores:
Nuno Bettencourt             3
Jimi Hendrix                 4
Lincoln Brewster             5
Joe Satriani                 6
Eric Gales                  12

Round 2
This hole is a 230m par 5
Club selection:
7 for 7 Iron (30)
D for Driver (100)
P for Putter (10)
3 for 3 Iron (60)
You are 230m from the hole, after 0 shot/s
Choose club: 7
Your shot went 25m
You are 205m from the hole, after 1 shot/s
Choose club: 3
Your shot went 58m
You are 147m from the hole, after 2 shot/s
Choose club: P
Your shot went 8m
You are 139m from the hole, after 3 shot/s
Choose club: d
Your shot went 87m
You are 52m from the hole, after 4 shot/s
```

```
Choose club: 7
Your shot went 24m
You are 28m from the hole, after 5 shot/s
Choose club: p
Your shot went 11m
You are 17m from the hole, after 6 shot/s
Choose club: D
Your shot went 97m
You are 80m from the hole, after 7 shot/s
Choose club: 3
Your shot went 59m
You are 21m from the hole, after 8 shot/s
Choose club: 7
Your shot went 29m
You are 8m from the hole, after 9 shot/s
Choose club: p
Your shot went 8m
Clunk... After 10 hits, the ball is in the hole!
Disappointing. You are 5 over par.
Would you like to save your score, Joe? (Y/N) n

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> v
Nuno Bettencourt             3
Jimi Hendrix                 4
Lincoln Brewster             5
Joe Satriani                 6
Eric Gales                  12

Golf!
(I)nstructions
(V)iew scores
(P)lay round
(Q)uit
>>> Q
Thanks for playing.
```

Marking Scheme:

Ensure that you follow the processes and guidelines taught in class in order to produce high quality work. Do not just focus on getting the program working. This assessment rubric provides you with the characteristics of exemplary, competent, marginal and unacceptable work in relation to task criteria.

Criteria	Exemplary	Competent	Marginal	Unacceptable
Planning **Pseudocode for algorithms**	**9** Clear, well-formatted, consistent and accurate pseudocode that completely and correctly solves the problem, split into functions.	**6** Some but not many problems (e.g. an incomplete solution, inconsistent use of terms, inaccurate formatting, not separated into functions).	**3** Many problems (e.g. an incomplete solution, inconsistent use of terms, inaccurate formatting, not separated into functions).	**0** Too many problems or pseudocode not done.
Program Execution **Correct gameplay**	**6** Game works correctly for all inputs including looping and calculations.	**4** Game mostly works correctly for all inputs including looping and calculations, but there is/are some small problem(s)	**2** Game mostly works incorrectly for all inputs including looping and calculations - there are some significant problem	**0** Game not done or very poorly done.
Loading and displaying scores	**3** Correct functionality, as instructed.	**2** Some aspect of this functionality is not correct.	**1** Significant aspects of this functionality are not correct.	**0** This functionality is not done or very poorly done.
Saving scores	**3** Correct functionality, as instructed.	**2** Some aspect of this functionality is not correct.	**1** Significant aspects of this functionality are not correct.	**0** This functionality is not done or very poorly done.
Error checking	**3** Invalid inputs are handled correctly as instructed, for all user inputs including menu and club selection.	**2** Invalid inputs are mostly handled correctly as instructed, but there is/are some small problem(s), e.g. only checking once instead of looping.	**1** Invalid inputs are mostly handled incorrectly as instructed - there is/are some significant problem, e.g. only checking once instead of looping.	**0** Error checking is not done.
Similarity to sample output (including all formatting)	**3** All outputs match sample output perfectly, or only one minor difference, e.g. wording, spacing.	**2** Between two and three differences (e.g. typos, spacing) in program output compared to sample output.	**1** More than three differences in program output compared to sample output.	**0** No reasonable attempt made to match sample output.
Quality of Code **Identifier naming**	**3** All function, variable and constant names are appropriate, meaningful and consistent.	**2** One or two function, variable or constant names are not appropriate, meaningful or consistent.	**1** Three or four function, variable or constant names are not appropriate, meaningful or consistent.	**0** More than four function, variable or constant names are not appropriate, meaningful or consistent.
Use of code constructs	**3** Appropriate and efficient code use, including no unnecessary duplication, good logical choices for loop control (not while True), good use of constants, etc.	**2** Mostly appropriate code use but with one or two problems, e.g. unnecessary duplication, while True or other poor loop control, incomplete use of constants.	**1** Three or four problems, e.g. unnecessary duplication, while True or other poor loop control, incomplete use of constants.	**0** More than four problems with code use.
Use of functions	**3** Two functions correctly used, as instructed.	**2** Two functions used, but not correctly, e.g. incorrect parameters or function calls or main code outside main function.	**1** Only one function used or other significant problems with functions.	**0** No functions used or functions used very poorly.

Code readability	**6** All formatting is appropriate, including correct indentation and helpful line spacing. Code contains inline comments where helpful, and docstring at top containing program details (name, date, basic description).	**4** Some problems with formatting reduces readability of code. Comments are mostly appropriate, but there is some noise (too many comments) or some missing program details in top docstring or some inappropriate or missing inline comments	**2** Significant problems with formatting reduces readability of code. Comments are mostly inappropriate - noise (too many comments) or missing program details in top docstring or some inappropriate or missing inline comments.	**0** Readability is poor due to formatting problems. Commenting is very poor either through having too many comments (noise) or too few comments (missing program details in top docstring or missing inline comments).
Part 2 – Object Oriented	**6** Correctly uses Score class including appropriate constructor and other methods, and class in separate file.	**4** Mostly correct use of Score class, but some aspect is incorrect and/or class not in separate file.	**2** Attempted, but mostly incorrect (e.g. Score class does more than a score should).	**0** Not attempted.