Zelle 3e Chapter 9 Coding Assignment

General Instructions

My expectations for your work on coding assignment exercises will grow as we progress through the course. In addition to applying any new programming techniques that have been covered in the current chapter, I will be expecting you to follow all of the good programming practices that we have adopted in the preceding weeks. Here is a quick summary of good practices that we have covered so far:

- Include a Python Docstring that describes the intent of the program.
- Place your highest-level code in a function named *main*.
- Include a final line of code in the program that executes the *main* function.
- Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function and the code surrounding that function.
- Choose names for your variables that are properly descriptive.
- Define CONSTANT_VALUES and use them in place of *magic numbers*.
- Always use f-strings for string interpolation and number formatting.
- When processing items from Python lists and tuples, unpack the values into variables with meaningful variable names to avoid using indexed expressions in your code.
- Close all files before the conclusion of the program.
- Remember that your program should behave reasonably when it is not given any input. This might be the result of the user pressing enter at a console prompt. Or, it might be the result of the user providing a an input file that is empty.
- Model your solution after the code that I demonstrate in the tutorial videos.
- Make sure that your test input/output matches the sample provided.
- Create a sub-directory named *data* within your PyCharm project to hold data files.
- Remember to submit all data files with your PyCharm project – including the files that were provided as starter files to this assignment.
- All functions that are not *main()* should have descriptive, action-oriented names.
- All functions should be of reasonable size.
- All functions should have high *cohesion*, and low *coupling*.
- Remember to test your program thoroughly before submitting your work.
- Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by *if, else*, and *elif* conditions in your program (boundary value tests).
- Use of the *break* statement is allowed but not encouraged.
- Use of the *continue* statement is forbidden.
- Regular expression patterns should be expressed as Python *raw strings*
- Your finished code must be refactored to meet all good program design practices covered in this course.

- Your refactored code must be retested to demonstrate that refactoring has not altered program functionality.

Exercise 1 (Required)
Create a program named *going_to_boston.py*. It should be modeled after the program that I demonstrated in the tutorial (*beat_that.py*). Your program should be different in the following respects:

1. Your program will simulate the playing of the dice game Going to Boston. For game rules, see https://funattic.com/dice-games/ .

2. Your program will simulate a game for 2 players with the number of rounds chosen by the user. Establish a minimum number of rounds that the user can request and a maximum number of rounds. Document these using Python constants. Use these constant values when validating user entries associated with desired number of rounds.

3. If the user signals that they don't want to play any rounds by just pressing enter, or by entering 0 for rounds, then do not simulate game play. Instead, print a message saying that no rounds were requested and end the game. See sample test output below.

4. When simulating game play, make all 3 of the rolls of the dice as specified by the rules and choose highest values from the dice rolled. Do not shortcut this process. The automated version of this game should simulate the way that human players would play the game with actual dice. For game rules, see https://funattic.com/dice-games/ .

5. Do not report game results when no rounds have been played.

6. When reporting game results, take care to print text that is grammatically correct. So, print "Player A has won 1 round." rather than "Player A has won 1 rounds.". See sample test output below.

7. When you have succeeded in demonstrating that your code passes all relevant tests, then look for and exploit any refactoring opportunities. These opportunities should include:

    a. Eliminating duplicate code.
    b. Eliminating dead code (code that is no longer used or has been commented out).
    c. Renaming variables, parameters, and functions.
    d. Improving the interfaces of functions (parameters in, return values out)
    e. Any changes that would make your code more readable, more testable, or more maintainable.

When running a test in which the user requests no rounds by pressing <Enter>, you should expect the following input/output on your console:

```
Welcome to Going to Boston.

Please enter the number of rounds to be played (<Enter> to stop):
No rounds were requested. Come play again soon.
```

When running a test in which the user requests no rounds by entering 0, you should expect the following input/output on your console:

```
Welcome to Going to Boston.

Please enter the number of rounds to be played (<Enter> to stop): 0
No rounds were requested. Come play again soon.
```

When running a test in which the user enters unexpected values for the number of rounds to be played, you should expect the following input/output on your console:

```
Welcome to Going to Boston.

Please enter the number of rounds to be played (<Enter> to stop): hi mom
An integer was expected. You entered hi mom.
Please enter the number of rounds to be played (<Enter> to stop): 2.2
An integer was expected. You entered 2.2.
Please enter the number of rounds to be played (<Enter> to stop): -1
A value between 0 and 1000 was expected. You entered -1.
Please enter the number of rounds to be played (<Enter> to stop): 1001
A value between 0 and 1000 was expected. You entered 1001.
Please enter the number of rounds to be played (<Enter> to stop): 2

Playing Round 1:
Player A's turn...
Player A rolls [4, 3, 4] and keeps 4.
Player A rolls [3, 3] and keeps 3.
Player A rolls and keeps 3.
==>Player A's turn score is 10
Player B's turn...
Player B rolls [2, 4, 3] and keeps 4.
Player B rolls [5, 4] and keeps 5.
Player B rolls and keeps 3.
==>Player B's turn score is 12
Player B wins the round!
```

```
Playing Round 2:
Player A's turn...
Player A rolls [3, 6, 5] and keeps 6.
Player A rolls [3, 3] and keeps 3.
Player A rolls and keeps 6.
==>Player A's turn score is 15
Player B's turn...
Player B rolls [1, 6, 1] and keeps 6.
Player B rolls [6, 2] and keeps 6.
Player B rolls and keeps 5.
==>Player B's turn score is 17
Player B wins the round!

Game Results:
Player A has won 0 rounds.
Player B has won 2 rounds.
Player B wins the game!
```

The following console input/output depicts a typical run that includes a tie for a round and a tie for the game:

```
Welcome to Going to Boston.

Please enter the number of rounds to be played (<Enter> to stop): 5

Playing Round 1:
Player A's turn...
Player A rolls [1, 3, 3] and keeps 3.
Player A rolls [1, 2] and keeps 2.
Player A rolls and keeps 1.
==>Player A's turn score is 6
Player B's turn...
Player B rolls [1, 5, 4] and keeps 5.
Player B rolls [6, 2] and keeps 6.
Player B rolls and keeps 6.
==>Player B's turn score is 17
Player B wins the round!

Playing Round 2:
Player A's turn...
Player A rolls [2, 4, 5] and keeps 5.
Player A rolls [6, 1] and keeps 6.
Player A rolls and keeps 4.
==>Player A's turn score is 15
Player B's turn...
Player B rolls [4, 3, 2] and keeps 4.
Player B rolls [2, 5] and keeps 5.
Player B rolls and keeps 2.
==>Player B's turn score is 11
Player A wins the round!

Playing Round 3:
Player A's turn...
Player A rolls [1, 5, 3] and keeps 5.
Player A rolls [4, 2] and keeps 4.
Player A rolls and keeps 4.
==>Player A's turn score is 13
Player B's turn...
Player B rolls [2, 4, 6] and keeps 6.
Player B rolls [1, 3] and keeps 3.
Player B rolls and keeps 2.
==>Player B's turn score is 11
Player A wins the round!

Playing Round 4:
Player A's turn...
Player A rolls [5, 1, 5] and keeps 5.
Player A rolls [5, 5] and keeps 5.
Player A rolls and keeps 2.
==>Player A's turn score is 12
Player B's turn...
Player B rolls [5, 6, 4] and keeps 6.
Player B rolls [6, 3] and keeps 6.
Player B rolls and keeps 4.
==>Player B's turn score is 16
Player B wins the round!

Playing Round 5:
Player A's turn...
Player A rolls [3, 3, 4] and keeps 4.
Player A rolls [5, 1] and keeps 5.
Player A rolls and keeps 5.
==>Player A's turn score is 14
Player B's turn...
Player B rolls [4, 4, 6] and keeps 6.
Player B rolls [4, 5] and keeps 5.
Player B rolls and keeps 3.
==>Player B's turn score is 14
The players tie the round.

Game Results:
Player A has won 2 rounds.
Player B has won 2 rounds.
The players tie the game.
```

The following console input/output depicts a typical run that includes examples of an overall game winner and text for a player who has won 1 round:

```
Welcome to Going to Boston.

Please enter the number of rounds to be played (<Enter> to stop): 5

Playing Round 1:
Player A's turn...
Player A rolls [2, 3, 4] and keeps 4.
Player A rolls [1, 1] and keeps 1.
Player A rolls and keeps 3.
==>Player A's turn score is 8
Player B's turn...
Player B rolls [5, 1, 4] and keeps 5.
Player B rolls [4, 5] and keeps 5.
Player B rolls and keeps 6.
==>Player B's turn score is 16
Player B wins the round!

Playing Round 2:
Player A's turn...
Player A rolls [4, 5, 1] and keeps 5.
Player A rolls [4, 2] and keeps 4.
Player A rolls and keeps 4.
==>Player A's turn score is 13
Player B's turn...
Player B rolls [4, 6, 3] and keeps 6.
Player B rolls [4, 4] and keeps 4.
Player B rolls and keeps 5.
==>Player B's turn score is 15
Player B wins the round!

Playing Round 3:
Player A's turn...
Player A rolls [2, 1, 4] and keeps 4.
Player A rolls [5, 2] and keeps 5.
Player A rolls and keeps 1.
==>Player A's turn score is 10
Player B's turn...
Player B rolls [4, 2, 2] and keeps 4.
Player B rolls [6, 6] and keeps 6.
Player B rolls and keeps 3.
==>Player B's turn score is 13
Player B wins the round!

Playing Round 4:
Player A's turn...
Player A rolls [2, 2, 6] and keeps 6.
Player A rolls [1, 6] and keeps 6.
Player A rolls and keeps 4.
==>Player A's turn score is 16
Player B's turn...
Player B rolls [3, 4, 5] and keeps 5.
Player B rolls [1, 6] and keeps 6.
Player B rolls and keeps 5.
==>Player B's turn score is 16
The players tie the round.

Playing Round 5:
Player A's turn...
Player A rolls [3, 4, 3] and keeps 4.
Player A rolls [3, 4] and keeps 4.
Player A rolls and keeps 5.
==>Player A's turn score is 13
Player B's turn...
Player B rolls [5, 1, 3] and keeps 5.
Player B rolls [2, 3] and keeps 3.
Player B rolls and keeps 2.
==>Player B's turn score is 10
Player A wins the round!

Game Results:
Player A has won 1 round.
Player B has won 3 rounds.
Player B wins the game!
```

Exercise 2 (Optional Challenge Exercise)
Create a program named *going_to_boston_pro_edition.py*. Start by copying the program that was created in Exercise 1 (*going_to_boston.py*).

Your program will be different from the program created in Exercise 1 in following respects:

2. Your program will simulate a game for a number of players chosen by the user. Establish a minimum number of players that the user can request and a maximum number of players. Document these using Python constants. Use these constant values when validating user entries associated with the desired number of players. Please note that while entering the desired number of players, the user will NOT be given the option of stopping the game. At this point, the user is committed to a run of the simulated game.

3. Your program should generate player names using the upper-case letters of the alphabet. The first player will be *Player A*, the next will be *Player B*, and so on.

4. Do not change the logic related to the number of rounds to be played. This should work the same in this new version as it did in the prior version.

5. Note that the potential involvement of more than two players complicates the determination of a winner for each round. Your code should concentrate on determining if there is just one player that has the highest score. In that case, that player wins the round. You should announce that this player has won the round and give them credit for 1 round won. Give the other players credit for 0 rounds won. See sample test output below.

   If there is more than one player with a highest score, then report that "There was no winner for this round." Give all players credit for 0 rounds won. See sample test output below.

6. Note that the potential involvement of more than two players also complicates the determination of the winner of the overall game. Your code should concentrate on determining if there is just one player that has the highest number of rounds won. In that case, report that this player has won the game. See sample test output below.

   Next, your code should concentrate on determining if there is more than one player with the highest number of rounds won. In that case, report that each of these players has tied for the win. See sample test output below.

7. When you have succeeded in demonstrating that your code passes all relevant tests, then look for and exploit any refactoring opportunities. These opportunities should include:

   a. Eliminating duplicate code.
   b. Eliminating dead code (code that is no longer used or has been commented out).
   c. Renaming variables, parameters, and functions.
   d. Improving the interfaces of functions (parameters in, return values out)
   e. Any changes that would make your code more readable, more testable, or more maintainable.

When running a test in which the user enters unexpected values for the number of players, you should expect the following input/output on your console:

```
Welcome to Going to Boston Pro Edition.

Please enter the number of rounds to be played (<Enter> to stop): 2
Please enter the number of players participating: 1
A value between 2 and 8 was expected. You entered 1.
Please enter the number of players participating: 9
A value between 2 and 8 was expected. You entered 9.
Please enter the number of players participating: 2.2
An integer was expected. You entered 2.2.
Please enter the number of players participating: hi mom
An integer was expected. You entered hi mom.
Please enter the number of players participating: 2

Playing Round 1:
Player A's turn...
Player A rolls [5, 2, 4] and keeps 5.
Player A rolls [3, 3] and keeps 3.
Player A rolls and keeps 1.
==>Player A's turn score is 9
Player B's turn...
Player B rolls [3, 5, 6] and keeps 6.
Player B rolls [1, 3] and keeps 3.
Player B rolls and keeps 2.
==>Player B's turn score is 11
Player B wins the round.

Playing Round 2:
Player A's turn...
Player A rolls [5, 2, 1] and keeps 5.
Player A rolls [2, 2] and keeps 2.
Player A rolls and keeps 6.
==>Player A's turn score is 13
Player B's turn...
Player B rolls [2, 5, 2] and keeps 5.
Player B rolls [5, 5] and keeps 5.
Player B rolls and keeps 4.
==>Player B's turn score is 14
Player B wins the round.

Game Results:
Player A has won 0 rounds.
Player B has won 2 rounds.
Player B wins the game!
```

The following console input/output depicts a typical run that includes examples of 4 players and a round in which there is no winner:

```
Welcome to Going to Boston Pro Edition.

Please enter the number of rounds to be played (<Enter> to stop): 2
Please enter the number of players participating: 4

Playing Round 1:
Player A's turn...
Player A rolls [5, 2, 2] and keeps 5.
Player A rolls [2, 2] and keeps 2.
Player A rolls and keeps 1.
==>Player A's turn score is 8
Player B's turn...
Player B rolls [6, 5, 3] and keeps 6.
Player B rolls [6, 2] and keeps 6.
Player B rolls and keeps 1.
==>Player B's turn score is 13
Player C's turn...
Player C rolls [2, 3, 3] and keeps 3.
Player C rolls [5, 2] and keeps 5.
Player C rolls and keeps 4.
==>Player C's turn score is 12
Player D's turn...
Player D rolls [4, 4, 3] and keeps 4.
Player D rolls [3, 6] and keeps 6.
Player D rolls and keeps 3.
==>Player D's turn score is 13
There was no winner of this round.

Playing Round 2:
```

```
Player A's turn...
Player A rolls [6, 4, 2] and keeps 6.
Player A rolls [6, 4] and keeps 6.
Player A rolls and keeps 1.
==>Player A's turn score is 13
Player B's turn...
Player B rolls [3, 4, 1] and keeps 4.
Player B rolls [4, 4] and keeps 4.
Player B rolls and keeps 4.
==>Player B's turn score is 12
Player C's turn...
Player C rolls [6, 1, 5] and keeps 6.
Player C rolls [3, 1] and keeps 3.
Player C rolls and keeps 4.
==>Player C's turn score is 13
Player D's turn...
Player D rolls [3, 1, 6] and keeps 6.
Player D rolls [1, 6] and keeps 6.
Player D rolls and keeps 3.
==>Player D's turn score is 15
Player D wins the round.

Game Results:
Player A has won 0 rounds.
Player B has won 0 rounds.
Player C has won 0 rounds.
Player D has won 1 round.
Player D wins the game!
```

The following console input/output depicts a typical run that includes examples of 6 players and multiple players tying for a win of the game:

```
Welcome to Going to Boston Pro Edition.

Please enter the number of rounds to be played (<Enter> to stop): 4
Please enter the number of players participating: 6

Playing Round 1:
Player A's turn...
Player A rolls [6, 1, 2] and keeps 6.
Player A rolls [1, 6] and keeps 6.
Player A rolls and keeps 4.
==>Player A's turn score is 16
Player B's turn...
Player B rolls [2, 5, 6] and keeps 6.
Player B rolls [6, 3] and keeps 6.
Player B rolls and keeps 1.
==>Player B's turn score is 13
Player C's turn...
Player C rolls [4, 2, 1] and keeps 4.
Player C rolls [5, 2] and keeps 5.
Player C rolls and keeps 2.
==>Player C's turn score is 11
Player D's turn...
Player D rolls [2, 4, 6] and keeps 6.
Player D rolls [4, 4] and keeps 4.
Player D rolls and keeps 6.
==>Player D's turn score is 16
Player E's turn...
Player E rolls [5, 1, 4] and keeps 5.
Player E rolls [3, 4] and keeps 4.
Player E rolls and keeps 5.
==>Player E's turn score is 14
Player F's turn...
Player F rolls [4, 3, 3] and keeps 4.
Player F rolls [6, 4] and keeps 6.
Player F rolls and keeps 3.
==>Player F's turn score is 13
There was no winner of this round.

Playing Round 2:
Player A's turn...
Player A rolls [2, 3, 5] and keeps 5.
Player A rolls [3, 2] and keeps 3.
Player A rolls and keeps 3.
==>Player A's turn score is 11
Player B's turn...
Player B rolls [3, 5, 6] and keeps 6.
Player B rolls [6, 6] and keeps 6.
Player B rolls and keeps 5.
==>Player B's turn score is 17
Player C's turn...
Player C rolls [5, 6, 1] and keeps 6.
Player C rolls [4, 3] and keeps 4.
Player C rolls and keeps 1.
==>Player C's turn score is 11
Player D's turn...
Player D rolls [2, 1, 5] and keeps 5.
Player D rolls [3, 6] and keeps 6.
Player D rolls and keeps 5.
==>Player D's turn score is 16
Player E's turn...
Player E rolls [5, 4, 2] and keeps 5.
Player E rolls [4, 4] and keeps 4.
Player E rolls and keeps 5.
```

```
==>Player E's turn score is 14
Player F's turn...
Player F rolls [5, 6, 4] and keeps 6.
Player F rolls [2, 3] and keeps 3.
Player F rolls and keeps 5.
==>Player F's turn score is 14
Player B wins the round.

Playing Round 3:
Player A's turn...
Player A rolls [2, 5, 6] and keeps 6.
Player A rolls [2, 1] and keeps 2.
Player A rolls and keeps 3.
==>Player A's turn score is 11
Player B's turn...
Player B rolls [6, 3, 4] and keeps 6.
Player B rolls [4, 4] and keeps 4.
Player B rolls and keeps 1.
==>Player B's turn score is 11
Player C's turn...
Player C rolls [2, 3, 4] and keeps 4.
Player C rolls [2, 1] and keeps 2.
Player C rolls and keeps 6.
==>Player C's turn score is 12
Player D's turn...
Player D rolls [3, 2, 4] and keeps 4.
Player D rolls [4, 1] and keeps 4.
Player D rolls and keeps 6.
==>Player D's turn score is 14
Player E's turn...
Player E rolls [2, 4, 5] and keeps 5.
Player E rolls [4, 4] and keeps 4.
Player E rolls and keeps 4.
==>Player E's turn score is 13
Player F's turn...
Player F rolls [3, 5, 5] and keeps 5.
Player F rolls [1, 4] and keeps 4.
Player F rolls and keeps 3.
==>Player F's turn score is 12
Player D wins the round.

Playing Round 4:
Player A's turn...
Player A rolls [6, 6, 4] and keeps 6.
Player A rolls [2, 2] and keeps 2.
Player A rolls and keeps 3.
==>Player A's turn score is 11
Player B's turn...
Player B rolls [3, 6, 2] and keeps 6.
Player B rolls [1, 5] and keeps 5.
Player B rolls and keeps 1.
==>Player B's turn score is 12
Player C's turn...
Player C rolls [4, 6, 6] and keeps 6.
Player C rolls [3, 6] and keeps 6.
Player C rolls and keeps 2.
==>Player C's turn score is 14
Player D's turn...
Player D rolls [1, 4, 6] and keeps 6.
Player D rolls [6, 4] and keeps 6.
Player D rolls and keeps 4.
==>Player D's turn score is 16
Player E's turn...
Player E rolls [5, 2, 6] and keeps 6.
Player E rolls [6, 2] and keeps 6.
Player E rolls and keeps 3.
==>Player E's turn score is 15
Player F's turn...
Player F rolls [1, 2, 6] and keeps 6.
Player F rolls [5, 1] and keeps 5.
Player F rolls and keeps 6.
==>Player F's turn score is 17
Player F wins the round.

Game Results:
Player A has won 0 rounds.
Player B has won 1 round.
Player C has won 0 rounds.
Player D has won 1 round.
Player E has won 0 rounds.
Player F has won 1 round.
Player B has tied for the win.
Player D has tied for the win.
Player F has tied for the win.
```

Tools
Use PyCharm to create and test all Python programs.

Submission Method
Follow the process that I demonstrated in the tutorial video on submitting your work. This involves:

- Locating the properly named directory associated with your project in the file system.
- Compressing that directory into a single .ZIP file using a utility program.
- Submitting the properly named zip file to the submission activity for this assignment.

File and Directory Naming
Please name your Python program files as instructed in each exercise. Please use the following naming scheme for naming your PyCharm project:

```
surname_givenname_exercises_zelle_3e_chapter_09
```

If this were my own project, I would name my PyCharm project as follows:

```
trainor_kevin_exercises_zelle_3e_chapter_09
```

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

```
surname_givenname_exercises_zelle_3e_chapter_09.zip
```

If this were my own project, I would name the zip file as follows:

```
trainor_kevin_exercises_zelle_3e_chapter_09.zip
```

Due By
Please submit this assignment by the date and time shown in the Weekly Schedule.

Last Revised
2022-03-18