CPSC 231 Assignments 01 Due Date: Mon., Sep28th by 11:30 PM

The Game of Pig

Pig is a very simple, but fun dice game invented by John Scarne in 1945. It is normally played by two players, and has an interesting jeopardy decision component during each player's turn.

The rules of Pig are very simple. Two players take turns trying to accumulate points, and the winner is the first player to reach 100 points. At the beginning of a player's turn, a separate turn score is started at zero, and the turn proceeds as follows:

1. The player makes a decision whether to roll (going to step 2) or to hold (going to step 3).
2. If the player decided to roll, he or she rolls a single six-sided die.

   1 – Pig out! The player scores nothing this turn, the turn ends, and the opponent's turn begins.

   2, 3, 4, 5, or 6 – The roll is added to the turn score, and the player continues his or her turn at step 1.

3. If the player decided to hold, the current turn score is added to the player's total score and his or her turn ends.

Players can toss a coin to decide who goes first. The first player to end their turn at or above 100 total points is the winner.

This assignment will be your first use of the Python programming language in this class to implement a version of the game that can be played within the console window on your computer. **Create and submit a separate Python program for each problem.**

Problem 1: Rolling a die (1 marks)

Write a Python program to simulate rolling a single six-sided die. Print the result of the roll to the terminal.

Inputs: None.
Outputs: The result of your die roll.

Example Problem 1 output:
- rolled a 3

Problem 2: Expected Value (2 marks)

If you study probability theory, you may think of the outcome of a die roll as a random variable. The expected value of the random variable long-term average or mathematical expectation of what the outcome would be if you repeated the random process many times.

Write a Python program to compute the expected value of a six sided die roll by simulating many rolls and calculating the average outcome. Use a command line argument to specify the number of simulated trials to perform.

Inputs: The number of rolls to simulate as a command line argument.
Outputs: Your estimation of the expected value of a die roll, presented in a form similar to that shown on the right.

Example Problem 2 invocation:
$ python my-p2.py 1000

Example Problem 2 output:

Rolling 1000 times...

Estimated expectation: 3.481

Problem 3: Expected Rolls (3 marks)

You might also wonder, on average, how many die rolls you can expect to make in a turn of Pig before you "pig out". Write a program to simulate rolling a die and counting the number of rolls before a 1 comes up. Then use the same simulation method from Problem 2 to estimate the expected number of turns before pigging out.

Inputs: A command line argument specifying the number of turns to simulate.
Outputs: Your estimation of the expected number of rolls before pigging out.

Example Problem 3 invocation:
$ python3 my-p3.py 500

Example Problem 3 output:
Simulating 500 turns...
Estimated expectation: 6.432

Problem 4: The Computer's Turn (3 marks)

First, we'd like to program the computer to be able to play a turn of pig. We wouldn't want the computer to keep rolling until it pigs out every turn of course, because it would never score any points! A very simple and good strategy for Pig is to keep rolling until you accumulate 20 or more points in the turn, then hold and keep that as your turn score.

Write a program that simulates a single turn of Pig with the computer using this strategy to play.

Inputs: None.
Outputs: Print the outcome of each die roll as it happens, then print the final score for the computer player at the end of the turn.

Example Problem 4 output:
- rolled a 5
- rolled a 6
- rolled a 6
- rolled a 4
Turn score = 21

Another sample run:
- rolled a 3
- rolled a 2
- rolled a 1
Pigged out!
Turn score = 0

Problem 5: The Computer's Strategy (3 marks)

We can make a little tweak to the strategy to make it do a lot better. For example, if we start the turn with 98 total points, there's really no reason to try to get 20 points because we only need 2 points to win!
Write a program that plays a turn of pig where, given the initial score, it uses the strategy of holding at 20 points or after accumulating enough points to reach 100. This will be our final computer AI's strategy.

Inputs: The current total score entered via a prompt.
Outputs: Print the outcome of each die roll as it happens, then print the turn score and total score for the computer player.

Example Problem 5 output:
Enter current score: 90
- rolled a 2
- rolled a 6
- rolled a 6
Turn score = 14
New total score = 104

Problem 6: The Solitaire Pig (4 marks)

With the computer AI complete, we can test it out by having it play a full game of Pig by itself.

Write a program that simulates a full one player game, played by the just the computer, starting at a total score of 0 and playing turn after turn until it reaches 100 points.

Inputs: None.
Outputs: A turn-by-turn transcript of the computer playing Pig by itself until it reaches the goal of 100 points.

Example Problem 6 output:
- rolled a 6
- rolled a 1

Pigged out!
Turn score = 0
New total score = 0

- rolled a 1
Pigged out!
Turn score = 0
New total score = 0

- rolled a 5
...

- rolled a 4
Turn score = 23
New total score = 90

- rolled a 4
- rolled a 5
- rolled a 5

Turn score = 14
New total score = 104

Problem 7: Computer vs. Computer (4 marks)

The game of Pig was really meant for two players, so we'll make that happen now. First, we'll have both players controlled by the computer, and both using the same strategy as before.

Write a program that simulates two computer AIs playing Pig against each other. Name the players "Player One" and "Player Two", or other names of you choosing, to distinguish them.

Your program should randomly choose which player goes first. It should keep a total score for each player, and alternate turns between the computer players until one player ends its turn with a score of

100 or higher. Print the outcomes of each roll that occurred during each turn, and at the end of each turn, print a summary of the total scores to that point in the game.

Inputs: None.
Outputs: A turn-by-turn transcript of two computer players playing Pig against each other.

Example Problem 7 output:

```
Player One's score: 0
Player Two's score: 0

It's Player One's turn
- rolled a 1
Pigged out!
Total turn score = 0

Player One's score: 0
Player Two's score: 0

It's Player Two's turn
- rolled a 6
- rolled a 2
...

Player One's score: 85
Player Two's score: 88

It's Player One's turn
- rolled a 2
- rolled a 6
- rolled a 2
- rolled a 4
- rolled a 3

Total turn score = 17
Final score: 102 vs 88

Player One wins!
```

Problem 8: Player vs AI (5 marks)

Finally, it's time to create the interactive game of Pig where you get to play a full game against the computer. Replace one of the computer players from your previous program with an interactive prompt for a human player's input. After each roll during the human player's turn, ask whether the player wants to [r]oll or [h]old.

Print a full transcript of the game playing out, just as you did with the previous problem. Play a few games against your computer AI when you're done, just to make sure everything works.

Submission:

After you have completed this assignment, you should have written a total of eight Python programs, saved as .py files.

Please ensure that your files are named descriptively, with the problem number included, so that your TA can easily see which program is associated with each problem.

Use the University of Calgary Desire2Learn system (http://d2l.ucalgary.ca) to submit your assignment work online. Log in using your UofC eID and password, then find our course, cpsc 231 l01 and l02, in the list. Then navigate to *Assessments -> Dropbox Folders*, and find the folder for Assignment #1 here.

Collaboration

Discussing the assignment requirements with others is a reasonable thing to do, and an excellent way to learn. However, the work you hand-in must ultimately **be your work**. This is essential for you to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not your original work, but is represented as such, is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code that you hand-in that are not your original work. You can put the citation into comments in your program. For example, if you find and use code found on a web site, include a comment that says, for example:
   ```
   # the following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```
   Use the complete URL so that the marker can check the source.
2. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. However, you may still get a low grade if you submit code that is not primarily developed by yourself.
3. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code that it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you find yourself exchanging code by electronic means, writing code while sitting and discussing with a fellow student, typing what you see on another person's console, then you can be sure that "your" code is not substantially your own, and your sources must then be cited to avoid plagiarism.
4. **Collaborative coding is strictly prohibited.** Your assignment submission must be strictly your code. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing code itself, or modeling code after another student's algorithm. **You cannot use (even with citation) another student's code.**
5. We will be looking for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).

Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor to get help, than it is to plagiarize.

Credits:

The game of Pig was originally invented by John Scarne [1]. To the best of our knowledge, the use of the Pig dice game to teach computer science was pioneered at Gettysburg College. A few problems appearing in this assignment are derived from originals posed by Todd Neller [2]. A few problems were also derived from problems posted by Sonny Chen.

1. John Scarne. Scarne on Dice. Wilshire Book Co., 8th edition, 1992
2. Todd W. Neller, Clifton G. M. Presser, Ingrid Russell, and Zdravko Markov. Pedagogical possibilities for the dice game Pig. Journal of Computing Sciences in Colleges, 21(6):149–161, 2006

Grading:

There are total 25 marks for this assignment. If you have completed all the problems, you would get 25 marks.