CS 1510: Intro to Computing - Fall 2017
Assignment 5: A Game of War

Code AND Worksheet Due: Friday, October 13, 2017, by 11:59 p.m.

Introduction

Somewhere along the line as a kid (at least those of you who grew up in the US) you played the card game called War. It's a simple two-player game. You divide a deck (or multiple decks) of cards into two, and each player gets half the cards. Next, each player turns over the top card of their decks. If player 1 has the higher card, player 1 wins that round and takes both cards. Both the card totals are added to player 1's score. If player 2 has the higher card, player 2 wins that round and takes both cards. Both the card totals are added to player 2's score. If there is a tie, different things can happen, depending on the rules.

The game is often seen as a game of chance. Whichever player wins is random, similar to flipping a coin or throwing a dice randomly.

Original Program Specifications

Before starting your code, read through the program specification below and fill in the worksheet for each task/subtask you will need to perform

Your program (saved in a file called noWar.py) will allow a human user to play several rounds of War with the computer. Each round of the game will have the following structure:

- The program will ask the user if they want to exit, play, or show the score.
    - If the user chooses exit by typing "e" or "E", then the game should print a goodbye message.
    - If the user chooses show score by typing "s" or "S", then the game should display the number of rounds won for each player and the total score of cards won for each player.
    - If the user chooses play by typing "p" or "P", then the program should choose a card randomly for both the human player and computer player. The program should then print who won, followed by the cards chosen for each player.
    - **If the round results in a tie, the program should print that a tie occurred (and the specific cards that caused the tie). It should not add the value of the cards to each player's score or increment the number of wons or losses. Instead, it should throw out those cards, and the program should immediately choose another card for each player. This behavior should repeat until a player actually wins. At that point, increment that player's rounds won counter by one and add the total score of the cards for just the tie-break round to the winner's score.**
    - If the user inputs something other than the exit, play, or show letters, the program should detect the invalid entry and ask the user to make another choice.

For example, one sample game might look like this:

```
>>>
Welcome to the game of war.
Press e to Exit, p to Play, and s to Show score y
Incorrect input.
Press e to Exit, p to Play, and s to Show score s

Your round wins: 0
My round wins: 0
Your card score: 0
My card score: 0

Press e to Exit, p to Play, and s to Show score p
We tied.
Better go again...
You had: A
Computer had: A

I won.
You had: 7
Computer had: 10

Press e to Exit, p to Play, and s to Show score p
You won!
You had: Q
Computer had: A

Press e to Exit, p to Play, and s to Show score s

Your round wins: 1
My round wins: 1
Your card score: 13
My card score: 17

Press e to Exit, p to Play, and s to Show score exit
Incorrect input.
Press e to Exit, p to Play, and s to Show score e

Thanks for playing!
```

For this version of this game, you must program the computer to pick cards randomly. To do this you should put the following code at the top of your program:

```
import random
random.seed()
```

This is the code necessary to set up a random number generator. Each time you need to pick a random number you use:

```
picked=random.randint(1,13)
```

This will produce a value from 1 to 13 INCLUSIVE (notice this is different from the way range works). 1=Ace, 2=2, 3=3 ... 10=10, 11=Jack, 12=Queen, and 13=King.

Getting Started (Worksheet)

1. Do all the standard startup things. Create a new file called noWar.py. Put your comments in at the top, save it.
2. Now you need to break the problem down into parts. Read the description and identify the tasks/subtasks that need to be solved in the worksheet. For example, one task would be to prompt for valid user input.
3. Now address one subtask, getting user input. Do this in stages as well. Can you:
    a. Prompt for and get a choice (a string) from the user?
    b. Once you can do that, can you check for "legal" character responses from the user, and print an error message when an illegal response is given?

    Once you can do all that, be sure all the code you have works before moving on to the next subtask.

4. Remember, save the file and run it all the time! It will make debugging the program easier.

Notes and requirements:

Make sure that you save your programs in the correctly named files

1. Use meaningful variable names with the proper style (use_snake_casing)
2. Use meaningful constants and names where appropriate and use proper style
3. Every file containing python code that you submit should contain a header commen block containing three pieces of information as shown below:

```
"""
File: filename
Author: your-name
Description: one-line description of the file
"""
```

Points: 20 points for the code, 5 points for the worksheet.

Final Submission

Submit files **war.py** and **worksheet.doc** to eLearning.