CECS 277 – Lab 1 – Python Basics

Guessing Game

Have the computer generate a random number between 1 and 100, then prompt the user to guess a number between 1 and 100. Test that the user's input is an integer within the range, if it isn't, tell them that it is invalid and allow them to continue guessing. Compare the user's number to the computer's number, if the user's number is lower than the computer's, tell them that it is too low, if it is higher, tell them that it is too high. Repeat until the user guesses the correct value. Keep count of the number of guesses and display it when the user wins (exclude invalid entries).

Example Output (user input is in italics)**:**

```
- Guessing Game –
I'm thinking of a number.  Make a guess (1-100): 50
Too low!  Guess again (1-100): 75
Too low!  Guess again (1-100): -1
Invalid input – should be within range 1-100.
Guess again (1-100): a
Invalid input – should be an integer.
Guess again (1-100): 95
Too high! Guess again (1-100): 90
Too high! Guess again (1-100): 80
Too low!  Guess again (1-100): 85
Correct!  You got it in 6 tries.
```

Notes:

1. Place your name, date, and a brief description of the program in a comment block at the top of your program. Place brief comments in your program to describe sections of code (you should not have a comment describing every single line).
2. Your code should be placed in a main function.
3. Use the check_input module provided on Canvas to check the user's input for invalid values. Add the .py file to your project folder to use the functions. Examples using the module is provided in a reference document on Canvas.
4. Use the random module to generate your random numbers. Examples for generating random numbers is provided in a reference document on Canvas.
5. No need to create extra functions or add lists to your code, you'll only need the main function with a while loop and some if statements.
6. Please read through the Coding Standards reference document on Canvas for guidelines on how to name your variables and to format your program.
7. Thoroughly test your program before submitting/demoing.
    a. Make sure the program only creates the random number once at the beginning of the program (not repeatedly changing the number as the user is guessing).
    b. Make sure the program accurately responds to high/low/correct guesses.
    c. Invalid guesses should not output 'Too high' or 'Too low'.
    d. Make sure the program accurately reports the number of guesses the user made. Count all valid guesses, including the final guess. Do not count any invalid guesses (ex. 0, negative numbers, or values greater than 100, or any strings).

Guessing Game Rubric – Time estimate: 2 hours

Guessing Game **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Random Number: 1. Generates a random number in the range 1-100 (inclusive). 2. Random number is only generated once at the beginning of the program.					
Checks: 1. Uses if statements to check if the user's guess is too high or too low. 2. Uses a while loop that repeats until the user's guess is correct.					
Validity: 1. Always checks that the user's guess is a valid integer within the range 1-100 (inclusive). 2. Displays an error message if the user's guess is invalid and does not increment counter.					
Output: 1. Prompts user to enter guess. 2. Displays whether user's guess is too high or too low. 3. Displays whether the user guessed the correct number. 4. Displays the number of tries it took to guess the number (does not count invalid guesses). 5. Does not display conflicting or erroneous messages (ex. Too high and Invalid).					
Code Formatting: 1. Code is in a main function. 2. Correct spacing. 3. Meaningful variable names. 4. No global variables. 5. Correctly documented.					