CSCI 1300 Computer Science I: Starting Programming
Homework 6

Objectives:

- to write a computer program to conduct an "experiment" that could be easily done in the physical world; in particular, to simulate the activity of throwing a six-sided die
- to explore events involving chance, via the use of random numbers.
- to design and implement solutions to problems involving loop patterns, including nested loop structures.
- to understand the concept of modular programming. Identify different parts of the game and translate this into different modules/functions of your program

Jeopardy Dice

This project's objective is to create a Jeopardy Dice game in which one player (the user) competes against the computer to reach 100 points. You will turn in one file called jeopardy_dice.py. This program must contain the following information in comments at the top of the file:

- Your Name
- Your TA's Name
- The homework's name
- The name of your partner (if any)
- A brief description of the program and what it does.

Game Mechanics

- There are two players: the user and the computer, who alternate turns until one of them reaches 100 points or higher.
- The user is always the first player.
- If any player reaches 100 points or more at the end of their turn, the game ends immediately and the other player does not get another turn.
- One turn consists of the following: the player rolls a 6-sided die.
    - If they roll a 1, they score 1 point and their turn is over. Even if a player has accumulated points from previous rolls, if they roll a 1, their score for that turn will be 1 point and their turn is over.
    - If they roll any other number, the roll score is added to the turn total.
    - Then, they have the option to continue rolling or to hold. There is no restriction on how many times a player can roll in a turn.
- Holding: if a player holds, they score their current turn total and their turn ends. If, for example, a player rolled a 3, 4, and 2, then decides to hold, they score 9 points.
- If the player is the user, they are given the option as to whether they would like to continue rolling or hold after each time they roll the die and do not get a 1.
- If the player is the computer, they will always continue rolling until their turn total reaches the value 10 or higher.

Game parameters

One could create a game where the number of points needed to win would be 50 and where the computer player holds at 7. Or a game where the number of points needed to win would be 150 and where the computer player holds at 20. To accommodate for different variations in the parameters that define the game, create variables to hold these values:

1. You will create a variable GAME_END_POINTS and set it to the value 100 at the beginning of your main() function. You should not change the value of this variable during the rest of the program.
2. You will create a variable COMPUTER_HOLD and set it to the value 10 at the beginning of your main() function. You should not change the value of this variable during the rest of the program.
3. You will create a variable *is_user_turn* and set it to the value *True* at the beginning of your main() function, since the user is always the first player. This variable will have its value reassigned to *False* or *True* every turn, in the function **get_next_player** (see below)

Functions
Your program *must* include the following functions:

- **print_current_player**: this function takes one parameter *is_user_turn* (a Boolean), and does not return any value. The function prints a statement indicating whose turn it is. If it is the computer's turn, the function prints It is now computer's turn. If it is the user's turn, the function prints It is now human's turn. See the example output below.
- **roll_die**: this function takes no parameters, rolls a 6-sided die and **returns** the value rolled.
- **take_turn**: this function is the powerhouse of your program. The function takes two parameters: *is_user_turn* (a Boolean) and COMPUTER_HOLD. The function **returns** the turn total for the current player. The function should call the **roll_die** function and return 1 if the player rolls a 1. Otherwise, it should print out a message informing the user of their current turn total, and then prompt the user for whether or not they want to roll again or hold. If the current player is the computer, determine based off of the value COMPUTER_HOLD whether or not to roll again. If the player chooses to roll, the function should call the **roll_die** function again and repeat the same actions until the player chooses not to roll. If the player is not rolling again, this is considered to be holding and this function should return the current turn total. If the roll of the die is 1, the function should return the value 1 as the turn total.
- **report_points**: this function takes two parameters, the user's point total and the computer's point total and prints a report of how many points each player has. See the example output below.
- **get_next_player**: this function takes one parameter *is_user_turn* (a Boolean), and **returns** a Boolean which is the complement of the value passed (if the function is called with *True* as an argument, the function should return *False*, and vice versa).

Your program *may additionally include* any other functions that are appropriate. Remember, the code block inside main() should read like a table of contents and should not include the main computational work of the program.

You *must* use a main function from inside an `if __name__ == '__main__':` statement, as shown in the last section of chapter 7.7 in your textbook: https://runestone.stage.csel.io/thinkcspy1200/Functions/mainfunction.html.
Doing this is **required**.

Example:

```
def main():
    print('This is the main function')

main()
```

Becomes:

```
def main():
print('This is the main function')

if __name__ == '__main__':
    main()
```

This allows your program to either be imported as a module (like random) or run as a script.

Global Variables

A global variable is a variable that is in scope for the entire program. See the example below:

```
MY_NUMBER = 7 # this is a global variable

def add_nums(x):
   return MY_NUMBER + x

def print_num():
   print('This is my number:', MY_NUMBER)

if __name__ == '__main__':
   print('My number plus 5 is', add_nums(5))
   print_num()
```

The use of global variables is not considered a good programming practice. No global variables should be used in your solution. Our advice is to create variables in your main() function and pass their values to functions as needed.

Input and Output

You may assume that the user will only enter valid input. Notice that the program only prompts the user if they want to roll again if it is the user's turn, and that the prompt is case insensitive (the user may type either "y" or "Y", "n" or "N").

Below are two examples of program output. User input is shown **in bold**. Your program should **reproduce** the output shown below (excluding differences due to random values).

Example Output 1

```
Welcome to Jeopardy Dice!

It is now human's turn

You rolled a 6
Your turn total is 6
Do you want to roll again (Y/N)? Y
You rolled a 3
Your turn total is 9
Do you want to roll again (Y/N)? y
You rolled a 3
Your turn total is 12
Do you want to roll again (Y/N)? N
computer: 0
human: 12

It is now computer's turn

You rolled a 5
Your turn total is 5
You rolled a 3
Your turn total is 8
You rolled a 6
Your turn total is 14
computer: 14
human: 12

It is now human's turn

You rolled a 6
Your turn total is 6
Do you want to roll again (Y/N)? y
You rolled a 5
Your turn total is 11
Do you want to roll again (Y/N)? Y
You rolled a 3
Your turn total is 14
Do you want to roll again (Y/N)? y
You rolled a 2
Your turn total is 16
Do you want to roll again (Y/N)? y
You rolled a 3
Your turn total is 19
Do you want to roll again (Y/N)? y
You rolled a 6
Your turn total is 25
Do you want to roll again (Y/N)? y
You rolled a 1
computer: 14
human: 13

It is now computer's turn

You rolled a 4
Your turn total is 4
You rolled a 1
computer: 15
human: 13

It is now human's turn

[...] (output truncated)
```

Example Output 2

```
Welcome to Jeopardy Dice!

It is now human's turn

You rolled a 1
computer: 0
human: 1

It is now computer's turn

You rolled a 1
computer: 1
human: 1

It is now human's turn

You rolled a 5
Your turn total is 5
Do you want to roll again (Y/N)? n
computer: 1
human: 6

It is now computer's turn

You rolled a 2
Your turn total is 2
You rolled a 5
Your turn total is 7
You rolled a 5
Your turn total is 12
computer: 13
human: 6

It is now human's turn

You rolled a 6
Your turn total is 6
Do you want to roll again (Y/N)? n
computer: 13
human: 12

It is now computer's turn

[...] (output truncated)

It is now human's turn

You rolled a 1
computer: 92
human: 57

It is now computer's turn

You rolled a 4
Your turn total is 4
You rolled a 5
Your turn total is 9
You rolled a 2
Your turn total is 11
computer: 103
```

```
It is now computer's turn
You rolled a 3
Your turn total is 3
You rolled a 5
Your turn total is 8
You rolled a 3
Your turn total is 11
computer: 79
human: 89

It is now human's turn

You rolled a 5
Your turn total is 5
Do you want to roll again (Y/N)? y
You rolled a 2
Your turn total is 7
Do you want to roll again (Y/N)? y
You rolled a 6
Your turn total is 13
Do you want to roll again (Y/N)? y
You rolled a 3
Your turn total is 16
Do you want to roll again (Y/N)? n
computer: 79
human: 105

Congratulations! human won this round of
jeopardy dice!
```

```
human: 57

Congratulations! computer won this round of
jeopardy dice!
```

Testing

It is your responsibility to test your program. The random element of this program makes it so that you should not expect to exactly reproduce the example outputs. However, aside from the specific numbers, your output should exactly match these examples.

Style and Comments (15 points)

Comments (5 points):

- Your code should be well commented. Use comments to explain what you are doing, especially if you have a complex section of code. These comments are intended to help other developers understand how your code works. These comments should begin with pound signs (#).
- Each function definition should have a docstring. This comment does two things: it describes the purpose of the function and it describes how to use it. You should not describe the mechanics of how the function works, but rather what it does. E.g. "This function counts the number of even numbers in a list" (good) versus "this function uses a for loop to loop through a list and increment a counter each time it finds an even number" (bad). Docstrings always go directly after the definition line.
    - Example (notice the usage of """ at the beginning and end of the comment):

    -
      ```
      def convertCF(celsius):
      """Convert degrees in celsius to farenheit.

      Args:
      celsius (int) -- the temperature in celsius

      Returns:
      int - the temperature in farenheit
      """
      (your code goes here)
      ```

Naming (2 points)

- Your variable names should be meaningful and concise
- Your variable names should be formatted like: turn_total (words start with lowercase letters and are separated with underscores)
- Function names should be formatted the same way as your variables: roll_die()(words start with lowercase letters and are separated with underscores)

Proper Variable Usage (2 points)

- No use of global variables
- Your program should have the 3 variables described above—COMPUTER_HOLD, GAME_END_POINTS, and is_user_turn

Main (6 points)

- You must use a main that runs from an `if __name__ == '__main__':` block.
- The code in main() may contain: function calls, control statements (if/elif/else), and print statements. It may contain a **maximum of 1 while loop and 5 print statements**.