NYU Tandon School of Engineering

CS-UY 1114 Fall 2022

Homework 04

Due: 11:59pm, Friday, October 14, 2022

Submission instructions

1. You should submit your homework on **Gradescope**.
2. For this assignment you should turn in 5 separate `.py` files named according to the following pattern: `hw4_q1.py`, `hw4_q2.py`, etc.
3. Each Python file you submit should contain a header comment block as follows:

```
"""
Author: [Your name here]
Assignment / Part: HW4 - Q1 (depending on the file name)
Date due: 2022-10-14, 11:59pm
I pledge that I have completed this assignment without
collaborating with anyone else, in conformance with the
NYU School of Engineering Policies and Procedures on
Academic Misconduct.
"""
```

No late submissions will be accepted.

REMINDER: *Do not use any Python structures that we have not learned in class.*

The use of `eval()` and `break` are not permitted in this and in future homework assignments.

For this specific assignment, you may use everything we have learned up to, **and including**, selection statements `for`- and `while`-loops. Please reach out to us if you're at all unsure about any instruction or whether a Python structure is or is not allowed.

Do **not** use, for example, user-defined functions (*except* for `main()` if your instructor has covered it during lecture), strings and string methods, file i/o, exception handling, dictionaries, lists, tuples, and/or object-oriented programming.

Problems

1. **(Odd) Baby Steps (`hw4_q1.py`)**
2. **Body Like An Hourglass (`hw4_q2.py`)**
3. **Count Your Steps (`hw4_q3.py`)**
4. **Mod Culture (`hw4_q4.py`)**

5. **Must Be Funny In A Rich Man's World (`hw4_q5.py`)**

Problem 1: *(Odd) Baby Steps*

Write a program that reads a positive integer (say, *n*), and prints the first *n* odd numbers. Write two versions in the file, one using a `for`-loop, and one using a `while`-loop. For example, one execution *could* look like this:

```
Please enter a positive integer: 5
Executing while-loop...
1
3
5
7
9

Executing for-loop...
1
3
5
7
9
```

Both of these implementations must be included in the same file.

Problem 2: *Like An Hourglass*

Ask the user to input a positive integer `n`, and print a textual image of an hourglass made of `2n` lines composed of asterisks. For example, if `n = 4`, the program should print the representation shown below:

```
*******
 *****
  ***
   *
  ***
 *****
*******
```

Problem 3: *Count Your Steps*

Write a program that asks the user to input a positive integer, and print a triangle of numbers aligned to the right, where the first line contains the number `1`. The second line contains the numbers `2`, `1`. The third line contains `3`, `2`, `1`. And so on. For example, if `n = 5`, the program should print:

```
    1
   21
  321
```

```
 4321
54321
```

You may **not** use the `end` parameter of the `print()` function in this problem. Note also that there is **no** empty line before the row containing only `1`.

Problem 4: *Mod Culture*

Ask the user to input a positive integer, say, `n`, and print all the numbers from `1` to `n` that have more even digits than odd digits. For example, the number `134` has two odd digits (`1` and `3`) and one even digit (`4`), therefore it should **not** be printed.

Printing should occur **all on one line, separated by spaces**. For example, if `n = 30`, the program should print:

```
2 4 6 8 20 22 24 26 28
```

Problem 5: *Must be funny in the rich man's world*

Let's say that you are tasked with writing the final money counter program for a Monopoly-like video game. If you don't know what **Monopoly** is, don't worry; all you need to know is that at the end of each game of Monopoly game, every player will have a certain amount of money, and the person with the most amount of money wins (a **terrible concept**, if you ask me. But I digress).

So, your task is to write a program called that:

1. Asks the user for an integer value which represents the number of players who played that round. The minimum number of players that can play a game of Monopoly is **2**, and the maximum number of players is (apparently) **8**. If the user enters any number under 2 or over 8, keep asking for that input until they enter a valid number of players.
2. Once they do so, each player will enter the values of each of their properties/assets.
3. Once a player enters all the values, the game will print out the sum.
4. The program repeats steps 2 and 3 until all players have been accounted for.
5. Print out at the very end which player had the most money.

Here's a sample execution:

```
Enter a valid number of players: 3
Enter the value of a property/asset, or DONE to finish: 100
Enter the value of a property/asset, or DONE to finish: 34
Enter the value of a property/asset, or DONE to finish: 54
Enter the value of a property/asset, or DONE to finish: DONE
Player 1 has 188.0 dollars.
Enter the value of a property/asset, or DONE to finish: 10000
Enter the value of a property/asset, or DONE to finish: DONE
Player 2 has 10000.0 dollars.
Enter the value of a property/asset, or DONE to finish: 43.34
```

```
Enter the value of a property/asset, or DONE to finish: DONE
Player 3 has 43.34 dollars.
2 wins with 10000.0 dollars!
```

```
Enter a valid number of players: 1
Enter a valid number of players: 10
Enter a valid number of players: 3
Enter the value of a property/asset, or DONE to finish: 23.54
Enter the value of a property/asset, or DONE to finish: 34.667
Enter the value of a property/asset, or DONE to finish: 123.3
Enter the value of a property/asset, or DONE to finish: DONE
Player 1 has 181.51 dollars.
Enter the value of a property/asset, or DONE to finish: 1969.00
Enter the value of a property/asset, or DONE to finish: DONE
Player 2 has 1969.0 dollars.
Enter the value of a property/asset, or DONE to finish: 12.0
Enter the value of a property/asset, or DONE to finish: 0.05
Enter the value of a property/asset, or DONE to finish: DONE
Player 3 has 12.05 dollars.
2 wins with 1969.0 dollars!
```

Note that:

- For this problem, you must round dollar values appropriately.
- You may assume that, for the value of a property/asset, the user will only enter either a positive numerical value or the string `"DONE"`.