Homework 2

Functions and Lists

Given January 29th – Due February 10th

Exercise 1. Automate the boring stuff, page 77, The Collatz Sequence.
Write a function named collatz() that has one parameter named number. If number is even, then collatz() should print number // 2 and return this value. If number is odd, then collatz() should print and return 3 * number + 1. Then write a program that lets the user type in an integer and that keeps calling collatz() on that number until the function returns the value 1. (Amazingly enough, this sequence actually works for any integer—sooner or later, using this sequence, you'll arrive at 1! Even mathematicians aren't sure why. Your program is exploring what's called the Collatz sequence, sometimes called "the simplest impossible math problem.") Remember to convert the return value from input() to an integer with the int() function; otherwise, it will be a string value. Hint: An integer number is even if number % 2 == 0, and it's odd if number % 2 == 1. The output of this program could look something like this:

Enter number:
3
10
5
16
8
4
2
1

Exercise 2. Automate the boring stuff, page 77, Input Validation.
Add try and except statements to the previous project to detect whether the user types in a noninteger string. Normally, the int() function will raise a ValueError error if it is passed a noninteger string, as in int('puppy'). In the except clause, print a message to the user saying they must enter an integer.

Exercise 3. Automate the boring stuff, page 102, Comma code.
Say you have a list value like this:
spam = ['apples', 'bananas', 'tofu', 'cats']
Write a function that takes a list value as an argument and returns a string with all the items separated by a comma and a space, with '*and*' inserted before

the last item. For example, passing the previous spam list to the function would return *'apples, bananas, tofu, and cats'*. But your function should be able to work with any list value passed to it.

Exercise 4. Automate the boring stuff, page 103, Character Picture Grid.
Say you have a list of lists where each value in the inner lists is a one-character string, like this:

```
grid = [['.', '.', '.', '.', '.', '.'],
        ['.', '0', '0', '.', '.', '.'],
        ['0', '0', '0', '0', '.', '.'],
        ['0', '0', '0', '0', '0', '.'],
        ['.', '0', '0', '0', '0', '0'],
        ['0', '0', '0', '0', '0', '.'],
        ['0', '0', '0', '0', '.', '.'],
        ['.', '0', '0', '.', '.', '.'],
        ['.', '.', '.', '.', '.', '.']]
```

You can think of grid[x][y] as being the character at the x- and y-coordinates of a "picture" drawn with text characters. The (0, 0) origin will be in the upper-left corner, the x-coordinates increase going right, and the y-coordinates increase going down. Copy the previous grid value, and write code that uses it to print the image.

```
..00.00..
.0000000.
.0000000.
..00000..
...000...
....0....
```

Hint: You will need to use a loop in a loop in order to print *grid[0][0]*, then *grid[1][0],* then *grid[2][0]*, and so on, up to *grid[8][0].* This will finish the first row, so then print a newline. Then your program should print *grid[0][1],* then *grid[1][1]*, then *grid[2][1],* and so on. The last thing your program will print is *grid[8][5].* Also, remember to pass the *end* keyword argument to *print()* if you don't want a newline printed automatically after each *print()* call.

Exercise 5. Guess the number

1. Develop a "Guess the number" game. Write a program that comes up with a random number and the player has to guess it. The program output can be like this:

```
I am thinking of a number between 1 and 20.
Take a guess.
10
Your guess is too low.
```

```
Take a guess.
15
Your guess is too low.
Take a guess.
17
Your guess is too high.
Take a guess.
16
Good job! You guessed my number in 4 guesses!
```

2. Complicate it by having random lower and upper bounds. Make sure your lower bound is not higher than you upper bound!

3. Now, instead of user input make the code guess it automatically. Make sure that for the automatic guesses you don't use the number to guess.

Submit your code files as *hm2_name_surname_ex_num.py*, where *num* is the exercise number 1, 2, 3 or 4. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

"""

Homework 2, Exercise 1 (or 2...)
Name
Date
Description of your program.

"""