CSCI 141 - Spring 2019
Lab 7: Strings
Due Date: See Canvas

Introduction

In this lab you'll gain practice using and manipulating strings. In lecture you've seen how to iterate over the characters in a string, access individual characters by index, and how to slice out substrings. We also touched on a few of the multitude of methods that can be called on string objects, including `upper`, `lower`, `find`, and `replace`.

Suppose you are an employee at a software company called Lotter.io. You have been tasked with writing a Python program that will simulate the drawing of a winning lottery pick, then prompt a user to guess the lottery pick, and inform the user if their pick is the winning pick.

Your company's secret sauce that will guarantee its explosive growth into a billion dollar startup is that the lottery doesn't just involve guessing numbers: Instead, this lottery involves both words *and* digits. A lottery pick is made up of one of three words (*monkey*, *dragon*, or *snake*), and one of three digits (1, 2 or 3), arranged either with the word first or the number first. For example, *monkey*1, and 2*dragon* are both possible winning picks. The user only wins if the correct word and the correct digit are guessed in the correct order.

The program consists of three main tasks:

1. Generate a winning pick.
2. Get a valid pick from the user.
3. Check the user's pick against the winning pick and tell them how they did.

Three sample invocations are shown in Figure 1.

1 Generate a winning pick

Random pick generation is to be implemented in the `generate_winner` function. The function header, specification, and some of its implementation have been written for you.

Easter Egg

Our lottery game will have a secret *easter egg*, a hidden feature that's only accessible to those who have read the code, or users who somehow stumble upon it. Our easter egg behavior is as follows.

```
Welcome, and thanks for playing Lotter.io!
Let's see if you've won. Today's word choices are
monkey, dragon, and snake; the digit choices were
1, 2, and 3. The winning pick is a word and a digit,
in either order. Press enter to begin:

Enter your pick: dog2
Warning: your guess does not contain one of the words.
Enter your pick: snake2
You guessed the correct word but not the correct number.
>>>
```

```
Welcome, and thanks for playing Lotter.io!
Let's see if you've won. Today's word choices are
monkey, dragon, and snake; the digit choices were
1, 2, and 3. The winning pick is a word and a digit,
in either order. Press enter to begin:

Enter your pick: snake2
You won a bronze scuplture of a zebra!! Great job!
```

```
Welcome, and thanks for playing Lotter.io!
Let's see if you've won. Today's word choices are
monkey, dragon, and snake; the digit choices were
1, 2, and 3. The winning pick is a word and a digit,
in either order. Press enter to begin:

Enter your pick: dragon1
You guessed neither the correct word nor the correct number.
```

Figure 1: Three sample invocations

The welcome message tells the user to press enter to continue; however, if the user enters something that looks like a valid pick and then presses enter, `generate_winner` will simply use their input as the correct pick.

The functionality for this easter egg has been implemented for you: the `begin_input` variable captures what the user enters, and is passed into the `generate_winner` function, which first checks to see if it should return the cheat pick. This "cheater mode" may help you more thoroughly test your program's behavior by giving you the option to choose the winning pick without any randomness involved. It will also make grading easier for us, for the same reasons, so be sure not to modify the code that implements this feature.

Your task is to write the code after this part of the function that generates and returns a truly random winning pick. Pseudocode for your portion of the function is provided as comments in the skeleton code.

2 Get a valid pick from the user.

Implement the `get_valid_guess` function to prompt the user, repeatedly if necessary, until they enter a guess that contains one of the words and begins or ends with one of the numbers. Notice that these criteria do make it possible to enter an invalid pick (e.g., `dragoncat3` contains one of the words and ends with one of the numbers. This is fine: cases such as these will be handled later in the program. The function header, specification, and pseudocode for this part are provided in the skeleton code.

3 Check the user's pick

Now, we need to tell the user whether any or all aspects of their pick were correct. Here's what the program should do once it has determined the winning pick and the user's pick. Your messages do not need to match mine exactly, but should convey the same content. Feel free to decide on your own prize for guessing the winning pick.

- If the user inputs a pick that exactly matches the winning pick character for character, the program outputs a message telling the user they've won.
- If the user's word and number both match the winning pick, but their pick doesn't match the winning pick exactly, print *You guessed the correct number and word in the wrong order!*
- If the number matches but the word doesn't, print *You guessed the correct number but not the correct word.*
- If the word matches but the digit isn't correct: print *You guessed the correct word but not the correct number.*
- If neither the word nor number are correct, print *You guessed neither the correct word nor the correct number.*

The `main` method includes calls to the two functions you completed in the prior tasks, followed by pseudocode for this step. You may want to, but are not required to, create a separate function to accomplish this task.

Getting Started

Download the skeleton code `lottery.py` from the course webpage. Open up and read through the skeleton code, making sure that you understand the provided code and function specifications. We recommend completing the three tasks in the order described above.

You may find the `in` and `not in` operators useful on strings, as well as indexing individual characters of a string and slicing substrings. Review the lecture slides for details on the syntax for these operators. Don't forget that when a program's logic gets complicated, it can be helpful to define boolean variables to keep the conditions of `if/elif` statements short and easy to read.

Submission

Submit your completed program `lottery.py` file via Canvas.

Rubric

Submitted a file called `lottery.py`	1
Comment at the top of program specifies author, date, and description	1
Easter egg code is unmodified	1
Random pick is generated	5
User is prompted again if guess doesn't contain one of the words	5
User is prompted again if guess doesn't begin or end with one of the numbers	5
Message printed for correct pick	5
Message printed for correct word and number but incorrect order	5
Message printed for correct word but not number, and vice versa	5
Message printed for neither correct	5
Coding style (commenting, variable names, etc.)	2
Total	40 points