Laboratory Exercise: More Python

Overview

This assignment asks you to write two programs in Python. As before, you can do this assignment either in the computer lab or on your own computer, if you have one.

Program #1: Rock-Paper-Scissors

This is an old child's game. Suppose Alice and Bob play together. Each makes a hand in the shape of a rock (a fist), a sheet of paper (a flat hand), or a pair of scissors (index and middle fingers extended, and the rest closed). If both make the same shape, the game is a draw. If not, the winner is determined as follows:

If Alice makes the shape of a rock, and Bob the shape of a sheet of paper, Bob wins ("paper wraps rock").

If Alice makes the shape of a rock, and Bob the shape of a pair of scissors, Alice wins ("rock smashes scissors").

If Alice makes the shape of a sheet of paper, and Bob the shape of a pair of scissors, Bob winas ("scissors cut paper").

Your job: write a program to let you play this game against the computer. Your program's output is to look like that in the "Sample Run" below. Don't worry if you can't duplicate the "rock", "paper", and "scissors" words n exactly that order; they are generated randomly. But aside from that, and the name (use your own name, please), everything else should be the same.

There are lots of ways to do this, so we won't give you step-by-step instructions. But here are some suggestions.

1. When you write this, get one round right, then use a "while" loop to enclose the part of the program that plays each round.
2. To determine whether you (or the computer) make the shape of a rock, a sheet of paper, or a pair of scissors, generate a random number between 0 and 2. Use "randrange(3)" to do this, as was done in the sample program shown in class; it's available on the class web site. Then treat 0 as "rock", 1 as "paper", and 2 as "scissors", supplying those names only when printing. Don't forget the "import random" line!
3. You will need two different variables to play: one to hold your shape (or number corresponding to the shape), and another to hold the computer's.
4. You can use the while loop, and the request to play again, in the die throwing program shown in class.

Don't forget to greet the player, and say goodbye.

Sample Run

In what follows, what I typed is black; what the computer typed is blue (just like in the IDLE GUI).

```
What is your name? Matt
Hello, Matt
Matt says paper, computer says scissors:  computer wins!
Do you want to play again (yes/no)? yes
Matt says rock, computer says scissors:  Matt wins!
Do you want to play again (yes/no)? yes
Matt says scissors, computer says rock:  computer wins!
Do you want to play again (yes/no)? yes
Matt says paper, computer says paper:  Draw!
Do you want to play again (yes/no)? what
I'm sorry, I only understand yes or no.
Do you want to play again (yes/no)? yes
Matt says rock, computer says scissors:  Matt wins!
Do you want to play again (yes/no)? no
Thank you for playing, Matt
```

Program #2. Is It an Integer?

In class, you saw that trying to convert a string that contained non-digits to an integer using the conversion function int() caused your program to stop running (the technical term is “crash”). We’re going to tackle this in two parts. This week, you’ll write a program that will tell you whether you typed an integer or not, without crashing.

First, what is an integer? An integer is a string of (decimal) digits, specifically “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, and “9”, with possibly a leading “+” or “–” sign. There may be white space (blanks or tab characters) both before and after it. So, determining if an input string is an integer is the same as stripping off leading and trailing blanks, seeing if the first character is a “+” or a “–” sign or a digit, and the rest are digits. That’s the outline of your program.

Here it is, with a bit more structure:

1. Prompt the user for a string and read it in.
1. Remove leading and trailing blanks; use the method “strip()” for this.
2. Now, check the length of the remaining string. Use the function “len()” for this. If it is 0, then the user just typed a series of blanks and tabs, not an integer.
3. Next, check the first character of the remaning strng to see if it is a “+” or “-”. or a digit. If not, then the input is not a string.
4. Otherwise, check each of the remaining characters to see if they are all digits. If not, then the input is not a string. If so, then the input is a string.
5. Announce the result. If it is an integer, use the converter “int()” to convert the input to an integer.

Again, thee are many ways to do this program, so please be creative! But your output should match the output in the Sample Run below exactly.

Sample Runs

Because your program is to check only one number at a time, we ran ours several times to get the following output. Each time we ran it, the “RESTART” line, and the prompt beneath it, appeared.

Here, what the program output is in blue, what we typed is in black, and the portions from the IDLE GUI are in red. You don’t need to duplicate the red parts.

```
Type your input: 27
The integer you typed is 27
>>> ================================ RESTART ================================
>>>
Type your input:   345
The integer you typed is 345
>>> ================================ RESTART ================================
>>>
Type your input: -345
The integer you typed is -345
>>> ================================ RESTART ================================
>>>
Type your input: - 345
You didn't type an integer
>>>
```

Turning This In

To hand everything in, create a ZIP file named lab8.zip containing the three python files. In the labs, you can use the FilZip utility to create the ZIP file. Go to Start, then to All Programs, then to Utilities, then to FilZip. In FilZip, go to File, then to New Archive, and create lab8.zip on the Desktop. Click the Add (+) button and select the files. Note that you must click two Add buttons before they are added to the archive. Save your zip file in MySpace, and also submit it to MyUCDavis.