CSCA20 - Lab 5

Functions

Learning Objectives

This lab focuses on using functions to break problems down into smaller, more easily managed chunks, and learning to separate algorithmic planning from detailed implementation.

Prelab

The starter code has a few functions that you will need to implement. You don't need to show up to the lab with them completed, but you should at least have an idea of what they will do, and have written a simple menu that asks the user for their info and know which function(s) you will need to call.

Demonstration & Evaluation

Successfully completing this lab will demonstrate competency in User Input/Output, Variables, Loops and Functions. There will also be an opportunity to demonstrate mastery in user Input/Output and Variables. You do not need to complete the mastery portion to demonstrate competency in the other areas.

The Scenario

The Umbilical Territorial Stem-cell Collective (UTSC) needs help with their booking system. They have someone to handle the actual database part, but what they really need is a menu that will ask the users for their information, and make sure that it's in the format that they can use in their records. Lots of donors are mis-formatting their info, and it's causing problems with the system.

Functions

The starter code has some function definitions already started for you. In order to complete this task (for competency) you will need to implement the first 2 functions, and then use them in your code.

Matching a Template Character

In this scenario, we want to check whether a string matches a specific template. So for example, we know that phone numbers should be entered as a series of 9 numbers with brackets and a dash in the correct place, but we don't care about what the numbers actually are. So we use a template to say that `#` can represent any number, and then the template for a phone number would be `(###)###-####`. In this system, we use `#` to represent any number and `&` to represent any letter.

Matching a Template String

Once we have the function to match single characters, matching entire strings of those characters should be relatively easy. In fact, if you wanted, you should be able to write your entire `matches_string` function without having written your `matches_character` function. The goal here is to think about breaking the ideas down and only thinking about one thing at a time. So having all the details of matching characters and templates dealt with in `matches_character` should mean that in `matches_string`, you only have to worry about looping over the string(s).

Menu

You'll need to build a menu that prompts the user for their phone number, donor ID, and the booking time, and then makes sure the format is correct (and informs them if it isn't).

Mastery

You can demonstrate mastery of user input/output by building a robust menu that takes into account all sorts of bad input. For competency, we're only worrying about the format of the input, but for mastery we need to make sure that users can't do things like trying to book for 97:73QM (which matches the time format, but is clearly not a valid booking time). Also, the donor ID needs to match their name. That is, the first 2 letters of the donor ID must be the first 2 letters of the user's family name. Finally, the menu should actually be user friendly and give good quality feedback when the user enters bad info rather than just crashing or exiting out to the main menu.

You can also demonstrate mastery in variables on this lab by building a more robust template matching system where `#` represents any sequence of numbers and `&` represents any sequence of letters. This is more work than just matching single characters, but is do-able, if you break up the work appropriately.

Hints

Here are a few hints that might help you with this assignment:

- Work on one function at a time. Don't waste mental resources thinking about how you're going to use the character matching function while you're implementing it, and likewise, when you're implementing it, just trust that the function itself will work

- Test each function before you move on. It can get really complicated if you have a bug in your function that you don't realize until it's being called by another function.

- Document your functions thoroughly, and read the provided documentation. This goes back to separating algorithm from implementation: focus on the algorithm with the documentation, then you can worry about implementation separately.

- There are a few assumptions you can make that will simplify your life. For example: Since the documentation for `matches_string` says the strings must be of the same length, inside the function you can just take that for granted. Then worrying about different length strings becomes a separate (and easily solved) problem.

Extra Practice

This is a pretty full lab, particularly if you're trying to complete the mastery portion. But you can always build on your own code. Here are some ideas:

- Adding extra templates is fairly easy, but what about adding extra template types? For example, what about adding a a symbol that works for only upper case letters? Or what about one that works for anything **except** spaces?

- If you like the fun of the templates part, you can learn more about these sort of tools: we call them **regular expressions**, and they can actually be very powerful. We will see these more later in the term, but you can play around with them now: `https://www.w3schools.com/python/python_regex.asp`