CS 120 (Spring 22): Introduction to Computer Programming II

Short Project #1

due at 5pm, Thu 20 Jan 2022

1 Overview

In this assignment, you will write several short Python programs, each doing a relatively small task.

Precision
Follow the directions given *exactly:* your code will be graded automatically, so any deviation from the program spec can result in a significant loss of credit. If you are unsure about something, ask for clarification in Office Hours or Discord.

Style
Please pay attention to the programming style guidelines for this class. For this assignment you will be notified of style violations but not penalized for them; style violations will be penalized in subsequent assignments.

Submitting Your Solution
This assignment requires that you submit multiple files. Because of the the way that the auto-grader script works, in order to get full credit for your work you have to submit **all** of these files each time that you want to resubmit a solution to either problem.

Special Note
Because we're having you write a whole lot of little programs, a `main()` function will not be required for the Short Project 1. But starting with Long Project 1, they will be required!

2 `add_two_ints.py`

Write a program, in a file named `add_two_ints.py` , which calls `input()` twice (not giving the user any prompt). Convert each one to an integer, and then print out their sum.

You may assume that the input will be two integers (one per line); you don't have to do any error checking.

3 `call_imported_funcs.py`

Write a program, in a file named `call_imported_funcs.py` . Your program must import the file `short1_thing` , which I will provide. You will be calling three functions from inside that file: `foo(), bar(), baz()` .

First, read a line of input from the user. Do not modify it in any way; just read it. Save it for later; you're going to use it multiple times.

Second, read another line of input, and convert it to an integer. Save it away as well.

ALERT: The steps below are different than what was originally published in this spec, because the spec did not match the released testcases and autograder.

Third, call the function `foo()` (which you imported from the file). Pass it both of the two values you saved and print out the value that the function returns, and save the value.

Fourth, call the function `bar()`, and pass it only the second thing that you read. Print out what it returns; you do not need to save it.

Fifth, call the function `baz()`, and pass it both the first and second things that you read. Print and save its return value.

Finally, take the two return values that you saved, and pass them to `foo()` - the return value from `foo()` the first time, and then the return value from `baz()`. Print out what `foo()` returns.

NOTE 1:
The fuctions `foo(),bar(),baz()` may or may not print out something; don't worry about that. You should **only** print out what I've told you to print out, above.

NOTE 2:
Do **not** upload your own copy of `short1_thing.py` to GradeScope; the autograder will provide it for you.

NOTE 3:
The autograder's version of `short1_thing.py` may be different than the one that I have given you; don't worry! Just call the functions as required, and you will still pass the testcase. After all, isn't the whole point of functions that you **don't need to know** how they work?

(spec continues on the next page)

4 `skip_along.py`

Write a program, in a file named `skip_along.py` , which reads a single line of input. Use `strip()` to remove all leading and trailing whitespace, but leave any internal whitespace intact.

First, print out the length of the string, like this:

```
Length (after stripping): 10
```

Now, use the slice operator `[start:end]` to slice out substrings from the string. You will then print out the various substrings, separated by commas, on a single line, like this:

```
a,def,klmno,vwxyz
```

You must be able to handle any length string - meaning that you must be able to handle an arbitrary number of substrings. The slices that you keep will follow this pattern:

- Slice out exactly **1** character from the beginning
- Then reject the next **2** characters
- Then slice out the next **3**
- The reject the next **4**
- and so on...

If the last slice is shorter than you want, print it anyway (see the example above).

NOTE: There may be spaces in the middle of your input; treat them exactly like any other character.

5 Turning in Your Solution

You must turn in your code using GradeScope.