Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2013

Lab 8 - Python Lists

Objectives

For this lab, you'll develop functions that work with Python lists. All of the exercises are problems on the CodingBat Web site.

Attendance/Demo

When you have finished all the exercises, call a TA, who will review the functions you wrote. You may be asked to demonstrate your functions using CodingBat or Wing IDE 101. For those who don't finish early, the TAs will ask you to present whatever functions you've completed, starting at about 30 minutes before the end of the lab period.

If you don't finish all the exercises during today's lab, you must finish them on your own time as "homework".

References

Think Python: How to Think Like a Computer Scientist, Chapter 10.

About CodingBat

The CodingBat site (codingbat.com) provides numerous small-scale programming problems to help students develop fundamental programming skills (e.g., writing code that uses Boolean logic, loops, lists and strings). The problems are "live": you type your Python code in a box displayed in a Web browser window, and you test your solution by clicking a button. You receive immediate feedback indicating which tests passed and which tests failed.

Getting Started with CodingBat

Step 1: Visit codingbat.com. Click on the link Warmup-2 to display the list of medium warmup string/list problems. Click on the link array_count9.

Step 2: Read the description of the problem, which is similar to one of this week's lecture examples. Everywhere you see the word *array* in a problem description, substitute *list*. In some programming languages, list-like collections are known as arrays.

Type this function definition in the CodingBat editor window. (Yes, there's a bug. Don't fix it.)

```
def array_count9(nums):
    count = 0

    for item in nums:
        if item != 9:
            count = count + 1

    return count
```

Step 3: Click the Go button. A test program on the CodingBat server will run several tests on the functions, and the test results will be displayed in a table. Each row of the table summarizes one test, indicating the arguments that are passed to the function when it is called, the result that a correctly implemented function is expected to return (column Expected), and the actual result returned by the function (column Run).

For example, when `array_count9` is passed the list [1, 2, 9], we expect that the function will return 9, because there's one 9 in the list. Instead, we see that the function returned 2.

After reviewing all the test results, it becomes apparent that value returned by the function is a count of the number of list items that are **not** 9. Checking the Python code reveals the bug: the condition in the `if` statement is incorrect.

Step 4: Correct the bug, changing:

```
if item != 9:
```

to:

```
if item == 9:
```

Step 5: Click the Go button. All the tests should pass, and a check-mark and the phrase "All Correct" should appear.

Part 1

For this lab, you're going to use CodingBat as the programming tool, but you'll need a copy of your solutions to show the TA and to submit to cuLearn. To do this:

- Launch Wing IDE 101 and create a new file. Save the file as codingbat-problems.py.
- As you complete each CodingBat problem, copy/paste your code from the CodingBat editor window into codingbat-problems.py. Save the file, then click Run to make sure that the code compiles and loads into the Python engine properly.

Go to the Python > List-1 section of CodingBat (Basic Python list problems).

Complete all of the problems in this section, using CodingBat to test your code, not Wing IDE. Your solutions should not have any loops.

Part 2

The three list processing examples that were presented using the Online Python Tutor in Tuesday's lecture are posted in the Week 8 section of cuLearn. Run these examples in OPT, and make sure you understand the algorithms before you do the following problems.

Go to the Python > List-2 section of CodingBat (Medium Python list problems).

As you did in Part 1, copy/paste your code for each completed problem from the CodingBat editor window into codingbat-problems.py. Save the file, then click Run to make sure that the code compiles and loads into the Python engine properly.

- Do problem count_evens.
- Do problem big_diff. Although the most Pythonesque solution is:

  ```
  def big_diff(nums):
      return max(nums) - min(nums)
  ```

  don't use this as your solution. Your solution should have one loop.
- Do problem has22.

Wrap-up

1. Remember to have a TA review your codingbat-problems.py file and give you a grade (Satisfactory, Marginal or Unsatisfactory) before you leave the lab. The TA may ask you to demonstrate some of the functions in codingbat-problems.py, using Wing IDE's Python shell to interactively call the functions.
2. Remember to save a copy of your codingbat-problems.py file (copy it to a flash drive, or email a copy to yourself, or store it on your M: drive - remember, files left on your desktop or in your Documents folder are not guaranteed to be there the next time you log in).
3. Log in to cuLearn, click the Submit Lab 8 link and submit codingbat-problems.py. After you click the Add submission button, drag the file to the File submissions box. After the icon for the file appears in the box, click the Save changes button. At this point, the submission status for your files is "Draft (not submitted)". You can resubmit the file by clicking the Edit my submission button. After you've finished uploading the file, remember to click the Submit assignment button to change the submission status to "Submitted for grading".

Challenge Problems

The solutions to these problems require a "summing loop", which has not yet been presented in lectures. Read Section 10.7, *Map, filter and reduce*, in *Think Python*.

- Do problem centered_average.
- Do problem sum13.
- Do problem sum67.