Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2014

Lab 8 - Learning about 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

To receive credit for this lab, you must make reasonable progress towards completing the exercises and demonstrate the code you complete. **Also, you must submit your lab work to cuLearn by the end of the lab period**. (Instructions are provided in the *Wrap Up* section at the end of this handout.)

When you have finished all the exercises, call a TA, who will review the code you wrote. You may be asked to demonstrate your functions using CodingBat or Wing IDE 101. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before your next lab.**

References

See the Week 8 section of the cuLearn course page for the reading assignment from *Think Python* and *Practical Programming (Second Edition)*, and the lecture examples that introduce lists.

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 in the Python section (right-hand side of the page) to display the list of medium-difficulty warmup string/list problems. Click on the link array_count9.

Step 2: Read the description of the problem, which is similar to one of last 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. The Expected column indicates the arguments that were passed to the function when it was called and the result that a correctly implemented function is expected to return. The Run column indicates the actual result returned by the function.

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

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

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.
2. Go to the Python > List-1 section of CodingBat (Basic Python list problems).
    - Complete all 12 problems in this section, using CodingBat to test your code, not Wing IDE. Your solutions should not have any loops. 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.

Part 2

1. The Week 8 section of the cuLearn course page contains links to the Online Python Tutor examples that were presented in last week's lectures. Run these examples in OPT, and make sure you understand the list processing algorithms before you do the following problems.
2. Go to the Python > List-2 section of CodingBat (Medium Python list problems).
3. 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.
   - Do problem centered_average. Hint: your solution will be shorter if you use Python's `min` and `max` functions to determine the smallest and largest values in the list.

Wrap-up

1. Remember to have a TA review your codingbat-problems.py file, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/sign-out sheet. The TA may ask you to use CodingBat to demonstrate one of your solutions (you can copy/paste the function from codingbat-problems.py to the CodingBat editor).
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. Before you leave the lab, log in to cuLearn and submit codingbat-problems.py. To do this:

   3.1. Click the Submit Lab 8 link. A page containing instructions and your submission status will be displayed. After you've read the instructions, click the Add submission button. A page containing a File submissions box will appear. Drag codingbat-problems.py to the File submissions box. Do not submit another type of file (e.g., a zip file, a RAR file, a .txt file, etc.)

3.2. After the icon for codingbat-problems.py appears in the box, click the Save changes button. At this point, the submission status of your file is "Draft (not submitted)". If you're ready to finish submitting the file, jump to Step 3.4. If you instead want to replace or delete your "draft" file submission, follow the instructions in Step 3.3.

3.3. You can replace or delete the file by clicking the Edit my submission button. The page containing the File submissions box will appear.

 3.3.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the File submissions box, then click the Overwrite button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the Save changes button.

 3.3.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the Delete button., then click the OK button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, click the Save changes button.

3.4. Once you're sure that you don't want to make any changes, click the Submit assignment button. A Submit assignment page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the Continue button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading". **Make sure you do this before you leave the lab.**

Extra Practice

- Go to the Online Python Tutor site (there's a link at the start of the Labs section in the cuLearn course page). For each of the problems, copy/paste your function into the OPT editor. You'll need to add some statements to call the function (suggestion: use the CodingBat test cases as a starting point). Execute the function, statement-by-statement, and make sure you understand the diagrams produced by OPT.
- Do problem sum13 in the Python > List-2 section of CodingBat.
- Do problem sum67 in the Python > List-2 section of CodingBat.