Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2017

Lab 8 - Learning about Python Lists

Objectives

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

Demo/Grading

When you have finished all the exercises, call a TA, who will review your solutions, ask you to demonstrate some of them using CodingBat or Wing IDE 101, and assign a grade. For those who don't finish early, a TA will grade the work 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.**

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 about the tests that passed and the tests that failed.

Part 1 - Getting Started with CodingBat

Step 1: Visit codingbat.com. Click on the Python tab, then click the link Warmup-2 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 the 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.)

Step 3: 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 4: Click the Go button. A test program on the CodingBat server will run several tests on the functions, then displays the results in a table. Each row of the table summarizes one test:

- The cell in the Expected column indicates the arguments that were passed to the function and the result that a correctly implemented function is expected to return.

- The cell in the Run column indicates the actual result returned by the function.
- The cells in next two columns contain OK and a green box if the test passed. If the test failed, these columns contain an `x` and a red box.

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. We see that this test failed (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 5: Correct the bug by changing:

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

to:

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

Step 6: Click the Go button. All the tests should now pass, and a green check-mark and the phrase "All Correct" should appear.

Part 2 - Basic List Problems

Step 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. To do this, launch Wing IDE 101 and create a new file. Save the file as codingbat-problems.py.

Step 2: Go to the Python > List-1 section of CodingBat (Basic Python list problems).

Step 3: 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 interpreter properly.

Part 3 - Medium Difficulty List Problems

Step 1: The Lecture Materials section of the cuLearn course page contains links to several Python Tutor examples that use lists. If you have not done so already, run these examples in PyTutor, and make sure you understand the list processing algorithms before you do the following problems.

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

Step 3: As you did in Part 2, 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 interpreter 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 grading/sign-out sheet. The TA may ask you to use CodingBat to demonstrate some of your solutions (you can copy/paste the function from codingbat-problems.py to the CodingBat editor).
2. Remember to backup codingbat-problems.py before you leave the lab; for example, copy it to a flash drive and/or a cloud-based file storage service.

Extra Practice

- Go to the Python Tutor website (pythontutor.com). Copy/paste your `count_evens` function from codingbat-problems.py into the PyTutor editor. You'll need to write a short script 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 PyTutor.

  Repeat this exercise for your `big_diff`, `has22` and `centered_average` functions.
- Do problem sum13 in the Python > List-2 section of CodingBat.
- Do problem sum67 in the Python > List-2 section of CodingBat.