Carleton University
Department of Systems and Computer Engineering
ECOR 1051 - Fundamentals of Engineering I

Lab 9 - Learning More about Python Lists

Objective

- To develop some functions that use loops to process lists.

Learning outcomes: 4, 5, 7; Graduate attributes: 1.3, 5.3 (see the course outline)

Overview

This lab consists of four exercises that you'll get checked by the TAs, plus two "challenge" exercises. Each solution requires no more than one loop.

Your solutions can use:

- the `[]` operator to get and set list elements (e.g., `lst[i]` and `lst[i] = a`)
- the `in` and `not in` operators (e.g., `a in lst`)
- the list concatenation operator (e.g., `lst1 + lst2`)
- the list replication operator (e.g., `lst * n` or `n * lst`)
- Python's built-in `len`, sum, `min` and `max` functions, unless otherwise noted.

Your solutions **cannot** use:

- list slicing (e.g., `lst[i : j]` or `lst[i : j] = t`)
- the `del` statement (e.g., `del lst[0]`)
- Python's built-in `reversed` and `sorted` functions.
- any of the Python methods that provide list operations; e.g., `append`, `clear`, `copy`, `count`, `extend`, `index`, `insert`, pop, `remove`, `reverse` and `sort`
- list comprehensions

Getting Started

Begin by creating a new file within Wing 101. **Save it as lab9.py**

Automated testing is required for this lab. (See Lab 6 for definitions of manual and automated testing).

Debugging hint: If a function fails one or more tests, remember the lessons in debugging in Lab 3. Alternatively, execute the code in PythonTutor and use the memory diagrams produced by PyTutor to help you locate the flaw in your solution.

Exercise 1

Use the function design recipe to develop a function named `count_evens`. The function takes a list of integers, which may be empty. The function returns the number of even integers in the list.

Common Question: Is zero odd or even? Are negative numbers odd or even?

Answer: These are questions that you can answer yourself! Please do a web search using "Are negative numbers odd or even". Wolfram is a reliable source.

Exercise 2

Use the function design recipe to develop a function named `big_diff`. The function takes a list of integers. (Assume that the list has at least two integers). The function returns the difference between the largest and smallest elements in the list. For example, when the function's argument is `[10, 3, 5, 6]`, the function returns 7.

For this exercise, your function is **not** permitted to call Python's `min` and `max` functions. It is true that Python offers a very elegant and efficient solution (a version shown below) but in this exercise, we want you to practice writing loops.

For now, this wonderful Python solution is not permitted in this lab, but free feel to file it away for later use!

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

Exercise 3

Use the function design recipe to develop a function named `has22`. The function takes a list of integers, which may be empty. The function returns `True` if the list contains a 2 next to a 2. Otherwise, the function returns `False` . For example, when the function's argument is `[1, 2, 2, 3]`, the function returns `True`. When the function's argument is `[4, 2, 3, 2]`, the function returns `False`.

Exercise 4

The *centered average* of a list of numbers is the arithmetic mean of all the numbers in the list, except for the smallest and largest values. If the list has multiple copies of the smallest or largest values, only one copy is ignored when calculating the mean. For example, given the list `[1, 1, 5, 5, 10, 8, 7]`, one 1 and 10 are ignored, and the centered average is the mean of the remaining values; that is, 5.2.

Use the function design recipe to develop a function named `centered_average`. The function takes a list of integers. (Assume that the list has at least three integers). The function returns the centered average of the list.

Hint: In this exercise you are permitted and encouraged to use Python's `min` and `max` functions.

Exercise 5 (OPTIONAL CHALLENGE, not required for marking)

Use the function design recipe to develop a function named `sum13`. The function takes a list of integers, which may be empty. The function returns the sum of the numbers in the list, returning 0 for an empty list. The number 13 is very unlucky, so any 13 in the list is not counted, and any number that comes immediately after a 13 is not counted. For example, when the function's argument is `[1, 2, 2, 1, 13]`, the function returns 6. When the function's argument is `[13, 1, 2, 13, 2, 1, 13]`, the function returns 3.

Exercise 6 (OPTIONAL CHALLENGE, not required for marking)

Use the function design recipe to develop a function named `sum67`. The function takes a list of integers, which may be empty. The function returns the sum of the numbers in the list, returning 0 for an empty list. The function will ignore sequences of integers starting with a 6 and extending to the next 7, inclusive. (Assume that every 6 will be followed by at least one 7.) For example, when the function's argument is `[1, 2, 2, 6, 99, 99, 7]`, the function returns 5 (the 6, 99, 99 and 7 are ignored.)

Wrap Up

Ensure that your code meets the posted marking rubrics for the labs.

Submit file lab9.py.

You are required to keep a backup copy of (all) your work for the duration of the term.

Last edited: April 23, 2020