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

Lab 10 - 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

Each solution requires no more than one loop. Each of your function definitions must have type annotations and a complete docstring. Use the Python shell to test your functions.

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`), unless otherwise noted.
- the list replication operator (e.g., `lst * n` or `n * lst`).
- Python's built-in `len`, `min` and `max` functions.

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 `sum`, `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`, unless otherwise noted. (See *Practical Programming*, pages 141-142.)
- list comprehensions.

Getting Started

Begin by creating a new file within Wing 101. **Save it as `lab10.py`**

You must use automatic testing in this lab. Now that you are familiar with loops, it should make your automatic testing easier.

Exercise 1

Use the function design recipe to develop a function named `bank_statement`. The function takes a list of floating point numbers, which will always have at least one number. Positive numbers represent deposits into a bank account and negative numbers represent withdrawals from the account. The function returns a new list of three numbers: the first will be the sum of the deposits, the second (a negative number) will be the sum of the withdrawals, and the third will be the current account balance. These numbers must be rounded to two digits of precision after the decimal point (read Chapter 3, pages 33-34).

Your function must have exactly one loop.

Note: when the list returned by the function is displayed, number such as 15.0 or -17.3 will be displayed with one digit after the decimal point instead of two. This is ok.

Exercise 2

Use the function design recipe to develop a function named `divisors`. The function takes a positive integer *n* and returns a list containing all the positive divisors of *n*. For example, if `divisors` is called with argument 6, the list will contain 1, 2, 3 and 6. If `divisors` is called with argument 9, the list will contain 1, 3 and 9.

Your function must have exactly one loop. Your function can call Python's list methods, but this isn't necessary.

Exercise 3

Use the function design recipe to develop a function named `reverse` that takes a list, which may be empty. The function returns a new list that contains all the elements of the original list in reverse order. For example, when the function's argument is `[4, 2, 3, 2]`, the function returns the new list `[2, 3, 2, 4]`. When the function's argument is an empty list, it returns a new empty list.

Your function must have exactly one loop. Your function cannot use the concatenation (`+`) operator or call the `append` method. Instead, use the `[]` operator to get and set list elements.

Wrap Up

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

Submit file lab10.py.

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

See extra practice questions on the next page.

Extra Practice

1. Rewrite your `reverse` function from Exercise 3 so that the reversed list is initially an empty list. The function must use the `+` operator to build the reversed list.

2. Rewrite your `reverse` function from Exercise 3 so that the reversed list is initially an empty list. The function must use the `append` method to build the reversed list.

3. During the midterm and final exams, you will be expected to draw diagrams similar to those created by Python Tutor. Use PyTutor to visualize the execution of your exercise solutions. Remember, PyTutor doesn't have a shell. After copying your function definitions into the PyTutor editor, you will have to type assignment statements that call the function and assign the returned values to variables.

Last edited: April 23, 2020