Carleton University
Department of Systems and Computer Engineering
ECOR 1041 - Fundamentals of Engineering I

Lab 9 - Learning about Python Lists

Objective

- To learn some of the operators supported by Python's `list` type and some built-in functions that operate on lists.
- To develop functions that have lists as arguments and/or create and return lists.

Overview

This lab consists of seven short exercises.

You must do automated testing for the functions that do not return a list. Bringing the functions you wrote for automatic testing during the previous labs will be helpful.

You can do manual testing for the functions that return a list.

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`, `min` and `max` functions

Your solutions to this lab **cannot** use:

- `for` or `while` loops
- 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

Debugging hint: Refer back to Lab 3 for instructions on using the debugger in Wing.

Getting Started

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

Exercise 1

Use the function design recipe to develop a function named `first_last`. The function takes a list of integers (which may be empty) and an integer number x. The function returns `True` if the integer number x is the first element, the last element, or if the first and last elements are the integer number x. Otherwise, the function returns `False`.

Common Question: What items should my list contain? Are we supposed to put in random values?

Answer: Yes, you must come up with the content of the lists to be tested. The values should not be completely random, but we are not imposing any conditions on their values (outside of those in the exercise itself). For example, you are welcome to use 10 or 1002 or -10, but for this exercise, it would be logical to include x value as one of the elements of a list.

Common Question: How many docstring test examples are needed.

Answer: The number of tests and the list contents for each test will depend on what the function does. For this exercise, the following examples are logical:

1. - a list in which the first element is x
2. - a list in which the last element is x
3. - a list in which both the first and last element is x
4. - a list that has no x value
5. - a list that has one or more x value, but they are not the first or last elements

The docstring test examples are not an exhaustive set of test cases, but each should demonstrate a different aspect of the function's behavior. For example, the three reasons for which the function will return True can be illustrated by 3 different docstring examples - see the first three points above. The remaining examples illustrate cases in which the function will return False.

Exercise 2

Use the function design recipe to develop a function named `common_end`. The function takes two lists of integers that are not empty but which may have different lengths. The function returns `True` if they have the same first element or the same last element or if the first and last elements of both lists are the same. Otherwise, the function returns `False`.

Exercise 3

Use the function design recipe to develop a function named `average`. The function takes a list containing five integers. The function returns the average of all the elements.

Exercise 4

Use the function design recipe to develop a function named `reverse`. The function takes a list containing five integers. The function returns a new list containing the same elements in reverse order.

For example, when the function is called with `[1, 2, 3, 4, 5]`,the function returns `[5, 4,3, 2, 1]`

Exercise 5

Use the function design recipe to develop a function named `max_list`. The function takes a list containing three integers. The function determines the largest element in the list. It returns a new list in which all the elements are initialized to that value of the largest element in the list. For example, when the argument is `[2, 9, 3]`, the function returns `[9, 9, 9]`.

Exercise 6

Use the function design recipe to develop a function named `middle_way`. The function takes two lists that each contain three integers. The function returns a new list containing its middle elements. For example, when the arguments are lists `[1, 2, 3]` and `[4, 5, 6]`, the function returns `[2, 5]`.

Exercise 7

Use the function design recipe to develop a function named `has_elements`. The function takes a list containing integers, and two integers parameters **x** and **y**. The function returns `True` if the list contains **x** or **y** or both values. Otherwise, the function returns `False`.

Wrap Up

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

- Make sure that you included your name and student number
- Check proper use constants (UPPER_CASE) and variables (lower_case) (There is a 10/100 deduction for misuse of UPPPER & lower case)
- Check the indents of the function bodies. (There is a 10/100 deduction for misuse of indentation)
- Check file organization: (1) imports, (2) CONSTANTS, (3) all function definitions; (4) Main Script (There is a 10/100 deduction for not organizing the file according to the instructions)
- Confirm that your filename matches exactly.
- Confirm that your .py script runs properly, otherwise, the TA will also assign a zero.
- Submit the file on Brightspace.

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

Last edited: January 31, 2022