CSC108H Lists Lab

1 Lists

In this section, you will write short functions or statements that involve lists. In most of the exercises, you will need to use list methods so that you can practice using list tools. Remember to use the Python functions `dir` and `help` to get information about methods.

1. Type this assignment statement into the Python shell:

   ```
   names = ['Bob', 'Ho', 'Zahara', 'Amitabha', 'Dov', 'Maria']
   ```

   For the following steps, use `names` and slice notation.

   (a) Write a slicing expression that produces this new list: `['Zahara', 'Amitabha', 'Dov']`

   (b) Write a slicing expression that produces this new list: `['Bob']`

   (c) Write a slicing expression that produces this new list: `['Amitabha', Dov', 'Maria']`

2. Given a list `L` and a value `v`, write an expression that removes the first occurrence of `v` from `L`. You can solve this using slicing or a list method. (Remember that slicing always returns a new list and that you want to change `L`.)

3. Write an expression that adds the string `"How are you?"` to the **front** of the list `["I am well."]` so that you end up with the list `["How are you?","I am well."]`. Again, this can be done using slicing or a list method.

Switch driver and navigator.

4. Write code that turns `[2, 4, 99, 0, -3.5, 86.9, -101]` into `[99, 86.9, 4, 2, 0, -3.5, -101]`. You should use at most two method calls.

5. Open a new file and in it write a function `every_third` that takes a list as a parameter and returns a new list that contains every third element of the original list, starting at index 0. You may use a for-loop (with `range`) or while-loop in your solution; do not use slice notation. For example, the call `every_third([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])` should return `[1, 4, 7, 10]`.

6. In the same file write a second function `every_ith` that takes a list `L` and an integer `i` as parameters and returns a list consisting of every `i`*th* element of `L`, starting at index 0. Again, use a for-loop or while-loop and do not use slice notation.

Switch driver and navigator.

7. Now that you have `every_ith` available, write a third function `every_third_revisited` that has the same behaviour as `every_third` but is only a single line of code.

Show your TA your work and continue with the loop exercises in the next section.

2 Lists and Loops

Use loops to write the following functions in a new file. It is up to you to choose the type of loop to use (`while` or `for` or `for-range`). Do *not* use list methods.

`print_list(L)`	Print each element of list `L` on a separate line. `L` is unchanged.
`print_list_even(L)`	Print the elements of list `L` that occur at even indices. `L` is unchanged.
	Switch driver and navigator.
`print_list_reverse(L)`	Print the elements of list `L` from the end of the list to the front. `L` is unchanged.
`sum_elements(L)`	Sum the elements of the list `L` of ints, starting from the front of list, until the total is over 100 or the end of the list is reached, and return the sum at that point (as an int). Do not change `L`.
`duplicates(L)`	Return `True` iff list `L` contains at least two adjacent elements with the same value. For example, `duplicates([1,1,2])` returns `True` and `duplicates([1,2,1])` returns `False`.

3 Nested Lists

List elements may be lists themselves. When this is the case, we refer to the whole structure as a *nested list* or a *list of lists*:

```
    pets = [["Shoji", "cat", 18], ["Hanako", "dog", 15], ["Sir Toby", "cat", 10],
        ["Sachiko", "cat", 7], ["Sasha", "dog", 3], ["Lopez", "dog", 13]]
```

We can access each element of list `pets` using its index:

```
>>> pets[3]
["Sachiko", "cat", 7]
```

We can also access elements of the inner lists. For example, since `pets[3]` refers to a list, we can use `pets[3][2]` to access its element at position 2:

```
>>> pets[3][2]
7
```

This is saying that element 2 of element 3 of the list `pets` (the age of the cat named "Sachiko") is 7.

Write the following loops and functions and call each function to verify your work.

1. Write a for-loop that prints each list from list `pets` on a separate line.
2. Write a for-loop that prints the second element of each inner list in list `pets` on a separate line.

   Switch driver and navigator.

3. Write a for-loop that computes the number of dogs in `pets`.
4. Write a for-loop that computes the sum of the ages of the animals in `pets`. Ages are the third element of the inner lists.
5. Write a function `nested_lengths` that takes a list `L` as a parameter and returns a list of the lengths of the sublists. For example if `L=[[1,2],['a','b','c']]`, calling the function on `L` would return the list `[2,3]`.

Show your TA your work so that you can get credit for the lab.