**Practice 05 - Sequences and Data Abstraction**

**Practice 1:** List Indexing

For each of the following lists, what is the list indexing expression that evaluates to 7? For example, if x = [7], then the answer would be x[0]. You can use the interpreter or Python Tutor to experiment with your answers.

| >>> x = [1, 3, [5, 7], 9]  \_\_\_\_\_\_  >>> x = [[7]]  \_\_\_\_\_\_  >>> x = [3, 2, 1, [9, 8, 7]]  \_\_\_\_\_\_  >>> x = [[3, [5, 7], 9]]  \_\_\_\_\_\_ | >>> lst = [3, 2, 7, [84, 83, 82]]  >>> lst[4]  \_\_\_\_\_\_  >>> lst[3][0]  \_\_\_\_\_\_ |
| --- | --- |

**Practice 2:** What would Python Display?

| >>> [x\*x **for** x **in** range(5)]  \_\_\_\_\_\_  >>> [n **for** n **in** range(10) **if** n % 2 == 0]  \_\_\_\_\_\_  >>> ones = [1 **for** i **in** ["hi", "bye", "you"]]  >>> ones + [str(i) **for** i **in** [6, 3, 8, 4]]  \_\_\_\_\_\_  >>> [i+5 **for** i **in** [n **for** n **in** range(1,4)]]  \_\_\_\_\_\_ |
| --- |
| >>> [i\*\*2 **for** i **in** range(10) **if** i < 3]  \_\_\_\_\_\_  >>> lst = ['hi' **for** i **in** [1, 2, 3]]  >>> print(lst)  \_\_\_\_\_\_  >>> lst + [i **for** i **in** ['1', '2', '3']]  \_\_\_\_\_\_ |

### 

### Practice 3: Merge

### We've provided you with an implementation of the function merge. merge takes 2 *sorted* lists lst1 and lst2, and returns a new list that contains all the elements in the two lists in sorted order.

### A list is sorted if the elements appear in nondecreasing order. [1, 2, 3] is a sorted list, but [3, 1, 2] is not.

### This solution does not pass the doctests. It is your job to figure out what is wrong with the implementation and correct the error.

### *Hint:* if you're having trouble figuring out what's going on in the implementation, you might want to try the following:

### Use print expressions to debug. If you write print("DEBUG:", x), then you can print out the value of variable x without causing the ok tests to fail due to extra output. You should, of course, replace x with whatever expression you want to print.

### Come up with some additional doctests to understand the behavior of the function. You should not delete the doctests that are currently present, but you are free to add some of your own.

### Draw out an environment diagram for one of the doctests to see where the value of a variable differs from what you expect.

### You are free to make any changes to the implementation as long as they do not fundamentally alter the approach (e.g. you should not convert the recursive solution to an iterative approach).

### 

| **def** **merge**(lst1, lst2):  """Merges two sorted lists.  >>> merge([1], [2])  [1, 2]  >>> merge([2], [1])  [1, 2]  >>> merge([1, 3, 5], [2, 4, 6])  [1, 2, 3, 4, 5, 6]  >>> merge([5, 7], [2, 4, 6])  [2, 4, 5, 6, 7]  """  **if** **not** lst1 **or** **not** lst2:  **return** []  **elif** lst1[0] < lst2[0]:  **return** [lst1[0]] + merge(lst1[1:], lst2)  **else**:  **return** [lst2[0]] + merge(lst1, lst2[1:]) |
| --- |

### 