

Study Guide #6 (List Memory Diagrams and Nested Loop)

Note: list comprehensions and sorting will **not** be covered on Quiz 6, but on a later quiz.

Optimal use of this study guide:

1. Try to complete the questions below in 15 minutes. (This study guide is longer than a usual one, so don't be surprised if it takes you ***much*** longer than 15 minutes!)
2. Review the Study Guide Solutions
3. If there are places where things are unclear, ask a tutor/instructor about it

Learning Goals:

- Understand how to read and draw memory diagrams involving lists
- Understand how list slot assignment, `.append`, `.insert`, `.pop`, and aliasing involving lists are represented in memory diagrams.

Question 1: Memory Diagrams

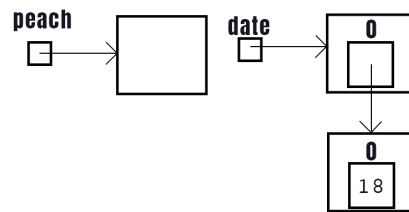
Part 1a: Write some Python code that produces the memory diagram below. You are already given two statements that show the initial state for the variable `item` and `obj` and your goal is to modify these variables to achieve the state in the given diagram. When writing your code, you may not create any new lists. You must only modify the two lists given. Feel free to use list slot assignment and the methods `.pop`, `.append`, and `.insert` to mutate either list.

The diagram illustrates the following structure: - A variable <code>item</code> points to a list object with two slots, 0 and 1. - A variable <code>obj</code> points to a list object with one slot, 0. - The list object pointed to by <code>item</code>'s slot 1 points to another list object with two slots, 0 and 1. - The list object pointed to by <code>obj</code>'s slot 0 points to the same list object as the one pointed to by <code>item</code>'s slot 1. - The final list object (pointed to by both <code>item</code>'s slot 1 and <code>obj</code>'s slot 0) contains the values 3 in slot 0 and 10 in slot 1.	<pre>obj = [[3], 7] item = [obj] # Add your code here # After you add your code, # what is printed by <code>print(item)</code>?</pre>
---	---

Part 1b:

You are given the memory diagram that represents the state of memory after these two lines have been executed:

```
peach = []  
date = [[18]]
```



In the box below, **draw the memory diagram** that represents the state *after all three* of the following lines have been executed.

```
date[0].append(peach)  
peach.append(date)  
peach.insert(0,2)
```

Learning Goals:

- Be able to predict outcome of nested loops
- Understand how to structure nested loops to generate desired outcomes

Question 2: Fill in the blanks in the code snippets to produce the output shown at right:

<pre>for age in range(____, ____): for letter in ____: print(age * letter)</pre>	<pre>XX 00 XXX 000</pre>
<pre>for x in range (____): for y in range (____): print(_____)</pre>	<pre>0 0 0 1 1 0 1 1 2 0 2 1 3 0 3 1</pre>

Question 3: Predicting the behavior of programs with nested loops

For each row below, write on the right what will be printed by the code shown on the left.

Code	What is printed?
<pre>names = [["Peter", "Mawhorter"], ["Sohie", "Lee"]] for name in names: for part in name: if 'o' in part: print(part)</pre>	
<pre>for h in 'hi': for b in 'bye': print(h + b)</pre>	
<pre>result = [] for x in range(1, 4): for y in range(2, 5): result.append(x*y) print(result)</pre>	

Question 4: hideCharsInWords (Defining a function with a nested loops)

Define a function `hideCharsInWords` that takes two arguments: (1) a list of words (strings) and (2) a string `charsToHide` containing characters that must be “hidden” in each word. `hideCharsInWords` **returns** a list of words that is the same as the original list of words except that in each word, each occurrence of a character in the word that appears in `charsToHide` is replaced by an underscore character ('_').

Below are some examples of `hideCharsInWords`. Note that case matters for alphabetic characters and it’s OK to have duplicate characters in `charsToHide`.

```
>>> hideCharsInWords(['Here', 'There', 'and', 'Everywhere'], "they")
['H_r_', 'T__r_', 'and', 'Ev_r_w__r_']

>>> hideCharsInWords(['Here', 'There', 'and', 'Everywhere'], "THEY")
['_ere', '_here', 'and', '_verywhere']

>>> hideCharsInWords(['What', 'happens', 'to', 'a', 'dream', 'deferred'],
                      "does it explode")
['Wha_', 'ha___n_', '___', 'a', '_r_am', '___f_rr__']

>>> hideCharsInWords(['462', 'widgets', 'cost', '$937.18!'], "st12345$") =>
['_6_', 'widge__', 'co__', '_9_7._8!']
```

*Define your `hideCharsInWords` function in this box. The function **must** use nested **for** loops to solve the problem. You are **not** allowed to define any helper functions.*