NextClass6

CS173: Intermediate Computer Science — Spring 2014
Instructor: Thomas Bressoud — 2014-02-11

Consider the following line-annotated Python code. Type the code into Python Tutor (pythontutor.com), editing the code, selecting the option of Python3. To understand what is going on, you may wish to toggle back and forth the options on inline primitives/render all on the heap and arrow references versus text labels. Then answer the following questions.

```
1  x = [1, 2, 3]
2  y = x
3  y.append(4)
4  print(x)
5  del y
6  del x
7
8  x = [1, 2, 3]
9  y = x[:]
10 y.append(4)
11 print(x)
12 del y
13 del x
14
15 def copy(L):
16     newL = []
17     for element in L:
18         newL.append(element)
19     return newL
20
21 x = [1, 2, 3]
22 y = copy(x)
23 y.append(4)
24 print(x)
25 del y
26 del x
27
28 x = [[1, 2, 3], 4]
29 y = x
30 y[0].append(5)
31 print(x)
32 del y
33 del x
34
35 x = [[1, 2, 3], 4]
36 y = copy(x)
37 y[0].append(5)
38 print(x)
```

1. In terms of both inspection of the source code and the resultant memory picture in Python Tutor, what is the difference between lines 1-4 and lines 8-11? What common programmer mistake does this illustrate?

2. Inspecting lines 21-24, what result do you expect from the print() on line 24, and what result in the memory picture for this case? *After* you predict the memory picture, use Python Tutor to walk through the code. Was you prediction on target?

3. Lines 28-31 and lines 35-38 are similar in form to lines 1-4 and lines 21-24. Predict and draw the resultant memory picture for these two cases. Then execute through the code in Python Tutor and identify exactly what is different between your prediction and the actual result.