Week: 05 Date: 09/25/2025

15-110 Recitation Week 5

Reminders

- Recitation feedback form
- Check 3 due Monday, Sept. 29 @ noon
- Check 2 and HW 2 revisions due Tuesday, Sept. 30 @ noon
- Midterm 1 on 10/01
 - o Review Sessions
 - o Small groups!
 - o OH and Piazza are always there for individual help

Overview

- Aliasing
- List methods
- 2D lists
- Recursion

Problems

LIST ALIASING

Code trace and compare the following two options for ways to create "empty" 2D lists:

Option 1:

```
inner = [0, 0, 0, 0]
outer = [inner, inner, inner]

Option 2:
  rows = 3
  outer = []
  for row in range(rows):
     outer.append([0, 0, 0, 0])
```

For each option, after running the code above, what are the values in outer?

```
Option 1: outer =
Option 2: outer =
```

After adding the following line of code and running it:

```
outer[0][0] = 42
```

What are the values in outer?

```
Option 1: outer =
Option 2: outer =
```

Be sure you can explain what difference you are seeing, and which option you should use and why.

LIST CODE WRITING

Write a function removeMatches(L, matchList) that takes in a list of numbers L, and removes all of the elements in L that are also in matchList. Write both a non-destructive and destructive version of this function.

Say we have L = [1,2,3,4,5]. Then, removeMatches(L, [1,5,10,15]) returns [2,3,4]. When it finishes running, L = [1,2,3,4,5]. And destructiveRemoveMatches(L, [1,5,10,15]) returns None, but L = [2,3,4] when done running.			
		Non-destructive:	Destructive:

2D LIST CODE TRACING

Examine the mystery function below. What would the following function calls return?

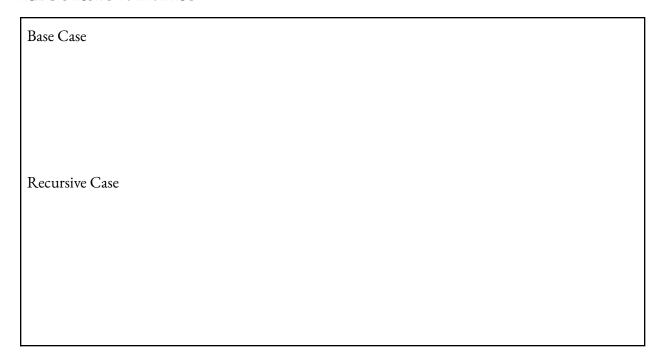
```
def mysteryFunction(rows, cols):
    outerList = []
    for i in range(rows):
        innerList = []
        for j in range(cols):
            if j % 2 == 0:
                innerList.append(i)
            else:
                innerList.append("wow")
        outerList.append(innerList)
    return outerList
```

Function calls:

```
mysteryFunction(3,4)

mysteryFunction(5,3)
```

RECURSION INTRO



Rewrite the following function using recursion (write on the right empty space):

```
def double(lst):
    result = []
    for i in range(len(lst)):
        result.append(2 * lst[i])
    return result

# double([1,2,3]) -> [2,4,6]
```