

# 15-110 Recitation Week 5

## Reminders

- [Recitation feedback form](#)
- Check 3 due Monday, Feb. 16 @ noon
- Check 2 and HW 2 revisions due Tuesday, Feb. 17 @ noon
- Midterm 1 on Feb. 18
  - Review Session: Sunday 2/15, 4pm @ Rashid Auditorium (GHC 4401)
  - Small groups!
  - Knowledge Tree for more programming practice
  - OH and Piazza are always there for individual help

## Overview

- Aliasing
- List methods
- 2D lists
- Recursion

# Problems

## LIST ALIASING

Examine the lines of code below. What does each variable hold once the code has run?

```
w = ['red', 'orange', 'yellow', 'green', 'blue']
x = w
x.remove('orange')
y = w + ['purple']
z = y[2:]
newLst = ['white', 'black']
y.append(newLst)
newLst.extend(['rainbow'])
```

| Variable | Value |
|----------|-------|
| w        |       |
| x        |       |
| y        |       |
| z        |       |
| newLst   |       |

Which variables are aliased at the end of this code?

## 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 ?

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

Base Case

Recursive Case

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]
```