Objectives

- Lists, continued
  - Making copies
  - Passing as parameters

Review

- What is a list?
- What is the syntax for a list?
- How can we make a list of numbers with a fixed step in one line of code?
- How are lists and strings similar?
  - What similar things can we do to lists and strings?
- How are they different?
  - What are the implications of those differences?
- What does **None** mean? When does it come up?

Review: Lists and Strings in Common

Concatenation	`<seq> + <seq>`
Repetition	`<seq> * <int-expr>`
Indexing	`<seq>[<int-expr>]`
Length	`len(<seq>)`
Slicing	`<seq>[:]`
Iteration	`for <var> in <seq>:`
Membership	`<expr> in <seq>`

Review: Strings vs. Lists

- Strings are **immutable**
  - Can't be mutated?
  - Err, can't be modified/changed
    - A change requires recreation
- Lists are **mutable**
  - Can be changed
    - Called "change in place"
  - Changes how we call/use methods

```
groceryList=["milk", "eggs", "bread", "Doritos", "OJ", "sugar"]

groceryList[0] = "skim milk"
groceryList[3] = "popcorn"

groceryList is now ["skim milk", "eggs", "bread", "popcorn", "OJ", "sugar"]
```

Review: Strings vs. Lists

Strings

- Methods that are meant to change a string return a *changed copy* of the String
- Consequence: Call the method and assign that to a variable
- Example use:
  - `upper = my_str.upper()`

Lists

- Methods that are meant to change a list change the list *in place*
  - Don't return anything
- Consequence: Call the method but don't assign it to a variable
- Example use:
  - `my_list.sort()`

Review: Special Value: None

(Similar to `null` in Java)

- Special value we can use
  - E.g., Return value from function/method when there is an error
  - Or if function/method does not return anything
- If you execute

```
myList = myList.sort()
print(myList)
```

  - Prints None because `myList.sort()` does **not** *return* anything

How do you fix the above code to do what you want?

Understanding Lists

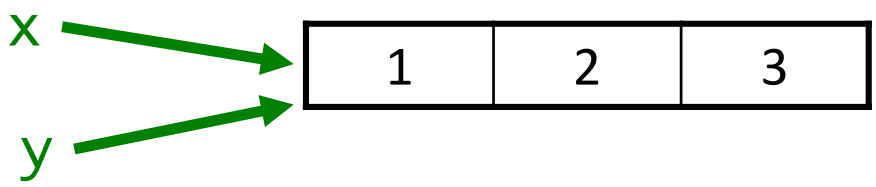
View in Python visualizer

- What does the following code display?

```
x = [1, 2, 3]
y = x
y[0] = -1
print(y)
print(x)
```

List Identifiers are **Aliases**

```
x = [1, 2, 3]
y = x
```



- y is **not** a copy of x
- y is another alias to that list/object
  - y points to what x points to
- How to make a copy of x?

```
y = x + []
```

Empty list

OR

```
y = []
y.extend(x)
```

Immutable vs Mutable Parameters

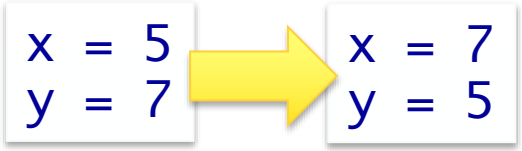
PASSING PARAMETERS

Passing Parameters

- Only ***copies*** of the actual parameters are given to the function
  - For **immutable** data types

Which are?

- The ***actual*** parameters in the calling code do not change
- **Swap example:**
  - Swap two values in script
  - Then, put into a function



Recall: Immutable Data is Passed by Value

```
def main():
    x = 5
    y = 7

    swap(x, y)

    print("x =", x)
    print("y =", y)

def swap(a, b):
    tmp = a
    a = b
    b = tmp
    print(a, b)

main()
```

This code does not have the desired effect in that x and y are not swapped.

Since integers are passed ***by value***, the values of x and y are not changed by the call to the swap function.

swap.py

Lists as Parameters to Functions

- Lists are not passed-by-value/copied
- Different from immutable types (e.g., numbers, strings)
- Function parameter is actually an ***alias*** to the list in memory

Impact: If a list that is passed as a parameter into a function is **modified *in* the function**, the list **is modified *outside* the function**

Problem:
Sort a list of 3 numbers, in descending order

```
# order list such that list3[0] >= list3[1] >= list3[2]
def descendSort3Nums( list3 ):
```

Called as:

```
myList = …
descendSort3Nums(myList)
print(myList)
```

Implement two ways:

- Using *only* list methods
- No list methods and only 3 comparisons

descendSort.py

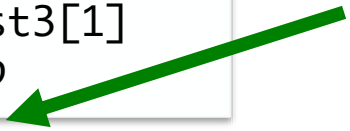
Descend Sort a List w/ 3 elements

```
def descendSort3Nums(list3):
    if list3[1] > list3[0]:
        # swap 'em
        tmp = list3[0]
        list3[0] = list3[1]
        list3[1] = tmp

    if list3[2] > list3[1]:
        tmp = list3[1]
        list3[1] = list3[2]
        list3[2] = tmp

    if list3[1] > list3[0]:
        tmp = list3[0]
        list3[0] = list3[1]
        list3[1] = tmp
```

```
def main():
    list = [1,2,3]
    descendSort3Nums(list)
    print(list)
```



Function does **not** *return* anything.
Simply modifies the `list3` parameter.

Comparing List Functions

[Impure?] Function

```
def descendSort3Numbers(list3):

    if list3[1] > list3[0]:
        # swap 'em
        …
    if list3[2] > list3[1]:
        # swap 'em
        …
    if list3[1] > list3[0]:
        # swap 'em
        …
```

Pure Function

```
def createDescendSort3Numbers(list3):
    copyOfList3 = list3 + []

    if copyOfList3[1] > copyOfList3[0]:
        # swap 'em
        …
    if copyOfList3[2] > copyOfList3[1]:
        # swap 'em
        …
    if copyOfList3[1] > copyOfList3[0]:
        # swap 'em
        …

    return copyOfList3
```

Testing List Functions

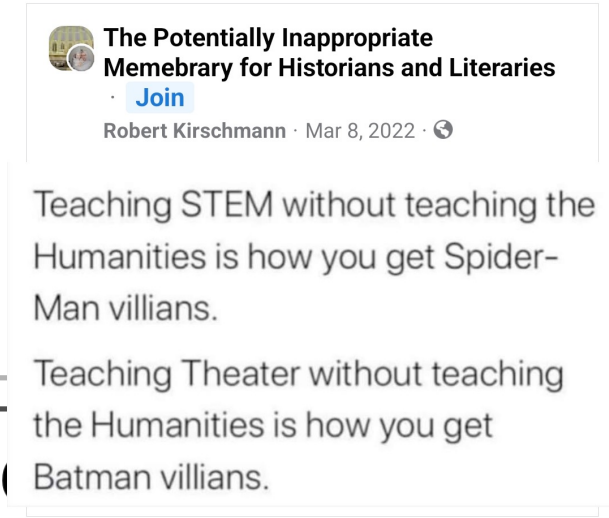
Testing a function that modifies the list parameter, nothing returned

```
def testDescendSort3Nums():
    origList = [1, 2, 3]
    descendSort3Nums(origList)
    # test that the list sorted is in reverse order
    test.testEqual( origList, [3, 2, 1] )
```

Testing a pure function that returns a copy of the list, modified

```
def testCreateDescendingSort3Nums():
    origList = [1, 2, 3]
    test.testEqual( createDescendingList(origList), [3, 2, 1])
    # verify that the original list didn't change.
    test.testEqual( origList, [1, 2, 3] )
```

Broader Issue



Pod 1	Pod 2	Pod 3	Po	
Abrar Jaz Juyoung Rowen	Ben R. Caleb Ilaria Julia	Berkley Brielle Ruoan Wesley	Devin Liliane Renee Sam	Ben T. Brett Cheng Hudson

Broader Issue

- What is public-key encryption?
- What is cryptocurrency?
- In the movie *Tombstone*, Ike Clanton says, “Listen, Mr. Kansas Law Dog. Law don't go around here. Savvy?”
  - Compare the privacy concerns in the articles and the role of the government/the law
  - How has the role of government/law changed over time? Where do we go from here?

Update

- “A US judge sentenced BitMEX co-founder Samuel Reed to 18 months of probation, saying he had a lesser role than two other co-founders in the offenses at the cryptocurrency derivative exchange.”

Looking Ahead

- Pre Lab 8 due before lab on Tuesday