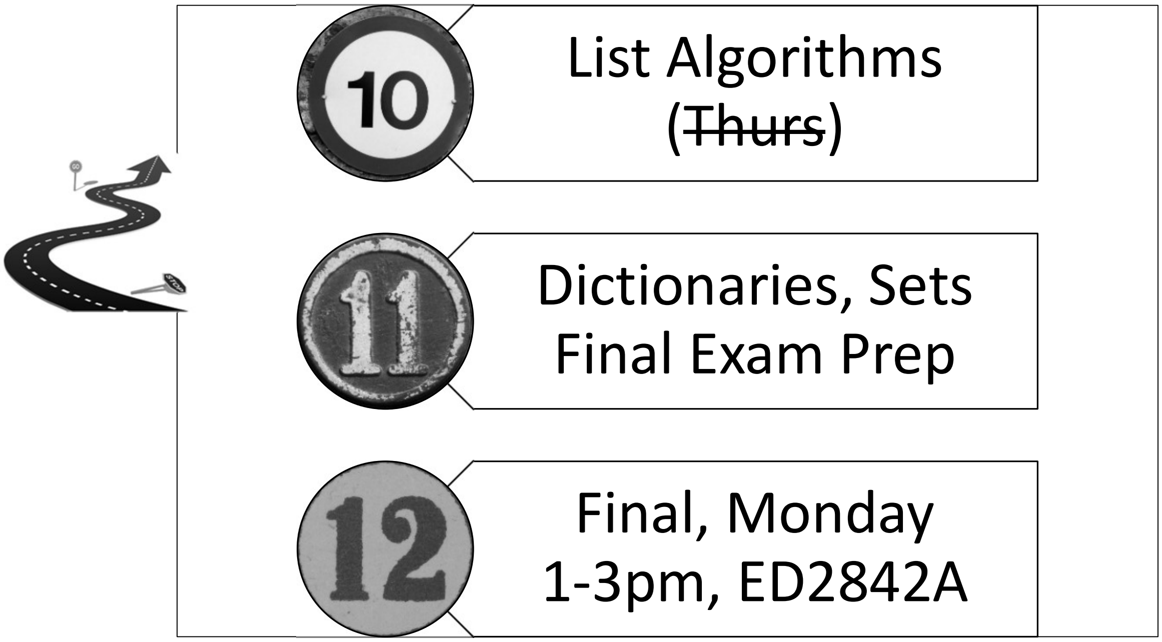
List Algorithms (Ch. 7)

CSC110: INTRO TO COMPUTER PROGRAMMING WITH PYTHON

INSTRUCTOR: BILL BARRY

1

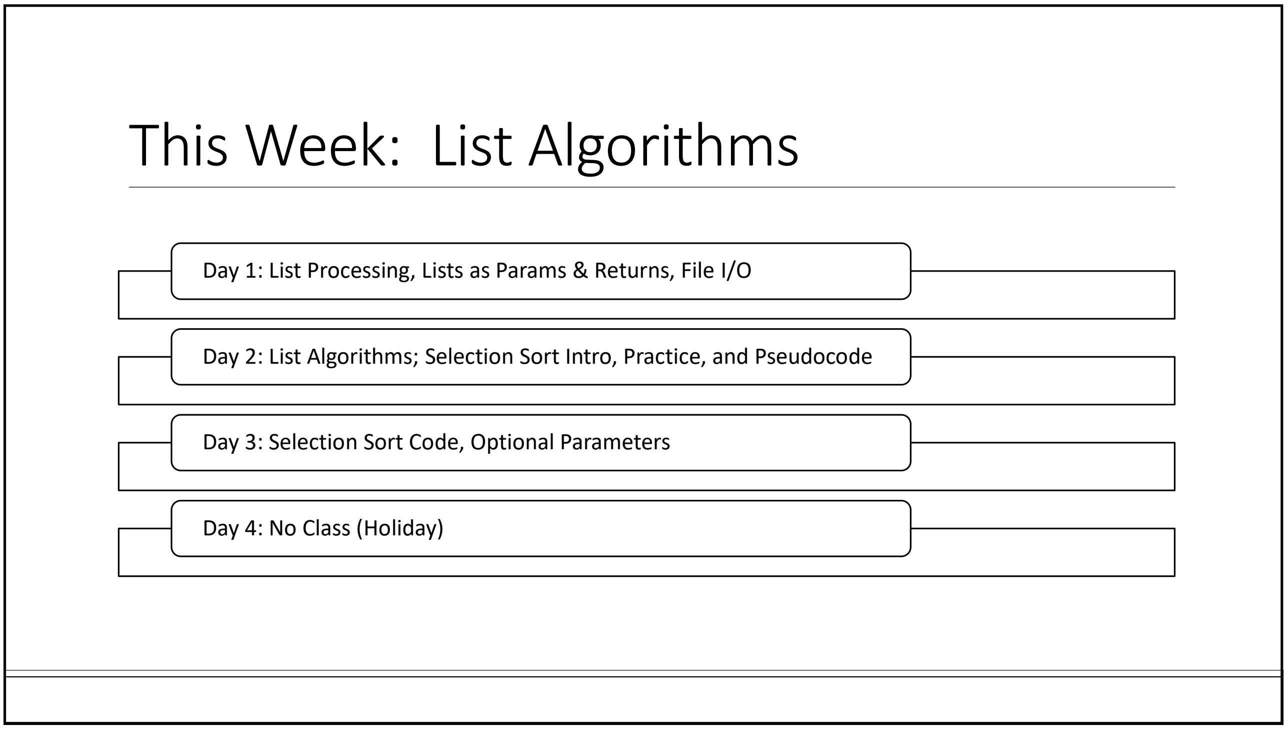


List Algorithms (~~Thurs~~)

Dictionaries, Sets
Final Exam Prep

Final, Monday
1-3pm, ED2842A

2



This Week: List Algorithms

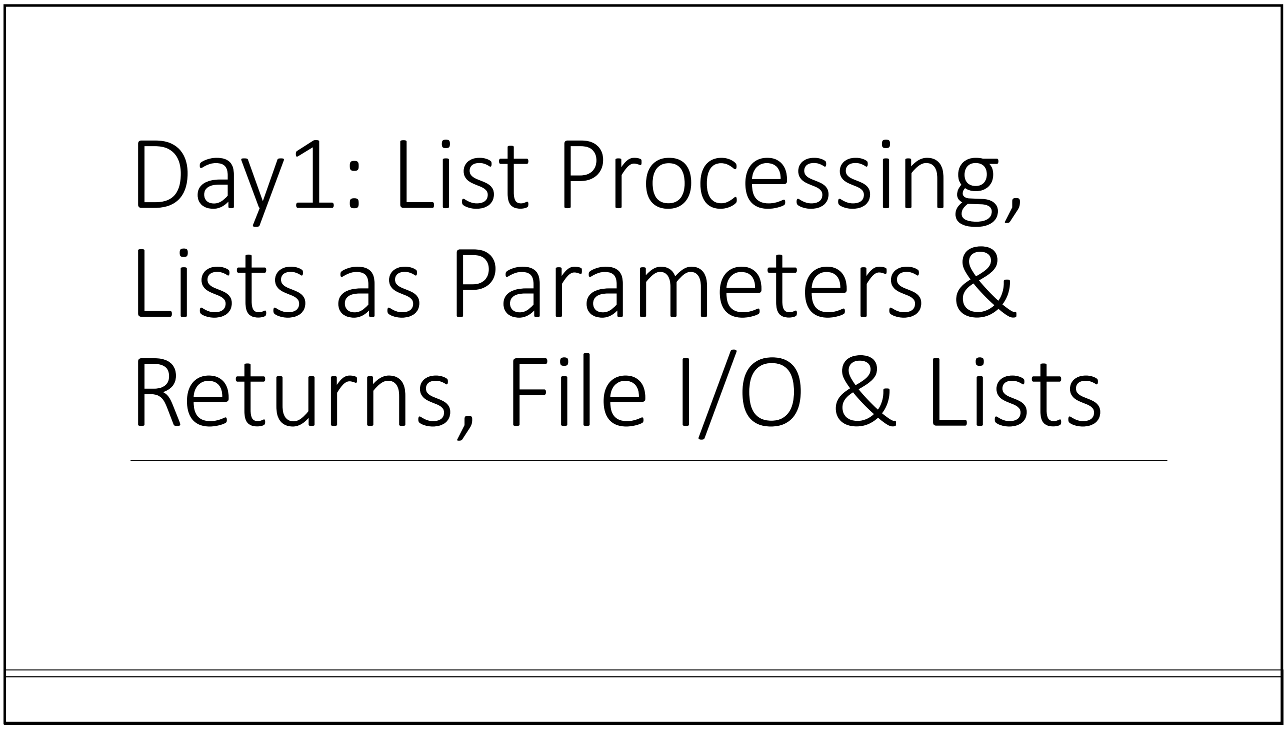
Day 1: List Processing, Lists as Params & Returns, File I/O

Day 2: List Algorithms; Selection Sort Intro, Practice, and Pseudocode

Day 3: Selection Sort Code, Optional Parameters

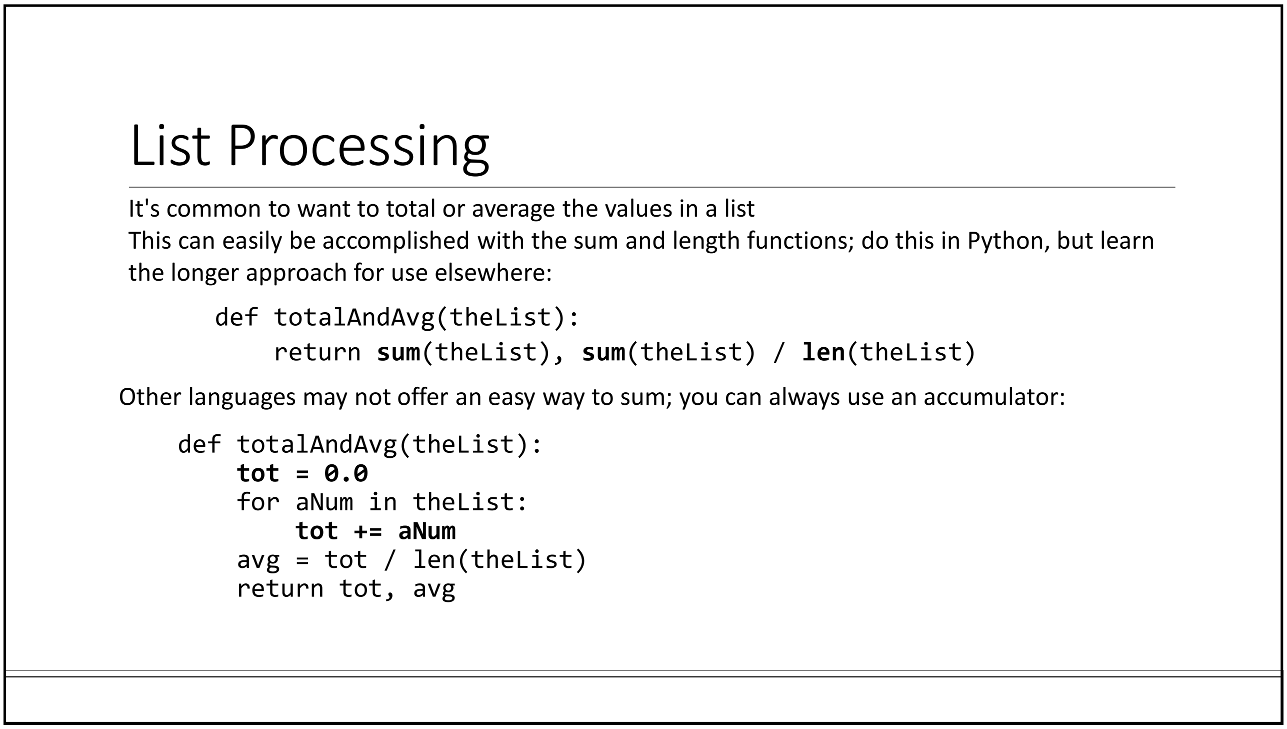
Day 4: No Class (Holiday)

3



Day1: List Processing, Lists as Parameters & Returns, File I/O & Lists

4



List Processing

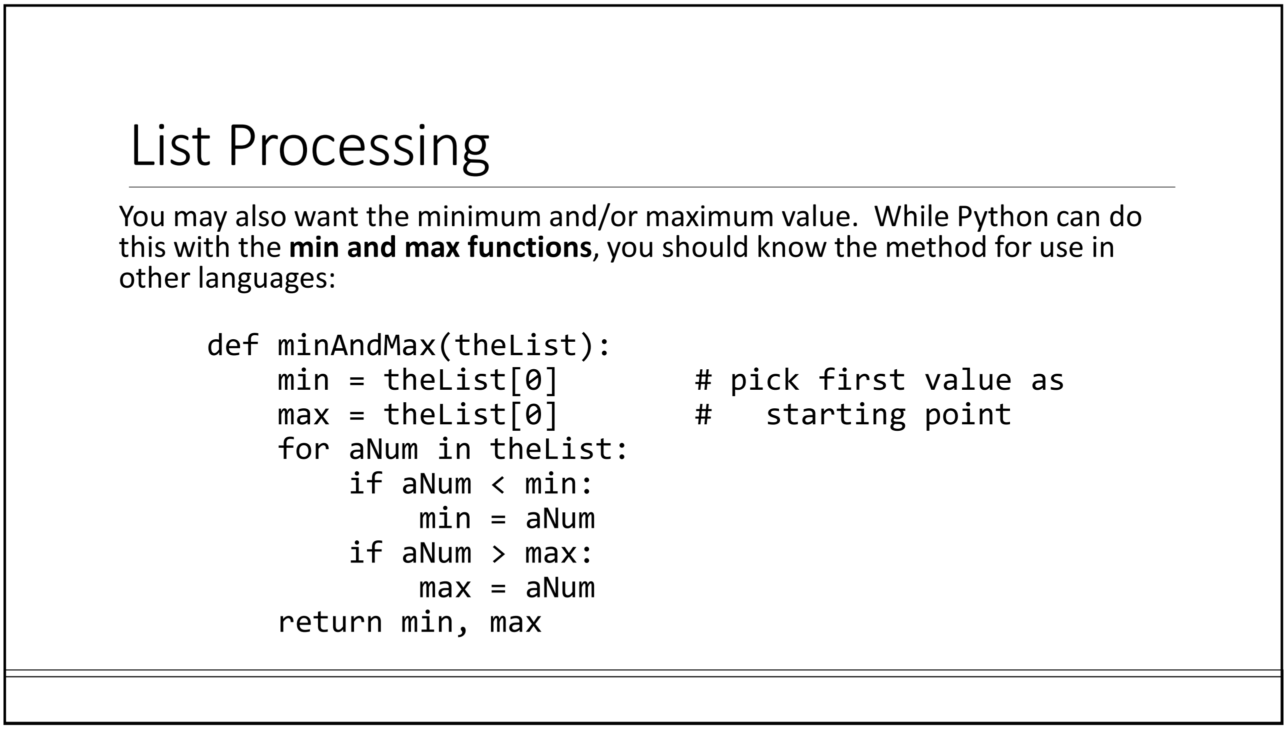
It's common to want to total or average the values in a list
This can easily be accomplished with the sum and length functions; do this in Python, but learn the longer approach for use elsewhere:

```
def totalAndAvg(theList):
    return sum(theList), sum(theList) / len(theList)
```

Other languages may not offer an easy way to sum; you can always use an accumulator:

```
def totalAndAvg(theList):
    tot = 0.0
    for aNum in theList:
        tot += aNum
    avg = tot / len(theList)
    return tot, avg
```

5

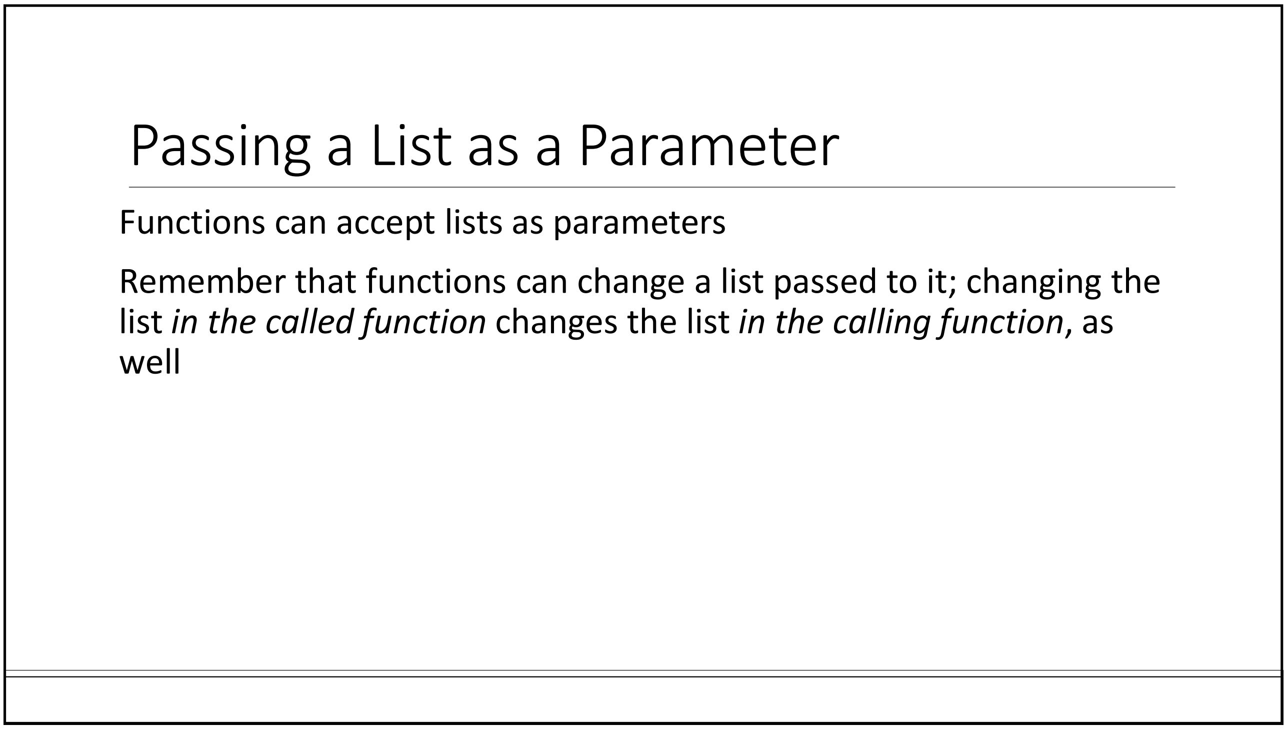


List Processing

You may also want the minimum and/or maximum value. While Python can do this with the **min and max functions**, you should know the method for use in other languages:

```
def minAndMax(theList):
    min = theList[0]        # pick first value as
    max = theList[0]        #   starting point
    for aNum in theList:
        if aNum < min:
            min = aNum
        if aNum > max:
            max = aNum
    return min, max
```

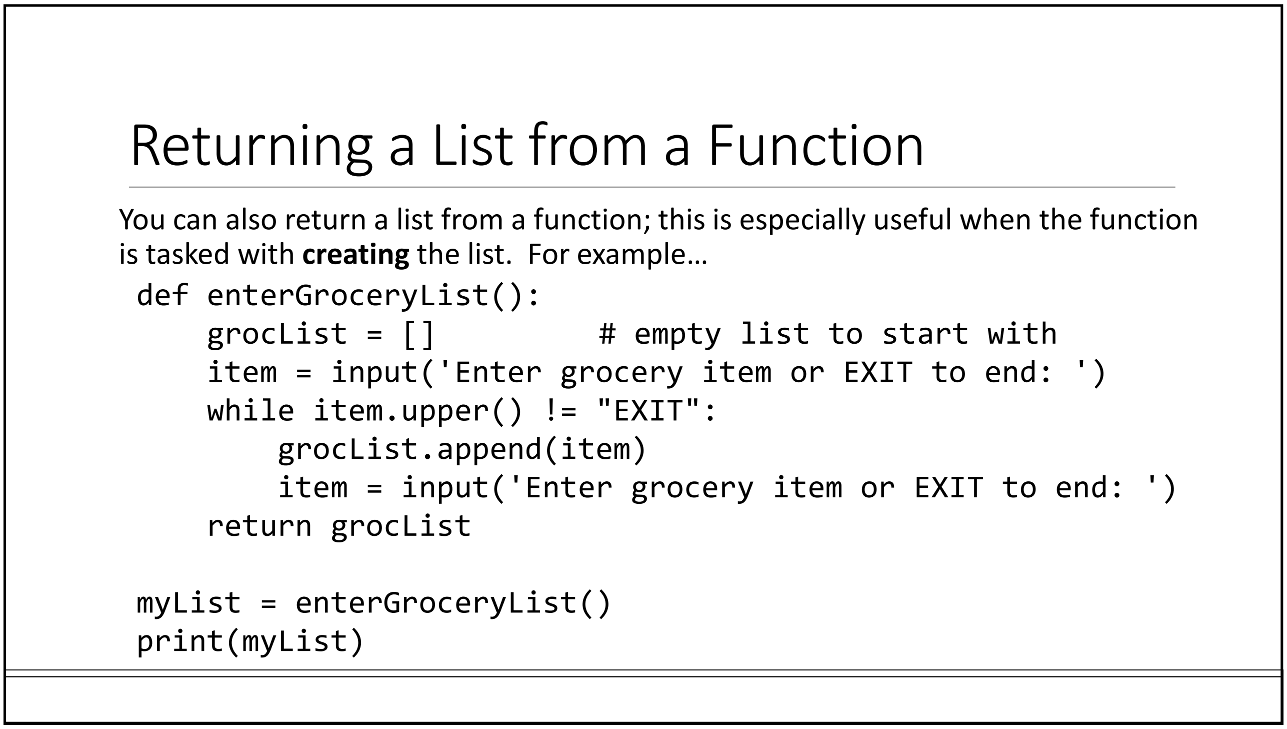
6



Passing a List as a Parameter

Functions can accept lists as parameters

Remember that functions can change a list passed to it; changing the list *in the called function* changes the list *in the calling function*, as well

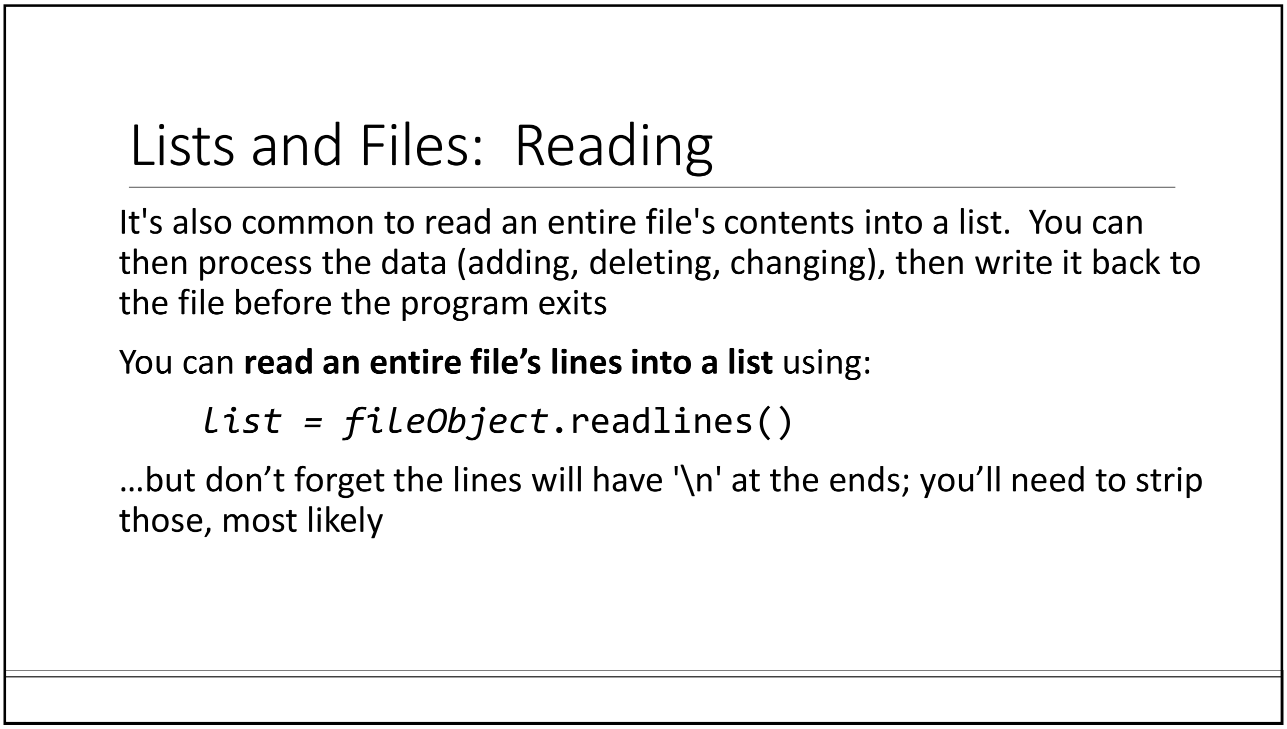


Returning a List from a Function

You can also return a list from a function; this is especially useful when the function is tasked with **creating** the list. For example…

```
def enterGroceryList():
    grocList = []         # empty list to start with
    item = input('Enter grocery item or EXIT to end: ')
    while item.upper() != "EXIT":
        grocList.append(item)
        item = input('Enter grocery item or EXIT to end: ')
    return grocList

myList = enterGroceryList()
print(myList)
```



Lists and Files: Reading

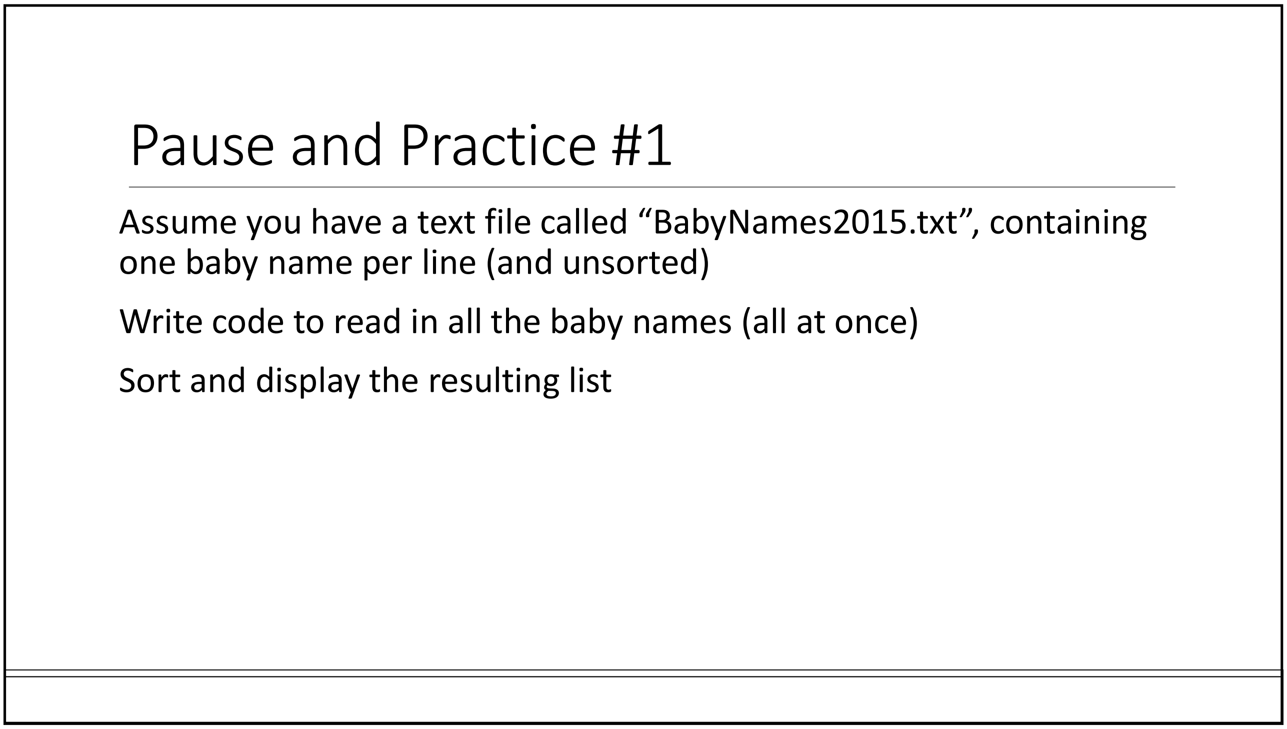
It's also common to read an entire file's contents into a list. You can then process the data (adding, deleting, changing), then write it back to the file before the program exits

You can **read an entire file's lines into a list** using:

```
    list = fileObject.readlines()
```

…but don't forget the lines will have '\n' at the ends; you'll need to strip those, most likely

9



Pause and Practice #1

Assume you have a text file called "BabyNames2015.txt", containing one baby name per line (and unsorted)

Write code to read in all the baby names (all at once)

Sort and display the resulting list

10

Pause and Practice #1: Code

```
# Open file, read in names
babyNameFile = open('BabyNames2015.txt', 'r')
babyNameList = babyNameFile.readlines()
babyNameFile.close()

# Strip off \n characters
for index in range(len(babyNameList)):
    babyNameList[index] = babyNameList[index].strip()

# Sort and display the list
babyNameList.sort()
print(babyNameList)
```

Lists and Files: Writing

You can **write a list into a file** using:

```
    fileObject.writeLines(list)
```

…but this won't automatically add '\n' at the end of each line

One alternative is to loop through the list, writing individual elements and appending '\n' to each element as you go

Pause and Practice #2

Continuing the last project…

Add code to write all the sorted names into this text file: “BabyNames2015-Sorted.txt”

Make sure they appear one name per line

13

Pause and Practice #2: Code (bad)

```
# NOT what we want!
sortedFile = open('BabyNames2015-Sorted.txt', 'w')
sortedFile.writelines(babyNameList)
sortedFile.close()
```

File Contents

```
AlexisAmariAngelArielArmaniAveryAzariahBlakeBriarCameron…
```

Alternative: we could edit each name and concatenate a '\n', but that’s not ideal, either, as it alters otherwise usable data. But let’s do it another way…

14

Pause and Practice #2: Code (better)

```
# Write to new file
sortedFile = open('BabyNames2015-Sorted.txt', 'w')
for name in babyNameList:
    sortedFile.write(name + '\n')
sortedFile.close()
```

15

Pause and Practice #2: Code (best)

Another “Pythonic” option: use `join` to create a string by joining list elements (consider it the opposite of `split`):

```
    # Write to new file
    sortedFile = open('BabyNames2015-Sorted.txt', 'w’)
    namesWithNewLines = '\n'.join(babyNameList)
    sortedFile.write(namesWithNewLines)
    sortedFile.close()
```

Could also condense to:

```
    sortedFile = open('BabyNames2015-Sorted.txt', 'w’)
    sortedFile.write('\n'.join(babyNameList))
    sortedFile.close()
```

16

Day2: List Algorithms; Selection Sort Intro, Practice, and Pseudocode

17

Algorithms

An algorithm is "a set of steps that are followed in order to solve a mathematical problem or to complete a computer process"

There are many classic algorithms for typical programming challenges, e.g., sorting lists; programmers need to know these, and leverage these in solutions

One easy-to-understand/code sorting algorithm is the **Selection Sort**

It's also a good time to introduce Big O Notation, a way to classify algorithms by how they respond to changes in list size

Links to check out: Wikipedia definition, Big O cheat sheet, Big O analysis PDF

18

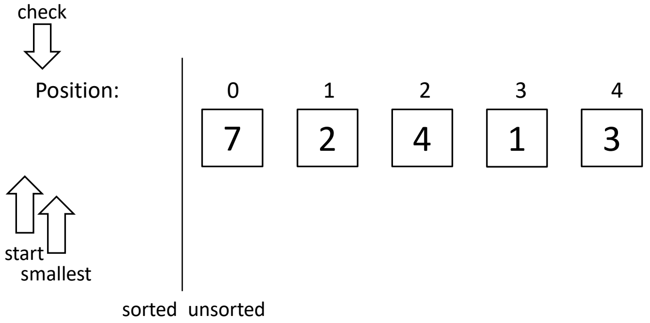
Selection Sort

This is O(n^2); for a list of 5 items the inner loop runs ~25 times, best case, worst case, and average!

The idea:

- For a list of size *n*, make *n-1* passes through the list. In each pass, locate the smallest number in the list and swap it with the element at the start of the unsorted portion of the list.

Here's how it works:



Watch YouTube video for a demonstration

19

Selection Sort Pseudocode

Let’s write pseudocode for the Selection Sort

We’ll later convert it into working Python code

20

Selection Sort Pseudocode

```
pass:  loop through item indexes 0 to list length - 2
    set smallest item position to starting pass position
    check:  loop through item indexes pass + 1 to list length - 1
        if item at check index is less than item at smallest index
            set smallest item position to check position
    swap values at start of pass position and smallest position
return nothing (list is updated in place)
```

21

Day3: Selection Sort Code, Optional Parameters

22

Selection Sort in Code

Writing Selection Sort

- Do you see that we'll need a nested loop?
- The **outer** loop will manage the Passes; the **inner** loop will manage the Checks

If ambitious, write this yourself before watching the video or copying code

Hints:

- You'll almost certainly get it wrong before you get it right
- “Off-by-one” errors are common, leaving the last item unsorted or indexing past the end
- Use the debugger (especially locals) to help solve problems
- “Poor man's debugging” is a bunch of strategically placed print() statements

23

Selection Sort Code

```
def selectionSort(aList):
    for passesIdx in range(len(aList)-1):
        smallIndex = passesIdx
        # Find the smallest value and remember its index
        for checkIndex in range(passesIdx + 1, len(aList)):
            if aList[checkIndex] < aList[smallIndex]:
                smallIndex = checkIndex
        # Swap first unsorted item with the smallest item
        temp = aList[passesIdx]
        aList[passesIdx] = aList[smallIndex]
        aList[smallIndex] = temp
```

24

Pause and Practice #3

Copy and paste your selectionSort function to create selectionSortReverse, which will sort lists, but in *reverse* order

Yes, we can do this, but is that the best design? Why or why not?

25

SelectionSortReverse: Code

```
def selectionSortReverse(aList):
    for passesIdx in range(len(aList) - 1):
        largestIndex = passesIdx
        # find the largest value and remember its index
        for checkIndex in range(passesIdx + 1, len(aList)):
            if aList[checkIndex] > aList[largestIndex]:
                largestIndex = checkIndex
        # swap first unsorted item with the largest item
        temp = aList[passesIdx]
        aList[passesIdx] = aList[largestIndex]
        aList[largestIndex] = temp
```

26

Optional Parameters

Let’s learn a new trick: defining functions with optional parameters

Optional parameters must come at the *end* of the parameter list

We can set its default value (the value Python will assign if the caller does not provide that parameter):

Syntax: def *function*(*reqParams*, *optParam* = *defValue*):

Notes:

- You can have more than one optional parameter
- Python looks at the default value when the function is *defined*, not when it’s *run*, so you can get some strange behavior

27

Optional Parameter Example

```
def writeLogEntry(comment = ''):
    import datetime
    logFile = open('log.txt', 'a')
    logFile.write('{:30}{}\n'.format( \
                  str(datetime.datetime.today()), comment))
    logFile.close()

writeLogEntry()
writeLogEntry('Maintenance performed')
```

Log.txt Contents:

```
2017-06-08 07:14:02.586320
2017-06-08 07:14:02.586320    Maintenance performed
```

28

Pause and Practice #4

Copy and paste your selectionSort function to create selectionSortSmart, which takes an optional Boolean parameter indicating whether the list should be sorted in reverse order; the default should be forward ordering

Then prove that it works using all three possibilities:

```
selectionSortSmart(list)         # should be forward order
selectionSortSmart(list, True)   # should be reverse order
selectionSortSmart(list, False)  # should be forward order
```

29

SelectionSortSmart: Code (dup code)

```
def selectionSortSmart(aList, reverse = False):
    for passesIdx in range(len(aList)-1):
        minMaxIdx = passesIdx
        # find the smallest value and remember its index
        for checkIndex in range(passesIdx + 1, len(aList)):
            if not reverse and aList[checkIndex] < aList[minMaxIdx]:
                minMaxIdx = checkIndex
            elif reverse and aList[checkIndex] > aList[minMaxIdx]:
                minMaxIdx = checkIndex
        # swap first unsorted item with the identified item
        temp = aList[passesIdx]
        aList[passesIdx] = aList[minMaxIdx]
        aList[minMaxIdx] = temp
```

30

SelectionSortSmart: Code (less dup)

```
def selectionSortSmart(aList, reverse = False):
    for passesIdx in range(len(aList) - 1):
        minMaxIdx = passesIdx
        # find the smallest value and remember its index
        for checkIndex in range(passesIdx + 1, len(aList)):
            if (not reverse and aList[checkIndex] < aList[minMaxIdx] ) \
               or  (reverse and aList[checkIndex] > aList[minMaxIdx] ):
                minMaxIdx = checkIndex
        # swap first unsorted item with the identified item
        temp = aList[passesIdx]
        aList[passesIdx] = aList[minMaxIdx]
        aList[minMaxIdx] = temp
```

31

Day4: No Class (Holiday)

32

The End

33