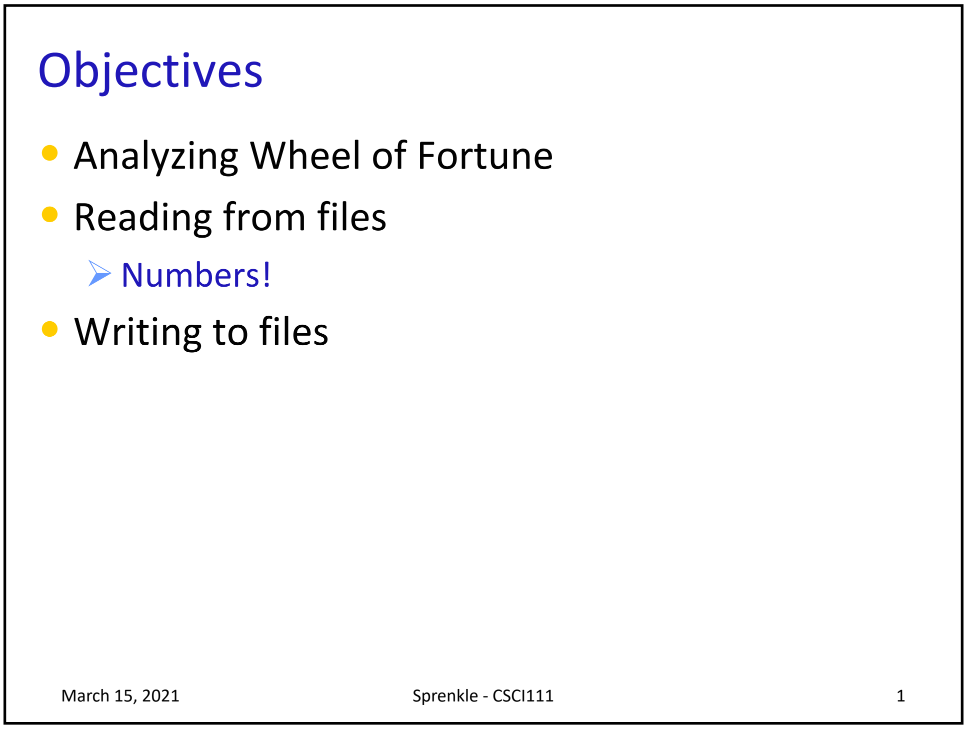
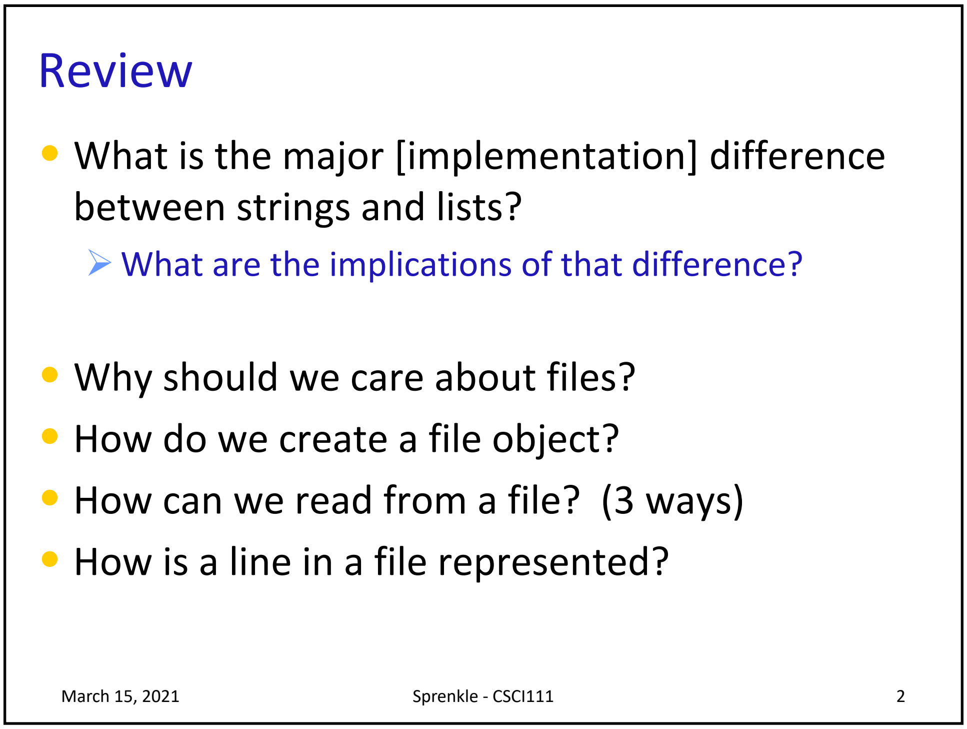


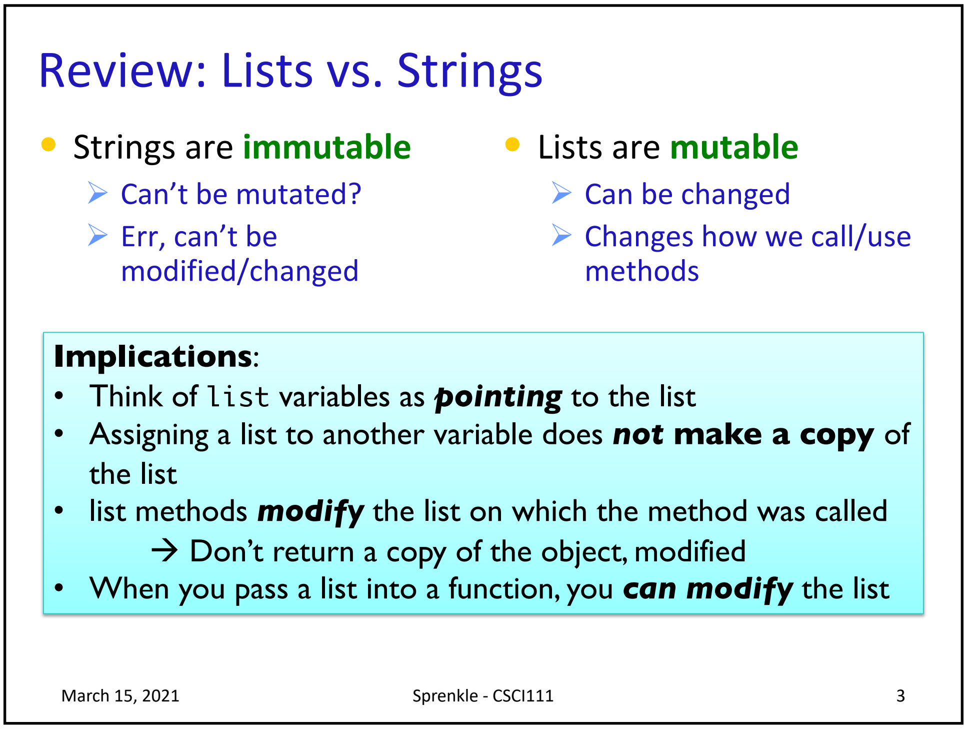
Objectives

- Analyzing Wheel of Fortune
- Reading from files
  - Numbers!
- Writing to files



Review

- What is the major [implementation] difference between strings and lists?
  - What are the implications of that difference?
- Why should we care about files?
- How do we create a file object?
- How can we read from a file? (3 ways)
- How is a line in a file represented?

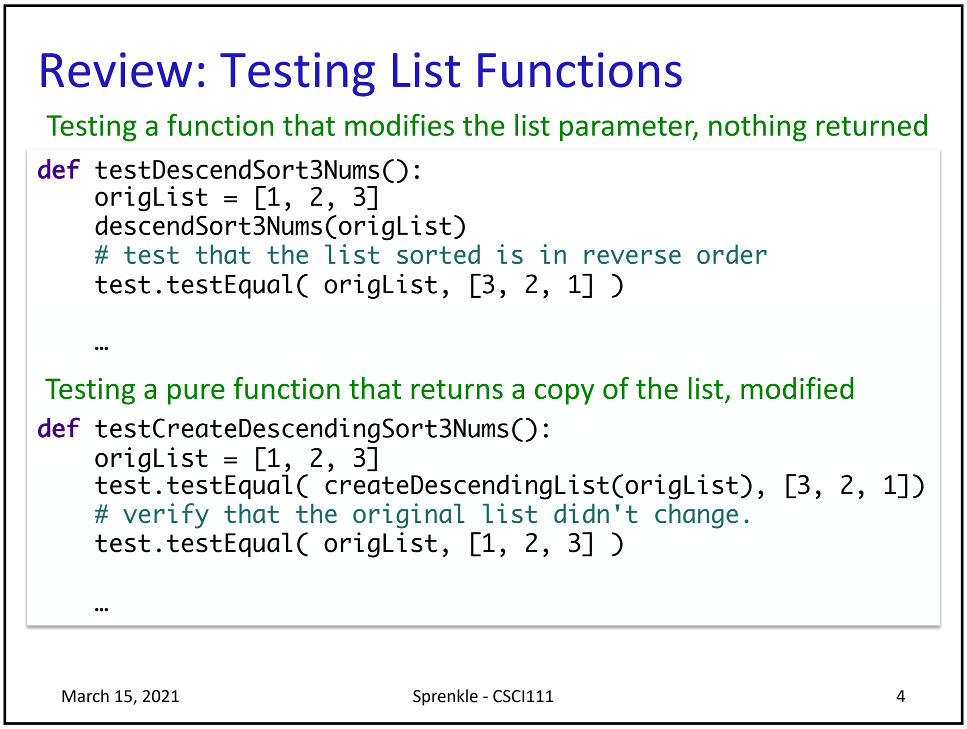


Review: Lists vs. Strings

- Strings are **immutable**
  - Can't be mutated?
  - Err, can't be modified/changed
- Lists are **mutable**
  - Can be changed
  - Changes how we call/use methods

Implications:

- Think of `list` variables as ***pointing*** to the list
- Assigning a list to another variable does ***not make a copy*** of the list
- list methods ***modify*** the list on which the method was called
  → Don't return a copy of the object, modified
- When you pass a list into a function, you ***can modify*** the list



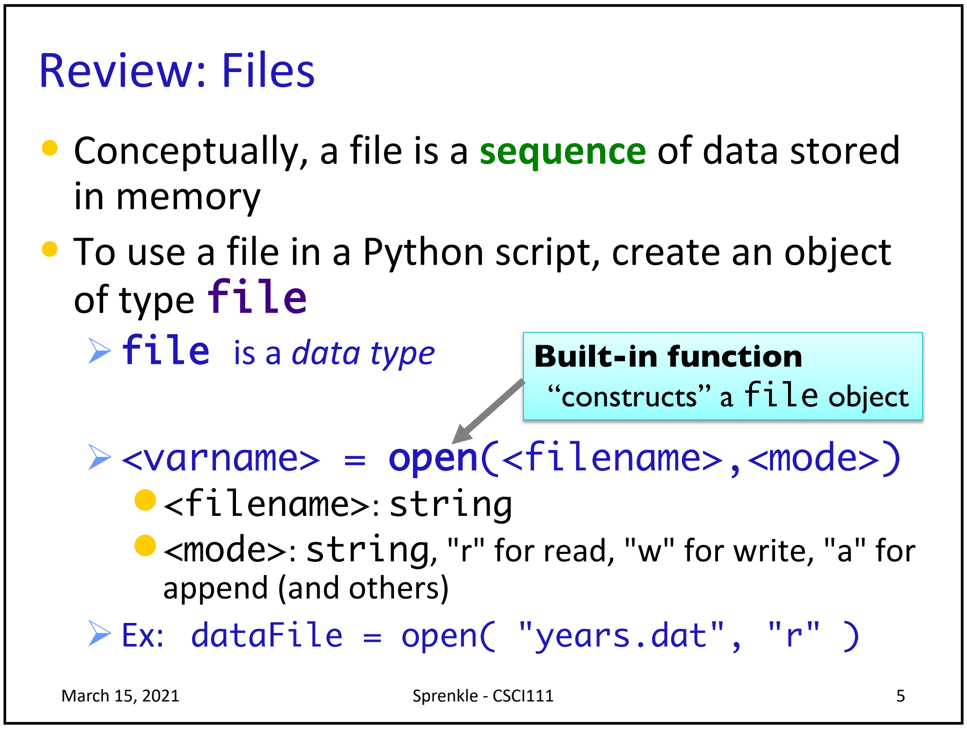
Review: Testing List Functions

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

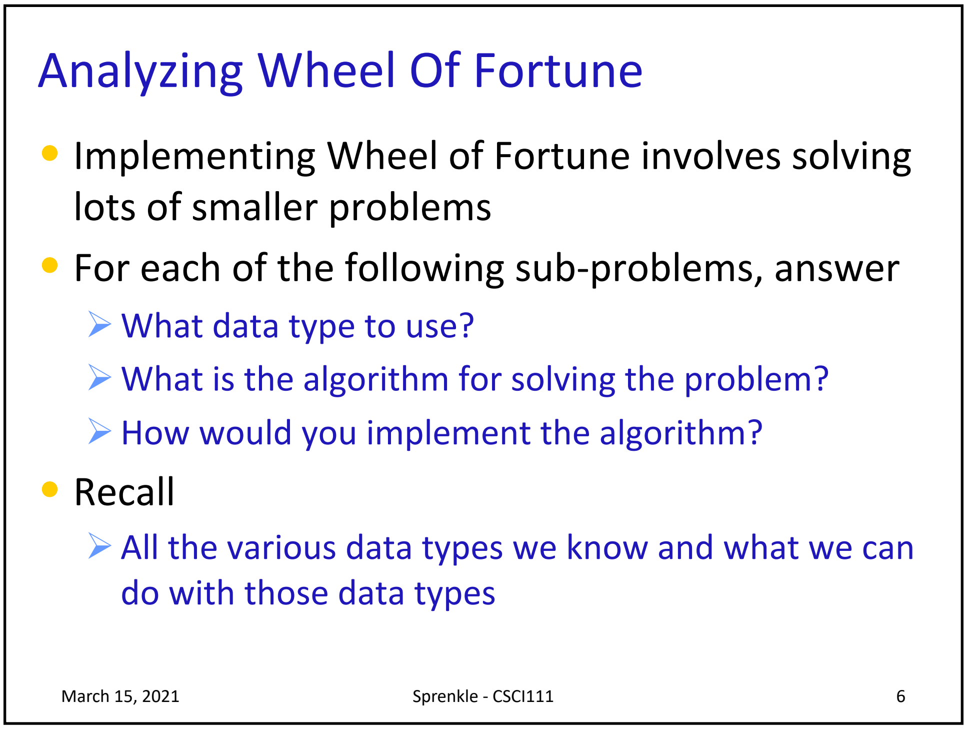
```
def testCreateDescendingSort3Nums():
    origList = [1, 2, 3]
    test.testEqual( createDescendingList(origList), [3, 2, 1])
    # verify that the original list didn't change.
    test.testEqual( origList, [1, 2, 3] )

    …
```



Review: Files

- Conceptually, a file is a **sequence** of data stored in memory
- To use a file in a Python script, create an object of type `file`
  - `file` is a *data type*
  - `<varname> = open(<filename>,<mode>)`
    - `<filename>`: `string`
    - `<mode>`: `string`, "r" for read, "w" for write, "a" for append (and others)
  - Ex: `dataFile = open( "years.dat", "r" )`

Built-in function
“constructs” a `file` object



Analyzing Wheel Of Fortune

- Implementing Wheel of Fortune involves solving lots of smaller problems
- For each of the following sub-problems, answer
  - What data type to use?
  - What is the algorithm for solving the problem?
  - How would you implement the algorithm?
- Recall
  - All the various data types we know and what we can do with those data types

Analyzing Wheel Of Fortune: Puzzle Categories

- We want the user to pick from several different categories of puzzles
  - The puzzles are in text files named for the category, e.g., `data/whatareyoudoing.txt`

```
Press 1 for whatareyoudoing
Press 2 for beforeandafter
Press 3 for famous_pairs
Which category do you choose? 1
```

- How do you solve this problem?
  - Goal: Make easy to add/remove a category

Analyzing Wheel Of Fortune: Puzzles

- The puzzles are in text files named for the category, e.g., `data/whatareyoudoing.txt`
- Each puzzle is on its own line in the file, e.g.,

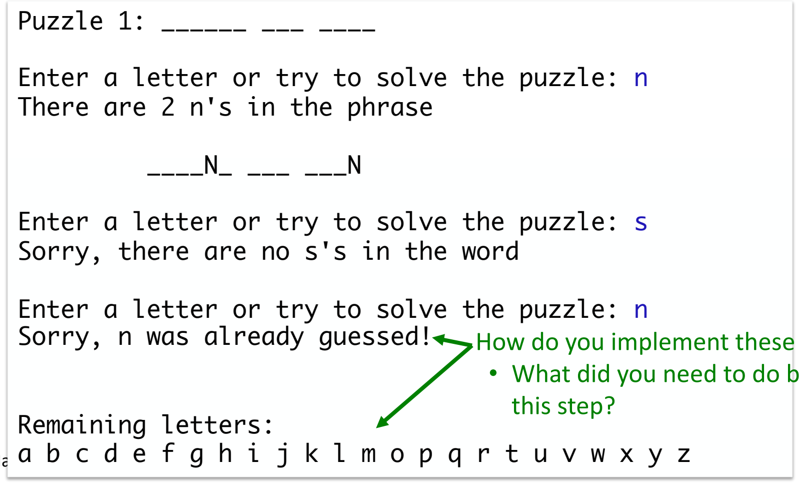
```
Mowing the lawn
Crossing the street
Teaching code to a fourth grader
Oversharing on social media
…
```

- How do you model/implement retrieving/displaying a puzzle?

```
Puzzle 1: ______ ___ ____
```

Analyzing Wheel of Fortune: Remaining Letters

- We need to keep track of which letters are still available to be guessed



```
Puzzle 1: ______ ___ ____

Enter a letter or try to solve the puzzle: n
There are 2 n's in the phrase

         ____N_ ___ ___N

Enter a letter or try to solve the puzzle: s
Sorry, there are no s's in the word

Enter a letter or try to solve the puzzle: n
Sorry, n was already guessed!


Remaining letters:
a b c d e f g h i j k l m o p q r t u v w x y z
```

How do you implement these steps?

- What did you need to do before this step?

Writing to a File

- Create a file object in **write** mode:
  - ➢ `myFile = open("myfile.txt", "w")`

- Example: create a file from user input
  - ➢ `file_write.py`

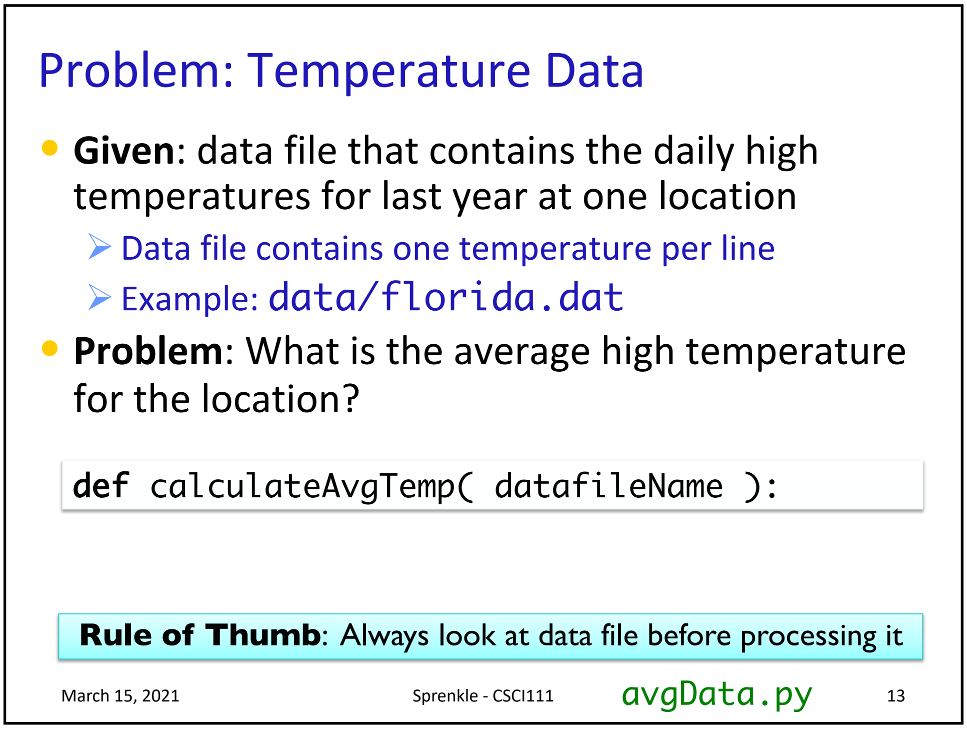
What happens if you execute the program again with different user input?

Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
- How can we **write numbers** to a file?

Handling Numeric Data

- We have been dealing with reading and writing *strings* so far
  - Read from a file: get a string
  - Write to file: use a string
- What do we need to do to **read numbers** from a file?
  - Cast as a numeric type, e.g., `int` or `float`
- How can we **write numbers** to a file?
  - Cast number as a `str` or use `format` method



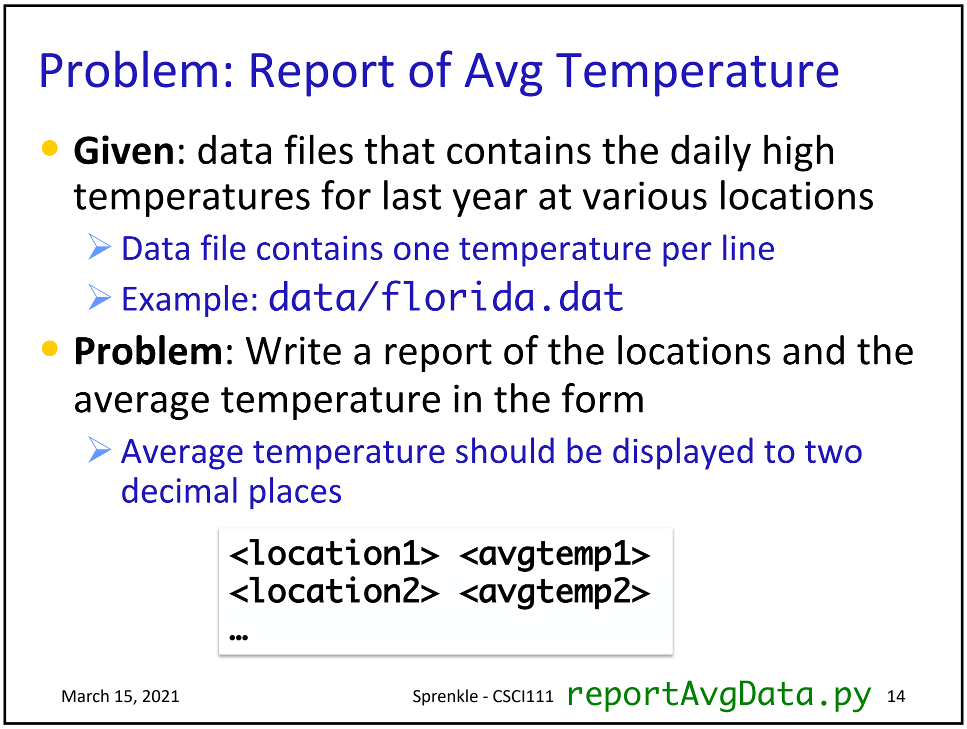
Problem: Temperature Data

- **Given**: data file that contains the daily high temperatures for last year at one location
  - Data file contains one temperature per line
  - Example: `data/florida.dat`
- **Problem**: What is the average high temperature for the location?

```
def calculateAvgTemp( datafileName ):
```

Rule of Thumb: Always look at data file before processing it

avgData.py

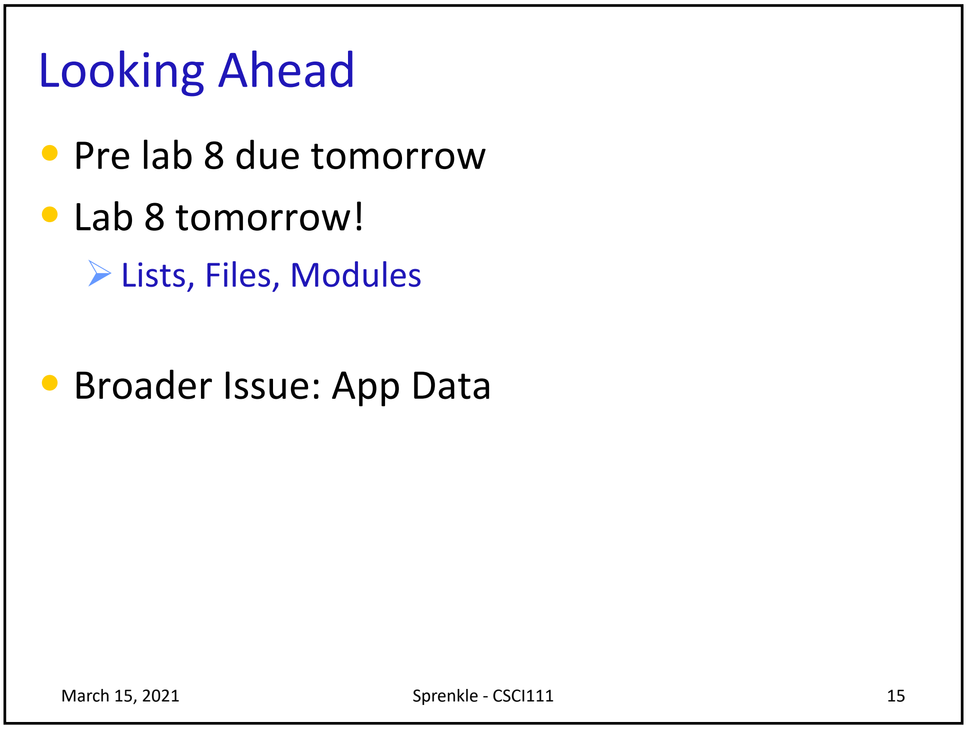


Problem: Report of Avg Temperature

- **Given**: data files that contains the daily high temperatures for last year at various locations
  - Data file contains one temperature per line
  - Example: `data/florida.dat`
- **Problem**: Write a report of the locations and the average temperature in the form
  - Average temperature should be displayed to two decimal places

```
<location1> <avgtemp1>
<location2> <avgtemp2>
…
```

reportAvgData.py



Looking Ahead

- Pre lab 8 due tomorrow
- Lab 8 tomorrow!
  - Lists, Files, Modules
- Broader Issue: App Data

15