Objectives

- Introduction to Files
- 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?
- What is a “pure” function?

Summary: Lists vs. Strings

- Strings are **immutable**
  - Can't be changed after created
- Lists are **mutable**
  - Can be changed

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

  → They do *not* return a copy of the object, modified
- When you pass a list into a function, you ***can modify*** the list

Review: 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
```

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] )
```

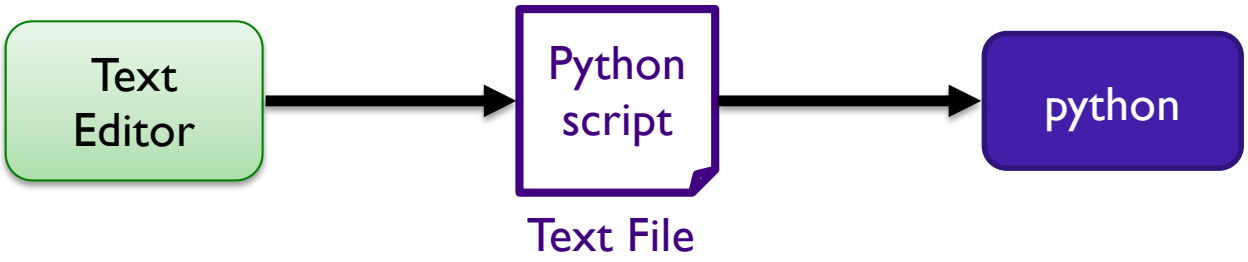
FILES

Sources of Input to Program: User Input

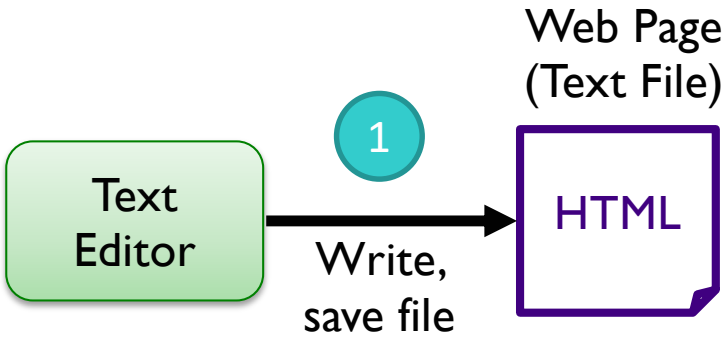
- Pros
  - Easy!
  - Intuitive!
- Cons
  - Slow if need to enter a lot of data
  - Error-prone
    - User enters the wrong value!
  - What if want to run again after program gets modified?

Sources of Input to Program: Text Files

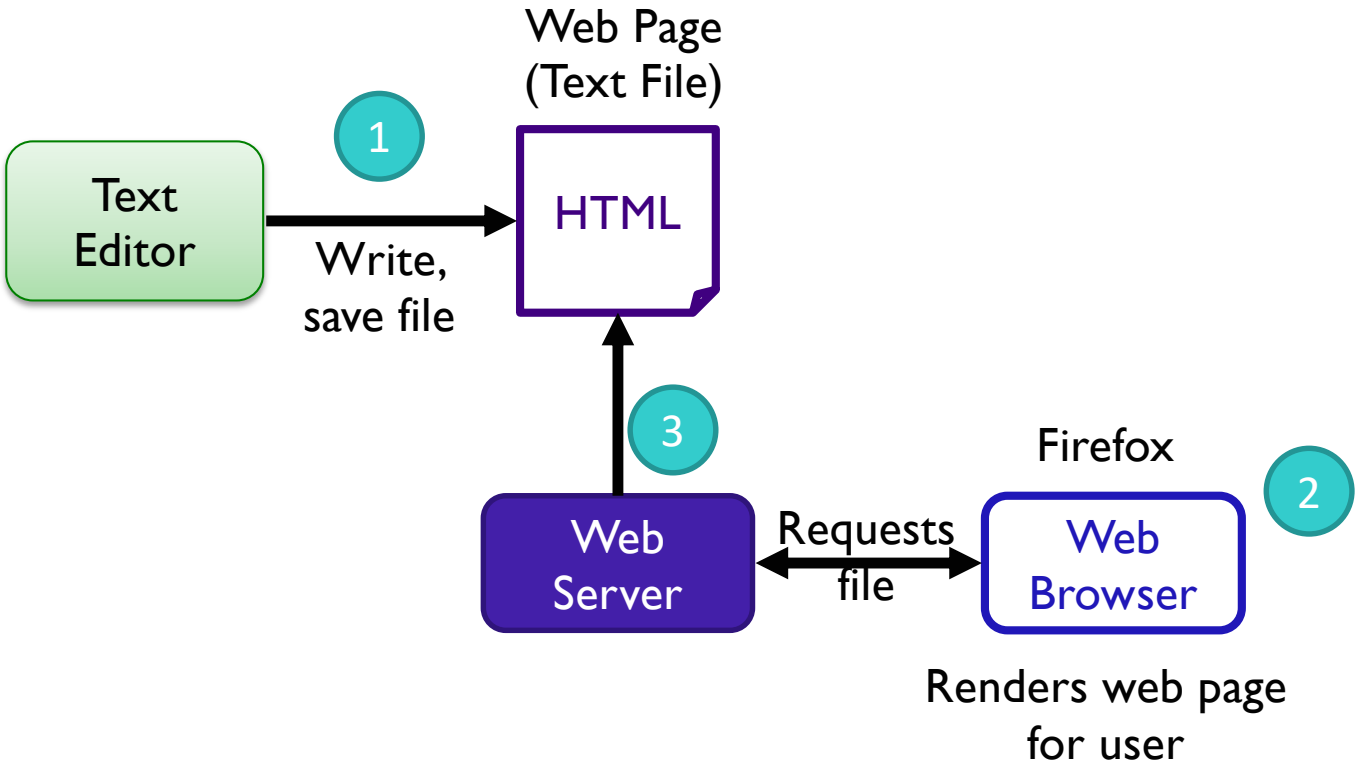
- Pros
  - Enter data once into a file, save it, and reuse it
  - Good for large amounts of data
  - Programs can use files to *communicate*
  - Need to be able to *read from* and *write to* files
- Cons
  - Not as intuitive in programming
  - Requires creating a file



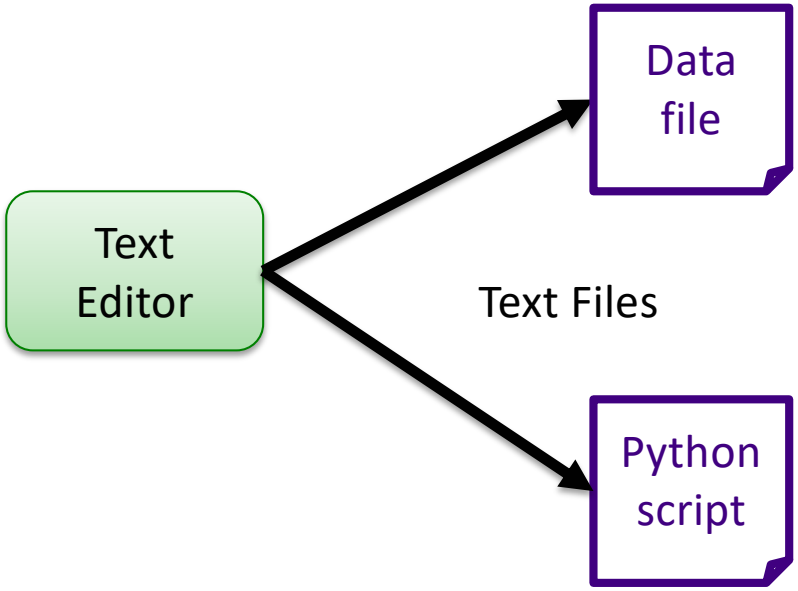
Example Use of Files: on the Web



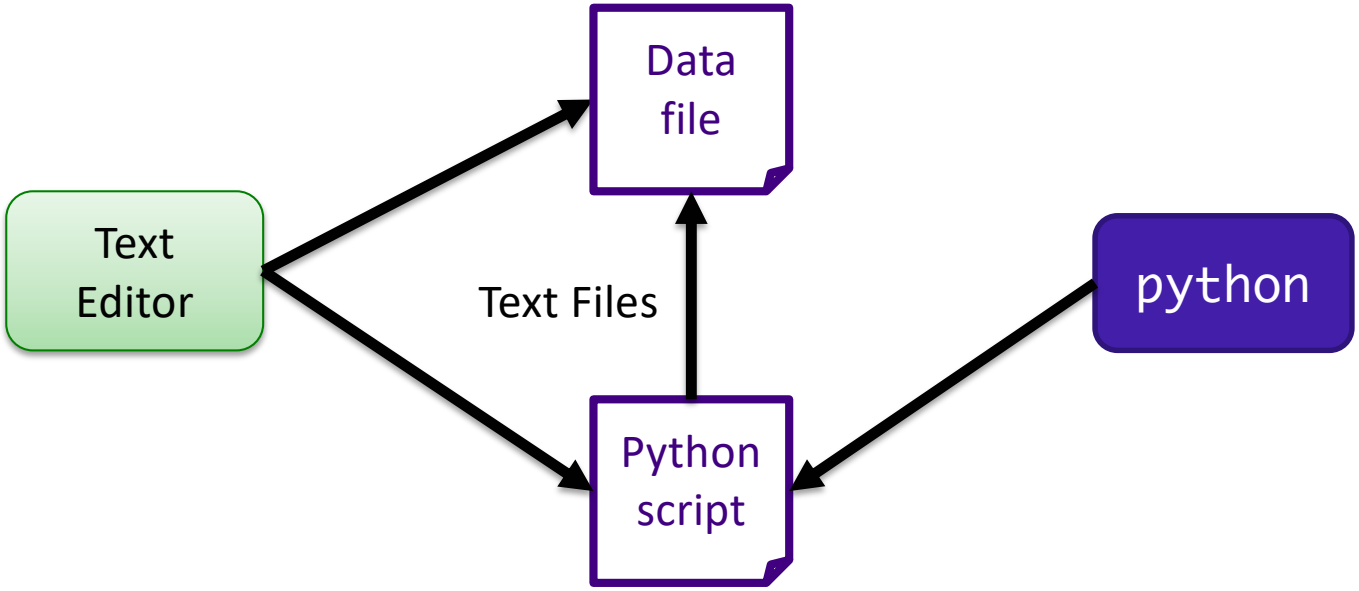
Example Use of Files: on the Web



Example Use of Text File as Input: Data!



Example Use of Text File as Input: Data!



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( "temps.dat", "r" )`

Built-in function

"constructs" a file object

Common File Methods

Method Name	Functionality
`read()`	Read all the content from the file, returned as a string object
`readline()`	Read the *next* line from file, returned as a string object (which includes the "\n"). If it returns "", then you've reached the end of the file
`write(string)`	Write a string to the file
`close()`	Close the file. Must close the file after done reading from/writing to a file

Reading from a File

- Examples of reading from a file using file methods
  - Example: `data/famous_pairs.txt`
- `file_read.py` (using `read`())
  - How is what Python printed different than the file's content?
  - How to fix?
- Using `readline`()

Typically use `.dat` or `.txt` file extension to name files containing data or text

Note directory organization

`file_read.py`
`using_readline.py`

In the Python Interpreter

```
>>> filename = "data/famous_pairs.txt"
>>> myfile = open(filename, "r")
>>> contents = myfile.read()
>>> contents
'Romeo & Juliet\nPeanut Butter & Jelly\nOrville & Wilbur
Wright\nMeriwether Lewis & William Clark\nSonny & Cher\nWhifield
Diffie & Martin Hellman\nBarbie & Ken\n'
>>> print(contents)
Romeo & Juliet
Peanut Butter & Jelly
Orville & Wilbur Wright
Meriwether Lewis & William Clark
Sonny & Cher
Whifield Diffie & Martin Hellman
Barbie & Ken

>>>
```

In the Python Interpreter

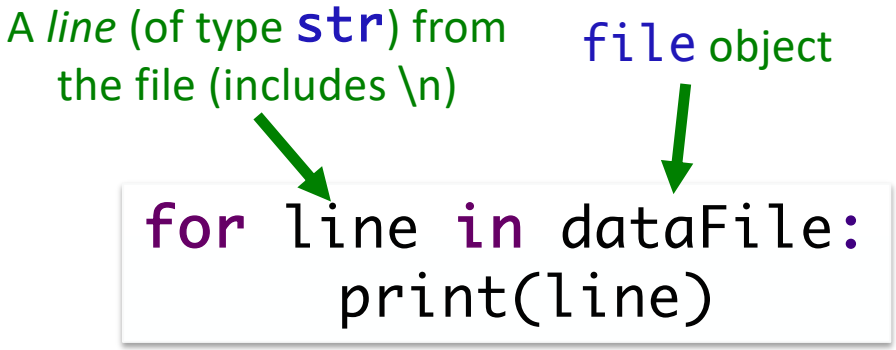
```
>>> filename = "data/famous_pairs.txt"
>>> myfile = open(filename, "r")
>>> myline = myfile.readline()
>>> myline
'Romeo & Juliet\n'
>>> print(myline)
Romeo & Juliet

>>> contents = myfile.read()
>>> contents
'Peanut Butter & Jelly\nOrville & Wilbur Wright\nMeriwether Lewis &
William Clark\nSonny & Cher\nWhifield Diffie & Martin
Hellman\nBarbie & Ken\n'
>>>
```

Nuance: Clarify what the `read()` method does

Reading from a File

- Recall that a file is a ***sequence*** of data
- Can use a `for` loop to iterate through a file



➢ Read as: for each line in the file, do …

Data Types of Loop Variables

What are the data types of the loop variable **x**?
What does **x** represent?

```
myString = "some string"
dataFile = open("datafile.dat", "r")

for x in range(len(myString)):
        # loop body …

for x in myString:
        # loop body …

for x in dataFile:
        # loop body …
```

Data Types of Loop Variables

What are the data types of the loop variable **x**?

```
myString = "some string"
dataFile = open("datafile.dat", "r")

for x in range(len(myString)):
        # loop body …

for x in myString:
        # loop body …

for x in dataFile:
        # loop body …
```

integer

string → single characters

string → line (include \n)

Writing to a File

- Create a file object in **write** mode:
  - `myFile = open("demo.txt", "w")`
- Call write method on file object:
  - `myFile.write("Write string to file")`
  - `myFile.write("Also this string")`
- Close the file:
  - `myFile.close()`

What will demo.txt contain after executing program?
After executing the program a second time?

Writing to a File

- Create a file object in **write** mode:
  - `myFile = open("demo.txt", "w")`
- Call write method on file object:
  - `myFile.write("Write string to file")`
  - `myFile.write("Also this string")`
- Close the file:
  - `myFile.close()`

Good template for working with files:

1. Open file
2. Process file
3. Close file

Wheel of Fortune

- (OK, maybe more like hangman because there is no wheel)
- Uses a file of puzzles
  - Can modify puzzle file – add lots more puzzles!

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`

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

Searching a File

Do on your own

- Display which lines and how many lines a search term is in a file
- Example output:

```
dog is found in data/wikipedia.txt on lines:
4 The dog or domestic dog (Canis familiaris[4][5] or Canis lupus familiaris[5]) is a
6 The dog is derived from an ancient, extinct wolf,[6][7] and the modern wolf is the
7 dog's nearest living relative.[8] The dog was the first species to be
11 Due to their long association with humans, dogs have expanded to a large number of
13 that would be inadequate for other canids.[11] Over the millennia, dogs became
17 The dog has been selectively bred over millennia for various behaviors, sensory
for a total of 6 lines
```

`file_search.py`

Looking Ahead

- Pre lab 8 due tomorrow
  - Long; note that some list sections are skipped
- Lab 8 tomorrow!
  - Lists, Files, Modules