

# Objectives

- Introduction to Files
- Reading from files
  - Numbers!
- Writing to files

March 13, 2023

Sprenkle - CSCI111

1

1

# Review

- What is the major [implementation] difference between strings and lists?
  - What are the implications of that difference?

March 13, 2023

Sprenkle - CSCI111

2

2

# 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

March 13, 2023

Sprenkle - CSCI111

3

3

# 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
```

March 13, 2023

Sprenkle - CSCI111

4

4

# 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] )
```

March 13, 2023

Sprenkle - CSCI111

5

5

# FILES

March 13, 2023

Sprenkle - CSCI111

6

6

# 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?

March 13, 2023

Sprenkle - CSCI111

7

7

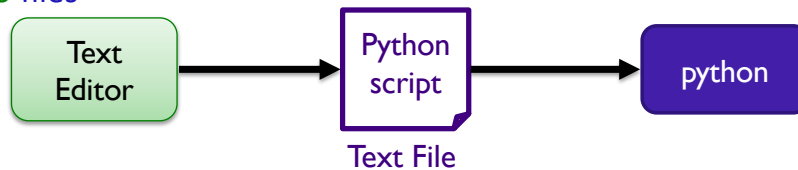
# Sources of Input to Program: Text Files

- 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



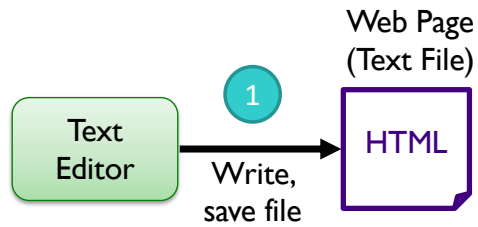
March 13, 2023

Sprenkle - CSCI111

8

8

# Example Use of Files: on the Web



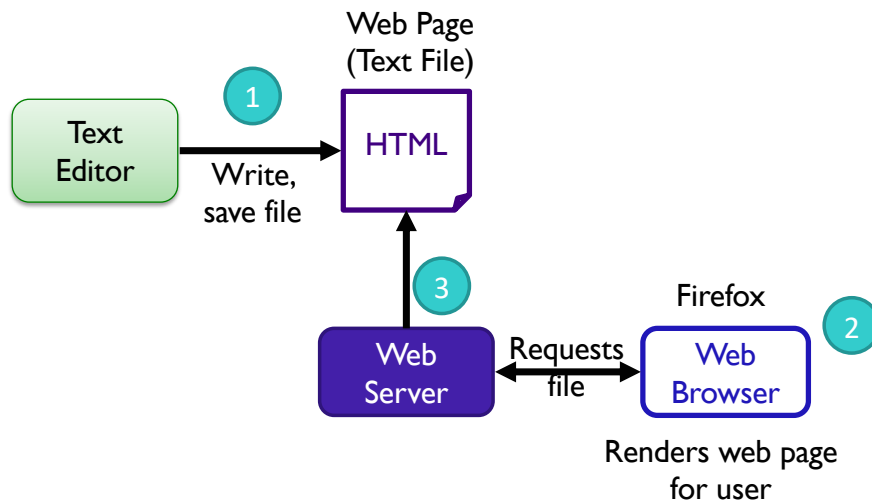
March 13, 2023

Sprenkle - CSCI111

9

9

# Example Use of Files: on the Web



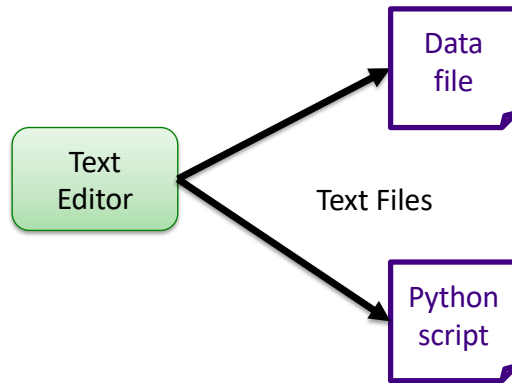
March 13, 2023

Sprenkle - CSCI111

10

10

## Example Use of Text File as Input: Data!



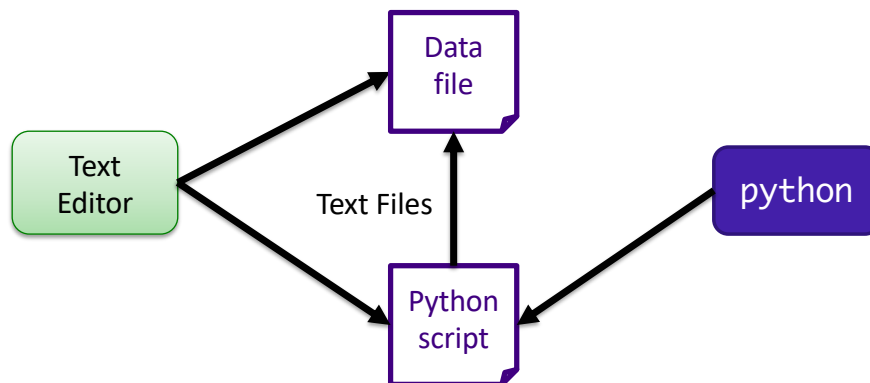
March 13, 2023

Sprenkle - CSCI111

11

11

## Example Use of Text File as Input: Data!



March 13, 2023

Sprenkle - CSCI111

12

12

# 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>)`
    - **open** is a **Built-in function** "constructs" a file object
    - `<filename>`: string
    - `<mode>`: string, "r" for read, "w" for write, "a" for append (and others)
  - Ex: `dataFile = open( "temps.dat", "r" )`

March 13, 2023

Sprenkle - CSCI111

13

13

# Common File Methods

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

March 13, 2023

Sprenkle - CSCI111

14

14

## Reading from a File

- Examples of reading from a file using file methods

➤ Example: `data/famous_pairs.txt`

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

- `file_read.py` (using `read()`)

➤ How is what Python printed different than the file's content?

➤ How to fix?

- Using `readline()`

`file_read.py`  
`using_readline.py`

March 13, 2023

Sprenkle - CSCI111

15

15

## Reading from a File

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

A line (of type `str`) from the file (includes `\n`)

file object

```
for line in dataFile:  
    print(line)
```

➤ Read as: for each line in the file, do ...

March 13, 2023

Sprenkle - CSCI111

`for_file_read.py`

16

16



# 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 ...
```

March 13, 2023

Sprenkle - CSCI111

17

17

# 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)): integer
    # loop body ...

for x in myString:           string → single
    # loop body ...         characters

for x in dataFile:          string → line
    # loop body ...         (include \n)
```

March 13, 2023

Sprenkle - CSCI111

18

18

## 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?

March 13, 2023

Sprenkle - CSCI111

demoWrite.py

19

19

## 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!

March 13, 2023

Sprenkle - CSCI111

20

20

## 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?

March 13, 2023

Sprenkle - CSCI111

21

21

## 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**

March 13, 2023

Sprenkle - CSCI111

22

22

## 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

March 13, 2023

Sprenkle - CSCI111

`avgData.py`

23

23

## Looking Ahead

- Pre lab 8 due tomorrow
- Lab 8 tomorrow!
  - Lists, Files, Modules

March 13, 2023

Sprenkle - CSCI111

24

24