File Input/Output

CS 8: Introduction to Computer Science, Spring 2019
Lecture #12

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- Homework #6 – will be posted tonight: due next Tuesday
- Lab05 − due on Sunday by midnight (11:59 pm) on **Gradescope**!

- **Project Lab description is now up!**
  - Project counts as 2 lab grades
  - Due at the end of the quarter (June 2nd)

- Midterm Exam #2 is on **May 23rd**
  - **More information/prep material will be forthcoming on Piazza**

- There will **NOT** be a lecture on **Thursday, May 16th**

Reviewing Your Midterm #1 Exam

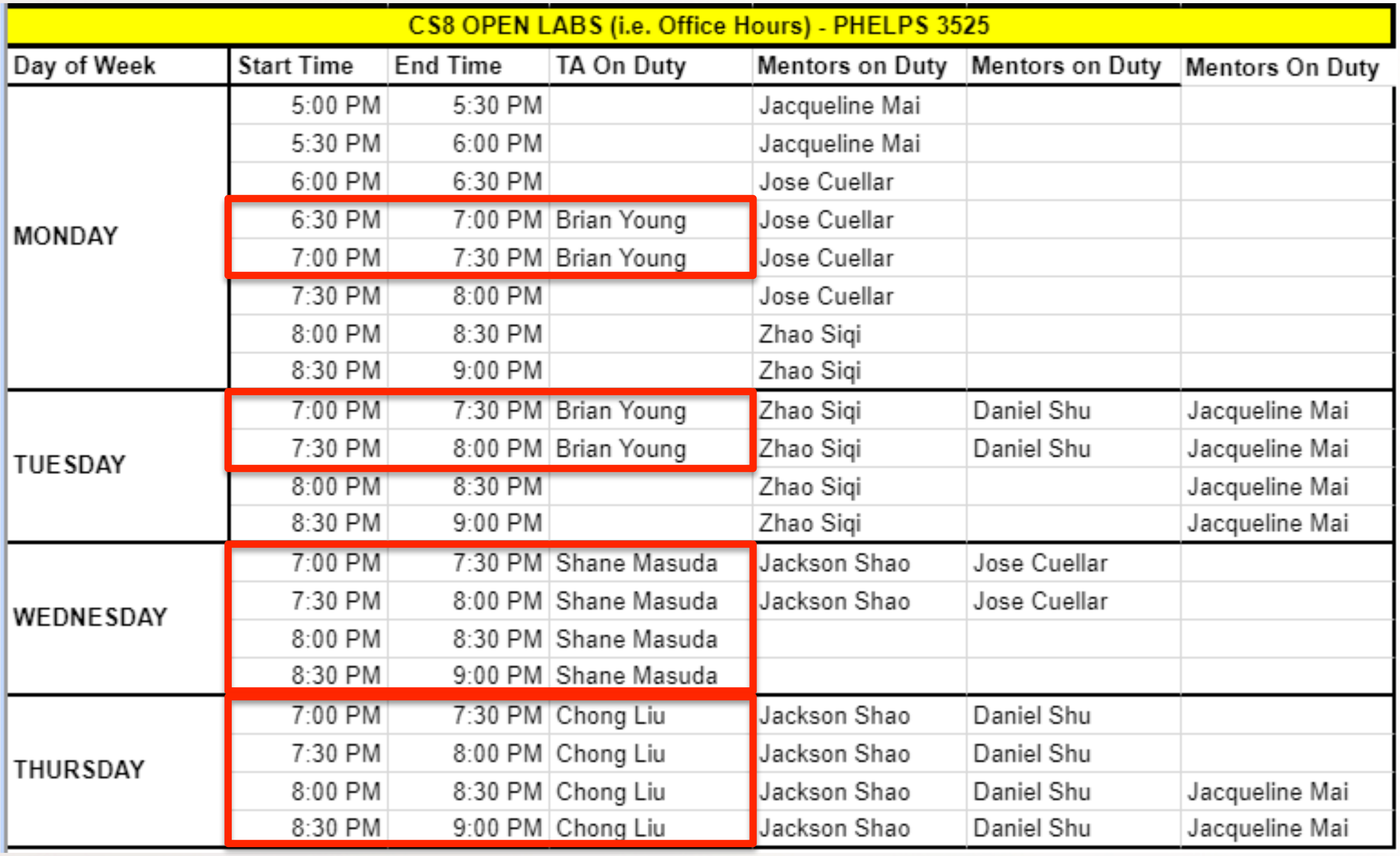
- Optional, but recommended for you to understand your mistakes

- If you're in the **8 AM** lab – go to **Chong Liu's** office hours
- If you're in the **9 AM** lab – go to **Brian Young's** office hours
- If you're in the **10 AM** lab – go to **Shane Masuda's** office hours
- If you're in the **11 AM** lab – go to **Prof. Matni's** office hours

When Reviewing Your Exams (IMPORTANT!)

- Do **not** take pictures, do **not** copy the questions
- You can **only** view the exam during office hours
- You **cannot** take the exam with you
- TA cannot change your grade
  - If you have a legitimate case for grade change, the prof. will decide
  - **Legitimate case** = When we graded, we added the total points wrong
  - **Not legitimate case** =

    “Why did you take off *N* points on this question????”



CS8 OPEN LABS (i.e. Office Hours) - PHELPS 3525

Day of Week	Start Time	End Time	TA On Duty	Mentors on Duty	Mentors on Duty	Mentors On Duty
MONDAY	5:00 PM	5:30 PM		Jacqueline Mai		
	5:30 PM	6:00 PM		Jacqueline Mai		
	6:00 PM	6:30 PM		Jose Cuellar		
	6:30 PM	7:00 PM	Brian Young	Jose Cuellar		
	7:00 PM	7:30 PM	Brian Young	Jose Cuellar		
	7:30 PM	8:00 PM		Jose Cuellar		
	8:00 PM	8:30 PM		Zhao Siqi		
	8:30 PM	9:00 PM		Zhao Siqi		
TUESDAY	7:00 PM	7:30 PM	Brian Young	Zhao Siqi	Daniel Shu	Jacqueline Mai
	7:30 PM	8:00 PM	Brian Young	Zhao Siqi	Daniel Shu	Jacqueline Mai
	8:00 PM	8:30 PM		Zhao Siqi		Jacqueline Mai
	8:30 PM	9:00 PM		Zhao Siqi		Jacqueline Mai
WEDNESDAY	7:00 PM	7:30 PM	Shane Masuda	Jackson Shao	Jose Cuellar	
	7:30 PM	8:00 PM	Shane Masuda	Jackson Shao	Jose Cuellar	
	8:00 PM	8:30 PM	Shane Masuda			
	8:30 PM	9:00 PM	Shane Masuda			
THURSDAY	7:00 PM	7:30 PM	Chong Liu	Jackson Shao	Daniel Shu	
	7:30 PM	8:00 PM	Chong Liu	Jackson Shao	Daniel Shu	
	8:00 PM	8:30 PM	Chong Liu	Jackson Shao	Daniel Shu	Jacqueline Mai
	8:30 PM	9:00 PM	Chong Liu	Jackson Shao	Daniel Shu	Jacqueline Mai

Lecture Outline

- Quick review of random numbers, others

- File Input / Output in Python

Random Numbers

- "Pseudo-random" values can be generated using special functions in most programming languages

- In Python use functions of the **random module**
  - You have to ***import random*** first

- Simplest way to make a random number: **random.random()**
  - Returns a floating point value between 0.0 and 1.0

Random Numbers

- Also: **randrange(n)**, **randint(low, high)** and many others
  - **randrange(n)** returns int random number between 0 and n-1
  - **randint(low, high)** returns int random number between low and high (*inclusive*)

- Try typing **help(random)** in IDLE to learn more…
  - And play around with it

One More Note on **namedtuple()**

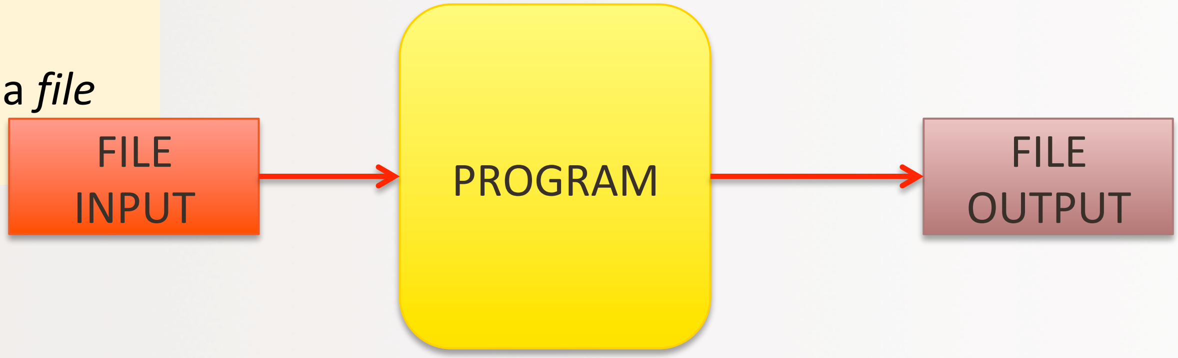
- Since tuples are **immutable**,
  you cannot change parts of them once they are defined
  - You can only **re-assign** the whole thing
- For example:

```
…
Mything = Item(item1 = 42, item2 = 99)
print(Mything.item1)      # prints 42
Mything.item1 = 0         # DOES NOT WORK!!! ☹
Mything = Item(item1 = 0, item2 = 99)    # WORKS! ☺☺
Mything = Item(item1 = 0) # DOES NOT WORK! ☹
```

Instead of getting data from a *standard* input (i.e. keyboard) and presenting data to a *standard* output…

vs.

We can get data from a *file* input and present data to a *file* output…



Files

- Mostly handled like any *sequential data type*
- What's some examples of data types that can be read sequentially?

- Files are **a sequence of characters** *if they are text files*,

  or **a sequence of bits** *if they are binary file*

- Can you name some common file *types* that are textual?

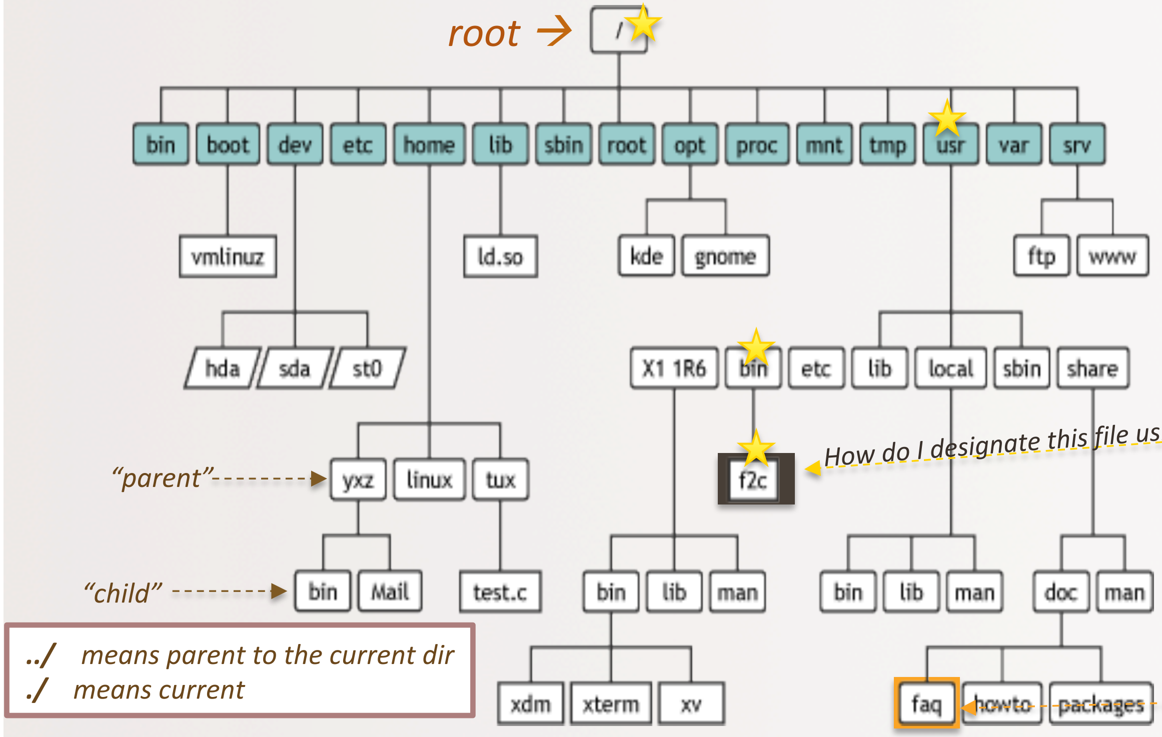
  Or that are binary?

Why Use Files?

4 Good Reasons:

1. Files allow you to store data **permanently** and **conveniently**!
2. Data output that goes to a file stays there **after the program ends**
   - You can usually view the data without the need of a Python program
3. An input data file **can be used over and over again**
   - No need to type data again and again for testing
4. Files allow you to deal with larger data sets
   - Imagine putting all historical weather data for the USA in one list or string!!! 😐

Recall: Organization of Files in a Computer



How do I designate this file using the full directory "path"?

/usr/bin/f2c.exe

/usr/share/doc/faq.txt

../ means parent to the current dir
./ means current

Is done **hierarchically**
Uses **folders** (aka directories)
Starts at the "**root**" directory
designated with a **/**

File I/O: Simple Examples

Example of READING from a file

```
infile = open('DataFile.txt', 'r')

line = infile.read()
# read everything in one string!
# Yes: there are other ways…

print(line)

infile.close()
# DON'T FORGET TO CLOSE!!!
```

Example of WRITING to a file

```
outfile = open('MyOuts.txt', 'w')

x = 3
y = 4
n = (x + y)**y

outfile.write('Number' + str(n))

outfile.close()
# DON'T FORGET TO CLOSE!!!
```

What you write in a file HAS to be a **string** type

Different Ways of Reading File Input

```
line = infile.read()
                      # Read everything into 1 string
line = infile.read(n)
                      # Read the first n chars into 1 string
line = infile.readline()
                      # Read 1 line (ends in '\n') into 1 string
line = infile.readlines()
                      # Read all lines into 1 list
```

DEMO!
Let’s try it!

File I/O: More Examples

Example of READING from a file

```
filename = input
("What is the name of the file to
open? ")

InFile = open(filename, 'r')

count = 0
for line in InFile:
    count += 1
    print(line)
print("There are", count, "lines in
the file", filename)

InFile.close()
```

Example of WRITING to a file

```
filename = input
("What is the name of the file to
open? ")

OutFile = open(filename, 'w')

newl = '\n'
for n in range(10):
    OutFile.write
    ('Number' + str(n) + newl)

OutFile.close()
```

Read File

Example of READING from a file

```
filename = input
("What is the name of the file to open? ")

InFile = open(filename, 'r')

count = 0
for line in InFile:
     count += 1
     print(line)
print("There are ", count, " lines in the
file ", filename)

InFile.close()
```

***open()** function, using the **'r'** option means that we want to READ this file. Note that **filename** is a string.*

*This is what we're doing to the lines that we read from the file. Note that the use of the **print()** function here means that the output goes to **"standard output"** (i.e. your screen)*

*Always **close()** the file after opening it!*

Alternative instruction: `InFile = open(filename, 'r', encoding='utf-8')`

Write File

Example of WRITING to a file

```
filename = input
("What is the name of the file to
open? ")

OutFile = open(filename, 'w')

for n in range(10):
    myFile.write('Number ‘ +
str(n))

OutFile.close()
```

***open()** function, using the **‘w’** option means that we want to WRITE to this file. Note that **filename** is a string.*

*This is the data that we’re creating to put into the file. Note that the use of the **write()** function here means that the output goes to **“file output”** (not “standard output”)*
NOTE: ENTRIES HAVE TO BE STRING DATA TYPES!!!

*Always **close()** the file after opening it!*

To Reset Reading a File

- To go back to the start of a file that's being read, you can `infile.close()` and `infile.open()` again
  - Assuming **`infile`** is the object name you used for the input file…

- Another way is to use `infile.seek(0)`

Demonstration

- **Given**: An input file with information on rainfall (in inches) for various geographical locations. Looks like this:

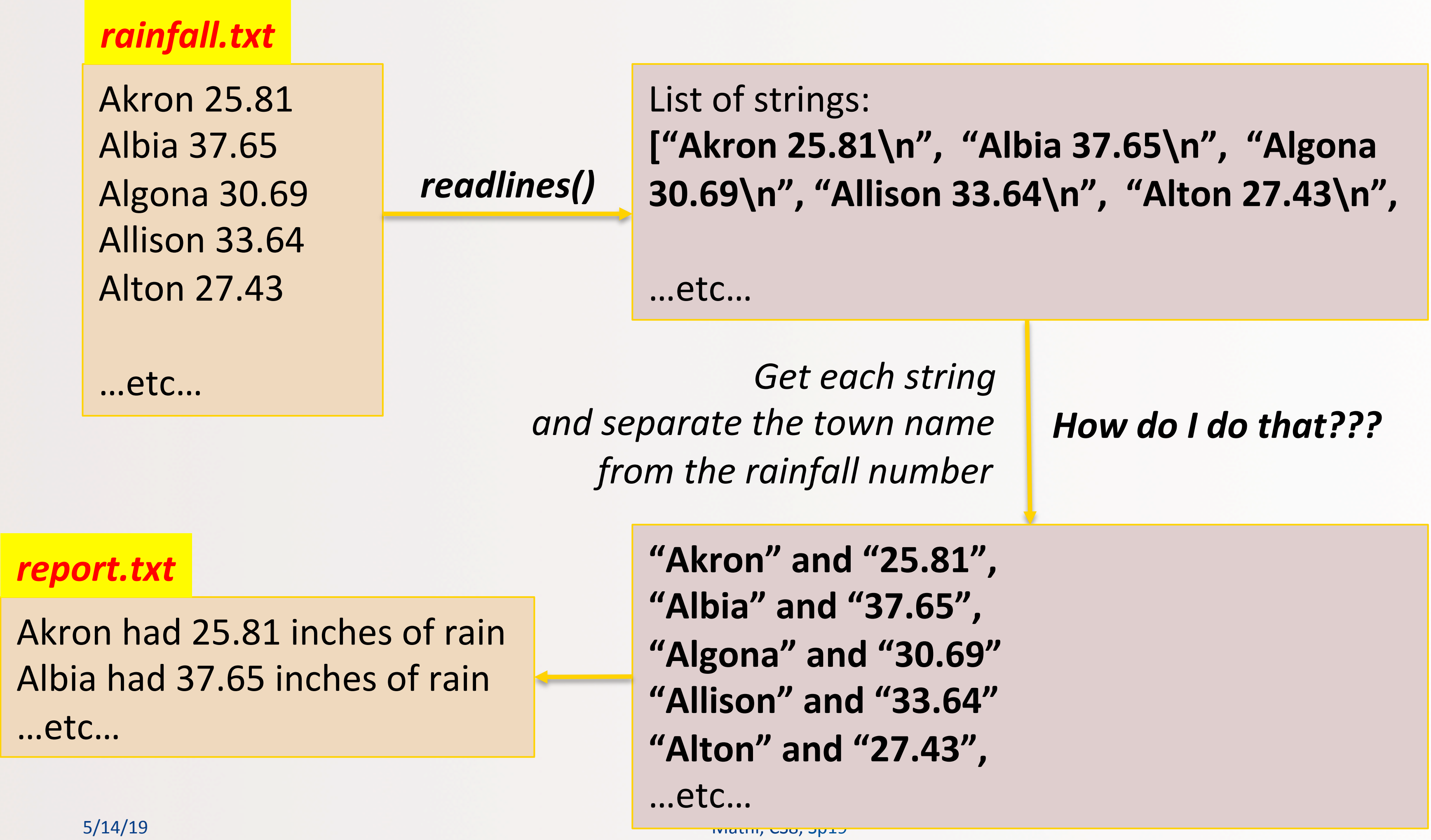
```
Akron 25.81
Albia 37.65 …etc…
```

- **You have to**: Create an output file that reads each line and outputs:

```
Akron had 25.81 inches of rain.
Albia had 37.65 inches of rain.
```

…etc…

See **rainfall.py** and **rainfall_advanced.py**



rainfall.txt

Akron 25.81
Albia 37.65
Algona 30.69
Allison 33.64
Alton 27.43

…etc…

readlines()

List of strings:
[“Akron 25.81\n”, “Albia 37.65\n”, “Algona 30.69\n”, “Allison 33.64\n”, “Alton 27.43\n”,

…etc…

Get each string
and separate the town name
from the rainfall number

How do I do that???

“Akron” and “25.81”,
“Albia” and “37.65”,
“Algona” and “30.69”
“Allison” and “33.64”
“Alton” and “27.43”,
…etc…

report.txt

Akron had 25.81 inches of rain
Albia had 37.65 inches of rain
…etc…

… To be continued next lecture…

YOUR TO-DOs

- □ **Homework** #6 due **Tuesday, 5/21**
- □ Finish **Lab5** (turn it in by **Sunday**)
- □ Remember that this Thursday (5/16), there’s NO lecture

- □ Don’t forget: we live by the beach… take advantage of it!

</LECTURE>