THE UNIVERSITY OF CHICAGO

MASTERS IN COMPUTATIONAL SOCIAL SCIENCE
THE UNIVERSITY OF CHICAGO

MACS 30111

Working with Files

Agenda/Misc

- EXAM FRIDAY!!!
  - Part 1: IN CLASS – pencil and paper
  - Part 2: Take-home and TIMED (two questions)

- **Needed file for today:**
- **names:** https://uchicago.box.com/s/kp207rd2vita1a4zz6749oe8jkvk85l6
- **Instructors.csv**: https://uchicago.box.com/s/oim7c3si6p8ju1b72nciqcdgp283ab1f
- Reflections up with deadlines

Topics:

- Basic file I/O
  - Open: load the data from disk
  - Read: manipulate the data
  - Close: print the results or write the data back to disk
- Working with tabular data using CSV files
- Working with JSON files
- Other file formats

Common Programming Pattern

Common pattern when working with data:

1. **Read** the contents of a file (or files) from disk and **load** the data into one or more data structures
2. **Manipulate** the data in some way
3. **Print** the result or **write** the data back to disk

Sample application

Given a file of email addresses (username@domain), construct a file with the corresponding user names.

`instructor-email.txt`

```
amr@cs.uchicago.edu
borja@cs.uchicago.edu
yanjingl@cs.uchicago.edu
mwachs@cs.uchicago.edu
dupont@cs.uchicago.edu
```

`instructor-email-sorted.txt`

```
["amr@cs.uchicago.edu",
 "borja@cs.uchicago.edu",
 "dupont@cs.uchicago.edu",
 "mwachs@cs.uchicago.edu",
 "yanjingl@cs.uchicago.edu"]
```

Coding practice: 4.1.1

Common Programming Pattern

Common pattern when working with data:

1. **Read the contents of a file (or files) from disk and load the data into one or more data structures**
2. **Manipulate** the data in some way
3. **Print** the result or **write** the data back to disk

Opening a file

Basic File I/O

To access the contents of a file, we first need to open it:

```
f = open("instructor-email.txt")
```

file pointer

To read data from a file, we use the read method:

```
addrs = f.read()
```

read the entire contents into a string

When we are done with a file, we need to close it:

```
f.close()
```

close the file pointer

Coding practice: 4.1.1

Alternative to *close()*

The ***with*** statement to ensure that a file is closed once we're done with it:

```
with open("instructor-email.txt") as f:
    s = f.read()
    email_addresses = sorted(s.split())
```

Coding practice: 4.1.1

Read the file one line at a time

Use a *for* loop to iterate over a text file line by line:

```
with open("instructor-email.txt") as f:
    for line in f:
        print(line)
```

extra empty line

```
with open("instructor-email.txt") as f:
    for line in f.readlines():
        print(line)
```

line.strip()

Coding practice: 4.1.1

Common Programming Pattern

Common pattern when working with data:

1. **Read** the contents of a file (or files) from disk and **load** the data into one or more data structures
2. **Manipulate** the data in some way
3. **Print the result or write the data back to disk**

Write data to a file

To write to a file, we must open the file in **write mode**.

```
with open("names.txt", "w") as f:
    f.write("Anne Rogers\n")
    f.write("Borja Sotomayor\n")
    f.write("Yanjing Li\n")
    f.write("Matthew Wachs\n")
    f.write("Todd Dupont\n")
```

We can also use ***print*** to avoid having to worry about the newline.

```
with open("names2.txt", "w") as f:
    print("Anne Rogers", file=f)
    print("Borja Sotomayor", file=f)
    print("Yanjing Li", file=f)
    print("Matthew Wachs", file=f)
    print("Todd Dupont", file=f)
```

Very important:

- Opening an existing file in write mode will **wipe all its contents**!
- Opening a file that does not exist in write mode will **create** the file.

Coding practice: 4.1.2

Summary

The common programming pattern:

1. Load the data from disk:
    a. **Open** a file to **read**
    b. Read the contents of the file from disk
    c. Load the data into a data structure
2. Manipulate the data in some way
3. Print the result or write the data back to disk
    a. **Write** the data
    b. **Close** the file (or use a with statement when you open it)

Coding practice: 4.1.2

Topics:

- Basic file I/O
  - Open: load the data from disk
  - Read: manipulate the data
  - Close: print the results or write the data back to disk
- **Working with tabular data using CSV files**
- Working with JSON files
- Other file formats

CSV (Comma Separated Values) format

CSV files are useful for storing **tabular data**: any data that can be organized into rows, each with the same columns (or "fields")

`instructors.csv`

```
id,lname,fname,email
amr,Rogers,Anne,amr@cs.uchicago.edu
borja,Sotomayor,Borja,borja@cs.uchicago.edu
yanjingl,Li,Yanjing,yanjingl@cs.uchicago.edu
mwachs,Wachs,Matthew,mwachs@cs.uchicago.edu
dupont,Dupont,Todd,dupont@cs.uchicago.edu
```

header

Applied practice!

Working with text

Exercise

- You are working on a project creating a directory of buildings based on where MACSS classes typically meet.
- Use text file: https://uchicago.box.com/s/kp207rd2vita1a4zz6749oe8jkvk85l6

- How would you load your data?
- Next, you want to select the following buildings: `sched = ["1155", "SS", "TTI", "K"]` and output the new list into a separate file

Sample application

1. Read the original data from `instructors.csv`
2. Manipulate the data by:
    a. getting field information for each row
3. Print the formatted output of the data

Coding practice: 4.1.3

Read file using *csv* module

- *csv.DictReader* - read rows from a CSV file into dictionaries
- *csv.DictWriter* - write dictionaries into rows of a CSV file

Alternatively, we could also use:

- *csv.reader* - read rows from a CSV file into a list of lists
- *csv.writer* - write lists into rows of a CSV file

Topics:

- Basic file I/O
  - Open: load the data from disk
  - Read: manipulate the data
  - Close: print the results or write the data back to disk
- Working with tabular data using CSV files
- **Working with JSON files**
- Other file formats

JSON (JavaScript Object Notation) format

JSON is a **lightweight** data-interchange / data-storage format commonly used in web services.

Supports different types:

- Object: key-value pairs separated by commas
  - Keys must be strings
  - Values must be valid JSON data types
- Array: empty list or list of objects
- Value: string, number, object, array, true, false, null

File operation using JSON module

String operation:

- *json.dumps*: encodes data into JSON format string
- *json.loads*: decodes JSON format string into a data structure

File operation:

- *json.dump*: encodes data into a JSON file
- *json.load*: decodes data from a JSON file into a data structure

Coding practice: 4.1.4

Topics:

- Basic file I/O
  - Open: load the data from disk
  - Read: manipulate the data
  - Close: print the results or write the data back to disk
- Working with tabular data using CSV files
- Working with JSON files
- **Other file formats**

Other file formats

- HTML: HyperText Markup Language (beautifulsoup)
- XLS, XLSX: Excel formats (xlrd)
- XML: eXtensible Markup Language (beautifulsoup)
- YAML: YAML Ain't Markup Language (yaml)

Recap

- Sometimes you have text or data that you need to work with
- Python is here for you! You can pull it in and write over it / work with it in files
- Be careful of how you access the text (write may overwrite a file)
- Think about your goals and the best way to work through things

Prep for Thursday!

- Post on ED:
  - One question you have (be specific, include an example)
  - One question you think would be good for the midterm

Thursday: group practice

- **Data work**: Nobel prizes
  - Pull in data and create a dictionary from an original dataset.
- **Simulation**: Conway's game of life: https://playgameoflife.com/
  - Create a simple simulation of Conway's game of life