In class work + brainstorming

Agenda

- Keyboard shortcuts and you
- Jupyter notebooks
- Virtual Environments
- Best practices
- Exam prep
- CSV exercise

KEYBOARD SHORTCUTS

- Faster and more efficient!
- Can set up custom shortcuts in vscode for things you do often (e.g. run all cells in jupyter notebook)
- Jupyter:
  - Esc + A: new cell above
  - Esc + B: new cell below
  - Cntrl + Enter: run the cell
- Terminal:
  - Cmd + C : stop the running process
  - Cmd + L: clear the screen (just gives fresh workspace)

Places to Python

- Jupyter notebook:
  - Akin to rmarkdown if you're used to R
  - Can have blocks of code and cells
  - Can test code chunks
  - Mostly intuitive
- Terminal: ipython or python
  - In-terminal testing/coding: ONLY FOR SMALL EXERCISES! DANGER!
  - Can import files and test them using `import filename`
  - Complements script files well
- Script file:
  - 'Traditional' approach
  - Harder to test directly
  - Nice to run all together vs chunks of jupyter code coming together

Jupyter notebooks

- Have ‘kernel’ that runs under the hood
- Also uses a virtual environment!
- Blocks of code and markdown text
- Interactive and nice – can leave with cell output so you can see what you got before
- Not as smooth for large amounts of code that is all together
- Great for teaching and practice

https://code.visualstudio.com/docs/datascience/jupyter-notebooks

Virtual Environments

- Mise en place: get all your elements together
- Every python ‘thing’ uses a virtual environment – THEY SHOULD MATCH
  - Meaning: your terminal is running a virtual environment
  - Your jupyter notebook is running a virtual environment
  - Your script file is running a virtual environment
- If you are using python, you are using a virtual environment

Best practices: homework assignments

- Script files work best for autograder
- Test OFTEN
- Have clear workflow for how/where you test vs where you place the code

Best practices: general coding

- Devise ways to test your code
- Focus on ‘primary’ cases AND edge cases
- Sometimes I use random number generators to check a particular case and follow it through
- GOOD COMMENTING practice

Github and you

- <u>**No file uploads**</u>
- Set up in terminal
- Git commands:
  - git add
  - git commit –m “message”
  - git push
- Use informative messages!

Exam prep

Exam goals

- Measure of what you are internalizing
- Deep familiarity with topics covered so far (e.g. functions, loops, data structures)
- Opportunity to showcase what has ‘stuck’ and how deeply you’re retaining the information

Task: take home questions

Three types of questions:

- **Definition and use**: assessing your understanding of familiar concepts
- **Spot the error**: practice troubleshooting common issues in code
- **Sketch the code**: opportunity to practice pythonic thinking!
- Propose new!

Definition and use

Prompt:

For each of these terms, provide a definition, a code example and explanation of what it does, and when you might use them:

***Example**: if you were to explain = we would have the following answer:*

- *Definition: the equal sign (=) is an operator we use to assign a value (or list, or tuple, etc.) to a variable.*
- *Code example: var = 4 or lst = [3,4,5].*
- *Code explanation and usecase: the first example assigns the value 4 to the variable named `var'. You would use this when you want to assign a value to a variable for later use. You can also assign lists to a variable using the same structure.*

Spot the error

Short answer: **choose x of y prompts**. Each question has a maximum of z points available.

For each prompt there are at least three substantial issues to address that affect either how the code will /not run, is unpythonic, or is just bad practice. You can submit up to five issues but only need to identify three substantial ones. You can identify either large (a points each) or medium issues (b points each) but try to prioritize 'large' issues.

A **large** issue is something incorrect with the code such that the code would not run as written.
A **moderate** issue is one where the code will likely run as written but
a) may not do what is expected/described in the docstring and/or
b) is unpythonic and/or
c) is inefficient.

These guidelines apply to all of the remaining questions.
Example error description:

```
def multiplies_by_three(lst)
   [x**3 for x in list]
   print(lst)
```

Example errors: missing colon in definition (3 points), actually takes function to the third instead of multiplying (3 points), misspells intake value (should be 'lst' not list)(3 points), 'print' does not help much here (2 points).
Example correction:

```
def multiplies_by_three(lst):
   [x*3 for x in lst]
```

Sketch the code

For the following exercise, you need to produce the following: a list of relevant buildings with addresses AND a dictionary that has the buildings, their classrooms, and the capacities for the relevant list (no addresses) for a given class size. For example, in the dictionary, you might want classrooms that can seat up to 40 students from a list of buildings.

Here is an example excerpt of the txt file:

```
Building Name and Address  classroom capacity
1155   1155 E 60th St         140C    70
1155   1155 E 60th St         295     48
```

Provide a sketch of your code to accomplish this: you ONLY NEED TO PROVIDE function names and docstrings. Be sure to be clear about what the input and output will be and whether any prior functions will be called.

Propose new!

- You can design a different type of question for this – it’s up to you.

In class exercise: Ed discussion

- Eventually, you will post TWO questions of DIFFERENT types
- Start with one type – we will see how time goes!
- Label your question: e.g. DEFINITION: (question)
- Provide your question and answer / explanation

- In-class practice

Opening + working with files CSV version

- Can open them from csv using our regular 'with open'
- Can use csv reader – see today's example for why ☺

```
with open('file.csv', newline='') as csvfile:
        data = csv.reader(csvfile)
        names_csv = next(data)
        for row in data:
                data_csv.append(row)
```

https://docs.python.org/3/library/csv.html

Tasks

track_name,track_add_date,track_add_time,multiple_artists_bool,name_of_artists,album_name,album_release_date,album_release_date_precision,number_of_tracks_in_album,position_in_playlist,track_duration_ms,track_popularity,track_explicit,images_path,data_collection_date

Flowers,3/31/23,10:02:16,FALSE,['Miley Cyrus'],Endless Summer Vacation,3/10/23,day,13,1,200600,87,FALSE,./images/flowers_miley_cyrus.jpg,3/31/23

Kill Bill,3/31/23,10:02:16,FALSE,['SZA'],SOS,12/8/22,day,23,2,153946,94,FALSE,./images/kill_bill_sza.jpg,3/31/23

- Upload the data
- # Task 1: dictionary to find how many times each song is in playlist
- # Task 2: dictionary providing a list AND COUNT of songs an artist has

TASK: choose your difficulty level!

- No hints: GO! Data file here
- Some hints: Jupyter notebook here
- More hints: Jupyter notebook here

How to improve

- Practice
- KAGGLE: start a new project!
- Practice
- Review other (good) code and see how it compares to yours
- Practice
- Talk through your code with someone else – see it from a different perspective
- Practice

Recap

- Feel comfortable-ish in the content so far
- PRACTICE IS YOUR FRIEND
- Reflection: midterm reflection – best to wait until after the midterm to complete
- Be honest – much better to be honest and do the reflection than to have it be something like, ‘my biggest flaw is being too perfect and caring too much’