# DATA203 Foundational Python (Prof. Maull) / Fall 2024 / HW0

| Points Possible | Due Date | Time Commitment (estimated) |
|---|---|---|
| 10 | Wednesday, September 11 | *up to* 8 hours |

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by University or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

## OBJECTIVES

- Familiarize yourself with the JupyterLab environment, Markdown and Python
- Explore JupyterHub Linux terminal console integrating what you learned in the prior parts of this homework
- Learn more about and use string manipulation

## WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hw0`. Put all of your files in that directory. Then zip or tar that directory, rename it with your name as the first part of the filename (e.g. `maull_hw0_files.zip`, `maull_hw0_files.tar.gz`), then download it to your local machine, then upload the `.zip` to Canvas.

If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using zip in Linux.

If you choose not to use the provided notebook, you will still need to turn in a `.ipynb` Jupyter Notebook and corresponding files according to the instructions in this homework.

## ASSIGNMENT TASKS

**(0%) Familiarize yourself with the JupyterLab environment, Markdown and Python**

As stated in the course announcement Jupyter (https://jupyter.org) is the core platform we will be using in this course and is a popular platform for data scientists around the world. We have a JupyterLab setup for this course so that we can operate in a cloud-hosted environment, free from some of the resource constraints of running Jupyter on your local machine (though you are free to set it up on your own and seek my advice if you desire).

You have been given the information about the Jupyter environment we have setup for our course, and the underlying Python environment will be using is the Anaconda (https://anaconda.com) distribution. It is not necessary for this assignment, but you are free to look at the multitude of packages installed with Anaconda, though we will not use the majority of them explicitly.

As you will soon find out, Notebooks are an incredibly effective way to mix code with narrative and you can create cells that are entirely code or entirely Markdown. Markdown (MD or `md`) is a highly readable text format that allows for easy documentation of text files, while allowing for HTML-based rendering of the text in a way that is style-independent.

We will be using Markdown frequently in this course, and you will learn that there are many different "flavors" or Markdown. We will only be using the basic flavor, but you will benefit from exploring the "Github flavored" Markdown, though you will not be responsible for using it in this course – only the "basic" flavor. Please refer to the original course announcement about Markdown.

**§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.**

Play with and become familiar with the basic functions of the Lab environment given to you online in the course.

**§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.**

Please *create a markdown document* and read the documentation for basic Markdown here. Learn to use all of the following:

- headings (one level is fine),
- bullets,
- bold and italics

Again, the content of your documentcan be whatever you like, just learn some of the basic functionality of Markdown.

**(0%) Explore JupyterHub Linux terminal console integrating what you learned in the prior parts of this homework**

The Linux console in JupyterLab is a great way to perform command-line tasks and is an essential tool for basic scripting that is part of a data scientist's toolkit. Open a terminal console in the lab environment and familiarize yourself with your files and basic commands.

**§ Task: Understand basic Linux file operations**

Basic file operations go a long way to understand the way Linux works. In this part, you will understand folders, files and making revisions to a file. These files will be visible within Jupyter, which makes moving from one platform to another seemless. We will create a folder, file, make edits.

- open a Jupyter console
- create a file called `README.md`
- type `mkdir your_folder_name` to create a folder in filesystem *in the current folder where you are*
- use `cd your_folder_name` to "change directory" and move into the folder you just created
- use `pwd` to "print working directory" to verify you are in the folder you created
- create a file by type `touch README.md` the `touch` command creates a file if it does not already exist, otherwise it will change the timestamp of that file when it is "touched"
- type `echo "Hello this is test text." > README.md`. This will take the words you typed and "append them" into the file `README.md`
- to see the contents of your file typing `cat README.md` or `more README.md` or `less README.md`

**§ Task: Learn to quickly obtain remote files in Linux**

The commands `wget` and `curl` are useful for grabbing data and files from remote resources off the web. Using these tools from the command line streamline your workflows and are often faster than writing a program to do the same. These tools will also expand and strengthen you data science skills, added a few more tools to your toolkit is rarely a bad idea.

1. Read the documentation on each of these commands by typing `man wget` or `man curl` in the terminal.

- `man` stands for *manual* and nearly all versions of Linux have such documentation pages for the majority of commands. If it fails, try the command with the `-h` or `--help` flag, such as `wget --help`

2. Make sure your output goes to a file and study the documentation to the select the proper flags to do so.
3. You can obtain nearly any file anywhere on the Internet with these commands. For example, the Library of Congress interview with Barry White from 1987: https://www.loc.gov/item/jsmith000021/
    - click on this interview link
    - choose the dropdown for the **mp3**
    - when the page opens up, there will be a player that starts the interview, copy the URL
    - go to you Jupyter terminal and run the command `wget <the_url_you_just_copied>`, where you will paste the URL you just copied in the `<the_url_you_just_copied>`
4. Either on the Library of Congress site or somewhere else, play further with `wget` and `curl` to download some other files you might have of interest.

**(100%) Learn more about and use string manipulation**

We learned in lecture that strings are *sequences* in Python.

We also learned that the sequence types have a number of basic operations like concatenation, length, etc.

Strings have a lot of other operations on them that make them exceedingly useful for text processing. In fact, Python is exceptionally good at processing text, as you will see.

First things first, please go to the documentation on Python strings, also known as **`str`** or "Text Sequence Type":

- https://docs.python.org/3/library/stdtypes.html#textseq

Study it and especially the String methods (see more here: https://docs.python.org/3/library/stdtypes.html#string-methods).

You will be using this short **AI-generated writeup** about record players:

> The first recorded music was made in the early 1800s, when phonographs were invented. The earliest phonographs were cylinder-based devices that used a needle to etch sound waves onto a rotating cylinder coated with wax or tin foil. These early machines were not designed for playback, but rather for recording and reproduction of spoken words and sounds. In the late 1880s, Thomas Edison developed the first practical phonograph that could both record and play back audio, using a spiral groove on a cylinder.
>
> The next major innovation in recorded music came with the invention of flat discs, known as records, by Emile Berliner in the 1890s. Berliner's gramophone used a needle to etch grooves onto a flat disc made of shellac or other materials, which were then played back using a needle and a horn-shaped speaker. The first recordings on these flat discs were made around 1887, with Victor Records (founded by Eldridge R. Johnson) being one of the earliest commercial record labels.
>
> The development of vinyl records in the early 20th century revolutionized the music industry. Vinyl was a more durable and long-lasting material than shellac, allowing for mass production and wider distribution of recorded music. The introduction of 78 rpm records (played at a speed of 78 revolutions per minute) around 1925 marked the beginning of the modern record player era. As radio broadcasting became popular, record players became a staple in many homes, allowing people to listen to their favorite artists and songs.
>
> The post-war period saw significant improvements in record player technology, with the introduction of high-fidelity (hi-fi) stereophonic sound around 1950. Hi-fi systems used two channels (left and right) to reproduce music, providing a more immersive listening experience. The development of LPs (long-playing records) by Columbia Records in 1948 further increased record players' popularity, as they could store up to 23 minutes of music per side.
>
> The rise of compact discs (CDs) and digital music formats in the late 20th century marked the beginning of the end for record players. CD players became widely available around the mid-1980s, offering higher sound quality and greater durability than vinyl records. By the early 2000s, CDs had largely replaced LPs as the preferred format for recorded music. However, in recent years, there has been a resurgence of interest in vinyl records, with many music fans seeking out vintage record players and new albums released on vinyl to enjoy the unique sound and tactile experience they offer.

**§ Task: Basic String functions.**

Use the passage provided and answer the questions. **Provide your answer in a Jupyter Notebook.**

1. Set a variable called `passage` with the string provided into a multi-line string using `"""` which we talked about in lecture. Make sure you preserve the spaces between paragraphs, which are `"\n\n"` (two newlines).
2. Use `splitlines()` to determine how many paragraphs are in the string.
   Recall, `splitlines()` returns a *sequence* (i.e. a list sequence), so counting the paragraphs is easier with `len()`.
3. Use `lower()` to produce the lowercase version of the whole string.
4. Get the first (index 0) paragraph, and return the number of words in the paragraph. **NOTE:** you can include punctuation as part of the word. You will make use of your solution in #2 and `split()`.

**§ Task: Advanced String functions.**

Now that you have `passage` in a variable, there are a few more String operations we want to try to familiarize ourselves with.

1. Use `replace()` to replace all instances of the word `" the "` with `###`.
   You would be advised to use `lower()` first so that all of your words are *normalized*. Note also the spaces around the word.
2. How many `" the "` words are in the passage?
3. How many times was the word `" music "` used in the passage?

**§ Task: Sequence iteration.**

Now we will put looping into our work and ask more complex questions of the text.

To produce a list sequence of all the words in `passage` then you will learn that `split()` will be very valuable. Remember your use of `split()` in your prior solution.

1. Study what `passage.split()` does and write a sentence fragment explaining what it does.
2. Use a `for` loop with `split()` to find all words that end in `"ing"`. Provide the count of such words. **Note**: you do not have to remove punctuation, so if you end up with an item like `"walking,"` you can ignore it in your count.
3. How many words end in `"ed"`?

**§ Task: BONUS**

This part will early you *up to* an extra +2 BONUS points.

1. [+**1 point**] Remove all _ending_and *beginning* punctuation from `passage`. Beginning and ending punctuation is defined as any symbol at the *beginning or end of a word* in the set of symbols: `!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~`
2. [+**1 point**] Print and provide the count of all words that end in `"s"`.