DATA203 Foundational Python (Prof. Maull) / Spring 2025 / HW2

Points Possible	Due Date	Time Commitment (estimated)
25	Sunday, March 23	*up to* 15 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

- Explore JupyterHub Python *shell* commands inside cells
- Understand and use functions to process data.
- Understand and use dictionaries for complex data.
- [BONUS]

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/hw1`. 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_hw1_files.zip`, `maull_hw1_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%) Explore JupyterHub Python *shell* commands inside cells

In the last time we learned to run the terminal console commands in JupyterLab, which 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. Last time we used a terminal console in the lab environment this time we familiarize ourselves with Jupyter *shell* escape commands **within** a notebook.

Study:

- Python Data Science Handbook: IPython and Shell Commands

for full documentation on *shell* … they are **very** useful!

§ Task: Use Jupyter *shell* commands to perform the same commands as last time.

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 seamless. We will create a folder, file and make edits.

- type `!mkdir your_folder_name` to create a folder in filesystem *in the current folder where you are*
- 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"
- edit the file in Jupyter with the text editor
- to see the contents of your file typing `!cat README.md`

§ Task: Use *shell* command `wget` to quickly obtain remote files in Linux

As before get a remote file this time it will be from the Internet Archive:

- in a cell type `!wget https://ia801306.us.archive.org/15/items/fouraddressesats00howa/fouraddressesats00howa`
- execute the cell
- verify the file was retrieved by opening it

(50%) Understand and use functions to process data.

We learned in lecture that function are necessary tools in Python.

I have provided a starter notebook for you to use, which will greatly enhance you ability to complete the assignment. See that in the folder here. The name of the notebook is `hw1/hw1_starter.ipynb`:

- https://github.com/kmhuads/s25_data203/tree/main/hw1/hw1_starter.ipynb

You will notice the is a text file in `hw1/data/`:

- `data/countries.txt`

I recommend you look at these files and see what is in them. Text files form the basis of many data files that you will interact with in your future as a data scientist – *maybe* one day AI will take over our reliance to write just a little code to deal with them, but we will see … but not today.

In the notebook are some scaffolding code. You will need to study it and use it in the solutions being asked.

§ Task: 1.0: Practice explaining code.

Use the code in the first cell of the provided notebook and answer the question below:

1. Explain in your own words what the cell is doing does. Your explanation will include the description of the inputs (if any), and you will need to review the Python file I/O libraries and specifically `readlines()` for details of the function.

§ Task: 1.1: Write a function that takes the `countries` list and returns the cleaned list.

In this task, you will enter into a new cell the function that takes a single parameter `countries_list` and returns the cleaned data. By "cleaned data", your return list will not have any extraneous characters, spaces or newlines in it.

§ Task: 1.2: Write a function that takes two parameters `cstr` and `start_char` and returns `True` if the `cstr` starts with `start_char`, otherwise it returns `False`.

Just remember, there are many ways to solve this problem, but do not overthink it!

§ Task: 1.3: Answer the questions. Show your code in the notebook cells.

You can only obtain full credit if you show the *working* Python code that solves the questions being asked.

1. How many countries start with `s`?
2. Are there more countries that start with `m` than `t`, if so how many more?
3. Show the function call that gives the number of countries that start with `m`.

(50%) Understand and use dictionaries for complex data.

We learned in lecture that dictionaries are very flexible data structures in Python.

In the same starter notebook, `hw1/hw1_starter.ipynb`:

- https://github.com/kmhuads/s25_data203/tree/main/hw1

You will notice the is another text file in `hw1/data/countries_census.csv`:

- `data/countries_census.csv`

This file has 3 columns: the country name, the population census and the percent population change (in that order).

§ Task: 2.0: Explain what the function `mystery_maker()` does. Use the provided sample notebook file.

Explain in your own words what the function `mystery_maker()` in the cell below is doing. Your explanation will include the description of the inputs (if any) and outputs. **Do not overthink it and run the function to study it's outputs!**

§ Task: 2.1: Follow the steps to complete the task. 1. Write the code that assigns a variable named `d_countries` to the execution of `mystery_maker`. 2. In a single `for` loop, put all the countries from your `countries_cleaned` list above into the structure `d_countries`. 3. Print the result of d_countries. Describe the output.

(0%) [BONUS]

This task requires you to think a little, but it is much easier than you think.

We stated before that the last column of our file is the growth rate of the country. Your function will take a number (float) and minimally:

0. take a single parameter `growth_rate`,
1. load the file of country data (you may use code already written in your solutions above),
2. filter just the data in the file where the last column of data is within 10% of the provided `growth_rate` (HINT: when you read the data, it will be a string, you will need to convert it to a number like this, if `v="12.3"`, `float(v)` converts v to the number `12.3`).
3. return the countries that were found, the returned data should be the list of those countries (e.g. `['Kenya', 'Uganda']`)

This will require you to combine all your skills together. You will need to use `for` loops, store data in variables and return data as lists.

GRADING:

- up to 5 BONUS points for this part of the assignment
- PARTIAL SOLUTIONS WILL BE ACCEPTED, partial points will be awared accordingly
- if you have the time and are up for the challenge, you don't have much to lose!

§ Task: Implement the completed or partial solution.