**Module 1: Lab Activity - Commenting and Documentation**

**Submissions:**

**1 – conversion.py, login.py, and average.py with added comments**

**2 – New Word document with answers to Part 2**

**Practice Commenting:**

Learning to comment well is a valuable tool. Not only will you learn how to write more clearly and concisely in general, but you’ll no doubt gain a deeper understanding of Python as well.

Comments are created in Python using the pound sign (#) and should be brief statements no longer than a few sentences. Here’s a simple example:

# This is a comment

**According to the Python Style Guide** [**PEP8**](https://pep8.org/#maximum-line-length) **the maximum line length for comments should be 72 characters.**

**Tips**

1. Keep comments as close to the code being described as possible. Comments that aren’t near their describing code are frustrating to the reader and easily missed when updates are made.
2. Don’t use complex formatting (such as tables or ASCII figures). Complex formatting leads to distracting content and can be difficult to maintain over time.
3. Don’t include redundant information. Assume the reader of the code has a basic understanding of programming principles and language syntax.
4. Design your code to comment itself. The easiest way to understand code is by reading it. When you design your code using clear, easy-to-understand concepts, the reader will be able to quickly conceptualize your intent.

**Part 1:** Start writing comments for yourself in code. Make a point to include simple comments from now on where necessary.

1 - Download **conversion.py** from D2L > Module 1: Introduction to Programming > Module 1: Lab Activity – Commenting and Documentation.

- Open with a text editor or IDE such as PyCharm **(Do Not Use Word)**

What do you think this program does? Can you easily tell what’s going on without any commenting? Run the program and see if you were correct.

- Based on observation, add comments to the program to:

A – give authorship

B – give a date for last edit (today)

C – give a simple explanation of what the program does

D – comment on significant lines of code or anything else that might be helpful to you or someone else looking at this code

2 - Do the same for **login.py** and **average.py**

**Practice Documentation:**

A README is often the first item a visitor will see when visiting your repository. README files typically include information on:

* What the project does
* Why the project is useful
* How users can get started with the project
* Where users can get help with the project
* Who maintains and contributes to the project

A thorough README may include:

* Project Title -
* Motivation
* The Build Status
* Code Style
* Screenshots
* Tech/framework Used
* Features
* Code Example
* Installation -
* API Reference
* Tests
* How to use
* Contribute
* Credits
* License

**Part 2:** Review some ReadMe files. Answer the questions below.

3 - Let’s look at a ReadMe file - <https://github.com/freqtrade/freqtrade>

- Display the README.md file in GitHub by clicking on it

- What are you able to tell about this project based on the README.md file?

4 - Use this link: <https://github.com/alirezamika/autoscraper>

- Open the README.md file in GitHub by clicking on it

- Does this README clearly include the information above? If not, what is it missing?