Sam Bircher
Foundations in programming: Python
Module 6

**INTRODUCTION**
Throughout this module, we learned about strategies that improve script legibility and utility. We started by developing an understanding of functions. We then reviewed local vs global variables within the context of functions. Next, we learned how to incorporate parameters into functions. Finally, we learned how to create classes that group functions and how to organize code through separation of concerns patterns.

**TASK 1**
In this task, we read the module notes and reviewed the videos. Here, we reviewed common techniques for organizing code, which becomes more important as the code becomes more complex.

Functions: Reusable block of code that performs a specific set of tasks. Functions are modular, which means you can break down the code into smaller pieces. Functions are reusable, meaning that once you define a function, you can use its code multiple times throughout the program. Functions are labeled with a specific name. Functions are defined after the variable declarations since they must be defined before a script is called. This area is called the "main body." You "call" the function later in your code.

Global and local variables: Variables can be classified as global or local. Local variables are declared within a function. They can only be used within the function that they are declared. Global variables are declared outside of any function (typically in the main body of the script). They apply to the entire body of the script. Inside a function, the local variable takes precedence over a global variable. So to use the global variable within a function, you have to define the global variable first.

Parameters: Parameters allow you to pass data into your function. Parameters can be used instead of global variables. A significant advantage of using parameters is that it avoids unintentional modifications of a global variable within a function. Parameters allow you to encapsulate data required for the function. Parameters provide enhanced flexibility and more predictable behavior. The values that you pass into parameters is called an "argument". When we pass a variable to a function as an argument, it will not change the variable if it is immutable (i.e., int, float, tuple). If the variable is mutable (i.e., list, dictionary), then changes within the function changes the variable throughout the script.

Return Values: Values that a function returns. They allow functions to act as an expression. Using return values improves the clarity and predictability of a function. (When you see a function that returns a value, you know what to expect from it). They allow you to test the function output independently. Return values promote immutability and reusability because they don't modify external variables. Functions that return values can be chained together to form pipelines of data processing.

Classes and Functions: Classes are a way of grouping functions, variables, and constants (by the name of the class). Classes provide a way to organize code. Static classes use functions that do not change. Document strings (Docstrings) are headers at the beginning of a class or function. This allows other developers to see what the function or class does.

Separation of Concerns Pattern: SOC is a software design principle that aims to enhance maintainability, scalability, and readability of code by breaking it down into self-contained components. A "concern" refers to a specific aspect or responsibility of a program's functionality (i.e., data storage, data processing, input validation, etc). Presentation concerns include user interface elements (how the data is displayed to the user). Logic concerns include the functionality of the application (i.e., data processing, calculations, etc). Data storage concerns include how data is stored, retrieved, and managed.

- The data layer is responsible for handling data-related operations.
- The processing layer is responsible for implementing the core functionality and rules of the application.
- The presentation layer focuses on presenting information to the user.

**TASK 2**
Here, we reviewed two videos that described how to use functions and different types of arguments.

**TASK 3**
Here, we reviewed a document further describing how to use functions with different types of arguments.

**TASK 4**
Here, we created a coding script that registered students for a course and saved the data as a list in a JSON file. The script starts out like all others with the appropriate header and definitions of constants and variables. The script is organized using a separation of concerns pattern.

The first layer is data processing and is organized into a class called “FileProcessor.” Here, code is written to open and read data from the existing JSON file “Enrollments.json”. Code for writing new data to the file is also embedded in this class.

The second layer is the presentation layer and is organized into a class called “IO”. Within this class are several functions containing code that 1) displays the menu to the user and elicits input, 2) prints a list of students and their registered courses from the JSON file.

The last portion of the script calls a function that displays the menu to the user. The functions that correlate with each menu option (Register a student for a course, display current data, save data to file, and exit the program) are also called in the last portion of the script.

**SUMMARY**
In this module, through videos, labs, and readings, we learned how to use functions, parameters, return values, classes, and separation of concerns patterns. We put all of this knowledge together in task 4 by creating a script that registered students for a course and saved this information to a JSON file.