Brian Pochert

03/04/2025

IT FDN 110A (WIN2025)

Assignment 06

https://github.com/bpoch/Python110-Winter2025

Assignment 06 Classes, Functions and SoC Pattern

Introduction

Module 06 showed us two very important concepts called classes and functions. In addition to classes and functions, we learned about Separation of Concerns (SoC). SoC, classes and functions all work together to make long code organized, easier to read, and more efficient. The concepts are straightforward, but can be tricky to implement. Organization is key! This was a fun but challenging module.

Functions

In Python, functions are blocks of code that can be reused over and over in your program. They set a series of code apart from the rest of the code to be referenced later in the program as many times as you need. Functions are named and they can be called later by name in the program. Function names usually provide an indicator or hint as to what they do. An example of a function name could be “save_file”. You establish a function in Python by using the “def” keyword with a set of parentheses () and a colon :

The code for creating a function named “save_file” will look like this:

```
def save_file():
```

Parameters are important to functions and they are listed inside the parentheses. Some functions may not need parameters, but most functions will have some parameters listed. Parameters are the data that the function will use, such as variables. It is important to know that any variables used inside the function are only valid in that function unless they are labeled a global variable.

Another important keyword used with functions is “pass”. The Python keyword “pass” is used below a function when no other code exists in the function. This would tell Python to recognize the function but ignore it until such time more code can be added and the function code is completed. It would look like this:

```
 def save_file():
```

```
pass
```

This is important for organization and programming construction. In assignment 06 we were given the required function names to use. I was able to organize the functions by name and then type pass under them, knowing I would complete the code at a later time.

You can learn more about functions here:

https://www.w3schools.com/python/python_functions.asp (external site)

Classes

Classes are also very important to the organization of larger programs. Classes are a collection of functions. Usually Classes are usually a group of functions that relate to each other in one way or another. The functions are likely centered around certain aspects of the programs' process. Grouping functions together in a class improves the readability of the code. Classes help organize the code and make it easier to read and work on.

Classes are established in a program by using the Python key word “class” followed by the Class name and a colon : Just like function names, class names will usually indicate what job they perform. Class names should also be indicative of their purpose. An example of a class name could be FileProcessor and all the functions in the class would relate to reading and writing data to the file. Code to create the Class FileProcessor would look like this:

```
class FileProcessor:
```

As Classes will hold a fair amount of code, it is customary to include some notes on the class and its purpose. These notes are called document strings or docstrings. They are similar to multi line comments we learned about earlier. To create a docstring, you would use “ “ “ on the line right under class name. This will allow you to complete the docstring to describe your class.

SoC

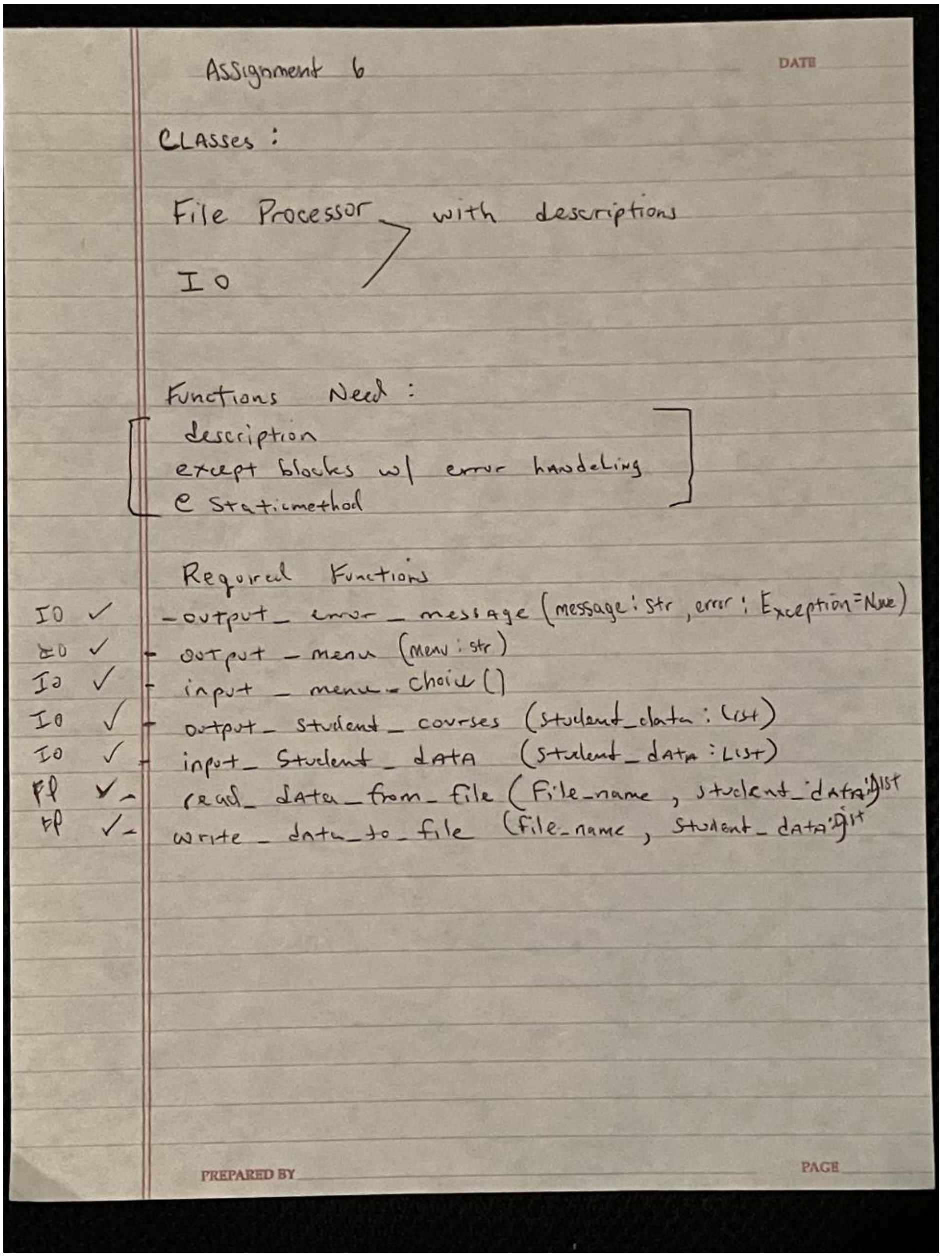
Separation of Concerns or SoC is a coding principle that provides guidance on how code should be constructed. The principle states that your code should be separated by “concern” or section. Sections can include a variety of topics, but in our instruction, we will focus on these three: Presentation, Logic, and Data Storage.

SoC is a principle that programmers adhere to for continuity in the industry. It helps programmers review and work on other’s scripts. You can read more about SoC on Wikipedia here:
https://en.wikipedia.org/wiki/Separation_of_concerns#:~:text=In%20computer%20science%2C%20separation%20of,code%20of%20a%20computer%20program. (external site)

Assignment 06

Assignment 06 asked us to use a starter program based on our existing student registration program from module 05. The assignment asks us to modify the starter py to include Classes and Functions. The assignment asked us to use lab 06-03 as a guide. The assignment instructions provided us with all the constants, variables, Classes, and functions to use. Any parameters utilized in the functions were also included in the assignment instructions.

I began by listing the variables and constants at the top of code and deleting anything not used any longer. I then saw there are two classes named Fileprocessor and IO (input/output). I began constructing my blueprint for the script using old fashion pen and paper. I listed all the functions out and then figured out which class they belong to. Here is my "blue print" in figure 1



Assignment 6

DATE

CLASSES :

File Processor
IO
with descriptions

Functions Need :

description
except blocks w/ error handeling
@ Staticmethod

Required Functions

IO ✓ - output_error_message (message: str, error: Exception=None)
IO ✓ - output_menu (menu: str)
IO ✓ - input_menu_choice ()
IO ✓ - output_student_courses (student_data: list)
IO ✓ - input_Student_data (student_data: List)
FP ✓ read_data_from_file (File_name, student_data: list)
FP ✓ write_data_to_file (file_name, Student_data: list)

PREPARED BY

PAGE

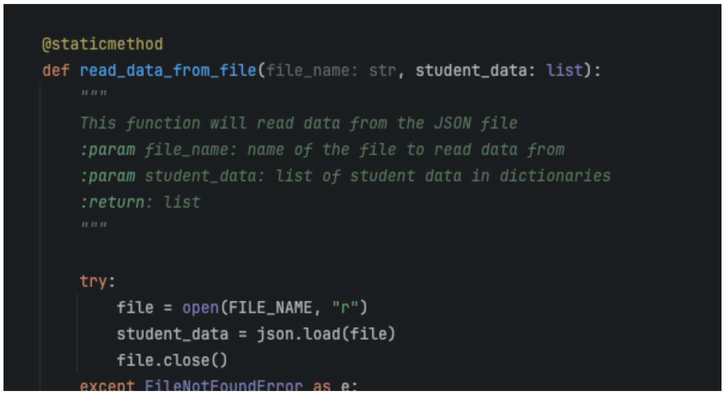
(figure 1)

I placed a FP or an IO to the left of the function to create my Class groupings. I then coded the "blue print" shell in Pycharm, listing the FileProcessing class and its two functions first. Then, I created the IO class and its five functions.   I utilized the keyword pass under the all the functions until I was able to enter the code later.

Writing the code for each function was challenging, and I utilized the Lab06-03 script often for help with syntax. I was also able to cut and paste code from the starter into most functions. As with other modules, I found cutting and pasting caused indentation issues which had to be fixed.

The biggest challenge I had was with the error handling and using a function for error messages instead of the print statements in Module 05.

I also learned that listing a parameter in the function parentheses without referencing it in the function's code cause the parameter to "gray out". I cut and pasted the try/except code from the starter script in reference to reading the JSON file. The starter used the constant FILE_NAME. When I put this in function, it did not match the provided parameter of file_name : str. This caused the parameter to gray out. I show this here in figure 2:



```
@staticmethod
def read_data_from_file(file_name: str, student_data: list):
    """
    This function will read data from the JSON file
    :param file_name: name of the file to read data from
    :param student_data: list of student data in dictionaries
    :return: list
    """

    try:
        file = open(FILE_NAME, "r")
        student_data = json.load(file)
        file.close()
    except FileNotFoundError as e:
```

(figure 2)

I was able to research this and fix it by renaming FILE_NAME to file_name.

Summary

Module 06 assignment was challenging. I found organizing the code and creating a blue print fun. There was not a lot of original code we needed to come up with. The starter script, assignment instructions, and lab example provided many of the clues need to make the script work as required. I struggled with keeping things in proper indentation and making sure syntax was correct. The Lab06-03 provided a good example and helped me figure out how to code certain things like the exception handling.