

Adam Eichholz
August 20, 2025
Foundations of Programming: Python
Assignment 6
<https://github.com/eichhoa1/IntroToProg-Python-Mod06>

Functions

Introduction

In this paper, I look at the organizational updates made to the student enrollment project through the addition of functions and Separations of Concerns. In a change from my previous assignments, I have elected to list my screenshots as Figures in their own section, located after the Summary. This change has been made to try and reduce repeated spacing in the Topic body. References to appropriate Figures will be made in parentheses.

Topic

In approaching the assignment for this week, I needed to take a careful step back to assess how I needed to do it. While I felt that I understood the mechanics of functions well enough, and especially the reasoning for Separation of Concerns, I did not feel like I had a good understanding yet of parameters. The labs had helped me understand well enough *what* I needed to do, but I did not yet feel like I knew *why* I would be using them or how they worked.

To understand parameters, I ended up needing to consult with some engineering friends, and referred to the W3Schools article on Functions and Parameters. I managed to figure out that parameters are essentially what keeps your code from having to “guess” details, and that with parameters, a function will only execute when the number of those parameters have been met (i.e., that a function with two parameters will not run if it has only received data for one parameter or receives data for what would be three parameters). A friend made an analogy that really helped me understand it: *“Another way to think of it is as a verb that has an object. So like it would make sense to say ‘go pick up a brick’, but if someone told you to ‘go pick up’, you’d be like, pick up what?”*

With this in hand, I was able to start setting up my code. Having gone through the labs, I found the code to be pretty similar, and was able to structure things easily. While it would have made sense to an extent to convert the code into formulas, then add parameters, then move everything into classes under separate concerns, I wanted to try and approach things from the top down, as if I were writing a fresh program.

To start, I set up the Processing and Presentation categories, then added the File Processing and IO classes. I did observe in the notes that three categories are recommended opposed to

our two, and I would be interested to see in the future if categories might contain multiple classes. Given our simple program, the scope made sense enough.

Things progressed smoothly from there, and the clear expectations for the names and parameters of the functions made it clear which block of code was supposed to go where, and what needed changing. I referenced my Lab 3 answers for what specific lines needed adjusting to meet the parameters, but at this point I do not feel overly confident that I would always know how I am supposed to know what to change. At one point, I experienced a continuous error, based on the `student_data` parameter not being found in my code. The frustratingly simple answer? I had **`student.data`** = `json.load(file)` written down. Once I finally found and fixed it, testing went smoothly from there (see: **Figure 1**).

With that issue finished, I completed the setup of my IO class functions and tested each one in the process (see: **Figure 2**). One final hurdle I ran into when testing was a lesson in the importance of return. I had grown a bit complacent with the IO functions that did not have a return at the end of the code block, so when setting up the function for Input Menu Choice, I had accidentally omitted the return at first (see: **Figure 3**). When I ran my testing, this resulted in a very amusing error, where entering a number would endlessly repeat the “Please only choose 1, 2, 3, or 4!” error without a break, forcing me to shutdown the execution. Once again, by referencing my Lab answers, I realized what was missing and fixed the code. From there, I was able to complete the rest of the criteria and my code ran smoothly (see: **Figure 4**).

Summary

The assignment for this week, while setting up some interesting code, felt like a very strong growth in theory. Categorizing your code structure into separate concerns makes a lot of sense, and I found it a very interesting revelation that things would largely be set up ahead of time with functions, and that the actual execution of the program itself might be smaller in comparison. I still do not feel like I have the best understanding of parameters and returns, and this is something I would like to develop my comprehension of in the future.

Figures

Figure 1: File Processor Class and Read Data From File Function

```
# Processing ----- #
class FileProcessor: 2 usages
    """
    A collection of processing layer functions that work with Json files

    ChangeLog:
    Adam Eichholz,8/20/2025, Created Class
    """

    @staticmethod 1 usage
    def read_data_from_file(file_name: str, student_data: list):
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages(message: "Text file must exist before running this script!", e)
        except Exception as e:
            IO.output_error_messages(message: "There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data
```

Figure 2: Presentation Concern and Output Error Message Function

```
# Presentation ----- #
class IO: 11 usages
    """
        A collection of presentation layer functions that manage user input and output

        ChangeLog: (Who, When, What)
        Adam Eichholz, 8/20/2025, Created Class
        Adam Eichholz, 8/20/2025, Added menu output and input functions
        Adam Eichholz, 8/20/2025, Added a function to display the data
        Adam Eichholz, 8/20/2025, Added a function to display custom error messages
    """

    @staticmethod 7 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error message to the user

        ChangeLog: (Who, When, What)
        Adam Eichholz, 8/20/2025, Created function

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
```

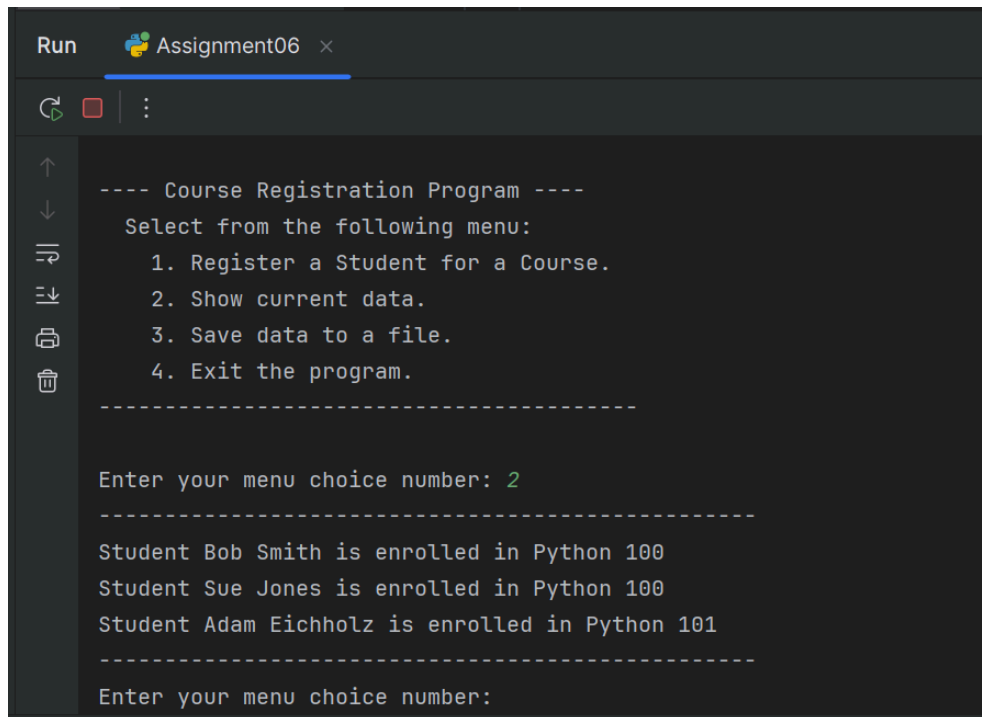
Figure 3: Input Menu Choice Function

```
@staticmethod 1 usage
def input_menu_choice():
    """ This function gets a menu choice from the user

    :return: string with the users choice
    """

    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1", "2", "3", "4"): # Note these are strings
            raise Exception("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())
    return choice
```

Figure 4: Code Execution



```
Run Assignment06 x
↻ □ ⋮
↑
↓
≡
⇅
🖨
🗑
---- Course Registration Program ----
Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----

Enter your menu choice number: 2
-----

Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Adam Eichholz is enrolled in Python 101
-----

Enter your menu choice number:
```