Tulip O'Neill

May 28, 2025

Foundations of Programming

Assignment 06

06: Functions

Introduction

This past week, we were introduced to numerous new concepts in Python. This includes using functions, parameters, arguments, and classes to make our scripts more well-rounded and easier to understand. I personally really liked learning about these concepts because I tend to be an organized person and was quickly finding myself lost and confused within my own code.

This week's assignment required us to build off of week 05 to implement functions, classes, and even use the separation of concerns pattern. Throughout this document, I will be highlighted how I completed this assignment and sharing my thoughts from along the way.

Drafting the Script

In previous weeks, I found myself completely lost for far too long from trying to combine the acceptance criteria with the assignment starter with no real plan. This week, I decided to start the draft of my script by copying the assignment starter into PyCharm, moving the bulk of the script below where I would be drafting, and creating more of an outline for myself. I decided I would fill in the relevant parts of the starter into my draft as needed to reduce as much confusion as possible. This skeleton so to speak ended up resembling the code that resulted from lab 03 very closely.

Figure 1: Initial outline of script

Once I had this outline created, I went through the acceptance criteria for the input/output, processing, and error handling. I then moved in any relevant code from the starter and made

adjustments to account for the new requirements concerning functions. Starting with menu choice 1, I added code to the output_menu() and input_menu_choice() functions. I quickly found that the code provided in the starter file did not translate well to what I was hoping to accomplish, so I ended up moving code mostly from lab 03. I also found that starting with menu choice 1 and going from there was a little confusing because that's not how the program starts. I had some code thrown in before I decided to go about this differently, which made things a little more confusing than needed. I decided to work through the code the way the program would run, starting with reading the data from a file and ending by breaking the loop. I got into a groove and didn't take any screengrabs until after I had a draft. By the time I had my draft put together, I felt very accomplished and felt that there might not even need to be any edits made!

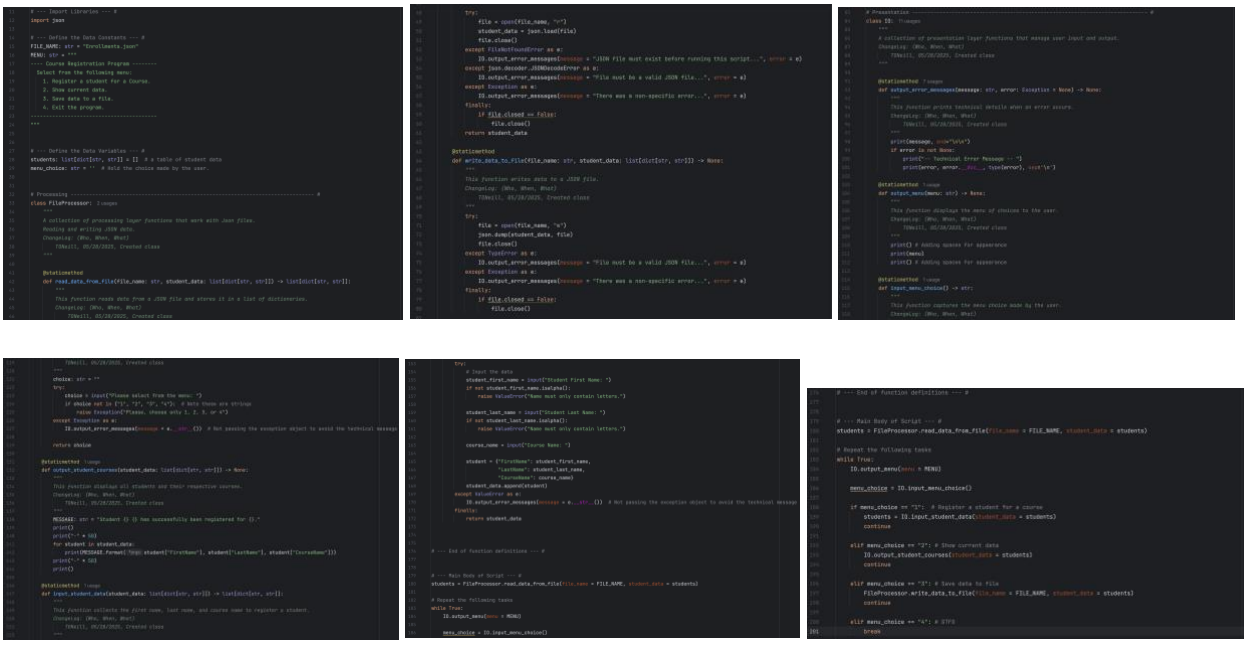

Figure 2: First draft of script

When I tested the script in PyCharm, everything seemed to work well. I decided to make some minor formatting changes to some of the messages that are printed to the user, but other than that, I left the script as is. The final test was to run the script in my terminal, which worked! Honestly by this point, I was so wiped that I decided to call it a night and hope for the best.

```
/usr/local/bin/python3.13 /Users/tulip/Documents/Python/PythonCourse/Module06/Assignment/Assignment06.py


---- 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.
----------------------------------------


Please select from the menu: 1
Student First Name: Vic
Student Last Name: Vu
Course Name: Python 200


---- 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.
----------------------------------------


Please select from the menu: 2

-------------------------------------------------
Student Bob Smith has successfully been registered for Python 100.
Student Sue Jones has successfully been registered for Python 100.
Student Vic Vu has successfully been registered for Python 200.
-------------------------------------------------
```

```
---- 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.
----------------------------------------


Please select from the menu: 3


---- 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.
----------------------------------------


Please select from the menu: 4

Process finished with exit code 0
```

Figure 3: Output of script draft in PyCharm

Summary

In conclusion, I was able to draft a more thoughtful and organized program that takes in, displays, and saves information about a student's course registration status through the use of functions, classes, and using the separation of concerns pattern. This assignment easily took the most time for me to complete, but by taking things slow and being more methodical, it made a lot of sense. I once again found myself in a situation where the assignment starter file made things more confusing than helpful, but I'm working on it.

Appendix

Link to GitHub:
https://github.com/tuliponeill/IntroToProg-Python-Mod06

Final script:

```
# ------------------------------------------------------------------------
--------------- #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot, 1/1/2030, Created script
#   TONeill, 05/28/2025, Created script
# ------------------------------------------------------------------------
--------------- #


# --- Import Libraries --- #
import json

# --- Define the Data Constants --- #
FILE_NAME: str = "Enrollments.json"
MENU: str = """
```

```
---- 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.
----------------------------------------
"""


# --- Define the Data Variables --- #
students: list[dict[str, str]] = []  # a table of student data
menu_choice: str = ''  # Hold the choice made by the user.


# Processing ------------------------------------------------------------------
--------------- #
class FileProcessor:
    """
    A collection of processing layer functions that work with Json files.
    Reading and writing JSON data.
    ChangeLog: (Who, When, What)
        TONeill, 05/28/2025, Created class
    """

    @staticmethod
    def read_data_from_file(file_name: str, student_data: list[dict[str,
str]]) -> list[dict[str, str]]:
        """
        This function reads data from a JSON file and stores it in a list of
dictionaries.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages(message = "JSON file must exist before
running this script...", error = e)
        except json.decoder.JSONDecodeError as e:
            IO.output_error_messages(message = "File must be a valid JSON
file...", error = e)
        except Exception as e:
            IO.output_error_messages(message = "There was a non-specific
error...", error = e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list[dict[str,
str]]) -> None:
        """
        This function writes data to a JSON file.
```

```
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
        except TypeError as e:
            IO.output_error_messages(message = "File must be a valid JSON
file...", error = e)
        except Exception as e:
            IO.output_error_messages(message = "There was a non-specific
error...", error = e)
        finally:
            if file.closed == False:
                file.close()


# Presentation ---------------------------------------------------------------
--------------- #
class IO:
    """
    A collection of presentation layer functions that manage user input and
output.
    ChangeLog: (Who, When, What)
        TONeill, 05/28/2025, Created class
    """

    @staticmethod
    def output_error_messages(message: str, error: Exception = None) -> None:
        """
        This function prints technical details when an error occurs.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod
    def output_menu(menu: str) -> None:
        """
        This function displays the menu of choices to the user.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        print() # Adding spaces for appearance
        print(menu)
        print() # Adding spaces for appearance

    @staticmethod
    def input_menu_choice() -> str:
        """
        This function captures the menu choice made by the user.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
```

```
        """
        choice: str = ""
        try:
            choice = input("Please select from the menu: ")
            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(message = e.__str__())  # Not passing
the exception object to avoid the technical message

        return choice

    @staticmethod
    def output_student_courses(student_data: list[dict[str, str]]) -> None:
        """
        This function displays all students and their respective courses.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        MESSAGE: str = "{}, {}, {}"
        print()
        print("-" * 50)
        for student in student_data:
            print(MESSAGE.format(student["FirstName"], student["LastName"],
student["CourseName"]))
        print("-" * 50)
        print()

    @staticmethod
    def input_student_data(student_data: list[dict[str, str]]) ->
list[dict[str, str]]:
        """
        This function collects the first name, last name, and course name to
register a student.
        ChangeLog: (Who, When, What)
            TONeill, 05/28/2025, Created class
        """
        try:
            # Input the data
            student_first_name = input("Student First Name: ")
            if not student_first_name.isalpha():
                raise ValueError("Name must only contain letters.")

            student_last_name = input("Student Last Name: ")
            if not student_last_name.isalpha():
                raise ValueError("Name must only contain letters.")

            course_name = input("Course Name: ")

            student = {"FirstName": student_first_name,
                       "LastName": student_last_name,
                       "CourseName": course_name}
            student_data.append(student)
        except ValueError as e:
            IO.output_error_messages(message = e.__str__())  # Not passing
the exception object to avoid the technical message
        finally:
```

```
            return student_data



# --- End of function definitions --- #


# --- Main Body of Script --- #
students = FileProcessor.read_data_from_file(file_name = FILE_NAME,
student_data = students)

# Repeat the following tasks
while True:
    IO.output_menu(menu = MENU)

    menu_choice = IO.input_menu_choice()

    if menu_choice == "1":  # Register a student for a course
        students = IO.input_student_data(student_data = students)
        continue

    elif menu_choice == "2": # Show current data
        IO.output_student_courses(student_data = students)
        continue

    elif menu_choice == "3": # Save data to file
        FileProcessor.write_data_to_file(file_name = FILE_NAME, student_data
= students)
        continue

    elif menu_choice == "4": # GTFO
        print("Thank you for using this program.")
        break
```