Rachael Leahy
5 September 2025
IT FDN 110A
Assignment 07
https://github.com/outlawthruhiker/IntroToProg-Python-Mod07

Functions

Introduction

This week's course material delved further into objects, classes, and attributes, as well as introduced constructors, data validation, and private attributes.

The assignment tasked us with modifying a previous script to include instance classes (Person and Student).

Creating the Script

To accomplish this assignment, I did the following.

1. Named the .py file "Assignment07.py" (Figure 1).

Figure 1. Assignment is saved as "Assignment06.py"

2. Populated an appropriate script header (Figure 2).

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   Rachael Leahy, 9/1/2025, Updated student object handling and file processing from starter script
# ------------------------------------------------------------------------------------------ #
import json
```

Figure 2. An appropriately populated script header

3. Created two constants: “MENU” set to a multi-line string value and FILE_NAME” set to the string value, “Enrollments.json” (Figure 3).

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

Figure 3: Delineating the constants

4. Defined the data variables (Figure 4). Additional data variables are defined in their classes.

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

Figure 4: Delineating the variables

5. Created two @StatiMethods classes with descriptive document strings: a FileProcessor class for functions related to processing information and an IO class for functions related to input and output (Figures 5-6).

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

    ChangeLog: (Who, When, What)
    Rachael Leahy, 9/2/2025, Edited class to work with Person & Student classes
    """
```

Figure 5: Creating the FileProcessor class

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

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created Class
    RRoot,1.2.2030,Added menu output and input functions
    RRoot,1.3.2030,Added a function to display the data
    RRoot,1.4.2030,Added a function to display custom error messages
    Rachael Leahy, 9/5/2025, Updated to work with Student data
    """
```

Figure 6: Creating the IO class

6. Within the FileProcessor class, functions were created to read the data from the file (read_data_from_file) (Figure 7) and to write data to the file (write_data_to_file). Within the IO class, functions were created to handle error messaging (output_error_messages), print the menu (output_menu) (Figure 8), print the data in the file to the screen (output_student_courses), and enter student information (input_student_data). Where appropriate, headers were updated. Where no changes were made, headers were not edited. All functions have a descriptive docstring, the @staticmethod decorator, and call for error handling via the output_error_messages function as appropriate.

```python
@staticmethod  1 usage
def read_data_from_file(file_name: str):
    """ This function reads data from a json file and loads it into a list of dictionary rows
    then returns the list filled with student data.

    ChangeLog: (Who, When, What)
    Rachael Leahy, 9/2/2025, Edited function to work with Student data

    :param file_name: string data with name of file to read from

    :return: list
    """
    try:
        # Get a list of dictionary rows from the data file
        file = open(file_name, "r")

        # Convert the list of dictionary rows into a list of Student objects
        student_objects = []

        list_of_dictionary_data = json.load(file)
        for student in list_of_dictionary_data:
            student_objects.append(
                Student(
                    first_name=student["FirstName"],
                    last_name=student["LastName"],
                    course_name=student["CourseName"],
                )
            )
    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: "Error: There was a problem with reading the file.", e)
    finally:
        if file.closed == False:
            file.close()
```

Figure 7: The FileProcessor class, “read_data_from_file”

```python
@staticmethod  1 usage
def output_menu(menu: str):
    """ This function displays the menu of choices to the user

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created function


    :return: None
    """
    print()  # Adding extra space to make it look nicer.
    print(menu)
    print()  # Adding extra space to make it look nicer.
```

Figure 8: The IO class function, “out_menu”

7. Two instance classes were created. “Person” with attributes for first and last name (Figure 9), and a child class “Student” with the previous attributes, as well as course name (Figure 10).

```python
#-----Person Class-----
class Person:  1 usage
    """
    An instance class to define a person by their first and last names.
    """
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.__first_name = first_name
        self.__last_name = last_name

    @property  6 usages (3 dynamic)
    def first_name(self):
        return self.__first_name

    @first_name.setter  4 usages (3 dynamic)
    def first_name(self, value: str):
        if value.isalpha():  # validation: only letters
            self.__first_name = value
        else:
            raise ValueError("First name must only contain letters")

    @property  6 usages (3 dynamic)
    def last_name(self):
        return self.__last_name

    @last_name.setter  4 usages (3 dynamic)
    def last_name(self, value: str):
        if value.isalpha():  # validation: only letters
            self.__last_name = value
        else:
            raise ValueError("Last name must only contain letters")

    def __str__(self):
        return f"Person: {self.first_name} {self.last_name}"
```

Figure 9: The instance class “Person”

```python
#-----Student Class-----
class Student(Person):
    """
    An instance class to describe a student by their first and last names, as well as course name. First and last
    names are inherited from the "Person" class.
    """
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name, last_name)
        self.course_name = course_name

    @property  3 usages (1 dynamic)
    def course_name(self):
        return self.__course_name

    @course_name.setter  2 usages (1 dynamic)
    def course_name(self, value: str):
        if value != "": # validation: must have a value
            self.__course_name = value
        else:
            raise ValueError("Course Name cannot be blank")

    def __str__(self):
        return f"Student: {self.first_name} {self.last_name} {self.course_name}"
```

Figure 10: The instance class “Student”

8. Upon opening the script, it was to read the contents of “Enrollments.json” automatically into the list variable ‘students’ as a two-dimensional list and convert the data into a list of student object rows (Figure 11).

```python
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME)
```

Figure 11. Reading data from file via the FileProcessor class and read_data_from_file function

9. Use the class.functions to:
    a. Allow the user to pick a menu choice (1, 2, 3, 4).
    b. Menu choice 1.) Prompt user to enter the student’s first name, last name, and course name. These entries are then stored in the respective variables. , as well as added to the dictionary named student_data. Student_data is then appended to students.
    c. Menu choice 2.) The data entered are printed to the screen as a string of comma-separated values for each row collected in the ‘students’ variable.
    d. Menu choice 3.) The entered data are saved to Enrollments.json and the data saved in the file are displayed to the user.

e. Menu choice 4.) The program prints, “Program Ended” and the program ends (Figure 12).

```
# Input user data
if menu_choice == "1":  # This will not work if it is an integer!
    students = IO.input_student_data(student_data=students)
    continue

# Present the current data
elif menu_choice == "2":
    IO.output_student_and_course_names(students)
    continue

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

# Stop the loop
elif menu_choice == "4":
    break  # out of the loop
else:
    print("Please only choose option 1, 2, or 3")

int("Program Ended")
```

Figure 12: Allow the user to select menu choices and enact the options therein via the appropriate class.function

10. Error handling is accomplished via a call to a function specifically designed to handle errors (Refer to Figures 7-9).

Testing the Script

1. To test the script in PyCharm, I first saved the final draft and then ran it.
    a. After choosing menu option 1.), I was prompted to enter a first and last name, and a course name (Figure 13) and then prompted to make another selection.

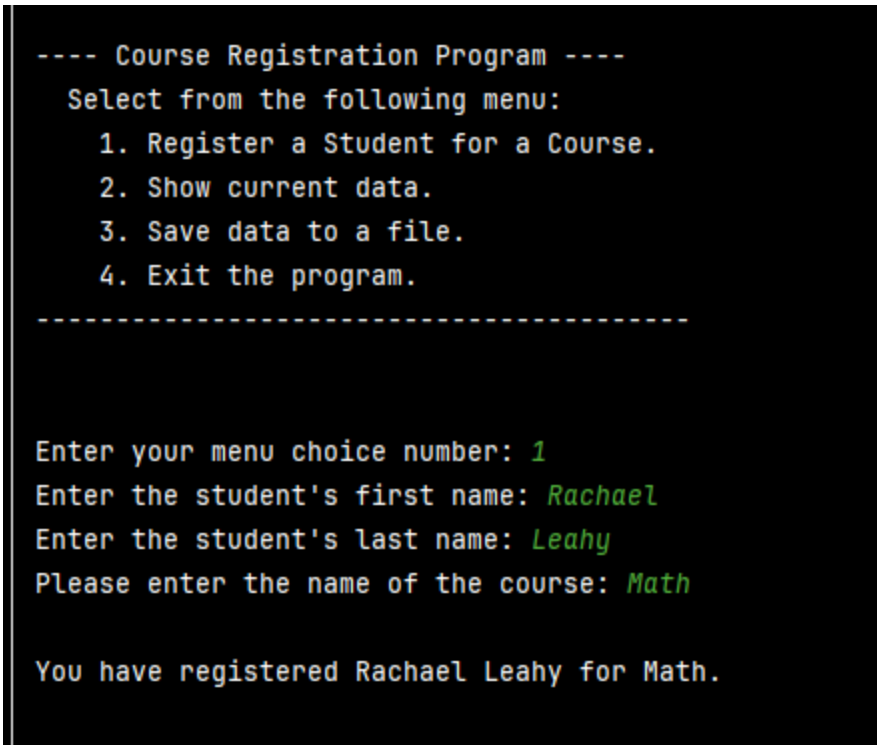

Figure 13: Screenshot showing the input prompts and print command output in PyCharm.

b. After choosing menu option 2.), the entered data were printed to the screen and I was prompted to make another selection (Figure 14).

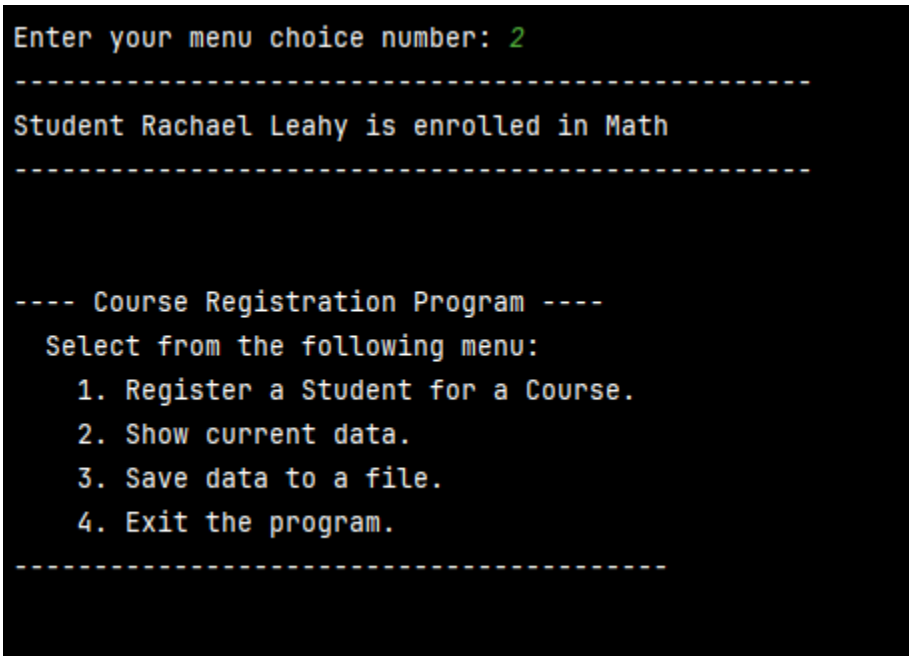

Figure 14: Screenshot showing the input data printed to the screen after selecting menu choice 2.

c. After choosing menu option 3.), the entered data are supposed to be printed to the screen. I am stuck here and I'm at the point where I just need to get this in and stop

stressing about it. The error I'm getting is below (Figure 15). If you could help me figure out where I've gone wildly astray, I would be grateful.

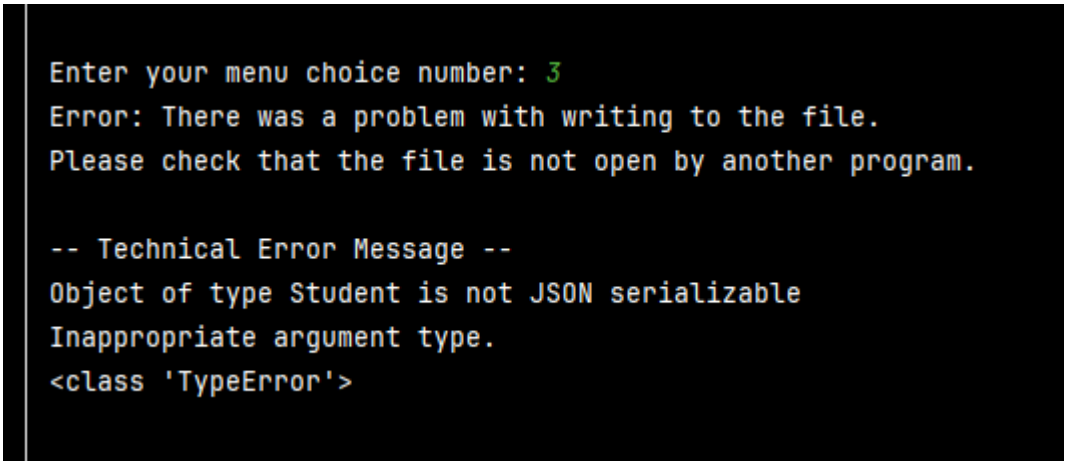

Figure 15: Screenshot showing the entered data printed to the screen after selecting menu choice 3.

2. To test the script in the Command Line, I first navigated to the appropriate directory and then entered the command 'python "Assignemnt07.py"' both to open the appropriate file to be read and to ensure the computer interpreted the file as a python script. I received a similar error!

Summary

I don't have a great summary or any insight this week. I'm frustrated that I can't figure this out and I'm frustrated that I'm running so far behind the class (for which you have my apologies). Thank you for the note about the hard deadline for the final assignment – I will be sure to have *something* in on time.

At this point in my learning, I feel too novice to use Python for anything. Do you have any recommendations on next steps to continue learning (and to reinforce what the class has covered) such that someday I might feel somewhat more proficient in this language?