Sabrina Fechtner

17 November 2023

IT FDN 110 B Au 23

Assignment 06

https://github.com/sfechtner42/IntroToProg-Python

Interactive Course Registration Part 4

Introduction

This week I essentially reorganized last week's program into classes with their own functions. This demonstrates separation of concerns which is a design principle of computer programming. Each section addresses a concern which is defined as a set of information that affects the code.

Constant Variables

The constant variables have not been modified from previous submissions. However, TextIO is no longer needed (Figure 1).

```
import json

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

Figure 1: Defining the constant variables

Data Variables

Unlike the constants, the variables this week have significantly reduced in number. Here, I've only defined the variables used in the main program. Based on their placement in the script, these variables are global variables (Figure 2).

```
# Define the Data Variables
students: list = []
menu_choice: str
```

Figure 2: Data variables

Function Class 1: File Processing

The first concern I addressed in my script was fille processing. For this program, files are either: 1) opened/read and appended (Figure 3) or 2) created and written to (Figure 4). Therefore, I wrote two static method functions to handle these options for file processing. Note the exception handling I incorporated last week is now in these functions. Note in the document string of the read data function where this function will return the input student_data back to the user because the program will ultimately save this information into the JSON file.

```
@staticmethod
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads previous JSON file with student and course data

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated Function

    :return: Student Data
    """
    try:
        with open(file_name, "r") as file:
            student_data = json.load(file)
            print("Data successfully loaded from the file.")
    except FileNotFoundError:
        print("File not found, creating it...")
        with open(file_name, "w") as file:
            json.dump( obj: student_data,  fp: file)
            print("File created successfully.")
    except json.JSONDecodeError as e:
        print(f"Invalid JSON file: {e}. Resetting it...")
        with open(file_name, "w") as file:
            json.dump( obj: student_data,  fp: file)
            print("File reset successfully.")
    except Exception as e:
        print(f"An unexpected error occurred while loading data: {e}")

    return student_data
```

Figure 3: File Processing Function: Read Data

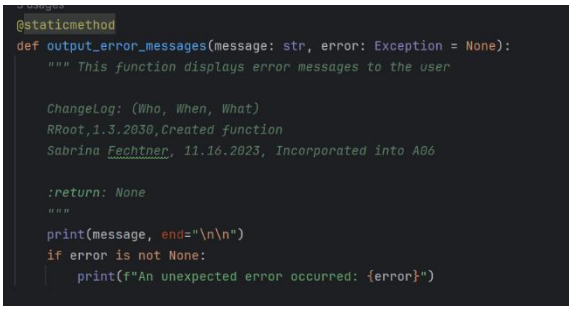
```
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes student and course data to JSON file

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated Function

    :return: None
    """
    try:
        with open(file_name, "w") as file:
            json.dump( obj: student_data,  fp: file)
            print("Data successfully written to the file.")
    except TypeError as e:
        IO.output_error_messages( message: "Please check that the data is a valid JSON format",  error: e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!",  error: e)
```

Figure 4: File Processing Function: Write Data

Function Class 2: Input/Output (IO) Data

The next concern I addressed was IO Data. As before, the program inputs are: 1) Ask the user to choose from the menu and 2) ask for user to enter first and last name, and course name. The program outputs are: 3) to display the menu and selection 4) display student information along with any prior student data

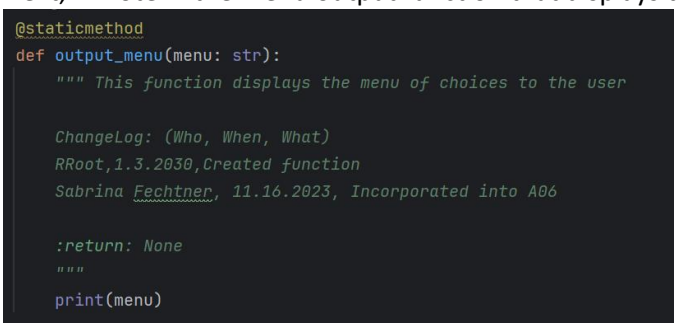
and 5) exception handle. Since error handling is incorporated into all the IO functions, I began by writing a handling function that can be incorporated into each function in this class (Figure 5)



```
@staticmethod
def output_error_messages(message: str, error: Exception = None):
    """ This function displays error messages to the user

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated into A06

    :return: None
    """
    print(message, end="\n\n")
    if error is not None:
        print(f"An unexpected error occurred: {error}")
```

Figure 5: IO Function: Error Handling

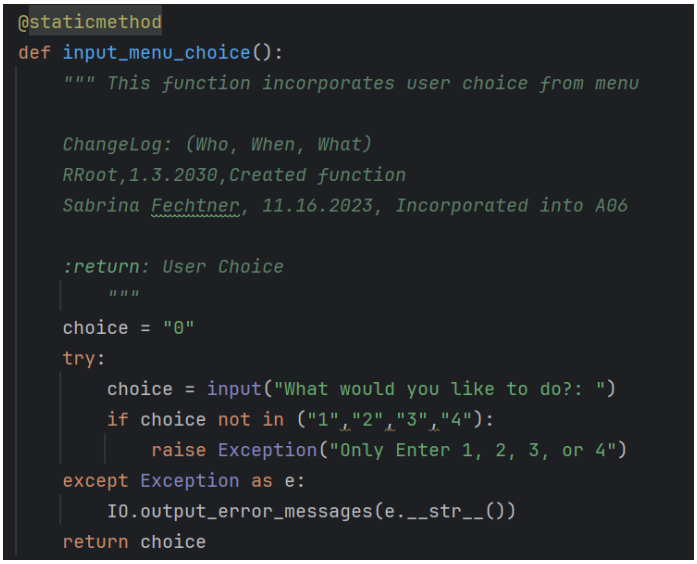
Next, I wrote in the menu output function that displays the user selection (Figure 6).



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

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated into A06

    :return: None
    """
    print(menu)
```

Figure 6: IO Function: Menu Output

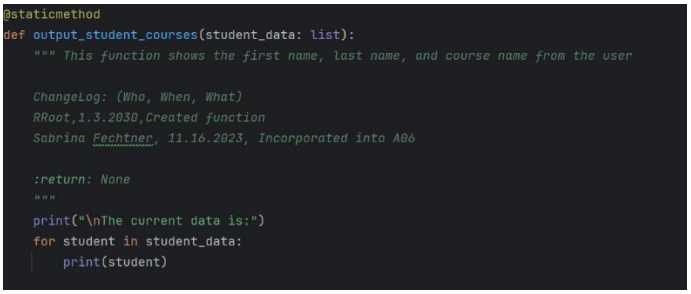
The menu input function is a bit more interesting than the menu output function. Here, the choice from the menu is preset to “0” which is not an option that the user can select from (see Figure 1). In doing so, this will ensure that the try-except block is always executed, and the user input is validated (Figure 7).



```
@staticmethod
def input_menu_choice():
    """ This function incorporates user choice from menu

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated into A06

    :return: User Choice
        """
    choice = "0"
    try:
        choice = input("What would you like to do?: ")
        if choice not in ("1","2","3","4"):
            raise Exception("Only Enter 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())
    return choice
```

Figure 7: IO Function: Menu Input

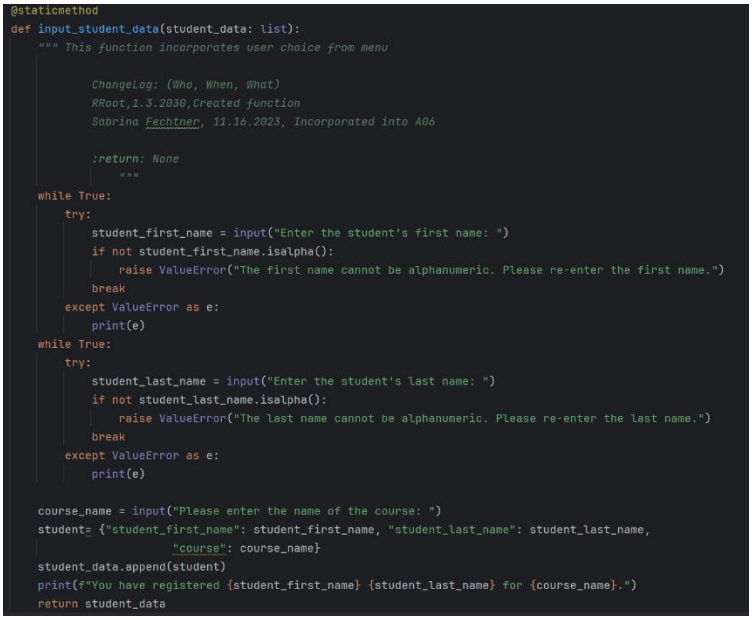
Next, function I wrote is an output student data function. This captures “option 2” from previous homework where the program will display entered data (Figure 8).



```
@staticmethod
def output_student_courses(student_data: list):
    """ This function shows the first name, last name, and course name from the user

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function
    Sabrina Fechtner, 11.16.2023, Incorporated into A06

    :return: None
    """
    print("\nThe current data is:")
    for student in student_data:
        print(student)
```

Figure 8: IO Function: Student data Output

Finally, the last function in the class is the input student data function. Again, this captures “option 1” from previous homework. Therefore, this script was taken directly from assignment 5 and incorporated into this function (Figure 9).



```
@staticmethod
def input_student_data(student_data: list):
    """ This function incorporates user choice from menu

            ChangeLog: (Who, When, What)
            RRoot,1.3.2030,Created function
            Sabrina Fechtner, 11.16.2023, Incorporated into A06

            :return: None
                """
    while True:
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The first name cannot be alphanumeric. Please re-enter the first name.")
            break
        except ValueError as e:
            print(e)
    while True:
        try:
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The last name cannot be alphanumeric. Please re-enter the last name.")
            break
        except ValueError as e:
            print(e)

    course_name = input("Please enter the name of the course: ")
    student= {"student_first_name": student_first_name, "student_last_name": student_last_name,
                      "course": course_name}
    student_data.append(student)
    print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    return student_data
```

Figure 9: IO Function: Student data Output

Main Program

Now the main function calls all the defined functions (Figure 10). To begin, I defined the students variable with the read_data function. This function requires to parameters: a file which I defined with the constant, and a list of student data which I defined as the student_data variable. Option 1 incorporated my input student data function. Option 2 used the output student courses function. Option 3 used the write data to file function. And option 4 ends the program by breaking out of the loop.

```python
#Main Program:
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

while True:
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    if menu_choice == "1": #get student input
        students = IO.input_student_data(student_data=students)
        continue

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

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

    elif menu_choice == "4":  # End the program
        break  # out of the while loop
```

Figure 10: Main Program

Final Result

I ran my program successfully in PyCharm and in the terminal (Figure 11) where I registered myself for Python 100 and my other cat for Python 101. The terminal run of my program saw my previous enrollments JSON file from last week and was able to append to it. Both the methods of running the code produced a .JSON file showing Cinnamon and I registered for our respective Python courses (Figure 12).

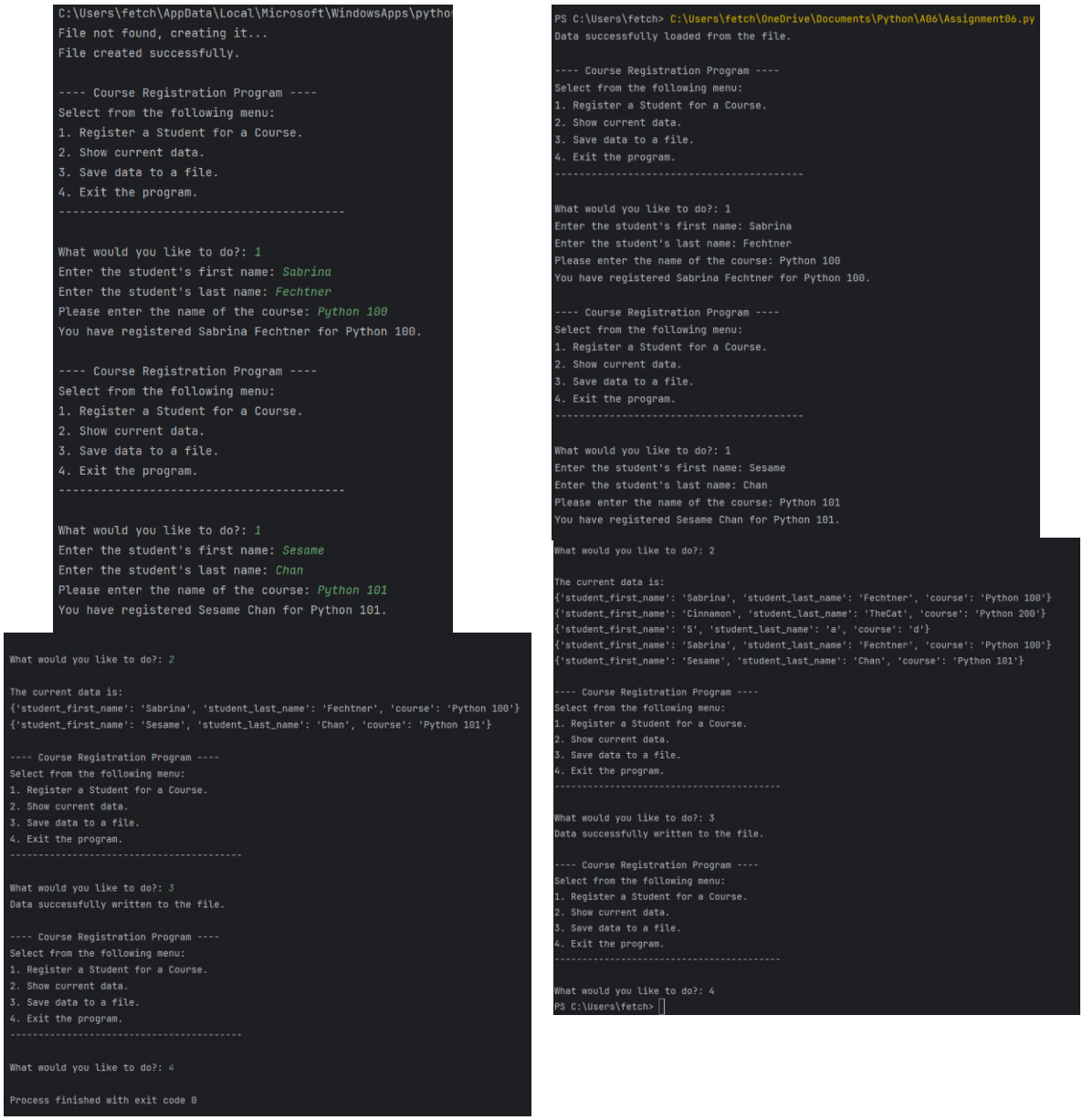


Figure 11: IO Function: Student data Output

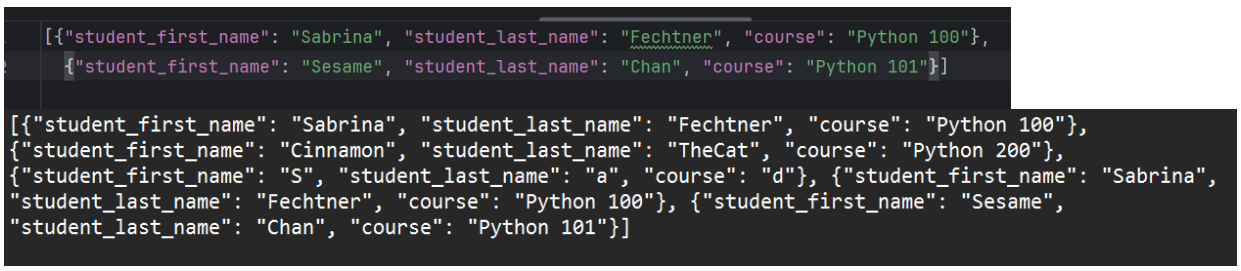


Figure 12: IO Function: Student data Output

Summary

This week, I modified my interactive course registration program from last week where I basically incorporated last week's script into new functions and called those in the main program. The result was a short script for the main program, but a new and large function defining section. There were many times where I tried simply copying/pasting last week's script into this week's assignment and would get a pylance or an indention error. So, this week I also learned the importance of indentation. Although this week's lecture was a bit overwhelming since this is the first time I've heard about these concepts, doing this assignment was a "lightbulb" moment for me.