Assignment 6: Functions

Rucha P. Nimbalkar

University of Washington

IT FDN 110: Fall 2024

Prof. Randall Root

Nov. 20, 2024

Abstract

Many a times the same code can be repeated several times in the script file. To avoid repetition, the same code can be used once in a function definition and the function can be called several times as per requirement in the code. Custom classes are used to store data and process the data using class methods (functions).

Keywords: Classes, Functions, Class Methods.

Assignment 6: Functions

Assignment 6 is the sixth assignment of the IT Fundamentals (IT FDN 110 A) course I am taking at University of Washington. This assignment will help me understand the usage of functions and classes. I will be re-organizing my code using functions in the assignment. This the link to my GitHub account for Assignment 06.

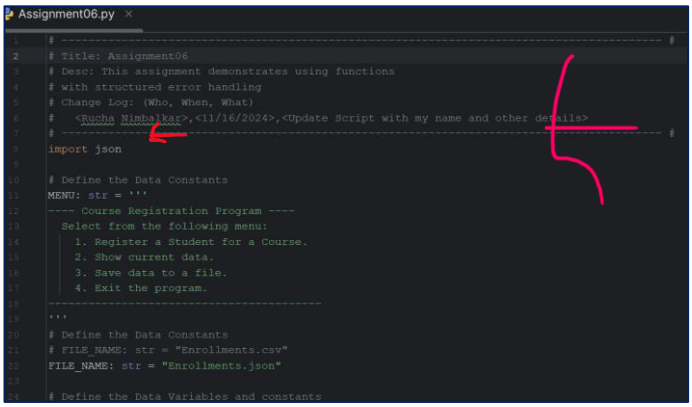
In this assignment, a Python script gives the user a menu of choices. It is an upgrade of Assignment 5 wherein we are performing similar tasks, but we are replacing the repetitive code with functions.

Reflection

In this week, I learnt about functions. According to Professor Root, functions (or methods) help in replacing repeated code lines in the script which can benefit everyone in the project team (2024). Furthermore, Professor Root also states that Classes can be used to store custom collection of data or process the data using functions (or class methods) (2024).

Program Summary

I began the program using the starter file provided for Assignment 6. I updated the file with my name in the script header change log. ***Figure 1*** *Assignment 05 Update Script Header*

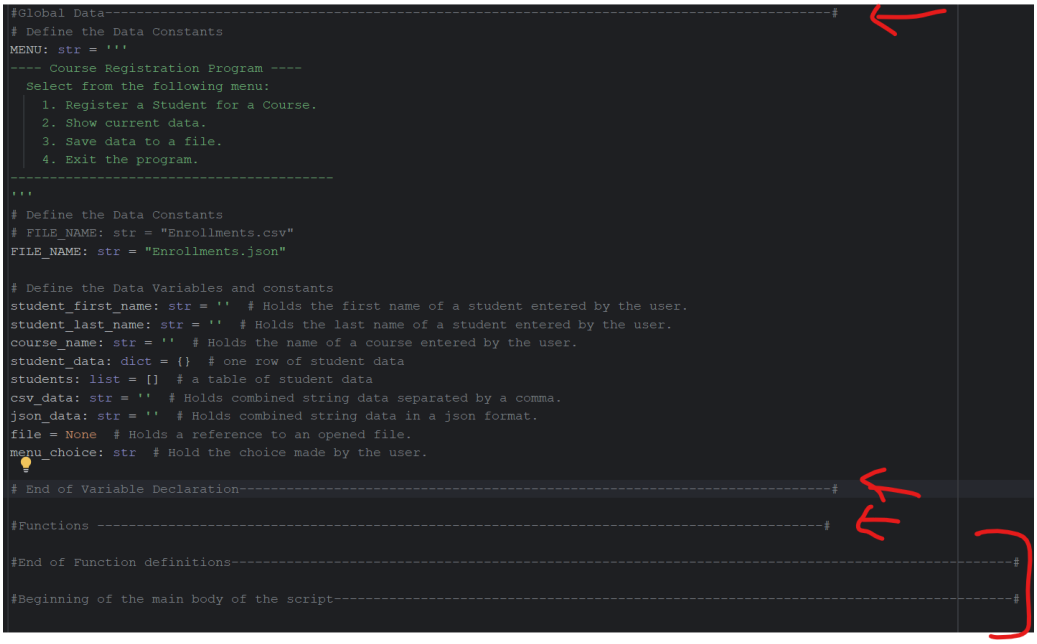


According to Professor Root, there are three layers of a programming script:

1. Data
2. Processing
3. Presentation.

After studying using the notes provided for this module, I added sections Global Variables, Function and the main body in comment form in my program as shown in figure 2.

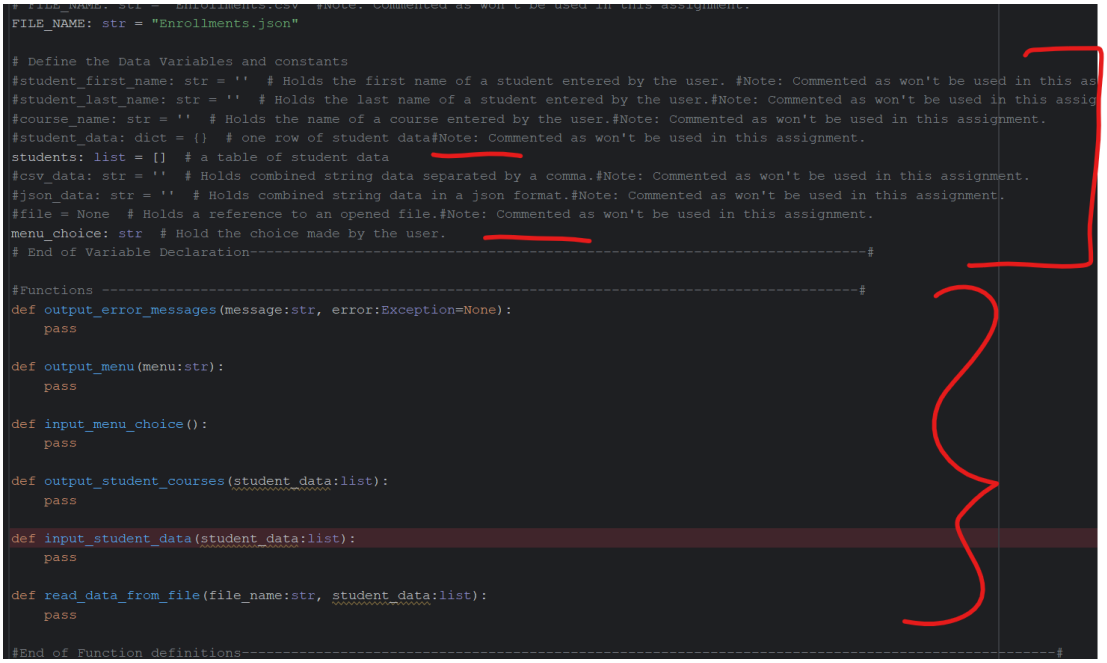
Figure 2 *Assignment 06 reorganize code into sections*



```
#Global Data--------------------------------------------------------------------------------------#
# 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.csv"
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.

# End of Variable Declaration----------------------------------------------------------------------#

#Functions ---------------------------------------------------------------------------------------#

#End of Function definitions---------------------------------------------------------------------------------------#

#Beginning of the main body of the script--------------------------------------------------------------------------#
```

I commented out the variables I would not be using in this assignment by adding a “#” symbol to the variable declarations. I added the functions using the “pass” keyword as a placeholder as I was not planning to start writing the definitions for the functions yet (please refer to figure 3).

Figure 3 *Assignment06 Commenting unused variables and adding function definitions using placeholder statement "pass"*



```
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
#student_first_name: str = ''  # Holds the first name of a student entered by the user. #Note: Commented as won't be used in this as
#student_last_name: str = ''  # Holds the last name of a student entered by the user.#Note: Commented as won't be used in this assig
#course_name: str = ''  # Holds the name of a course entered by the user.#Note: Commented as won't be used in this assignment.
#student_data: dict = {}  # one row of student data#Note: Commented as won't be used in this assignment.
students: list = []  # a table of student data
#csv_data: str = ''  # Holds combined string data separated by a comma.#Note: Commented as won't be used in this assignment.
#json_data: str = ''  # Holds combined string data in a json format.#Note: Commented as won't be used in this assignment.
#file = None  # Holds a reference to an opened file.#Note: Commented as won't be used in this assignment.
menu_choice: str  # Hold the choice made by the user.
# End of Variable Declaration----------------------------------------------------------------------#

#Functions ---------------------------------------------------------------------------------------#
def output_error_messages(message:str, error:Exception=None):
    pass

def output_menu(menu:str):
    pass

def input_menu_choice():
    pass

def output_student_courses(student_data:list):
    pass

def input_student_data(student_data:list):
    pass

def read_data_from_file(file_name:str, student_data:list):
    pass

#End of Function definitions----------------------------------------------------------------------------------------------------#
```

When the script execution begins, the contents of the file “Enrollments.json” are read into the students (two-dimensional list table) variable, so I started working on making adjustments to add code in the function definition for read_data_from_file() and adding a function call in the main body of the code for the same function (please refer to figure 4). I then moved the print (MENU) code statement to the output_menu(menu) function and updated the print statement parameter and function call in the main body of the code. (please refer to figure 5).

Figure 4 *Assignment 06 Add function definition to read data from the file and then call the function*

```
def read_data_from_file(file_name:str, student_data:list):  1 usage
    try:
        file = open(file_name, "r") #Open the file in read mode
        student_data = json.load(file)#Load the file contents
        print(student_data)
        file.close()  # close the file
        # CSV Answer #Not applicable in this assignment,
        # for row in file.readlines():
        #     # Transform the data from the file
        #     student_data = row.split(',')
        #     student_data = {"FirstName": student_data[0],
        #                     "LastName": student_data[1],
        #                     "CourseName": student_data[2].strip()}
        #     # Load it into our collection (list of lists)
        #     students.append(student_data)

    except Exception as e:
        print("Error: There was a problem with reading the file.")
        print("Please check that the file exists and that it is in a json format.")
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    finally:
        if file.closed == False:
            file.close()
    return student_data
```

```
# When the program starts, read the file data into a list of lists (table)
# Call function to extract the data from the file
students = read_data_from_file(file_name=FILE_NAME, student_data=students)
```

Figure 5 *Assignment 06 Add function definition to display menu and then call the same function*

```
def output_student_courses(student_data:list):  1 usage
    # Process the data to create and display a custom message
    print("-" * 40)
    for student in student_data:
        print(f'Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 40)
```

```
 # Present the current data
 elif menu_choice == "2":
     #Call function to display student course
     output_student_courses(student_data=students)
     continue
```

I reorganized the code for the function input_student_data(student_data:list) as shown in figure 6. I added the function call for menu_choice “1”. I made sure to run my code each time I added the function definitions to make sure it was still working.

Figure 6 *Assignment 06 Add function definition to accept user input in the user input_student_data() and then add a function call in the main body under the if condition menu_choice = “1”*

```
def input_student_data(student_data:list):  1 usage
    student: dict = {}
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")
        student = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        student_data.append(student)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        print(e)  # Prints the custom message
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    except Exception as e:
        print("Error: There was a problem with your entered data.")
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    return student_data
```

```
# Input user data
if menu_choice == "1":  # This will not work if it is an integer!
    students = input_student_data(student_data=students) #call function to accept new student data
    continue
```

I reorganized the code for the function input_menu_choice () and write_data_to_file(file_name : str , student_data :list). I added the function call for the former function after the output_menu(menu:str) function.

Figure 7 *Assignment 06 Add function definition to accept user choice and call the function in main body after the function call output_menu(MENU).*

```
def input_menu_choice():  1 usage
    user_choice = input("What would you like to do: ")
    return user_choice
```

```
# Present the menu of choices
output_menu(MENU)
menu_choice=input_menu_choice() #call the function to accept user choice after displaying the menu
```

For the later function I added the function call in the if condition when menu_choice equals “3”. I tested my code again to make sure it was working smoothly accepting user choice and saving(writing) data to the file.

Figure 8 *Assignment 06 Add function definition to write_ data_to_ file() and call the function in main body in the if condition for menu_choice “3”*

```
def write_data_to_file(file_name:str, student_data:list):  1 usage
    try:
        file = open(file_name, "w")
        # CSV answer
        # for student in students:
        #     csv_data = f'{student["FirstName"]},{student["LastName"]},{student["CourseName"]}\n'
        #     file.write(csv_data)

        # # JSON answer
        json.dump(student_data, file) #Write the contents in the file

        file.close()
        print("The following data was saved to file!")
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    except Exception as e:
        if file.closed == False:
            file.close()
        print("Error: There was a problem with writing to the file.")
        print("Please check that the file is not open by another program.")
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
```

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

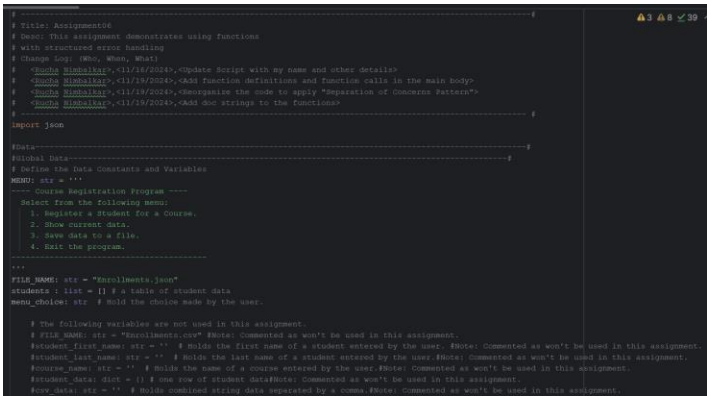
I added the definition for output_error_messages(message:str, error:Exception = None) and made sure to add function call for this function in the exception handling sections in the input_student_data(), read_data_from_file() and write_data_to_file() functions.

Figure 9 *Assignment 06 Add function definition to output_error_messages(message:str,e:Exception) and call the function in each function's error/exception handling section*

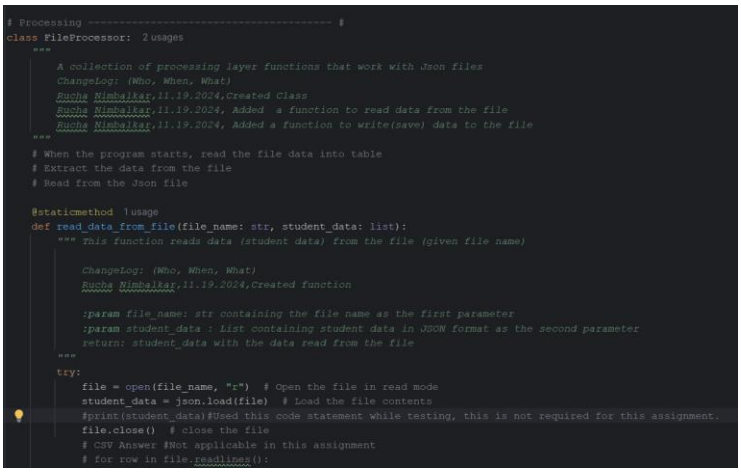
```
#Functions ------------------------------------------------------------
def output_error_messages(message:str , e:Exception):  5 usages
    print(message)  # Prints the message
    print("-- Technical Error Message -- ")
    print(e)  # Prints the custom message
    print(e.__doc__)
    print(e.__str__())
```

After that I started working on including the two classes in my program. Referring to the lab 03 code and class notes, I knew this was the important part of this assignment wherein I would process the data and present the data using the classes. I created two classes: FileProcessor and IO. The FileProcessor class processes the data from the file (read data from the file and write data to the file) while the IO class presents (and works with) the data on the console(output menu, input the user choice, output student data and output error messages). I moved the corresponding functions in the classes which then became the class methods. The function calls were then modified with "classname.methodname()" in the script. I made sure to remove the spaces and reorganize my code by removing white spaces to make it legible and systematic. I applied the Separation of Concerns (SoC) principle which I learnt in lab 03 of this module by writing docstrings for the class methods. I took screenshots of my entire script (please refer to figure 10).

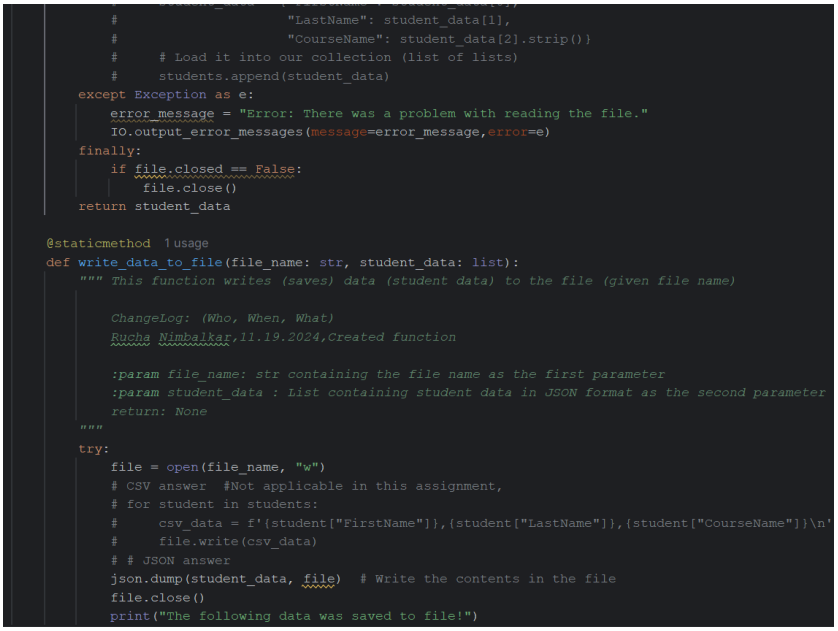
Figure 10 *Assignment 06 with SoC to make the code legible.*



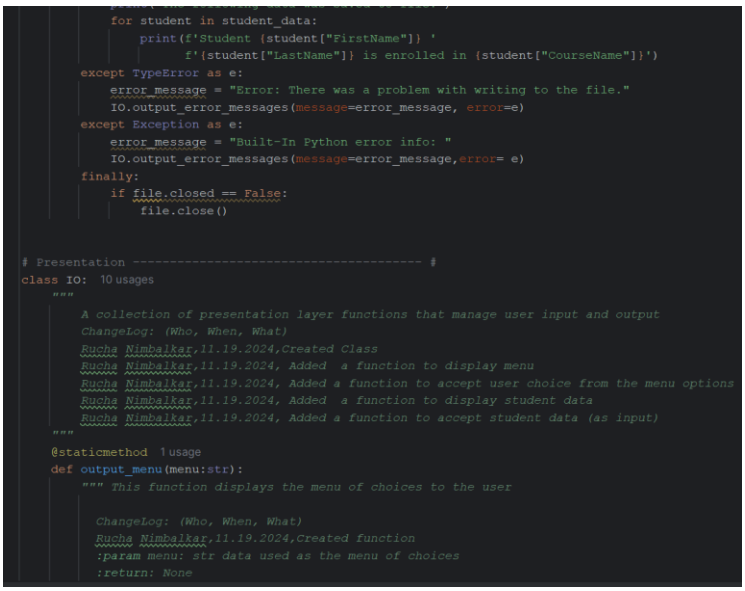
```
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Ruchs Nimbalkar,11/16/2024,<Update Script with my name and other details>
#   Ruchs Nimbalkar,11/19/2024,<Add function definitions and function calls in the main body>
#   Ruchs Nimbalkar,11/19/2024,<Reorganize the code to apply "Separation of Concerns Pattern">
#   Ruchs Nimbalkar,11/19/2024,<Add doc strings to the functions>
import json

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

    # The following variables are not used in this assignment.
    # FILE_NAME: str = "Enrollments.csv" #Note: Commented as won't be used in this assignment.
    #student_first_name: str = ''  # Holds the first name of a student entered by the user. #Note: Commented as won't be used in this assignment.
    #student_last_name: str = ''  # Holds the last name of a student entered by the user.#Note: Commented as won't be used in this assignment.
    #course_name: str = ''  # Holds the name of a course entered by the user.#Note: Commented as won't be used in this assignment.
    #student_data: dict = {} # one row of student data#Note: Commented as won't be used in this assignment.
    #csv_data: str = ''  # Holds combined string data separated by a comma.#Note: Commented as won't be used in this assignment.
```



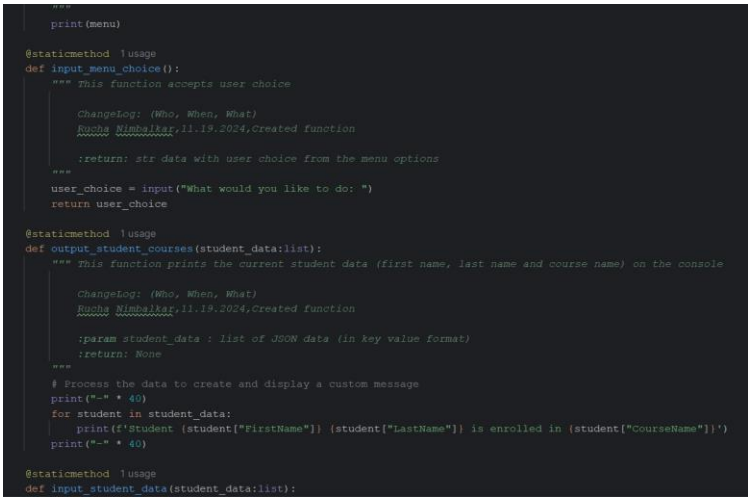
```
# Processing ------------------------------------- #
class FileProcessor:  2 usages
    """
        A collection of processing layer functions that work with Json files
        ChangeLog: (Who, When, What)
        Ruchs Nimbalkar,11.19.2024,Created Class
        Ruchs Nimbalkar,11.19.2024, Added  a function to read data from the file
        Ruchs Nimbalkar,11.19.2024, Added a function to write(save) data to the file
    """
    # When the program starts, read the file data into table
    # Extract the data from the file
    # Read from the Json file

    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data (student data) from the file (given file name)

            ChangeLog: (Who, When, What)
            Ruchs Nimbalkar,11.19.2024,Created function

            :param file_name: str containing the file name as the first parameter
            :param student_data : List containing student data in JSON format as the second parameter
            return: student_data with the data read from the file
        """
        try:
            file = open(file_name, "r")  # Open the file in read mode
            student_data = json.load(file)  # Load the file contents
            #print(student_data)#Used this code statement while testing, this is not required for this assignment.
            file.close()  # close the file
            # CSV Answer #Not applicable in this assignment
            # for row in file.readlines():
```



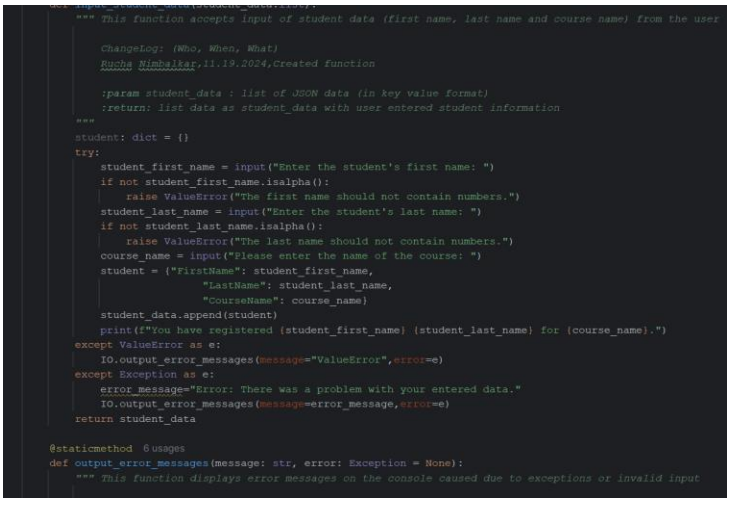
```
                #                     "LastName": student_data[1],
                #                     "CourseName": student_data[2].strip()}
                #     # Load it into our collection (list of lists)
                #     students.append(student_data)
            except Exception as e:
                error_message = "Error: There was a problem with reading the file."
                IO.output_error_messages(message=error_message,error=e)
            finally:
                if file.closed == False:
                    file.close()
            return student_data

    @staticmethod  1 usage
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes (saves) data (student data) to the file (given file name)

            ChangeLog: (Who, When, What)
            Ruchs Nimbalkar,11.19.2024,Created function

            :param file_name: str containing the file name as the first parameter
            :param student_data : List containing student data in JSON format as the second parameter
            return: None
        """
        try:
            file = open(file_name, "w")
            # CSV answer  #Not applicable in this assignment,
            # for student in students:
            #     csv_data = f'{student["FirstName"]},{student["LastName"]},{student["CourseName"]}\n'
            #     file.write(csv_data)
            # # JSON answer
            json.dump(student_data, file)  # Write the contents in the file
            file.close()
            print("The following data was saved to file!")
```



```
            for student in student_data:
                print(f'Student {student["FirstName"]} '
                      f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        except TypeError as e:
            error_message = "Error: There was a problem with writing to the file."
            IO.output_error_messages(message=error_message, error=e)
        except Exception as e:
            error_message = "Built-In Python error info: "
            IO.output_error_messages(message=error_message,error= e)
        finally:
            if file.closed == False:
                file.close()


# Presentation ------------------------------------ #
class IO:  10 usages
    """
        A collection of presentation layer functions that manage user input and output
        ChangeLog: (Who, When, What)
        Ruchs Nimbalkar,11.19.2024,Created Class
        Ruchs Nimbalkar,11.19.2024, Added  a function to display menu
        Ruchs Nimbalkar,11.19.2024, Added a function to accept user choice from the menu options
        Ruchs Nimbalkar,11.19.2024, Added  a function to display student data
        Ruchs Nimbalkar,11.19.2024, Added a function to accept student data (as input)
    """
    @staticmethod  1 usage
    def output_menu(menu:str):
        """ This function displays the menu of choices to the user

            ChangeLog: (Who, When, What)
            Ruchs Nimbalkar,11.19.2024,Created function
            :param menu: str data used as the menu of choices
            :return: None
```



```
    """
    print(menu)

@staticmethod  1 usage
def input_menu_choice():
    """ This function accepts user choice

        ChangeLog: (Who, When, What)
        Ruchs Nimbalkar,11.19.2024,Created function

        :return: str data with user choice from the menu options
    """
    user_choice = input("What would you like to do: ")
    return user_choice

@staticmethod  1 usage
def output_student_courses(student_data:list):
    """ This function prints the current student data (first name, last name and course name) on the console

        ChangeLog: (Who, When, What)
        Ruchs Nimbalkar,11.19.2024,Created function

        :param student_data : list of JSON data (in key value format)
        :return: None
    """
    # Process the data to create and display a custom message
    print("-" * 40)
    for student in student_data:
        print(f'Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 40)

@staticmethod  1 usage
def input_student_data(student_data:list):
```



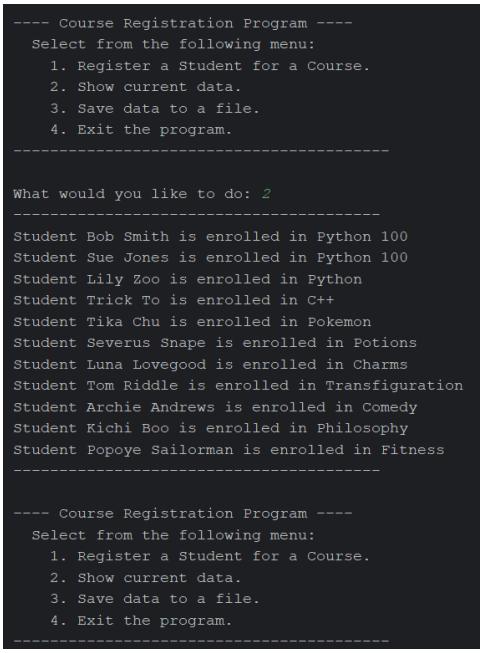
```
    """ This function accepts input of student data (first name, last name and course name) from the user

        ChangeLog: (Who, When, What)
        Ruchs Nimbalkar,11.19.2024,Created function

        :param student_data : list of JSON data (in key value format)
        :return: list data as student_data with user entered student information
    """
    student: dict = {}
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")
        student = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        student_data.append(student)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="ValueError",error=e)
    except Exception as e:
        error_message="Error: There was a problem with your entered data."
        IO.output_error_messages(message=error_message,error=e)
    return student_data

@staticmethod  6 usages
def output_error_messages(message: str, error: Exception = None):
    """ This function displays error messages on the console caused due to exceptions or invalid input
```

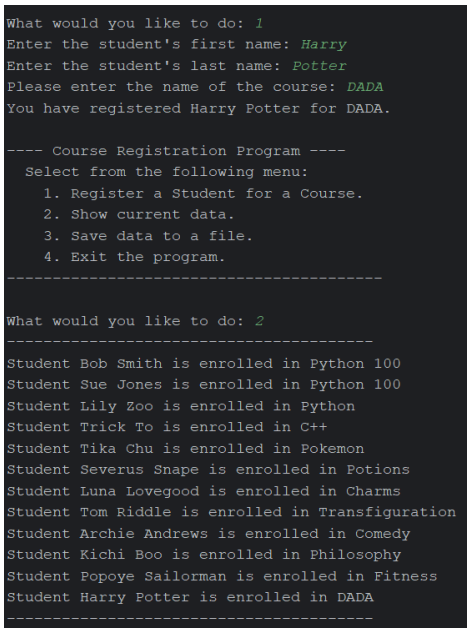
```
            ChangeLog: (Who, When, What)
            Rucha Nimbalkar,11.19.2024,Created function

            :param message : str containing error message as the first parameter
            :param error : Exception containing Exception occurred (default value is none)
            :return: None
        """
        print(message)  # Prints the message
        if error is not None: #Print the technical error message for Exception occurred.
            print("-- Technical Error Message -- ")
            print(error)  # Prints the custom message
            print(error.__doc__)
            print(error.__str__())
        return

#End of Function definitions-----------------------------------------------------------------------------------

#Beginning of the main body of the script----------------------------------------------------------------------

# When the program starts, read the file data into a list of lists (table)
# Call function to extract the data from the file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
while (True):

    # Present the menu of choices
    IO.output_menu(MENU)
    menu_choice=IO.input_menu_choice() #call the function to accept user choice after displaying the menu
```

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

    # Present the current data
    elif menu_choice == "2":
        #Call function to display student course
        IO.output_student_courses(student_data=students)
        continue

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

    # Stop the loop
    elif menu_choice == "4":
        print("Exiting the program")
        break  # out of the loop

    else:
        error_message="Please enter a valid option 1, 2 , 3 or 4"
        IO.output_error_messages(message=error_message)

print("Program Ended")#Program terminated
#End of the main body of the script-----------------------------------------------------------------------------
```

SoC is a design principle that makes the code readable (Root, 2024). I ran the code in PyCharm and in command terminal and it ran successfully (please refer to figures 11 and 12).

Figure 11 *Assignment 06 Execution output in PyCharm console*





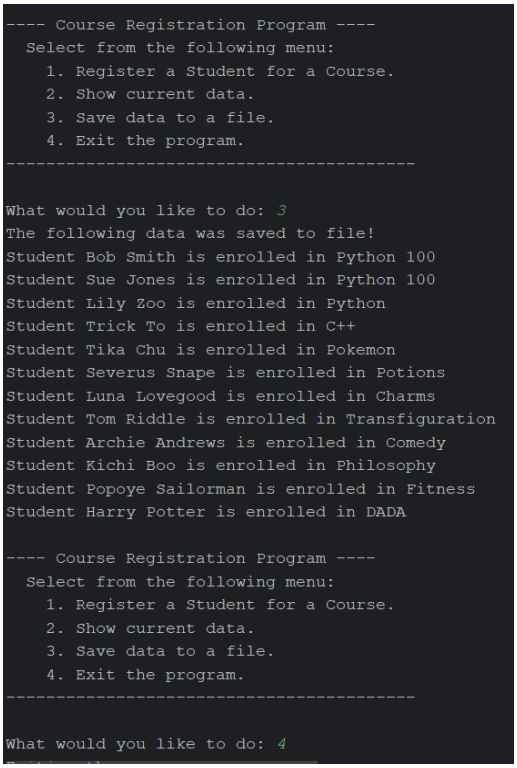
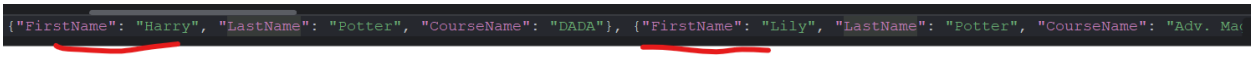


Figure 12 *Assignment 06 Execution output in command terminal*

To verify if the student data was saved in the file, I opened the file and confirmed that the students were saved in “Enrollments.json” (please refer to figure 13 below).

Figure 13 *Assignment 06 Data was saved in Enrollments.json file*



Conclusion

This assignment helped me understand that re-organizing the code using functions and classes makes the code readable and systematic. It helps other developers understand and read my code and I learnt to make my code modular and reusable.

References

Randall, R.(n.d.). *IT Fundamentals 110 A* [MOOC]. University of Washington. Foundations of Programming (Python) - UW Professional & Continuing Education