Renato Felicio
November 17, 2024
IT FDN 110 A
Assignment 06

Creating Python Scripts – Functions

Introduction

This Assignment 06 consists in creating a Python program that uses constants, variables, and print statements to display a message about a student's registration for a Python course. Building on the knowledge from Assignment 05, this assignment introduces new concepts, such as use of use of functions, classes, and using the separation of concerns pattern.

Preparation for this assignment

To prepare for this assignment, I reviewed the "Module 06 Notes" (Reference 1), completed the three lab examples, and watched both the Module 06 videos available on Washington University's Canvas platform (Figure 1) and the recommended external videos in References 2 through 3. Through these materials, I learnt how to work with functions, including their parameters and arguments, overloaded functions, and return values. I also gained an understanding of how to work with classes and apply the Separation of Concerns (SoC) pattern. I learned that functions and classes are fundamental building blocks of the SoC principle in Python programming.

I also learned the common practice of including a header at the beginning of a class or function, known as a "docstring" in Python. Including additional notes within the docstring can be helpful for developers, providing clear explanations and context for the code.

Mod06 Videos

Module	Name	Link
6	Demo01 - Using Functions	https://www.youtube.com/watch?v=8tZdqlArsbc 
6	Demo02 - Using Arguments	https://www.youtube.com/watch?v=a6dmUlaNB00 
6	Demo02 - Using Returns	https://www.youtube.com/watch?v=tTPdCTUsDb8 
6	Demo03 - Using Classes	https://www.youtube.com/watch?v=TAD_BczzOI0 
6	Demo04- Separation Of Concerns	https://www.youtube.com/watch?v=fapZdUP-vdw 

Figure 1 – Mod06-Videos

Python Scripting

I began by reviewing the provided Assignment06-Starter.py file and use it as the starting point for my script. The objective was to implement the use of function classes and organize the script according to the SoC pattern. The script should allow the user to register multiple students, display their inputs, and save the data back into a '.json' file containing all registered students. Additionally, structured error handling should be implemented when reading the file into dictionary rows, when the user enters first and last names, and when writing the dictionary rows back into the file.

The functions should be grouped into two classes: FileProcessor and IO The FileProcessor class contains functions responsible for reading data from and writing data to a .json file. The IO class includes functions that manage user interactions, such as collecting menu options, gathering input data, displaying messages, and printing error notifications.

Two global variables are defined: menu_choice (a string) to store the user's selected option, and students (a list) to hold the information of all registered students. These variables are initialized with an empty string and an empty list, respectively.

I began by organizing the script into major sections: the header, imports, variable and constant definitions, class and function definitions, and the main body. The functions were grouped into classes based on their roles, aligning with the data processing and presentation layers of concern.

```
# Header
# Import
# Global Data
# Data Layer
        Definition of data constants
        Definition of variable

# Class Definition

        Processing data layer
                Class FileProcessor created
                        Function read_data_from_file created
                        Function write_data_to_file created

        Presentation data layer

                Class IO created

                        Function output_error_messages created
                        Function input_menu_choice
                        Function output_student_courses
                        input_student_data

# Main body of the script
```

The script is displayed in separated figures, split across Figure 2 to Figure 5 according its correspondent sections.

Header, import, constants and variables script parts are presented in the Figure 2.

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Renato Felicio,11/16/2024,Created Script
#   <Your Name Here>,<Date>,<Activity>
# ------------------------------------------------------------------------------------------ #

# Import section
import json
from typing import TextIO

# Global Data Layer

# 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" # Constant holds the name of the file with students data

# Define the Data Variables and constants
students: list=[] # This variable holds the information of all registered students.
menu_choice: str=''  # It holds the user choice.

```

Figure 2 – Python Script Header Import and Variables

Processing data layer is presented in Figure 3.

```python

# Class definition

# Processing Data Layer
class FileProcessor:  2 usages
    """
        A collection of processing layer functions that work with json files

        ChangeLog: (Who, When, What)
        Renato Felicio,11/16/2024,Created Class
        """
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file into a list of dictionary rows

            Notes:
            - Data sent to the student_data parameter will be overwritten.

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :param file_name: string with the name of the file we are reading
            :param student_data: list of dictionary rows we are adding data to
            :return: list of dictionary rows filled with data
        """
        try:
            file: TextIO = open(file_name, "r")  # Open the JSON file for reading
            student_data: list = json.load(file) # File data is loaded into a table
            # Now 'student_data' contains the parsed JSON data as a Python list of dictionaries
            file.close()
        except FileNotFoundError as e:  # Handles error in case there is no initial file
            IO.output_error_messages("Data file must exist before running this script!")
            file = open(FILE_NAME, "w")  # Creates an empty initial file, in case of file not found
            IO.output_error_messages("Empty file was created!\n")
        except Exception as e:
            IO.output_error_messages("Error: There was a problem with reading the file.", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    @staticmethod  1 usage
    def write_data_to_file(file_name: [No Title], student_data: list):
        """ This function writes data to a json file from a list of dictionary rows

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :param file_name: string with the name of the file we are writing to
            :param student_data: list of dictionary rows containing our data
            :return: None
        """
        try:
            file: TextIO = open(file_name, "w")
            json.dump(student_data, file)  # It writes the list of dictionaries into a json file
            file.close()
        except Exception as e: # It handles any exception that could happen when writing the file
            if file.closed == False:
                file.close()
            IO.output_error_messages("There was a problem with writing to the file.", e)
            IO.output_error_messages("Please check that the file is not open by another program.", e)
            print()
# End of Processing Data Layer
```

Figure 3 – Python Script Processing Data Layer

Presentation data layer and main body are presented in Figure 4 and Figure 5, respectivaly.

```python

# Presentation Data Layer

class IO:  13 usages
    """A collection of presentation layer functions that manage user input and output

        ChangeLog: (Who, When, What)
        Renato Felicio,11/16/2024,Created Class
    """
    @staticmethod  8 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,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')

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

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :return: None
        """
        print(menu)

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

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :return: string with the users choice
        """
        choice="0"
        try:
            choice: str = input("What would you like to do: ")
            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__())  # Not passing e to avoid the technical message

        return choice

    @staticmethod  2 usages
    def output_student_courses(student_data: list):
        """ This function displays the current data to the user

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :return: None
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

    @staticmethod  1 usage
    def input_student_data(student_data: list):
        """ This function gets data from the user and adds it to a list of dictionary rows

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :param student_data: list of dictionary rows containing our current data
            :return: list of dictionary rows filled with a new row of data
        """
        try:
            student_first_name: str = input("Enter the student's first name: ") # Holds student first name input
            if not student_first_name.isalpha():
                raise ValueError("The last name should not contain numbers.") # Custom error
            student_last_name: str = input("Enter the student's last name: ") # Holds student last name input
            if not student_last_name.isalpha():
                raise ValueError("The last name should not contain numbers.") # Custom error
            course_name: str = input("Please enter the name of the course: ") # Holds course name input
            student: dict = {"FirstName": student_first_name,
                             "LastName": student_last_name,
                             "CourseName": course_name}
            student_data.append(student)
        except ValueError as e:
            IO.output_error_messages("", e)
        except Exception as e:
            IO.output_error_messages("Error: There was a problem with your entered data.",e)
        return student_data

# End of Presentation Data Layer

# End of class definition
```

Figure 4 – Python Script Presentation Data Layer

```python

# Start of the main body of the script

# Read data from a file
students:list = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

while (True): # Loops through the menu of options
    # Present the menu of choices
    IO.output_menu(MENU)
    menu_choice=IO.input_menu_choice()

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

    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message
        IO.output_student_courses(students)
        continue

    # Save the data to a file and present to user
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(FILE_NAME,students)
        IO.output_student_courses(students)
        continue

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

print("Program Ended")
```

Figure 5 – Python Script Main Body

Python Script Testing

I executed the Python script using PyCharm (see Figure 7) and also tested it in the Windows Command Prompt (see Figure 8), verifying that the script performed as expected in both environments. Additionally, I confirmed that the Enrollments.json file was updated correctly and contained the expected output (see Figure 9). Following the assignment instructions, I used the initial Enrollments.json file provided in the module_06.zip file. The original content of this file is shown in Figure 6.

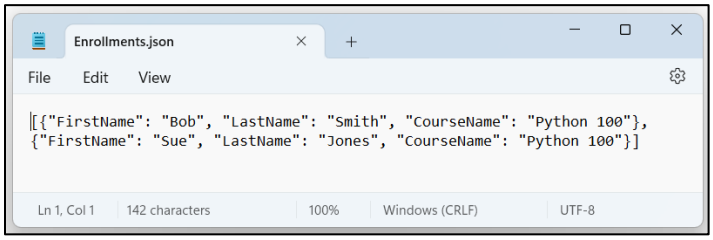

Figure 6 – Initial Enrollment File Python

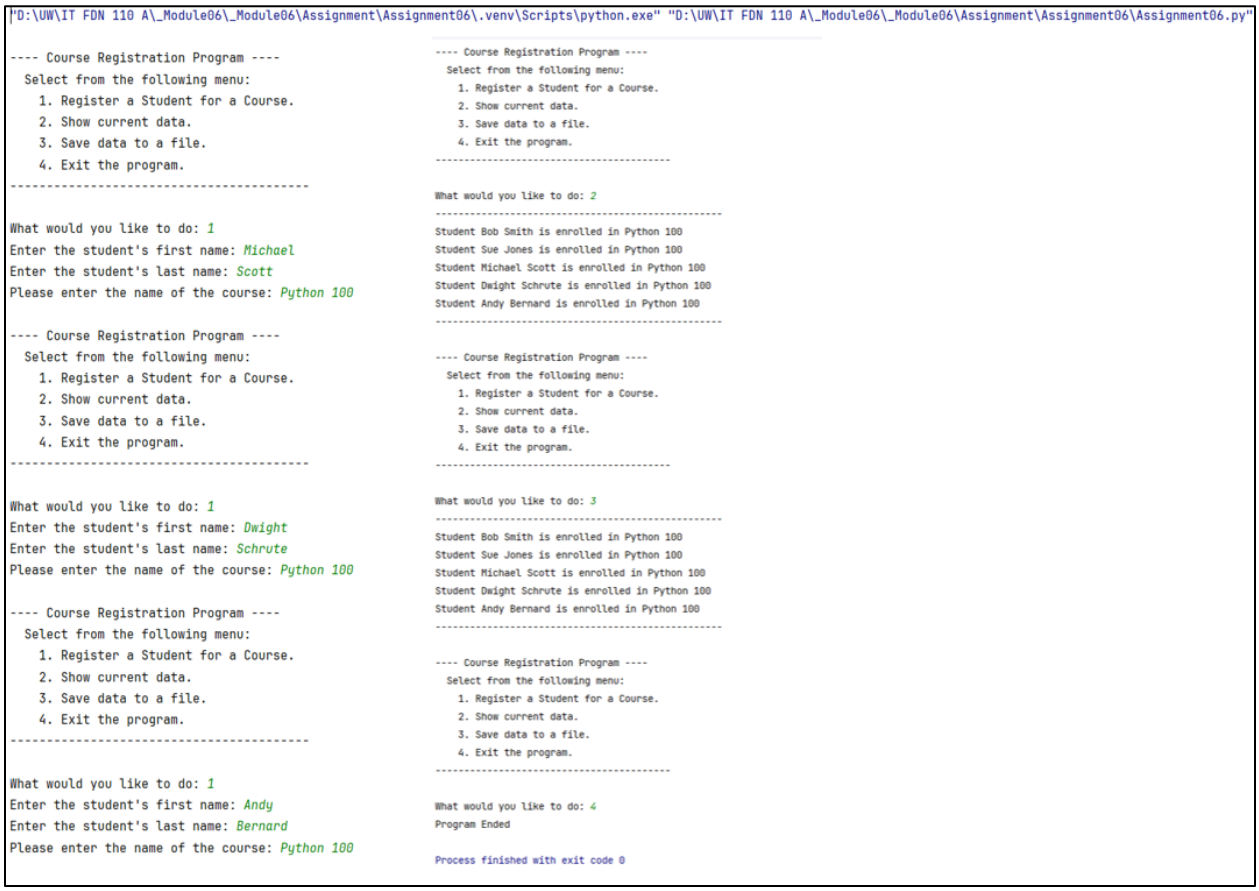

Figure 7 – Python Script PyCharm Run

```
D:\UW\IT FDN 110 A\_Module06\_Module06\Assignment\Assignment06>python 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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Andy
Enter the student's last name: Bernard
Please enter the name of the course: Python 100

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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
Student Andy Bernard is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
Student Andy Bernard is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended
```

Figure 8 – Python Script Windows Command Prompt Run

Figure 9 shows that the Enrollments.json file was updated correctly with the three new student data.

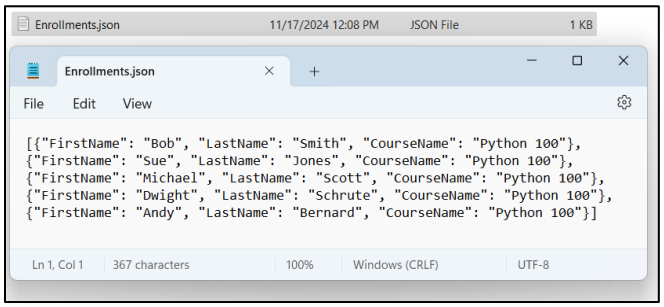

Figure 9 – Output JSON File Content

Python Script Error Handling Test

After testing the script with valid user inputs using a pre-existing Enrollments.json file, I evaluated its error-handling capabilities for scenarios such as a "file not found" exception and invalid first and last name inputs. All tests were conducted using PyCharm.

The Figure 10 below shows the FileNotFound handling and the Enrolments.json being updated correctly after menu choice 3 is selected.

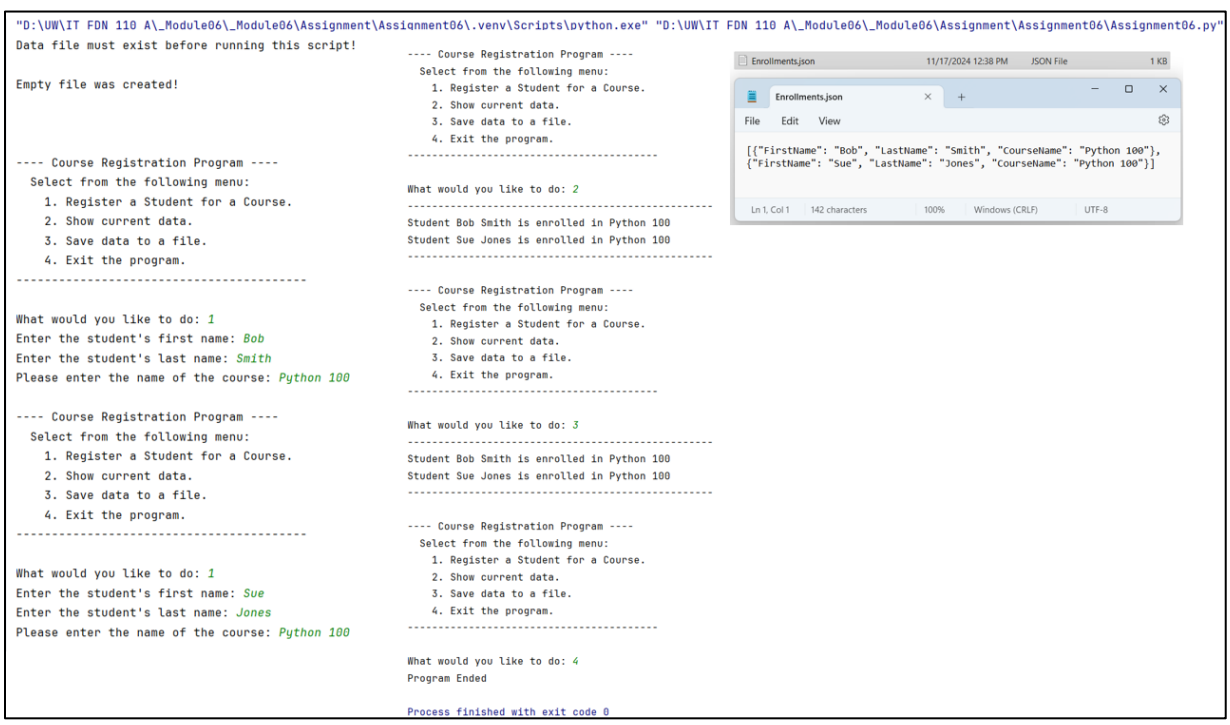

Figure 10 – FileNotFoundError Handling

The second error handling test involved checking the condition where the user enters non-alphabetic characters for the first or last names (see Figure 11). The Enrollments.json file was also correctly updated

```
"D:\UW\IT FDN 110 A\_Module06\_Module06\Assignment\Assignment06\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\_Module06\_Module06\Assignment\Assignment06\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.
----------------------------------------

What would you like to do: 1
Enter the student's first name: 1


-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

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

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: 2


-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 11 – Non-Alphabetic Characters Error Handling

GitHub

Script and documentation for this assignment is available in my GitHub site:

https://github.com/rfnaval/IntroToProg-Python-Mod06.git

Summary

This assignment built upon the previous one, providing an opportunity to learn and practice key Python concepts such as functions, function parameters, return values, and classes. Functions and classes are essential building blocks of the Separation of Concerns (SoC) principle in Python programming. By applying the SoC design principle, I was able to create a more encapsulated and organized script, significantly reducing the complexity of the main body of the code while improving its readability.

References

1. Module 06 - Functions, Randal Root, January 02, 2024.
2. External site: Let's Learn Python - Basics #6 of 8 - Functions, Anchor Rainbow.
3. External site: Python Functions || Python Tutorial || Learn Python Programming, Socratica.