

**Name:** Sohail Nassiri

**Date:** September 4<sup>th</sup>, 2024

**Course:** IT FDN 110 B Su 24 - Foundations of Programming: Python

**GitHub URL:** <https://github.com/ints221/IntroToProg-Python-Mod06>

## Assignment 06 – Creating a Python Script Using Functions and Structured Error Handling

### Introduction

This assignment requires the use of constants, variables, and a variety of functions, in a Python script. In addition, it incorporates the use of string formatting, while and for loops, programming menus, conditional logic, classes, exception handling and java script-oriented notation (JSON) files to work with data. The purpose of the Python script is to allow users to enter and display multiple registrations for students and their associated courses along with the ability to read and save the data to/from a JSON file if desired.

### Creating the Python Script

Using the PyCharm IDE, a Python script is created. The code begins with the script header, as shown in **Figure 1**, in order to give some background information on the purpose of the script. In this case, this script is being written in order to complete the requirements of Assignment 6 demonstrating the use of important concepts in Module 06 being:

- Classes
- Functions
- Parameter and Arguments
- Returns
- Global and local variables
- Separation of Concerns
- JSON

```
# ----- #
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   Sohail Nassiri,09/02/2024,Created Script
# ----- #
```

**Figure 1. Script Header**

First, the Main section is started by importing the JSON module since the script will be utilizing JSON files. Next, the data constants are defined (see **Figure 2**), which are designated in upper case. Also, type hints are used to allow the reader to know the data type associated with each constant or variable. The following constants are assigned:

- The course registration program menu (see below), is assigned to MENU (string).

---- 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

-----

- The name of the .json file, 'enrollments.json', is assigned to FILE\_NAME (string)

```
8 import json
9
10 # Define the Data Constants
11 MENU: str = ''
12 ---- Course Registration Program ----
13 Select from the following menu:
14     1. Register a Student for a Course.
15     2. Show current data.
16     3. Save data to a file.
17     4. Exit the program.
18 -----
19 ''
20 FILE_NAME: str = "enrollments.json" # Set the json file name
```

**Figure 2. Defining the Data Constants**

The global data variables, which are declared outside of the functions and accessible anywhere within the script, are defined and set to empty strings as follows (see **Figure 3**):

- menu\_choice (string) is set to empty string
- students (list) is set to an empty list

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

**Figure 3. Defining the Data Variables**

Then, the following two classes are created in order to organize the code and group the functions that will be used (see **Figures 4a-4e**):

- FileProcessor – a collection of processing layer functions that work with json files
  - The read\_data\_from\_file function is defined with parameters (local variables) file\_name (str) and student\_data (list). This function reads data from a .json file into a list of dictionary rows.
    - The file is opened using the open() function and read using “r” mode.
    - The json.load(file) function parses the data into a list of dictionary rows

- A try-except-finally block is added within the code to serve the following functions:
    - Try – Attempts to run the code in the respective block
    - Except – Runs if exception occurs in the Try block to allow the user to know the cause of the error. In this case, the specific exception is the FileNotFoundError if the .json file is not found. A general Exception error is added as well in case any other exception that is not specifically called out arises.
    - Finally – Runs regardless of whether code successfully executes or if exception is raised to ensure a specific action is carried out. In this case, it is to make sure the .json file is closed before performing any further actions.
  - The write\_data\_to\_file function is defined with parameters file\_name (str) and student\_data (list). This function writes data to a json file from a list of dictionary rows.
    - The current data collected by the user is written to the 'enrollments.json' file. The file is opened using the open() function. In this case, the file is opened using write mode ('w'). The .json file is written with the content of the student\_data variable using the write() function. Finally, the file is saved and closed using the close() function. Similar to when the .json file is read earlier in the script, a try-except-finally block is used in case the file is not found.
- IO – a collection of presentation layer functions that manage user input and output
  - The output\_error\_messages function is defined with parameters message (str) and error (Exception) with the argument of None. This function displays custom error messages to the user.
  - The output\_menu function is defined with parameter menu (str). This function displays the course registration program menu to the user.
  - The input\_menu\_choice function is defined. This function gets a menu choice from the user.
  - The output\_student\_courses function is defined with parameter student\_data (list). This function displays the current data to the user.
  - The input\_student\_data function is defined with parameter student\_data (list). This function gets data from the user and adds it to a list of dictionary rows.

```

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

        ChangeLog: (Who, When, What)
        Sohail Nassiri,09.02.2024, Created Class
        """

    1 usage
    @staticmethod
    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

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

        ChangeLog: (Who, When, What)
        Sohail Nassiri,09.02.2024, 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 = open(file_name, "r") # Reads file
            student_data = json.load(file) # Parses into a list of dictionaries
            file.close()
        except FileNotFoundError as e: # Raises exception if file is not found
            # Sending error messages to a function in IO
            IO.output_error_messages(message: "Text file must exist before running this script!", e)
        except Exception as e: # Raises any other general exception that is not specifically called out
            IO.output_error_messages(message: "There was a non-specific error when reading the file!", e)
        finally:
            if file:
                file.close() # Closes file regardless of whether code successfully executes or not
        return student_data

```

Figure 4a. Creating File Processor Class and Defining Functions

```

1 usage
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to a json file from a list of dictionary rows

    ChangeLog: (Who, When, What)
    Sohail Nassiri,09.02.2024, Created function

    :param file_name: string with the name of the file we are writing to
    :param student_data: list of dictionary rows we have in our data
    :return: None
    """
    try:
        file = open(file_name, "w") # Reads file
        json.dump(student_data, file) # Writes the table list to the JSON file
        file.close()
        IO.output_student_courses(student_data=student_data) # Sending output to a function in IO
    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="There was a non-specific error!", e)
    finally:
        if file:
            file.close()

```

Figure 4b. Creating File Processor Class and Defining Functions

```

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

    ChangeLog: (Who, When, What)
    Sohail Nassiri,09.02.2024, Created Class, Added menu input/output functions, displaying of data, and custom error
    messages
    """

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

        Note: Allows to customize error messages in one place and affect all error handling

        ChangeLog: (Who, When, What)
        Sohail Nassiri,09.02.2024, Created function and toggling technical message off if no exception object is passed

        :return: None
        """
        print(message, end='\n\n')
        if error is not None:
            print("--- Technical Error Message ---")
            print(error, error.__doc__, type(error), sep='\n')

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

        :return: None
        """
        print(menu, end='\n\n') # Adding extra space to make it look cleaner

    1 usage

```

Figure 4c. IO Class and Defining Functions

```

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

    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("What would you like to do: ")
        if choice not in ("1", "2", "3", "4"): # Note these are strings
            raise Exception("You must choose 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__()) # Not passing the exception object to avoid the technical message

    return choice

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

    :return: None
    """
    print("-" * 50)
    for student in student_data: # Iterates through each row of table
        print(
            f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
    print("-" * 50)

```

Figure 4d. IO Class and Defining Functions

```

1 usage
2 @staticmethod
3 def input_student_data(student_data: list):
4     """ This function gets data from the user and adds it to a list of dictionary rows
5
6     :param student_data: list of dictionary rows containing our current data
7     :return: list of dictionary rows filled with a new row of data
8     """
9
10    try:
11        student_first_name = input("Enter the student's first name: ")
12        if not student_first_name.isalpha(): # Requires user to input alphabetical name
13            raise ValueError("The first name should not contain numbers.") # Raises error is non-alphabetical
14        # character is entered
15        student_last_name = input("Enter the student's last name: ")
16        if not student_last_name.isalpha():
17            raise ValueError("The last name should not contain numbers.")
18        course_name = input("Please enter the name of the course: ")
19        student_data.append({"FirstName": student_first_name, "LastName": student_last_name,
20                            "CourseName": course_name}) # Table is appended with data from list of dictionary row
21        print(
22            f"You have registered {student_first_name} {student_last_name} for {course_name}." # Displays
23            # input registration
24    except ValueError as e:
25        IO.output_error_messages(message="Only use names without numbers", e) # Prints the custom message
26    except Exception as e:
27        IO.output_error_messages(message="There was a non-specific error when adding data!", e)
28    return student_data
29
30 # End of class definitions

```

**Figure 4e. IO Class and Defining Functions**

Lastly, the main body of the script is written where the classes and functions are called to carry out the desired outcome. The user is prompted to select from the course registration program menu. The options allow the user to enter multiple registrations for students and their courses, display the data collected, read and save data to/from a json file, and/or exit the program. In addition, exception handling is incorporated in order to guide the user through possible errors that may be encountered. This is carried out as follows (see **Figure 4f**):

- When the program begins, the data in “enrollments.json” is automatically read into a two-dimensional table (a list of dictionary rows) by calling the FileProcessor class, read\_data\_from\_file function and passing the FILE\_NAME and students arguments to the respective parameters.
- A while loop is set to true, which results in an infinite loop until exited by a break. Inside the while loop, the course registration program menu is printed and the user is asked what selection they would like to make by calling the IO class, output\_menu and input\_menu\_choice functions.
- Conditional logic is used, specifically if, else-if (elif) and else statements, to carry out the proper action based on the selection made:
  - If the user selects “1”, they are prompted to enter the student’s first and last name along with the course name. The data collected is displayed and added to the two-dimensional table. In addition, if a non-alphabetical name is entered for the first and last name, a ValueError is raised to notify the user and prompt them to try again. This is achieved by calling the IO class and input\_student\_data function and passing the students argument to the student\_data parameter.
  - If the user selects “2”, the current data collected is displayed. This is done by calling the IO class and output\_student\_courses function and passing the students argument to the student\_data parameter.

- If the user selects “3”, the current data collected by the user is written to the ‘enrollments.json’ file. This is done by calling the FileProcessor class and write\_data\_to\_file function and passing the FILE\_NAME and students arguments to the respective parameters.
- If the user selects “4”, the while loop is exited by introducing a break. In addition, this ends the program and the user is notified with a print statement.
- If the user enters in an invalid selection, the user is notified with a print statement and prompted to select from the menu again. This is handled by the input\_menu\_choice function in the IO class.
- As a result of the infinite while loop, the user will continuously be prompted to make selections from the menu until the choice of exiting the program is made.

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

# When the program starts, reads the file data into a table and extracts the data from the file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Repeat the follow tasks
while True:
    IO.output_menu(menu=MENU) # Present menu choices
    menu_choice: str = IO.input_menu_choice()

    if menu_choice == "1": # Input data and display data entered
        students = IO.input_student_data(student_data=students)
        continue

    elif menu_choice == "2": # Get new data and display the change
        IO.output_student_courses(student_data=students)
        continue

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

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

print("Program Ended")
```

**Figure 4f. Main Body of Script**

## Running the Python Script

To start, when running the script, the user is prompted to select from the course registration program menu.

**Figures 5a – 5c** and **Figures 6a – 6b** show what occurs with each selection as described in the “Creating the Python Script” section when running in the PyCharm IDE and command prompt, respectively. This example includes if the user enters a non-alphabetical name and multiple student registrations. When properly running the script, to prevent an error from occurring when reading the enrollments .json file, the following starting data was entered: Vic,Vu,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: J4
Only use names without numbers

-- Technical Error Message --
The first 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: John
Enter the student's last name: D4
Only use names without numbers

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

Figure 5a. Running the Python Script from the PyCharm IDE

```
---- 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: John
Enter the student's last name: Doe
Please enter the name of the course: Math 100
You have registered John Doe for Math 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 Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 100
-----
```

Figure 5b. Running the Python Script from PyCharm IDE

```
---- 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 Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 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 5c. Running the Python Script from PyCharm IDE

```

---- 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: J4
Only use names without numbers

-- Technical Error Message --
The first 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: John
Enter the student's last name: D4
Only use names without numbers

-- 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: John
Enter the student's last name: Doe
Please enter the name of the course: Math 100
You have registered John Doe for Math 100.

```

Figure 6a. Running the Python Script from the Command Prompt

```

---- 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 Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 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 Vic Vu is enrolled in Python 100
Student John Doe is enrolled in Math 100
-----

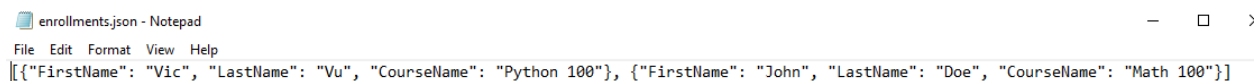
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----

What would you like to do: 4
Program Ended

```

**Figure 6b. Running the Python Script from the Command Prompt**

As shown below in **Figure 7**, when selecting Option #3, the data is written and saved to the 'enrollments.json' file.



```

enrollments.json - Notepad
File Edit Format View Help
[{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "John", "LastName": "Doe", "CourseName": "Math 100"}]

```

**Figure 7. Data Stored in .JSON File After Running the Python Script**

## Summary

The goal of this assignment was to demonstrate the creation of a Python script that prompts the user to select from a course registration program menu in order to allow for entering and displaying multiple registrations for students and their associated courses along with the ability to read data from/save data to a JSON file using the separation of concerns method. Constants, variables, classes, various functions, string formatting, while and for loops, programming menus, conditional logic, classes, exception handling and JSON files are utilized in this script in order to achieve the desired outcome.