Allan Duldulao
May 24, 2025
Foundations of Programming: Python
Assignment 06
GitHub: https://github.com/AllanDuldulao/IntroToProg-Python-Mod06

Functions

Introduction

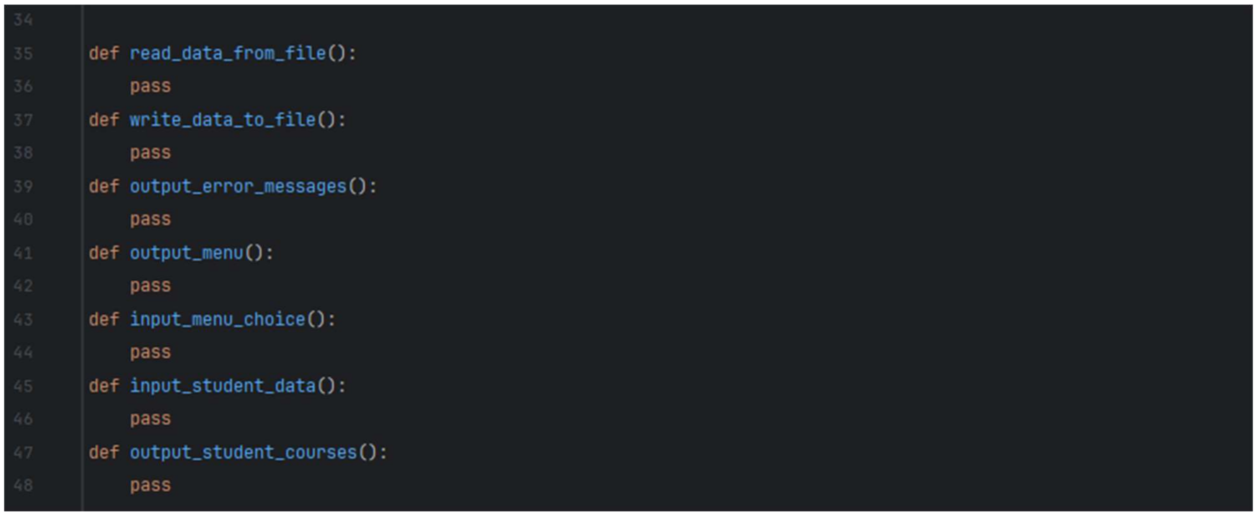
After we've learned in the Module 5 about Try-Except, List, dictionaries, JSON and GitHub. We are now heading to study Module 06 about functions. This script that we are going to be working on this module is almost the same as the last one. We are just going to add function, creating global variables, parameters, static classes, DocStrings and Separations of Concerns Pattern.

My Script for the Assignment

Started the assignment with the given starter file. And work slowly editing the script line-by-line, until achieved the objective of the assignment. First, I took my module 05 assignment and compare it to the starter file and make it look like the starter file. After that I begin the script by organizing my code using Functions. Second, I define function on the different part of my script. I defined 7 different functions for this assignment.

Functions

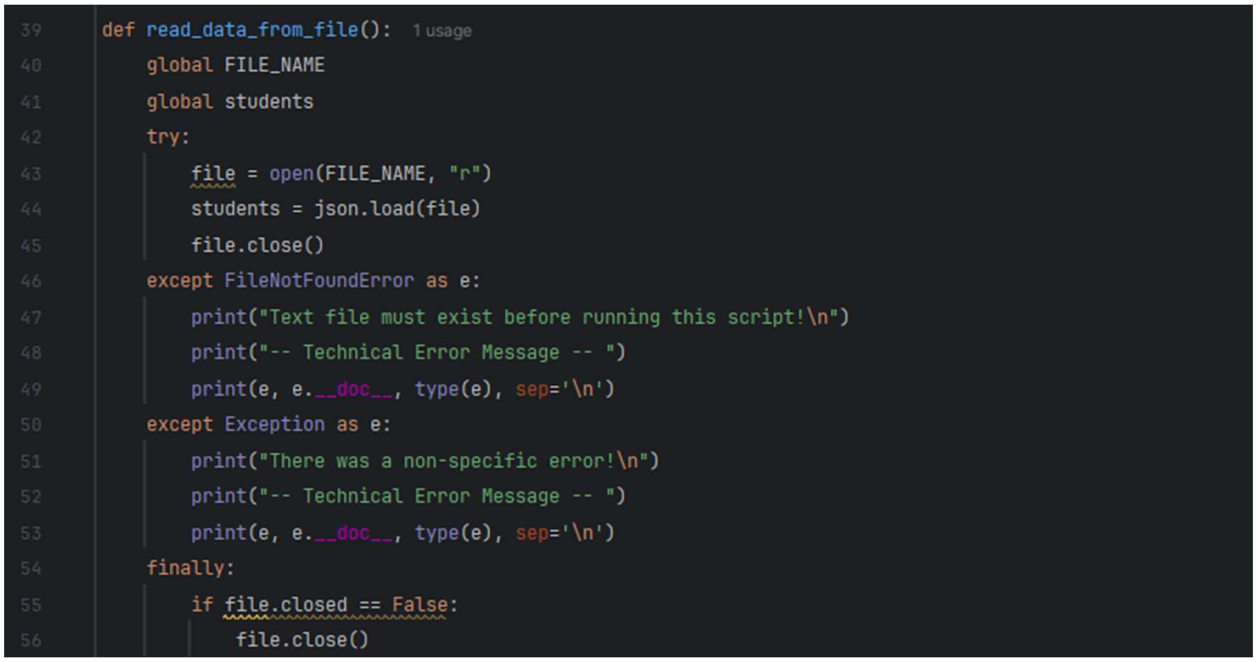
In programming, function is a reusable block of code that performs a specific task or a set of tasks. Functions are fundamental concepts in most programming languages and serve several important purposes. Functions groups a set programming statements and later reference them by a given name, they must be defined in a script before they can be called. Since these functions are yet to be implemented, added "pass" to get rid of the errors. The "pass" keyword in python is a place holder statement or a no operation. It is used when python requires some statement in your code, but you don want to do anything. For example, if we wanted to start organizing out code using functions, we could define the functions first, then add code to the functions later. Figure 1 are functions we defined in this assignment.



```python

def read_data_from_file():
    pass
def write_data_to_file():
    pass
def output_error_messages():
    pass
def output_menu():
    pass
def input_menu_choice():
    pass
def input_student_data():
    pass
def output_student_courses():
    pass
```

Fig: 1. Defined functions

After defining the functions, we will start slowly adding code to the functions that meant for it. Fig. 2 is the example of the defined read data from file.



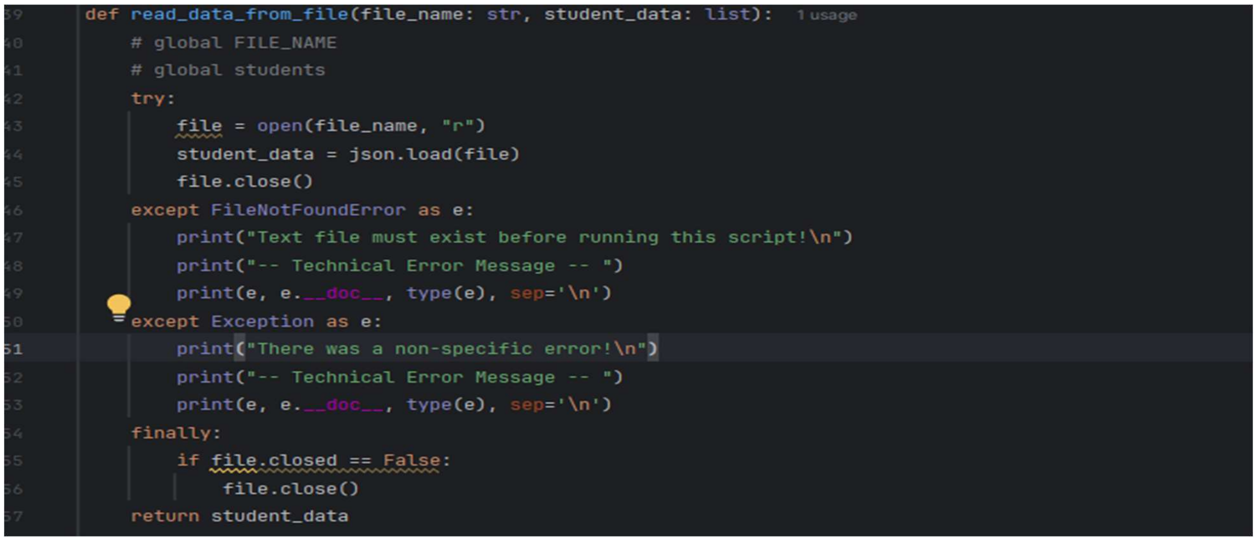
```python
def read_data_from_file():  1 usage
    global FILE_NAME
    global students
    try:
        file = open(FILE_NAME, "r")
        students = json.load(file)
        file.close()
    except FileNotFoundError as e:
        print("Text file must exist before running this script!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    except Exception as e:
        print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    finally:
        if file.closed == False:
            file.close()
```

Fig. 2: Added code to the defined function and declared variables

On this example we added the script to start the program by reading from a file name and display error if the file is missing. But because we are working with functions now, we have to declare global variables. In this defined function ”read_data_from_file”, we only needed to have FILE_NAME and students as global variable. A global variable is defined outside of any functions or class, making it accessible throughout the entire file or even across. Global variables can be accessed inside functions without any special declaration. After this step we now move forward to parameters.

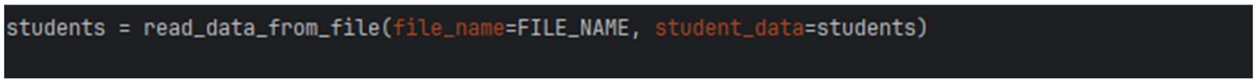
Parameters

Instead of accessing data using global variables in functions, data can be pass in to the functions by creating parameters. In the original version of the function, we used global variables to access and modify data. Global variable is defined outside of functions and can be accessed and modified from anywhere in the script. However, this approach relies on the existence of these global variable and their values being set correctly before calling the function. Any changes made to these variables within the functions will affect their values globally known as "side effects". By declaring this global variable as parameters in the function, we can provide data to the data when you call it. This is much better way to work with data in function. Few reason that we use parameters are, Encapsulation, Flexibility and Predictable behavior. Using parameters allows you to make functions more flexible and reusable because you can customize their behavior by providing different values when you call them. Fig. 3, We added parameters and commented out global variables. To run the new version of the function we need to pass it either values or references to variables and constants either based on their position in the function's definition or by name. Using the parameter name is preferred since it make the code easier to read and hard to pass on the wrong value or reference. To make this work, we need also to locate the function that's getting called and put an argument inside the parentheses. Fig. 4, Arguments added on to the functions.



```
def read_data_from_file(file_name: str, student_data: list):  1 usage
    # global FILE_NAME
    # global students
    try:
        file = open(file_name, "r")
        student_data = json.load(file)
        file.close()
    except FileNotFoundError as e:
        print("Text file must exist before running this script!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    except Exception as e:
        print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
    finally:
        if file.closed == False:
            file.close()
    return student_data
```

Fig. 3: Added parameters on the defined function

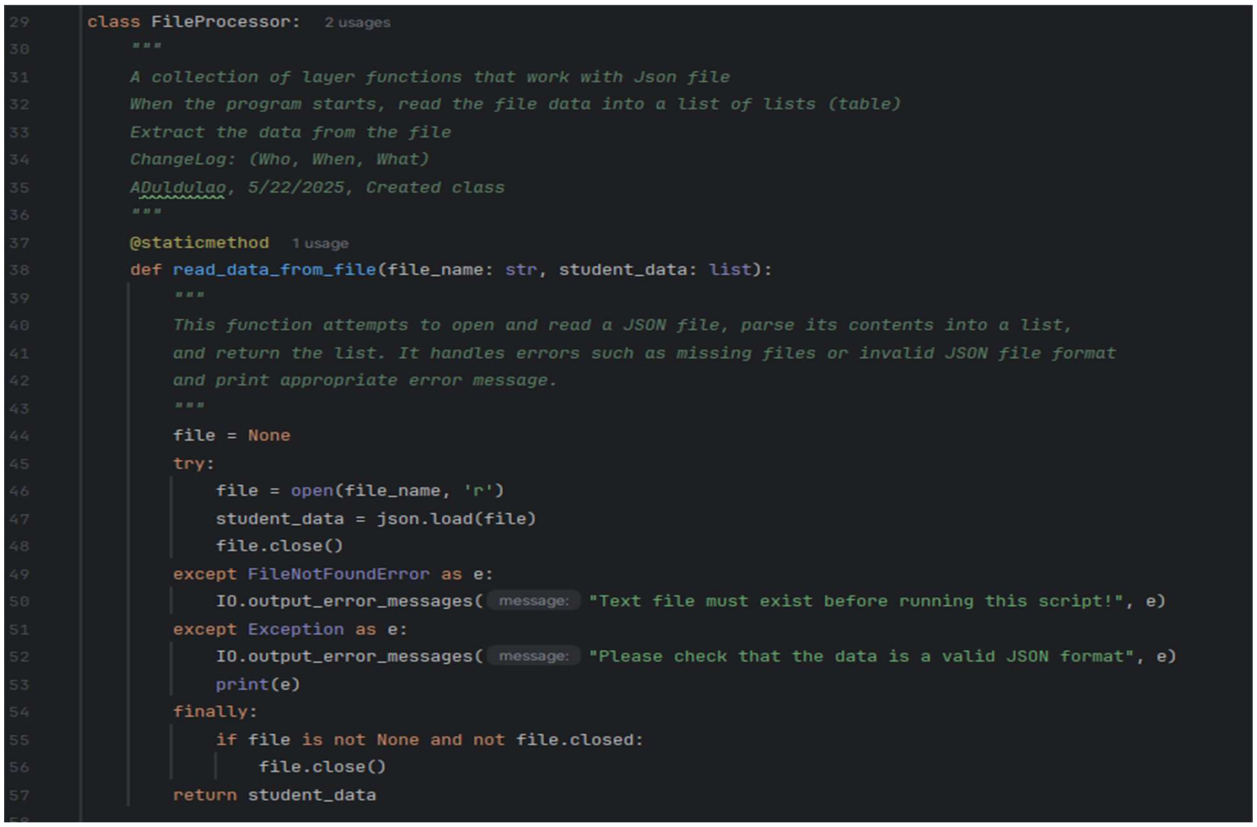


```
students = read_data_from_file(file_name=FILE_NAME, student_data=students)
```

Fig. 4: Added arguments

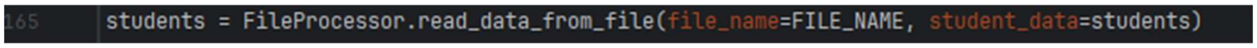
Classes and DocStrings

On the next step of our code to make it work is classes. Classes is a way to group related pieces of information be it functions or variables into a single object. Classes are a core feature of object-oriented programming, allowing to define custom data types with attributes and methods. Advanced features like static methods and class methods provide additional flexibility. When a function is defined with static, these methods don't take self or cls and act like regular functions and scoped to the class. Although the result data is different, depending on the object, the function code never changes and is "static". Next, we add docstrings on out code. Docstrings serves as a built-in way to provide clear, human-readable documentation about the purpose, usage, parameters, and return values of code components. Fig 5., we added class and docstring on the assignment script. Here we can see we added a new class FileProcessor. And now our script is almost done and working.



```python
class FileProcessor:
    """
    A collection of layer functions that work with Json file
    When the program starts, read the file data into a list of lists (table)
    Extract the data from the file
    ChangeLog: (Who, When, What)
    ADuldulao, 5/22/2025, Created class
    """
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """
        This function attempts to open and read a JSON file, parse its contents into a list,
        and return the list. It handles errors such as missing files or invalid JSON file format
        and print appropriate error message.
        """
        file = None
        try:
            file = open(file_name, 'r')
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("Text file must exist before running this script!", e)
        except Exception as e:
            IO.output_error_messages("Please check that the data is a valid JSON format", e)
            print(e)
        finally:
            if file is not None and not file.closed:
                file.close()
        return student_data
```

Fig. 5: Added Classes and Docstring to the assignment script



```python
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
```

Fig. 6: Added FileProcessor on the function

And on Fig.6, we added the created class FileProcessor to the function read_data_from_file. Without this, an unresolved error will come out.

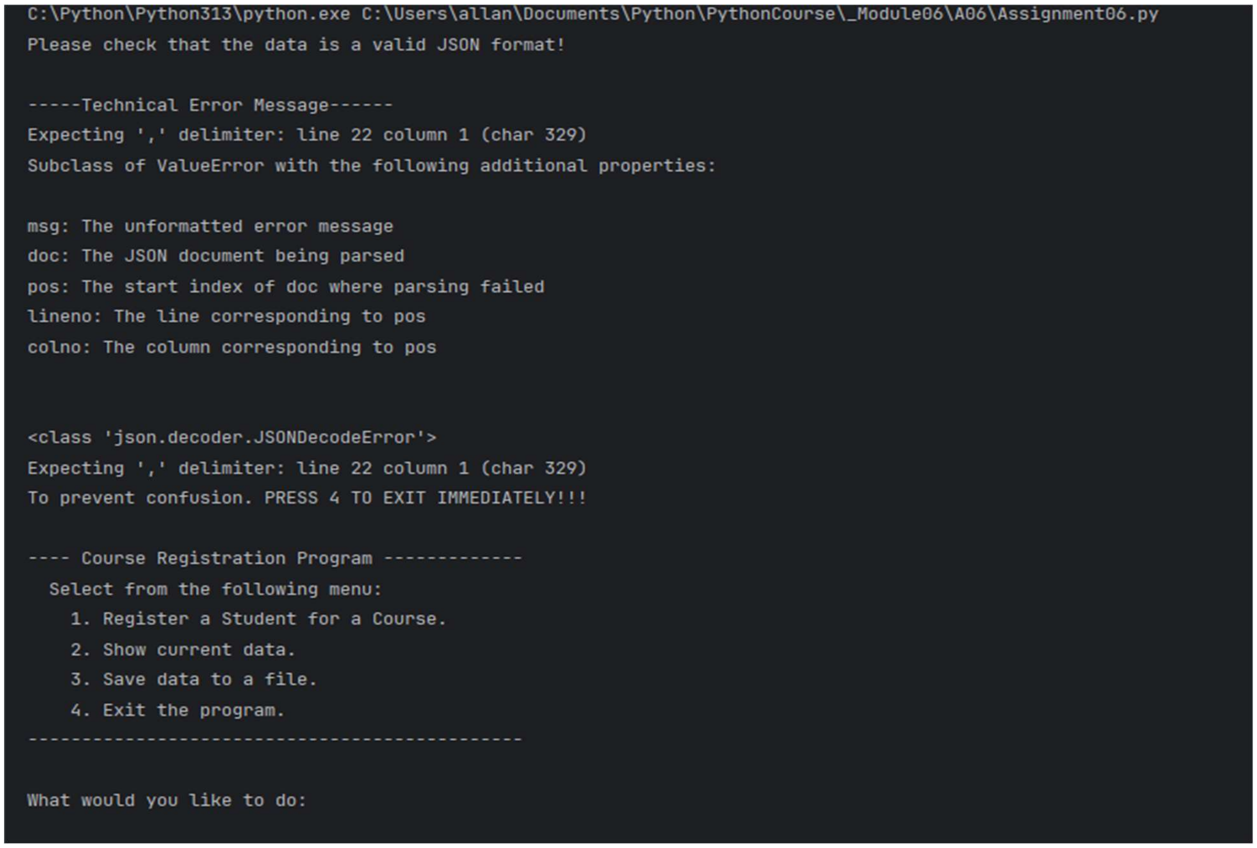
Separation of Concerns

The Separation of Concerns (SoC) is a fundamental software design principle that aims to enhance the maintainability, scalability, and readability of code by breaking it down into distinct, self-contained components, each responsible for a specific aspect of the application's functionality. This pattern encourages modularity, reduces code complexity, and makes it easier to manage and extend software systems. In Python, SoC can be implemented at various levels. Functions, Classes, Modules, and Application Architecture. Benefits of Soc can be varied from maintainability, testability, scalability, collaboration and reusability. By including Docstring it enhances SoC clearly documenting each component's purpose, inputs, and outputs, making it easier to understand and maintain separated concerns.

Results on running the script

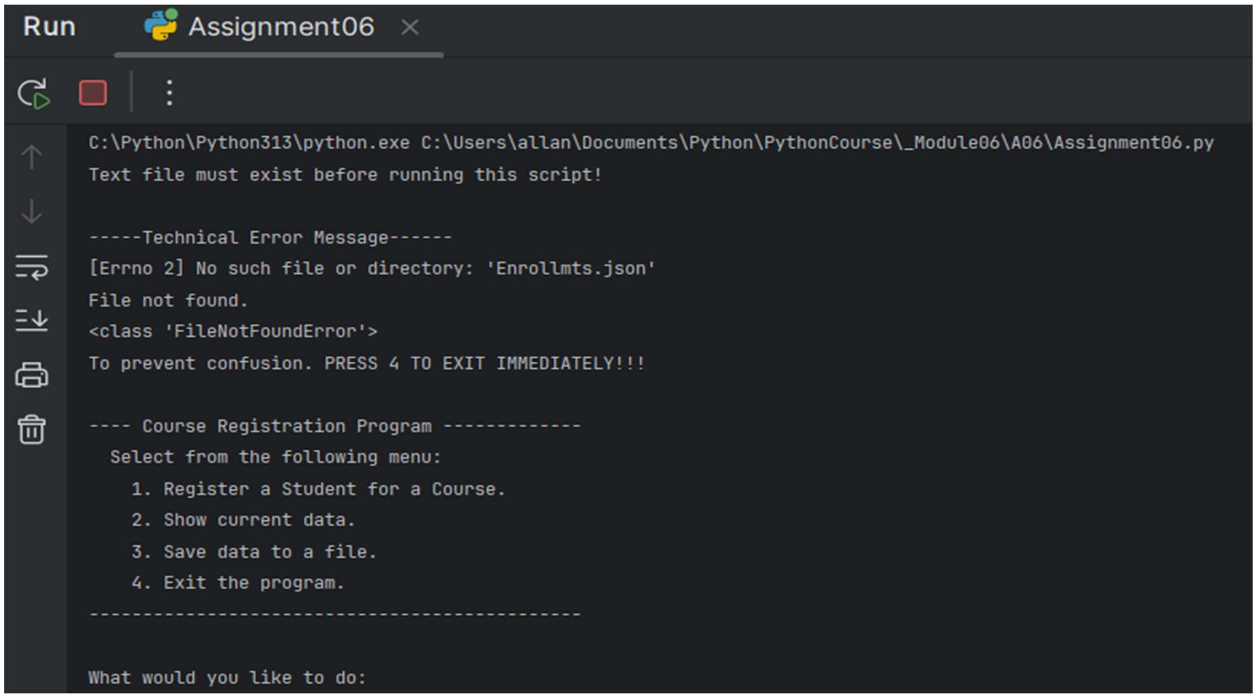
The following figures is the result when testing the script to achieve the objective of the module 06 assignment.

Error Handling Results



```
C:\Python\Python313\python.exe C:\Users\allan\Documents\Python\PythonCourse\_Module06\A06\Assignment06.py
Please check that the data is a valid JSON format!

-----Technical Error Message------
Expecting ',' delimiter: line 22 column 1 (char 329)
Subclass of ValueError with the following additional properties:

msg: The unformatted error message
doc: The JSON document being parsed
pos: The start index of doc where parsing failed
lineno: The line corresponding to pos
colno: The column corresponding to pos


<class 'json.decoder.JSONDecodeError'>
Expecting ',' delimiter: line 22 column 1 (char 329)
To prevent confusion. PRESS 4 TO EXIT IMMEDIATELY!!!

---- 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:
```

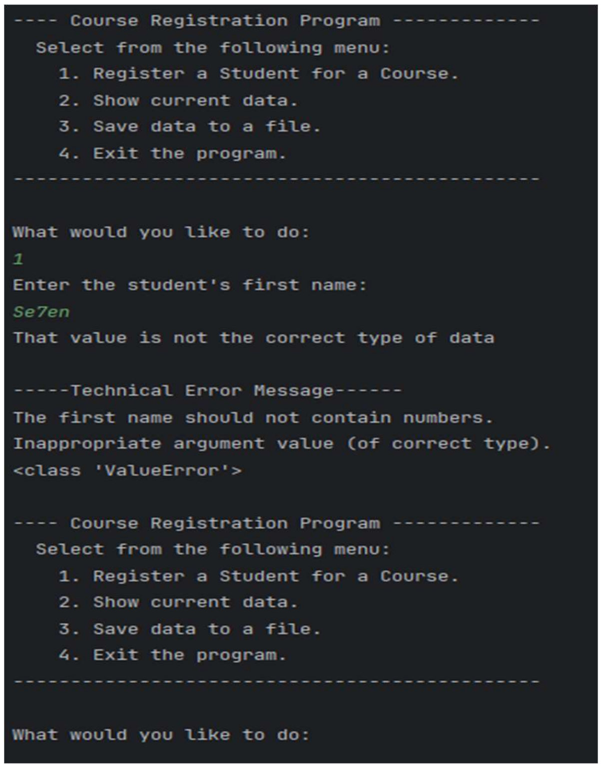
Fig. 7: Data file is not in valid JSON format



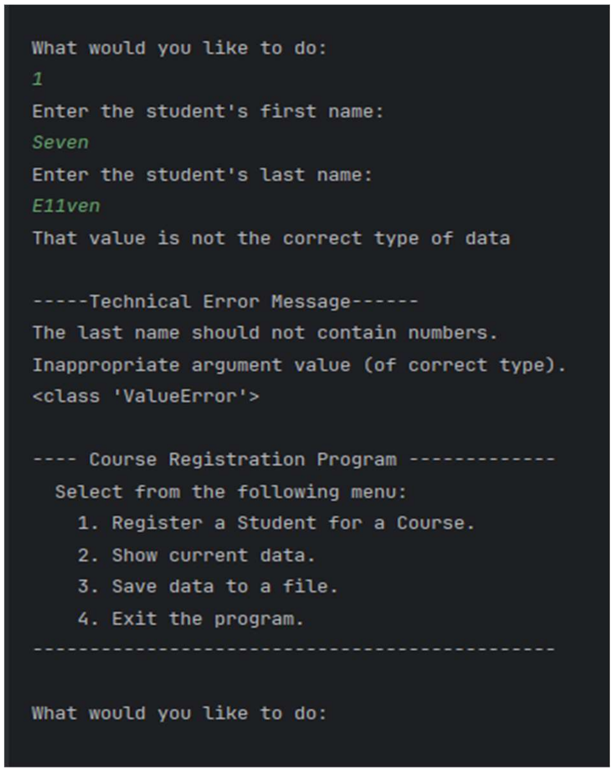
Run   Assignment06

```
C:\Python\Python313\python.exe C:\Users\allan\Documents\Python\PythonCourse\_Module06\A06\Assignment06.py
Text file must exist before running this script!

-----Technical Error Message------
[Errno 2] No such file or directory: 'Enrollmts.json'
File not found.
<class 'FileNotFoundError'>
To prevent confusion. PRESS 4 TO EXIT IMMEDIATELY!!!

---- 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:
```

Fig. 8: There is no file exist



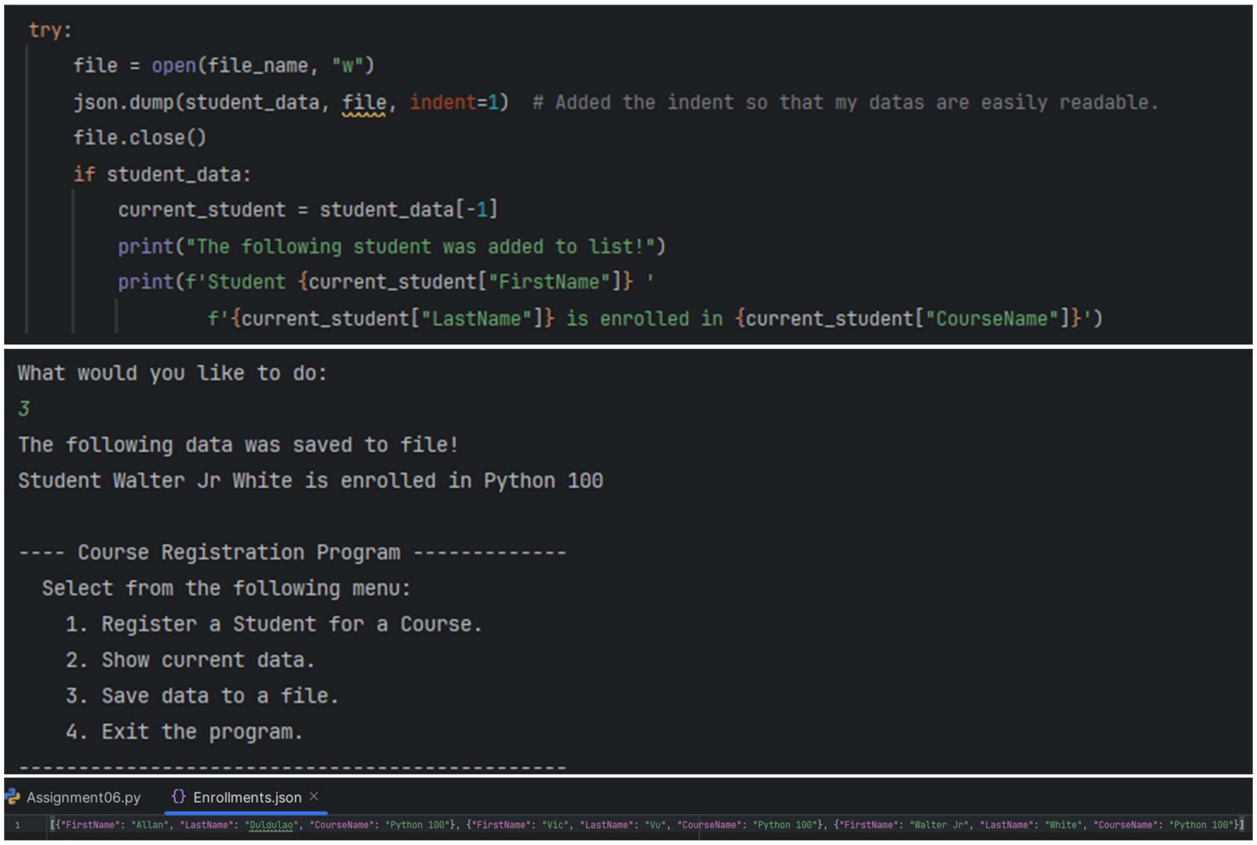
```
---- Course Registration Program -------------
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------------

What would you like to do:
1
Enter the student's first name:
Se7en
That value is not the correct type of data

-----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:
```



```
What would you like to do:
1
Enter the student's first name:
Seven
Enter the student's last name:
E11ven
That value is not the correct type of data

-----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:
```

Fig. 9: The error handling when entering non alphabet characters

I noticed that in the requirements on menu choice 3 is to dump all the data was made. But I decided just to show just the last input to be shown, since menu choice 2 already shown all of the data. Figure 10, is the code and result of it, even though there's multiple entry already on the JSON file.



```python
try:
    file = open(file_name, "w")
    json.dump(student_data, file, indent=1)  # Added the indent so that my datas are easily readable.
    file.close()
    if student_data:
        current_student = student_data[-1]
        print("The following student was added to list!")
        print(f'Student {current_student["FirstName"]} '
              f'{current_student["LastName"]} is enrolled in {current_student["CourseName"]}')
```

```
What would you like to do:
3
The following data was saved to file!
Student Walter Jr White 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.
-----------------------------------------------
```

```
[{"FirstName": "Allan", "LastName": "Duldulao", "CourseName": "Python 100"}, {"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "Walter Jr", "LastName": "White", "CourseName": "Python 100"}]
```

Fig. 10: Only last student on the list shown on menu choice 3

Summary

On this module, we have learned about functions, parameters, arguments, return values, global and class variables, organizing code and Separations of Concern pattern. As we've discussed on the importance of error handling on the script. Whereas in this module though we still have to work on exception handling, we had learned the importance of organizing our codes. Organizing the code to be easily read by other developer or by any user. An organized code using the proper functions, parameters and classes might be the one of the most important skill a developer should excel at. With an organized code, even non computer programmer person can understand the flow or process of the script.