Deborah Briggs

May 22, 2025

Foundations of Programming: Python

Assignment 06

https://github.com/jdbriggs3/IntroToProg-Python-Mod06

Functions

Introduction

Lesson 6 introduces three common methods to organize code: functions, classes and separation of concerns (SoC). In our assignment this week we reorganize the program from Assignment 05 implementing all three of these methods. To the user the program appears to run the same, all the changes are found within the structure of the code.

Functions

A function is a block of code that performs a specific task. When the specific task is needed the function is called. Utilizing parameters and arguments this block of code is reusable. Parameters, defined in the function, are variables and act as placeholders for the values we want to pass into the function. Arguments are the data passed to the function when it is called. The arguments map to the parameters. I got thoroughly discombobulated with this concept, so sought help from YouTube. Two videos that helped me to understand were: Python - Function Parameters and Arguments with Code Examples - Learn Python Programming APPFICIAL 2021 (external link) and Using Functions in Python | Learning Python for Beginners | Code with Kylie #7 2020 (external link)

Classes and Separation of Concerns (SoC)

Classes and Separation of Concerns work together forming the structure of the program. Grouping related functions within specific classes eases both reading, and maintenance of the code. In our assignment we utilized two classes to group functions. The first class, FileProcessor, contained the functions that worked with the JSON files, reading from and writing to the file. (Figure 1) The second class, IO (Input/Output) contained the functions that managed user input and output. (Figure 1.1)

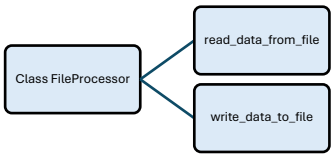

Figure 1: Class FileProcessor

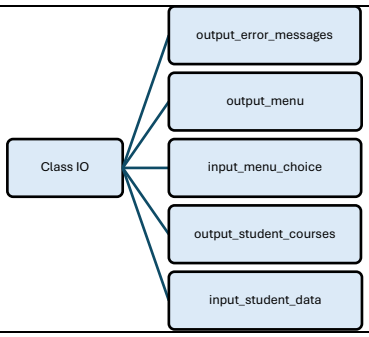

Figure 1.1: Class IO

Programming

In the assignment we are given a starter file, Assignment06-Starter.py, and in the assignment document we learn the process to complete the program is similar to the process outlined in Mod06-Lab03. Beginning with the starter code and carefully walking through the steps in the lab provided in our Assignment notes page 35-41 I was able to successfully complete the assignment. The simplified structure is seen foremost in the main body of the code – the while loop, where the functions are called. (Figure: 2)

Figure 2: Main Body of Code calling functions

Assignment06.py | Enrollments.json | Assignment06Test.py | Creates-Populates-JSON-file.py

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

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

# Present and Process the data
while (True):
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()

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

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

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

print("Program Ended")
```

```python
@staticmethod  1 usage
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads data from a file

    ChangeLog: (Who, When, What)
    DBriggs,5/24/25, Created function

    :return: None
    """
    try:
        file = open(file_name, "r")
        student_data = json.load(file)
        file.close()
    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.closed == False:
            file.close()
    return student_data
```

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

    ChangeLog: (Who, When, What)
    DBriggs,5,24,25, Created function

    :return: None
    """
    print()
    print(menu)
    print()  # Adding extra space to make it look nicer.
```

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

    ChangeLog: (Who, When, What)
    DBriggs,5,24,25, Created function

    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        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 the exception
    return choice
```

```python
@staticmethod  1 usage
def input_student_data(student_data: list):
    """ This function gets the first name, last name, and course name from the user

    ChangeLog: (Who, When, What)
    DBriggs, 5,24,25,Created function

    :return: None
    """
    try:
        # Input the data
        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: "That value is not the correct type of data!", e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!", e)
    return student_data
```

```python
@staticmethod  1 usage
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to the file

    ChangeLog: (Who, When, What)
    DBriggs,5/24/25, Created function

    :return: None
    """
    try:
        file = open(file_name, "w")
        json.dump(student_data, file, indent=2)
        file.close()
    except TypeError as e:
        IO.output_error_messages( message: "Please check that the data is a valid JSON format", e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!", e)
    finally:
        if file.closed == False:
            file.close()
```

Testing – Exception Handling

I performed the same testing as Assignment 05 to verify the program processed accurately including proper error handling.

Read File

The images below demonstrate testing with a non-existent file (Figure 3) and testing with an empty file. (Figure 3)

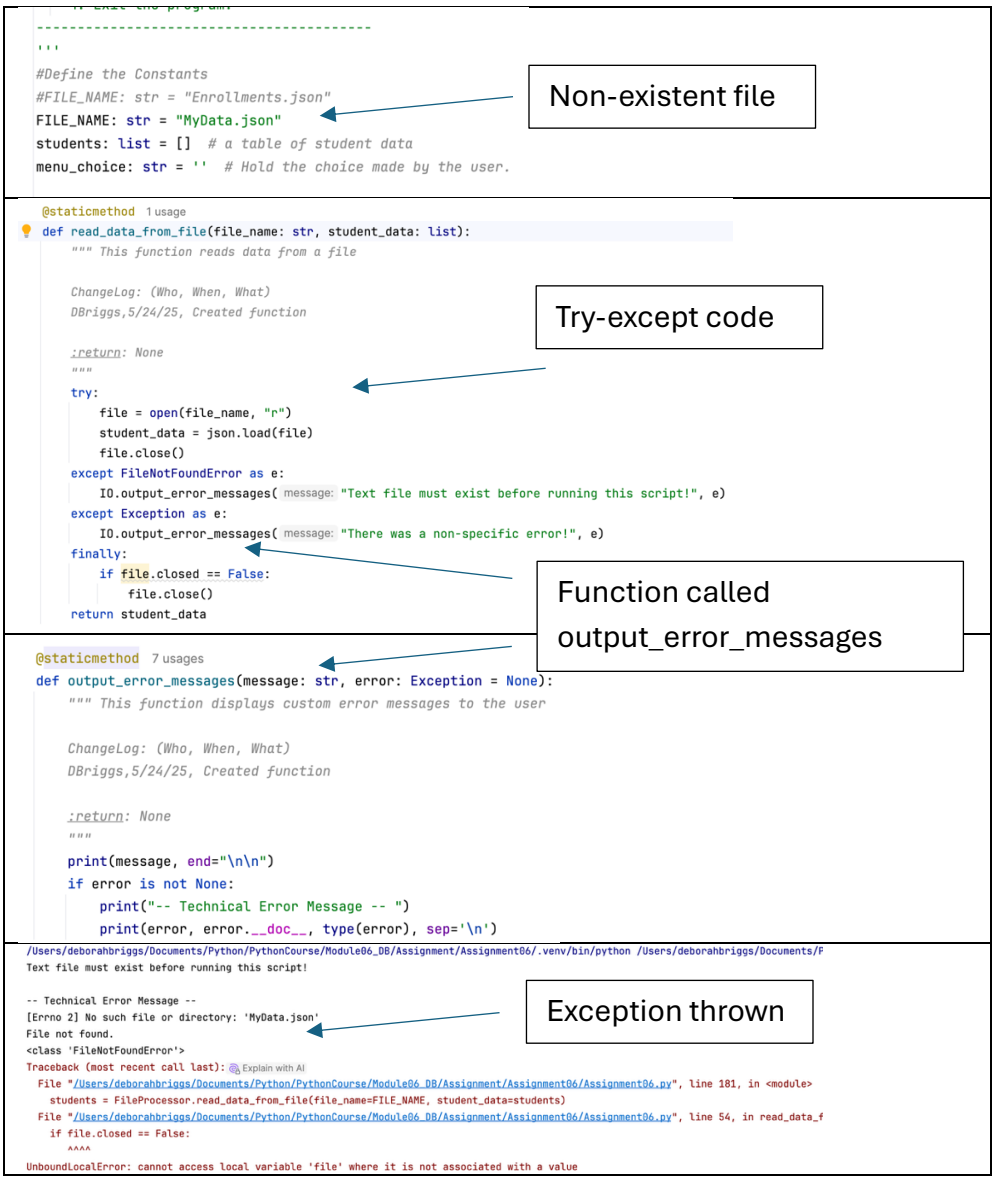

Figure 3: Exception handling - Non-existent file

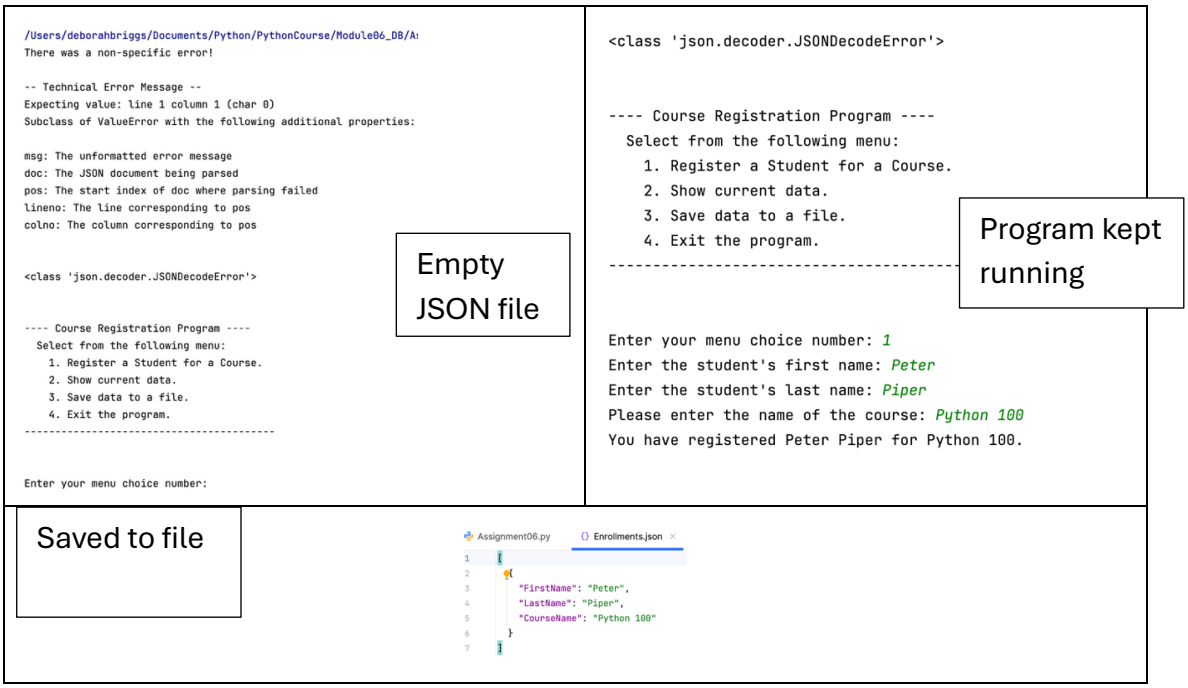

Figure: 3.1 Exception handling - JSON file empty

User Input

To test if the try-except code worked to catch input errors I ran the code using numbers for both first and last name. The exception was properly executed in both cases. The outline below demonstrates the user input flow of code from the main body to the exception code. (Figure 3.2)

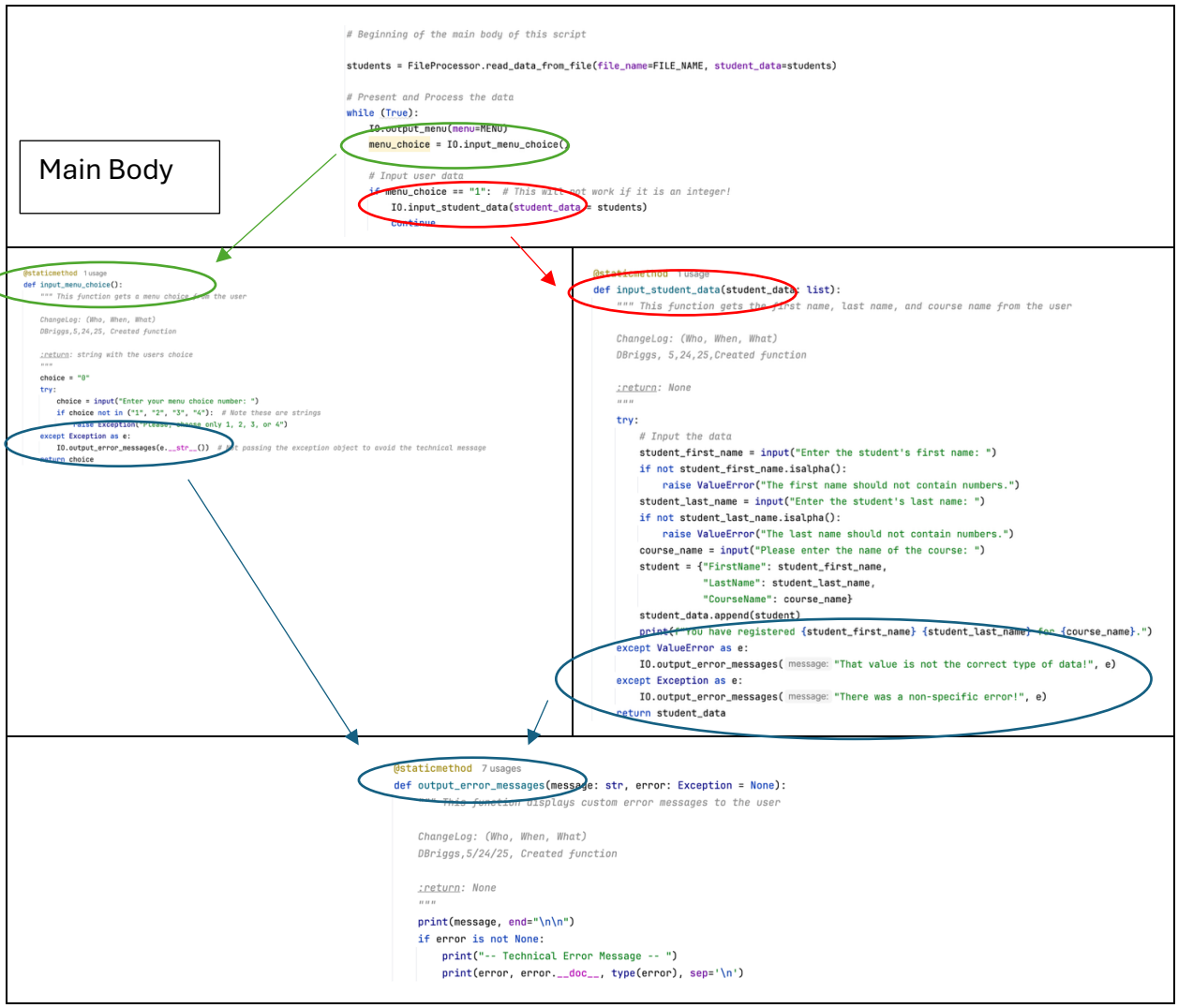

Figure 3.2: Testing User input exception handling

Interface

The images below are screen shots of the interface with the users in both PyCharm and the terminal. (Figure 4)

PyCharm

```
---- 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.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: Paul
Enter the student's last name: Petersburg
Please enter the name of the course: Python 100
You have registered Paul Petersburg for 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.
----------------------------------------


Enter your menu choice number: 2
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Paul Petersburg 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.
----------------------------------------


Enter your menu choice number: 3


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


Enter your menu choice number: 4
Program Ended

Process finished with exit code 0
```

Terminal

```
Assignment06 — -zsh — 73×
deborahbriggs@Deborahs-MacBook-Pro Assignment06 %


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


Enter your menu choice number: 1
Enter the student's first name: Sarah
Enter the student's last name: Smith
Please enter the name of the course: Python 100
You have registered Sarah Smith for 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.
----------------------------------------


Enter your menu choice number: 2
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Paul Petersburg is enrolled in Python 100
Student Sarah Smith 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.
----------------------------------------


Enter your menu choice number: 3


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


Enter your menu choice number: 4
Program Ended
```

Figure 4: User Interface

Output

The image below is a screenshot of the Enrollments.json file. (Figure 4.1)

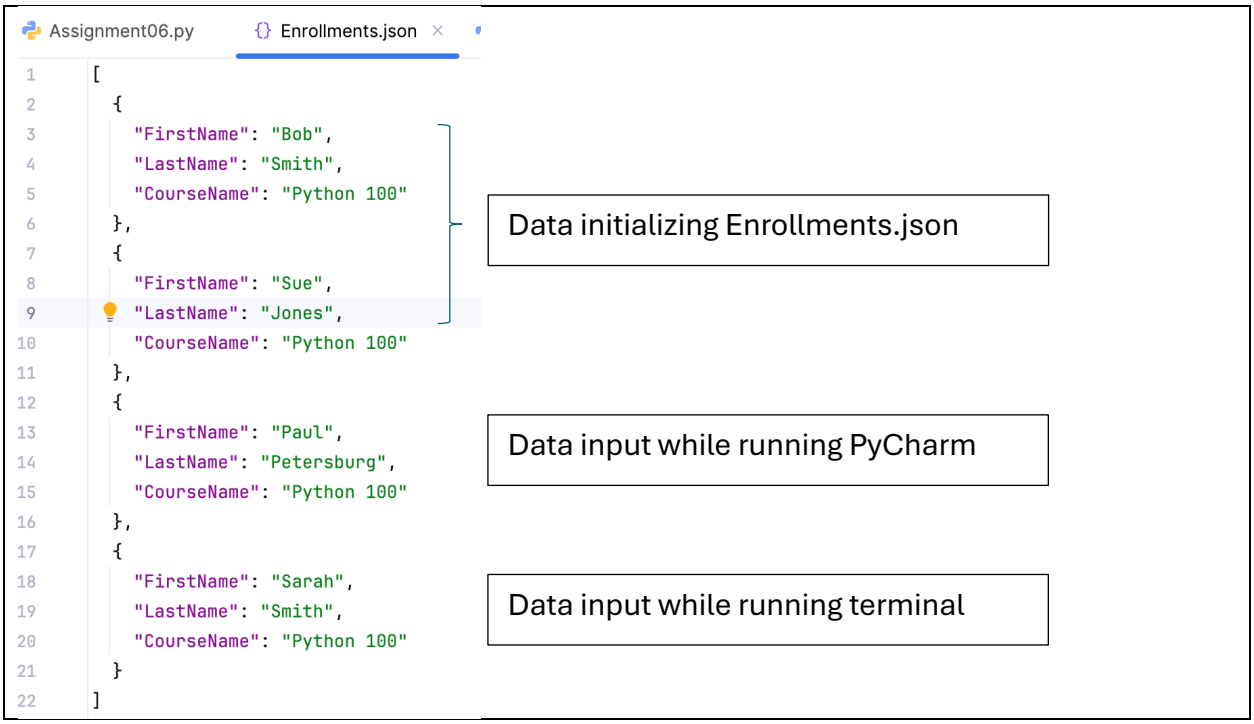

Figure 4.1: Enrollments.json

Summary

Lesson 6 introduced three common methods used to organize code: functions, classes and separation of concerns (SoC). In our assignment this week we reorganized the program from Assignment 05 implementing all three of these methods. On the surface the program for the user appears the same, all the changes are within the structure of the code. This clear structure enhances readability and maintainability of the code.