Deric Fetters

6/8/2024

IT FDN 110 A

Assignment 06

Functions

Introduction

Module 5 most notably introduces working with global vs local variables, classes, functions, parameters and return values. These topics improve the functionality, usability and shareability of our code.

Classes

Classes were utilized in this assignment to organize and modularize functions. Separation of concerns organizes functions into three categories: data layer, processing layer and presentation layer. Data layer handles data-related operations such as data storage and retrieval. Class ProcessFileData represents the data layer and contains the functions for reading and writing data to files. Class IO represents the presentation layer and contains functions for presenting information and collecting user input. In addition, the @staticmethod decorator was used to allow the calling of functions without creating a object. See Figure 1.

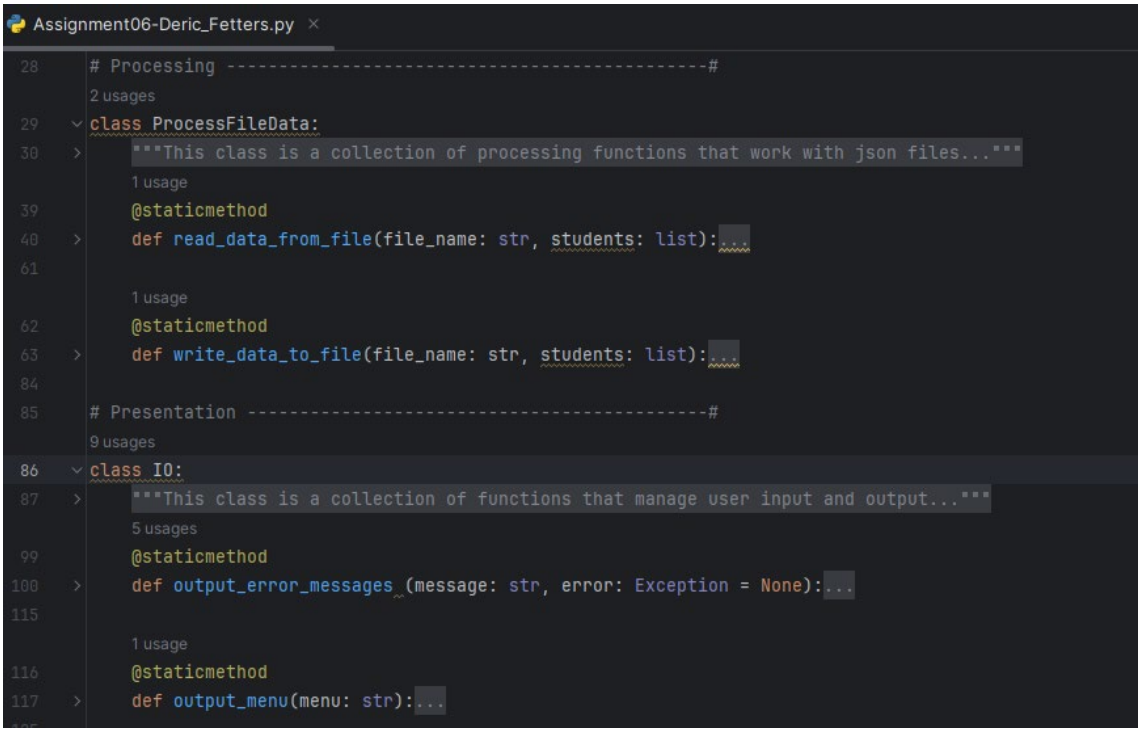

Figure 1: Assignment06.py Class organization (some code hidden)

Functions

Functions were used for modularity and reusability of code. The main program was greatly simplified by moving logic to functions and then calling those functions from the main program. See Figure 2.

```
while (True):

    # Present the menu of choices
    IO.output_menu(menu=MENU)
    # Request menu input from the user
    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=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
    elif menu_choice == "3":

        ProcessFileData.write_data_to_file( file_name: "FILE_NAME", 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 2: Assignment06.py Main Program

Function IO.output_error_messages is the best example of code reusability being called from the read_data_from_file, write_data_to_file and input_student_data functions. See Figure 3.

```
def read_data_from_file(file_name: str, students: list):
    """This function reads the data from the Enrollments.json file and returns list students..."""

    try:
        file = io.TextIOWrapper
        file = open(file_name, "r")
        students = json.load(file)
        file.close()
    except FileNotFoundError as e:
        IO.output_error_messages(message="Text file must exist before running this script", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
    finally:
        if file.closed == False:
            file.close()
    return students

1 usage
@staticmethod
def write_data_to_file(file_name: str, students: list):
    """This function writes the data to the Enrollments.json file..."""
    try:
        file = open(file_name, "w")
        json.dump(students, file)
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    except Exception as e:
        if file.closed == False:
            file.close()

        IO.output_error_messages(message="Error: There was a problem with writing to the file. \n"
                                         "Please check that the file is not open by another program.", error=e)
```

Figure 3: Assignment06.py IO.output_error_messages

Global vs Local Variables, Parameters, Arguments, Return Values

The use of global and local variables, parameters, arguments and return values has often caused me issues when programming. In this assignment, I was able to cut down global variables and constants down to four. See Figure 4.

```
# 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.
----------------------------------------
'''

FILE_NAME: str = "Enrollments.json"

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

Figure 4: Assignment06.py Global Variables

All other variables such as “student_first_name” and “student_last_name” were local variables within functions. See Figure 5 for an example.

```
def input_student_data(students: list):
    """ This function inputs student data from the user

    Change Log: (Who, When, What)
    DFetters, 05.30.2024, Created function

    """

    try:
        student_first_name = input("\n Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("\nThe last name should not contain numbers.")
        student_last_name = input("\n Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("\nThe last name should not contain numbers.")
        course_name = input("\n Please enter the name of the course: ")
        student_data = {"FirstName": student_first_name,
                        "LastName": student_last_name,
                        "CourseName": course_name}
        # adds new data to students list
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
    return students
```

Figure 5: Assignment06.py Local Variables

To update the global list “students” with new student data, the value of global list “students” was first passed to the “input_student_data” function as an argument where it was appended with the data input by the user in a local dictionary, named “student_data”. The updated local list “student_data” was returned by the function. The global list “students” value was then set equal local list “students”, updating the global list with the appended data. See figures 5 and 6.

```
            students = IO.input_student_data(students=students)
            continue
```

Figure 6: Assignment06.py Global List “students” update

GitHub

A GitHub account with a repository for classwork was created. It can be found at:

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

Summary

In Module 6, I learned about working with global vs local variables, classes, functions, parameters and return values. All of these topics improve the functionality, usability and shareability of our code.