Jake Jake Velasco

Feb. 19th, 2024

IT FDN 110 A Wi 24

Assignment 06

Organizing Code with Functions & Classes

Introduction

In this assignment the goal is to manipulate a previous Python program that demonstrates the use of constants, variables, and print statements to display a message about a student's registration for a Python course. More specifically there is an emphasis on using functions, classes, and the separation of concerns patterns (SoC).

Classes & SoC

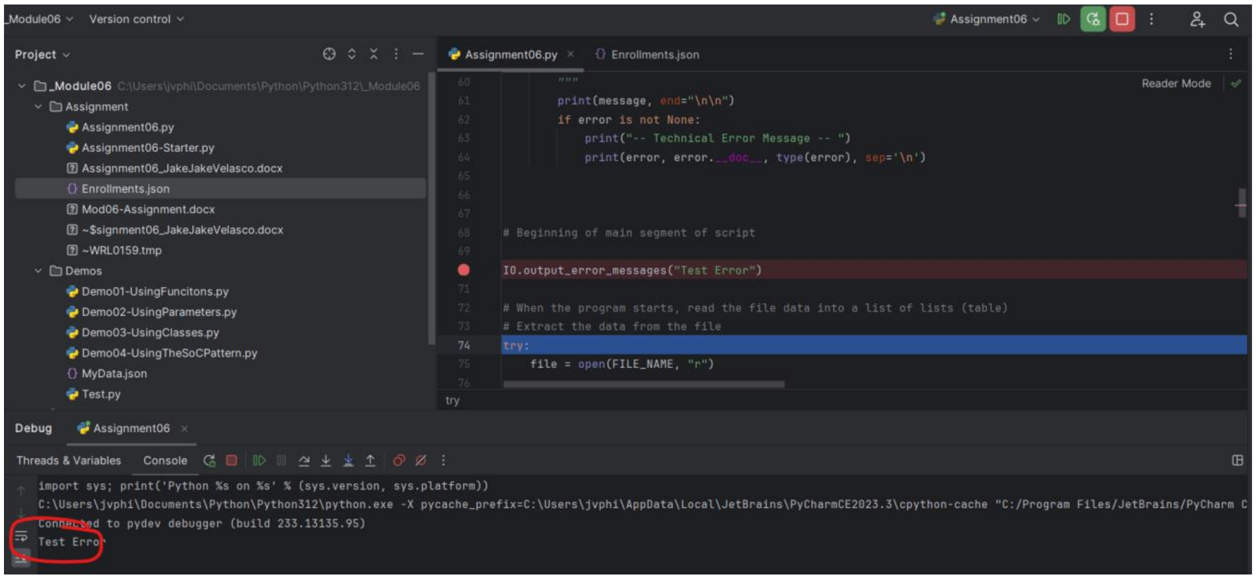
Mod06-Lab03: Working with Classes and SoC was heavily leveraged where it helped to setup and understand the use of class FileProcessor and class IO. These classes were inserted below the variable declarations. Then removed unnecessary comments within the comment strings from the Mod06-Lab03 classes example to cater it to my script.

```python
import json

# 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
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
```

```python
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.

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

    ChangeLog: (Who, When, What)
    Jake Jake Velasco,2.19.2024,Organizing Code and Classes
    """
    pass

# Presentation --------------------------------------- #
class IO:
    """
    A collection of presentation layer functions that manage user input and
output

    ChangeLog: (Who, When, What)
    Jake Jake Velasco,2.19.2024,Organizing Code and Classes
    """
    pass
```

Input/Output (IO) Functions and Error Handling

To increase efficiency, for the output error message function, I reused the static method within the lab examples because it already worked properly. As the old saying goes, “If it ain’t broke, don’t fix it.” Added in the input output error message to see if the debugger with via the console will print (test code) properly.



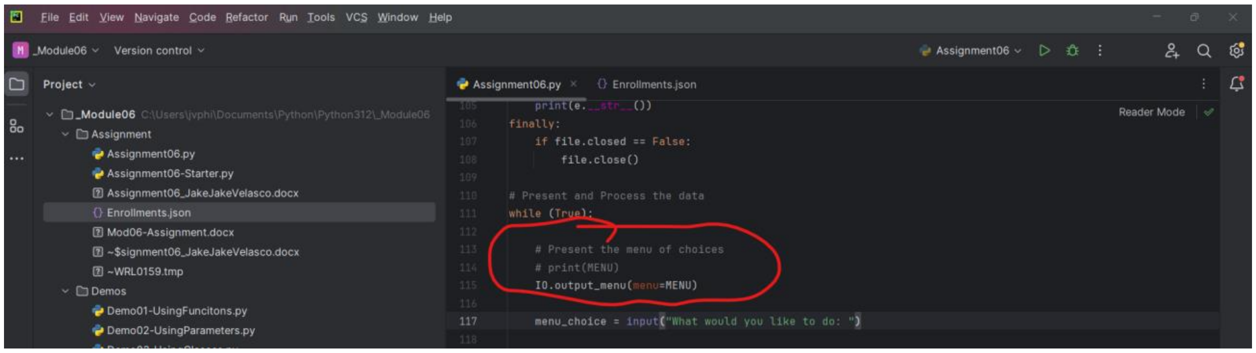
```python
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')



# Beginning of main segment of script

IO.output_error_messages("Test Error")

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
```

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
C:\Users\jvphi\Documents\Python\Python312\python.exe -X pycache_prefix=C:\Users\jvphi\AppData\Local\JetBrains\PyCharmCE2023.3\cpython-cache "C:/Program Files/JetBrains/PyCharm C
Connected to pydev debugger (build 233.13135.95)
Test Error
```

For the output menu function part of the assignment, to not spin my wheels, yet again I used an example from Mod06-Lab03 and indented as necessary so that the code will run with no issues.

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

    ChangeLog: (Who, When, What)
    Jake Jake Velasco,2.19.2024,Organizing Code and Classes

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

I then searched further into the code to locate within the present the menu of choices comment string to comment out the print for the MENU constant and insert the input output function for menu variable to pass onto the MENU constant.



```python
        print(e.__str__())
    finally:
        if file.closed == False:
            file.close()

# Present and Process the data
while (True):

    # Present the menu of choices
    # print(MENU)
    IO.output_menu(menu=MENU)

    menu_choice = input("What would you like to do: ")
```

For the input menu function setup, I leveraged the Mod06-Lab03 example and catered it to my liking. As expected, this process was repeated for setting up the remainder of this input output functions for the output student courses, input student data, read/write. For 4 grueling hours, I also leveraged the assignment review video and other fellow student github methods because I kept getting None Type value errors. The biggest issue was that I did not insert the return student data within my input student data function.

```python
# 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
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.

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

    ChangeLog: (Who, When, What)
    Jake Jake Velasco,2.19.2024,Organizing Code and Classes
    """
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file and loads it into a
list of dictionary rows

        ChangeLog: (Who, When, What)
        Jake Jake Velasco,2.19.2024,Organizing Code and Classes

        :return: list
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(messages="Error: There was a problem
with reading the file.", error=e)

        finally:
            if file.closed == False:
                file.close()
        return student_data

    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a lisf of
dictionary rows

             ChangeLog: (Who, When, What)
             Jake Jake Velasco,2.19.2024,Organizing Code and Classes

             :return: none
             """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            IO.output_student_and_course_names(student_data=student_data)
        except Exception as e:
            message= "Error: There was a problem with writing to the file.\n"
            message+= "Please check that the file is not open by another
program."
            IO.output_error_messages(message=message, error=e)
        finally:
            if file.closed == False:
                file.close()



# Presentation --------------------------------------- #
class IO:
    """
    A collection of presentation layer functions that manage user input and
```

```
output

    ChangeLog: (Who, When, What)
    Jake Jake Velasco,2.19.2024,Organizing Code and Classes
    """

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

        ChangeLog: (Who, When, What)
        Jake Jake Velasco,2.19.2024,Organizing Code and Classes

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

        ChangeLog: (Who, When, What)
        Jake Jake Velasco,2.19.2024,Organizing Code and Classes

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

    @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("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 e to avoid
the technical message
        return choice

    @staticmethod
    def output_student_and_course_names(student_data: list):
        """ This function displays the student registration entry data inputs
to user

        ChangeLog: (Who, When, What)
        Jake Jake Velasco,2.19.2024,Organizing Code and Classes
        Jake Jake Velasco,2.19.2024,Added code to toggle technical message
```

```
off if no exception object is passed
        :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
    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)
        Jake Jake Velasco,2.19.2024,Organizing Code and Classes

        :return: list
        """

        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The last 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()
            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!", error=e)
        except Exception as e:
            IO.output_error_messages(message="There was a non-specific
error!", error=e)
        return student_data

# Start Main Segments of Script

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file

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

# Present and Process the data
while (True):

    # Present the menu of choices
    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!
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_and_course_names(students)
        continue

    # 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, or 3")

print("Program Ended")
```

Summary

This assignment was by far the hardest concept to understand. Especially the use of the IO functions, passing the variables to the constants, and how to properly structure functions using the static methods decorator.