Nasmil Valera Cuevas
February 26th, 2024
Foundations of Programming, Python
Assignment 07
https://github.com/nasmilvc/IntroToProg-Python-Mod07

Classes and Objects

Introduction

In this assignment, I will go over how I created a Python program that demonstrates the use of core concepts of programming and version control. In this module, I learned about classes in more detail, and essential elements in object-oriented programming that enable us to create flexible and maintainable code. I also used things we have learned on past modules, such as data processing using lists and dictionaries, usage of JavaScript Object Notation (JSON) files and Structured Error Handling. This work was completed with the source material provided by Professor Randal Root, helpful demonstrations by Anubhaw Arya and a handful of external videos linked throughout the module.

Drafting the Code

I began by opening and reviewing the starter file Assignment07-Starter.py, and since I knew I would be working with JSON data, I imported the built-in JSON module onto the script. I also made sure the 'Enrollments.json' file had data.

Defining the Data Layer

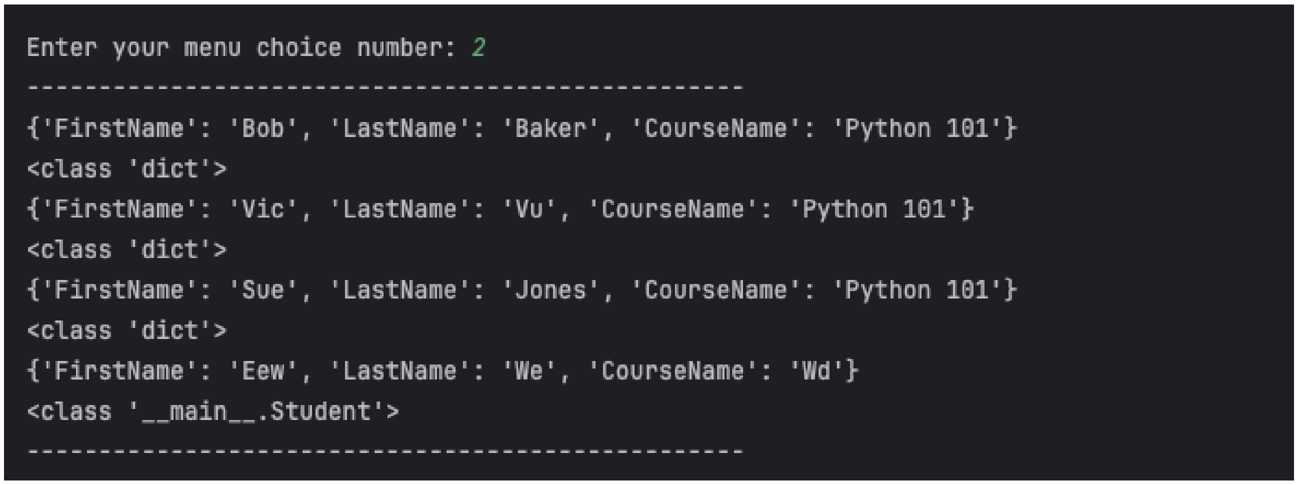
When defining the data items, I defined 'FILE_NAME' and 'MENU' as constants and 'students' and 'menu_choice' as global variables. Defining these as global variables allows them to be accessed and modified from anywhere within the script.

Using Inheritance

I created the Person class, and I created a constructor with private attributes for the student_first_name and student_last_name data, which allows the class to implicitly inherit code from Python's object class. I created the Person's class first name property (getter) and its respective setter, more formally known as "accessors" and "mutators". I also created a getter and setter for the Person's class last name property. Setter property functions allow us to add validation and error handling, as well as changing the syntax of the property. I overrode the __str__ () method in my class to customize it to return a more understandable and user-friendly string representation. The default shows the memory address of the object, so I changed it to be a formatted string with student_first_name and last_name.

I created a Student class that inherits from the Person class. To do this, I called the Person constructor and passed it the student_first_name and last_name by using the super() method. Then, I added the course_name attribute since that is unique to students. I added a getter and a setter for the course_name property. I overrode the __str__ () method in the Students class with the student first name, last name and course name to return them as a string dictionary.

While debugging, I would always get errors no matter what I changed so I went a little deeper to find the cause. The program was not able to run the functions because the new student information I was adding was of a different type than the information being read from the json file (Figure 1).

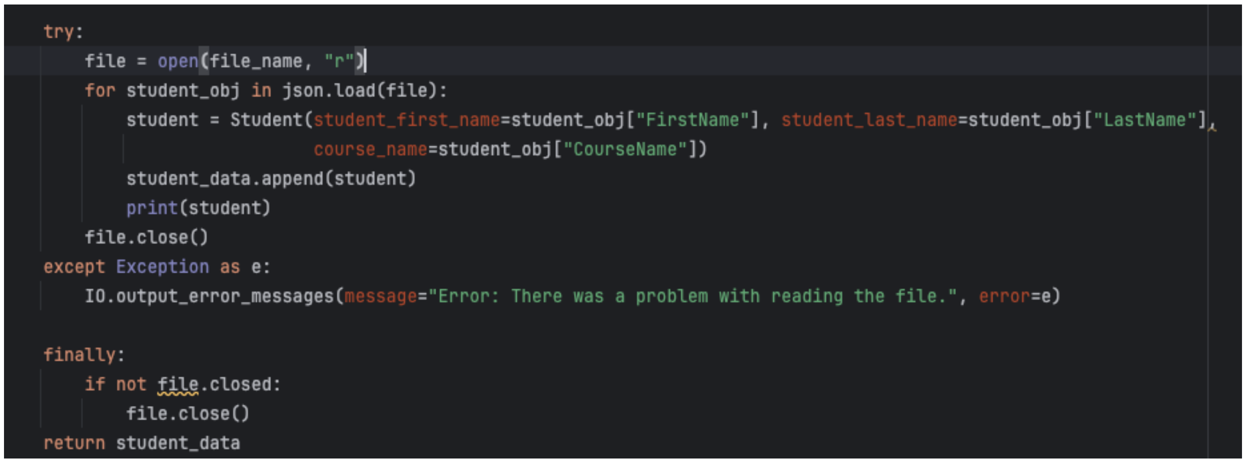


```
Enter your menu choice number: 2
--------------------------------------------------
{'FirstName': 'Bob', 'LastName': 'Baker', 'CourseName': 'Python 101'}
<class 'dict'>
{'FirstName': 'Vic', 'LastName': 'Vu', 'CourseName': 'Python 101'}
<class 'dict'>
{'FirstName': 'Sue', 'LastName': 'Jones', 'CourseName': 'Python 101'}
<class 'dict'>
{'FirstName': 'Eew', 'LastName': 'We', 'CourseName': 'Wd'}
<class '__main__.Student'>
--------------------------------------------------
```

Figure 1. Image of class incompatibility faced when inputting new student information and what prompted the solution.

I went to the FileProcessor class and the read_data_from_file method and set a for loop that would change the method into reading the information in the file as an object of the Student class (Figure 2). In other words, changed student_data to have a list of Student objects instead of the dictionary that's in the json file.

The program ran smoothly since the data being added was now congruent with the one inside of the Student class.

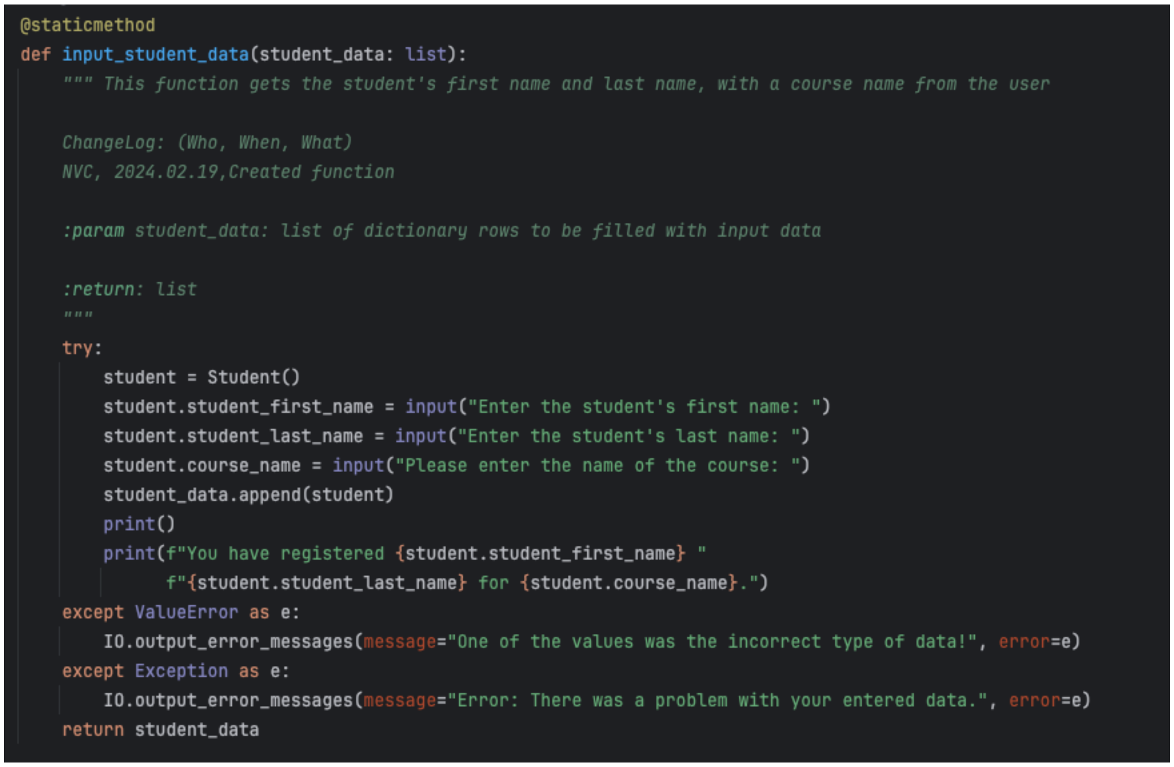


```
try:
    file = open(file_name, "r")
    for student_obj in json.load(file):
        student = Student(student_first_name=student_obj["FirstName"], student_last_name=student_obj["LastName"],
                          course_name=student_obj["CourseName"])
        student_data.append(student)
        print(student)
    file.close()
except Exception as e:
    IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)

finally:
    if not file.closed:
        file.close()
return student_data
```

Figure 2. Image of code from FileProcessor.read_data_from_file() where the class type of student_data was changed

I went over the program and updated the class function to all the variables (now properties), student_first_name, student_last_name and course_name, used. The properties were updated with their class attributes in the FileProcessor.write_data_to_file() method, IO.output_student_and_course_names() method, and IO.input_student_data() method.

I had to make the most changes to the IO.input_student_data() method because it had a lot of error handling code and it formatted the user input as a dictionary, both things that were already being taken care of in the Person and Student classes (Figure 3).



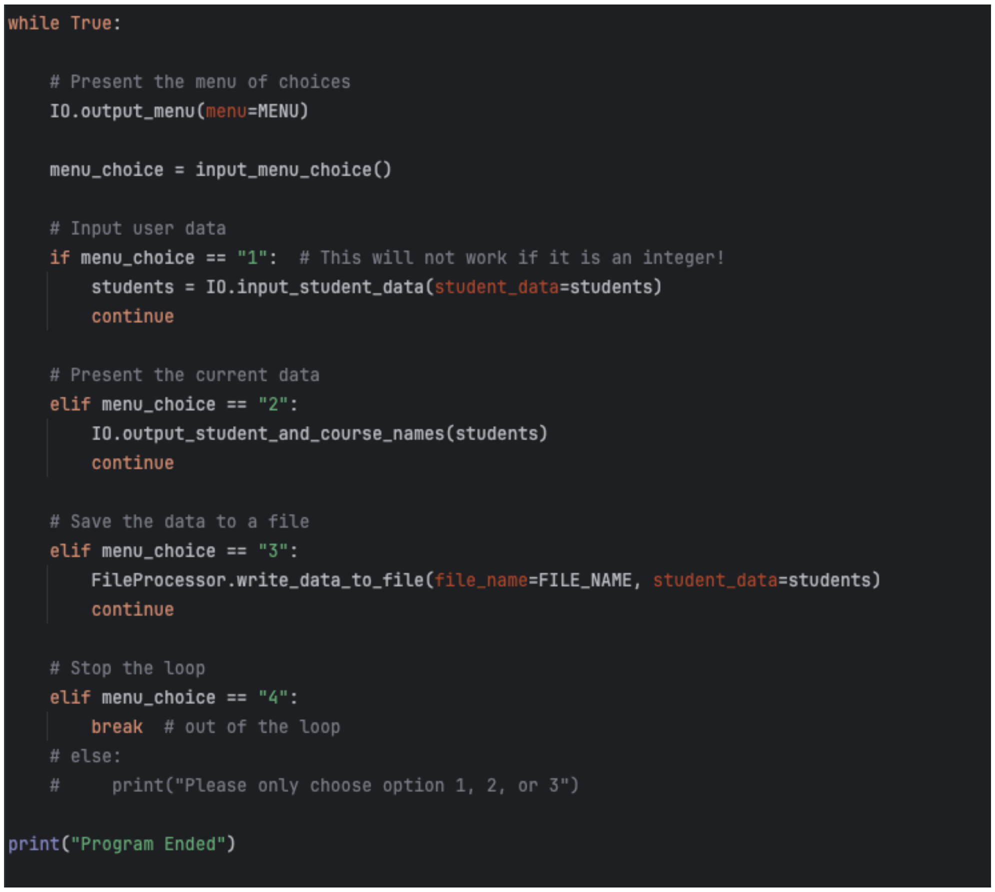
```
@staticmethod
def input_student_data(student_data: list):
    """ This function gets the student's first name and last name, with a course name from the user

    ChangeLog: (Who, When, What)
    NVC, 2024.02.19,Created function

    :param student_data: list of dictionary rows to be filled with input data

    :return: list
    """
    try:
        student = Student()
        student.student_first_name = input("Enter the student's first name: ")
        student.student_last_name = input("Enter the student's last name: ")
        student.course_name = input("Please enter the name of the course: ")
        student_data.append(student)
        print()
        print(f"You have registered {student.student_first_name} "
              f"{student.student_last_name} for {student.course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was the incorrect type of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
    return student_data
```

Figure 3. Image of code from IO.input_student_data() after redundant code was removed

Beginning of the Main Body of Script

When the program starts, the contents of the "Enrollments.json" file have to be automatically displayed in a two-dimensional list table so I called the read_from_file() function as students = 'FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)'. I added the class name before the function so that Python knows where to find the function. This also adds value to the 'students' variable that I will continue to use throughout the program.

The starter code provided the structure needed to plug in the functions where the code that defined them existed. Once I located the while loop, I added the output_menu() function from the IO class to display the menu to the user after every selection and defined the 'menu_choice' variable with the input_menu_choice() function from the IO class. Then, I continued to replace the existing code for the if and elif statements for menu_choice 1 through 3 with their respective functions (Figure 4). Menu choice 4 breaks the user out of the loop and after the loop is closed, the program notifies the user that the program has ended.

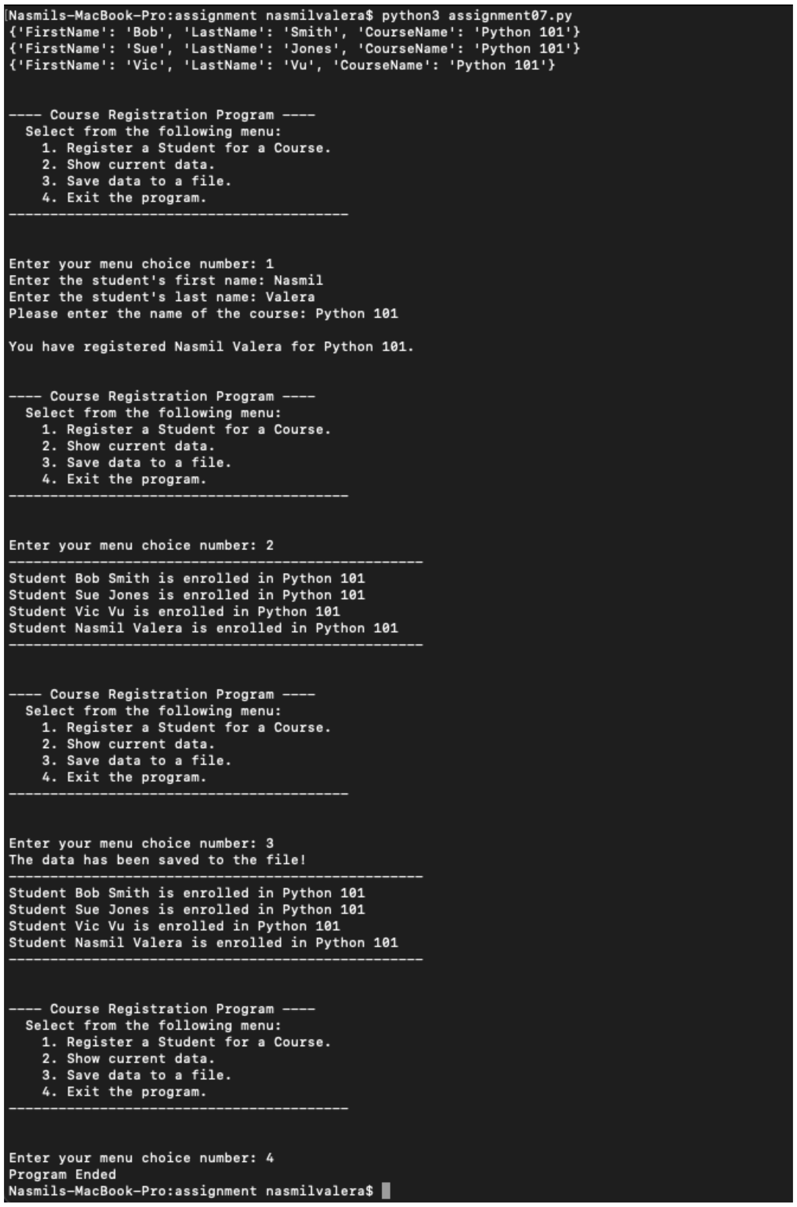


```
while True:

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

    menu_choice = 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")
```

Figure 4: Screenshot of while loop with recalled functions from classes FileProcessing and IO, after using inheritance

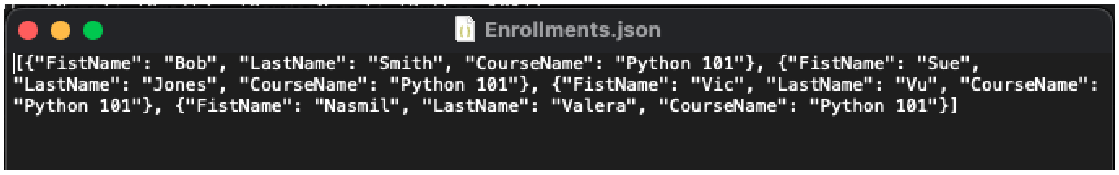
Running the Script in OS Command

I opened the Terminal console on my computer, navigated across directories and used the python3 command with file Assignment07.py to run my script. I followed the user prompts, and the script ran as expected (Figure 5), which was confirmed by the contents of the json file manipulated in this program (Figure 6).



```
Nasmils-MacBook-Pro:assignment nasmilvalera$ python3 assignment07.py
{'FirstName': 'Bob', 'LastName': 'Smith', 'CourseName': 'Python 101'}
{'FirstName': 'Sue', 'LastName': 'Jones', 'CourseName': 'Python 101'}
{'FirstName': 'Vic', 'LastName': 'Vu', 'CourseName': 'Python 101'}


---- 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: Nasmil
Enter the student's last name: Valera
Please enter the name of the course: Python 101

You have registered Nasmil Valera for Python 101.


---- 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 101
Student Sue Jones is enrolled in Python 101
Student Vic Vu is enrolled in Python 101
Student Nasmil Valera is enrolled in Python 101
--------------------------------------------------


---- 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
The data has been saved to the file!
--------------------------------------------------
Student Bob Smith is enrolled in Python 101
Student Sue Jones is enrolled in Python 101
Student Vic Vu is enrolled in Python 101
Student Nasmil Valera is enrolled in Python 101
--------------------------------------------------


---- 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
Nasmils-MacBook-Pro:assignment nasmilvalera$
```

Figure 5: Screenshot of program Assignment07.py running currently in the MacOS Terminal



```
[{"FistName": "Bob", "LastName": "Smith", "CourseName": "Python 101"}, {"FistName": "Sue",
"LastName": "Jones", "CourseName": "Python 101"}, {"FistName": "Vic", "LastName": "Vu", "CourseName":
"Python 101"}, {"FistName": "Nasmil", "LastName": "Valera", "CourseName": "Python 101"}]
```

Figure 6. Screenshot of file "Enrollments.json" text file with data from the program.

Summary

I was able to successfully create a Python program that demonstrated the use of core concepts of programming and version control. I showed my knowledge about classes and was able to distinguish dissecting data classes, presentation classes, and processing classes, while understanding their purpose. I also learned about essential elements in object-oriented programming that enable us to create flexible and maintainable code. I enjoyed the concept of class inheritance, that being said, I didn’t expect to have to change so many things in my code so I hope I didn’t overcomplicate it for myself.