Jeffrey Scott Hart

June 8th, 2025

Foundations of Python Programming

Assignment 07

https://github.com/mandubble/Python100-Spring-2025

Learning about: Getters , Setters, Objects, Classes, and the ancient tale of the Hierarchy of Inheritance!

In this module we studied object oriented programming or OOP. However makes no mistake, as OOP is not ‘oops!’ - it is very well thought out , useful and powerful unlike a mistake! We studied inheritance (by and of objects) and also the complex code templates that are necessary to carry these out. If ever there was a time to “dot your I's and cross your t’s” or should I say “underscore and double-underscore and dot your properties” this is it! Fast forward to my assignment – the beloved and coveted assignment_07 – I found I had some problems with making a double-underscore in a property ‘getter’ and a ‘double-self’ (which should never exist in the world of programming in any programming language yet). The thing about making mistakes on the object code is the problems turn up elsewhere and the errors won’t tell you what is what. I found this out the hard way!

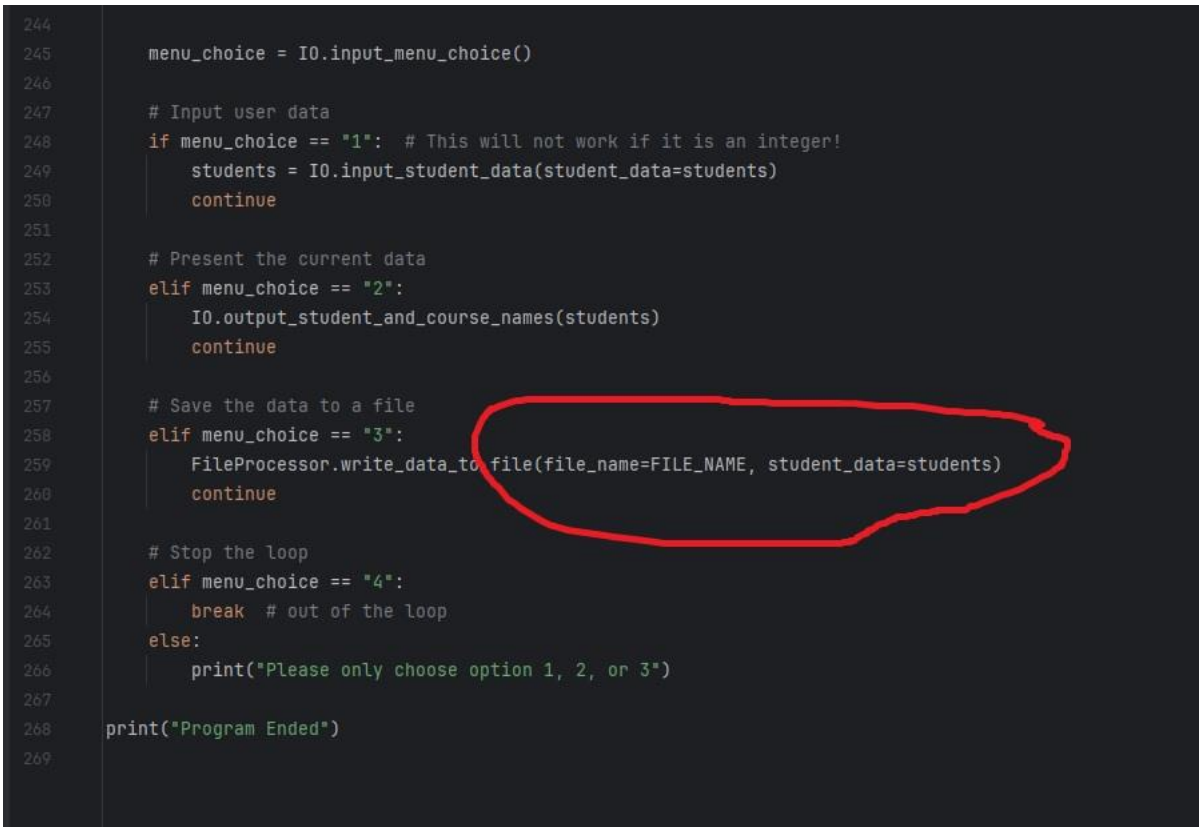
So getting back to the notes, videos and extended reading exercises: we studied about getters and setters and I found these ideas intriguing as you can’t change information/data/code inside an object directly. It is a protection mechanism , probably because there can be a lot going on inside such a seemingly small entity. It is like an engine in a car covered by a hood that unlatches inside the vehicle that is locked from the outside and you need a key to get into. There are as far as I can estimate 2-4 layers of security protecting an object.

I am not afraid of objects , OOP and classes. Randall Root gave numerous ‘trigger warnings’ about why and how not to be afraid of all of these things. However, I already delved not very deeply into object-oriented programming – definitely not to this extent in this module – some years ago. I also find inheritance interesting for the power of the convenience after you put in the work to lay down the train tracks.

So what I found out about the OOP code or *code for OOP*, as it were is that you really have to learn the patterns/templates as much of it is only visually *slightly* different. It is confusing in that a lot of it is the same looking and you aren’t sure if you just repeated three lines of code or not. I did not have the time to really spend twenty hours or more on this chapter, but I feel that one could easily spend thirty-five on it and ten hours could be spent just on practicing

the object code, the getter and setter code the parent/child transformative code until one really has the muscle memory imbedded deep concerning Module 7! This chapter deserves that. To harp a little more on a similar subject: I ran every single assignment, lab, lab answer, demo, class notes and assignment directions (all eight modules) through Claude AI asking it" how much time would you think it would take to learn all this stuff?" It came back with: 20-80 hours per module for the complexity , scope and advanced level of the material. Not to complain (too much) but yeah, I barely understand this stuff and I wish I had already mastered it. I DID talk to an associate about college courses and he said that in college the professors spray information at you like out of a firehose, and you are not expected to get all of it. Maybe I am greedy, but I want ALL the information and I don't want to waste any of it! All that information lost on the asphalt in the parking lot (or I don't know that's the best metaphor I could come up with). Anyway it's a sin to waste that information but alas! I forgot, the information is still here for five years! Whew. Okay, I need a glass of water and a time out. Moving right along....

So moving on to the ASSIGNMENT (drumroll, scary horror music...sighs ...gasps!) I must say that I really appreciated all of the TODOs. Somehow it makes it easier for me to just see everything (most everything) literally explained in a way I can understand. Where do I begin on this assignment? The object getter setter parts were pretty straightforward. I think that goes without saying as it is a pattern this is put down only one way, unlike f'-strings, .format, "for student in students" etc. I think getter and setter is pretty easy. Still I made critical errors which were not spotted by Pycharm directly. "Over-riding the __str__()" method was a little trickier and I had to get help on that. "Creating inheriting student class, with the 'super' , well I had to again, look that up. "Call to the Person constructor and pass it the first_name and last_name data" OUCH! At this point 'passing' anything (outside of the bathroom) is beyond me. (Fig.1) I don't get that yet and while we are on the subject of things I don't get I think I will go ahead and list some: tables: that's hard. Even though that was four modules ago, I am getting closer but still I'm not there yet. Student= student(s) (especially if passed as a parameter, nope not there yet!)



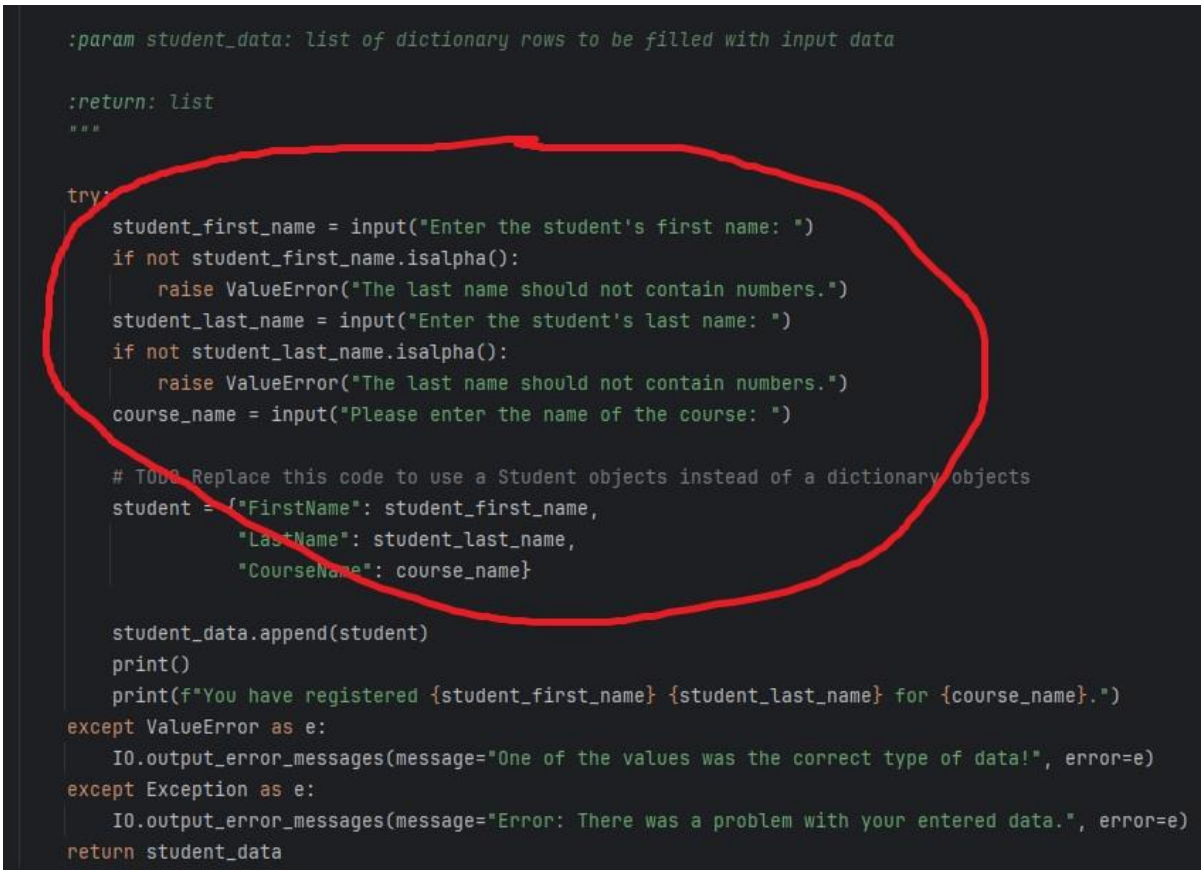
```
244
245     menu_choice = IO.input_menu_choice()
246
247     # Input user data
248     if menu_choice == "1":  # This will not work if it is an integer!
249         students = IO.input_student_data(student_data=students)
250         continue
251
252     # Present the current data
253     elif menu_choice == "2":
254         IO.output_student_and_course_names(students)
255         continue
256
257     # Save the data to a file
258     elif menu_choice == "3":
259         FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
260         continue
261
262     # Stop the loop
263     elif menu_choice == "4":
264         break  # out of the loop
265     else:
266         print("Please only choose option 1, 2, or 3")
267
268 print("Program Ended")
269
```

Figure 1. (passing stuff)

Adding in and changing over to objects loop on the assignment – as in fig.2 - and then the things you have to remove that you don't need anymore because the objects are simplifying the work. I have not yet grasped that. Also converting from dictionary rows to objects and lists of rows – fig. 3 – not there yet!

Back to the assignment: so basically again I had to resort to 'cleaning up my code by any means necessary'. (Sounds like something the military would say or Mission Impossible!) Well my code is pretty clean this time, if I say so myself. A lot of things that needed cleaning were: a couple of function parameters, "objects" should have been spelled "object" , typos in the property section and also turning the Person Object into a student object. Indentation problems which are the kill switch for any Python code. Some whole missing words in function calls/parameters . There was a lot of typing , correcting, researching, sending to AI for inspection and approval (some of which Pycharm missed, or because they were object/class related were not directly visible). However at the end of

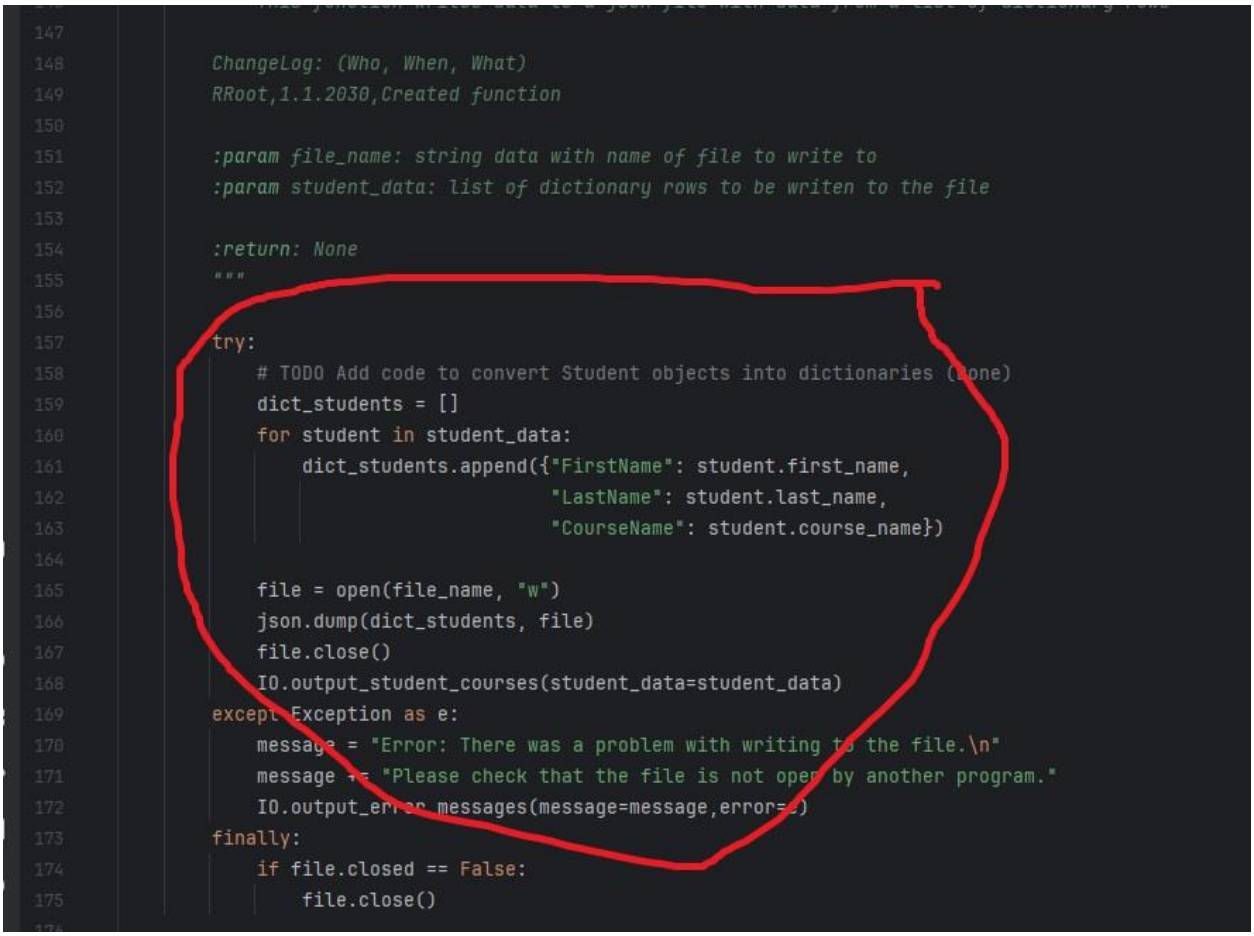
the day, end of the module almost end of the course I must say I rather enjoyed and could ‘get into’ and ‘dig’ the introspection on classes and objects even though I don’t know what a method or attribute is really. I just guided myself through with pattern recognition.



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

    :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: ")

        # TODO Replace this code to use a Student objects instead of a dictionary objects
        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="One of the values was the correct 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 2.

Figure 3.



```
147
148         ChangeLog: (Who, When, What)
149         RRoot,1.1.2030,Created function
150
151         :param file_name: string data with name of file to write to
152         :param student_data: list of dictionary rows to be writen to the file
153
154         :return: None
155         """
156
157         try:
158             # TODO Add code to convert Student objects into dictionaries (done)
159             dict_students = []
160             for student in student_data:
161                 dict_students.append({"FirstName": student.first_name,
162                                       "LastName": student.last_name,
163                                       "CourseName": student.course_name})
164
165             file = open(file_name, "w")
166             json.dump(dict_students, file)
167             file.close()
168             IO.output_student_courses(student_data=student_data)
169         except Exception as e:
170             message = "Error: There was a problem with writing to the file.\n"
171             message += "Please check that the file is not open by another program."
172             IO.output_error_messages(message=message,error=e)
173         finally:
174             if file.closed == False:
175                 file.close()
```