Alvin Jappeth Isip
December 7, 2024
IT FDN 110 A – Foundations of Programming: Python
Assignment 07

CLASSES AND OBJECTS

INTRODUCTION:

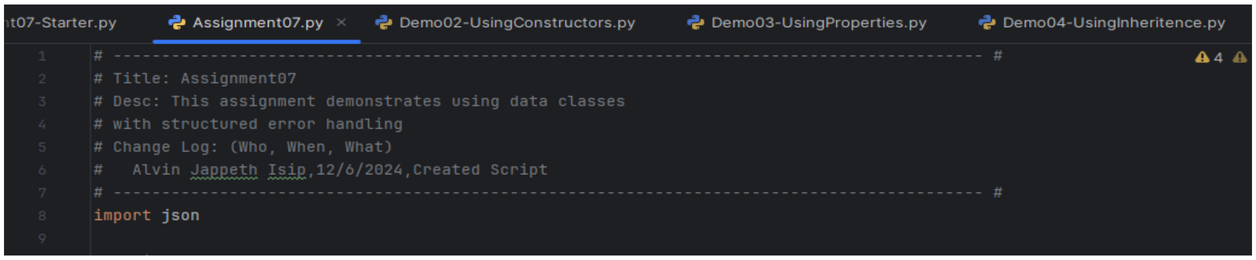
On this module, it was discussed classes and objects. We're also taught about the difference between statements, functions and classes. We're also taught about constructor, attribute, property, inheritance and overridden method.

MODULE 7 REQUIREMENTS

Requirement is like assignment 6 but with additional classes and class properties and methods.

WRITING THE SCRIPT

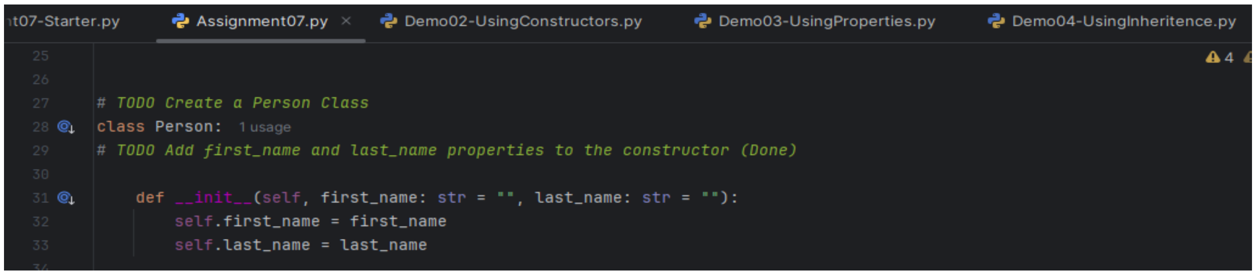
To start coding, I opened the Assignment07-Starter.py and saved it as Assignment07.py. I did changed the header script to reflect what I'll be doing for this assignment.


t07-Starter.py
Assignment07.py
Demo02-UsingConstructors.py
Demo03-UsingProperties.py
Demo04-UsingInheritence.py

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   Alvin Jappeth Isip,12/6/2024,Created Script
# ------------------------------------------------------------------------------------------ #
import json
```

Figure 1. Header Script

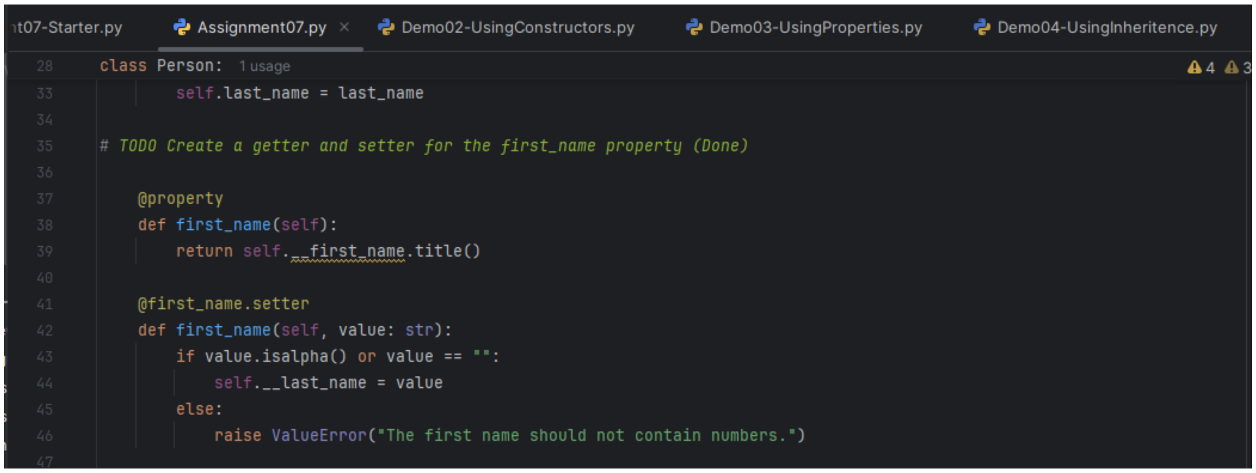
As mentioned on the requirement, the program needs to have a Class Person so I include Class Person on the script. Below that is I created a constructor for first name and last name. To start the constructor, I put __init__. This will initialize the object's attributes.



```
# TODO Create a Person Class
class Person:  1 usage
# TODO Add first_name and last_name properties to the constructor (Done)

    def __init__(self, first_name: str = "", last_name: str = ""):
        self.first_name = first_name
        self.last_name = last_name
```

Figure 2. Constructor for First Name and Last Name

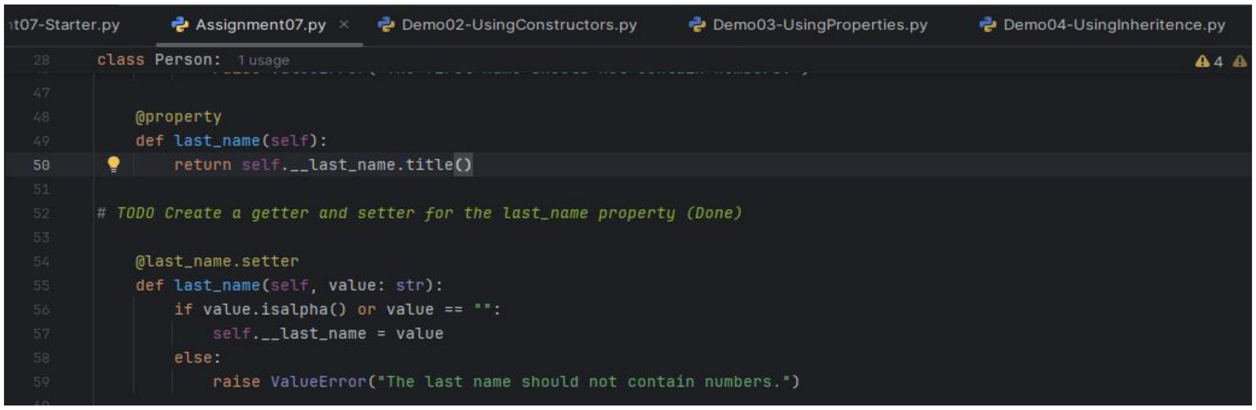
After doing the constructor, I created a getter and setter for first name and last name. To start the getter, I wrote @property. Getter enables us to access data while optionally applying formatting. Then for the setter, I wrote @name_of_property.setter. Setter function allows us to add validation and error handling.



```
class Person:  1 usage
        self.last_name = last_name

# TODO Create a getter and setter for the first_name property (Done)

    @property
    def first_name(self):
        return self.__first_name.title()

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":
            self.__last_name = value
        else:
            raise ValueError("The first name should not contain numbers.")
```

Figure 3. Getter and Setter for First Name

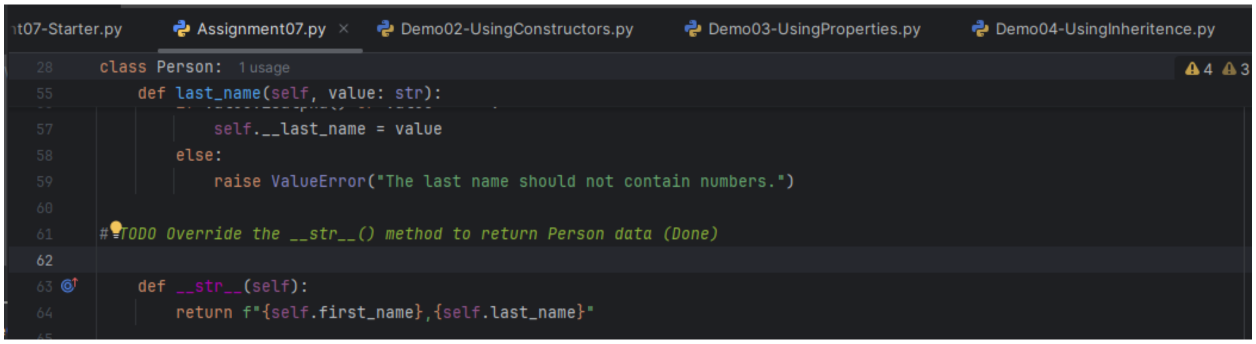
I did the same thing with the last name. I added getter and setter.



```
class Person:  1 usage

    @property
    def last_name(self):
        return self.__last_name.title()

# TODO Create a getter and setter for the last_name property (Done)

    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers.")
```

Figure 4. Getter and Setter for Last Name

Then to extract the Person's data, we added override by writing __str__().

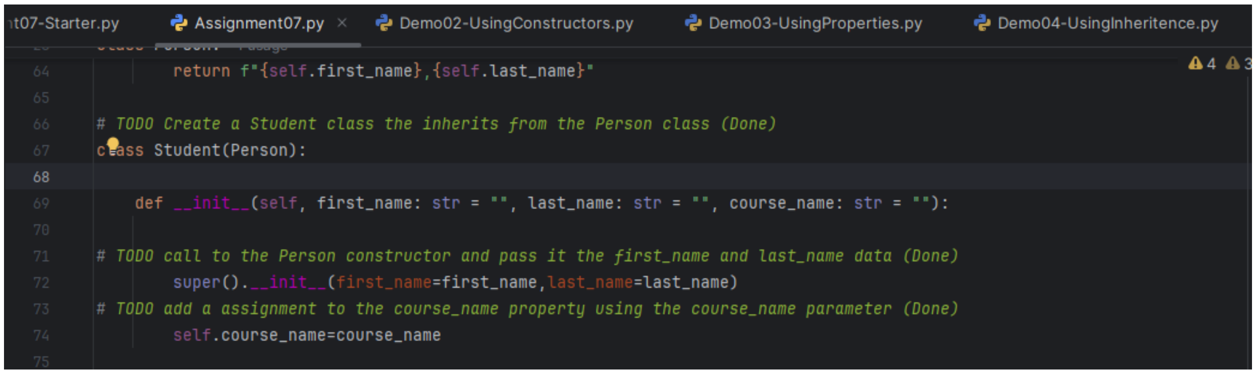


t07-Starter.py
Assignment07.py
Demo02-UsingConstructors.py
Demo03-UsingProperties.py
Demo04-UsingInheritence.py

```
class Person:  1 usage
    def last_name(self, value: str):
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers.")

#TODO Override the __str__() method to return Person data (Done)

    def __str__(self):
        return f"{self.first_name},{self.last_name}"
```

Figure 5. Override for first name and last name

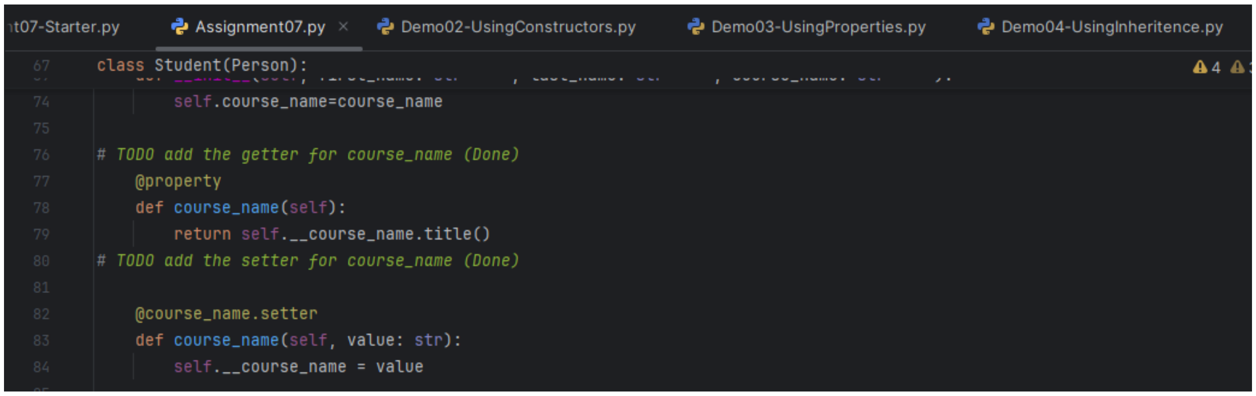
After coding the class Person, we now start to write the code for Class Student. This class will inherit data and behaviors from Person class.



t07-Starter.py
Assignment07.py
Demo02-UsingConstructors.py
Demo03-UsingProperties.py
Demo04-UsingInheritence.py

```
        return f"{self.first_name},{self.last_name}"

# TODO Create a Student class the inherits from the Person class (Done)
class Student(Person):

    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):

# TODO call to the Person constructor and pass it the first_name and last_name data (Done)
        super().__init__(first_name=first_name,last_name=last_name)
# TODO add a assignment to the course_name property using the course_name parameter (Done)
        self.course_name=course_name
```

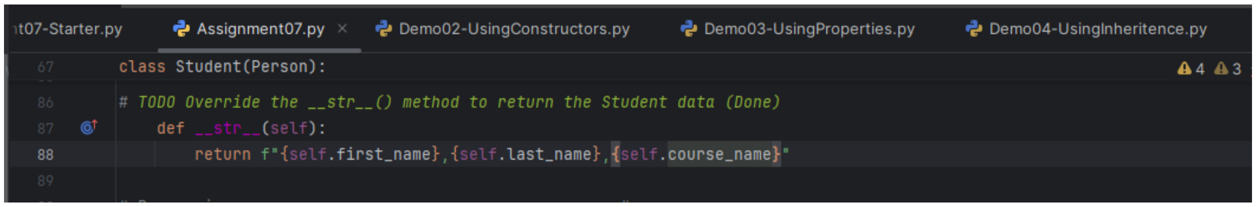
Figure 6. Class Inheritance for Student



t07-Starter.py
Assignment07.py
Demo02-UsingConstructors.py
Demo03-UsingProperties.py
Demo04-UsingInheritence.py

```
class Student(Person):
        self.course_name=course_name

# TODO add the getter for course_name (Done)
    @property
    def course_name(self):
        return self.__course_name.title()
# TODO add the setter for course_name (Done)

    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value
```

Figure 7. Added getter and setter for course_name.

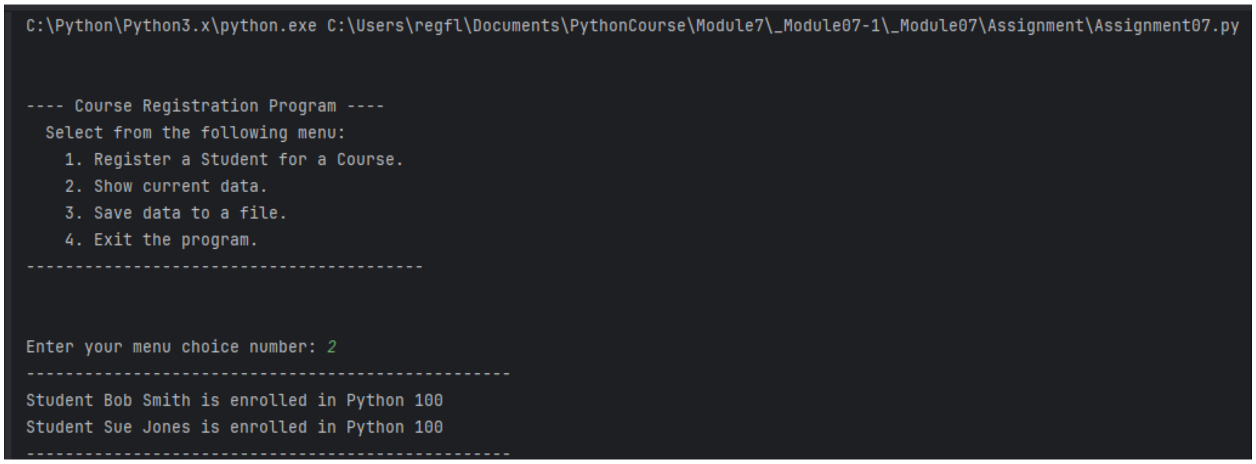


```
t07-Starter.py   Assignment07.py ×   Demo02-UsingConstructors.py   Demo03-UsingProperties.py   Demo04-UsingInheritence.py
class Student(Person):

# TODO Override the __str__() method to return the Student data (Done)
    def __str__(self):
        return f"{self.first_name},{self.last_name},{self.course_name}"
```

Figure 8. Override for first name, last name and course name.

TESTING - PYCHARM

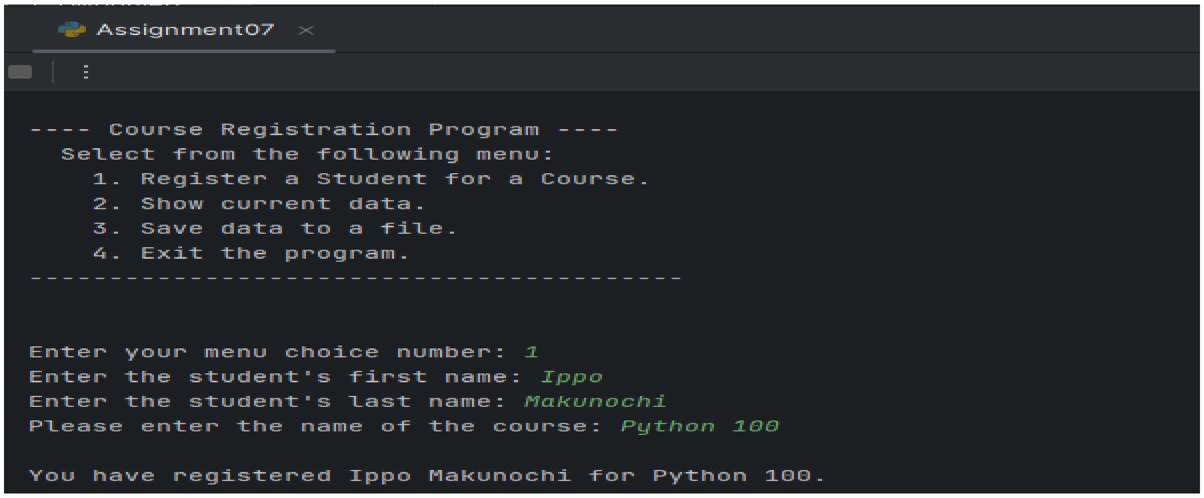
I did the testing using PyCharm first. I selected menu 2 first to show the current data on the Enrollment.json file.



```
C:\Python\Python3.x\python.exe C:\Users\regfl\Documents\PythonCourse\Module7\_Module07-1\_Module07\Assignment\Assignment07.py


---- 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
--------------------------------------------------
```

Figure 9. PyCharm Menu 2

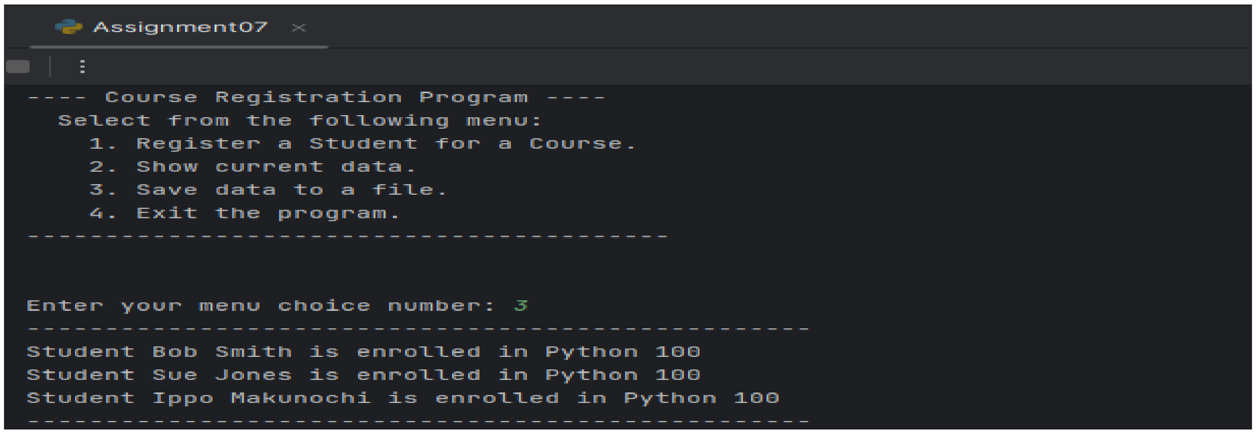
Then I selected menu 1 to add a record.



```
Assignment07 ×

---- 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: Ippo
Enter the student's last name: Makunochi
Please enter the name of the course: Python 100

You have registered Ippo Makunochi for Python 100.
```

Figure 10. PyCharm Menu 1

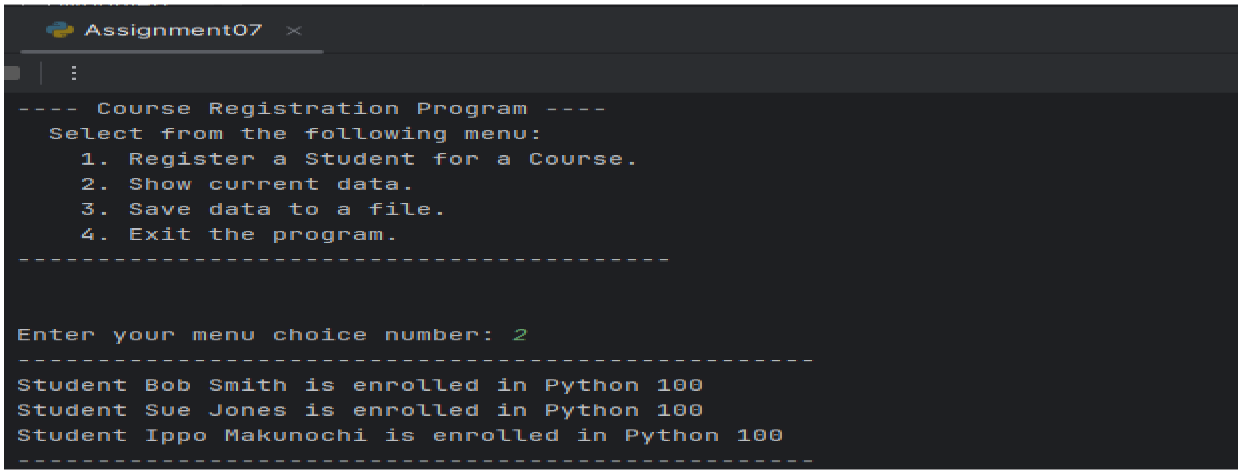
After that, I selected menu 3 to add a record.



```
Assignment07

---- 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
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Ippo Makunochi is enrolled in Python 100
--------------------------------------------------
```

Figure 11. PyCharm Menu 3

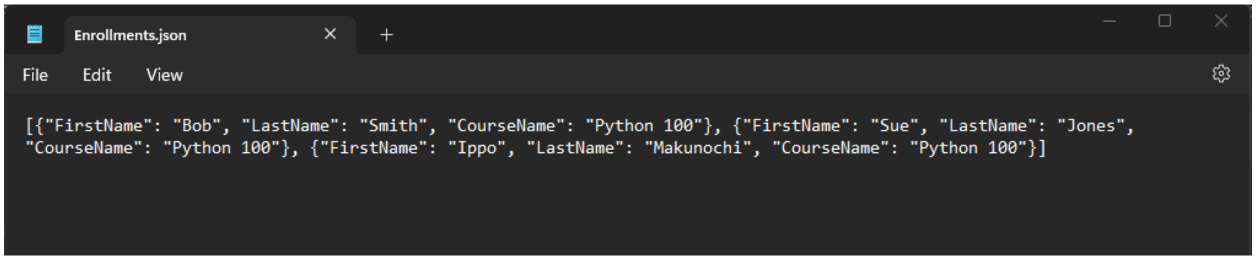
Then I selected menu 2 to see that the record has been added on the Enrollment.json file.



```
Assignment07

---- 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 Ippo Makunochi is enrolled in Python 100
--------------------------------------------------
```

Figure 12. PyCharm Menu 2 – After adding the record

I also checked the json file itself to see that the record has been added.

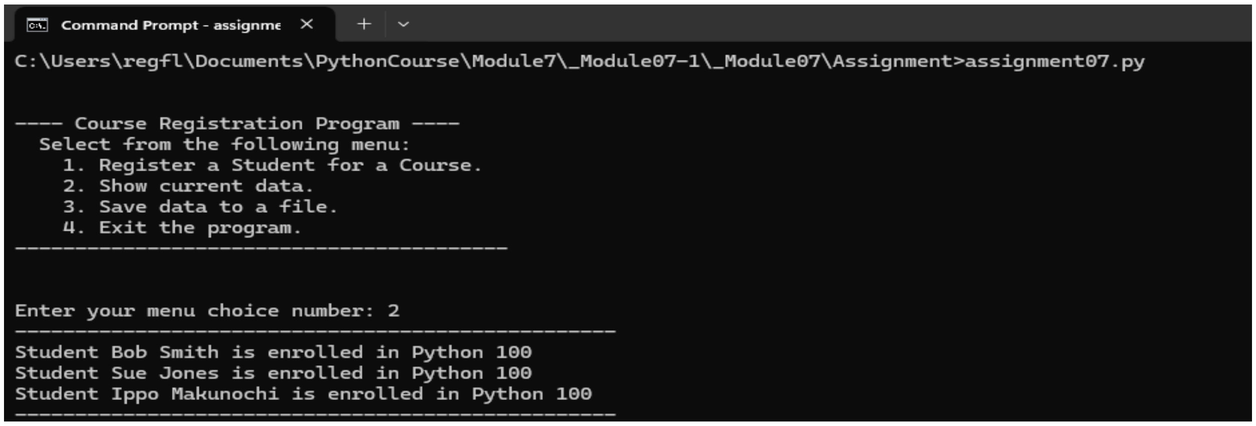


```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones",
"CourseName": "Python 100"}, {"FirstName": "Ippo", "LastName": "Makunochi", "CourseName": "Python 100"}]
```

Figure 13. Enrollment JSON file.

TESTING CMD

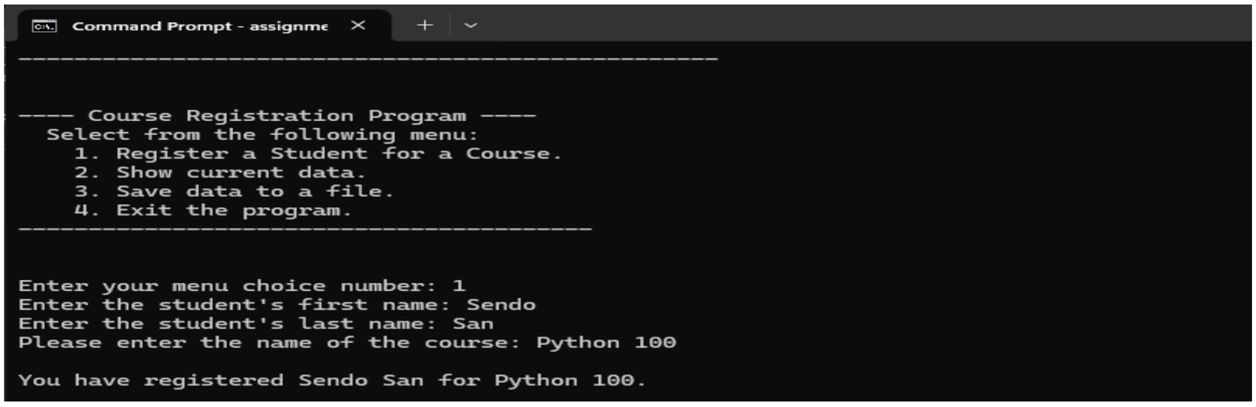
Selected menu 2 first to read the json file.



```
C:\Users\regfl\Documents\PythonCourse\Module7\_Module07-1\_Module07\Assignment>assignment07.py


---- 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 Ippo Makunochi is enrolled in Python 100
--------------------------------------------------
```

Figure 14. CMD – Menu 2

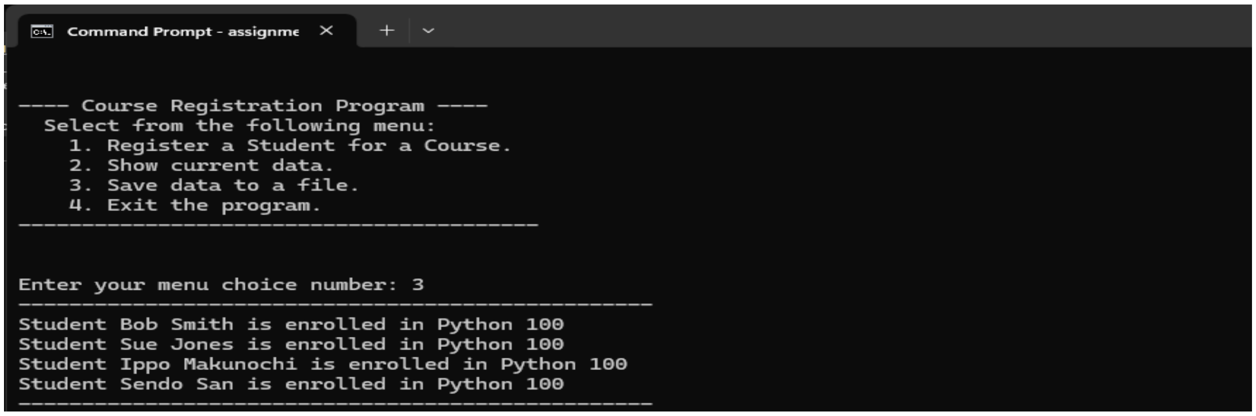
Selected menu 1 to register a new student



```
--------------------------------------------------


---- 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: Sendo
Enter the student's last name: San
Please enter the name of the course: Python 100

You have registered Sendo San for Python 100.
```

Figure 15. CMD – Menu 1

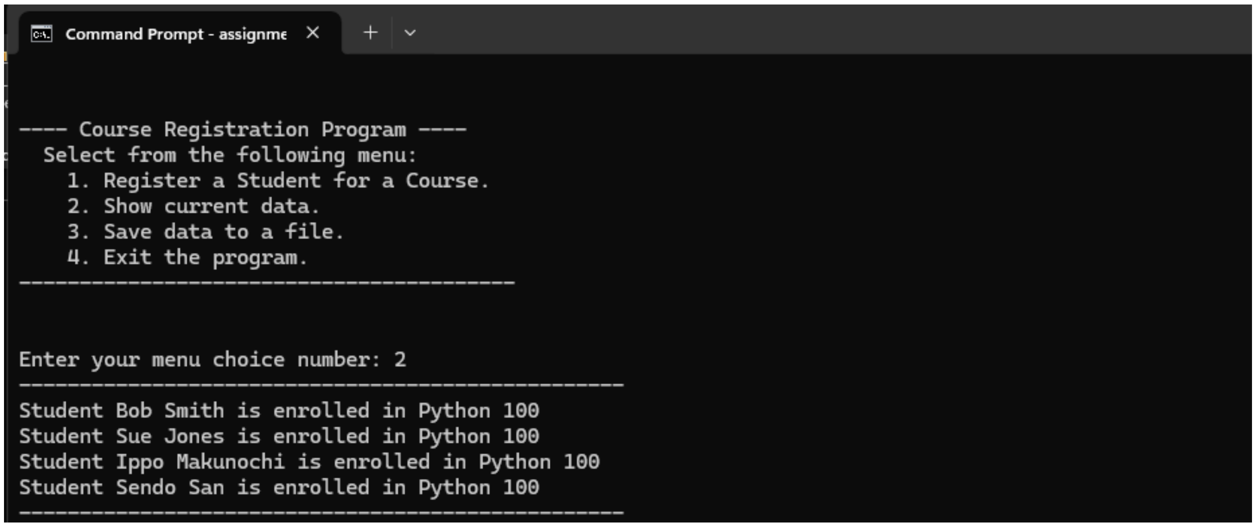
Selected menu 3 to add the new record on the file.



```


---- 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
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Ippo Makunochi is enrolled in Python 100
Student Sendo San is enrolled in Python 100
--------------------------------------------------
```

Figure 16. CMD – Menu 3

Then to check that the record has been added on the file, I selected menu 2 again.

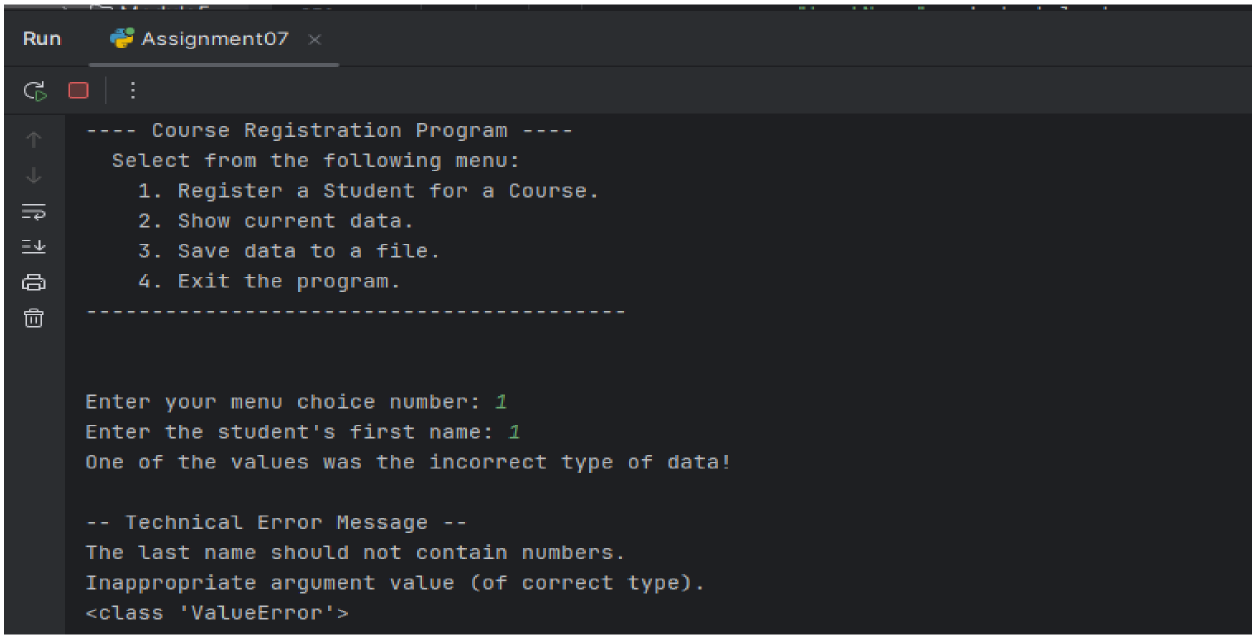


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Ippo Makunochi is enrolled in Python 100
Student Sendo San is enrolled in Python 100
--------------------------------------------------
```

Figure 17. CMD – Menu 2

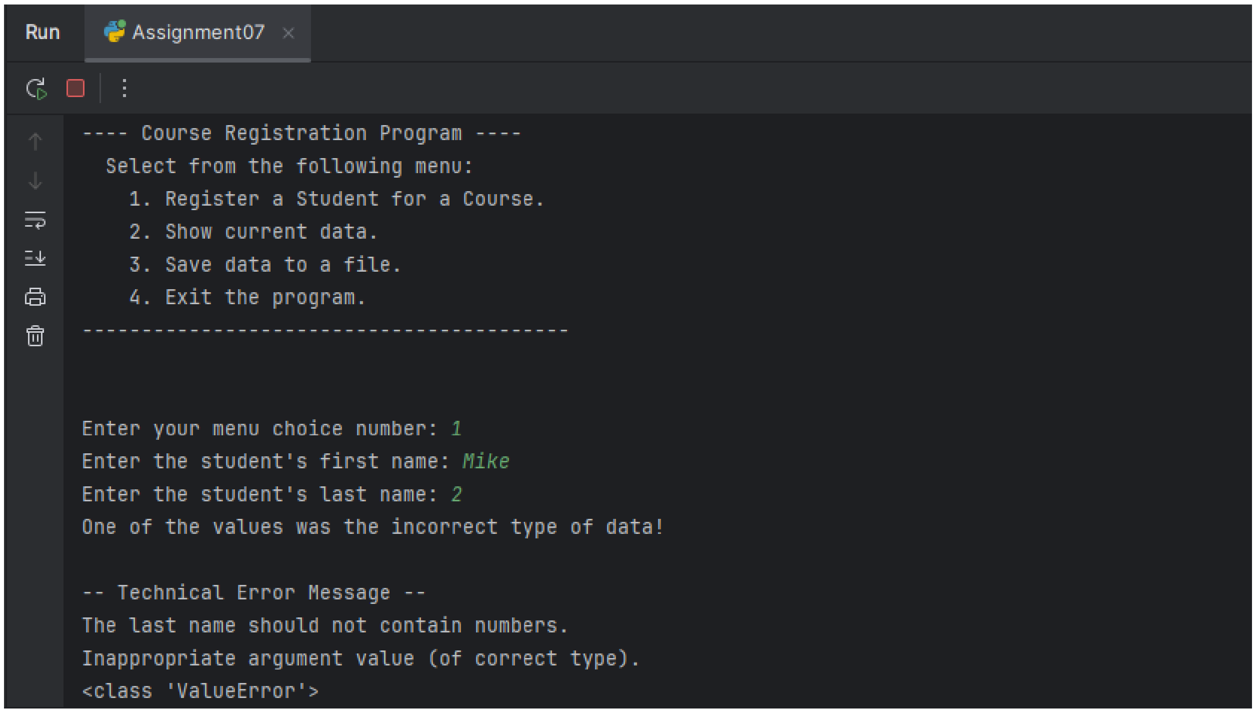
ERROR HANDLING

Below are the error handling that was included on the program.



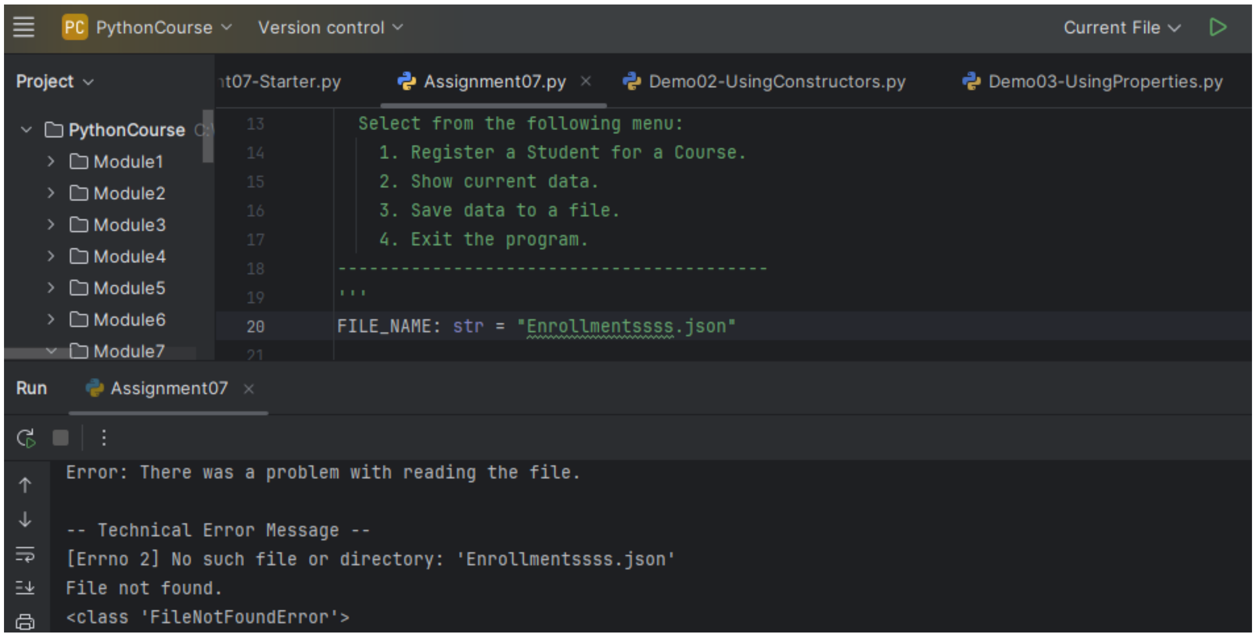
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: 1
One of the values was the incorrect type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Figure 18. Error handling for last name



```
---- 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: Mike
Enter the student's last name: 2
One of the values was the incorrect type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Figure 19. Error handling for first name



```
  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 = "Enrollmentssss.json"
```

```
Error: There was a problem with reading the file.

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollmentssss.json'
File not found.
<class 'FileNotFoundError'>
```

Figure 20. Error handling for json file

Source Control:

This is the link to my GitHub repository:

https://github.com/AJRIsip/IntroToProg-Python-Mod07

CONCLUSION:

On this module, we were able to learn how to create classes to organize and manage our data. We've learned the difference between statement, functions and classes. We were able to apply as well how to put getter and setter using @property and how can a class inherit a parents class attributes.