Mine Hantal Wernsing
5/26/2024
IT FDN110
A07 – Classes and Objects
Mine Wernsing GitHub

Creating and Testing a Python Script with Data Classes and Inheritance

Introduction

Python supports the object-oriented programming (OOP) paradigm through classes. Classes serve as a powerful tool for structuring and organizing code. They provide an elegant way to define reusable pieces of code that encapsulate data and behavior in a single entity. Inheritance is a core concept in OOP where a new class can inherit data and behaviors (properties and methods) from an existing class. In this Python script I've created and tested these concepts on PyCharm and Command Prompt.

Creating the Script

File Name:

The file is named assignment07_minewernsing.py (Fig. 1)

Script Header:

The script header includes the title, description, change log and is updated with name and the current date (Fig. 1).

Constants:

The constant **MENU: str** is set to the value:

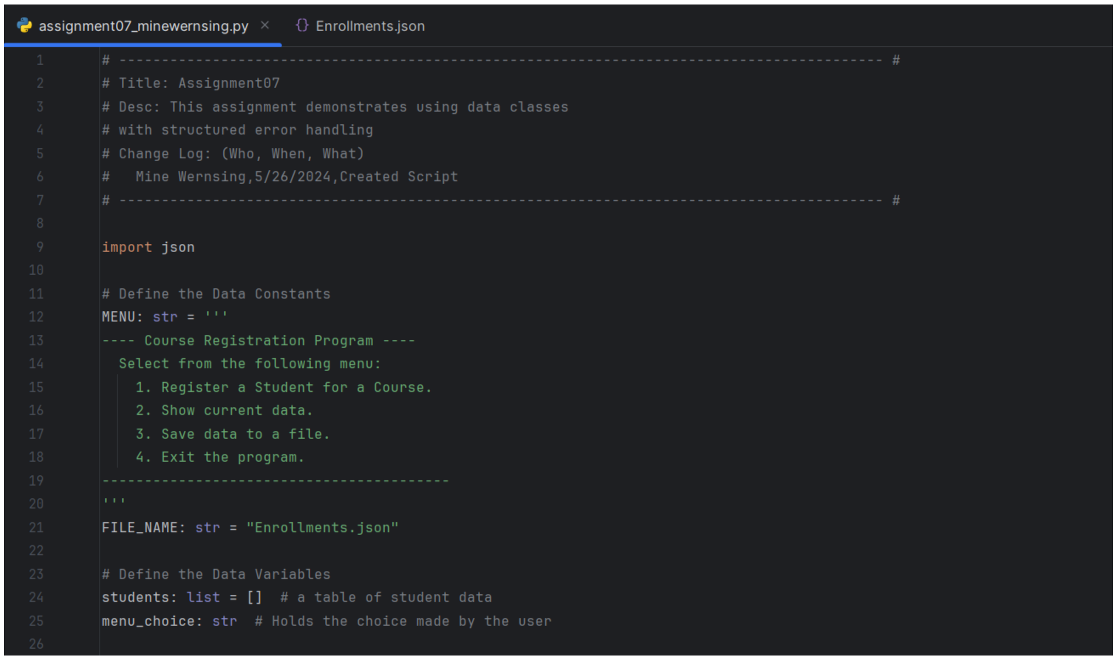
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------
```

The constant **FILE_NAME: str** is set to the value "Enrollments.json" (Fig. 1).

Variables:

menu_choice: str is set to a string

students: list = [] is set to an empty list (Fig. 1)



assignment07_minewernsing.py   Enrollments.json

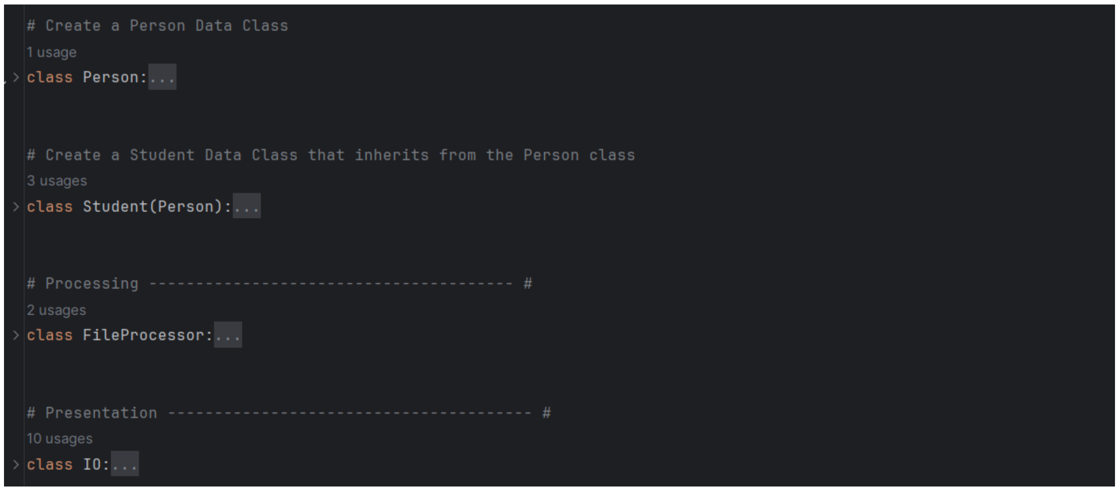
```
# ---------------------------------------------------------------------------------------- #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   Mine Wernsing,5/26/2024,Created Script
# ---------------------------------------------------------------------------------------- #

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
students: list = []  # a table of student data
menu_choice: str  # Holds the choice made by the user

```

Figure 1. File name, script header, defining data constants and variables.

Classes

The program includes classes Person, Student, FileProcessor and IO.

Person and Student classes are data classes. The Student class inherits from the Person class (parent class). The FileProcessor class is the processing class. IO (Input/Output) class is the presentation class (Fig. 2).



```
# Create a Person Data Class
1 usage
class Person:...


# Create a Student Data Class that inherits from the Person class
3 usages
class Student(Person):...


# Processing --------------------------------------- #
2 usages
class FileProcessor:...


# Presentation --------------------------------------- #
10 usages
class IO:...
```

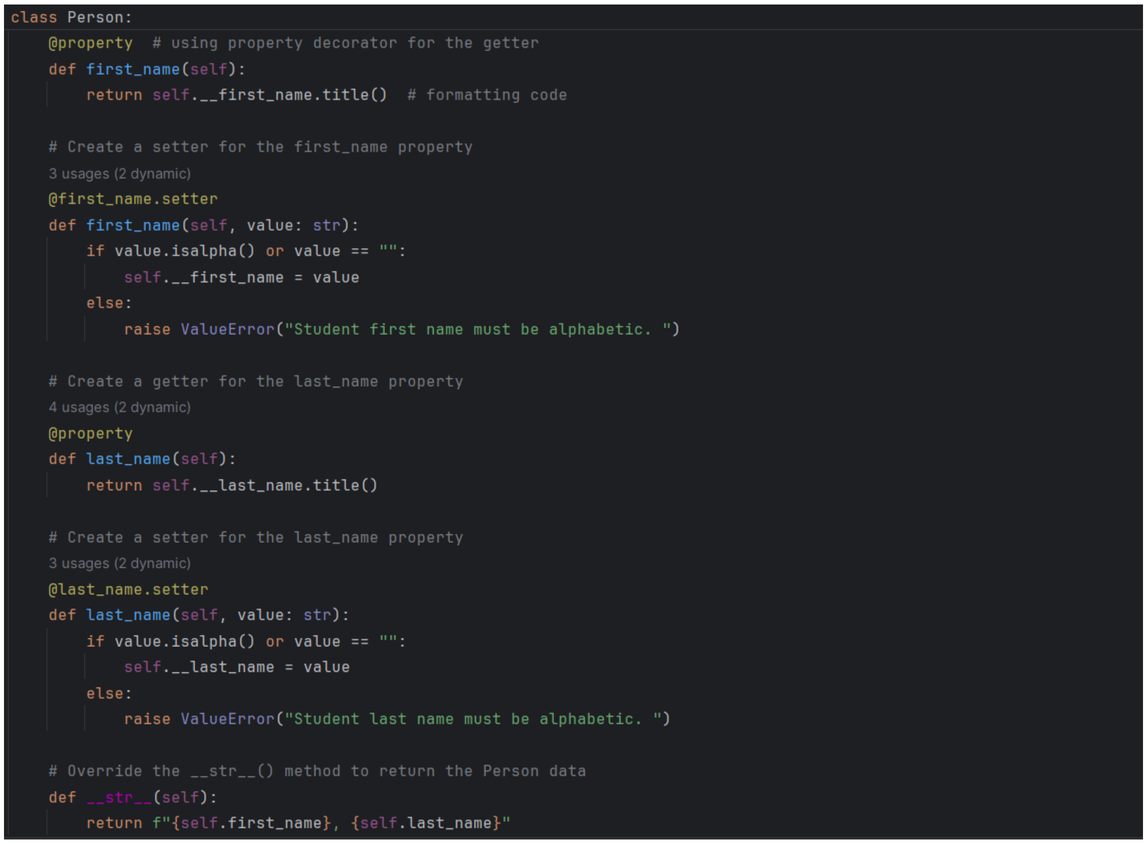
Figure 2. Creating the data, processing, and presentation classes.

For Person and Student(person) classes __init__ constructor is used.

The first_name, last_name, course_name properties are assigned by the constructor.

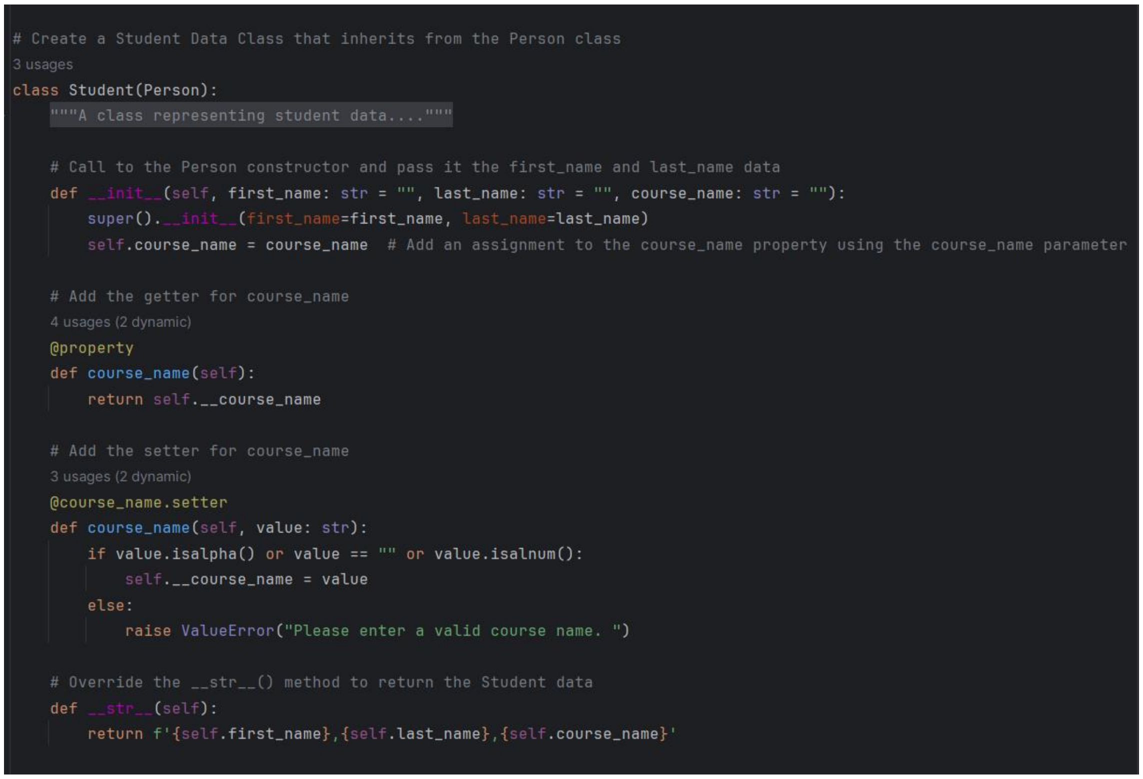
Setters and getters are created for the first_name, last_name and course_name properties in the class (Figs. 3 and 4).

Property decorator (@property) is used to create a getter.



```python
class Person:
    @property  # using property decorator for the getter
    def first_name(self):
        return self.__first_name.title()  # formatting code

    # Create a setter for the first_name property
    3 usages (2 dynamic)
    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":
            self.__first_name = value
        else:
            raise ValueError("Student first name must be alphabetic. ")

    # Create a getter for the last_name property
    4 usages (2 dynamic)
    @property
    def last_name(self):
        return self.__last_name.title()

    # Create a setter for the last_name property
    3 usages (2 dynamic)
    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":
            self.__last_name = value
        else:
            raise ValueError("Student last name must be alphabetic. ")

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

Figure 3. Creating setter and getter for the first_name and last_name in Person class.

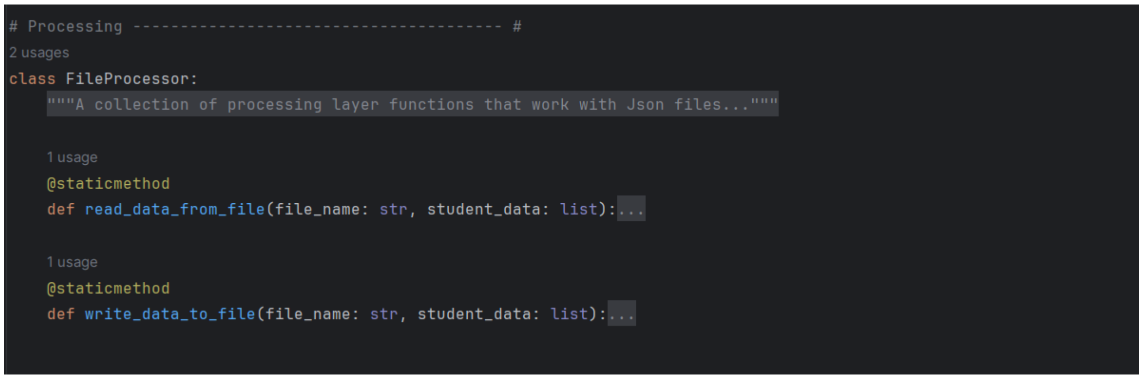


```
# Create a Student Data Class that inherits from the Person class
3 usages
class Student(Person):
    """A class representing student data...."""

    # Call to the Person constructor and pass it the first_name and last_name data
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name  # Add an assignment to the course_name property using the course_name parameter

    # Add the getter for course_name
    4 usages (2 dynamic)
    @property
    def course_name(self):
        return self.__course_name

    # Add the setter for course_name
    3 usages (2 dynamic)
    @course_name.setter
    def course_name(self, value: str):
        if value.isalpha() or value == "" or value.isalnum():
            self.__course_name = value
        else:
            raise ValueError("Please enter a valid course name. ")

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

Figure 4. Creating setter and getter for the course_name in Student(Person) class.

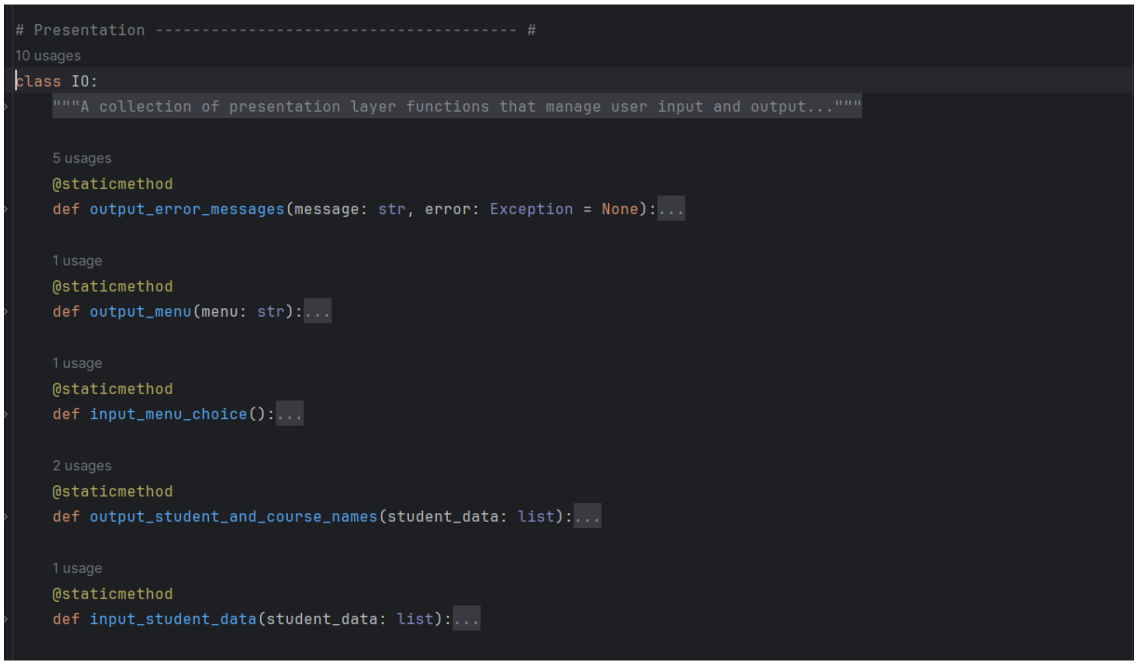
Functions

FileProcessor and IO class functions use the staticmethod decorator (Figs. 5 and 6).



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

    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):...

    1 usage
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):...
```

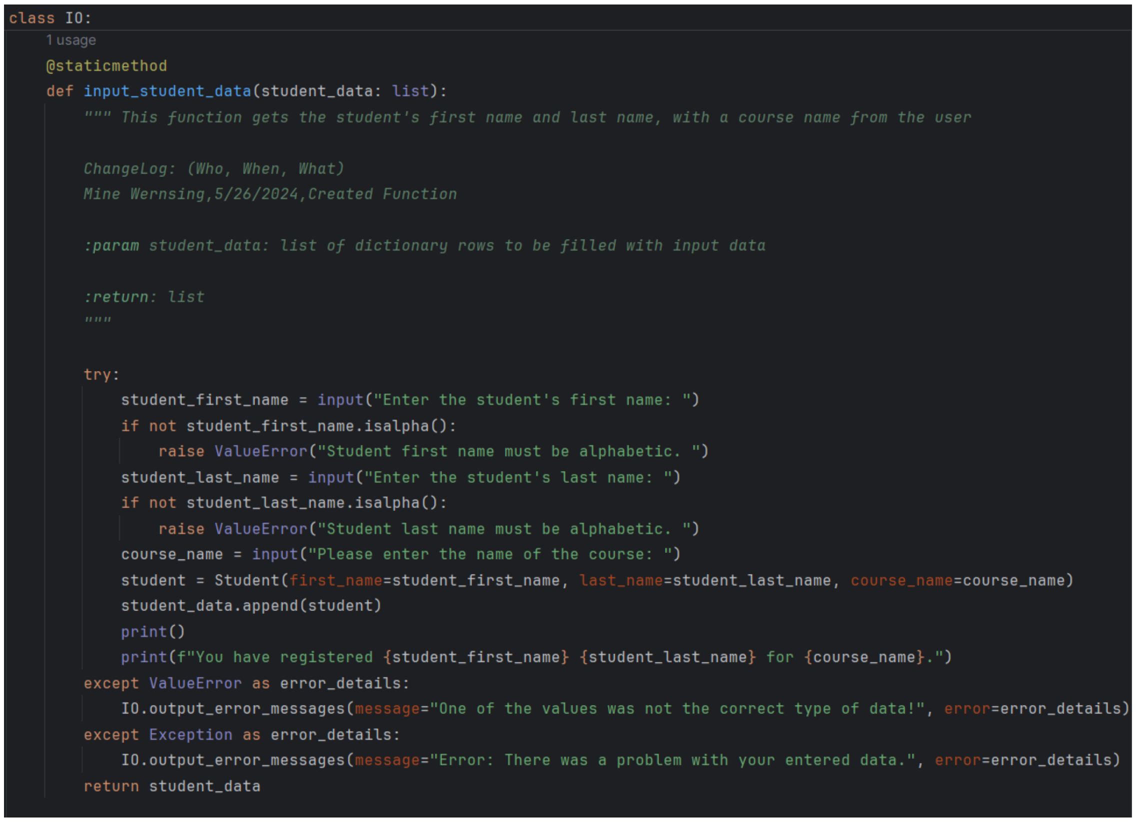
Figure 5. FileProcessor class with read_data_from_file and write_to_to_data functions.



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

    5 usages
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):...

    1 usage
    @staticmethod
    def output_menu(menu: str):...

    1 usage
    @staticmethod
    def input_menu_choice():...

    2 usages
    @staticmethod
    def output_student_and_course_names(student_data: list):...

    1 usage
    @staticmethod
    def input_student_data(student_data: list):...
```

Figure 6. IO class with its functions.

All functions except blocks include calls to the function handling error messages (Fig. 7).



```
class IO:
    1 usage
    @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)
        Mine Wernsing,5/26/2024,Created Function

        :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("Student first name must be alphabetic. ")
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Student last name must be alphabetic. ")
            course_name = input("Please enter the name of the course: ")
            student = Student(first_name=student_first_name, last_name=student_last_name, course_name=course_name)
            student_data.append(student)
            print()
            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        except ValueError as error_details:
            IO.output_error_messages(message="One of the values was not the correct type of data!", error=error_details)
        except Exception as error_details:
            IO.output_error_messages(message="Error: There was a problem with your entered data.", error=error_details)
        return student_data
```

Figure 7. Error handling.

Input / Output:

On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.

On menu choice 2, the program presents a string by formatting the collected data using the print() function.

Data collected for menu choice 1 is added to a list of Students.

All data in the list is displayed when menu choice 2 is used.

Processing

When the program starts, the contents of the "Enrollments.json" are automatically read into a list of Student objects.

On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. It writes the content of the students list to the file using the json.dump() function, then file is closed using the close() method. The program then displays what was stored in the file (Fig. 8 and 12).

On menu choice 4, the program ends (Fig. 12).

Error Handling

The program provides structured error handling when the file is read into the list of Student objects.

The program provides structured error handling when the user enters a first name (Figs. 7, 10 and 13).

The program provides structured error handling when the user enters a last name (Figs. 7, 10 and 13).

The program provides structured error handling when the list of Student objects are written to the file.

Testing the Script

Test

The program takes the user's input for a student's first, last name, and course name.

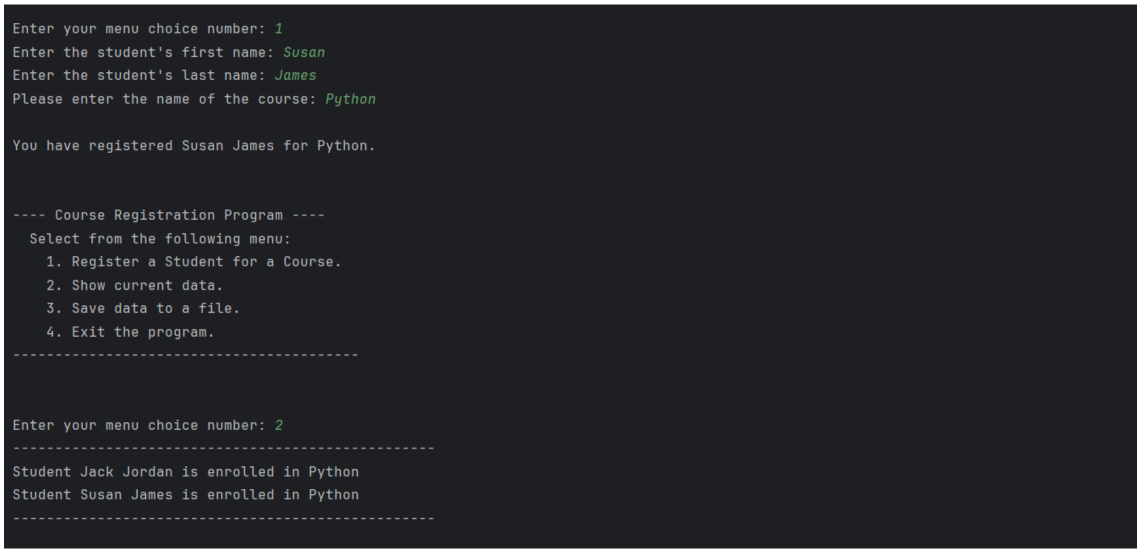
The program displays the user's input for a student's first, last name, and course name (Figs. 8 and 11).

The program allows users to enter multiple registrations (first name, last name, course name).

The program allows users to display multiple registrations (first name, last name, course name) (Figs. 8 and 11).

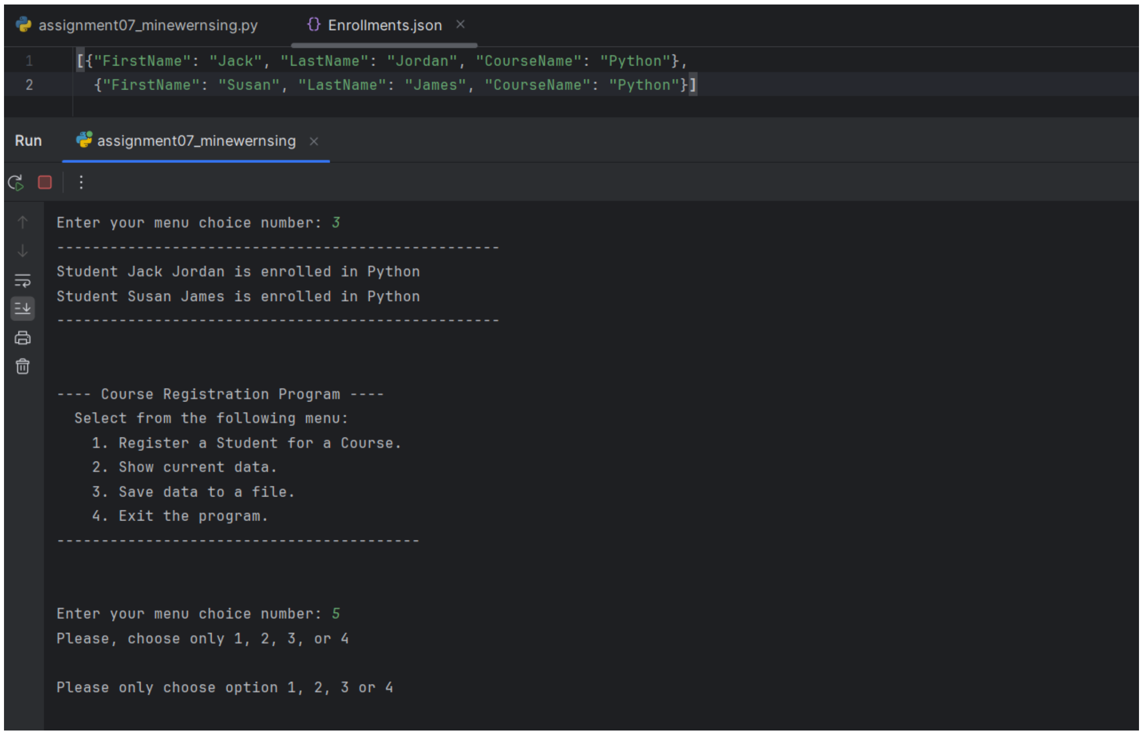
The program allows users to save multiple registrations to a file (first name, last name, course name).

The program runs correctly in both **PyCharm** and **Command Prompt**.



```
Enter your menu choice number: 1
Enter the student's first name: Susan
Enter the student's last name: James
Please enter the name of the course: Python

You have registered Susan James for Python.


---- 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 Jack Jordan is enrolled in Python
Student Susan James is enrolled in Python
--------------------------------------------------
```

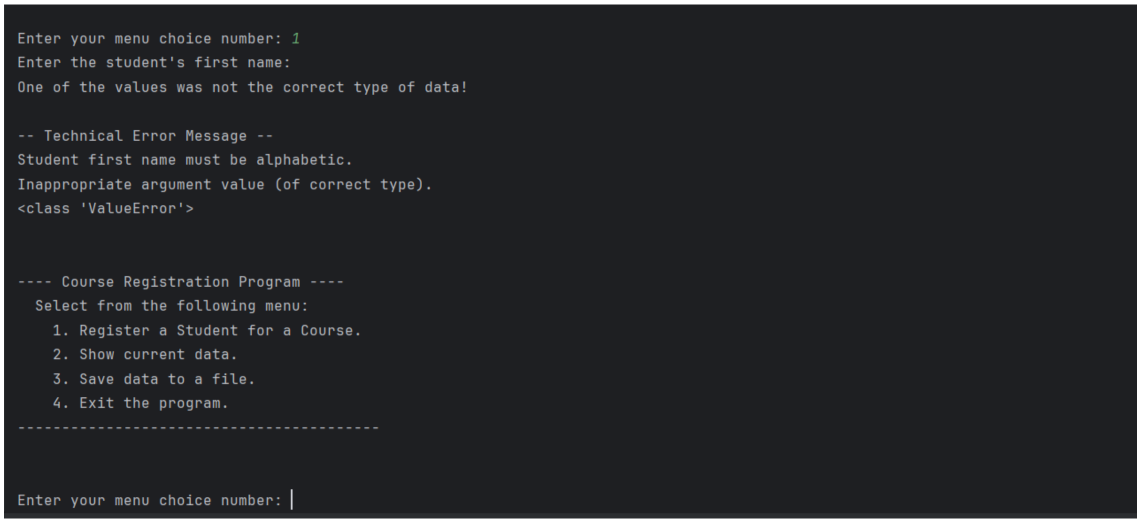
Figure 8. Testing menu options 1 and 2 in PyCharm.



```
[{"FirstName": "Jack", "LastName": "Jordan", "CourseName": "Python"},
  {"FirstName": "Susan", "LastName": "James", "CourseName": "Python"}]
```

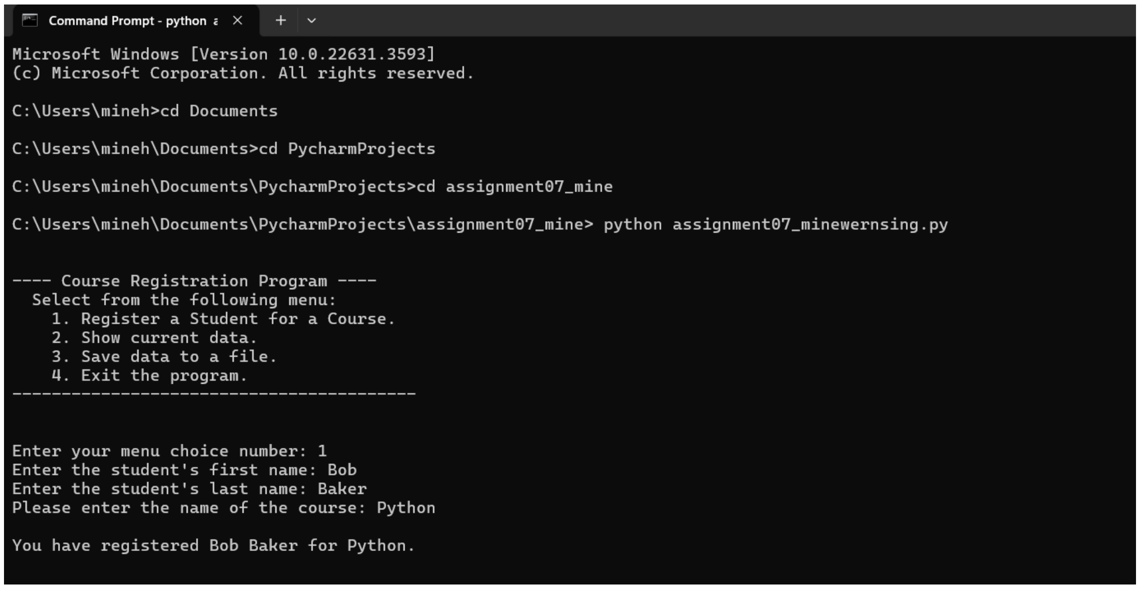
```
Enter your menu choice number: 3
--------------------------------------------------
Student Jack Jordan is enrolled in Python
Student Susan James is enrolled in Python
--------------------------------------------------



---- 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: 5
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, 3 or 4
```

Figure 9. Testing menu option 3 and error handling in PyCharm.



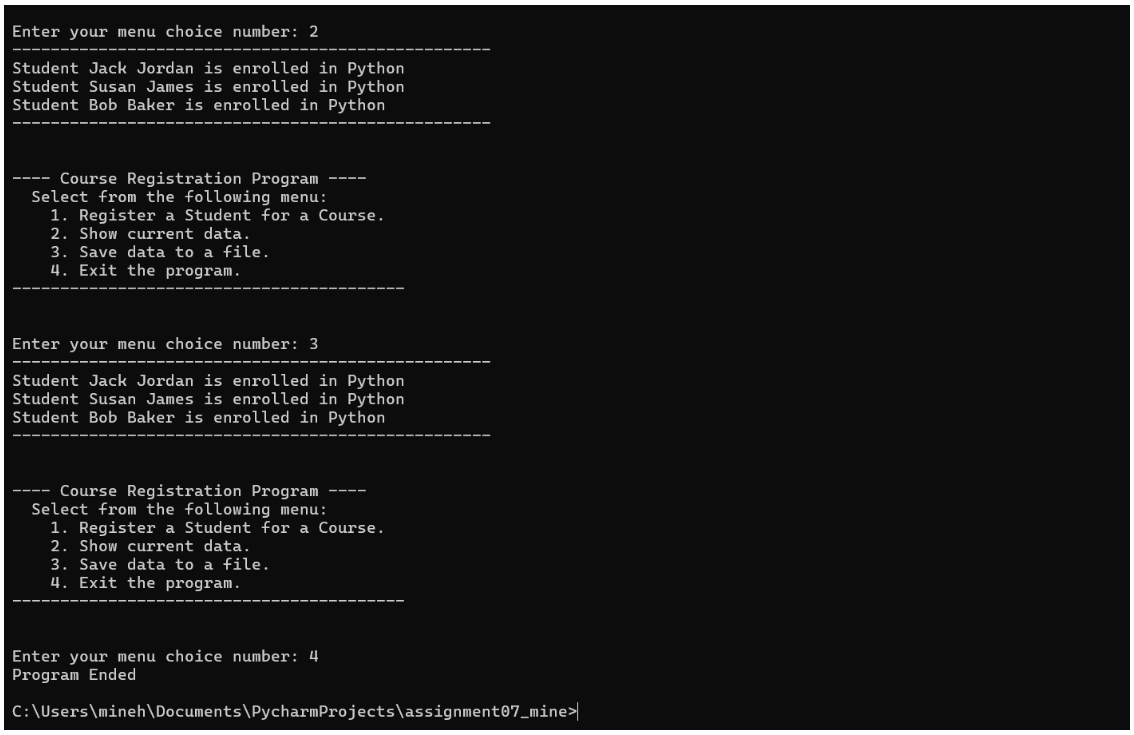
```
Enter your menu choice number: 1
Enter the student's first name:
One of the values was not the correct type of data!

-- Technical Error Message --
Student first name must be alphabetic.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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: 
```

Figure 10. Testing error handling in PyCharm.



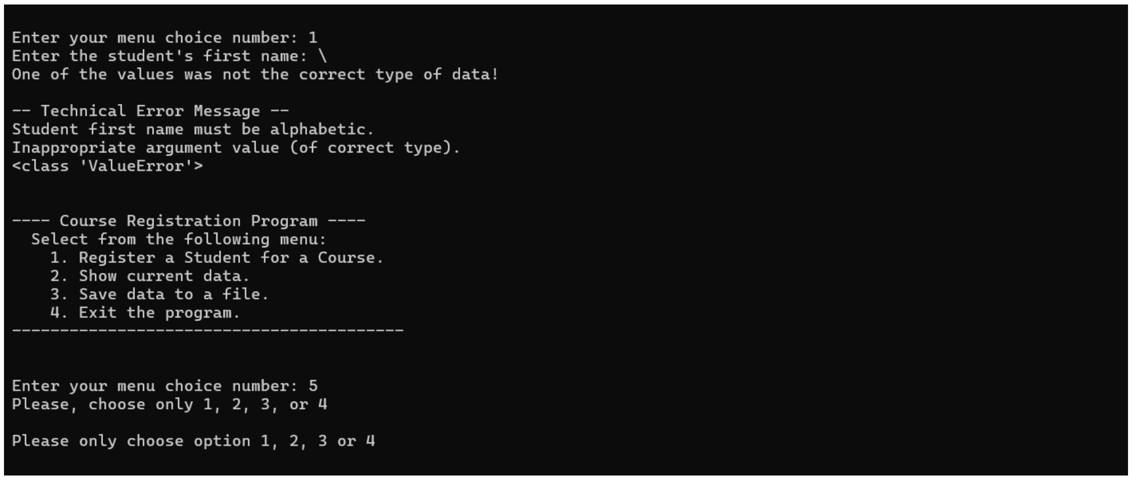
```
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mineh>cd Documents

C:\Users\mineh\Documents>cd PycharmProjects

C:\Users\mineh\Documents\PycharmProjects>cd assignment07_mine

C:\Users\mineh\Documents\PycharmProjects\assignment07_mine> python assignment07_minewernsing.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: 1
Enter the student's first name: Bob
Enter the student's last name: Baker
Please enter the name of the course: Python

You have registered Bob Baker for Python.
```

Figure 11. Testing the menu option 1 in Command Prompt.



```
Enter your menu choice number: 2
--------------------------------------------------
Student Jack Jordan is enrolled in Python
Student Susan James is enrolled in Python
Student Bob Baker is enrolled in Python
--------------------------------------------------


---- 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 Jack Jordan is enrolled in Python
Student Susan James is enrolled in Python
Student Bob Baker is enrolled in Python
--------------------------------------------------


---- 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

C:\Users\mineh\Documents\PycharmProjects\assignment07_mine>
```

Figure 12. Testing the menu options 2, 3, and 4 in Command Prompt.



```
Enter your menu choice number: 1
Enter the student's first name: \
One of the values was not the correct type of data!

-- Technical Error Message --
Student first name must be alphabetic.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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: 5
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, 3 or 4
```

Figure 13. Testing error handling in Command Prompt.

Source Control

The script file and the knowledge document are hosted on a GitHub repository.

A link to the repository is below:

[MineWernsing/IntroToProg-Python-Mod07 (github.com)](https://github.com/MineWernsing/IntroToProg-Python-Mod07)

A link to the repository is in the GitHub links forum below:

Topic: Assignment 07 Documents for Review! (uw.edu)

Summary

Classes provide a way to organize code by grouping functions and data needed by those functions, making the code more structured and readable, especially in larger projects with multiple programmers. In this script, I created Person and Student data classes, FileProcessor processing class and IO presentation class to create a modular structure, making it easier to manage and maintain the code. The Student class (child class) inherited code from existing Person class (parent class). The code is tested on PyCharm and Prompt Console successfully.