Name: Kornel Cieslik

Date: 11/24/24

Class: IT FDN 110 B

Github Link: https://github.com/Kornel93/IntroToProg-Python_Mod07

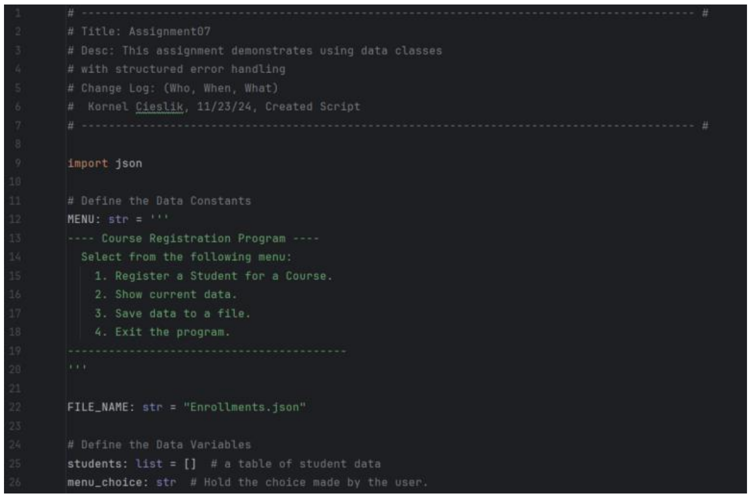
Assignment 07 – Getters and Setters

Introduction:

The subject matter of this assignment again builds on the skills we have developed so far. We continue to grow faculties in conditional logic, the use of dictionaries, error handling, classes & methods, and utilizing JSON files. The focus for this week is the utilization of getters and setters, initializing constructors, having private attributes via a subclass inheriting traits from a super class, and overwriting special methods.

Creation/Thought Process:

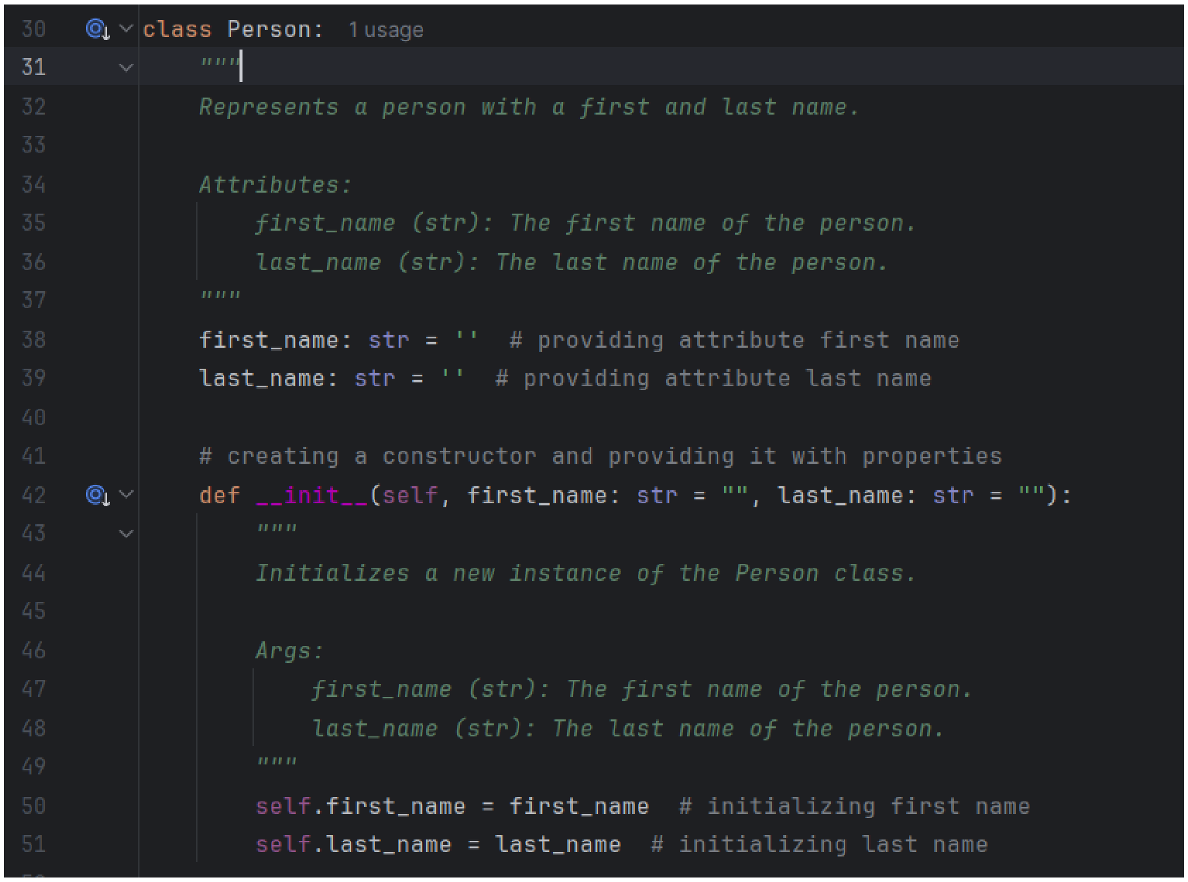
The ultimate goal of the project is the same as previous projects, creating a student registration form. The additional focus is on encapsulation to restrict direct access to an object's data while also providing validation for the data prior to assigning it to an attribute. We accomplish this through the use of getters and setters. These are methods that assist in providing controlled access to private or protected attributes. Attributes are variables that belong to either an object or a class in Python. Best practice dictates that a header is provided for a who, what, and where. **Fig 1.1** displays the declaration of variables, the establishment of the JSON enrollments file, and the menu options offered up to the user.



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#  Kornel Cieslik, 11/23/24, 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  # Hold the choice made by the user.
```

Fig 1.1 - Provision of header, menu items, and declaration of variables.

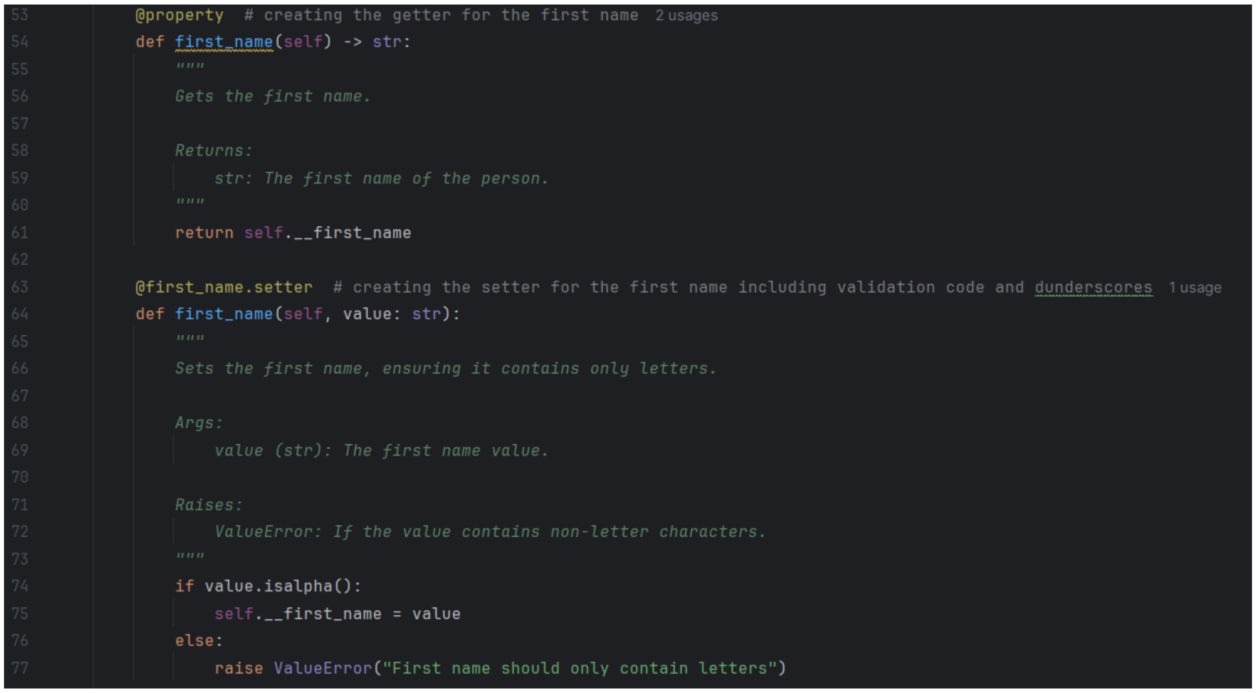
Prior to any menu choices being offered, one of the first asks of the assignment was to create a Person class. The Person class will contain the attributes first_name and last_name. These variables will are then added to a constructor that will later allow the use of getters and setters to restrict direct access to the data. The creation of the class and constructor can be seen in **Fig 1.2**. The constructor receives two attributes here, first and last name.



```
30  class Person:  1 usage
31      """
32      Represents a person with a first and last name.
33  
34      Attributes:
35          first_name (str): The first name of the person.
36          last_name (str): The last name of the person.
37      """
38      first_name: str = ''  # providing attribute first name
39      last_name: str = ''  # providing attribute last name
40  
41      # creating a constructor and providing it with properties
42      def __init__(self, first_name: str = "", last_name: str = ""):
43          """
44          Initializes a new instance of the Person class.
45  
46          Args:
47              first_name (str): The first name of the person.
48              last_name (str): The last name of the person.
49          """
50          self.first_name = first_name  # initializing first name
51          self.last_name = last_name  # initializing last name
```

Fig 1.2 – Creation of the person class, the constructor, and initializing the attributes to be used outside of the constructor.

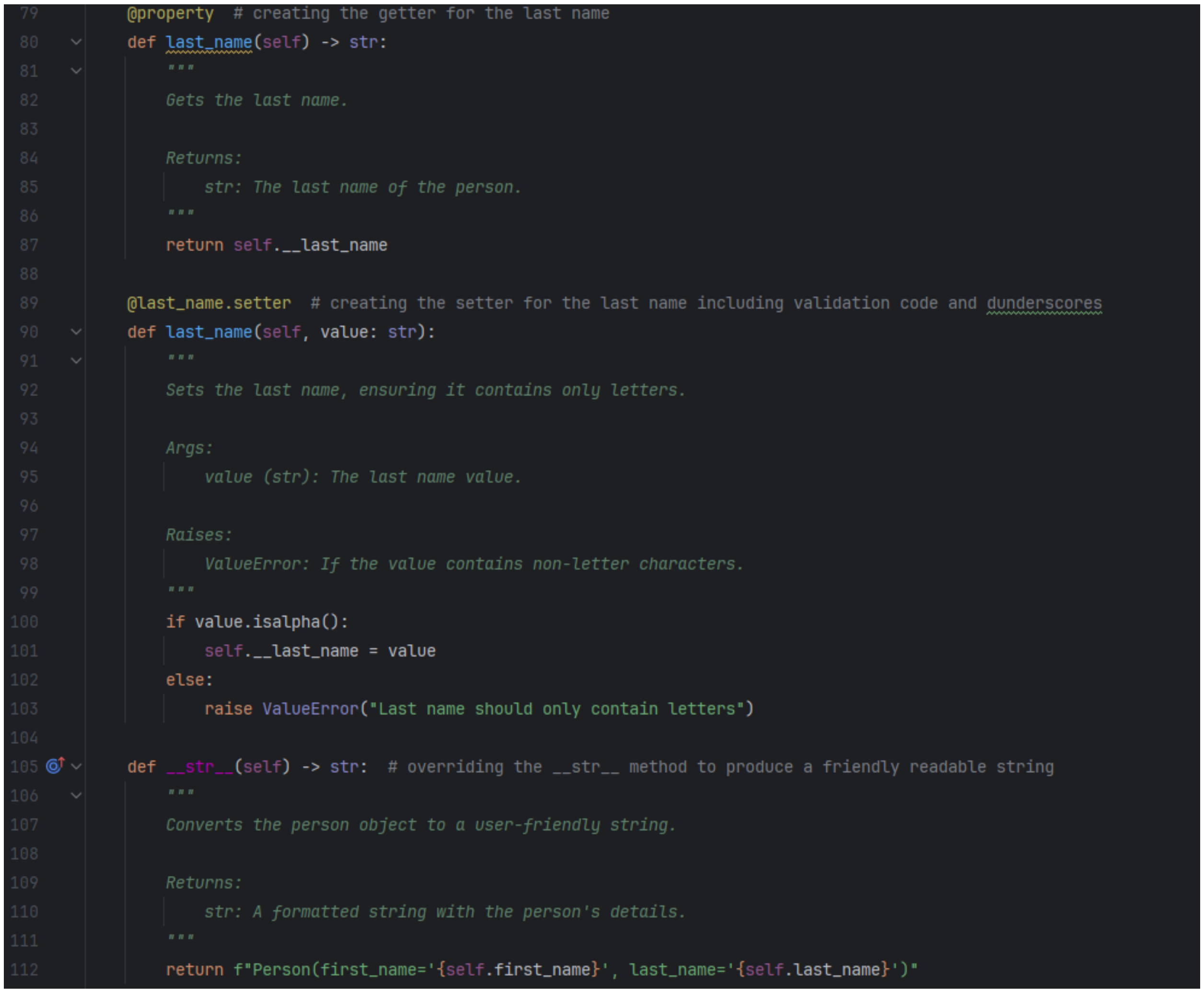
The attributes are initialized in lines 50 & 51 to allow the use of the attributes by tying them to the instance of the object. **Fig 1.3** displays the set up for the getter and setter for the first name attribute. Note that there are double underscores which make the attribute private. By being private, this essentially protects the attribute from being modified from outside of the class. Additionally, there is validation code placed within the setter method to check if the name that the user enters contains only letter characters.



```python
    @property  # creating the getter for the first name  2 usages
    def first_name(self) -> str:
        """
        Gets the first name.

        Returns:
            str: The first name of the person.
        """
        return self.__first_name

    @first_name.setter  # creating the setter for the first name including validation code and dunderscores  1 usage
    def first_name(self, value: str):
        """
        Sets the first name, ensuring it contains only letters.

        Args:
            value (str): The first name value.

        Raises:
            ValueError: If the value contains non-letter characters.
        """
        if value.isalpha():
            self.__first_name = value
        else:
            raise ValueError("First name should only contain letters")
```

Fig 1.3 – Getter amd Setter creation for first name

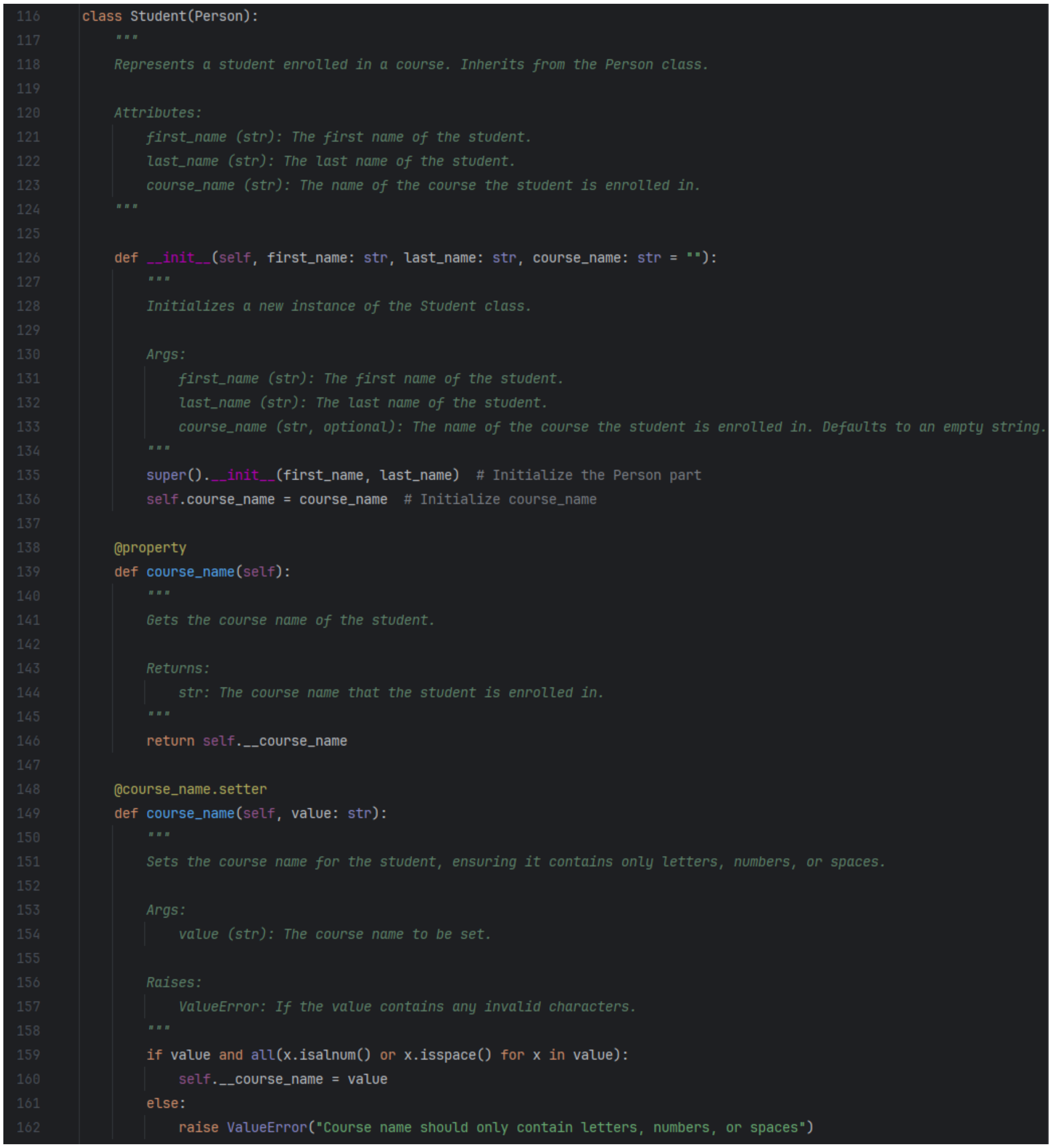
This same process is followed for the last name as can be seen in **Fig 1.4**.



```python
    @property  # creating the getter for the last name
    def last_name(self) -> str:
        """
        Gets the last name.

        Returns:
            str: The last name of the person.
        """
        return self.__last_name

    @last_name.setter  # creating the setter for the last name including validation code and dunderscores
    def last_name(self, value: str):
        """
        Sets the last name, ensuring it contains only letters.

        Args:
            value (str): The last name value.

        Raises:
            ValueError: If the value contains non-letter characters.
        """
        if value.isalpha():
            self.__last_name = value
        else:
            raise ValueError("Last name should only contain letters")

    def __str__(self) -> str:  # overriding the __str__ method to produce a friendly readable string
        """
        Converts the person object to a user-friendly string.

        Returns:
            str: A formatted string with the person's details.
        """
        return f"Person(first_name='{self.first_name}', last_name='{self.last_name}')"
```

Fig 1.4 – Getters and Setters for last name with validation code

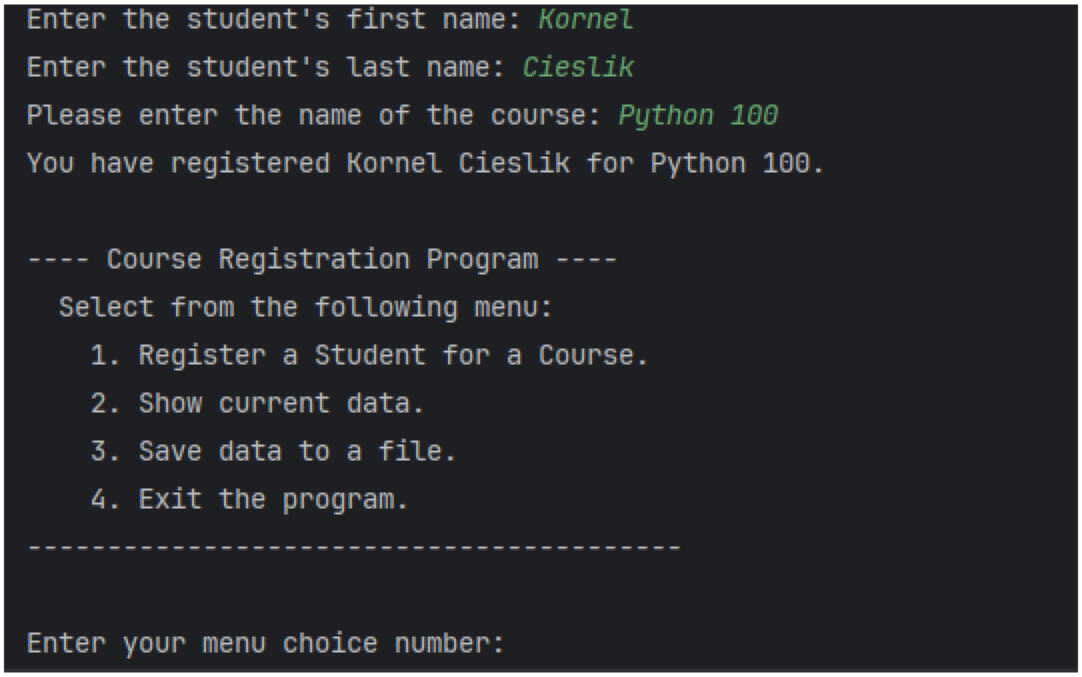
Once the getters and setters for both the first and last name were developed, our next task was to create a student class that inherits the traits from the earlier person class. We also want to add a course name for the student class as well. Much the same, the process is very similar with creating a constructor and initializing attributes. This can be seen in **Fig 1.5**



```python
class Student(Person):
    """
    Represents a student enrolled in a course. Inherits from the Person class.

    Attributes:
        first_name (str): The first name of the student.
        last_name (str): The last name of the student.
        course_name (str): The name of the course the student is enrolled in.
    """

    def __init__(self, first_name: str, last_name: str, course_name: str = ""):
        """
        Initializes a new instance of the Student class.

        Args:
            first_name (str): The first name of the student.
            last_name (str): The last name of the student.
            course_name (str, optional): The name of the course the student is enrolled in. Defaults to an empty string.
        """
        super().__init__(first_name, last_name)  # Initialize the Person part
        self.course_name = course_name  # Initialize course_name

    @property
    def course_name(self):
        """
        Gets the course name of the student.

        Returns:
            str: The course name that the student is enrolled in.
        """
        return self.__course_name

    @course_name.setter
    def course_name(self, value: str):
        """
        Sets the course name for the student, ensuring it contains only letters, numbers, or spaces.

        Args:
            value (str): The course name to be set.

        Raises:
            ValueError: If the value contains any invalid characters.
        """
        if value and all(x.isalnum() or x.isspace() for x in value):
            self.__course_name = value
        else:
            raise ValueError("Course name should only contain letters, numbers, or spaces")
```

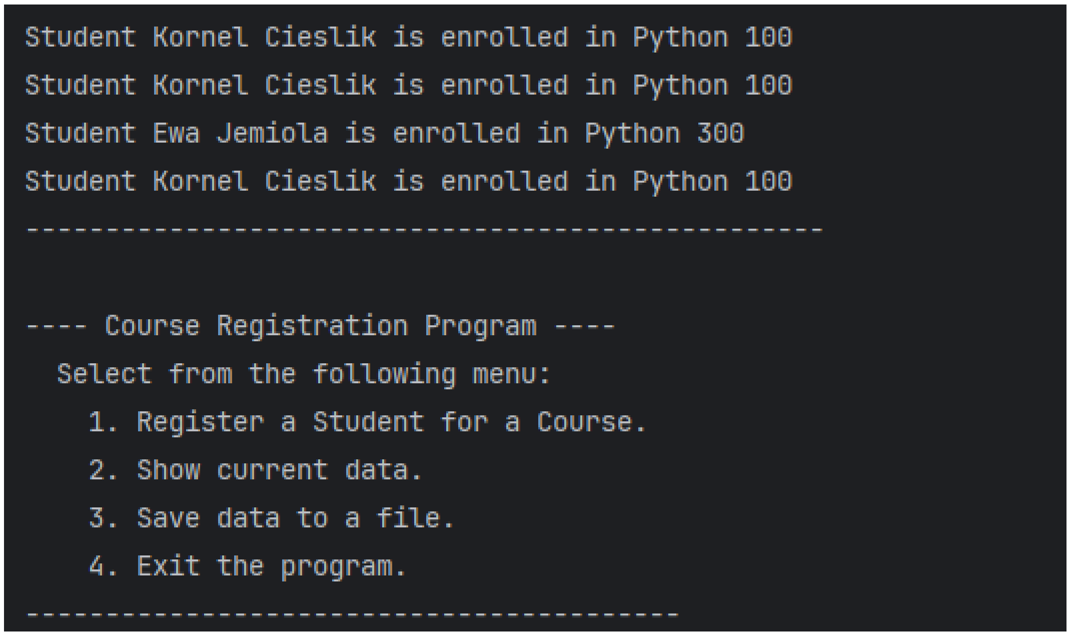
Fig 1.5 – Creation of Student class with inherited attributes with getters and setters for course name. Validation code also included to check if numbers, letters, and spaces are used only.

The FileProcessor and IO classes have been modified slightly and have been documented with doc strings within the code. The following figures below will show results of the different options and the command terminal.



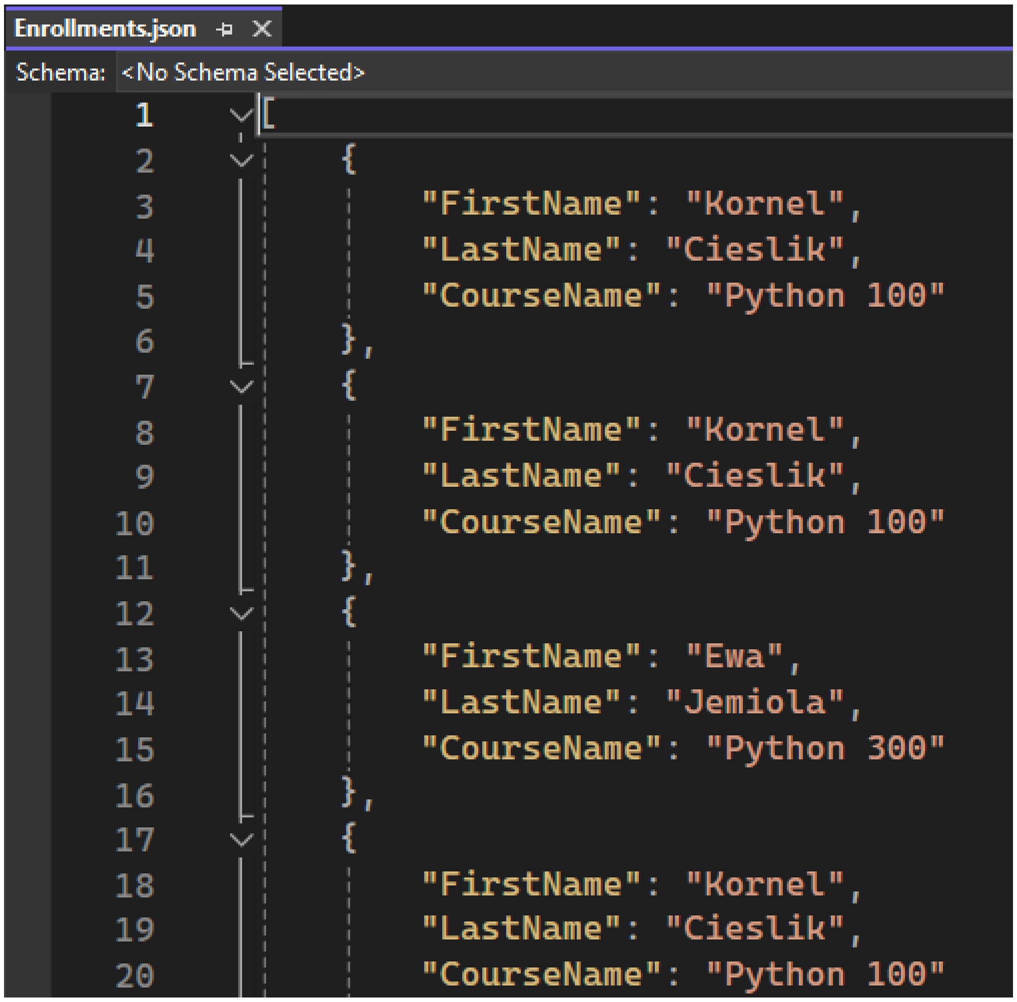
```
Enter the student's first name: Kornel
Enter the student's last name: Cieslik
Please enter the name of the course: Python 100
You have registered Kornel Cieslik for Python 100.

---- 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:
```

Fig 1.6 – Results of option 1



```
Student Kornel Cieslik is enrolled in Python 100
Student Kornel Cieslik is enrolled in Python 100
Student Ewa Jemiola is enrolled in Python 300
Student Kornel Cieslik is enrolled in Python 100
--------------------------------------------------


---- 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.
-----------------------------------------
```

Fig 1.7 – Results of option 2



```
[
    {
        "FirstName": "Kornel",
        "LastName": "Cieslik",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Kornel",
        "LastName": "Cieslik",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Ewa",
        "LastName": "Jemiola",
        "CourseName": "Python 300"
    },
    {
        "FirstName": "Kornel",
        "LastName": "Cieslik",
        "CourseName": "Python 100"
```

Fig 1.8 – Results of option 3, saved to file.

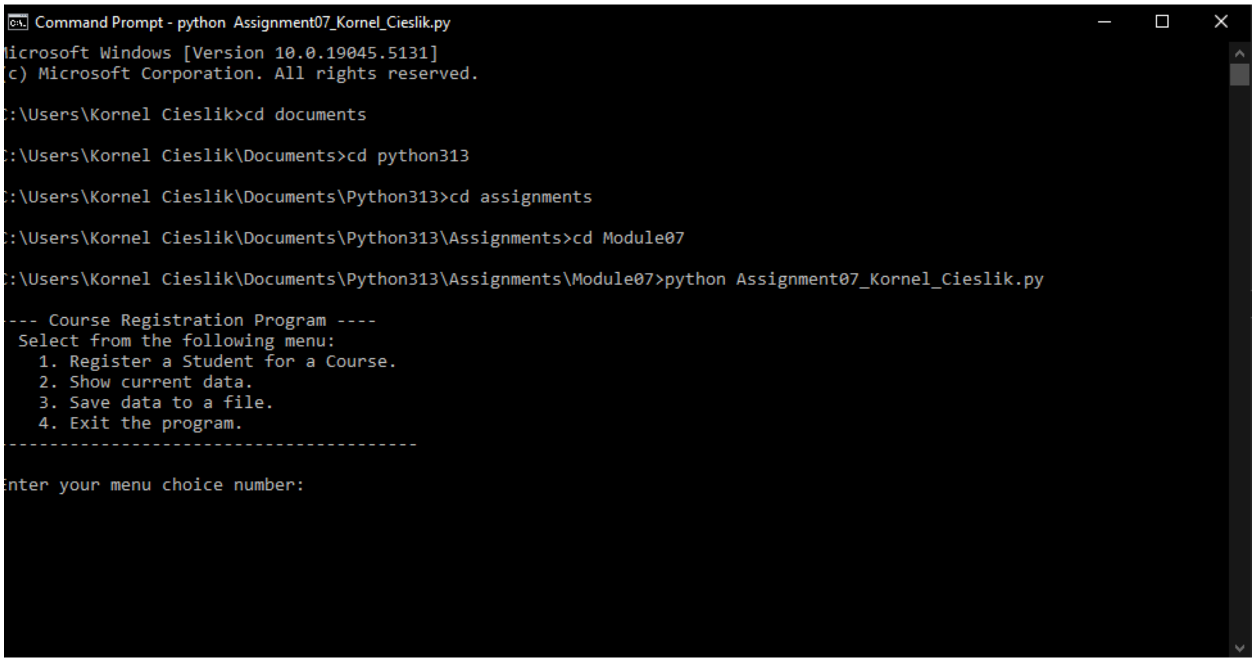

```
Command Prompt - python  Assignment07_Kornel_Cieslik.py
icrosoft Windows [Version 10.0.19045.5131]
c) Microsoft Corporation. All rights reserved.

:\Users\Kornel Cieslik>cd documents

:\Users\Kornel Cieslik\Documents>cd python313

:\Users\Kornel Cieslik\Documents\Python313>cd assignments

:\Users\Kornel Cieslik\Documents\Python313\Assignments>cd Module07

:\Users\Kornel Cieslik\Documents\Python313\Assignments\Module07>python Assignment07_Kornel_Cieslik.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.
----------------------------------------

nter your menu choice number:
```

Fig 1.9 – Code working through command terminal