Tellrell White

06/03/2025

Foundations of Programming: Python

Assignment 6

https://github.com/twhite320/IntroToProg-Python-Mod7

Classes and Functions

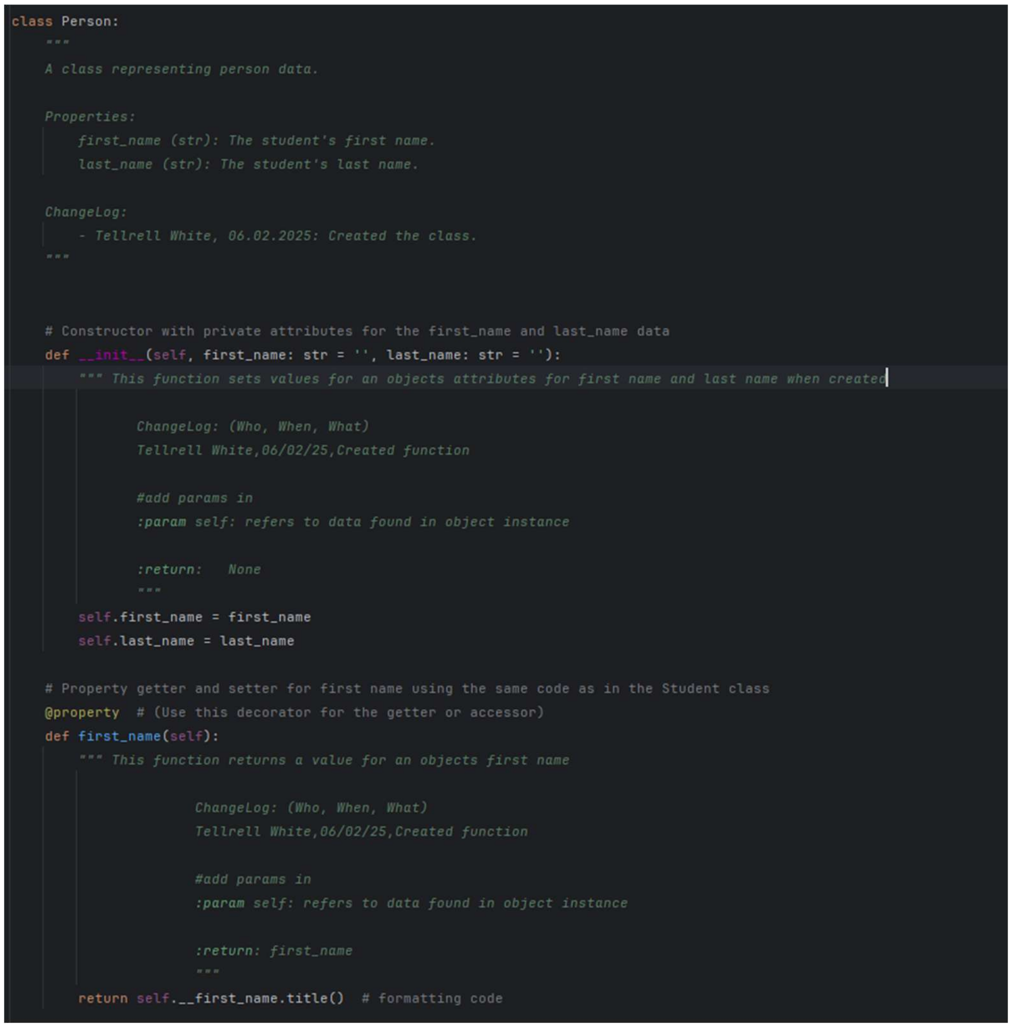
Introduction

In this document I will discuss the steps I used to create and test a Python script that displays a message about a student's registration in a Python course. This assignment is very similar to Assignment 6; however, the additional concepts of classes, inheritance, and data management were added to complete this assignment using the Pycharm Integrated Development Environment (IDE).

Development of Python Script

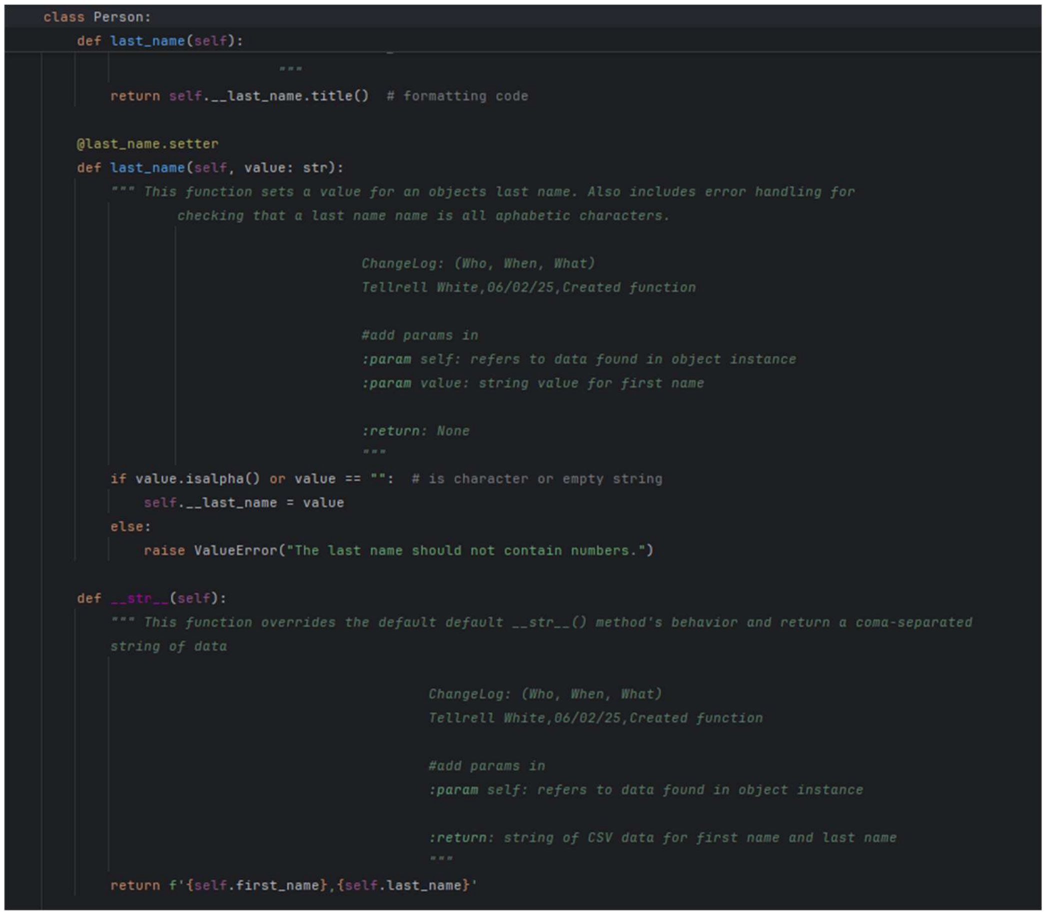
The Python script created for this assignment is very similar to the script developed in completing Assignment 6 with a few changes. One of the changes involves the creation of two data classes: A class named "Person" which functions as a parent or source class with associated variables or characteristics, and a class named "Student" which inherits or derives characteristics from the source class, Person, and also adds an additional characteristic. In order to develop these two data classes, the concepts of Attributes, Constructors, and Properties were used. Attributes are variables that hold data specific to an object which can be strings, integers or custom data types. A Constructor is a special method that is automatically called when an object of a class is created that sets initial values for the attributes of an object. Properties are functions used to manage attribute data, primarily for the use of getting and setting attribute values. The "@property decorator" indicates a property as a getter or accessor of data, while the "@name_of_property.setter" denotes a property that is used for setting data values.

The two classes, Person and Student, created in this assignment are shown below in Figures 1 and 2. The Person Class functions as a parent or source class to the "Student" class. The Person class contains a constructor (__init__), shown in Figure 1, that takes two optional parameters: first_name and last_name and leverages the properties denoted by @first_name.setter and @last_name.setter in Figure for setting the values of the first and last name for newly created objects and also for validating that the first and last name only contain alphabetic characters. The __str__ method shown in Figure 2 for the class prints out CSV string data for the first name and last name of an object. The Student class is a subclass of the Person class, and as such inherits or utilizes all the attributes and properties of the Person class allowing for code re-usability, and adds an additional attribute “course_name”. Figure 3 shows the constructor for Student which uses the super().__init__ to call the constructor of the parent Person class, which is used for setting and getting the values for first_name and last_name. The class also utilizes getter and setter functions for the additional attribute “course_name”. Figure 4 displays the “setter function” used for setting the course_name attribute for an object. Additionally, the Person class __str__ is extended in the Student class to include a value for “course_name” for each object of the Student class which is also shown in Figure 4.



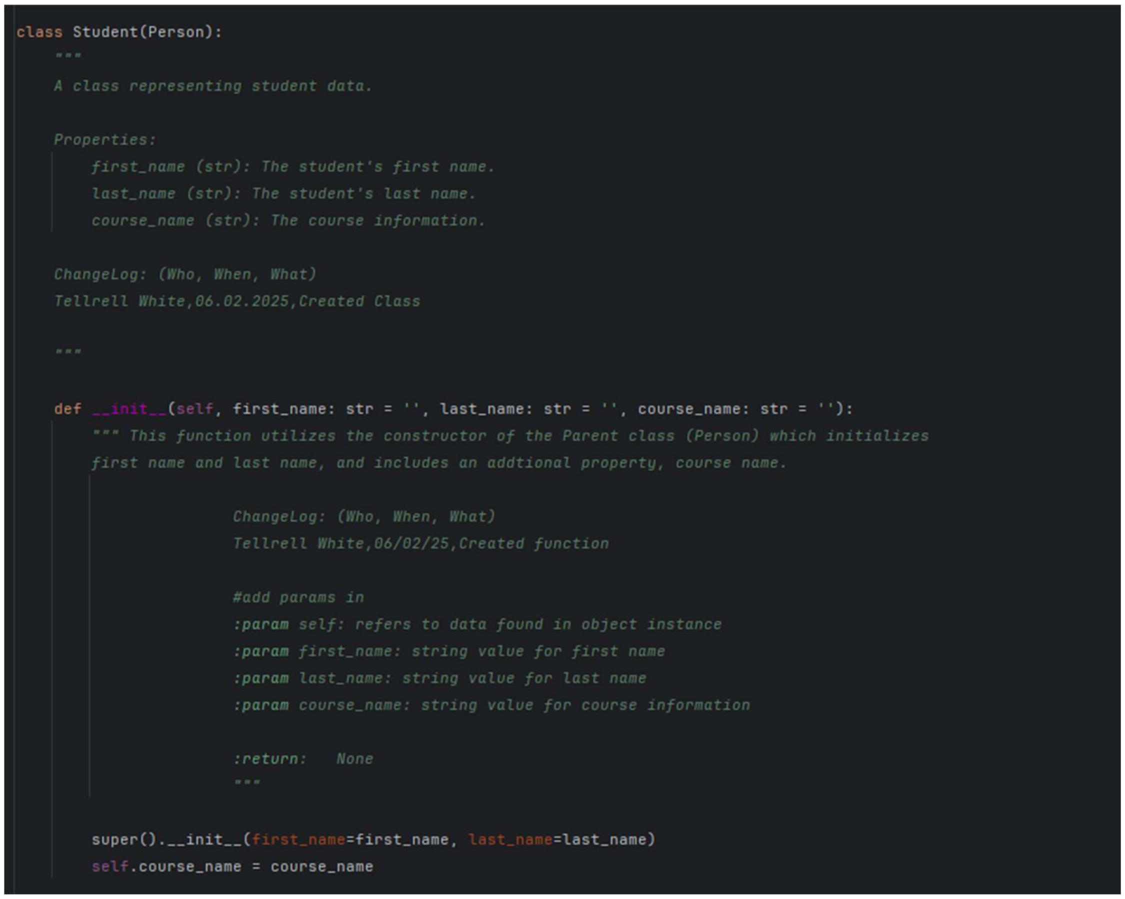
```python
class Person:
    """
    A class representing person data.

    Properties:
        first_name (str): The student's first name.
        last_name (str): The student's last name.

    ChangeLog:
        - Tellrell White, 06.02.2025: Created the class.
    """


    # Constructor with private attributes for the first_name and last_name data
    def __init__(self, first_name: str = '', last_name: str = ''):
        """ This function sets values for an objects attributes for first name and last name when created

               ChangeLog: (Who, When, What)
               Tellrell White,06/02/25,Created function

               #add params in
               :param self: refers to data found in object instance

               :return:   None
               """
        self.first_name = first_name
        self.last_name = last_name

    # Property getter and setter for first name using the same code as in the Student class
    @property  # (Use this decorator for the getter or accessor)
    def first_name(self):
        """ This function returns a value for an objects first name

                      ChangeLog: (Who, When, What)
                      Tellrell White,06/02/25,Created function

                      #add params in
                      :param self: refers to data found in object instance

                      :return: first_name
                      """
        return self.__first_name.title()  # formatting code
```

Figure 1: Python Script showing constructor for Person Class



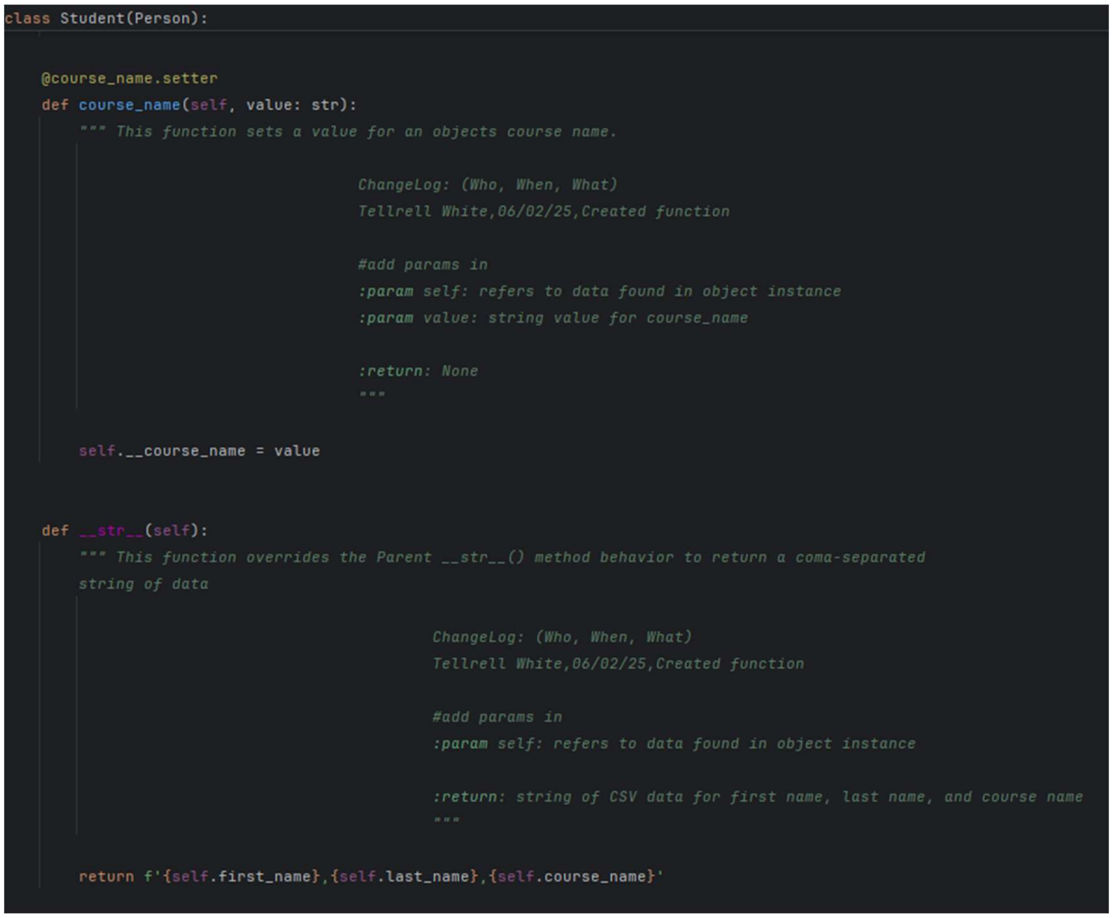
```
class Person:
    def last_name(self):
                     """
        return self.__last_name.title()  # formatting code

    @last_name.setter
    def last_name(self, value: str):
        """ This function sets a value for an objects last name. Also includes error handling for
                checking that a last name name is all aphabetic characters.

                                ChangeLog: (Who, When, What)
                                Tellrell White,06/02/25,Created function

                                #add params in
                                :param self: refers to data found in object instance
                                :param value: string value for first name

                                :return: None
                                """
        if value.isalpha() or value == "":  # is character or empty string
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers.")

    def __str__(self):
        """ This function overrides the default default __str__() method's behavior and return a coma-separated
        string of data

                                        ChangeLog: (Who, When, What)
                                        Tellrell White,06/02/25,Created function

                                        #add params in
                                        :param self: refers to data found in object instance

                                        :return: string of CSV data for first name and last name
                                        """
        return f'{self.first_name},{self.last_name}'
```

Figure 2: Python Script showing setter property for last_name attribute and __string__ () method



```
class Student(Person):
    """
    A class representing student data.

    Properties:
        first_name (str): The student's first name.
        last_name (str): The student's last name.
        course_name (str): The course information.

    ChangeLog: (Who, When, What)
    Tellrell White,06.02.2025,Created Class

    """

    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        """ This function utilizes the constructor of the Parent class (Person) which initializes
        first name and last name, and includes an addtional property, course name.

                       ChangeLog: (Who, When, What)
                       Tellrell White,06/02/25,Created function

                       #add params in
                       :param self: refers to data found in object instance
                       :param first_name: string value for first name
                       :param last_name: string value for last name
                       :param course_name: string value for course information

                       :return:   None
                       """

        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name
```

Figure 3:Python Script showing constructor for Student Class



```python
class Student(Person):

    @course_name.setter
    def course_name(self, value: str):
        """ This function sets a value for an objects course name.

                                ChangeLog: (Who, When, What)
                                Tellrell White,06/02/25,Created function

                                #add params in
                                :param self: refers to data found in object instance
                                :param value: string value for course_name

                                :return: None
                                """

        self.__course_name = value


    def __str__(self):
        """ This function overrides the Parent __str__() method behavior to return a coma-separated
        string of data

                                        ChangeLog: (Who, When, What)
                                        Tellrell White,06/02/25,Created function

                                        #add params in
                                        :param self: refers to data found in object instance

                                        :return: string of CSV data for first name, last name, and course name
                                        """

        return f'{self.first_name},{self.last_name},{self.course_name}'
```

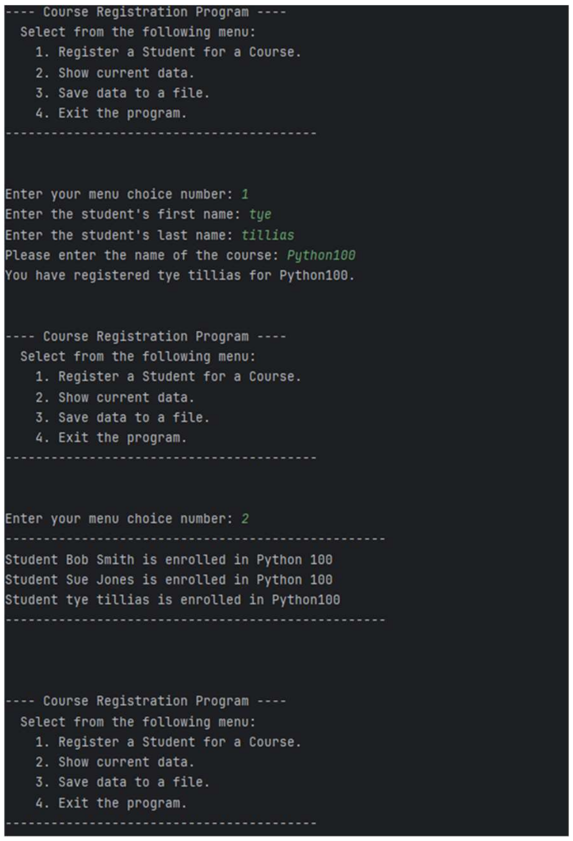
Figure 4: Python Script showing setter property for course_name attribute and __str__() method

Testing of Python Script

To validate that the Python script developed for this assignment worked as intended, two methods were used to test the script. Within the Pycharm IDE, I ran the "Assignment07.py script to not only ensure that there weren't any errors, but also to verify the behavior of the script was correct. Figures 5, 6, and 7 show the output from running the "Assignment07.py" script in Pycharm. As shown in the figures, the user inputs a value matching a menu choice, and the conditional statement matching the user's input is executed. As shown in Figure 5, the user first selects "menu choice 1", which prompts the user for a first name, last name and course information. During the second loop or iteration, the user is shown the menu again, and the user selects "option 2", which displays the registration information contained within the list "students".

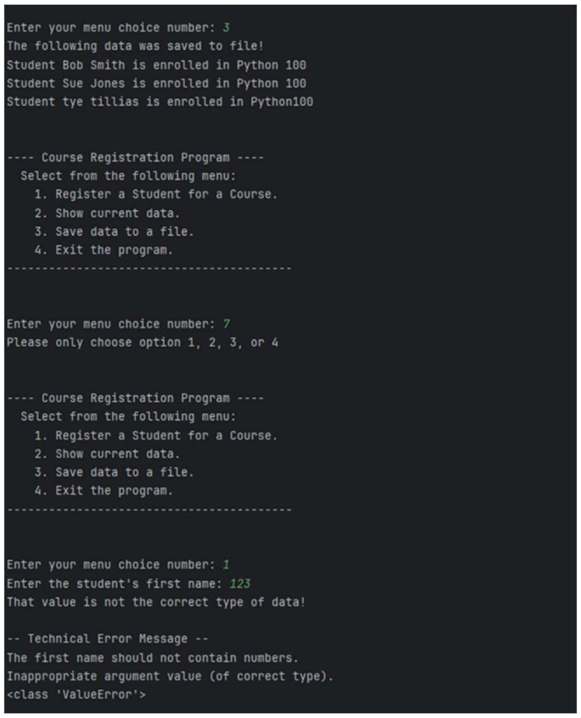
Next, the user selects "option 3", which saves the user's registration information to a JSON file and prints the student registration information written to the screen. The contents of the original JSON file, “Enrollments.json” is shown in Figure 8 and the modified JSON file containing all the data written to the file from the “students” variable in shown in Figure 9. During the next loop the menu is presented, and the user enters a value of “7” shown in Figure 6, which isn’t a valid choice, and the user is shown the message “Enter a valid value (1-4)” and the program restarts the loop. Next, the user selects menu choice 1 and inputs “123” when prompted for a first name. Eror handling is demonstrated by printing a message telling the user that the first name cannot contain numbers along with additional information. Next, when prompted for a menu selection, the user enters the value of “4” and this ends the program.

Another method of testing the script was through using the command terminal in windows. After launching the command terminal and navigating to the correct directory where the script exists, I launched the python interpreter using the “Python” command and provided the name of the script as an argument to run it. The output of running the script is shown in Figures 10 and 11. Like the results obtained from Pycharm, when selecting menu choices 1 through 4, the program displayed correct results.



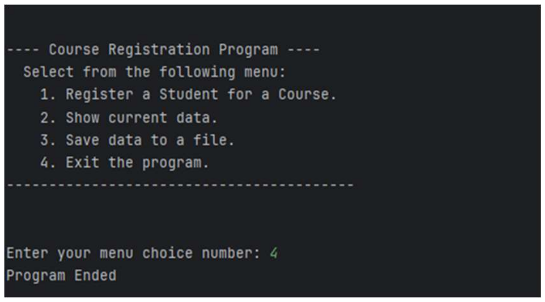
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: tye
Enter the student's last name: tillias
Please enter the name of the course: Python100
You have registered tye tillias for Python100.


---- 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 tye tillias is enrolled in Python100
--------------------------------------------------



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

Figure 5: Output from Pycharm displaying output for selection of menu choices 1 and 2



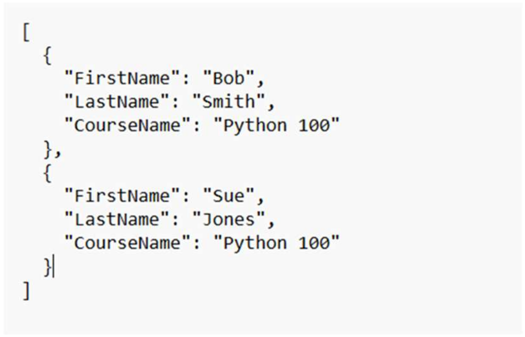
```
Enter your menu choice number: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student tye tillias is enrolled in Python100


---- 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: 7
Please only choose option 1, 2, 3, or 4


---- 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: 123
That value is not the correct type of data!

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

Figure 6: Output from Pycharm showing Output for menu selection 3 and an " invalid menu selection"



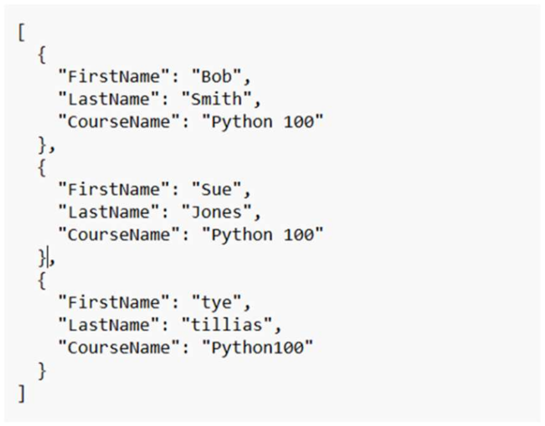
```

---- 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
```

Figure 7: Output from Pycharm showing selection of menu choice "4" to End Program



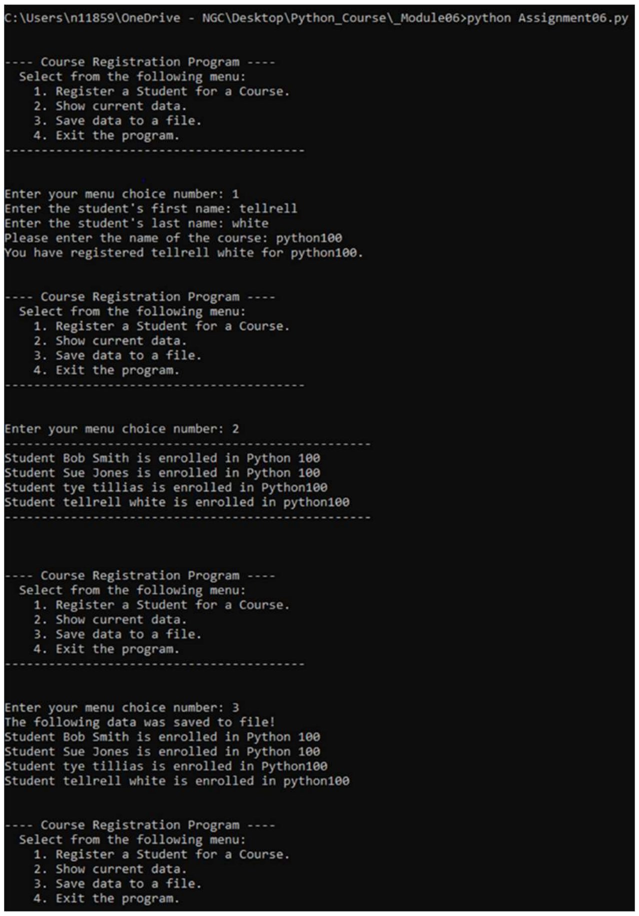
```json
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  }
]
```

Figure 8: Original JSON File



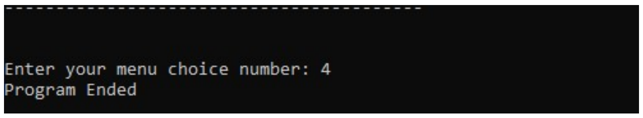
```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "tye",
    "LastName": "tillias",
    "CourseName": "Python100"
  }
]
```

Figure 9: Modified JSON File with student Added



```
C:\Users\n11859\OneDrive - NGC\Desktop\Python_Course\_Module06>python Assignment06.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: tellrell
Enter the student's last name: white
Please enter the name of the course: python100
You have registered tellrell white for python100.


---- 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 tye tillias is enrolled in Python100
Student tellrell white is enrolled in python100
--------------------------------------------------



---- 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 following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student tye tillias is enrolled in Python100
Student tellrell white is enrolled in python100


---- 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.
```

Figure 10: Windows Terminal Output for Menu Selections 1- 3



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

Enter your menu choice number: 4
Program Ended
```

Figure 11: Windows Terminal Output for Menu Choice 4 to "End Program"

Summary

As discussed in this document, a Python script was developed utilizing the concepts of functions, classes and SoC learned in module 6 to display to a user information about registration into a Python course. This script was tested using Pycharm, the terminal in windows, and examination of a JSON file for correct behavior.