Deborah Briggs

May 22, 2025

Foundations of Programming: Python

Assignment 07

https://github.com/jdbriggs3/IntroToProg-Python-Mod07

Classes and Objects

Introduction

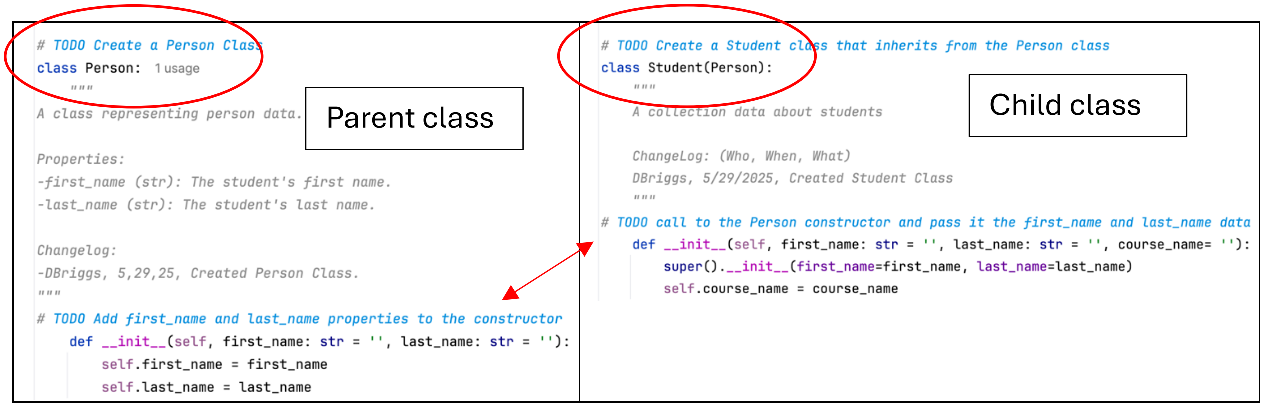
This assignment built upon our last assignment. Although, the program appears the same to the user changes were made within the code with the addition of two data classes, a parent class and child class. Beginning with the Assignment07-starter.py, I followed an iterative process completing each of the TODO statements. By completing each step I was able to successfully finish the assignment.

Data Classes

In our last assignment we added two classes. A processor class, FileProcessor, to read from and write to a file, and a presentation class, IO, to manage input and output of the data. This week we added two classes to manage the data, a Person class and a Student class. The Student class is a sub class, or child class of the Person class, thus can inherit properties, functions, and data from its parent, the Person class. (Figure 1)

Inheritance

In this program the Person class initializes two attributes, first and last name. The Student class inherits both and adds another, course name. (Figure 1)



```
# TODO Create a Person Class
class Person:  1 usage
    """
A class representing person data.

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

Changelog:
-DBriggs, 5,29,25, Created Person Class.
"""
# TODO Add first_name and last_name properties to the constructor
    def __init__(self, first_name: str = '', last_name: str = ''):
        self.first_name = first_name
        self.last_name = last_name
```

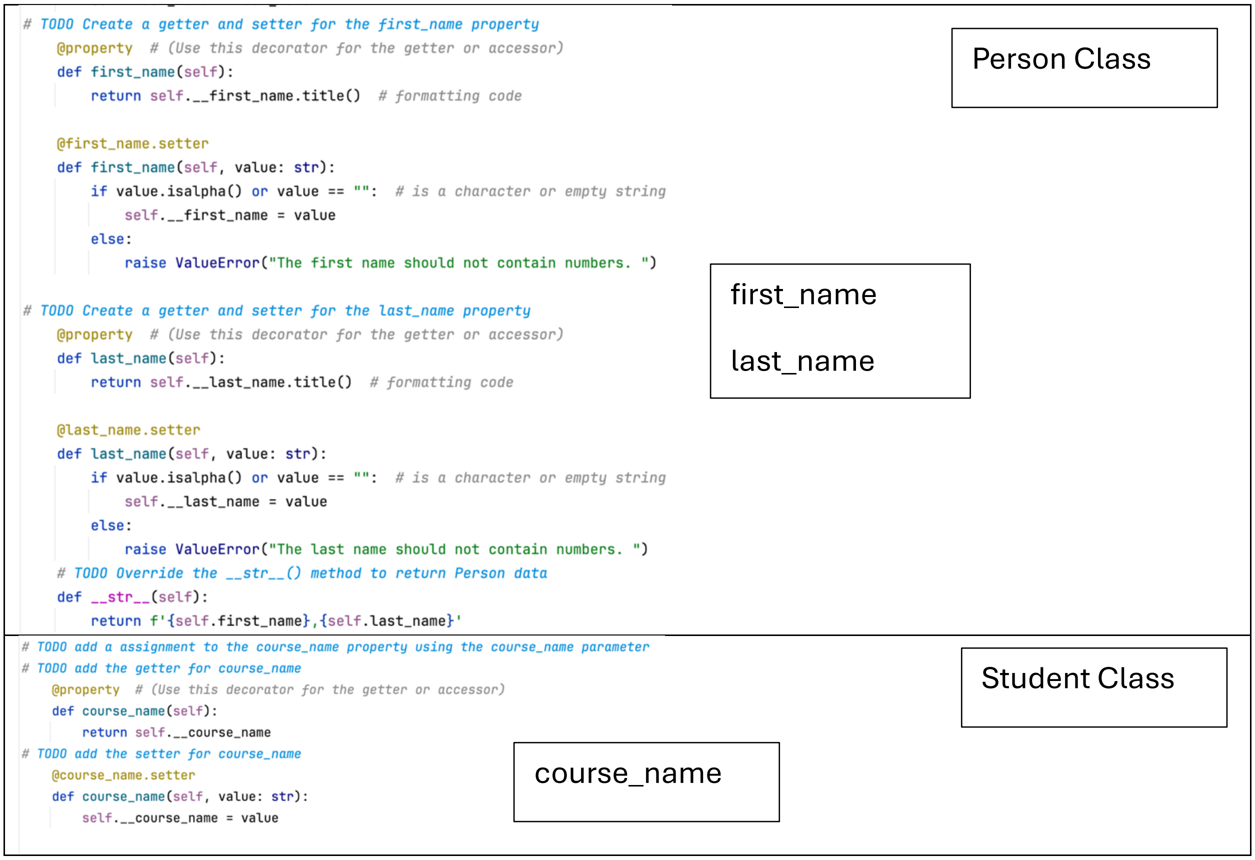
```
# TODO Create a Student class that inherits from the Person class
class Student(Person):
    """
    A collection data about students

    ChangeLog: (Who, When, What)
    DBriggs, 5/29/2025, Created Student Class
    """
# TODO 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= ''):
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name
```

Figure:1 Parent Class and Child/Sub Class

Getter and Setter

The Person and Student classes contain getter and setter properties to access data and manage error handling. (Figure 2)



```
# TODO Create a getter and setter for the first_name property
    @property  # (Use this decorator for the getter or accessor)
    def first_name(self):
        return self.__first_name.title()  # formatting code

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":  # is a character or empty string
            self.__first_name = value
        else:
            raise ValueError("The first name should not contain numbers. ")

# TODO Create a getter and setter for the last_name property
    @property  # (Use this decorator for the getter or accessor)
    def last_name(self):
        return self.__last_name.title()  # formatting code

    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":  # is a character or empty string
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers. ")
    # TODO Override the __str__() method to return Person data
    def __str__(self):
        return f'{self.first_name},{self.last_name}'
```

```
# TODO add a assignment to the course_name property using the course_name parameter
# TODO add the getter for course_name
    @property  # (Use this decorator for the getter or accessor)
    def course_name(self):
        return self.__course_name
# TODO add the setter for course_name
    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value
```

Figure 2: Getter and Setter properties

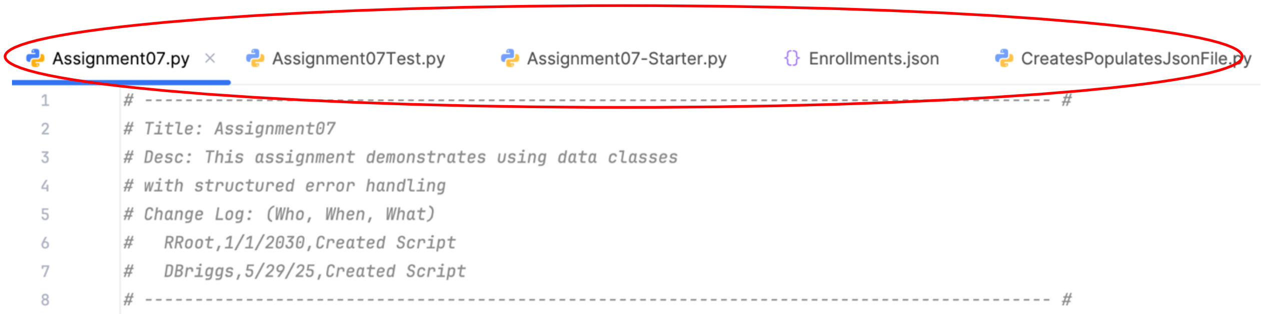
Process

As the assignments have become progressively longer and more detailed, I've found it helpful to maintain the integrity of my code by working with multiple files in PyCharm. I begin by opening the Assignment07-starter.py file and copy its contents into two new files: Assignment07.py and Assignment07Test.py. I use the test file to write and test sections of the code. Once a section is working properly, I copy it into Assignment07.py. I repeat this process iteratively until the code is complete. This method ensures I always have at least one working version of the code-actually two in this case: Assignment07.py, and Assignment07-starter.py (Figure 3)

In this assignment, we were given TODO statements throughout the code. By referencing the Mod07-Lab03 code, I was able to complete each TODO statement step by step, periodically testing for accuracy.

I've recently taken up the hobby of knitting and have noticed parallels between debugging code and fixing my knitting mistakes-especially as a beginner. In both activities, mistakes can be frustrating, but they can also be great learning opportunities. In knitting one technique I've found helpful is using a lifeline-a piece of waste yarn periodically woven into my work. This is especially useful when knitting complex lace patterns or colorwork. If I make a mistake, I can unravel the knitting just down to the lifeline rather than unraveling the whole project and starting completely over. This saves hours of work.

Similarly, in my coding process, maintaining two .py files—one for testing and one to hold corrected code - acts as my programming lifeline. It allows me to add new code and test it, knowing I can always return to a stable version if something goes wrong.



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   DBriggs,5/29/25,Created Script
# ------------------------------------------------------------------------------------------ #
```

Figure 3: Utilizing multiple files

GitHub

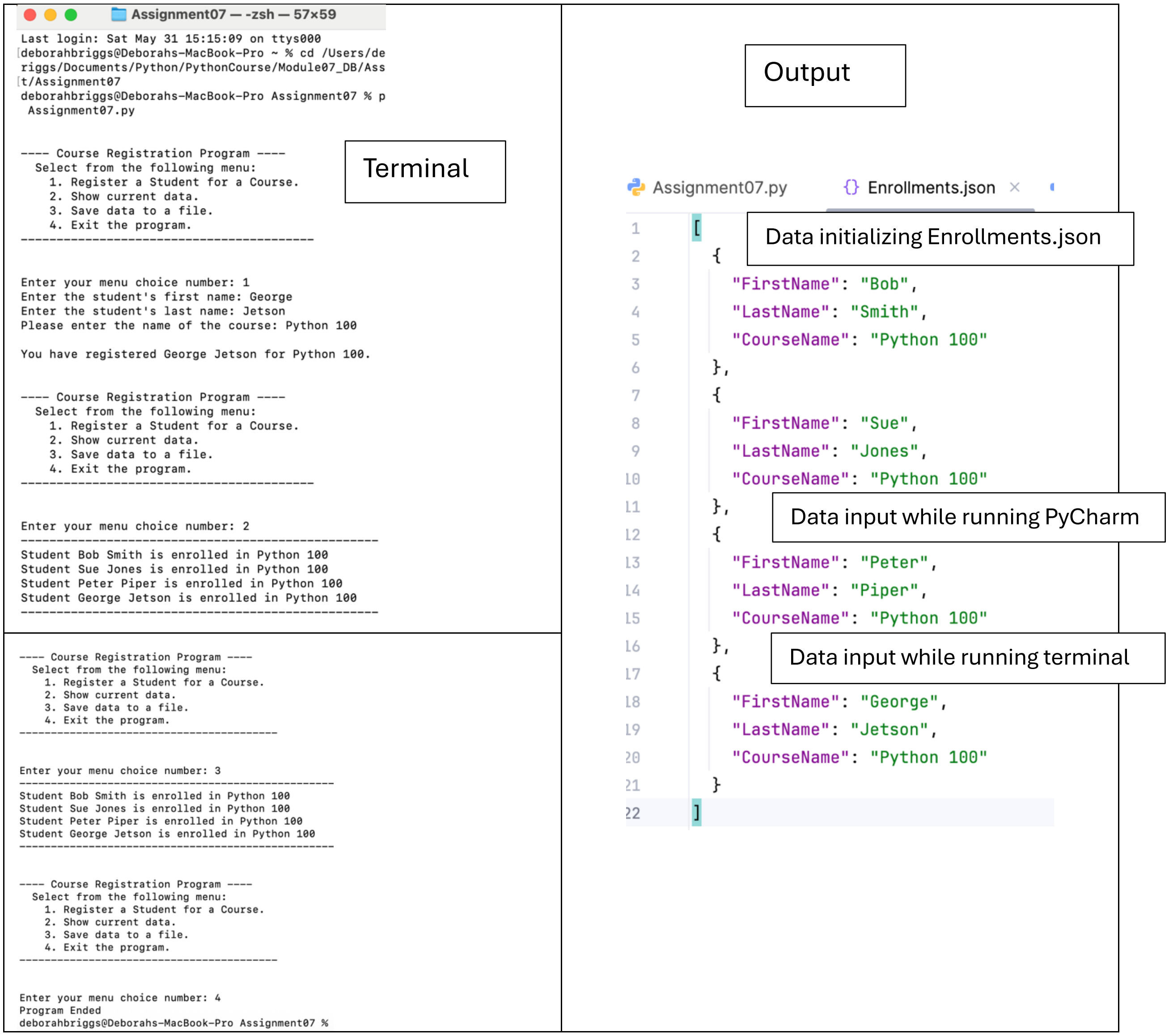
And it seems the Git feature works as a lifeline saving successive versions of the code.

Testing

I tested the file in both PyCharm and the terminal in the same manner as the last assignment with success.

Output

Below is a screen shot of the program run in the terminal environment and the final enrollments.json file. (Figure 4)



Terminal

```
Assignment07 — -zsh — 57×59
Last login: Sat May 31 15:15:09 on ttys000
deborahbriggs@Deborahs-MacBook-Pro ~ % cd /Users/de
riggs/Documents/Python/PythonCourse/Module07_DB/Ass
t/Assignment07
deborahbriggs@Deborahs-MacBook-Pro Assignment07 % p
 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: 1
Enter the student's first name: George
Enter the student's last name: Jetson
Please enter the name of the course: Python 100

You have registered George Jetson 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: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Peter Piper is enrolled in Python 100
Student George Jetson 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.
-----------------------------------------


Enter your menu choice number: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Peter Piper is enrolled in Python 100
Student George Jetson 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.
-----------------------------------------


Enter your menu choice number: 4
Program Ended
deborahbriggs@Deborahs-MacBook-Pro Assignment07 %
```

Output

Data initializing Enrollments.json

Data input while running PyCharm

Data input while running terminal

```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Peter",
    "LastName": "Piper",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "George",
    "LastName": "Jetson",
    "CourseName": "Python 100"
  }
]
```

Figure 4: Terminal and output

Summary

This assignment built upon our last assignment. The program's functionality appears the same to the user, the changes occurred within the structure of the code with the addition of two data classes, a parent class and child class. Beginning with the Assignment07-starter.py I followed an iterative approach completing each TODO statement, testing along the way. This method allowed me to maintain clean working code and successfully complete the assignment.