Noah Jon

December 2nd, 2024

IT FDN 100 A

Assignment 07

https://github.com/noahjon247/IntroToProg-Python-Mod07

Assignment 07: Classes and Objects

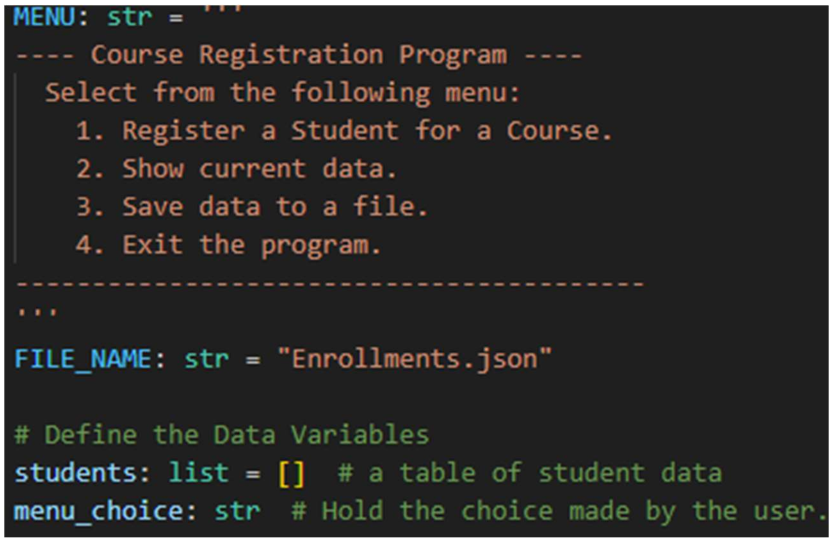
Introduction

This assignment's focus was on expanding on what we had done previously in Assignment 06 and adding extra features, including functions, classes, and using the separation of concerns pattern.

Features in the Code

Constants and Variables

The constants and variables used in this assignment were very simple, even less than what we had last week:



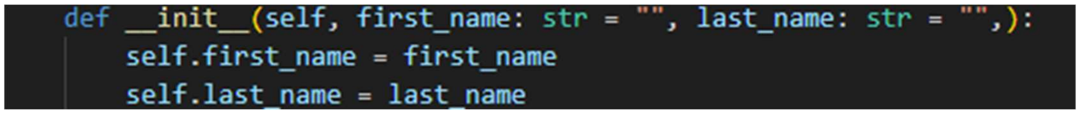
```
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```

Functions and Classes

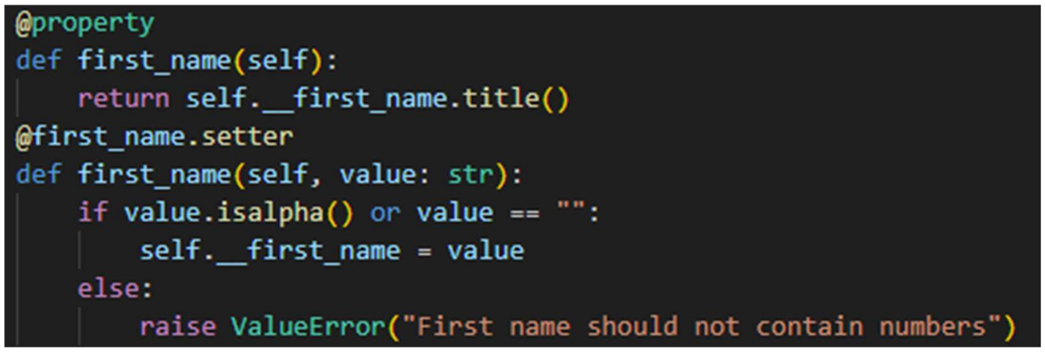
In this assignment, we were tasked with adding two new classes to our code, Person and Student(Person). Previously, we had used the classes FileProcessor and IO, which were focused on processing and presenting the data, respectively. These new classes are focused on organizing data about a certain student, allowing the data to be processed and presented by the other two classes.

In the Person class, I first added the first and last name properties to the constructor:

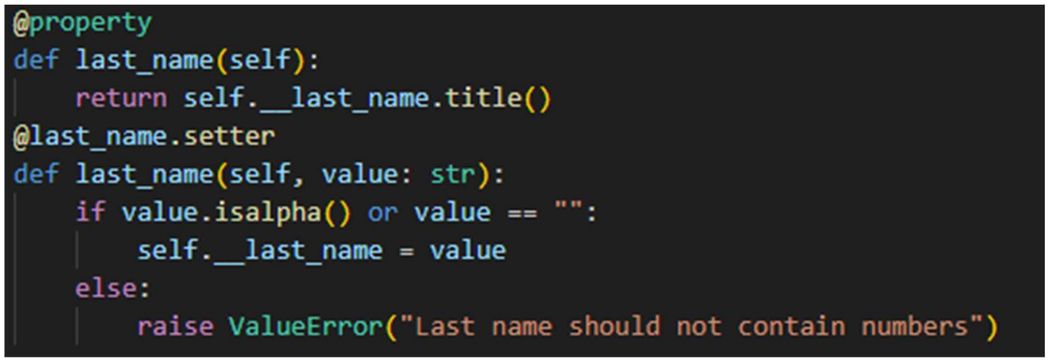


```python
    def __init__(self, first_name: str = "", last_name: str = "",):
        self.first_name = first_name
        self.last_name = last_name
```

After that, I added a “getter” and a “setter” for the first name and last name properties:

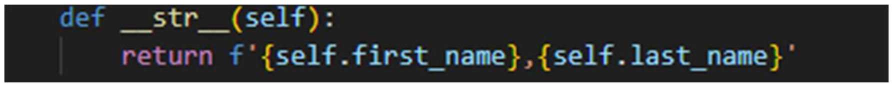


```python
@property
def first_name(self):
    return self.__first_name.title()
@first_name.setter
def first_name(self, value: str):
    if value.isalpha() or value == "":
        self.__first_name = value
    else:
        raise ValueError("First name should not contain numbers")
```



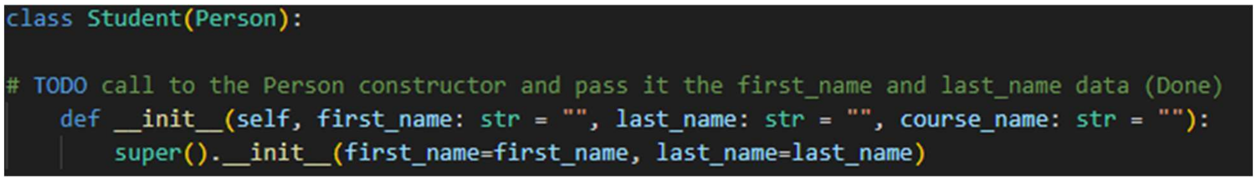
```python
@property
def last_name(self):
    return self.__last_name.title()
@last_name.setter
def last_name(self, value: str):
    if value.isalpha() or value == "":
        self.__last_name = value
    else:
        raise ValueError("Last name should not contain numbers")
```

Then I overrid the __str__() method to return the Person’s data:



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

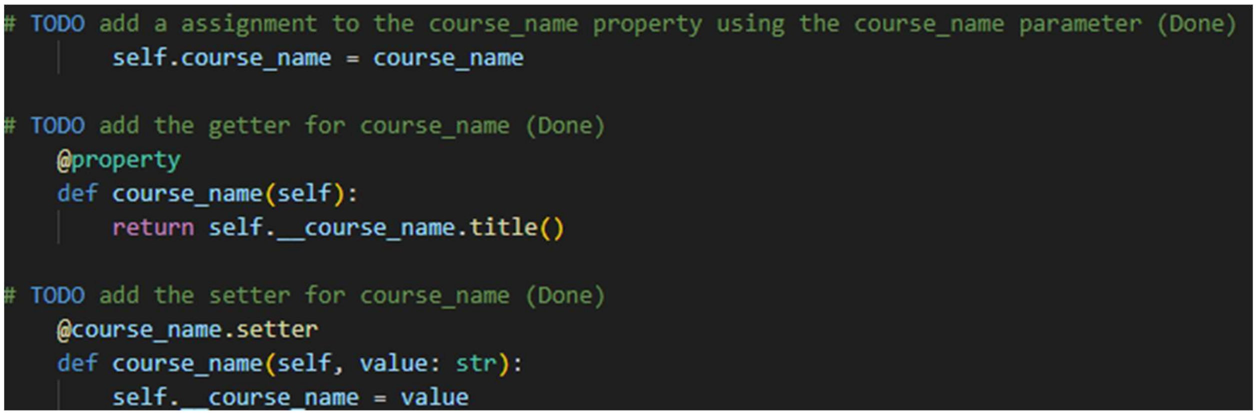
For the Student(Person) class, I first called the Person constructor and passed it to the first and last name data, allowing us to call the pervious definitions of first and last name in this class:



```python
class Student(Person):

# TODO call to the Person constructor and pass it the first_name and last_name data (Done)
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name=first_name, last_name=last_name)
```

Then I added the course_name assignment to the property and added a "getter" and a "setter" for it:

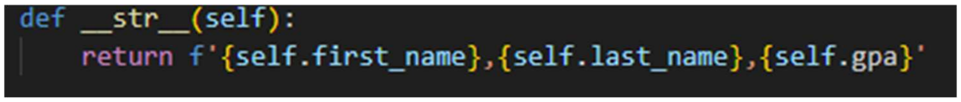


```
# TODO add a assignment to the course_name property using the course_name parameter (Done)
        self.course_name = course_name

# 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
```

Finally, I overrid the __str__() method to return the Student's data.



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

The rest of the code was identical to the previously completed assignment.

Conclusion

This assignment helped us understand the importance of organizing the data through various classes and functions. Although we had previously learned to use classes and functions for processing and presenting data in a more efficient manner, this allows us to organize the data that we are processing and presenting better as well. For the data class specifically, we learned to use constructors, attributes, and properties to allow us to maintain and create flexible code.