Matthew Lizada
11/23/2025
IT FDN 110 A
https://github.com/lizadamf/IntroToProg-Python-Mod07

# Assignment07 – Classes and Objects

## Introduction

Assignment 07 introduced object-oriented programming (OOP) representing a major shift from the procedural programming techniques introduced. Most of the work thus far has concentrated on fundamental statements, variables, and functions. Taking those ideas a step further, this module showed how classes can group data and actions together, so it is easier to manage, explaining key concepts like constructors, attributes, properties, and inheritance.

## Writing the Script

For our last assignment, we needed to create both a FileProcessor class and an IO class. This assignment took things further and included a Person and Student class along with the main programming loop. Since we learned what the main FileProcessor and IO Class does and how the main program loop operates in a previous assignment, I will only include a summary of its functions.

### FileProcessor

The FileProcessor Class managed file operations consisting of two main functions:

- read_data_from_file()
- write_data_to_file()

The main purpose of this class was to move data in and out of Enrollments.json

### IO Class (Input/Output)

The IO Class is responsible for only interacting with the user and divided into the following functions:

- output_error_messages()
- output_menu()
- input_menu_choice()

- output_student_and_course_names()
- input_student_data()

The IO class acts as a presentation layer of the program handling everything the user sees and/or inputs.

### Main Programming Loop

The main programming loop acts as the control layer, using the IO, FileProcessor, Person and Student classes to manage the program flow. It decided what to do based on the menu choices:

1. Registering a student (calls IO.input_student_data)
2. Display data (calls IO.output_student_courses)
3. Save the data and write to JSON file (calls FileProcessor.write_data_to_file)
4. Exit the program

## New Classes (Person/Student)

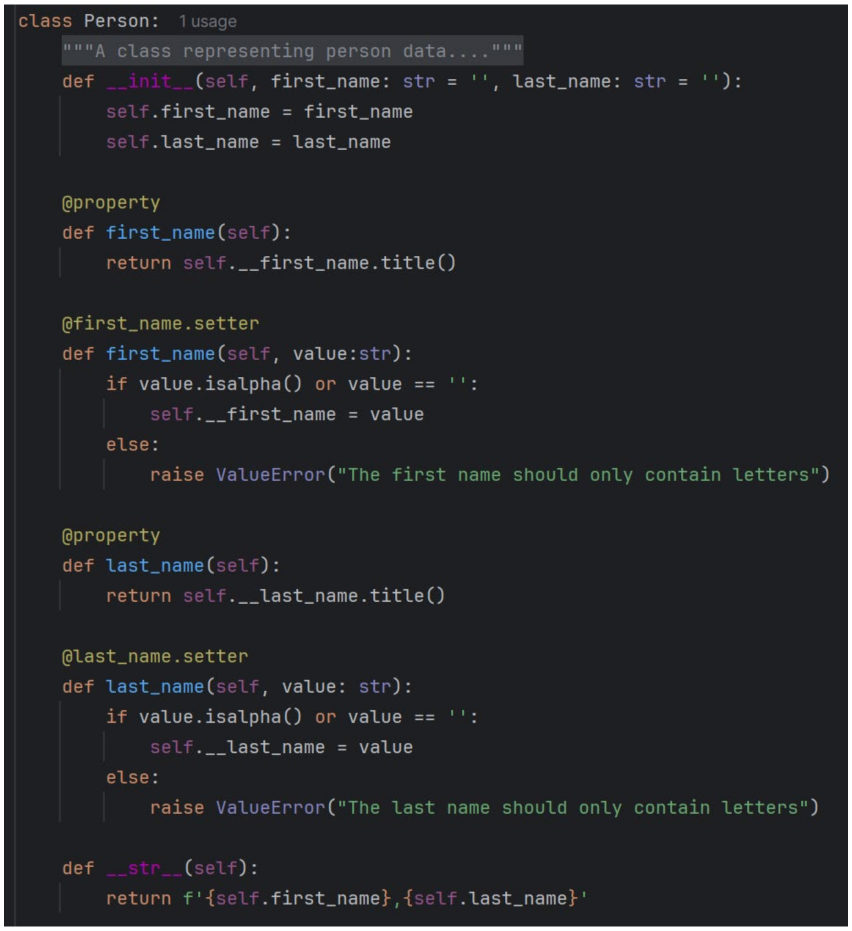
In addition to the FileProcessor and IO classes, and main programming loop. The Person and Student classes were necessary in this assignment.

### Person

The Person class is the base class that stores basic information about a person: their first name and last name.

It included a constructor (__init__) that sets the first and last name, properties that validate the names to ensure it contained only letters, automatic formatting, and a __str__() method returning the person's information as a comma-separated string.

This class provides the shared behavior that the Student class builds on (Figure 1.1)



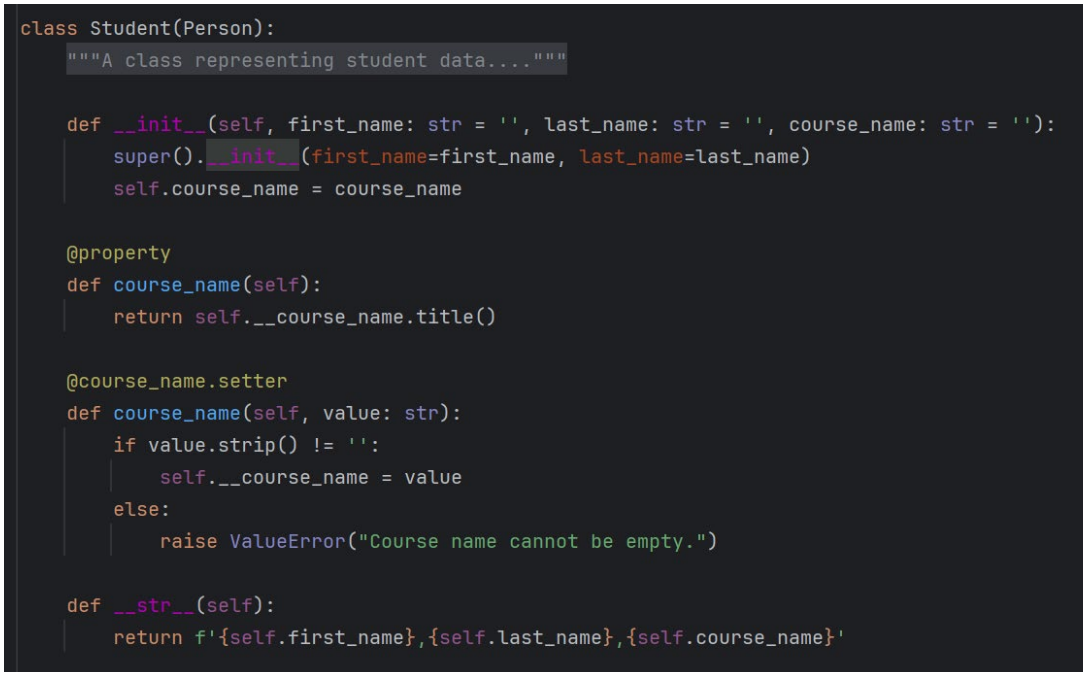
```python
class Person:  1 usage
    """A class representing person data...."""
    def __init__(self, first_name: str = '', last_name: str = ''):
        self.first_name = first_name
        self.last_name = last_name

    @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("The first name should only contain letters")

    @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("The last name should only contain letters")

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

**Figure 1.1: Person Class storing basic information**

## Student

The Student class represents an individual student enrolling in a course, inheriting information from the Person class. By inheriting the information, it automatically gets all the functionality of Person without the need to rewrite it

Building upon the first and last name, Student adds the course_name. The property includes validation to ensure the course name is not blank and cannot be only spaces preventing the program from saving incorrect or incomplete information. The __str__() method also returns the full student record in a comma-separated format (Figure 1.2).



```python
class Student(Person):
    """A class representing student 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

    @property
    def course_name(self):
        return self.__course_name.title()

    @course_name.setter
    def course_name(self, value: str):
        if value.strip() != '':
            self.__course_name = value
        else:
            raise ValueError("Course name cannot be empty.")

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

**Figure 1.2: Student class demonstrating inheritance by reusing what already exists**

## Summary

By creating separate classes for storing data, file handling, and managing user interaction, the program becomes cleaner and less repetitive. By inheriting from the Person class to create the Student class, you can reuse existing code instead of having to write everything from the beginning. Object-oriented programming can turn a simple script into a more structured application.