Nolan Davis

November 26, 2023

IT FDN 110

Assignment 07

https://github.com/ndavis91/IntroToProg-Python-Mod07

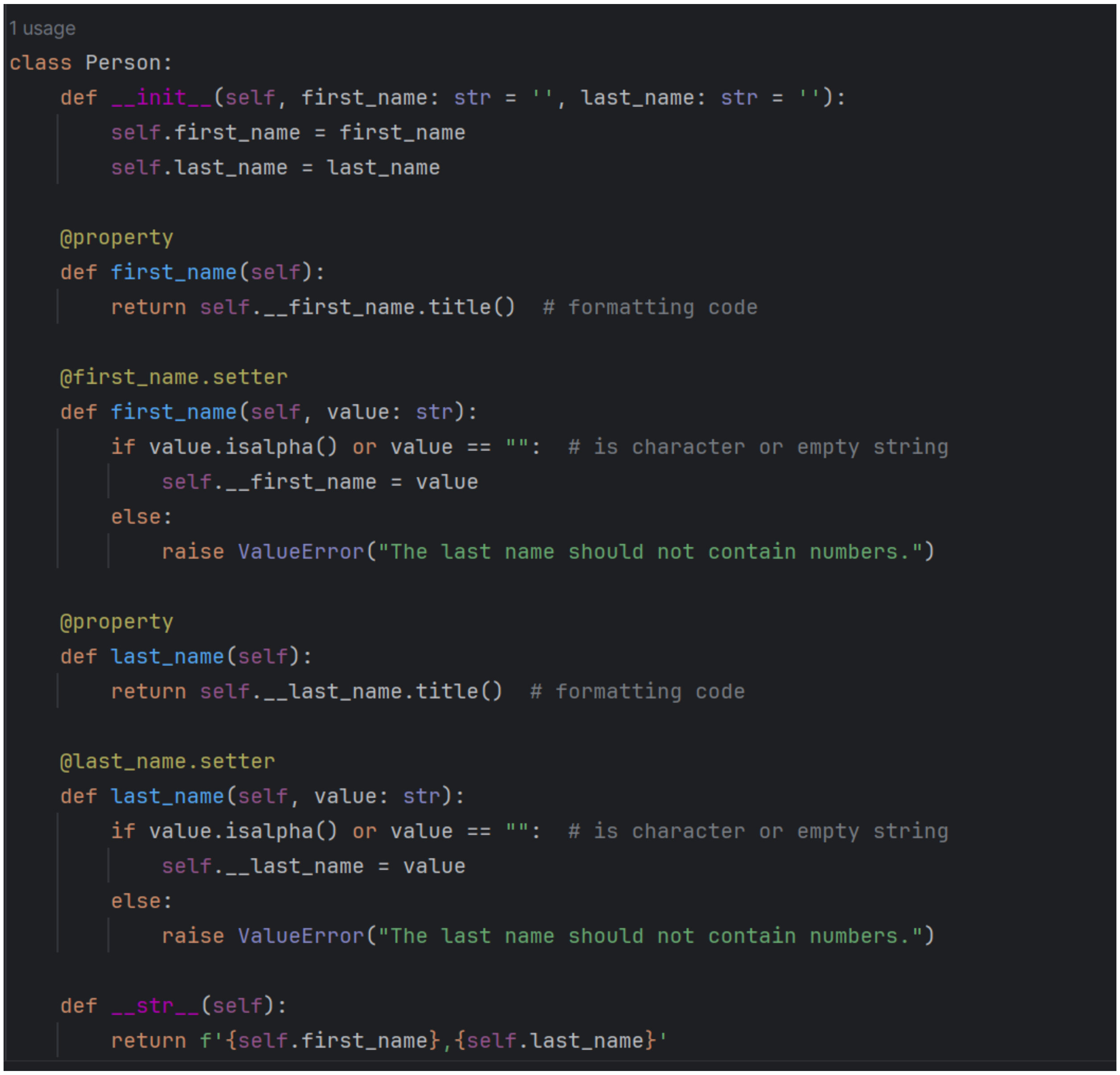
Working with Classes, Objects and Inheritance

Introduction

This week continued the theme of organizing our code by introducing classes and objects. Classes are a useful way of grouping functions and data together to improve readability and maintainability. Functions that are defined within classes are called methods. Additionally, by using a constructor inside of a class we can initialize objects with a defined state.

Creating the Program

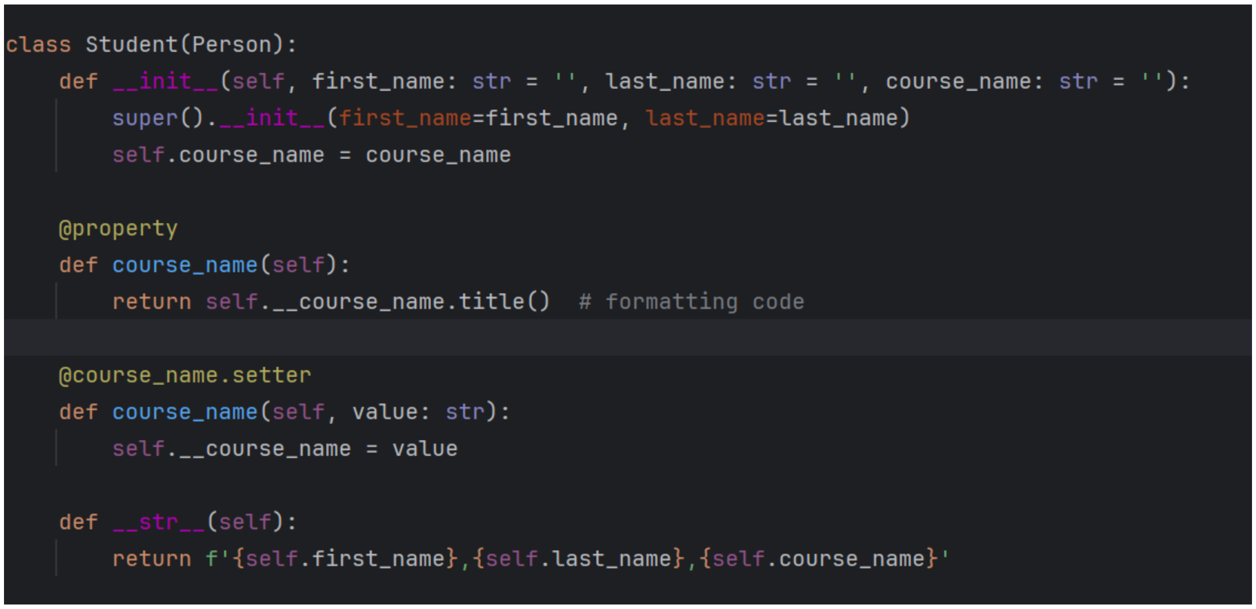
I started by creating the Person class, and using the constructor to set default values of blank strings for first_name and last_name. I created “getters” and “setters” for first_name and last_name, and moved the error handling into the setters so that any new names that are input get checked to make sure no numbers were included. See Figure 1.



```
1 usage
class Person:
    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()  # formatting code

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

    @property
    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 character or empty string
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers.")

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

Figure 1 – Defining the Person class, with first_name and last_name

Next I created the Student class, and input the Person class into the parenthesis to inherit the attributes first_name and last_name using the super().__init__() constructor. After that I created a getter and setter for course_name inside the Student class. Finally I returned the F-string to override the default memory location. See Figure 2.



```python
class Student(Person):
    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()  # formatting code

    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value

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

Figure 2 – Defining Student class and inheriting from Person class.

Once the new classes were defined, I went back into the program and updated the calls to use the class.function call and removed error handling from the input_student_data() function. I also added lines to create a local variable for a dictionary for reading and writing from the JSON file, because we were now using student_data as a list, instead of a dictionary. Lastly I went back and added helpful comments to my functions and classes to make sure it was more readable for future maintenance.

Testing the program

To verify the program I ran in Pycharm, saving new students to the Enrollments.json file, reading and writing to the file, as well as verifying that the error handling still worked. Once I was confident that the program worked in Pycharm, I verified that it also ran smoothly in Command Prompt.

Summary

Using classes was hard to wrap my head around at first, especially the use of “self” in front of the parameters. However once I had watched all of the recommended videos as well as some I found on my own, it began to click. It seemed like a lot of work for such a small program, but I do agree that for larger programs this method of organization will be extremely helpful.