Derek Klock
3/12/2025
IT FDN 110 A
GitHubURL

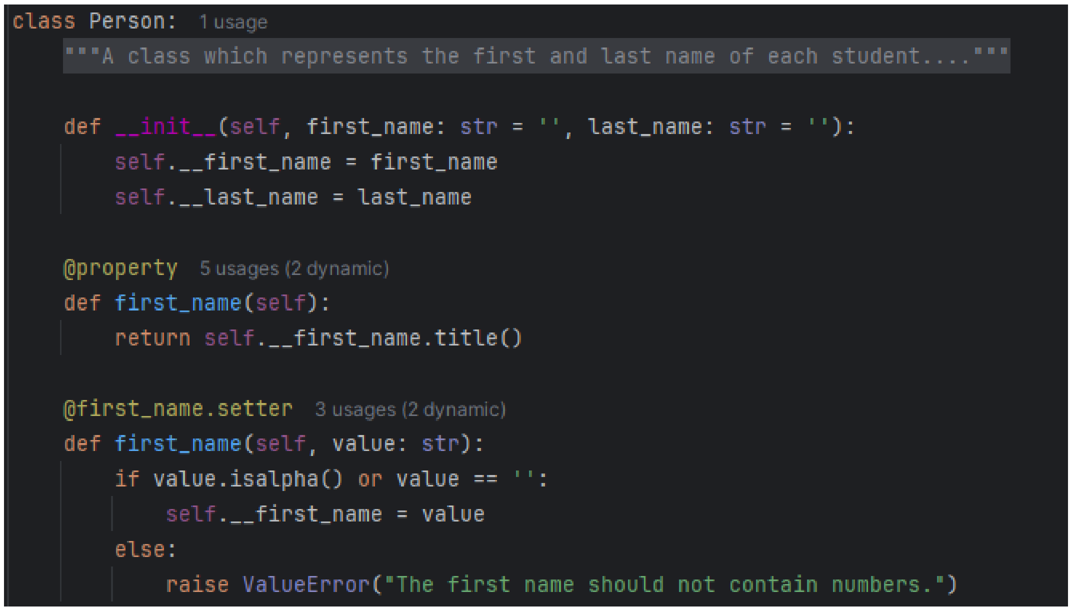
Assignment 7 – Classes and Functions II

Introduction

This week saw the completion of the student registration script which has been the focus of this class up until this week. The final evolution of this script included concepts such as the "self" keyword, inheritance, and properties to manage attribute data. No new functionality was added to the script this week but it was substantially refined, segmented, and encapsulated.

Class Improvements

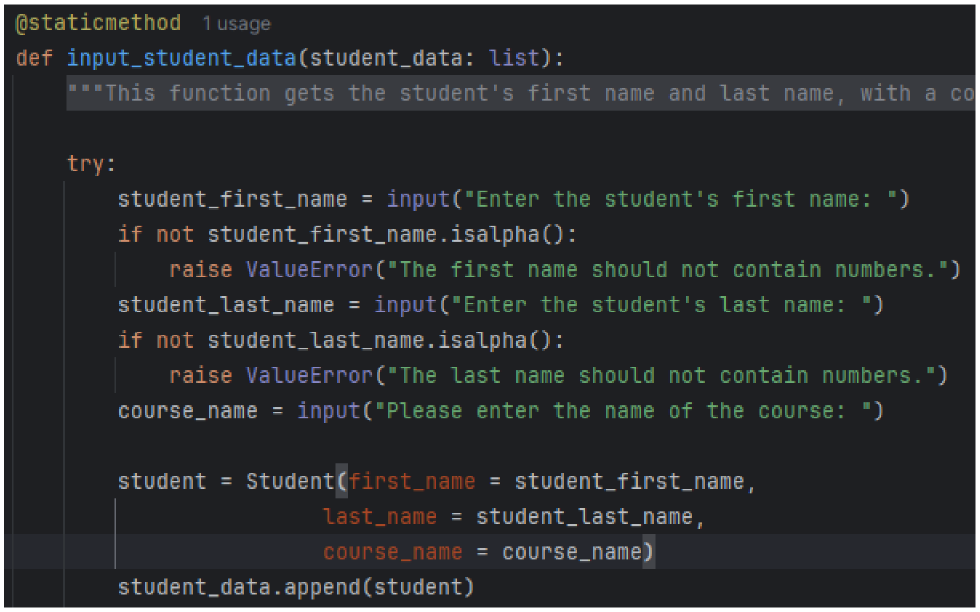
Starting this week's script two new classes were created "Person" and "Student". Class Person represented the first and last name of each student and class Student represented the registered course of each student. This functionality was pulled out the IO class of assignment 6. Each of the two new classes utilize a constructor "__init__" to initialize attributes. The attributes within these two classes were marked private with the double "_" before their names which helps to protect them from unwanted modification, figure 1.



```
class Person:  1 usage
    """A class which represents the first and last name of each student...."""

    def __init__(self, first_name: str = '', last_name: str = ''):
        self.__first_name = first_name
        self.__last_name = last_name

    @property  5 usages (2 dynamic)
    def first_name(self):
        return self.__first_name.title()

    @first_name.setter  3 usages (2 dynamic)
    def first_name(self, value: str):
        if value.isalpha() or value == '':
            self.__first_name = value
        else:
            raise ValueError("The first name should not contain numbers.")
```

Figure 1. The setter method includes validation logic for the user input.

To modify these private attributes a getter method with the @property decorate retrieves the value of the attribute. Then the setter method, whose decorator name matches the getter method name, sets the attribute to equal the parameter "value". The setter method also has some simple data validation logic to ensure "value" is composed of only alphabetic characters. "Value" takes on the value passed into the method when it was called. For the Person and Student classes these values are the user input captured in the IO class method input_student_data, figure 2. Only the Student class is called in the input_student_data method but the functionality of both classes Person and Student are activated. This is because Student inherits the returned values of "first_name" and "last_name" from the Person class.



```
@staticmethod  1 usage
def input_student_data(student_data: list):
    """This function gets the student's first name and last name, with a co

    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")

        student = Student(first_name = student_first_name,
                          last_name = student_last_name,
                          course_name = course_name)
        student_data.append(student)
```

Figure 2. The specific error messages for the first and last name validation are contained within the IO lass method input_student_data.

Summary

The inputted student information is now protected within the Person and Student classes. As opposed to being directly defined within the IO method input_student_data. This structure improves the quality of the data and allows for the functionality to be more modular. As the Persona and Student classes can now be called by multiple methods if desired without having to duplicate the script logic. This makes the script as a whole more easily maintained and updated and cuts down on its volume. This functionality was only made possible by using the new concepts of the getter/setter methods and inheritance.