<Kseniia Temnikova>

<09/11/2024>

<Programming Basics>

<Assignment_7>

<GitHub URL>

# Basic Python program

## Introduction

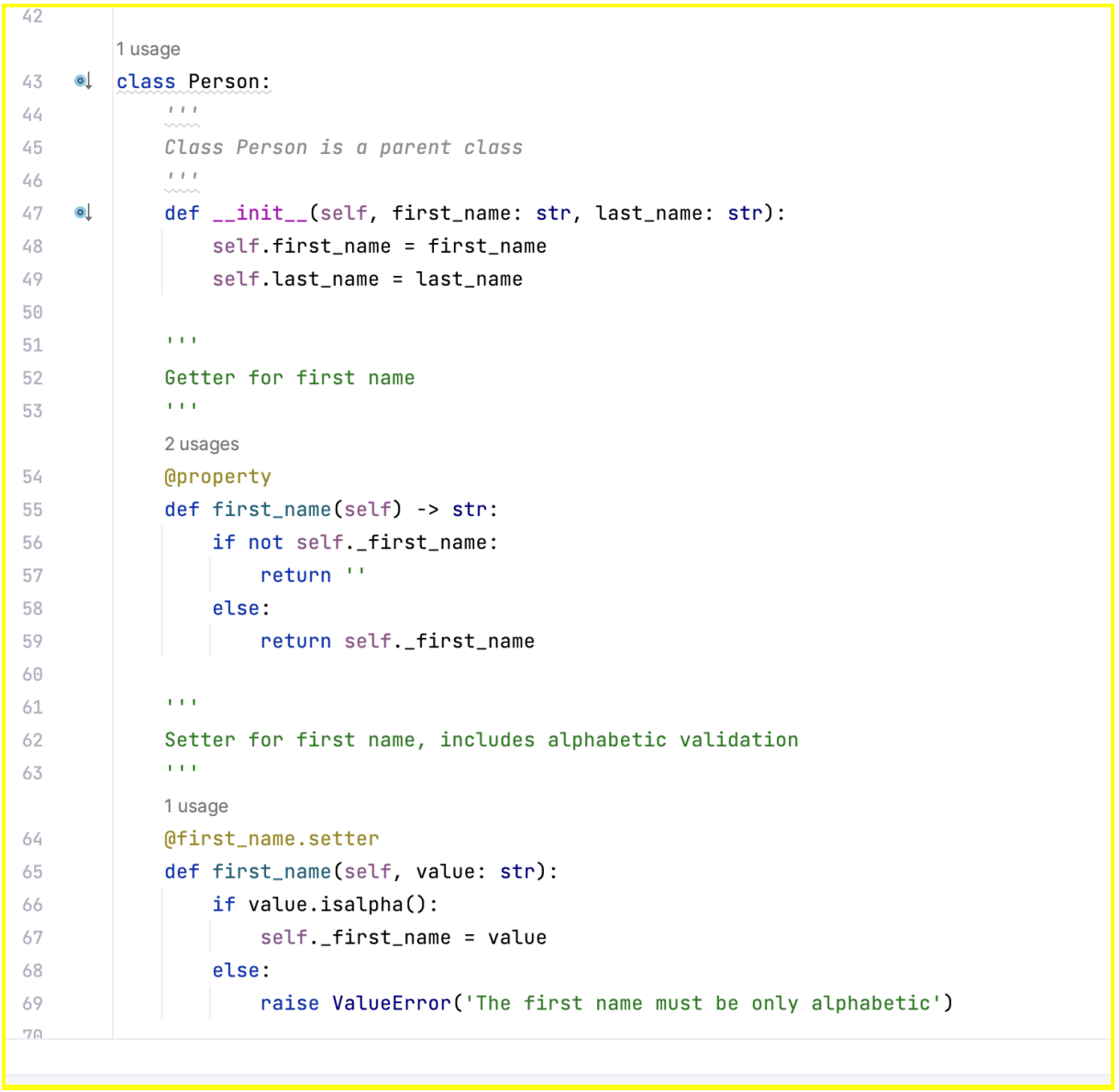
In this assignment, I created a Python program that simulates a basic course registration system. The program enables users to register students by inputting their first name, last name, and the course they want to enroll in. The data is structured using Python objects and stored in a list, which can then be saved to a JSON file for future reference.

## *Classes*

The program features two main classes: **Person** and **Student**. The `Person` class represents a basic entity with two properties: `first_name` and `last_name`. (Figure 1) These properties are managed using getters and setters with simple validation to ensure that only alphabetic characters are allowed. The class also includes a `__str__()` method that formats the output, providing a readable string representation of the person's details.

Building on the `Person` class, the `Student` class adds another property: `course_name` (Figure 2). This class inherits the properties and methods of `Person` and introduces its own validation for the `course_name`. The `Student` class also overrides the `__str__()` method to include the course name in the output.

In addition to defining these classes, the program demonstrates the process of creating class instances. For each new student, the program instantiates a `Student` object by passing the first name, last name, and course name to the constructor. The student object is then stored in a list and can be displayed or saved to a file.



```
1 usage
class Person:
    '''
    Class Person is a parent class
    '''
    def __init__(self, first_name: str, last_name: str):
        self.first_name = first_name
        self.last_name = last_name

    '''
    Getter for first name
    '''
    2 usages
    @property
    def first_name(self) -> str:
        if not self._first_name:
            return ''
        else:
            return self._first_name

    '''
    Setter for first name, includes alphabetic validation
    '''
    1 usage
    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha():
            self._first_name = value
        else:
            raise ValueError('The first name must be only alphabetic')
```

Figure 1: Class Person

```
96
97
      4 usages
98    class Student(Person):
99        def __init__(self, first_name: str, last_name: str, course_name: str|float):
100           super().__init__(first_name=first_name, last_name=last_name)
101           self.course_name = course_name
102
103       '''
104       Getter for course name, includes alphabetic validation
105       '''
          1 usage
106       @property
107       def course_name(self) -> str:
108           if not self._course_name:
109               return ''
110           else:
111               return self._course_name
112
113
114       '''
115       Setter for course name, includes alphabetic validation
116       '''
          1 usage
117       @course_name.setter
118       def course_name(self, value: str | float):
119           if not value:
120               raise ValueError("The course name can't be empty")
```

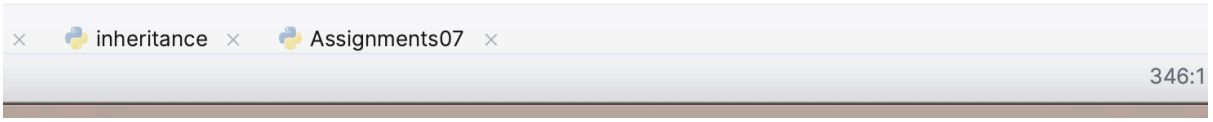



Figure 2: Class Student

## Test

I confirmed that the program can save the user's input to a JSON file, ensuring that the data is written correctly and can be viewed in a text editor. I tested the program's capability to handle multiple registrations, verifying that each set of data is stored in a list of dictionaries and displayed accurately when requested.(Figure 3)

Furthermore, I checked that the program correctly saves all registrations to the JSON file, preserving the data for future use. Finally, I ran tests in both PyCharm and the terminal to ensure the program works consistently and as expected in different environments.(Figure 4)

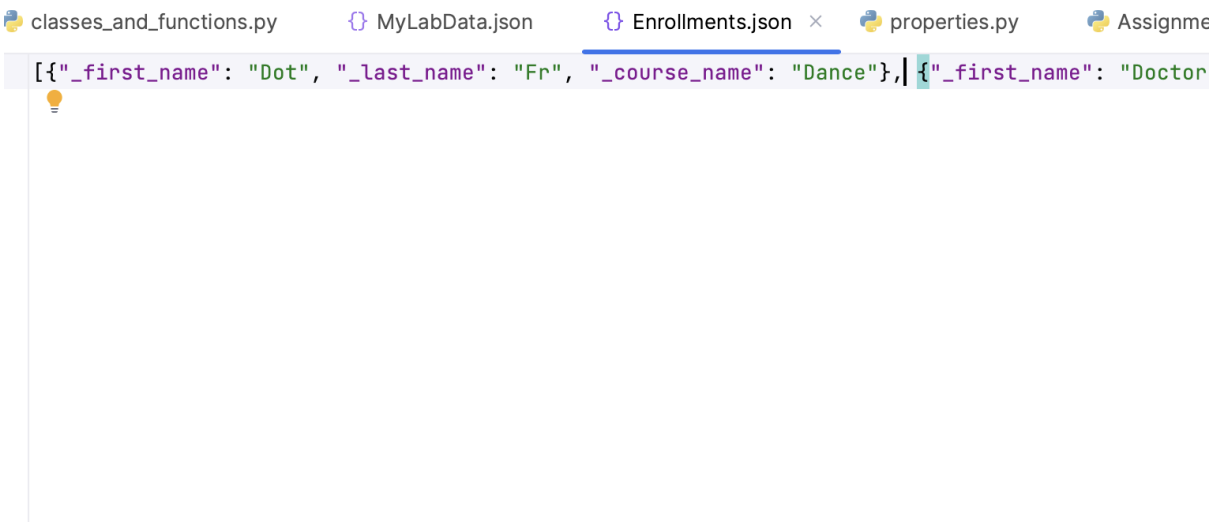



Figure 3: JSON file

**Summary**

This assignment helped me better understand how to create and use classes in Python, and how to manage class inheritance and encapsulation. By using the `Person` and `Student` classes, I was able to efficiently organize and manipulate student data in a more structured way.

By following these practices, the program maintains a clear and modular structure, making it easy to understand, use, and maintain.

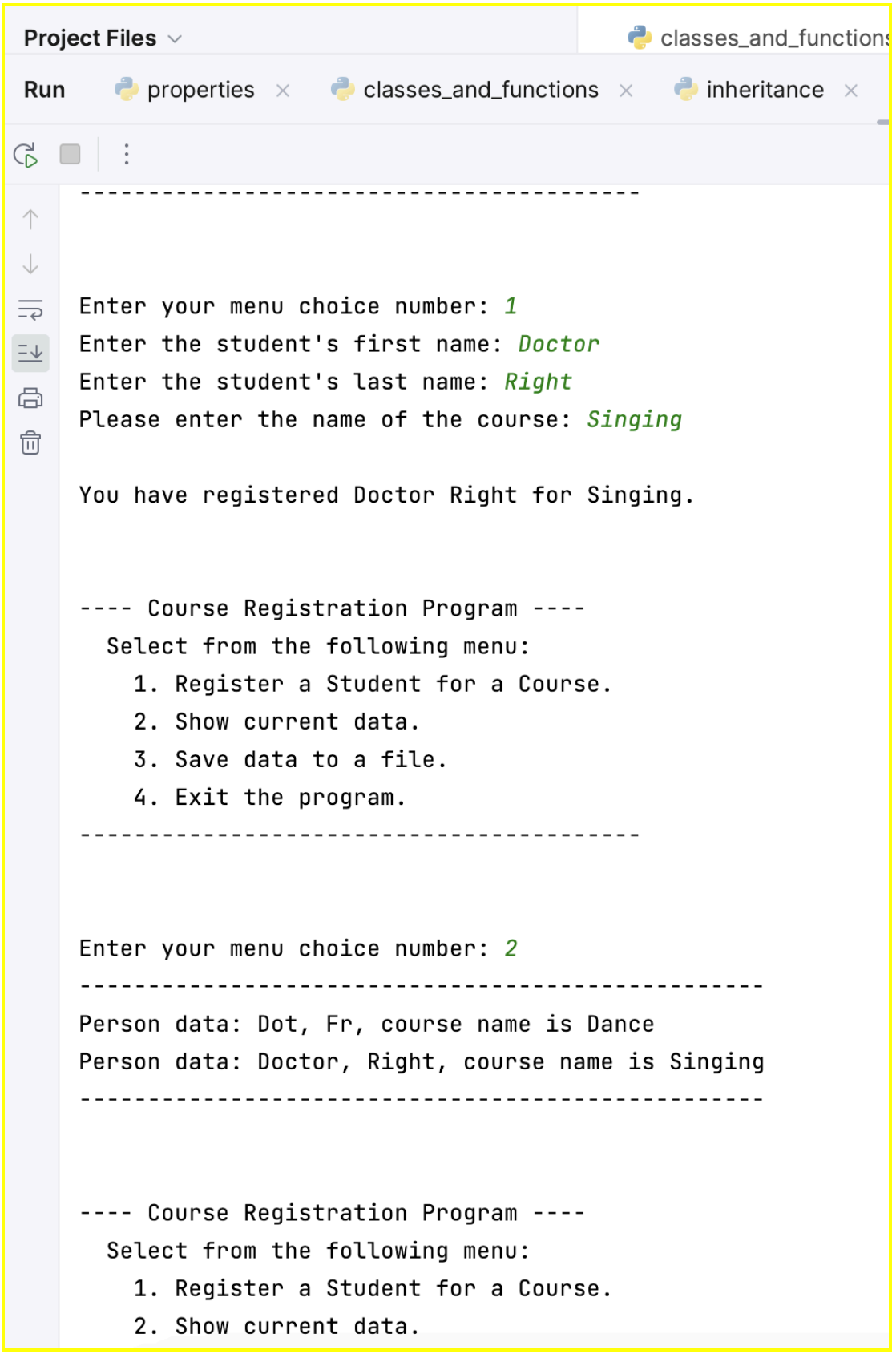



Figure 3: PyCharm test