FERNANDO TAMAYO GRADOS
December 9, 2023
IT FDN 110 B
Assignment 7
https://github.com/fernandotg123/IntroToProg-Python-Mod07

Assignment 7

1. Intro

This assignment doubles down on functions and classes in Python, expanding on the implementation of classes. In particular, it focuses on the data classes, which allows to manage data about people and students, through attributes i.e., characteristics, and constructors.

2. Data Classes

The program now has three types of classes: data classes, processing classes and presentation classes. In the previous assignment we worked on the latter two classes. To enhance the program, there will a total of four classes, from which two will be data classes: Person, which will hold first and last name, including all attributes. And then, Student (Person), which will describe students and the course they are taking. For a student to exist, a person must exist first.

3. Class Person

See the code below to identify all attributes of the “person”:

```
class Person:
    """
    A class representing person data.

    Properties:
    - first_name (str): The person's first name.
    - last_name (str): The person's last name.

    ChangeLog:
    - Fernando Tamayo Grados, 12/9/2023, Executed Homework
    """
    first_name: str = ''
    last_name: str = ''

    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 not contain numbers.")

    @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 not contain numbers.")

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

4. Class Student (Person)

Also, here is the code for students:

```
class Student(Person):
    """
        A class representing student data.

        Properties:
        - first_name (str): The student's first name.
        - last_name (str): The student's last name.
        - course (str): The name of the course.

        ChangeLog:
        - Fernando Tamayo Grados, 12/9/2023, Executed Homework.
    """
    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
```

5. Resulting changes in other classes

To effectively use these data classes, we need to convert the student dictionaries to student objects, which we will do on the *FileProcessor* class. Hence, both functions within this class will need changes in the code. Below in bold I am showing the changes in the read_data_from_file function:

```
@staticmethod
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads data from a json file and loads it into a list of
dictionary rows

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created function
    Fernando Tamayo Grados, 12/11/2023, Converted dictionary to objects

    :param file_name: string data with name of file to read from
    :param student_data: list of objects to be filled with file data

    :return: list
    """
    student_object_list = []
    try:
        file = open(file_name, "r")
        student_data = json.load(file)
        for student in student_data:
            student_object =
Student(student["first_name"],student["last_name"],student["course_name"])
            student_object_list.append(student_object)
        # access info on the objects
        file.close()
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with
reading the file.", error=e)
    finally:
        if file.closed == False:
            file.close()
    return student_object_list
```

Furthermore, we need to pursue the change in the presentation class IO. We will do the change in the function output_student_and_course_names:

```
@staticmethod
def output_student_and_course_names(student_data: list):
    """ This function displays the student and course names to the user

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created function
    Fernando Tamayo Grados, 12/11/2023, Converted dictionary to objects

    :param student_data: list of dictionary rows to be displayed

    :return: None
    """

    print("-" * 50)
    for student in student_data:
        print(student.first_name, student.last_name, student.course_name)
    print("-" * 50)
```

And also the function input_student_data:

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

```
    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created function
    Fernando Tamayo Grados, 12/11/2023, Converted dictionary to objects

    :param student_data: list of dictionary rows to be filled with input data

    :return: list
    """

    try:
        first_name = input("Enter the student's first name: ")
        if not first_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        last_name = input("Enter the student's last name: ")
        if not last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")
        new_student = Student(first_name=first_name, last_name=last_name,
course_name=course_name)
        student_data.append(new_student)
        print()
        print(f"You have registered {first_name} {last_name} for
{course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was the correct type
of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your
entered data.", error=e)
    return student_data
```

8. Conclusion

Overall, the rest of the classes works as expected, thus running the program without issues.