Name: Joseph Rideout **Date :** November 29, 2023 **Course:** File demonstrates python class
Mod07-Assignment 07

Python Class

Introduction

This document introduces the seventh assignment, which showcases the utilization of class. Covering the fundamentals. Additionally, I found Mr. Randel Root live sessions to be helpful for understanding python programming.

Class

A class is a blueprint or a template for creating objects. A class defines the properties and behaviors that objects created from it will possess. Once a class is defined, you can create multiple instances of that class with their own unique data but sharing the same structure and behavior defined in the class.

In Python classes, __**init**__ and __**str**__ are special methods that serve distinct purposes:
__**init**__ is the constructor method in Python classes. It's automatically called when an instance of the class is created. It allows the class to accept parameters during object instantiation and set initial values for attributes.

__**str**__ is a special method used to represent a human-readable string representation of the object. It's called when the **str()** function is used or when an object is converted to a string, for example, by using **print**. It should return a string that provides a meaningful description of the object's state.

Assignment 07

For this week assignment is to create a person and student class. The Python class, **Person**, is designed to represent a person with attributes for their first and last names while enforcing specific rules for their input. The Python class, **Student**, inherits from the **Person** class and adds specific attributes and methods related to a student's course.

Here is example of the Python 3 code:

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,11/27/2023,Created Script
#   <Joseph Rideout>,<11/27/23>,<Activity>
# ------------------------------------------------------------------------------------------ #
import json


# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
   1. Register a Student for a Course.
    2. Show current data.
   3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.

class Person:
    """ presents a person."""

    def __init__(self, first_name: str = "", last_name: str = ""):
        self.student_first_name = first_name
        self.student_last_name = last_name

    @property
    def student_first_name(self):
        return self.student_first_name.title()

    @student_first_name.setter
    def student_first_name(self, value: str):
        if value.isalpha() or value == '':
            self._first_name = value.title()
        else:
            raise ValueError("The first name should not contain numbers.")

    @property
    def student_last_name(self):
        return self._last_name

    @student_last_name.setter
    def student_last_name(self, value: str):
        if value.isalpha() or value == '':
            self._last_name = value.title()
        else:
            raise ValueError("The last name should not contain numbers.")
    def __str__(self):
        return f"Student Name: {self.student_first_name} {self.student_last_name}"

    @staticmethod
    def validate_name(name):
        if isinstance(name, str):
            return name.title() if name.strip() else ""
        else:
            return ""

class Student(Person):
    """Represents a student."""

    def __init__(self, first_name="", last_name="", course_name=""):
        super().__init__(first_name, last_name)
        self._course_name = course_name

    @property
    def course_name(self):
        return self._course_name

    @course_name.setter
    def course_name(self, value):
        # Add validation logic if needed
        self._course_name = value

    def __str__(self):
        return f"Student Name: {self.student_first_name} {self.student_last_name}, Course:
{self.course_name}"


# Processing --------------------------------------- #
class FileProcessor:
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created Class
    """

    @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

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

        :return: list
        """

        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            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_data

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a list of dictionary rows

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :param file_name: string data with name of file to write to
        :param student_data: list of dictionary rows to be writen to the file

        :return: None
        """

        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            IO.output_student_and_course_names(student_data=student_data)
        except Exception as e:
            message = "Error: There was a problem with writing to the file.\n"
            message += "Please check that the file is not open by another program."
            IO.output_error_messages(message=message, error=e)
        finally:
            if file.closed == False:
                file.close()


# Presentation --------------------------------------- #
class IO:
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created Class
    RRoot,1.2.2030,Added menu output and input functions
    RRoot,1.3.2030,Added a function to display the data
    RRoot,1.4.2030,Added a function to display custom error messages
    """
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays the a custom error messages to the user

        ChangeLog: (Who, When, What)
        RRoot,1.3.2030,Created function

        :param message: string with message data to display
        :param error: Exception object with technical message to display

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function



        :return: None
        """
        print()  # Adding extra space to make it look nicer.
        print(menu)
        print()  # Adding extra space to make it look nicer.

    @staticmethod
    def input_menu_choice():
        """ This function gets a menu choice from the user

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :return: string with the users choice
        """
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if choice not in ("1", "2", "3", "4"):  # Note these are strings
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages(e.__str__())  # Not passing e to avoid the technical
message

        return choice

    @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

        :param student_data: list of dictionary rows to be displayed

        :return: None
        """

        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

    @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

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

        :return: list
        """

        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.replace("-", "").replace("'", "").isalpha():
                raise ValueError("The first name should only contain letters, hyphens, or
apostrophes.")

            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.replace("-", "").replace("'", "").isalpha():
                raise ValueError("The last name should only contain letters.")

            course_name = input("Please enter the name of the course: ")
            student = {"FirstName": student_first_name, "LastName": student_last_name,
"CourseName": course_name}
            student_data.append(student)
            print(f"You have registered {student_first_name} {student_last_name} for
{course_name}.")
        except ValueError as e:
            IO.output_error_messages(message="Please enter valid data for student details!",
error=e)
        except Exception as e:
            IO.output_error_messages(message="Error encountered while inputting student
data.", error=e)
        return student_data
```

```
# Start of main body

# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
while True:

    # Present the menu of choices
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_and_course_names(students)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Summary

This document demonstrates the basic classes and methods used in Python programming. The knowledge gained from this week's assignment was an essential part of understanding Python 3 programming