Kiem Hollis
3/12/2025
Foundations of Python
Assignment06
GitHubURL

Module 7

Introduction: I am creating a program that demonstrates using constants, variables, set of data classes, and print statements to display a message about a student's registration for a Python course.

I copied and pasted the Script Header provided by the instructor and updated the name and current date section.

I opened up the starter file and used it as a starting point for my assignment.

I added the two classes: Person and Student

```
class Person:
    # TODO Add first_name properties to the constructor
    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()

    # TODO Create a getter and setter for the first_name property
    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":  # Adding validation code
            self.__first_name = value

        else:
            raise ValueError ("The first name should only contain alphabets.")

    @property
    # TODO Add last_name properties to the constructor
    def last_name(self):
        return self.__last_name.title()
    # TODO Create a getter and setter for the last_name property
    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":  # Adding validation code
            self.__last_name = value
        else:
            raise ValueError("The last name should only contain alphabets.")
```

```
# TODO Create a Student class the inherits from the Person class
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.
    - gpa (float): The gpa of the student.

    ChangeLog:
    - RRoot, 1.1.2030: Created the class.
    """

    def __init__(self, first_name: str = '', last_name: str = '', course_name:
str = ''):
        Person.__init__(self, first_name, last_name)
        self.course_name = course_name

    # TODO Override the __str__() method to return Person data
    def __str__(self):
        return f'{self.first_name},{self.last_name},{self.course_name}'

    @property
    # TODO add the getter for course_name
    def course_name(self):
        return self.__course_name.title()

    # TODO add the setter for course_name
    @course_name.setter
    def course_name(self, value: str):
        self.__course_name = value
```

I updated the File Processor and IO:

```
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):
        """ This function reads data from a json file and loads it into a list
of dictionary rows
        then returns the list filled with student data.
```

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

        :param file_name: string data with name of file to read from

        :return: list
        """
        student_objects = []
        try:
            # Get a list of dictionary rows from the data file
            file = open(file_name, "r")
            json_students = json.load(file)

            # Convert the list of dictionary rows into a list of Student objects

            # TODO replace this line of code to convert dictionary data to
Student data

            for json_data in json_students:
                first_name = json_data["FirstName"]
                last_name = json_data["LastName"]
                course_name = json_data["CourseName"]
                student = Student(first_name, last_name, course_name)
                student_objects.append(student)

            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_objects

    @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:
            # TODO Add code to convert Student objects into dictionaries (Done)
            student_list = list()
            for student in student_data:
                data = {"FirstName":student.first_name,
                        "LastName":student.last_name,
                        "CourseName":student.course_name}
                student_list.append(data)

            file = open(file_name, "w")
            json.dump(student_list, 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:

            # TODO Add code to access Student object data instead of dictionary
data
            print(student)

        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.isalpha():
                raise ValueError("The last 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: ")

            # TODO Replace this code to use a Student objects instead of a
dictionary objects
            student = Student(student_first_name, student_last_name,
course_name)

            student_data.append(student)
            print()
            print(f"You have registered {student_first_name} {student_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
```

In summation, I created a program that demonstrates using constants, variables, set of data classes, and print statements to display a message about a student's registration for a Python course..