Julia Westfall
May 27, 2024
Foundations of Programming: Python
Assignment07

More Advanced Usage of Functions and Classes

Repository Link:
https://github.com/Retrocupcake/IntroToProg-Python-Mod07

Introduction

This assignment continued the use of functions and classes in a more advanced way. I did this assignment by looking through videos in the course module and also worked heavily with a professional python software programmer to learn the material. I did this assignment immediately after assignment 5 and 6 due to being behind due to family visiting town for two weeks.

Starting the Assignment

As stated in the assignment directions, this assignment was very similar to assignment 05. I had a particularly tough time with assignment 05 so this did not excite me. However, in the end it was not too bad. I started the assignment by opening the starter file 07. The following code is what the starter file looked like before any edits were made. I really appreciated how the file had to do comments written in there to help me follow what I was supposed to be doing.

```
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.


# TODO Create a Person Class
# TODO Add first_name and last_name properties to the constructor (Done)
# TODO Create a getter and setter for the first_name property (Done)
# TODO Create a getter and setter for the last_name property (Done)
# TODO Override the __str__() method to return Person data (Done)

# TODO Create a Student class the inherits from the Person class (Done)
# TODO call to the Person constructor and pass it the first_name and last_name
data (Done)
# TODO add a assignment to the course_name property using the course_name
parameter (Done)
# TODO add the getter for course_name (Done)
# TODO add the setter for course_name (Done)
# TODO Override the __str__() method to return the Student data (Done)



# 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
```

```python
        :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.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: ")
            student = {"FirstName": student_first_name,
                            "LastName": student_last_name,
                            "CourseName": 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


 # 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")
```

Assignment07_starter code file

Actual Edits Made to Starter Code

A lot of the initial constants and variables were the same as previous coding assignments. The biggest additions to the starter code were adding the Person class and afterward the student class then adding all the contents inside of those classes (called methods and attributes) that will allow us to assign attributes to those classes and “get” and “set” the attributes within a particular class. I found this hard concept to wrap my head around initially and so I chose to comment and doc string throughout my code in hopes of furthering my understanding in these areas quicker. The following provides a snapshot of my commented code.

```
class Person:
   """ This is the person class
```

```
    """
    def __init__(self,
                 first_name: str = '',
                 last_name: str = ''):

        # Check to make sure the given values are strings, otherwise error
        if isinstance(first_name, str) and isinstance(last_name, str):
            self._first_name = first_name
            self._last_name = last_name
        else:
            raise TypeError('First Name and Last Name must be strings')

    def get_first_name(self):
        """ Allows us to get the first name"""
        return self._first_name

    def get_last_name(self):
        """ Allows us to get the last name"""
        return self._last_name

    def set_first_name(self, first_name):
        """ Allows us to set the first name"""
        if isinstance(first_name, str):
            self._first_name = first_name
        else:
            raise TypeError('first_name must be a string.')

    def set_last_name(self, last_name):
        """ Allows us to set the last name"""
        if isinstance(last_name, str):
            self._last_name = last_name
        else:
            raise TypeError('last_name must be a string.')

    def __str__(self):
        return f'{self._first_name}, {self._last_name}'


class Student(Person):
    """ This is a student class which inherits attributes from Person object
    """
    def __init__(self,
                 first_name: str = '',
                 last_name: str = '',
                 course_name: str = ''):
        """ Super allows to take attributes from Person object
        """
        super().__init__(first_name, last_name)
        if isinstance(course_name, str):
            self._course_name = course_name
        else:
            raise TypeError('course_name must be a string.')

    def get_course_name(self):
        return self._course_name

    def set_course_name(self, course_name):
        if isinstance(course_name, str):
            self._course_name = course_name
        else:
            raise TypeError('course_name must be a string.')
```

Actual code from python assignment 07

Additional Edits Made to Code

This addition created one student for each iteration of the loop from our json data (See below). Recall our json data in this case is a list of dictionaries. We are appending student objects to student data.

```
try:
   file = open(file_name, "r")
   temp_student_data = json.load(file)
   file.close()

   # load the data into Student objects and append them to student_data
   for student in temp_student_data:
       _student = Student(first_name=student['FirstName'],
                          last_name=student['LastName'],
                          course_name=student['CourseName'])
       student_data.append(_student)
```

Biggest Challenges/Takeaways

I would like to do more with classes as I still find them confusing. Error handling is probably my favorite piece of this as well as the functions.

Summary

In summary, this was a difficult assignment but allowed me to gain more practice with classes, functions, and error handling. I am very thankful for the starter file this time and that I have a tutor in real time I could turn to to ask questions and complete these assignments.