Introduction to Programming with Python

Assignment 07

https://github.com/Mossy-Squatch/IntroToProg-Python-Mod07

Module 07 – JSON, Classes, Functions and Objects

Introduction

In this document I will describe the steps I took to create a Python script that reads a JSON file, then stores that data. It then presents the user with menu choices, takes the user's selection and performs a task based on that input. User provided information can then be stored with the data read from the JSON file. This process loops until the user selects to exit the program. This assignment expands on the assignment from Module 06, in addition to classes for file processing and I/O we add person and student, with student inheriting from the person class. We define all classes and functions prior to the main program and call these from within the program body depending on case. We also continue to work with structured error handling within the script.

Creating the script

I began by opening Visual Studio Code and reviewing the starter file for this assignment and the instructions in the Mod07-Assignment file.

- Before defining any constants or variables I used “import json” and “from sys import exit”.
- Under “# Define the Data Constants” I created a “MENU” constant that is used to present the user with their options, I also created a “FILE_NAME” constant that is used for the JSON file.
- Under “# Define the Data Variables” I created Variables for menu_choice and students. All other variables are now defined locally in the functions where they are used.
    - Imports, constants and variables can be seen in the image below.

```
import json
from sys import exit

# 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 and constants

menu_choice: str ="" # Hold the choice made by the user.
students: list = []  # A table of student data

```

- Next, I defined the class “Person”. Within this I defined first and last name properties and included some error handling to that the user didn’t enter any non-letter characters.
    - This can be seen in the image below.

```python
#Define class Student
class Person:

    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("First name should only contain letters!")

    @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 only contain letters!")

```

- After defining “Person” I created and defined the class “Student”. This inherited first and last name from “Person” as well as included the property for course name. I also use error handling to check that the value was not left blank from the user input.
    - This is shown in the screenshot below.

```python
# Define class Student, with inheritance from Person
class Student(Person):

    def __init__(self, first_name: str, last_name: str, course_name: str = ""):
        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: str):
        if value != "":
            self.__course_name = value
        else:
            raise ValueError("The course name cannot be left blank!")

```

- Under “# File processing” I defined the class “FileProcessor” and then created functions for reading and writing to the JSON file. Each function is converted to a static function that belongs to the class FileProcessor by using @staticmethod. Error handling is also incorporated in these functions.
    - This can be seen in the screenshot below.

```python
# File processing - Define class FileProcessor
class FileProcessor:
    # Function for reading data from file
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        try:
            with open(file_name, "r") as file:
                student_data = json.load(file)
        except FileNotFoundError as e:
            IO.output_error_messages("File does not exist!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()
        return student_data

    # Function for writing user input data to file
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        try:
            with open(file_name, "w") as file:
                json.dump(student_data, file)
            print("-" * 50)
            print("INFO: All rows saved to file!")
            print("-" * 50)
        except TypeError as e:
            IO.output_error_messages("Data is not in valid JSON format!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if file.closed == False:
                file.close()


```

- Next, I created the class IO. Here I created functions for handling error messages, prompting user menu choices, gathering student information, displaying the data, saving the data and exiting the program.
    - The classes, related functions and error handling can be seen in the screenshots below.

```python
# User input and output - Define class IO
class IO:
    # Function for displaying error messages
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message --")
            print(error, error.__doc__, type(error), sep="\n")

    # Function for displaying the menu options to the user
    @staticmethod
    def output_menu(menu: str):
        print()
        print(menu)
        print()

    # Function for prompting user menu selection and storing choice
    @staticmethod
    def input_menu_choice():
        choice = "0"
        try:
            choice = input("Select a menu option (1-4): ")
            if choice not in ("1", "2", "3", "4"):
                raise Exception("Invalid selection!\nYou must select (1-4)!")
        except Exception as e:
            IO.output_error_messages(e.__str__())
        return choice

    # Function for displaying current data to user
    @staticmethod
    def output_student_courses(student_data: list):
        print()
        print("The current data is: ")
        for student in student_data:
            student_first_name = student["FirstName"]
            student_last_name = student["LastName"]
            course_name = student["CourseName"]
            print(f"{student_first_name},{student_last_name},{course_name}")

```

```python
    # Function for user to enter student data
    @staticmethod
    def input_student_data(student_data: list):
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The student's first name should only contain letters!")

            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The student's 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()
            print(f"You have registered {student["FirstName"]} {student["LastName"]} for {student["CourseName"]}!")
            print()
        except ValueError as e:
            IO.output_error_messages("Value entered was not the correct type of data!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        return student_data

```

- Once the file processing and IO had all been defined, the main body of the program was created. It starts with reading the data from the file using FileProcessor.read_data_from_file and filling students with the data found there. After that I use a match case loop to run the pre-defined functions based on user choices (1-4).
    - This is shown below.

```python
# Present and Process the data
if __name__ == "__main__":
    while (True):
        IO.output_menu(menu=MENU)
        menu_choice = IO.input_menu_choice()
        match menu_choice:
            # Case 1 enter new student data
            case "1":
                student = IO.input_student_data(student_data=students)
            # Case 2 show current data
            case "2":
                IO.output_student_courses(student_data=students)
            # Case 3 save data
            case "3":
                FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
            # Case 4 exit program
            case "4":
                print("Exiting the program")
                exit()
# End
```

Testing the script

- I ran the script from Visual Studio Code and tested for the following.
    - Takes user input for first name, last name and course name.
    - Program displayed this input.
    - The program saved this to the correct JSON file, this was verified by opening the file and reviewing the contents.
    - I was able to enter multiple registrations.
    - I was able to display multiple registrations.
    - I was able to save multiple registrations to file.

Summary

Using the class demonstrations, my assignment from module 6 and documents provided for module 7 I was able to classes, define properties and functions that take user input, presents it to the user and saves it into a JSON file. I expanded my ability to create a loop, take a user's input, save it to file and add to that file as the user inputs more data. I also learned how to check for certain error conditions and present the error information to the user.