Danielle
Lilleston
Python 100
21 May 2024

Assignment 06

Introduction

This script is identical to assignment 05. This program allows users to register students for courses, view current registrations, and save the data to a json file. However, unlike assignment 05, this script will be utilizing functions.

Step One: Script Header

I can never say this enough, each script should start with a header!

```
# ------------------------------------------------------------------------------------------ 3 1
# Title: Assignment06
# Desc: This assignment demonstrates using functions
# Change Log: (Who, When, What)
#   Danielle Lilleston, 21  May 2024, Created Script
# ------------------------------------------------------------------------------------------ #
```

Figure 1. This is my header for assignment 06.

Step Two: Importing Files, Constants, Variables

This script starts by calling to import an external json file. For constants we only have two, the Menu option that will be utilized later in the script and the file name. This is like assignment 05. For variables, we only have two! These variables are students, which is a list, and menu_choice which is a string. This script has so few variables due to the new use of functions that will now do a lot of the work.

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

Figure 2. This image shows both the call to import an external json file, the constants used, and the defined variables.

Step Three: Processing

We are going to create a new class. This class is considered a blueprint for creating objects or data structure. In this case, we are creating a class that reads and writes data based on the static method. The static method is used to define functions that will perform an operation that is related to the class. In this step, I ask the program to read the data from the file and send out an error message if it cannot read it. Then I ask it to write the data or json string to the file.

```
class FileProcessor:

    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        file = None
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the
        finally:
            if file and not file.closed:
                file.close()
        return student_data
```

Figure 3. This is only part of what was written. Here, it shows the class defined as FileProcessor and then the use of the staticmethod to read data from the file.

Step Four: Presentation

Before the main part of the code that the user will interact with can even run, we have to now set up a collection of functions that will manage the users input and output. In this case, we use five different staticmethods for five different functions. The first creates an error message to the user if an error occurs when trying to run the program. The second is the function that will display the menu choices to the user. The third function takes the menu choice inputted by the user. The fourth function displays the students names and courses they are enrolled in to the user. Finally, the fifth function gets the students name and course that is provided by the user and inputs into a list.

```
@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 first 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="Incorrect data type", 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
```

Figure 4. This is an example of one of the functions that were written. In this function, it is taking the data inputted by the user such as student name and course name, then saving it to a file. It will also output a message stating that the student has been enrolled in the class. There are also error messages utilized in case the user is inputting incorrect data or if there was a problem saving.

Step Five: Processing the Data

Now that the functions are in place, we are ready to set up the main body of the script. This is where the user will be presented with the menu choices and asked to pick one. Then depending on what is picked, they will be inputting data that will then be later saved to a json file.

```
# 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(student_data=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
```

Figure 5. This is just a piece of the main body of the script. Here, it is clearly asking the user to make choices and to input data. This is also where all the data is saved and stored and can be accessed later.

Summary

Functions play a critical role in programming. It allows you to break down the code into smaller sections making it easier to understand and debug. They are also customizable and can be altered depending on the parameters and values. As time goes on, codes will become more and more complexed. Functions will help keep it organized and help the reader understand what the code is doing.