Name: Tien Nguyen

Date: May 22nd, 2024

Course: IT FDN 110A

Github: [tienn1000/IntroToProg-Python-Mod06 (github.com)]

Functions

Introduction

This week, we learn how to add the functions, classes and how to use the separation of concerns pattern. Using functions allow us define a function and call that function whenever we need, therefore our code would be organized and easily managed.

Steps to complete the assignment

Step 1: Read the file "Mod06-Notes" and watch the demonstration videos on the Canvas Module page.

Step 2: Watch reference videos

Step 3: Read about module topics

Step 4: Create a Python script

- Open file in PyCharm
- Create new file named Assignment06.py
- Add the script header and ugrade script
- Start the script by importing json module

```
import json
```

Figure 1: Import JSON module.

- Add the variables and constants

```
# Define the Data Variables and constants
students:list = []
menu_choice: str = ''
```

Figure 2: Data Variables and Constants

- Add the Classes and Functions

Two classes consist of FileProcessor and IO. FileProcessor class includes the bundle of functions in order to read, write and extract the data from file, while IO class plays a role in processing the menu selection including input, output…Using the parameters to perform the functions.

```
class FileProcessor:
    @staticmethod
    def read_data_from_file(file_name:str, student_data:list):
        try:
            file = open(FILE_NAME, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="There was an error 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):
        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 = "There was an error writing the file"
            IO.output_error_messages(message=message,error=e)
        finally:
            if file.closed == False:
                file.close()
class IO:
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(message, end="\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
    @staticmethod
    def output_menu(menu: str):
        print(menu)
    @staticmethod
    def input_menu_choice():
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if choice not in ("1","2","3","4"):
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages(e.__str__())
```

Figure 3: Classes and Function

- Use the loop to display the menu selection

```
while (True):
    IO.output_menu(menu=MENU)
    menu_choice= IO.input_menu_choice()
    if menu_choice == "1":
        students = IO.input_student_data(student_data=students)
        continue

    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")
```

Figure 4: Menu processing using the loop.

Conclusion

In this module, we understand how the functions work and how to group multiple functions into a class to organize the code easily.