Name: Ronnie Sokha

Date: 11/21/2023

Course: Foundations of Python Programming

GitHub Link: https://github.com/RSokha/IntroToProg-Python-Mod05

Assignment 06: Functions

Introduction

This week, we jump into a new topic of functions, classes, and Separation of Concerns. Functions are essentially reusable code we can call whenever we need a certain logic or statement executed. In a way, it acts as a variable… a very complex variable, but not exactly the same. Functions take on parameters and return values – think of a mathematical function; for every input, there is an output. It is important to note that variables inside functions are either local or global; that is, if a variable is defined inside a function, it is local. Similarly, if a variable is defined outside a function such as the main script, it is a global variable. Classes help us organize and group our functions. Creating a class creates a new type of object. Objects can be anything around us such as a chair, dining table, couches, etc. A class would be the home, since this will help us organize all the objects or functions inside the home. Lastly, separation of concerns is a design principle that helps us divide our program into sections to help address each. Ultimately, we see that this week's assignment relies on the idea of programming efficiency (functions for reusability), and organization (classes & separation of concerns). Let's look at some code below, which is a continuation of our assignment that takes a user's input on a student's information as well as the class they are enrolling in.

Utilizing Functions & Classes

Figure 1 – we have our constants and variables. I have commented them out since they are global and I will be using them inside the function; therefore, making them local. This will eliminate the error of 'Shadows name' that PyCharm keeps throwing.

```
 # Define the Data Variables and constants -- but commenting out after
using them inside class to avoid shadowing

MENU = '''---- 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 = "Enrollments.json"


# student_first_name: str = ''  # Holds the first name of a student
entered by the user.
# student_last_name: str = ''  # Holds the last name of a student entered
by the user.
# course_name: str = ''  # Holds the name of a course entered by the user.
# student_data: dict = {}  # one row of student data
# students: list = []  # a table of student data
# csv_data: str = ''  # Holds combined string data separated by a comma.
# json_data: str = ''  # Holds combined string data in a json format.
# file = None  # Holds a reference to an opened file.
# menu_choice: str  # Hold the choice made by the user.
```

Figure 2 – we create our first class FileProcessor and use a function to read and write to the file.

```
class FileProcessor:
    @staticmethod # Function to read the file
    def read_data_from_file(file_name: str, student_data: list):
        try:
            with open(file_name, 'r') as file:
                student_data.extend(json.load(file))
        except FileNotFoundError:
            pass  # File doesn't exist yet; it will be created later
        except Exception as e:
            IO.output_error_messages("Error reading data from file.", e)

    @staticmethod # Function to write/create the file if it doesnt exist
then save to it aka option 3.
    def write_data_to_file(file_name: str, student_data: list):
        try:
            with open(file_name, 'w') as file:
                json.dump(student_data, file, indent=2)
            print(f"Data saved to {file_name}")
        except Exception as e:
            IO.output_error_messages("Error writing data to file.", e)
```

Figure 3 – we create an IO (Input/Output) class to handle user input and return an output such as the user inputting the student's first and last name and the course name they are enrolling the student in. Also, it displays the list of student(s) that are currently enrolled and what course they are enrolled in.

```
# IO Class
class IO:
    @staticmethod # Function to handle output error messages such as
selecting invalid option
    def output_error_messages(message: str, error: Exception = None):
        print(f"Error: {message}")
        if error:
            print(f"Details: {error}")

    @staticmethod # Display menu of options once the code runs
    def output_menu(menu: str):
        print(menu)

    @staticmethod # Function that takes user input
    def input_menu_choice():
        return input("Please select an option from the menu:  ")

    @staticmethod # Function that displays output as a result of option
"2"
    def output_student_courses(student_data: list):
        for student in student_data:
            print(f"Student Name: {student['first_name']}
{student['last_name']}, Course: {student['course']}")

    @staticmethod # Function that displays output as a result of option
"1"
    def input_student_data(student_data: list):
        first_name = input("Enter student's first name: ")
        last_name = input("Enter student's last name: ")
        course_name = input("Enter course name: ")
        student_data.append({'first_name': first_name, 'last_name':
last_name, 'course': course_name})
        print(f'{first_name} {last_name} is now registered for
{course_name}')
```

Figure 4 – Now, we create a new class CourseEnrollments for our main program, and we implement/call our functions in the main program.

```
# Main Body
class CourseEnrollments:
    students = []

    @staticmethod
    def run():
        FileProcessor.read_data_from_file(FILE_NAME,
CourseEnrollments.students)
```

```
        menu_choice = ""
        while menu_choice != "4":
            IO.output_menu(MENU)
            menu_choice = IO.input_menu_choice()

            if menu_choice == "1":
                IO.input_student_data(CourseEnrollments.students)
            elif menu_choice == "2":
                IO.output_student_courses(CourseEnrollments.students)
            elif menu_choice == "3":
                FileProcessor.write_data_to_file(FILE_NAME,
CourseEnrollments.students)
            elif menu_choice == "4":
                print("Exiting the program. Goodbye!")
            else:
                IO.output_error_messages("Invalid choice. Please select a
valid option (1-4).")


if __name__ == "__main__":
    CourseEnrollments.run()
```

Conclusion

I thoroughly enjoyed learning the idea and concepts of functions and classes in Python. It helped clean up a ton of my code therefore, making my logic easier to read. In fact, I was able to eliminate over 60 lines of code (could've been my poor formatting habits), and row reduce much of my logic. As I worked through this assignment, I also kept the notion of Separation of Concerns in mind to help me look at parts/pieces of my program and focus on the blueprint I needed to ensure my script was outputting the correct result.