Name: Alex Kover
Date: 05/22/2024
Course: Introduction to Programming with Python
GitHub Link: https://github.com/AKover-UW/IntroToProg-Python-Mod06

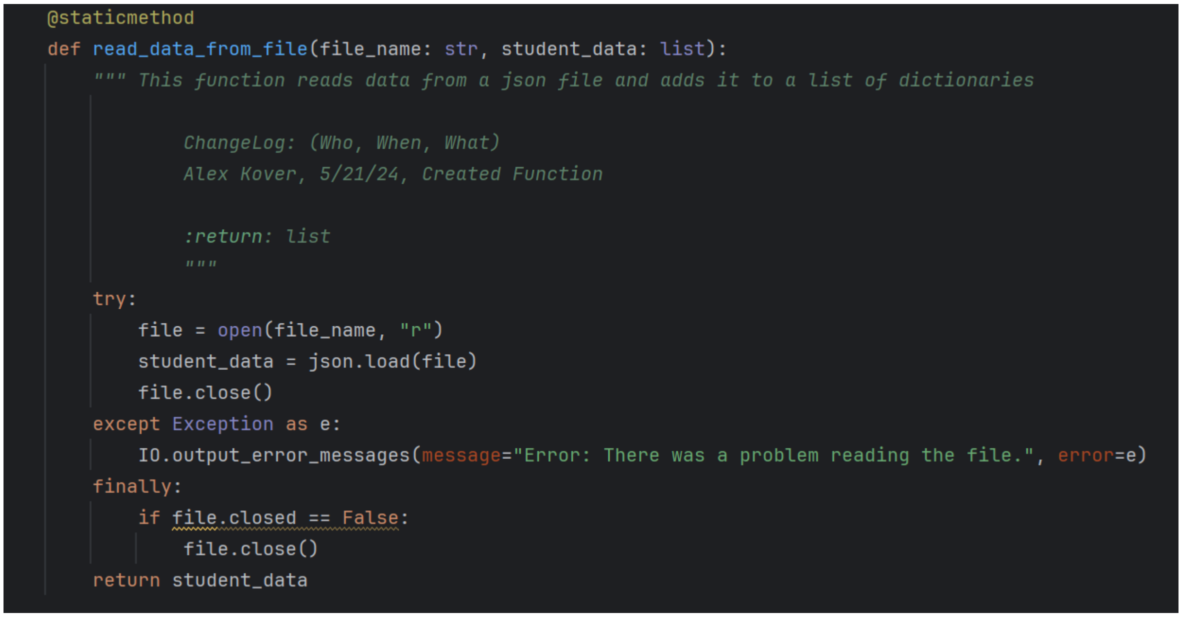
Assignment 06: Functions

Introduction

This document describes the topics and steps taken to complete this assignment. This assignment does the same thing as Assignment 05 but organizes and streamlines how the code is managed by using functions, classes, and separating areas of concern.

Functions

Functions are blocks of code that can perform a certain task or multiple tasks but are noted for their reusability. Functions are an important idea in most programming languages and allow the programmer to break down code into smaller, more manageable parts and allow the programmer to reuse these parts without needing to rewrite the same code again. Functions and their code are “defined” or written and explained near the top of the code before the main body. These functions are then called by the program in the main body. This makes the script much simpler to write as functions can be called multiple times without needing to write the same script again and again. These functions are largely self-contained and can be copied and used in other scripts without any major changes. When a function’s code is “static” meaning that it will not change in the script, a label is added just above it to indicate to the programmer that this function will never change.

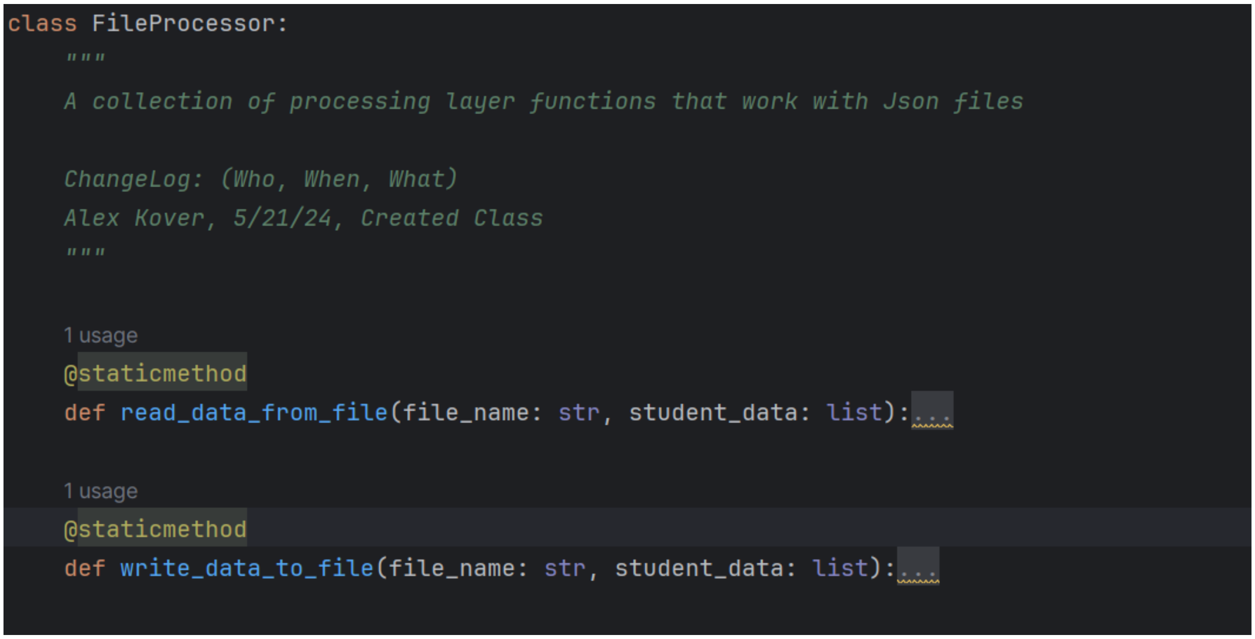


```
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file and adds it to a list of dictionaries

                ChangeLog: (Who, When, What)
                Alex Kover, 5/21/24, Created Function

                :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 reading the file.", error=e)
        finally:
            if file.closed == False:
                file.close()
        return student_data
```

Figure 1: Function used to read data from a json file and add the data into a list of dictionaries. Note that the error handling calls another function instead of having specific error handling code within the function.

Classes

Classes are another topic covered in this module that expands on the organization and order provided by the functions mentioned above. Classes provide the programmer with a way to group functions, variables, and constants depending on their use by the code. Several functions and associated variables and constants can be placed in one class where they will be used instead of globally where they could be accessed or changed by other functions and code. This compartmentalization of code also allows for it to be modular, and the entire class could be copied into a completely different script and still have all the code and variables needed to properly operate.

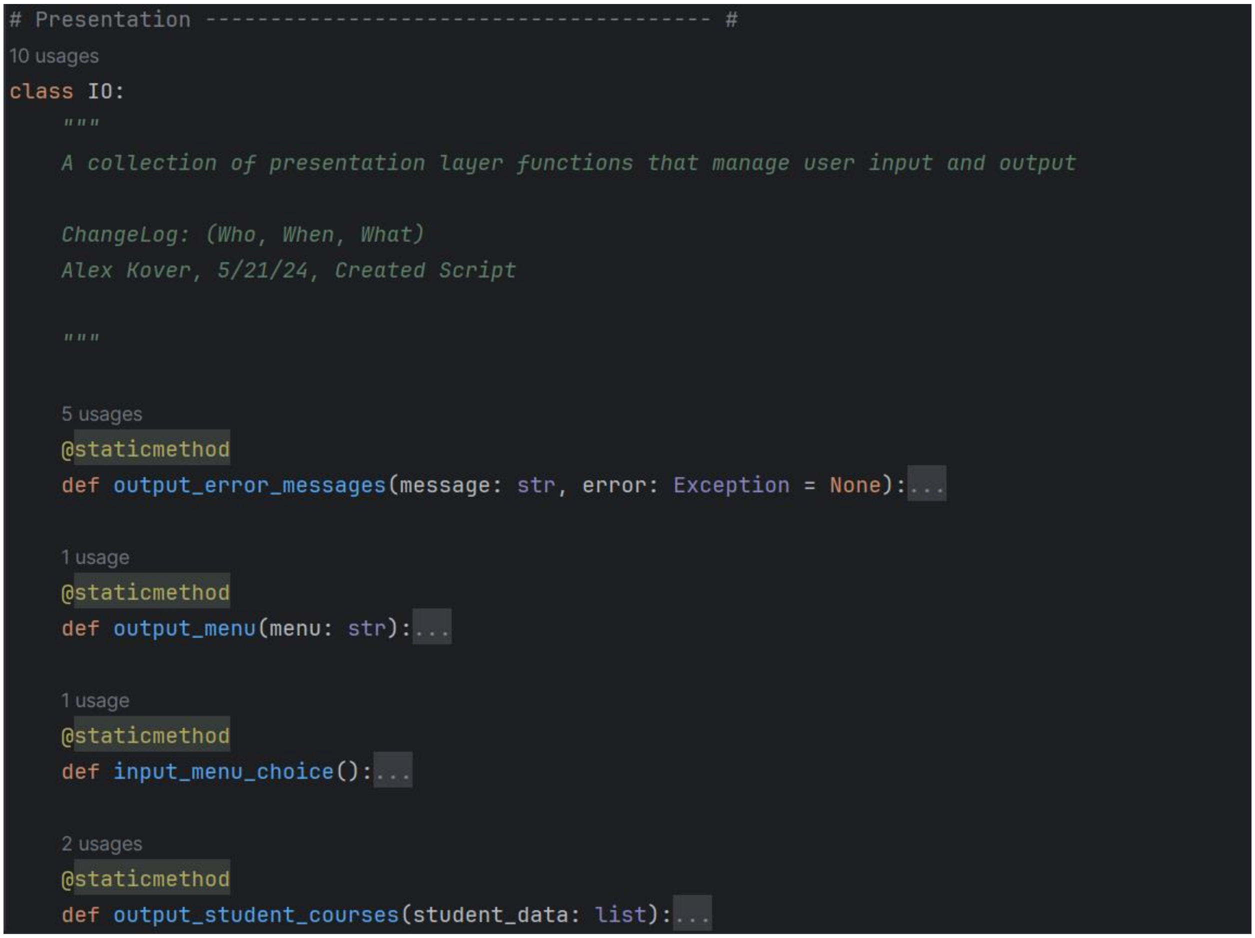


```
class FileProcessor:
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    Alex Kover, 5/21/24, Created Class
    """

    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):...

    1 usage
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):...
```

Figure 2: A class that handles data processing and consists of two functions. This class could be copied into a new script and still retain it's functionality with minimal changes needed.

Separations of Concern

A key topic covered in this module is the Separations of Concern pattern. This is an important idea in programming as it allows the code to be broken down into distinct components that apply to different areas of the overall code's function. Areas of concern refer to specific aspects or responsibilities of the functionality of the code. Some areas of concern are presentation, logic, and data storage. Presentation code typically deals with the Input/Output (I/O) of data by or for a user, logic handles the data processing and represents the core functionality of the code, and data storage handles how the code deals with data related operations such as saving and retrieving data from files or managing variables and constants. By breaking code up into specific areas of concern, it becomes easier to use, understand, and manipulate for programmers.



```
# Presentation -------------------------------------- #
10 usages
class IO:
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    Alex Kover, 5/21/24, Created Script

    """

    5 usages
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):...

    1 usage
    @staticmethod
    def output_menu(menu: str):...

    1 usage
    @staticmethod
    def input_menu_choice():...

    2 usages
    @staticmethod
    def output_student_courses(student_data: list):...
```

Figure 3: Some of the presentation area of concern for this assignment. Note how the section is separated in to a “Presentation” block and the class that handles all of the Input/Output functions is titled “IO”.

Summary

While this program functionally is the same as the previous assignment, it greatly enhances the usability and organization of understanding the code by implementing functions, classes, and areas of concern. These powerful additions make following, editing, and manipulating the code significantly easier for both the author of the program and other programmers who are less familiar with this code.