Nolan Davis

November 22, 2023

IT FDN 110

Assignment 06

https://github.com/ndavis91/IntroToProg-Python-Mod06

Functions and Separation of Concerns

Introduction

This week focused on improving the organization and readability of code using functions and the concept of Separation of Concerns (SoC). These practices also make life easier for programmers working on the program in the future because it makes maintenance easier. By defining the functions once and using calls in the main body, any future changes to the functions only need to be made once.

Creating the Program

To start I ran the starter program to verify it worked as intended. Then I created classes for FileProcessor (See Figure 1) and IO (See Figure 2) to categorize the functions that would be used. Per the assignment I added the comment line to describe the classes. Next I created the functions and copied the code from the program into the function definitions. Where the code previously was, I inserted calls to the function by calling out the class, followed by a period and then the name of the function. The first function I defined was the error handling, because once this was defined I could use it in the other functions to handle the errors and condense the code (See Figure 3).

```
2 usages
class FileProcessor:
    """
    This class processes data by reading from and writing to JSON files.
    """
    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 1 – File Processor class with defined functions to read and write data

```
11 usages
class IO:
    """
    This class presents information to the user and gets input from the user.
    """
    7 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():...

    1 usage
    @staticmethod
    def output_student_courses(student_data: list):...

    1 usage
    @staticmethod
    def input_student_data(student_data: list):...
```

Figure 2 – IO class and defined functions

```
@staticmethod
def output_error_messages(message: str, error: Exception = None):
    """ This function displays a custom error messages to the user

    :return: None
    """

    print(message, end="\n\n")
    if error is not None:
        print("-- Technical Error Message -- ")
        print(error, error.__doc__, type(error), sep='\n')
```

Figure 3 – Defining output_error_messages() function

After defining the function, the corresponding code in the main body of the program was replaced with a function call. I repeated the process for each function, and made sure to add parameters into the function definitions and calls. Once all of the functions were defined and my code was updated, I deleted almost all of the variables that were copied over from the starter file except for the list of students and the menu string. The final result of the main body looks much cleaner and is easier to read (See Figure 4).

```
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# 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":

        # Process the data to create and display a custom message
        IO.output_student_courses(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

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop


print("Program Ended")
```

Figure 4 – Main Body of program using functions

Testing the program

Testing was easy because the functionality of the program was the same as the previous week. I tested each menu option, as well as the error handling by using numbers outside of the option list and putting in numbers for the student names. Once I confirmed it still ran as expected, I repeated the process in the Command Prompt.

Summary

Organizing code using SoC and functions made the code much more readable. For larger programs I think this will be vital, and also serves as a rough draft for writing a program from scratch. The trickiest thing for me was understanding the parameters and arguments that were needed in defining the functions. I think more practice and familiarity will help clarify this in the future.