IB

May 20, 2024

IT FDN 110 A

Assignment 06

https://github.com/uwib25/

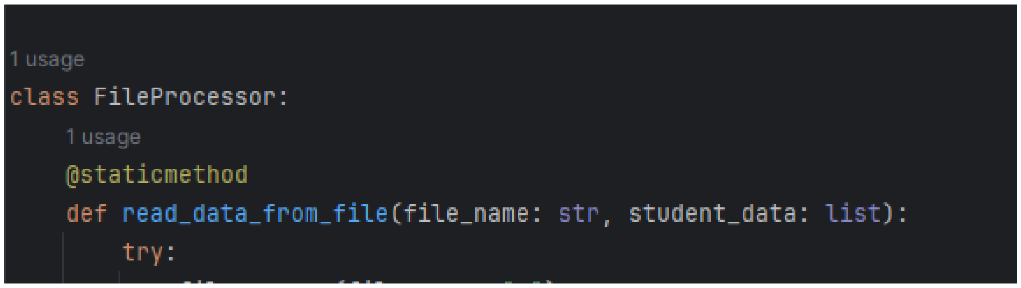
Organizing code through functions

Introduction

In the sixth module, we learned about organizing our code through functions and classes to make our code more modular, and how to create parameters to pass in different values into a function. In addition, we learned how to make our variables local as opposed to global, and how to pass arguments when calling our functions using parameters. Lastly, we learned how to return a value from a function using the ‘return’ statement.

Creating a class

One of the new additions to our acceptance criteria are classes. This allows our functions to be grouped together to make our code even more organized in a logical manner. While I’ll admit that I’m still not fully sure when to use a class to group functions and what ‘self’ is, I believe I’m starting to understand why this is useful, especially after completing the module 3 lab 3.

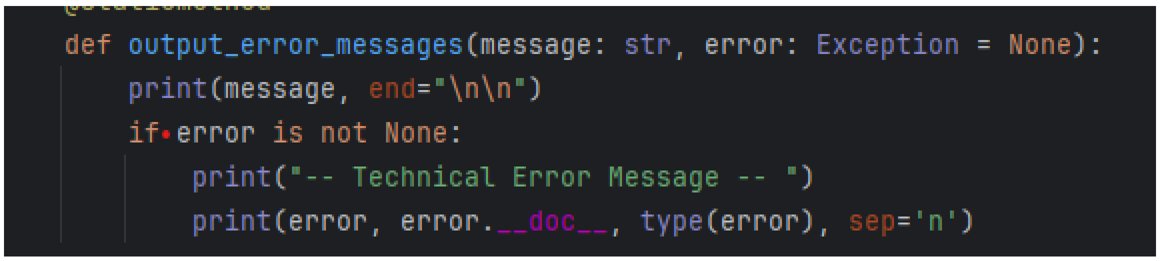


```
1 usage
class FileProcessor:
    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        try:
```

Figure 1: First class

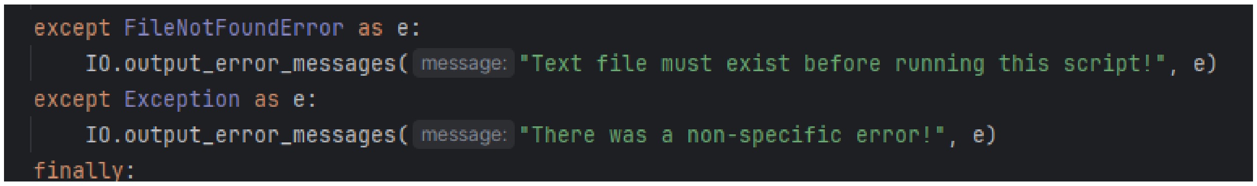
Error Function

While we learned a lot of very important things this week, one of the things that I was most excited to see was the ability to create an error function that can be reused within other functions. I find this very useful and also feel this makes the code look much cleaner when calling the error function, as opposed to creating the same error message many different times for different functions. This really helped represent the power of functions for me.



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

Figure 2: Creating an error function

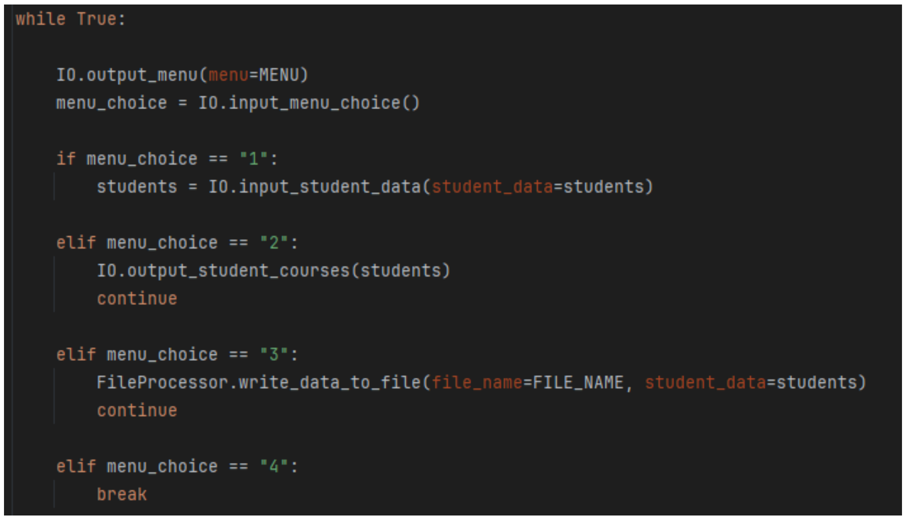


```
except FileNotFoundError as e:
    IO.output_error_messages( message: "Text file must exist before running this script!", e)
except Exception as e:
    IO.output_error_messages( message: "There was a non-specific error!", e)
finally:
```

Figure 3: Using error function from another class

Calling the functions

After creating the functions, we needed to call the function for it to perform the operation. This part took me a while to determine which arguments to use and how it works together with the parameters within the functions. I still have a ways to go before I can say I'm confident that I will structure my code correctly, but I believe this assignment has set me in the right direction!



```
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)

    elif menu_choice == "2":
        IO.output_student_courses(students)
        continue

    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    elif menu_choice == "4":
        break
```

Figure 4: Calling defined functions

Summary

In summary, there was a lot of content packed into this week that are more advanced concepts. The lesson this week helped us learn how to make our code more flexible, readable, and reusable. I believe this knowledge will allow us to write cleaner code, and also make it easier to break down our code to smaller, more manageable sections, and to make it clear what the objective is of that particular block of code. I made sure to perform additional testing of my code this week and believe my program is working as intended. I'm looking forward to hearing the feedback this week!