Ericka Moreno

Nov 20, 2024

IT FDN 110 A

Assignment_06

Functions

Introduction

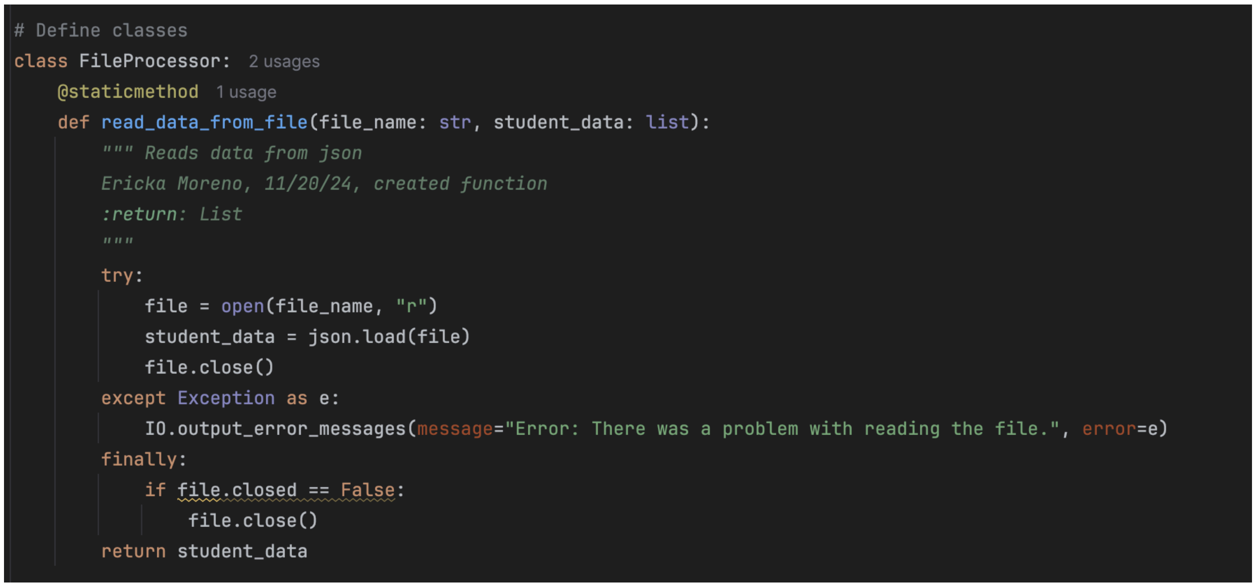
This assignment introduced classes for code organization as well as function parameters. Classes are a great way to organize our code by logic. If functions are related then they can be grouped into a class. Classes go after variable and constant definitions and before our main block of code. Parameters are a way to pass data into a function, they are useful so we do not need to define many global variables.

Organization

This assignment introduced some new code organization concepts and reiterated past concepts to maintain in our code. New concepts include classes and documenting within a class. Reiterated concepts include the change log, commenting code, and keeping constants and variables at the top of the script.

Classes

Classes are used to group related logic. In this assignment two classes were used: FileProcessor and IO (standing for input output). Classes are useful to keep code organized and keep similar functions grouped together. We call these classes within our main code using <className>.<functionName>. Classes keep our code clean and organized. Classes are defined by "class <className>. In this assignment all functions fit under two classes, however, more classes can be used depending on the script and variety of functionality.

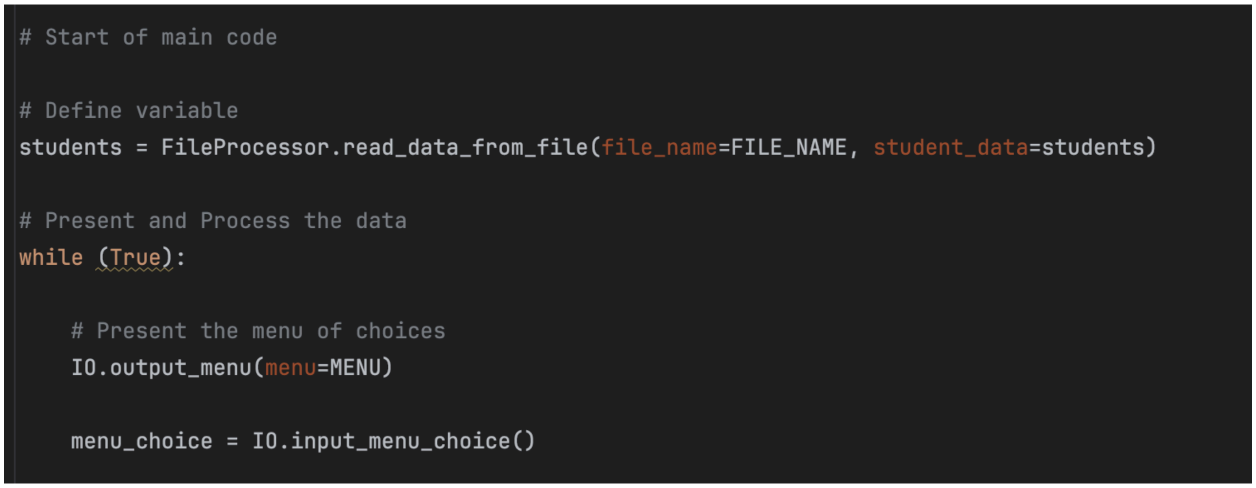


```
# Define classes
class FileProcessor:  2 usages
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """ Reads data from json
        Ericka Moreno, 11/20/24, created function
        :return: List
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
        finally:
            if file.closed == False:
                file.close()
        return student_data
```

Figure 1 Class example

Main code

The main code block is where classes are called on. I wrote a code comment to show where the split between classes and main code is at. In figure 2 you can see how output_menu class is called using IO.output_menu.



```
# Start of main code

# Define variable
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()
```

Figure 2 start of main code after class definitions

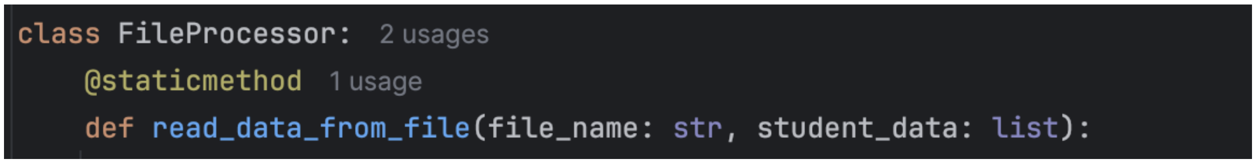
Global variables and Parameters

Global variables

Global variables are noted with “global <variable name>”. These are variables that we want to use multiple places in our code and are not strictly used for one function. When we make a variable in a function, the variable exists only in that function. If we want a variable to exist outside that function and be accessible by other functions, then it needs to be a global variable.

Parameters

Parameters can be used instead of global variables to pass data into functions. They are useful so we don't have to use global variables and instead just pass in data as parameters right into the function. Figure 3 shows an example of a parameter used in assignment_06. File_name and student_data are passed right into the function read_data_from_file as parameters.



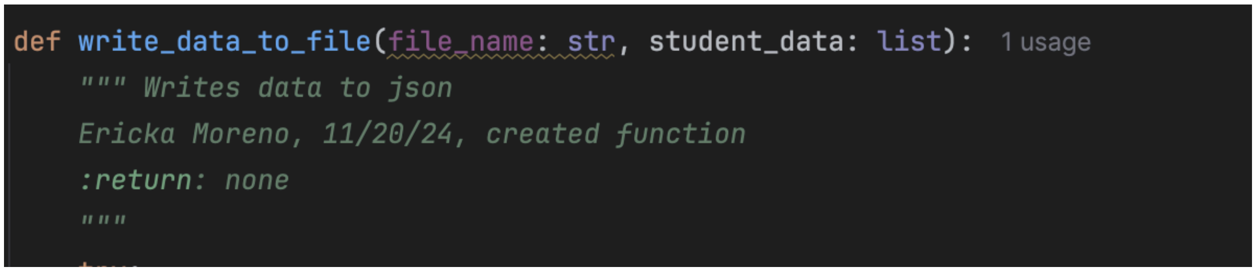
```
class FileProcessor:  2 usages
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
```

Figure 3 Parameter example

Documentation

Class documentation

It is good practice to document class functions. Figure 4 shows an example of how this was used within the assignment. This is similar to our change log at the top of the file, but is listed under each class function. This is helpful for when we go back to an assignment or piece of code to remind us what the code is doing. This is also helpful to future developers who may be reading our code.



```
def write_data_to_file(file_name: str, student_data: list):  1 usage
    """ Writes data to json
    Ericka Moreno, 11/20/24, created function
    :return: none
    """
```

Figure 4 Documentation in class function

Summary

This assignment built on past assignments and added on new concepts such as classes, parameters, and further code documentation. It is important to organize code and keep it simple so that it can be readable and easily understood by future developers that may read it and need to add onto it in the future. Classes help with this because it further breaks down and organizes code by logic. Documenting is also necessary so that functions are described and changes can be tracked.