Akashleena Chaudhuri

21 May 2024

IT FDN 110 B Sp 24: Foundations of Programming: Python

Assignment 06

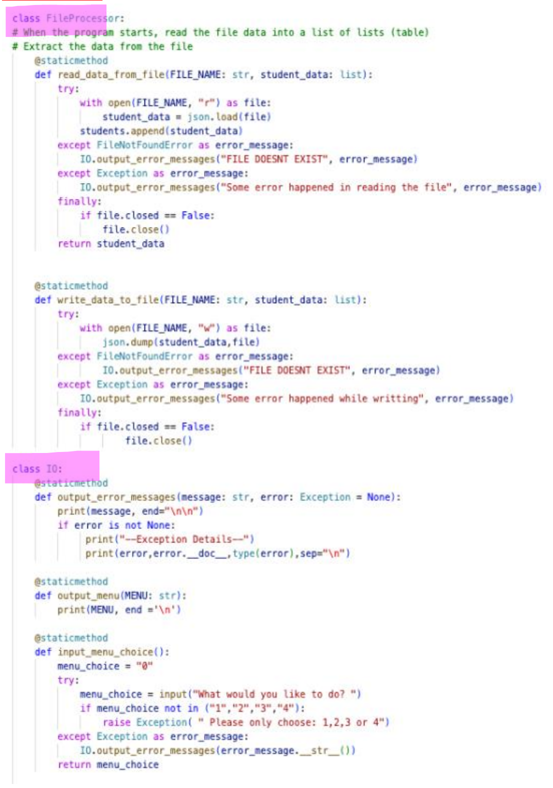
# Functions

## Introduction

This paper will be going through a step-by-step process on how I completed Assignment 06 and my learnings along the way. It focuses on concepts like classes, Functions, JSON files etc

## Part 1

### classes:



```
class FileProcessor:
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
    @staticmethod
    def read_data_from_file(FILE_NAME: str, student_data: list):
        try:
            with open(FILE_NAME, "r") as file:
                student_data = json.load(file)
            students.append(student_data)
        except FileNotFoundError as error_message:
            IO.output_error_messages("FILE DOESNT EXIST", error_message)
        except Exception as error_message:
            IO.output_error_messages("Some error happened in reading the file", error_message)
        finally:
            if file.closed == False:
                file.close()
        return student_data


    @staticmethod
    def write_data_to_file(FILE_NAME: str, student_data: list):
        try:
            with open(FILE_NAME, "w") as file:
                json.dump(student_data,file)
        except FileNotFoundError as error_message:
                IO.output_error_messages("FILE DOESNT EXIST", error_message)
        except Exception as error_message:
            IO.output_error_messages("Some error happened while writting", error_message)
        finally:
            if file.closed == False:
                    file.close()

class IO:
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(message, end="\n\n")
        if error is not None:
            print("--Exception Details--")
            print(error,error.__doc__,type(error),sep="\n")

    @staticmethod
    def output_menu(MENU: str):
        print(MENU, end ='\n')

    @staticmethod
    def input_menu_choice():
        menu_choice = "0"
        try:
            menu_choice = input("What would you like to do? ")
            if menu_choice not in ("1","2","3","4"):
                raise Exception( " Please only choose: 1,2,3 or 4")
        except Exception as error_message:
            IO.output_error_messages(error_message.__str__())
        return menu_choice
```

There are 2 classes in this code:
**class IO:**
IO is for input output
And it has the following definitions:

- output_error_messages(message: str, error: Exception = None)
- output_menu(menu: str)
- input_menu_choice()

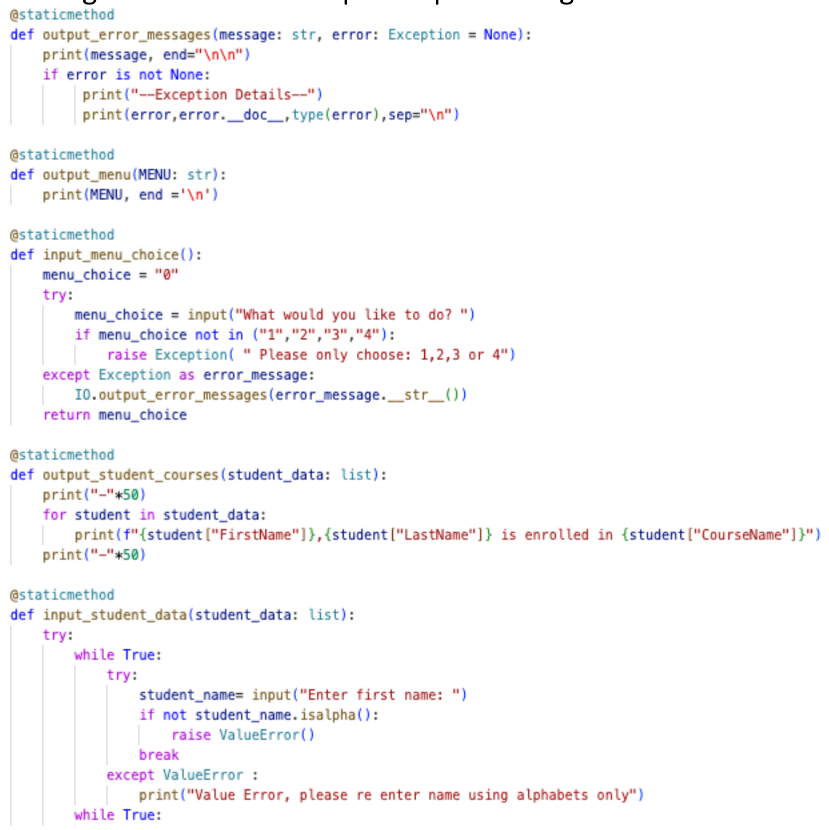
output_student_courses(student_data: list)
input_student_data(student_data: list)
**class FileProcessor:**

- read_data_from_file(file_name: str, student_data: list):
- write_data_to_file(file_name: str, student_data: list):

having 2 separate classes can store functions based on it's uses.
The nomenclature of class is having the first letter be in caps eg:
**class FileProcessor:**

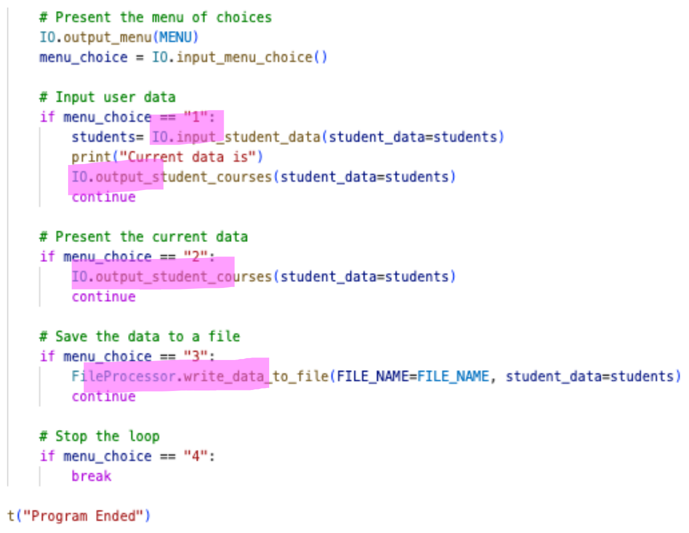
**functions:**

Creating function can be super help for being able to re use it throughout the script.



```
@staticmethod
def output_error_messages(message: str, error: Exception = None):
    print(message, end="\n\n")
    if error is not None:
        print("--Exception Details--")
        print(error,error.__doc__,type(error),sep="\n")

@staticmethod
def output_menu(MENU: str):
    print(MENU, end ='\n')

@staticmethod
def input_menu_choice():
    menu_choice = "0"
    try:
        menu_choice = input("What would you like to do? ")
        if menu_choice not in ("1","2","3","4"):
            raise Exception( " Please only choose: 1,2,3 or 4")
    except Exception as error_message:
        IO.output_error_messages(error_message.__str__())
    return menu_choice

@staticmethod
def output_student_courses(student_data: list):
    print("-"*50)
    for student in student_data:
        print(f"{student["FirstName"]},{student["LastName"]} is enrolled in {student["CourseName"]}")
    print("-"*50)

@staticmethod
def input_student_data(student_data: list):
    try:
        while True:
            try:
                student_name= input("Enter first name: ")
                if not student_name.isalpha():
                    raise ValueError()
                break
            except ValueError :
                print("Value Error, please re enter name using alphabets only")
        while True:
```

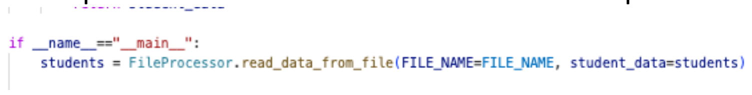
For eg: they are listed at the start of the script then called in the main body:



```
    # Present the menu of choices
    IO.output_menu(MENU)
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":
        students= IO.input_student_data(student_data=students)
        print("Current data is")
        IO.output_student_courses(student_data=students)
        continue

    # Present the current data
    if menu_choice == "2":
        IO.output_student_courses(student_data=students)
        continue

    # Save the data to a file
    if menu_choice == "3":
        FileProcessor.write_data_to_file(FILE_NAME=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    if menu_choice == "4":
        break

t("Program Ended")
```

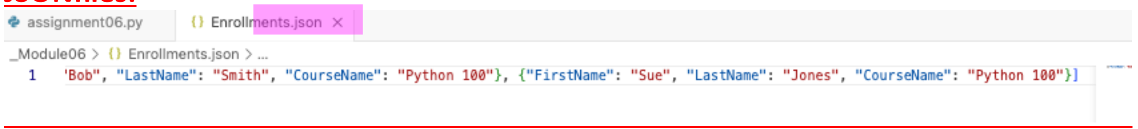
## Main:

This helps the user understand which is the main part of the script



```
if __name__=="__main__":
    students = FileProcessor.read_data_from_file(FILE_NAME=FILE_NAME, student_data=students)
```

## JSONfiles:



assignment06.py   {} Enrollments.json ×

_Module06 > {} Enrollments.json > ...

```
'Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"}]
```

JSON files retain the readability of a script, dictionary, list – we can see that with the color coding as opposed to a txt file.
These files can be easily read and segregated!

**Summary**

Creating classes and functions help create codes that can easily be extended onto.
It was a bit confusing to get used to the class syntax but more practice will be helpful!