Akashleena Chaudhuri

31 May 2024

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

Assignment 07

Classes and Objects

Introduction

This paper will talk about my understanding of classes and issues I face while writing this assignment.

Part 1

classes:
There are 4 classes in this code:
2 from assignment 06-
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):

2 New ones:

- **class Student**
- **class Person**

Person is a parent class for Student which is why it is written like this:

```
class Student(Person):
```

It is referencing

There are 2 decorators in this assignment i.e:

- @staticmethod
- @property

This allows one to get attributes like Getter, setter etc.

Getter: The method with the @property decorator is used to get the value of the attribute.

Setter: The method with the @property decorator is used to set the value of the attribute.
Eg:

```
@course_name.setter
    def course_name(self, value:str):
        self.__course_name = value
```

Method:

Some new ones in this assignment were:
__Init__

```
class Person:
    def __init__(self, first_name: str = '',last_name: str=''):
        self.first_name= first_name
        self.last_name=last_name
```

It takes in a parameter **self**

Some formatting differences from A06 to A07:

A07:

```
@first_name.setter
    def first_name(self, value:str):
        if value.isalpha() or value == "":
            self.__first_name = value
        else:
            raise ValueError ("No numbers please")
```

A06:

```
@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:
            try:
                student_last_name= input("Enter last name: ")
                if not student_last_name.isalpha():
                    raise ValueError()
                break
            except ValueError :
                print("Value Error, please re enter name using alphabets only")
```

in A07, we could assign. isalpha in the getter instead of putting in the if statement.

Challenges:

The JSON file was a list of dictionaries that I forgot to convert to class object in my 'read data from file' method
Which made my script run into errors

```
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads data from a json file and loads it into a list of


    try:
        file = open(file_name, "r")
        student_data_old = json.load(file)
        for item in student_data_old:
            student = Student(item["FirstName"], item["LastName"], item["CourseName"])
            student_data.append(student)
        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
```

This was the hardest part of the assignment as I couldn't figure out this was missing, and I kept getting tracebacks

Summary

Biggest learning in this assignment was creating print statements and printing 'types' to understand when what is a list/ dictionary/ class objects!

Having a clear undemanding of when each of this converts is vital to get a error free script.