**Name**: Varun Reddy Param

**Date:** Nov 26th 2024

**Course:** Foundations Of Programming: Python

**Assignment:** (Assignment 07) - Create a Python program that uses constants, variables, and print statements to display a student's registration for a Python course, incorporating data processing classes, presentation layer Classes, Functions and data classes with attributes.

**GitHub**:

.

# Learning usage on dictionaries, Json files, and exception handling

Introduction:  
In this assignment, I created a Python program that manages student registration for a course. The program incorporates concepts such as JSON file handling, dictionaries, lists, functions, classes, error handling, and separation of concerns.

## Functions, Classes, and Separation of Concerns

This assignment builds upon Assignment 05 by adding the use of functions, classes, and a separation of concerns pattern to improve code structure

### Classes

The program includes two classes: **FileProcessor** and **IO**.

* **FileProcessor**: Manages file-related operations, such as reading and writing student data to the JSON file.
* **IO**: Handles input and output operations, including displaying menus and capturing user input.
* **Person and Student**: Data class that are used to represent data entities and has attributes with getter and setters

The use of classes helps in logically grouping related functionalities which makes the code more structured and giving specific responsibility to deal with  
  
Functions

The program uses multiple functions within the classes to handle specific tasks, such as input handling, displaying data, and reading/writing JSON files. Which means each function has a specific set of instruction based on the parameters provided which either returns something or void

* **input\_student\_data()**: Captures student details from the user.
* **output\_student\_courses()**: Displays the current list of students.
* **read\_data\_from\_file()** and **write\_data\_to\_file()**: Handle file operations.
* **output\_menu()**: Displays the menu options for the user.
* **input\_menu\_choice()**: Gets the menu choice from the user.
* **output\_error\_messages()**: Displays custom error messages to the user.

Constructor

A constructor is a special method (*\_\_init\_\_*) in a class that initializes new objects. It sets initial values for attributes when an object is created.

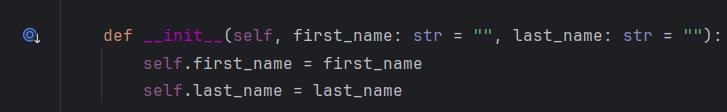


Figure 1: Initialize constructor

This constructor initializes Student objects with default or provided values.

Attribute

An attribute is a variable that belongs to a class and holds data about an object. Attributes are accessed using dot notation

In the Student class, *first\_name*, *last\_name*, and *course\_name* are attributes.

Property

A property is a special kind of attribute that includes getter, setter, and deleter methods. It provides controlled access to class attributes and allows for validation or computed values.

Class Inheritance

Inheritance allows a class (child) to acquire the properties and methods of another class (parent). This promotes code reuse and hierarchical relationships.

The *Student* class inherits from the *Person* class, reusing *Person* attributes and methods while adding *course\_name* as a unique property.

Overridden Method

Method overriding occurs when a child class provides a specific implementation of a method already defined in its parent class

The *Student* class overrides the *\_\_str\_\_* method of the *Person* class to include additional details:

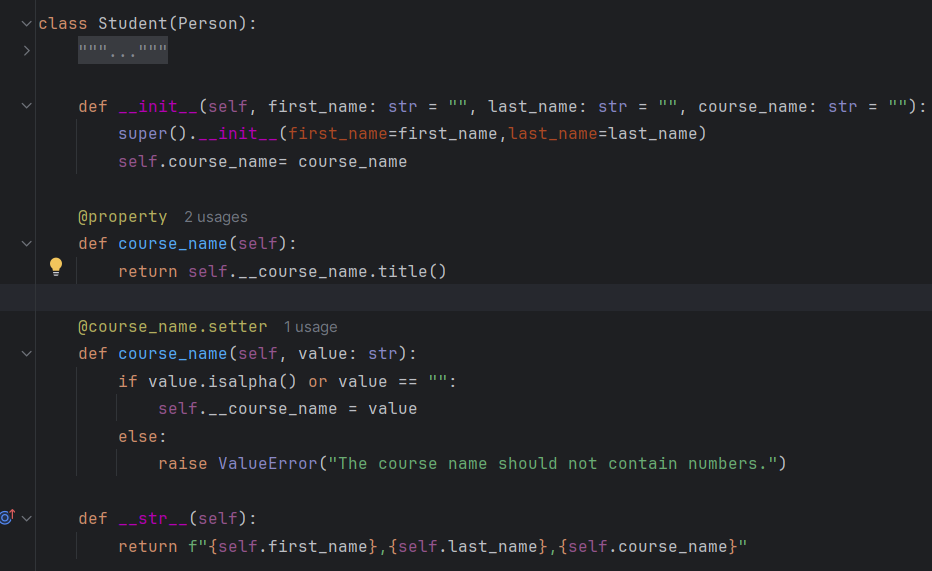


Figure 2: Student Class , Constructor Attributes, Properties and Class Inheritance

## OOPS Concepts

**Encapsulation**

* Bundling data and methods that operate on the data within a class.
* **Example**: The Person class encapsulates attributes like *first\_name* and *last\_name* along with validation logic via properties.

**2. Inheritance**

* Reusing the functionality of a parent class in a child class.
* **Example**: The *Student* class inherits from the Person class, adding new functionality with *course\_name*.

**3. Polymorphism**

* The ability to use a method in different ways depending on the context.
* **Example**: The *\_\_str\_\_* method is overridden in the *Student* class to display extended information specific to students.

**4. Abstraction**

* Hiding implementation details and showing only the necessary features.
* **Example**: The *FileProcessor* class abstracts file handling logic, so other parts of the program can simply call its methods without worrying about implementation details.

A screenshot of a computer program

Description automatically generated

Figure 3: Py charm IDE Execution

A screenshot of a computer

Description automatically generated

Figure 4: Executed using Idle

# Summary

In this assignment, I explored the use of variables, loops, properties, and error handling in Python. The menu-driven program effectively allows users to register students, display the current registration data, and save it to a JSON file. The use of classes for encapsulation, inheritance, and abstraction, combined with functions and separation of concerns