**Name**: Varun Reddy Param

**Date:** Nov 12th 2024

**Course:** Foundations Of Programming: Python

**Assignment:** (Assignment 05) - Create a Python program that uses constants, variables, and print statements to display a student's registration for a Python course, incorporating data processing using dictionaries and exception handling.

**GitHub**: <https://github.com/varunreddyparam/IntroToProg-Python-Mod05>

.

# 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.

# Declaring Data Constants

* **Menu Constant**: Displays the menu options for the user to choose from.
* **File Name Constant**: Defines the file name for saving student enrollments.
* Constants like MENU and FILE\_NAME are used to store values that do not change during the execution of the program.

A screenshot of a computer program

Description automatically generated

Figure 1: Declaring Constants with data Types

# Declaring Variables in Python

Store data that changes as the program runs.

* *file* is used for read and write data to Json file
* *menu\_choice* stores the user's choice from the menu
* *student\_table* is a list that holds data for list of dictionary Objects.



Figure 2: Declaring Variables

Collections: Using Lists and Dictionaries to Store Data

The program uses both single-dimensional and two-dimensional lists to manage student enrollment data effectively:

* Single-Dimensional Distionary (*student\_data)* Holds the information for a single student, including first name, last name, and course name.
* Two-Dimensional List (students), A collection of all *student\_data* entries, making it a list of dictionaries where each element represents an individual student's record.

### Properties of Lists

* **Ordered**: Lists maintain the order of items, meaning that items are stored and retrieved in the same sequence.
* **Mutable**: Lists can be modified after their creation. This allows adding, updating, or deleting elements easily.
* **Appending Data**: The program uses students.append(student\_data) to add new student records to the list. This method ensures that new data is added to the end of the list.

Using lists as collections helps in easily managing multiple records and performing operations such as adding, displaying, and saving data.

Properties of Dictionary

* **Key-Value Pair Structure**: Dictionaries store data in key-value pairs, making it easy to access values using their corresponding keys. For example*, student\_data* has keys like FirstName, LastName, and CourseName which map to the student's details.
* **Unordered**: Unlike lists, dictionaries do not maintain the order of items. The data is stored in a way that allows for fast access, regardless of the order in which items were added.
* **Mutable**: Dictionaries are mutable, meaning that they can be modified after their creation. This allows for adding, updating, or deleting key-value pairs as needed.
* **Adding and Updating Data**: The program uses statements like *student\_data['FirstName'] = student\_first\_name* to add or update data in the dictionary. This flexibility makes dictionaries ideal for managing structured data.

# Json File handling:

The program uses JSON to save student records and reload them when the program restarts.

* The program reads data from *Enrollments.json* at startup to populate the students list.
* The *json.load*(file) function is used to parse the JSON file into a list of dictionaries. Error handling is implemented to manage scenarios where the file might not exist.

A screenshot of a computer program

Description automatically generated

Figure 3: reading Json file to students List

### Writing Data to JSON:

* The *json.dump(students, file)* function is used to save the students list to Enrollments.json in JSON format. The file is then properly closed to ensure all data is saved and the file is no longer in use.

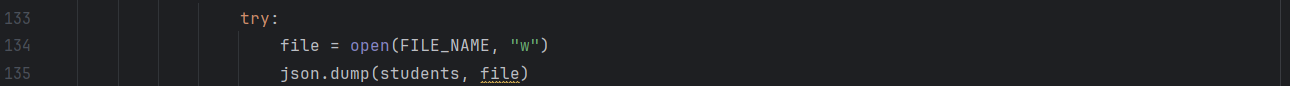


Figure 4: Writing Data to Json

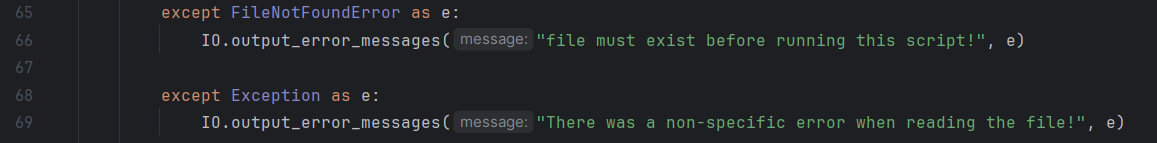
# Exception Handling and errors

Error handling is important because it helps make programs stable and easy to use. It makes sure that the program can deal with unexpected problems without crashing.

* Error handling is used to prevent program crashes when unforeseen errors occur, allowing the program to continue running or terminate gracefully.
* It provides meaningful feedback to the user, helping them understand what went wrong and how to fix it.
* It also helps developers identify and debug issues effectively.

### File Handling Errors:

* When working with files, there is always a possibility that the file might not exist, be corrupted, or be inaccessible. In such cases, a *FileNotFoundError* or other *IOError* might be raised.
* In this program, when trying to read from Enrollments.json, if the file does not exist, a *FileNotFoundError* is caught, and an appropriate message is displayed to the user.



A screen shot of a computer program

Description automatically generated

Figure 5 File Handling error and generic exception

### Generic Exception Handling:

* In situations where an unexpected error occurs that is not specifically handled (e.g., a value error, permission error, etc.), a generic except Exception as *e block* can be used to catch all other exceptions.
* This makes that the program provides a generic error message instead of crashing without explanation.

### Input Validation:

* When capturing user input, it's important to ensure that the data is valid. For example, an empty first name or last name is not acceptable. To enforce this, *a ValueError* is raised if the input is invalid.
* Raising custom errors helps maintain data integrity and ensures that only correct and expected data is processed.
* Gained some insights around type of exceptions we can use ( <https://docs.python.org/3.13/library/exceptions.html#ValueError> )
* The program prompts the user to enter the student's first name, last name, and the course name. These are stored in the respective variables.
* Collecting and storing this data allows for later use in displaying and saving the registration details.

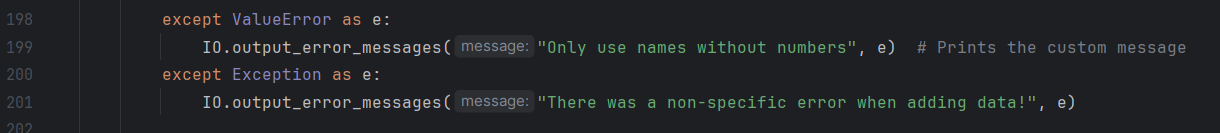


Figure 6 Input validation value error

## 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.

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.

A screenshot of a computer

Description automatically generated

Figure 7: Py charm IDE Execution

A screenshot of a computer

Description automatically generated

Figure 8: Executed using Idle

# Summary

In this assignment, I explored using constants, variables, loops, match-case statements, file handling, 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 functions, classes, and separation of concerns