MTech CSE 2024-26 Batch

LAB 1 ASSIGNMENT

|  |  |
| --- | --- |
| Course name | : ADVANCED ALGORITHMS LAB |
| Course code | : CS531 |
| Student name | : GAURAV SINGH RAJPUT |
| Roll number | : 240201**007** |

*Question 1: Command line argument in python programming.*

The arguments that are given after the name of the program in the command line shell of the operating system are known as **Command Line Arguments**. Python provides various ways of dealing with these types of arguments. The three most common are:

* Using sys.argv
* Using getopt module
* Using argparse module



**Using `sys.argv`**

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.  
One such variable is sys.argv which is a simple list structure. It’s main purpose are:

* It is a list of command line arguments.
* len(sys.argv) provides the number of command line arguments.

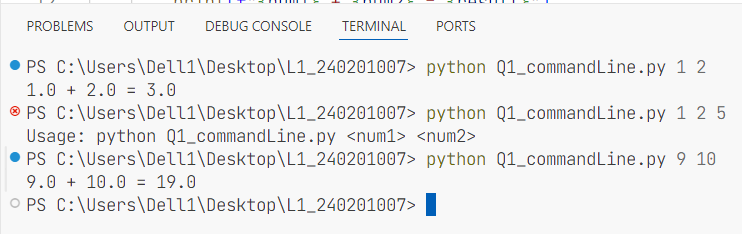
**How `sys.argv` works:**

* `sys.argv` is a list in Python that contains the command-line arguments passed to the script.
* `sys.argv[0]` is the name of the script.
* `sys.argv[1]` is the first number (num1).
* `sys.argv[2]` is the second number (num2).

**Control Flow:**

* The script checks if exactly two command-line arguments are provided. If not, it prints a usage message and exits.
* It then tries to convert the arguments to floating-point numbers and adds them together.
* If the arguments are not valid numbers, the script catches the `ValueError` and prints following error message: Usage: python Q1\_commandLine.py <num1> <num2>.

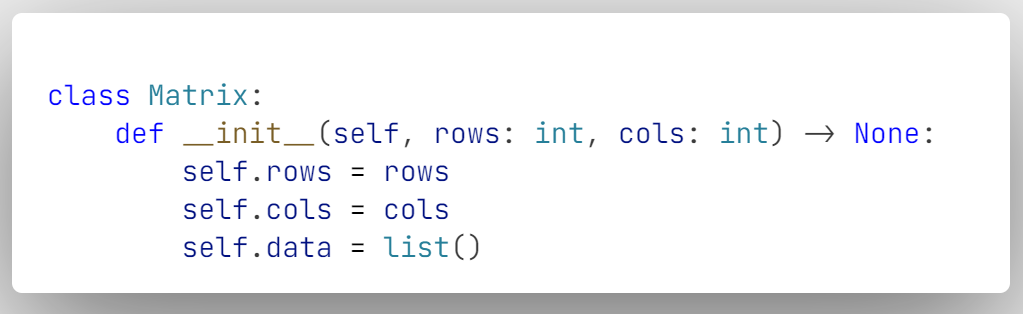
Running the Program:



*Question 2: Take input of the size of a matrix using command line argument and perform matrix multiplication.*

`**Class Matrix**`:

The Matrix class is designed to represent a mathematical matrix and perform operations such as input, output, multiplication and provides a string representation of the matrix.



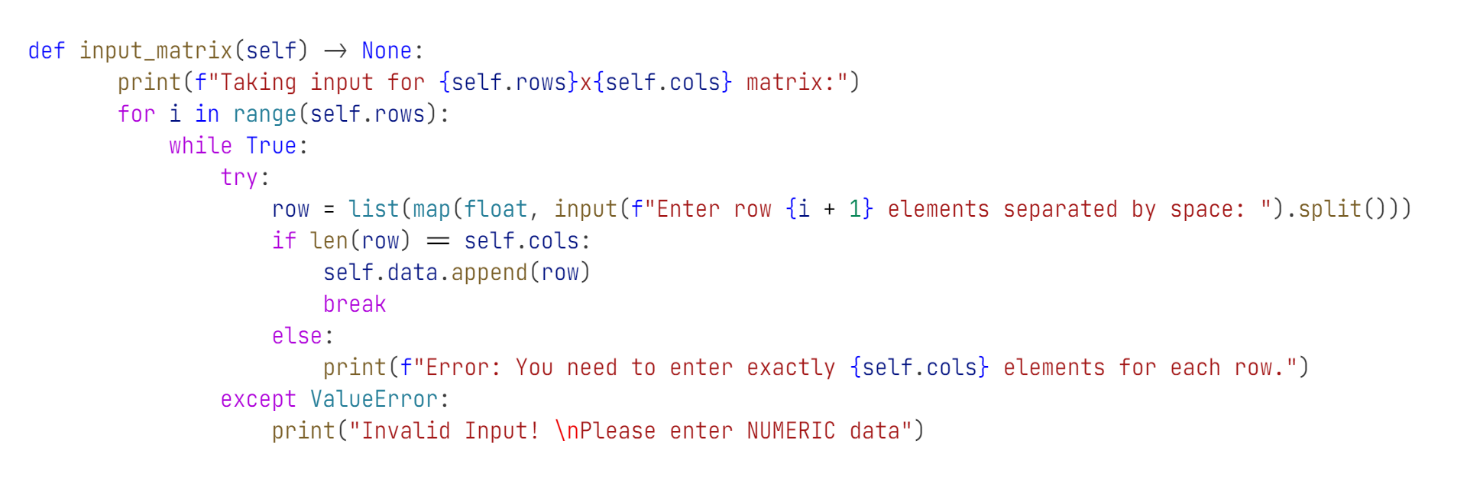
**Attributes:**

* rows: Number of rows in the matrix.
* cols: Number of columns in the matrix.
* data: A list of lists representing the matrix elements.

**Initialization:**

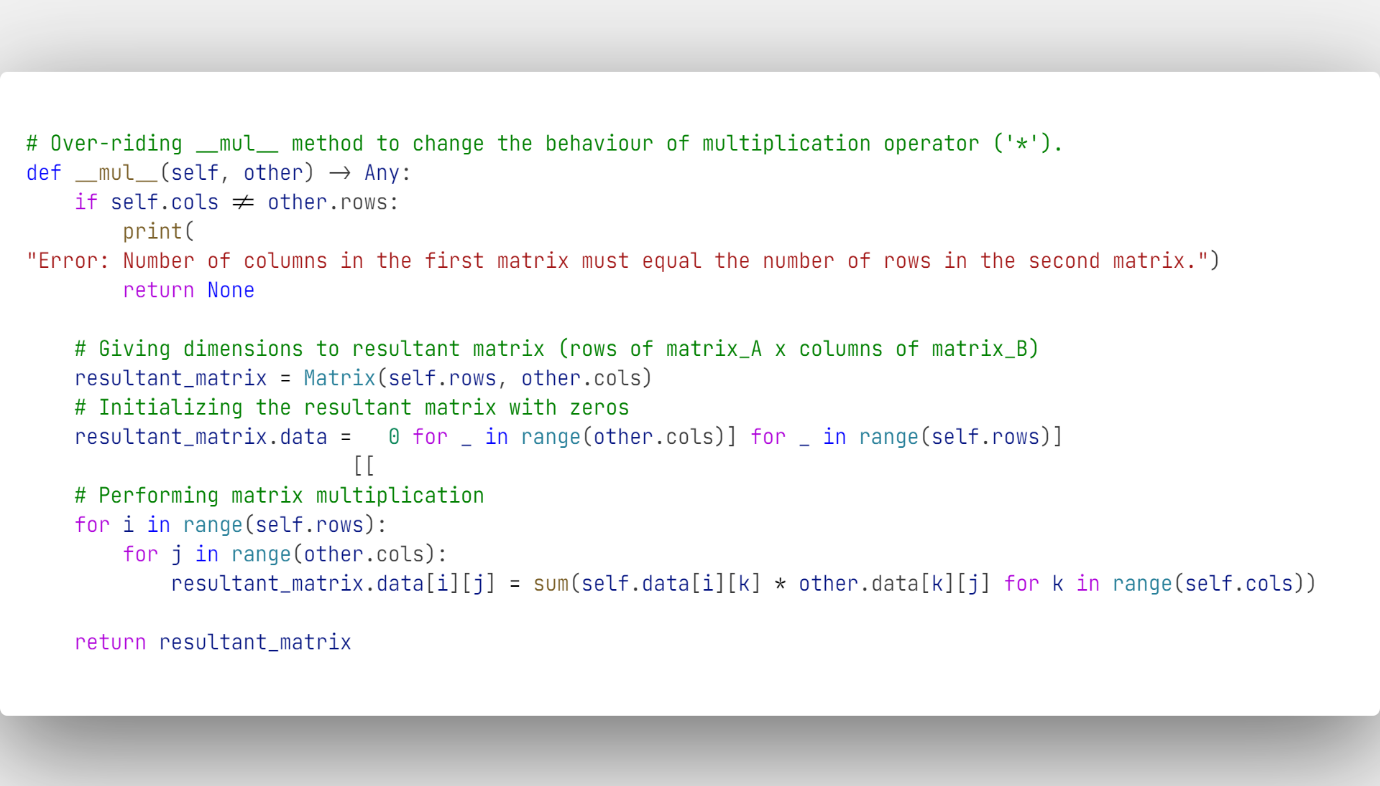
* The \_\_init\_\_ method is the constructor, which initializes the matrix's dimensions (rows and cols).
* Initializes self.data as an empty list, which will later hold the matrix elements.

`**input\_matrix**` method:

****

`input\_matrix` method takes user input to populate the matrix row by row and validates that the user inputs exactly the number of elements expected for each row, and ensures that the input is numeric.

`**\_\_mul\_\_**` method:

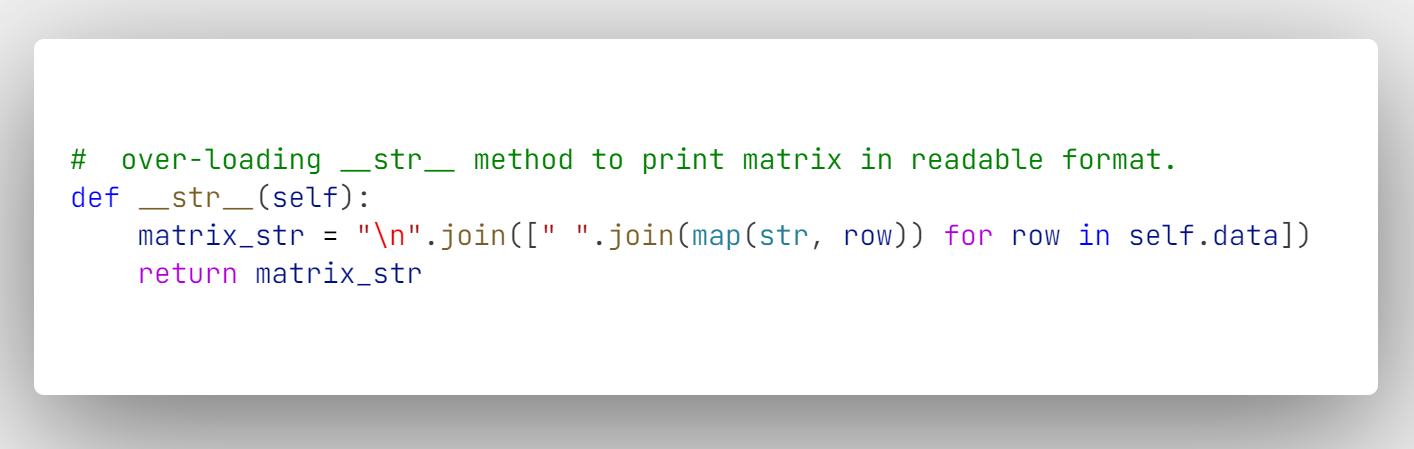


The \_\_mul\_\_ is a special method in python which is used to implement the multiplication operation for instances of a class. When you use the `\*` operator between two objects, Python internally calls the \_\_mul\_\_ method of the left-hand operand (the object on the left of the `\*` operator).

**Purpose:**

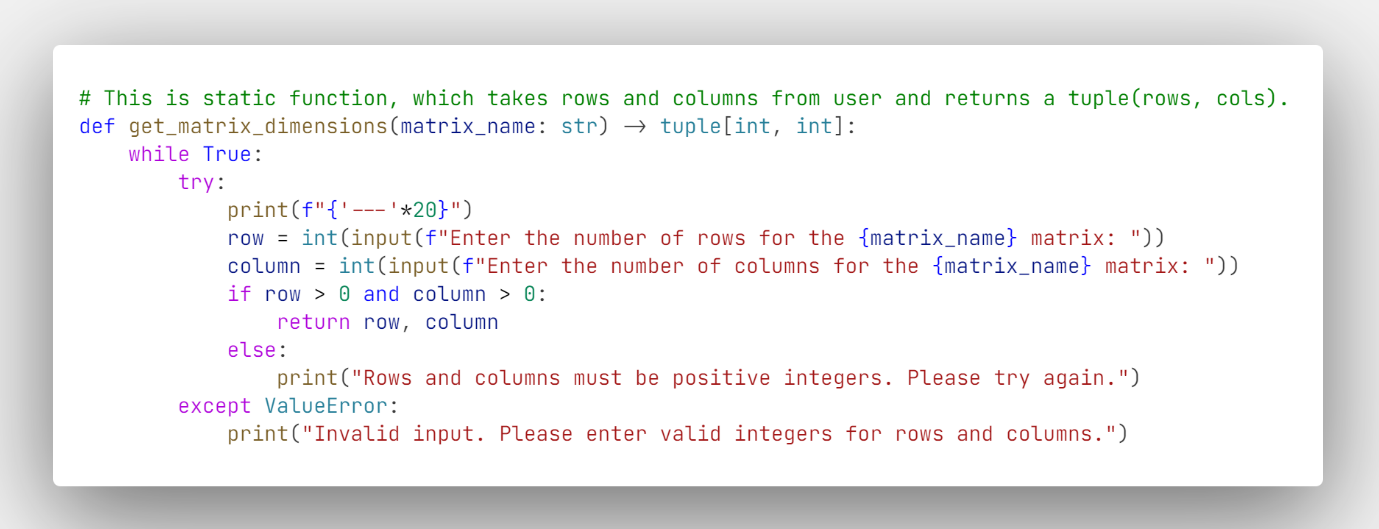
* Overloads the \* operator to perform matrix multiplication.
* Validates that the matrices can be multiplied (i.e., the number of columns in the first matrix matches the number of rows in the second matrix).
* Creates a new Matrix object `resultant\_matrix` to store the result of the multiplication.
* Performs the matrix multiplication using nested loops and the dot product formula.
* Returns a Matrix object that contains the product of the two matrices.

**`\_\_str\_\_`** Method:



* When we print an object, python internally calls `\_\_str\_\_` method to determine the string representation of that object.
* Here, the `\_\_str\_\_()` method is overloaded to provide a readable string representation of the matrix.
* Returns a string where each row of the matrix is formatted as a line of space-separated values.

**Handling Input for Rows and Columns:**



* The `get\_matrix\_dimensions(matrix\_name)` is a static function which is used to get the dimensions of matrix, it repeatedly prompts the user for valid input until they enter positive integers for rows and columns.
* If the input is invalid (e.g., non-integer or negative/zero values), the user is asked to try again without exiting the program.
* It returns a tuple (rows, columns) which contains the dimensions of the matrix.

*Question 3: Use structure to display student details and also search any student details. Parameters are stud\_names, stud\_roll, stud\_branch.*

Control flow of the code.

**1. Program Start (\_\_main\_\_ Block):**

* **Line:`** if \_\_name\_\_ == "\_\_main\_\_":`
  + The program execution starts here.
  + It checks if the script is being run as the main program (not imported as a module). If true, it proceeds with the following steps.
* **Line:** `db = StudentDatabase()`
  + An instance of `StudentDatabase` is created, initializing an empty list `student\_data` to store student records.
* **Line:** `db.menu()`
  + The `menu()` method of the `StudentDatabase` instance is called, which presents the user with a menu of options

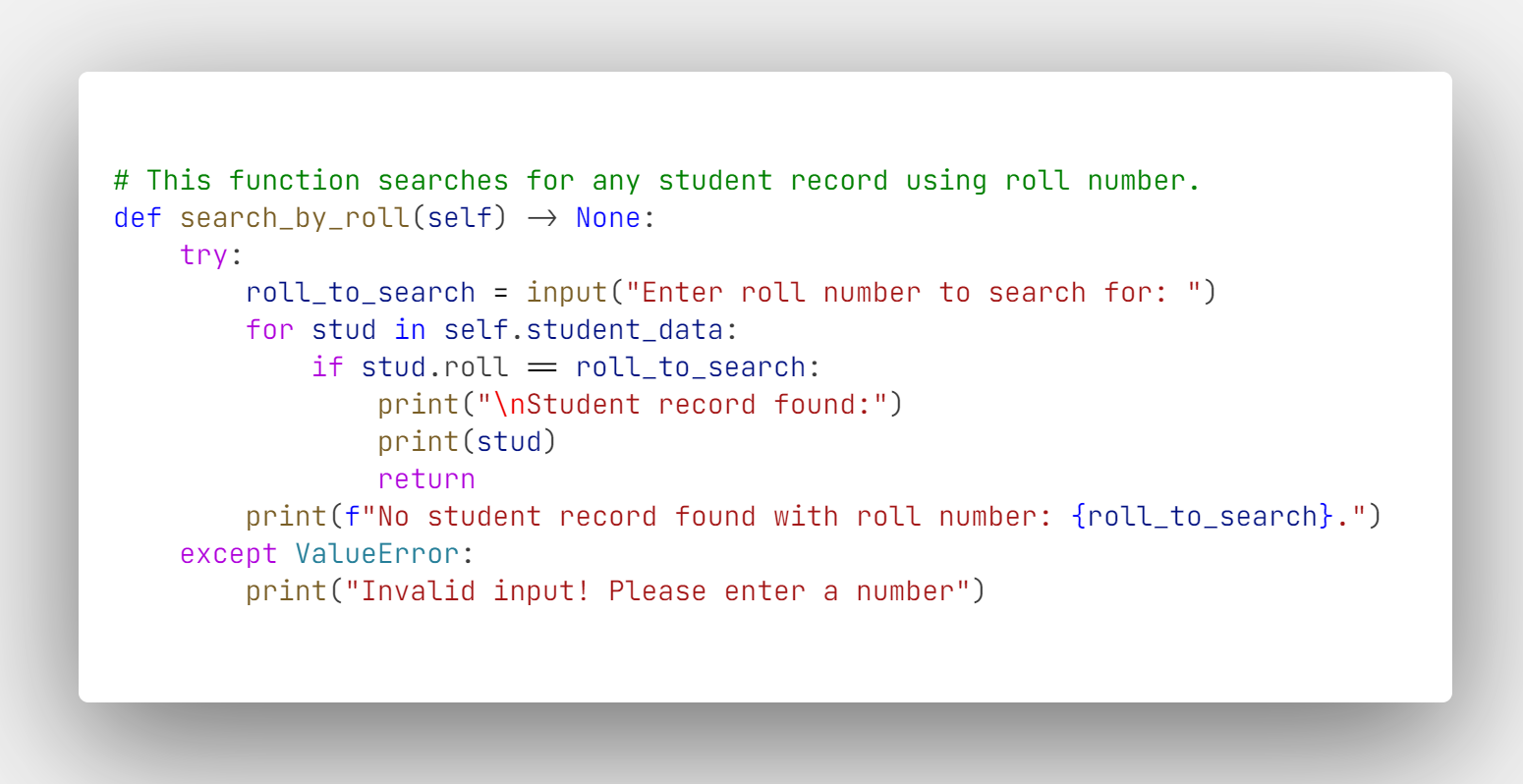
**2. Entering the menu() Method:**

* The program enters a loop that repeatedly displays a menu until the user decides to exit.
* **Line:** choice = input("Enter your choice (1-5): ")
  + The program prompts the user to choose an option by entering a number between 1 and 5.
    - Option 1: Add One Student Record
    - Option 2: Add Multiple Student Records
    - Option 3: Search Student via Roll Number
    - Option 4: Display All Student Records
    - Option 5: Exit
* User will select Option 1 at the beginning of the script, as this script performs various operations stated above on data provided during runtime.

**3. Handling User's Menu Choice:**

Based on the user's input (choice), the program executes different parts of the code:

* **Option 1: Add One Student Record**
  + **Line:** if `choice == '1':`
  + Calls `self.add\_one\_student()` method.
  + **Within `add\_one\_student()`:**
    - The program prompts the user for the student's name, roll number, and branch.
    - Calls the static method `choose\_branch()` to allow the user to select a branch.
    - A new Student object is created and added to the `student\_data` list.
    - Prints a success message and returns to the menu.
* Similarly, other options can be exercised by the user. Searching for any student record is implemented via roll number of student as it is unique for everyone.
* **Option 3: Search Student via Roll Number**
* **Line:** `elif choice == '3':`
* Calls `self.search\_by\_roll()` method.
* **Within search\_by\_roll():**
  + The program prompts the user to enter the roll number to search for.
  + Iterates through the student\_data list to find a student with the matching roll number.
  + If found, it prints the student’s details. If not, it prints a "not found" message.
  + Returns to the menu.



**4. Asking User to Continue or Exit:**

* After processing any valid option (1-4), the program asks the user whether they want to continue or exit.
* **Line:** `continue\_choice = input("<Press 'y' to continue & 'n' to exit> :").strip().lower()`
  + If the user inputs 'y' or 'yes', the menu loop continues.
  + If the user inputs 'n' or any other value, the program prints "Exiting the program..." and breaks the loop, terminating the program.

**5. Program Exit:**

* The program exits the loop either because the user chose option 5 or decided not to continue after performing an operation.
* The program terminates execution, returning control to the operating system.