Lab Instructions:

1. We **will not** provide help for **syntax errors**; please debug them on your own.
2. We **will assist** with **logic errors** and help you understand the concepts.
3. **No extra time will be given** beyond the allotted lab session.
4. Always test your code with sample inputs before asking for help.

LAB - 0
Introduction to OOP

Question 1 :-

The Grand one Bank Challenge

Alice inherits multiple bank accounts from her late grandfather. Implement a **BankAccount** class to help her manage deposits, withdrawals, and balance inquiries securely.

Class Definition

BankAccount(account_number: str, account_holder: str, balance: float = 0.0) → Initializes an account.

deposit(amount: float) -> None → Adds amount if valid.

withdraw(amount: float) -> bool → Deducts amount if sufficient funds exist.

get_balance() -> float → Returns current balance.

Constraints

1 ≤ len(account_number) ≤ 20

1 ≤ len(account_holder) ≤ 100

0 ≤ balance ≤ 10^6

0 ≤ amount ≤ 10^6

Input 1 :

Account = BankAccount("1234567890", "Alice Johnson", 5000.0)

account.deposit(2000.0)

account.withdraw(1000.0)

account.get_balance()

Output:

6000.0

Input 2

Account = BankAccount("9876543210", "Bob Smith", 3000.0)

account.withdraw(5000.0)

Output :

False # Withdrawal fails due to insufficient funds

Question 2 :-

At NextGen University, students frequently struggle with basic arithmetic operations. Traditional calculators require manual inputs for each operation, making calculations time-consuming.

The professor has introduced a new challenge: building a **Smart Calculator** that can perform arithmetic operations on two numbers efficiently.

However, the calculator must be intelligent enough to:
✓ Automatically process the given two numbers and an operator.
✓ Handle errors gracefully, such as division by zero or invalid inputs.
✓ Notify the user when the calculation session ends.

Actual Task:

You need to develop a program that:

- Accepts two integers and an operator (+, -, *, /) as input.
- Computes the result based on the given operator.
- Handles errors such as division by zero and invalid inputs.
- Notifies the user when the session ends.

Input Format:

✅ The input consists of two integers (1 ≤ number ≤ 1000) and an operator.
✅ The valid operators are: **+ , - , * , /**

Output Format:

✓ If the operation is valid, print the computed result.
✓ If there is division by zero, print an error message.
✓ If the input is invalid, print an error message.
✓ When the calculator session ends, display a notification message.

Input:
10*43
o/p:
Result: 430
Calculator object destroyed.

input:
20/0
o/p:Error: Division by zero!
Calculator object destroyed.

Input: 45 $ 12
O/P: Error: Invalid operator!
Calculator object destroyed.

Question 3 :-This is an advanced version of Question 2(If you solved with in lab time you will get bonus point)

Enhancements to Consider:

1. Implement a calculator that takes **Expression and an operator (+, –, *, /)** as input and computes the result.

- Handle **errors** such as division by zero and invalid operators.
- Display a **session end notification** after execution.

Input 1: `(10 + 5) * 2`
Output: `Result: 30`
`Calculator object destroyed.`

Input 2 : `20 * 3+(6/3)+2`
Output: `Result: 64`
`Calculator object destroyed.`

Input: `(10 + 5 * 2`
Output: `Error: Mismatched parentheses!`

`Calculator object destroyed.`