CALCULATOR

AIM:

Develop a user-friendly calculator in Python that performs basic arithmetic operations (addition, subtraction, multiplication, division) with an intuitive and efficient interface.

Objectives:

User Interface Design:

Create a clear and simple user interface to facilitate easy interaction.

Implement input methods for user-entered numerical values and operations.

Basic Arithmetic Operations:

Implement functions for addition, subtraction, multiplication, and division.

Ensure accurate and reliable calculations for a wide range of input values.

Error Handling:

Incorporate error handling mechanisms to address invalid inputs and prevent program crashes.

Provide informative error messages to guide users in correcting input mistakes.

Memory Functionality:

Include a memory feature for storing and recalling previous results.

Allow users to clear the memory when needed.

Unit Testing:

Conduct thorough unit testing to validate the accuracy of each arithmetic operation.

Identify and rectify any bugs or issues that may arise during testing.

Documentation:

Provide clear and concise documentation outlining the functionality and usage of the calculator.

Include examples to guide users on how to perform calculations.

User Experience Enhancement:

Implement features such as keyboard shortcuts and a responsive design to enhance the overall user experience.

Seek user feedback for improvements and make iterative updates based on suggestions.

Compatibility:

Ensure compatibility across different Python versions to maximize accessibility for users.

Security Measures:

Implement security measures to prevent unauthorized access or misuse of the calculator.

Validate inputs to mitigate potential security vulnerabilities.

Optimization:

Optimize the code for efficiency and responsiveness, especially during repeated calculations.

Minimize resource usage to ensure smooth performance on various computing environments.

SOURCE CODE:

def add(x, y):

    return x + y

def subtract(x, y):

    return x - y

def multiply(x, y):

    return x \* y

def divide(x, y):

    if y != 0:

        return x / y

    else:

        return "Error: Cannot divide by zero."

# Main calculator function

def calculator():

    print("Simple Calculator")

    print("Operations:")

    print("1. Addition (+)")

    print("2. Subtraction (-)")

    print("3. Multiplication (\*)")

    print("4. Division (/)")

    try:

        choice = int(input("Enter operation number (1-4): "))

        num1 = float(input("Enter first number: "))

        num2 = float(input("Enter second number: "))

        if choice == 1:

            result = add(num1, num2)

        elif choice == 2:

            result = subtract(num1, num2)

        elif choice == 3:

            result = multiply(num1, num2)

        elif choice == 4:

            result = divide(num1, num2)

        else:

            print("Invalid operation number.")

            return

        print(f"Result: {result}")

    except ValueError:

        print("Invalid input. Please enter valid numbers.")

    except Exception as e:

        print(f"An error occurred: {e}")

# Run the calculator

calculator()

OUTPUT:

Simple Calculator

Operations:

1. Addition (+)

2. Subtraction (-)

3. Multiplication (\*)

4. Division (/)

Enter operation number (1-4): 3

Enter first number: 6

Enter second number: 7

Result: 42.0

CONCLUSION:

In conclusion, creating a calculator in Python provides a practical and hands-on way to apply programming concepts. This project enhances your understanding of basic arithmetic operations, user input handling, and logic implementation. As you continue to explore Python, you'll find opportunities to expand the calculator's functionality, improve user interfaces, and delve into more advanced programming features. Building a calculator serves as a foundational step toward tackling more complex projects and honing your programming skills.