

# ChatGPT

Improved version of the basic calculator. This version includes additional features like exponentiation, modular arithmetic, and history tracking for past calculations. It uses a dictionary-based menu for cleaner logic, provides meaningful feedback, and uses a Python list to store the user's calculation history.

## Explanation of the code:

### Imports:


Used to clear the screen for better user interaction

### Clear Screen Function:

Clears the screen using the(**os**) module.

Detects the operating system (nt for Windows, others for Unix-based systems) and uses the appropriate command (**cls** or **clear**).

python


 Copy code

```
def clear_screen():  
    """Clears the console screen."""  
    os.system('cls' if os.name == 'nt' else 'clear')
```

### Mathematical Functions:

Defines addition and similar functions for subtraction, multiplication, division, power, and modulus.


python

 Copy code

```
def divide(a, b):  
    return a / b if b != 0 else "Error: Division by zero."
```

Includes error handling for division by zero.

python

 Copy code


```
def divide(a, b):  
    return a / b if b != 0 else "Error: Division by zero."
```

## History Display Function:

Iterates through the **history** list and prints each entry.

Displays a friendly message if no history is available.

python


 Copy code

```
def display_history(history):
    """Displays the history of calculations."""
    if history:
        print("\nCalculation History:")
        for index, record in enumerate(history, 1):
            print(f"{index}. {record}")
    else:
        print("\nNo calculations performed yet.")
```

## Calculator Logic:

Initialises an empty list to store calculation history.

python

 Copy code

```
operations = {
    "1": ("Addition", add),
    "2": ("Subtraction", subtract),
    "3": ("Multiplication", multiply),
    "4": ("Division", divide),
    "5": ("Exponentiation", power),
    "6": ("Modulus", modulus)
}
```

Uses a dictionary to map user choices (**1-6**) to operation names and their corresponding functions.


This improves code readability and avoids repetitive (*if-elif*) blocks.

python

```
if choice == '9':
    print("Goodbye!")
    break
elif choice == '7':
    display_history(history)
elif choice == '8':
    history.clear()
    print("\nHistory cleared.")
```

Handles special actions like viewing or clearing the history or exiting the program.

python


 Copy code

```
elif choice in operations:  
    operation_name, operation_func = operations[choice]
```

Matches the user's choice to an operation and retrieves its name and function.

Executes the chosen operation, appends the result to the history, and displays the result.

python

 Copy code

```
result = operation_func(num1, num2)  
history.append(f"{operation_name}: {num1} and {num2} => Result: {result}")
```

## Features of This Version

**Extensibility:** Easily add more operations to the dictionary.

**History Tracking:** Users can review or clear their past calculations.

**Input Validation:** Prevents crashes from invalid input or division by zero.

**Modular Design:** Separate functions for logic and presentation make the code easy to maintain.

**Enhanced Usability:** The interactive menu is more robust, with history viewing and clearing options.