Sum function

Original Program

```python
def add_numbers(x, y):
    return x + y

a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

result = add_numbers(a, b)
print("The sum is:", result)
```

Step 1: Read First Input

```python
def add_numbers(x, y):
    return x + y

a = 5  # First input read (example)
b = int(input("Enter second number: "))

result = add_numbers(a, b)
print("The sum is:", result)
```

Step 2: Read Both Inputs

```python
def add_numbers(x, y):
    return x + y

a = 5  # First input read
b = 10  # Second input read

result = add_numbers(a, b)
print("The sum is:", result)
```

Step 3: Passing Arguments to Function

```python
def add_numbers(x, y):
    return x + y

a = 5  # First input read
b = 10  # Second input read

result = add_numbers(5, 10)  # Pass the variables
print("The sum is:", result)
```

Step 4: Inside Function - Initial State

```python
def add_numbers(5, 10):
    return 5 + 10

a = 5  # First input read
b = 10  # Second input read

result = add_numbers(5, 10)  # Function call
print("The sum is:", result)
```

Step 5: Inside Function - Addition Performed

```python
def add_numbers(5, 10):
    return 15

a = 5  # First input read
b = 10  # Second input read

result = add_numbers(5, 10)  # Function call
print("The sum is:", result)
```

Step 6: Return the Result

```python
a = 5  # First input read
b = 10  # Second input read

result = 15  # Result from function
print("The sum is:", result)
```

Step 7: Final Output

```python
a = 5  # First input read
b = 10  # Second input read

result = 15  # Function result
print("The sum is:", 15)  # Final output
```

Multiply $+$ Sum function

Original Program

```python
def sum(a, b):
    return a + b

def a_times_b_plus_c(a, b, c):
    return a * sum(b, c)

# Variables initialized
a = 2
b = 3
c = 4

s = sum(a, b)
s_m = a_times_b_plus_c(a, b, c)
```

Step 1: Initial Variable Replacement

```python
def sum(a, b):
    return a + b

def a_times_b_plus_c(a, b, c):
    return a * sum(b, c)

# Variables initialized
a = 2
b = 3
c = 4

s = sum(2, 3)  # Replace a with 2, b with 3
s_m = a_times_b_plus_c(2, 3, 4)  # Replace a with 2, b with 3, c with 4
```

Step 1: Inside sum Function

```
def sum(2, 3):
    return 2 + 3

def a_times_b_plus_c(a, b, c):
    return a * sum(b, c)

# Variables initialized
a = 2
b = 3
c = 4

s = sum(2, 3)  # Replace a with 2, b with 3
s_m = a_times_b_plus_c(2, 3, 4)  # Replace a with 2, b with 3, c with 4
```

Step 2: Inside sum Function

```python
def sum(a, b):
    return 5  # Replace a with 2, b with 3, returns 5

def a_times_b_plus_c(a, b, c):
    return a * sum(b, c)

# Variables initialized
a = 2
b = 3
c = 4

s = sum(2, 3)  # Replace a with 2, b with 3
s_m = a_times_b_plus_c(2, 3, 4)  # Replace a, b, c
```

Step 3: Inside a_times_b_plus_c (sum(b, c))

```python
def a_times_b_plus_c(2, 3, 4):
    return 2 * sum(3, 4)  # Replace sum(b, c) with sum(3, 4)

# Variables initialized
a = 2
b = 3
c = 4

s = 5  # Result of sum(2, 3)
s_m = a_times_b_plus_c(2, 3, 4)  # Replace a with 2, b with 3, c with 4
```

Step 4: Inside sum(b, c)

```python
def sum(3, 4):
    return 3 + 4

def a_times_b_plus_c(a, b, c):
    return 2 * sum(3,4)  # sum(3, 4) evaluates to 3 + 4

# Variables initialized
a = 2
b = 3
c = 4

s = 5  # Result of sum(2, 3)
s_m = a_times_b_plus_c(2, 3, 4)  # Replace sum(3, 4) with 3 + 4
```

Step 5: Final Computation of a_times_b_plus_c

```python
def a_times_b_plus_c(a, b, c):
    return 2 * 7  # (3 + 4) evaluated to 7

# Variables initialized
a = 2
b = 3
c = 4

s = 5  # Result of sum(2, 3)
s_m = a_times_b_plus_c(2, 3, 4)
```

Step 6: Final Result

```python
def a_times_b_plus_c(a, b, c):
    return 14  # 2 * 7 = 14

# Variables initialized
a = 2
b = 3
c = 4

s = 5  # Result of sum(2, 3)
s_m = a_times_b_plus_c(2, 3, 4)
```

Step 6: Final Result

```
# Variables initialized
a = 2
b = 3
c = 4

s = 5  # Result of sum(2, 3)
s_m = 14
```

Factorial

Original Program

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

result = factorial(3)
```

Step 1: First Call (factorial(3))

```
    def factorial(3):
        if n == 0:
            return 1
        else:
            return 3 * factorial (2)  # Call factorial (3-1)

    result = factorial (3)
```

Step 2: Second Call (factorial(2))

```

def factorial (2):
    if n == 0:
        return 1
    else:
        return 2 * factorial (1)


def factorial(3):
    if n == 0:
        return 1
    else:
        return 3 * factorial (2)


result = factorial (3)
```

Step 3: Third Call (factorial(1))

```
def factorial (1):
    if n == 0:
        return 1
    else:
        return 1 * factorial (0)

def factorial (2):
    if n == 0:
        return 1
    else:
        return 2 * factorial(1)

def factorial (3):
    if n == 0:
        return 1
    else:
        return 3 * factorial (2)

result = factorial(3)
```

Step 4: Base Case Reached (factorial(0))

```
def factorial(0):
    if n == 0:
        return 1  # Base case reached

def factorial (1):
    if n == 0:
        return 1
    else:
        return 1 * factorial (0)

def factorial (2):
    if n == 0:
        return 1
    else:
        return 2 * factorial (1)

def factorial (3):
    if n == 0:
        return 1
    else:
        return 3 * factorial (2)

```

Step 5: Unwinding - factorial(1)

```python

def factorial (1):
    if n == 0:
        return 1
    else:
        return 1 * 1  # return 1

def factorial (2):
    if n == 0:
        return 1
    else:
        return 2 * factorial (1)

def factorial (3):
    if n == 0:
        return 1
else:
    return 3 * factorial (2)

result = factorial (3)
```

Step 6: Unwinding - factorial(2)

```python

    def factorial (2):
        if n == 0:
            return 1
        else:
            return 2 * 1  # factorial_1(1) returned 1

    def factorial (3):
        if n == 0:
            return 1
        else:
            return 3 * factorial (2)

    result = factorial_3(3)
```

Step 7: Final Result

```
def factorial (3):
    if n == 0:
        return 1
    else:
        return 3 * 2  # factorial_2(2) returned 2

result = factorial (3)
```

Step 7: Final Result

```
    result = 6
```