Sum function

Original Program

```
1  def add_numbers(x, y):
2      return x + y
3
4  a = int(input("Enter first number: "))
5  b = int(input("Enter second number: "))
6
7  result = add_numbers(a, b)
8  print("The sum is:", result)
```

Step 1: Read First Input

```
1  def add_numbers(x, y):
2      return x + y
3
4  a = 5  # First input read (example)
5  b = int(input("Enter second number: "))
6
7  result = add_numbers(a, b)
8  print("The sum is:", result)
```

Step 2: Read Both Inputs

```
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

```
1  def add_numbers(x, y):
2      return x + y
3
4  a = 5  # First input read
5  b = 10  # Second input read
6
7  result = add_numbers(5, 10)  # Pass the variables
8  print("The sum is:", result)
```

Step 4: Inside Function - Initial State

```
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

```
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

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

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

Step 7: Final Output

```
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

```
1   def sum(a, b):
2       return a + b
3
4   def a_times_b_plus_c(a, b, c):
5       return a * sum(b, c)
6
7   # Variables initialized
8   a = 2
9   b = 3
10  c = 4
11
12  s = sum(a, b)
13  s_m = a_times_b_plus_c(a, b, c)
```

Step 1: Initial Variable Replacement

```
1   def sum(a, b):
2       return a + b
3
4   def a_times_b_plus_c(a, b, c):
5       return a * sum(b, c)
6
7   # Variables initialized
8   a = 2
9   b = 3
10  c = 4
11
12  s = sum(2, 3)  # Replace a with 2, b with 3
13  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

```
1     def sum(2, 3):
2         return 2 + 3
3
4     def a_times_b_plus_c(a, b, c):
5         return a * sum(b, c)
6
7     # Variables initialized
8     a = 2
9     b = 3
10    c = 4
11
12    s = sum(2, 3)  # Replace a with 2, b with 3
13    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

```
1   def sum(a, b):
2       return 5  # Replace a with 2, b with 3,
                  returns 5
3
4   def a_times_b_plus_c(a, b, c):
5       return a * sum(b, c)
6
7   # Variables initialized
8   a = 2
9   b = 3
10  c = 4
11
12  s = sum(2, 3)  # Replace a with 2, b with 3
13  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))

```
1     def a_times_b_plus_c(2, 3, 4):
2         return 2 * sum(3, 4)  # Replace sum(b, c) with
              sum(3, 4)
3
4     # Variables initialized
5     a = 2
6     b = 3
7     c = 4
8
9     s = 5  # Result of sum(2, 3)
10    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)

```
1     def sum(3, 4):
2         return 3 + 4
3
4     def a_times_b_plus_c(a, b, c):
5         return 2 * sum(3,4)  # sum(3, 4) evaluates to
              3 + 4
6
7     # Variables initialized
8     a = 2
9     b = 3
10    c = 4
11
12    s = 5  # Result of sum(2, 3)
13    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

```
1     def a_times_b_plus_c(a, b, c):
2         return 2 * 7  # (3 + 4) evaluated to 7
3
4     # Variables initialized
5     a = 2
6     b = 3
7     c = 4
8
9     s = 5  # Result of sum(2, 3)
10    s_m = a_times_b_plus_c(2, 3, 4)
```

Step 6: Final Result

```
1     def a_times_b_plus_c(a, b, c):
2         return 14  # 2 * 7 = 14
3
4     # Variables initialized
5     a = 2
6     b = 3
7     c = 4
8
9     s = 5  # Result of sum(2, 3)
10    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

```
1  def factorial(n):
2      if n == 0:
3          return 1
4      else:
5          return n * factorial(n-1)
6
7  result = factorial(3)
```

Step 1: First Call (factorial(3))

```
1     def factorial(3):
2         if n == 0:
3             return 1
4         else:
5             return 3 * factorial (2)  # Call factorial
                  (3-1)
6
7     result = factorial (3)
```

Step 2: Second Call (factorial(2))

```
1
2       def factorial (2):
3           if n == 0:
4               return 1
5           else:
6               return 2 * factorial (1)
7
8
9       def factorial(3):
10          if n == 0:
11              return 1
12          else:
13              return 3 * factorial (2)
14
15
16      result = factorial (3)
```

Step 3: Third Call (factorial(1))

```
1     def factorial (1):
2         if n == 0:
3             return 1
4         else:
5             return 1 * factorial (0)
6
7     def factorial (2):
8         if n == 0:
9             return 1
10        else:
11            return 2 * factorial(1)
12
13    def factorial (3):
14        if n == 0:
15            return 1
16        else:
17            return 3 * factorial (2)
18
19    result = factorial(3)
```

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

```
1     def factorial(0):
2         if n == 0:
3             return 1  # Base case reached
4
5     def factorial (1):
6         if n == 0:
7             return 1
8         else:
9             return 1 * factorial (0)
10
11    def factorial (2):
12        if n == 0:
13            return 1
14        else:
15            return 2 * factorial (1)
16
17    def factorial (3):
18        if n == 0:
19            return 1
20        else:
21            return 3 * factorial (2)
22
```

Step 5: Unwinding - factorial(1)

```
1
2       def factorial (1):
3           if n == 0:
4               return 1
5           else:
6               return 1 * 1  # return 1
7
8       def factorial (2):
9           if n == 0:
10              return 1
11          else:
12              return 2 * factorial (1)
13
14      def factorial (3):
15          if n == 0:
16              return 1
17      else:
18          return 3 * factorial (2)
19
20      result = factorial (3)
```

Step 6: Unwinding - factorial(2)

```
1
2       def factorial (2):
3           if n == 0:
4               return 1
5           else:
6               return 2 * 1  # factorial_1(1) returned 1
7
8       def factorial (3):
9           if n == 0:
10              return 1
11          else:
12              return 3 * factorial (2)
13
14      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

```
1     result = 6
```