### Recursion

Recursion is a method where a function calls itself in order to solve a problem. Each call to the function handles a smaller version of the problem until it reaches a base case, which is a condition where the recursion stops. Recursive functions often break down complex problems into simpler, smaller cases, and then combine the solutions of these cases.

### Recursive Function and Tracing

A recursive function has two main parts:

1. Base Case: The condition under which the recursion stops. Without a base case, the function would continue calling itself indefinitely.

2. Recursive Case: The part where the function calls itself with a modified argument, moving towards the base case.

Let's look at an example of a recursive function to find the sum of the first \( n \) natural numbers.

### Sum of First \( n \) Natural Numbers Using Recursion

The sum of the first \( n \) natural numbers can be defined recursively as:

\[

\text{sum}(n) = n + \text{sum}(n-1)

\]

with the base case:

\[

\text{sum}(1) = 1

\]

#### Code Example:

def sum\_natural\_numbers(n):

# Base case

if n == 1:

return 1

# Recursive case

return n + sum\_natural\_numbers(n - 1)

# Example usage

n = 5

print(f"The sum of the first {n} natural numbers is: {sum\_natural\_numbers(n)}")

#### Tracing the Recursive Calls

For `sum\_natural\_numbers(5)`, the calls happen as follows:

- `sum\_natural\_numbers(5)` calls `sum\_natural\_numbers(4)`

- `sum\_natural\_numbers(4)` calls `sum\_natural\_numbers(3)`

- `sum\_natural\_numbers(3)` calls `sum\_natural\_numbers(2)`

- `sum\_natural\_numbers(2)` calls `sum\_natural\_numbers(1)`

- `sum\_natural\_numbers(1)` returns 1 (base case)

- `sum\_natural\_numbers(2)` returns \( 2 + 1 = 3 \)

- `sum\_natural\_numbers(3)` returns \( 3 + 3 = 6 \)

- `sum\_natural\_numbers(4)` returns \( 4 + 6 = 10 \)

- `sum\_natural\_numbers(5)` returns \( 5 + 10 = 15 \)

Thus, the sum of the first 5 natural numbers is 15.

### Assignment-18: Simple Problems on Recursion

#### T1. Write a Recursive Function to Print First \( N \) Natural Numbers

def print\_natural\_numbers(n):

if n > 0:

print\_natural\_numbers(n - 1)

print(n, end=' ')

# Example usage

print("First N natural numbers:")

print\_natural\_numbers(5)

print()

#### T2. Write a Recursive Function to Print First \( N \) Natural Numbers in Reverse Order

def print\_natural\_numbers\_reverse(n):

if n > 0:

print(n, end=' ')

print\_natural\_numbers\_reverse(n - 1)

# Example usage

print("First N natural numbers in reverse order:")

print\_natural\_numbers\_reverse(5)

print()

#### T3. Write a Recursive Function to Print First \( N \) Odd Natural Numbers

def print\_odd\_numbers(n):

if n > 0:

print\_odd\_numbers(n - 1)

print(2 n - 1, end=' ')

# Example usage

print("First N odd natural numbers:")

print\_odd\_numbers(5)

print()

#### T4. Write a Recursive Function to Print First \( N \) Even Natural Numbers

def print\_even\_numbers(n):

if n > 0:

print\_even\_numbers(n - 1)

print(2 n, end=' ')

# Example usage

print("First N even natural numbers:")

print\_even\_numbers(5)

print()

#### T5. Write a Recursive Function to Print First \( N \) Odd Natural Numbers in Reverse Order

def print\_odd\_numbers\_reverse(n):

if n > 0:

print(2 n - 1, end=' ')

print\_odd\_numbers\_reverse(n - 1)

# Example usage

print("First N odd natural numbers in reverse order:")

print\_odd\_numbers\_reverse(5)

print()

#### T6. Write a Recursive Function to Print First \( N \) Even Natural Numbers in Reverse Order

def print\_even\_numbers\_reverse(n):

if n > 0:

print(2 n, end=' ')

print\_even\_numbers\_reverse(n - 1)

# Example usage

print("First N even natural numbers in reverse order:")

print\_even\_numbers\_reverse(5)

print()

Each function follows the same recursive principle of breaking down the problem by reducing \( n \) until the base case is reached, then prints the numbers either in forward or reverse order as specified.

### Assignment-19: Simple Problems on Recursion

Here are recursive solutions for each of the problems given.

#### 1. Write a Recursive Function to Calculate the Sum of First \( N \) Natural Numbers

The sum of the first \( N \) natural numbers can be defined as:

\[

\text{sum}(N) = N + \text{sum}(N - 1)

\]

with the base case:

\[

\text{sum}(0) = 0

\]

Code:

def sum\_natural\_numbers(n):

if n == 0:

return 0

return n + sum\_natural\_numbers(n - 1)

# Example usage

n = 5

print(f"Sum of first {n} natural numbers: {sum\_natural\_numbers(n)}")

#### 2. Write a Recursive Function to Calculate the Sum of First \( N \) Odd Natural Numbers

The sum of the first \( N \) odd natural numbers can be defined as:

\[

\text{sum\\_odd}(N) = (2 \times N - 1) + \text{sum\\_odd}(N - 1)

\]

with the base case:

\[

\text{sum\\_odd}(0) = 0

\]

Code:

```python

def sum\_odd\_numbers(n):

if n == 0:

return 0

return (2 n - 1) + sum\_odd\_numbers(n - 1)

# Example usage

n = 5

print(f"Sum of first {n} odd natural numbers: {sum\_odd\_numbers(n)}")

#### 3. Write a Recursive Function to Calculate the Sum of First \( N \) Even Natural Numbers

The sum of the first \( N \) even natural numbers can be defined as:

\[

\text{sum\\_even}(N) = (2 \times N) + \text{sum\\_even}(N - 1)

\]

with the base case:

\[

\text{sum\\_even}(0) = 0

\]

Code:

def sum\_even\_numbers(n):

if n == 0:

return 0

return (2 n) + sum\_even\_numbers(n - 1)

# Example usage

n = 5

print(f"Sum of first {n} even natural numbers: {sum\_even\_numbers(n)}")

#### 4. Write a Recursive Function to Calculate the Factorial of a Number

The factorial of \( N \) can be defined as:

\[

\text{factorial}(N) = N \times \text{factorial}(N - 1)

\]

with the base case:

\[

\text{factorial}(0) = 1

\]

Code:

def factorial(n):

if n == 0:

return 1

return n factorial(n - 1)

# Example usage

n = 5

print(f"Factorial of {n}: {factorial(n)}")

#### 5. Write a Recursive Function to Calculate the Sum of Squares of First \( N \) Natural Numbers

The sum of squares of the first \( N \) natural numbers can be defined as:

\[

\text{sum\\_squares}(N) = N^2 + \text{sum\\_squares}(N - 1)

\]

with the base case:

\[

\text{sum\\_squares}(0) = 0

\]

Code:

def sum\_squares(n):

if n == 0:

return 0

return (n n) + sum\_squares(n - 1)

# Example usage

n = 5

print(f"Sum of squares of first {n} natural numbers: {sum\_squares(n)}")

Each function uses recursion to break down the problem into smaller sub-problems, solving them until reaching the base case, where recursion stops. The results are then combined as the recursive calls return to give the final answer.