রিকারসিভ ফাংশন (Recursive Function) হলো এমন একটি ফাংশন যা নিজেকে নিজে কল করে। এটি সাধারণত এমন সমস্যা সমাধানে ব্যবহৃত হয় যেখানে একটি বড় সমস্যাকে ছোট ছোট উপ-সমস্যায় ভাগ করে সমাধান করা যায়। রিকারসনের মূল বিষয় হলো:

1. **বেস কেস (Base Case)**: এটি এমন একটি শর্ত যেখানে ফাংশন আর নিজেকে কল করে না এবং সরাসরি ফলাফল ফেরত দেয়।
2. **রিকারসিভ কেস (Recursive Case)**: এখানে ফাংশন নিজেকে কল করে সমস্যাটিকে আরও ছোট করে।

এখন আমি পাইথনে রিকারসিভ ফাংশন বোঝানোর জন্য কয়েকটি সহজ উদাহরণ দেবো:

উদাহরণ ১: ফ্যাক্টোরিয়াল (Factorial) গণনা

ফ্যাক্টোরিয়াল হলো একটি সংখ্যার গুণফল, যেমন `5! = 5 × 4 × 3 × 2 × 1 = 120`। এটি রিকারসিভভাবে গণনা করা যায়।

```python
def factorial(n):
    # বেস কেস: যদি n = 0 বা 1 হয়, তাহলে ফ্যাক্টোরিয়াল 1
    if n == 0 or n == 1:
        return 1
    # রিকারসিভ কেস: n! = n × (n-1)!
    else:
        return n * factorial(n - 1)

# ফাংশন কল
print(factorial(5))  # আউটপুট: 120
```

কীভাবে কাজ করে?

- `factorial(5)` কল করলে:
  - `5 * factorial(4)`
  - `5 * (4 * factorial(3))`
  - `5 * (4 * (3 * factorial(2)))`
  - `5 * (4 * (3 * (2 * factorial(1))))`
  - `5 * (4 * (3 * (2 * 1)))`
  - `5 * (4 * (3 * 2))`
  - `5 * (4 * 6)`
  - `5 * 24 = 120`

ব্যাখ্যা:

- **বেস কেস:** `n == 0` বা `n == 1` হলে ফাংশন ১ ফেরত দেয়।
- **রিকারসিভ কেস:** `n` এর সাথে `factorial(n-1)` গুণ করে ফলাফল দেয়।
- এভাবে ফাংশন নিজেকে কল করে সমস্যাটিকে ছোট করে।

উদাহরণ ২: ফিবোনাচি সংখ্যা (Fibonacci Sequence)

ফিবোনাচি সংখ্যা হলো এমন একটি সিরিজ যেখানে প্রতিটি সংখ্যা আগের দুটি সংখ্যার যোগফল। যেমন: `0, 1, 1, 2, 3, 5, 8, 13, ...`

```python
def fibonacci(n):
    # বেস কেস
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    # রিকারসিভ কেস
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# ফাংশন কল
print(fibonacci(6))  # আউটপুট: 8
```

কীভাবে কাজ করে?

- `fibonacci(6)` কল করলে:
  - `fibonacci(5) + fibonacci(4)`
  - `(fibonacci(4) + fibonacci(3)) + (fibonacci(3) + fibonacci(2))`
  - এভাবে ক্রমাগত ভেঙে বেস কেসে পৌঁছায় (`fibonacci(0) = 0` এবং `fibonacci(1) = 1`)।
  - শেষে: `(5 + 3) = 8`

ব্যাখ্যা:

- বেস কেসে `n <= 0` হলে ০ এবং `n == 1` হলে ১ ফেরত দেয়।
- রিকারসিভ কেসে `n`-এর জন্য আগের দুটি সংখ্যার যোগফল গণনা করে।

উদাহরণ ৩: সংখ্যার যোগফল (Sum of Numbers)

ধরি, আমরা ১ থেকে n পর্যন্ত সংখ্যার যোগফল বের করতে চাই। যেমন: `1 + 2 + 3 + 4 + 5 = 15`।

```python
def sum_numbers(n):
    # বেস কেস
    if n <= 0:
        return 0
    # রিকারসিভ কেস
    else:
        return n + sum_numbers(n - 1)

# ফাংশন কল
print(sum_numbers(5))  # আউটপুট: 15
```

কীভাবে কাজ করে?

- `sum_numbers(5)` কল করলে:
  - `5 + sum_numbers(4)`
  - `5 + (4 + sum_numbers(3))`
  - `5 + (4 + (3 + sum_numbers(2)))`
  - `5 + (4 + (3 + (2 + sum_numbers(1))))`
  - `5 + (4 + (3 + (2 + 1)))`
  - `5 + (4 + (3 + 3))`
  - `5 + (4 + 6)`
  - `5 + 10 = 15`

ব্যাখ্যা:

- বেস কেসে `n <= 0` হলে ০ ফেরত দেয়।
- রিকারসিভ কেসে `n` এর সাথে `sum_numbers(n-1)` যোগ করে।

রিকারসনের সুবিধা ও অসুবিধা

সুবিধা:

1. জটিল সমস্যাকে সহজে ভাগ করে সমাধান করা যায়।
2. কোড সংক্ষিপ্ত ও পড়তে সহজ হয়।
3. কিছু সমস্যা (যেমন: ট্রি ট্রাভার্সাল, ফিবোনাচি) রিকারসন দিয়ে সহজে সমাধান করা যায়।

অসুবিধা:

1. অতিরিক্ত মেমোরি ব্যবহার হয় কারণ প্রতিটি ফাংশন কল স্ট্যাক মেমোরিতে সংরক্ষিত হয়।
2. বড় ইনপুটের জন্য রিকারসন ধীর হতে পারে (যেমন ফিবোনাচি উদাহরণে)।
3. "Stack Overflow" ত্রুটি হতে পারে যদি রিকারসন খুব গভীর হয়।

কখন রিকারসন ব্যবহার করবেন?

- যখন সমস্যাটি ছোট ছোট উপ-সমস্যায় ভাগ করা যায়।
- যখন কোডের সরলতা গুরুত্বপূর্ণ।
- তবে, পারফরম্যান্সের জন্য বড় ইনপুটের ক্ষেত্রে লুপ বা **মেমোইজেশন** (যেমন ফিবোনাচির ক্ষেত্রে) ব্যবহার করা ভালো।

মেমোইজেশন উদাহরণ (ফিবোনাচি উন্নত করা):

```python
def fibonacci_memo(n, memo={}):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    elif n in memo:
        return memo[n]
    else:
        memo[n] = fibonacci_memo(n - 1, memo) + fibonacci_memo(n - 2, memo)
        return memo[n]

print(fibonacci_memo(50))  # দ্রুত আউটপুট পাওয়া যায়
```

এখানে `memo` ডিকশনারি ব্যবহার করে পূর্বের ফলাফল সংরক্ষণ করা হয়, যা সময় বাঁচায়।

উপসংহার

রিকারসিভ ফাংশন পাইথনে জটিল সমস্যা সমাধানের একটি শক্তিশালী হাতিয়ার। তবে এটি ব্যবহারের আগে বেস কেস এবং রিকারসিভ কেস ভালোভাবে বোঝা জরুরি। উপরের উদাহরণগুলো (ফ্যাক্টোরিয়াল, ফিবোনাচি, যোগফল) দিয়ে আশা করি রিকারসনের ধারণা পরিষ্কার হয়েছে। যদি আরও কোনো প্রশ্ন থাকে, জানাও!