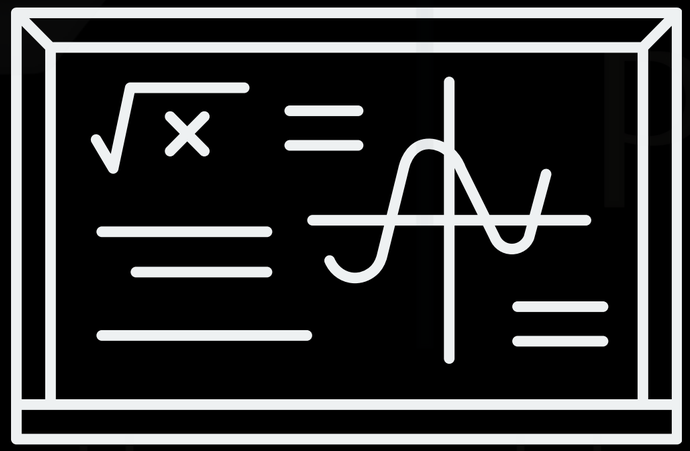
Python Recursive Functions

Pankaj Chouhan
www.codeswithpankaj.com

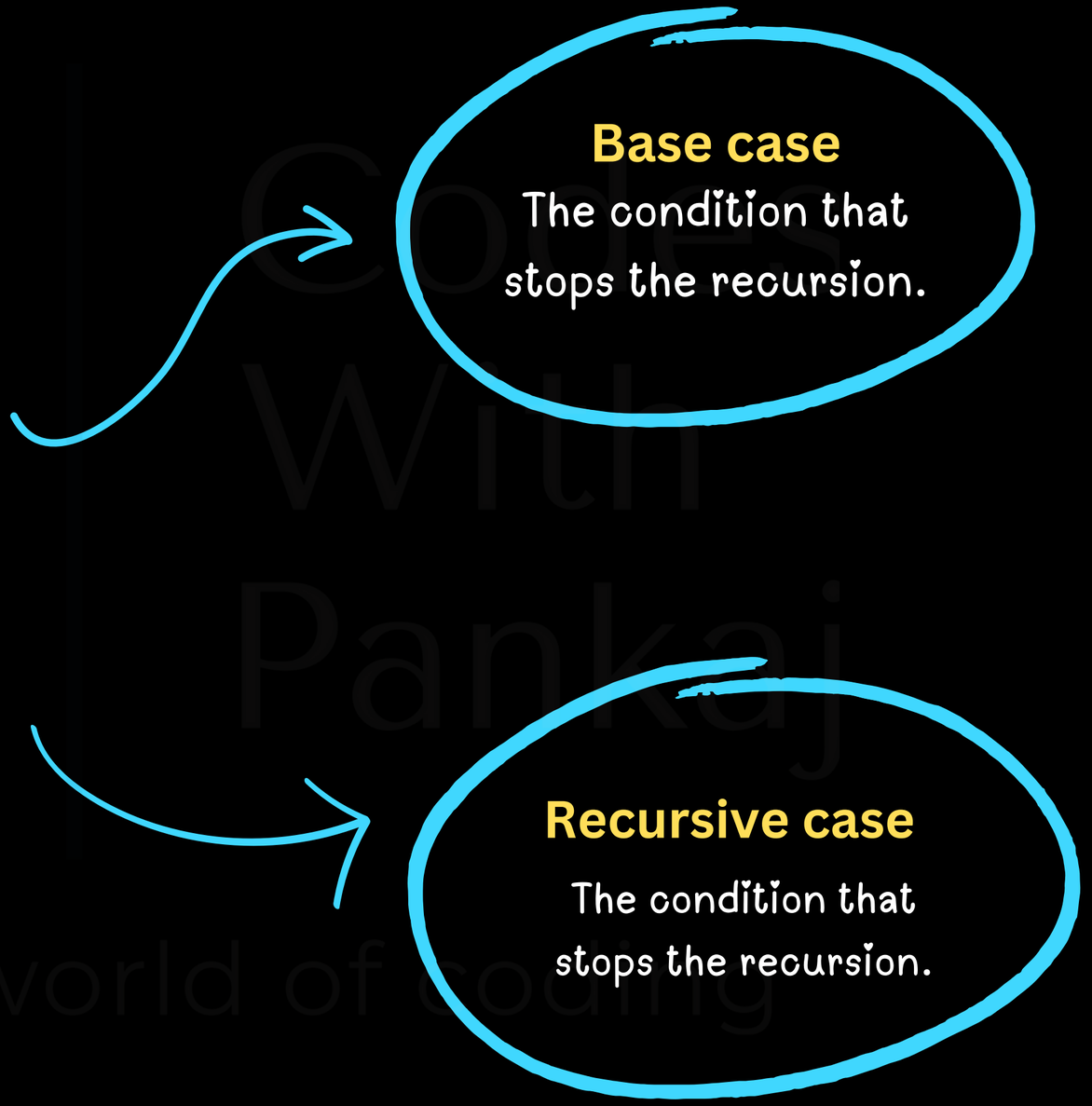
Recursion is a fundamental programming concept where a function calls itself to solve smaller instances of a problem. In Python, recursive functions are commonly used to solve problems that can be broken down into smaller, similar subproblems, such as factorial calculation, Fibonacci series, and tree traversals.



Pankaj Chouhan
www.codeswithpankaj.com

What is Recursion ?

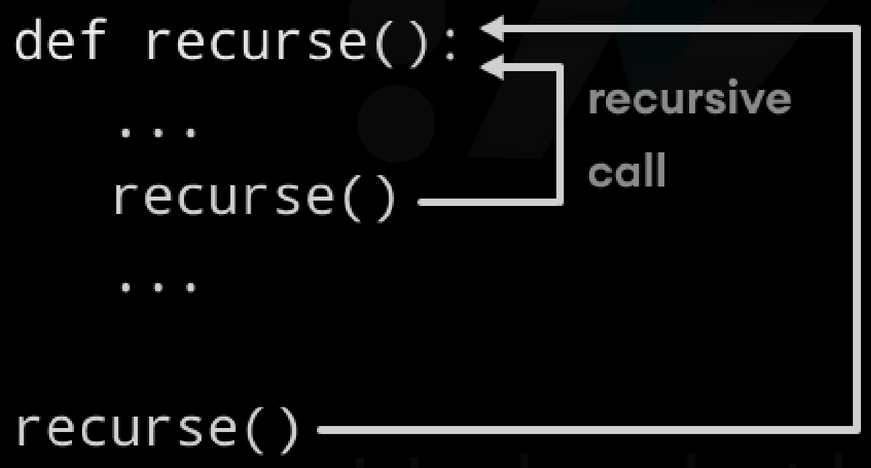
Recursion is a process where a function calls itself. It continues until a stopping condition (called the base case) is met.



Pankaj Chouhan
www.codeswithpankaj.com

How Does a Recursive Function Work ?

When a function calls itself, each call is stored in the call stack. The function keeps calling itself until it reaches the base case. Then, the function starts returning values and unwinding back through the call stack.



Example

```
def recurse():
    print("Calling recurse()")
    recurse()
    # Recursive call

recurse()
```

output

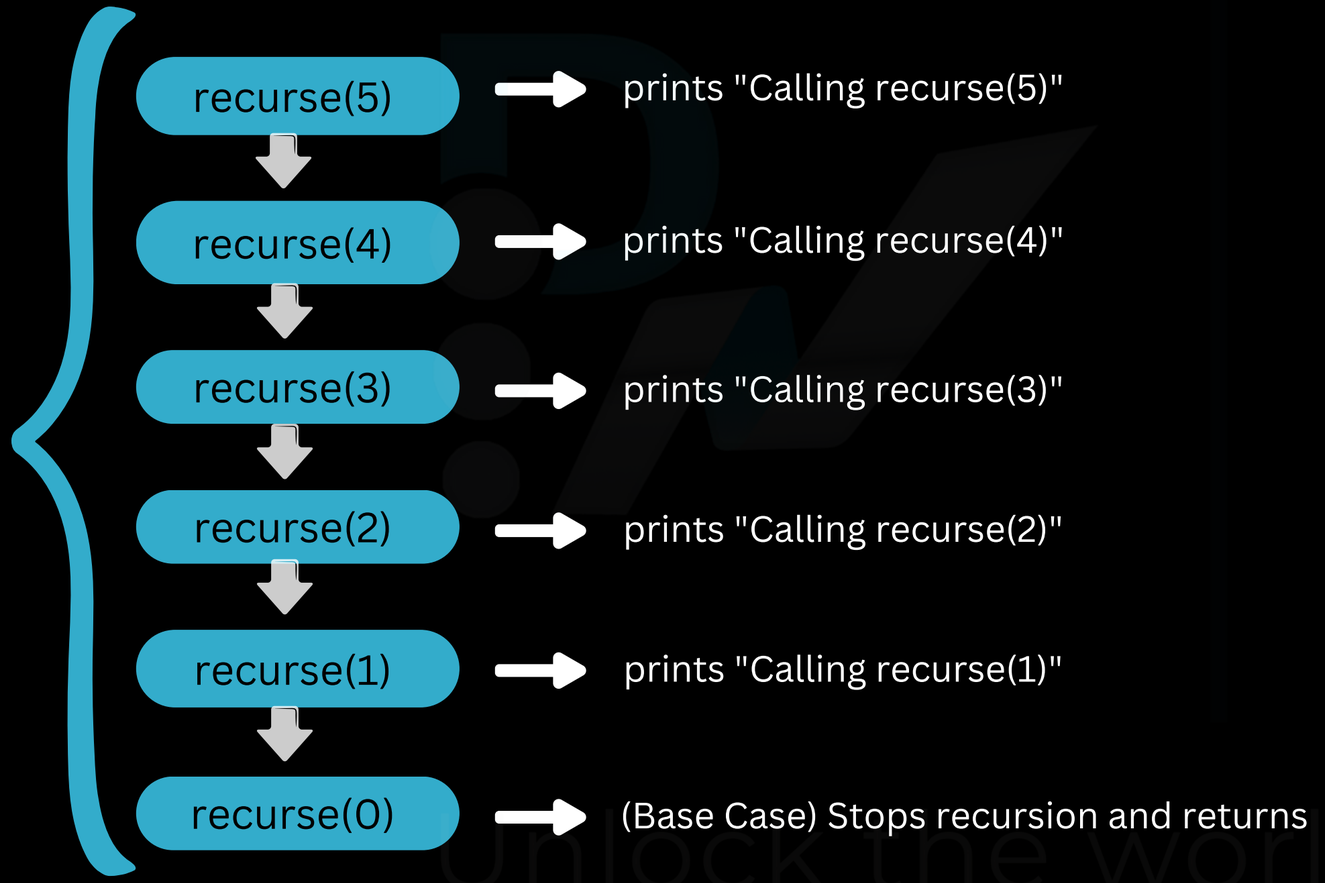
```
Calling recurse()
Calling recurse()
Calling recurse()
...
RecursionError: maximum recursion depth exceeded
```

- The function recurse() prints "Calling recurse()".
- Then, it calls itself, leading to an infinite recursion (no base case).
- This will eventually cause a RecursionError because Python has a recursion limit.

Pankaj Chouhan
www.codeswithpankaj.com

To prevent infinite recursion, always include a base case like this :

Step-by-Step Execution



Pankaj Chouhan
www.codeswithpankaj.com

Example

```
def recurse(n):
    if n == 0:  # Base case to stop recursion
        return
    print(f"Calling recurse({n})")  # Print current call
    recurse(n - 1)  # Recursive call

recurse(5)  # Starts recursion
```

Output

```
Calling recurse(5)
Calling recurse(4)
Calling recurse(3)
Calling recurse(2)
Calling recurse(1)
```

Examples of Recursive Functions

Factorial of a Number

The **formula for factorial** of a number n is:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

Or, using **recursion**:

$$n! = \begin{cases} 1, & \text{if } n = 0 \text{ or } n = 1 \quad \text{(Base Case)} \\ n \times (n-1)!, & \text{if } n > 1 \quad \text{(Recursive Case)} \end{cases}$$

Examples:

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- $4! = 4 \times 3 \times 2 \times 1 = 24$
- $3! = 3 \times 2 \times 1 = 6$
- $2! = 2 \times 1 = 2$
- $1! = 1$ (Base Case)
- $0! = 1$ (Base Case)

Pankaj Chouhan
www.codeswithpankaj.com

Example

```
def factorial(n):
    if n == 0 or n == 1:  # Base case
        return 1
    else:  # Recursive case
        return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

Output { **120**

Step-by-Step Execution

```
def factorial(n):
    if n == 0 or n == 1:  # Base case
        return 1
    else:  # Recursive case
        return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

Function Calls (Going Down the Stack)

Each function call breaks the problem into a smaller one:

Function Call	Computation
factorial(5)	5 * factorial(4)
factorial(4)	4 * factorial(3)
factorial(3)	3 * factorial(2)
factorial(2)	2 * factorial(1)
factorial(1)	Base Case → Returns 1

Function Returns (Unwinding the Stack)

Now, the recursive calls return their values:

Function Call	Returns
factorial(1)	1
factorial(2)	2 * 1 = 2
factorial(3)	3 * 2 = 6
factorial(4)	4 * 6 = 24
factorial(5)	5 * 24 = 120

Thus, factorial(5) returns 120.

Pankaj Chouhan
www.codeswithpankaj.com