# Functions in Python: Defining and Calling Functions

## Introduction to Functions

A function is a block of reusable code that performs a specific task. Functions help in organizing code, reducing redundancy, and improving readability and maintenance of programs. Python functions can accept input, process it, and return a result.

## Types of Functions in Python

## 1. Built-in Functions

These are predefined functions provided by Python. Examples include `print()`, `len()`, `type()`, etc.

## 2. User-defined Functions

These are functions created by users to perform specific tasks.

## 3. Lambda Functions

Also known as anonymous functions, these are small, single-expression functions.

## 4. Recursive Functions

Functions that call themselves until a base condition is met. Useful for tasks like calculating factorials or Fibonacci numbers.

## Function Syntax and Definition

A function in Python is defined using the `def` keyword followed by the function name and parentheses containing optional parameters.

## Function Parameters and Arguments

## Types of Arguments:

1. Positional Arguments – Passed in the same order as defined in the function.  
2. Keyword Arguments – Passed with the parameter name, and order does not matter.  
3. Default Arguments – Have default values if no argument is passed.  
4. Variable-length Arguments – Allow multiple values to be passed.  
 \* `\*args`: Non-keyword variable-length arguments.  
 \* `\*\*kwargs`: Keyword variable-length arguments.

## Scope of Variables in Functions

1. Local Scope – Variables declared inside a function, only accessible within that function.  
2. Global Scope – Variables declared outside all functions, accessible everywhere.  
3. Nonlocal Scope – Variables in the enclosing scope of a nested function.

## Types of User-defined Functions

1. Simple Function – Performs a specific task and may or may not return a value.  
2. Functions with Parameters – Accept arguments to process specific data.  
3. Functions with Return Values – Return data to the caller using the `return` statement.  
4. Lambda Functions – One-liner functions for simple operations.  
5. Recursive Functions – Functions that call themselves.

## Nested Functions

A nested function is a function defined inside another function. This allows the inner function to access variables of the outer function.

## Recursive Functions

A recursive function calls itself until a base condition is met. It is useful for tasks like calculating factorials or Fibonacci numbers.

## Function Annotations

Function annotations provide metadata about the types of function arguments and return values. These annotations do not affect the function’s behavior but help with documentation and type hints.

## Advantages of Functions

- Code Reusability – Functions can be called multiple times.  
- Modularity – Code is organized into smaller blocks.  
- Improved Readability – Functions make the code easier to understand.  
- Easier Maintenance – Bugs are easier to locate and fix.

## Summary

Functions are essential for writing reusable, modular code in Python. Understanding the different types of functions and parameters helps create flexible, efficient programs. Mastering function concepts allows developers to build scalable and maintainable software systems.