Functions - Building Reusable Code Blocks in Python

Introduction:

- ***Definition:*** A function is a block of reusable code that performs a specific task.
- ***Advantages:*** Encapsulation, Reusability, Modularity.

Function Syntax & Components:

- **Syntax**

```
def function_name(parameters):
    # Function body
   # Code to perform the task
    return result   # Optional
```

- **Components**
  1. Function Keyword: def indicates the start of a function definition.
  2. Function Name: Descriptive name for the function.
  3. Parameters: Input values that the function receives (if any).
  4. Colon : (:) Marks the beginning of the function body.
  5. Function Body: Block of code defining the function's task.
  6. Return Statement: Specifies the value the function sends back (if any).

```
def greet(name):

    """This function greets the person passed in as a parameter."""

    print("Hello, " + name + "!")
```

Function Invocation:

Call a function by using its name followed by parentheses.

```
greet("John")  # Output: Hello, John!
```

Return Statement:

Use return to send a value back from the function.

```
def add(x, y):
    """This function adds two numbers."""
    return x + y

result = add(3, 5)  # Result: 8
```

Default Parameters:

Set default values for parameters.

```
def power(base, exponent=2):
    """This function raises 'base' to the power of 'exponent'."""
    return base ** exponent

result1 = power(2)     # Result: 4
result2 = power(2, 3)  # Result: 8
```

Conclusion:

- Functions are essential for writing organized, modular, and reusable code.
- Effective use of functions enhances code readability and maintainability.