# **1. What is a Function?**

A function is a named block of code created to perform a specific task. Instead of repeating the same instructions, you define a function once and reuse it whenever needed. Think of a function like a washing machine: you put clothes in (input), press start (call it), and get clean clothes out (output).

# **2. Defining a Function using `def`**

In Python, functions are defined using the `def` keyword. The function must have a name, followed by parentheses which may include parameters. The logic is indented under a colon.  
  
Example:  
def greet():  
 print("Hello!")

# **3. Calling a Function**

To use a function, you call it using its name followed by parentheses. This executes the code inside the function.  
  
Example:  
greet() # Output: Hello!

# **4. Function Parameters and Arguments**

Parameters let functions accept input values. These values are known as arguments when passed during the function call.  
  
- Positional: matched by order.  
- Keyword: matched by name.  
- Default: used if no value is provided.  
- \*args: accepts multiple positional arguments.  
- \*\*kwargs: accepts multiple keyword arguments.

# **5. Return Statement**

`return` sends a result back to the caller.  
  
Example:  
def multiply(x, y):  
 return x \* y  
result = multiply(3, 4)

# **6. Docstrings and Documentation**

Docstrings describe what a function does. They're written as triple-quoted strings below the function header and can be accessed with help().  
  
Example:  
def greet(name):  
 """Greets the user by name."""  
 print(f"Hello, {name}")

# **7. Function Scope**

Scope defines where variables are accessible.  
- Local: inside function only.  
- Global: outside and accessible everywhere.  
- Use `global` to modify a global variable inside a function.

# **8. Lambda (Anonymous) Functions**

Lambda functions are small, unnamed functions for simple expressions.  
  
Example:  
square = lambda x: x \* x

# **9. Higher-Order Functions**

Functions that take other functions as arguments or return them as results.  
  
Example:  
def speak(func, message):  
 return func(message)

# **10. Nested Functions**

Functions defined inside other functions. They are useful for hiding internal logic and closures.

# **11. Function Composition**

Combining multiple functions where the output of one becomes the input of another.  
  
Example:  
def double(x): return x \* 2  
 def square(x): return x \* x  
 print(square(double(3)))

# **12. map(), filter(), and reduce()**

`map()` applies a function to all items, `filter()` filters items, and `reduce()` collapses items to a single value.  
  
Example:  
map(lambda x: x\*2, [1,2,3])

# **13. List Comprehensions vs Lambda**

List comprehensions are preferred over lambda for readability.  
  
Example:  
[x\*x for x in range(5)] vs list(map(lambda x: x\*x, range(5)))

# **14. Decorators**

Decorators wrap functions to modify behavior. They are useful for logging, access control, and timing.  
  
Example:  
@decorator  
def greet():  
 print("Hello")

# **15. Closures**

Closures are functions that remember variables from their enclosing scopes even after the outer function has finished execution.

# **16. Testing Functions using assert**

`assert` is used to write test cases to ensure functions work correctly.  
  
Example:  
assert add(2, 3) == 5

# **17. Built-in vs User-defined Functions**

- Built-in: already provided (e.g. print(), len()).  
- User-defined: created by you using `def`.

# **18. Recursive Functions**

A function that calls itself to solve smaller instances of a problem.  
  
Example:  
def factorial(n):  
 if n==0: return 1  
 return n \* factorial(n-1)

# **19. Function Annotations (Type Hinting)**

Type hints help specify expected input and output types for better code clarity.  
  
Example:  
def greet(name: str) -> str:  
 return f"Hello, {name}"

# **20. Async Functions**

Async functions allow non-blocking operations. Useful in networking, APIs, and real-time systems.  
  
Example:  
async def fetch():  
 await some\_async\_task()