1️⃣ Introduction to Functions

A **function** in Python is a block of reusable code that performs a specific task. Instead of writing the same lines repeatedly, we define a function once and use it multiple times.

Example (without function):

```
print("Hello Saumya")
print("Hello Saumya")
print("Hello Saumya")
```

Using a function:

```
def greet():
    print("Hello Saumya Singh")

greet()
greet()
```

Practice Questions

1. Write a function named `welcome_message()` that prints "Welcome to Python Programming!" three times.

2. Define a function `inspire()` that prints a motivational quote with your name.

3. Create a function `good_morning()` that prints `"Good Morning, Saumya!"`. Call it twice.

4. Why are functions used in programming? Write two advantages.

2️⃣ Function Definition and Calling

We use the keyword `def` to define a function, and call it by writing its name followed by parentheses `()`.

Syntax:

```
def function_name():
    # code block
```

Example:

```
def welcome():
    print("Welcome to Python, Saumya!")

welcome()
```

Practice Questions

1. Write a function `display_python()` that prints `"Python is Fun!"`.
2. Create a function `learn()` that prints three Python topics.
3. Explain what happens if you call a function before defining it.

3️⃣ Function Parameters & Arguments

Functions can accept **parameters** , data passed from outside.
The values given when calling the function are **arguments**.

Example:

```
def greet(name):
    print("Hello", name)

greet("Saumya Singh")

def add(a, b):
    print("Sum =", a + b)

add(5, 10)
```

Practice Questions

1. Write a function `show_age(name, age)` that prints: `"Saumya Singh is 21 years old."`

2. Create a function `add_numbers(a, b)` that prints both the sum and difference.

3. Write a function `fav_food(food)` that prints `"Saumya loves <food>"`.

4️⃣ Return Statement

The `return` statement is used to send a value back from a function.
After `return`, the function stops execution.

Example:

```
def add(a, b):
    return a + b

result = add(10, 20)
print("Result =", result)
```

Practice Questions

1. Write a function `square(num)` that returns the square of a number.

2. Write a function that takes a string and returns the **count of vowels and consonants** separately.
3. Define a function `convert_to_upper(word)` that returns the uppercase version of the string.

4. Create a function `full_name(fname, lname)` that returns the full name joined with a space.

5️⃣ Default & Keyword Arguments

Default Arguments

If no argument is provided, a **default** value is used.

```
def greet(name="Saumya"):
    print("Hello", name)

greet()
greet("Riya")
```

Keyword Arguments

We can use the parameter name while passing values.

```
def student_info(name, age):
    print(name, "is", age, "years old.")

student_info(age=21, name="Saumya Singh")
```

Practice Questions

1. Define a function `message(text="Keep Learning!")` and call it with and without an argument.

2. Create a function `login(username, password="1234")` that prints the credentials.

6️⃣ Variable Scope (Local vs Global)

- **Local Variable:** Defined inside a function — accessible only within it.
- **Global Variable:** Defined outside any function — accessible everywhere.

Example:

```
x = 10   # global variable

def show():
    x = 5   # local variable
    print("Inside function:", x)

show()
print("Outside function:", x)
```

Practice Questions

1. Write a program with a local variable `score` inside a function and a global one outside.

2. Create a program using `global` keyword to modify a variable from inside a function.

3. Explain the difference between local and global scope in your own words.

7️⃣ None in Python

`None` means *no value*.
If a function does not return anything, it automatically returns `None`.

Example:

```
def greet():
    print("Hello Saumya!")

result = greet()
print(result)
```

Output:

```
Hello Saumya!
None
```

8️⃣ Recursive Functions

A **recursive function** is one that calls itself.
It must have a **base case** (stopping condition) and a **recursive case** (repeats the function).

Example (Factorial):

```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

Example (Fibonacci):

```
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

for i in range(6):
    print(fibonacci(i), end=" ")
```

Practice Questions

1. Write a recursive function that prints numbers from 1 to N.
2. Write a recursive function to calculate the factorial of a number.
3. Write a recursive function to print the Fibonacci series up to N terms.

9️⃣ Recursion vs Iteration

Aspect	**Recursion**	**Iteration**
Definition	Function calls itself repeatedly	Loop repeats statements
Base Case	Must have a base case	Not needed
Memory	Uses more memory (stack)	Memory efficient
Speed	Slower	Faster
Example	Factorial using recursion	Factorial using `for` loop

Example (Iteration):

```
def factorial_iterative(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

```
print(factorial_iterative(5))
```

Practice Questions

1. Write both a recursive and iterative function to find factorial - compare results.

🔹 Practice Set 8 (Consolidated)

1. Write a function to calculate the factorial of a number.
2. Write a recursive function to print numbers from 1 to N.
3. Write a function that checks if a number is prime.
4. Write a recursive function to find the sum of first N natural numbers.
5. Write a function `greet_user(name)` that prints a personalized message for Saumya Singh.
6. Write a recursive program to print the reverse of a string.
7. Write a function to return the largest of 3 numbers.
8. Write a recursive function to print even numbers from 2 to N.
9. Write a function that returns both the sum and average of 5 inputs.
10. Write a program to count vowels in a string using a function.

Summary

Concept	Description	Example
Function	Block of reusable code	`def greet(): print("Hi")`
Arguments	Values passed into function	`greet("Saumya")`
Return	Sends a value back	`return a + b`

Default Arg	Predefined value	`def fun(x=10)`
Global/Local	Scope of variables	`global x`
Recursion	Function calling itself	`factorial(n-1)`