📘 Session 10: Mastering Iteration in Python (Part 2)

Topic: Defining Functions

🔹 What is a Function?

- A **function** is a block of reusable code that performs a specific task.
- Helps in **modularity** (breaking code into parts) and **reusability** (use code multiple times).

👉 General syntax:

```
def function_name(parameters):
    """docstring (optional): description of function"""
    # code block
    return result
```

🔹 Defining and Calling Functions

Example:

```
def greet():
    print("Hello, Welcome to Python!")

# Calling the function
greet()
```

Output:

```
Hello, Welcome to Python!
```

🔹 Function Arguments

1. Positional Arguments

Values passed in the **same order** as defined.

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

print(add(3, 5))  # 8
```

---

2. Keyword Arguments

Values passed using **parameter names** (order doesn’t matter).

```
def introduce(name, age):
    print(f"My name is {name}, I am {age} years old.")

introduce(age=20, name="Vipin")
```

Output:

```
My name is Vipin, I am 20 years old.
```

---

3. Default Arguments

Provide a default value if not passed.

```
def power(base, exp=2):
    return base ** exp

print(power(5))     # 25  (exp defaults to 2)
print(power(5, 3))  # 125
```

---

4. Variable Number of Arguments

- `*args` → collects **positional arguments** into a tuple.
- `**kwargs` → collects **keyword arguments** into a dictionary.

```
def total_sum(*args):
    return sum(args)

print(total_sum(1, 2, 3, 4))  # 10


def student_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

student_info(name="Rahul", age=22, course="AI")
```

🔹 Return Values and Multiple Returns

Single Return

```
def square(x):
    return x * x

print(square(4))  # 16
```

Multiple Returns

```
def min_max(numbers):
    return min(numbers), max(numbers)

low, high = min_max([3, 8, 1, 6])
print("Min:", low, "Max:", high)
```

Output:

```
Min: 1 Max: 8
```

📌 Notes for Word/PDF Format

Session 10 – Defining Functions in Python

- Functions allow reusability and modularity.

- Syntax: `def function_name(parameters):`
- Types of arguments:
  - Positional
  - Keyword
  - Default
  - Variable (`*args`, `**kwargs`)
- Return statement: used to send values back.
- Multiple return values: returned as a tuple.

🎯 Practice Problems

1. Write a function to calculate the factorial of a number.
2. Write a function that takes a list and returns the sum and average.
3. Create a function that accepts `*args` and returns the product of all numbers.
4. Write a function that accepts name and age, but age should have a default value = 18.
5. Write a function that returns both the largest and smallest element of a list.
