Day 11

Function Arguments in Python:

What are Function Arguments?

Arguments are the values we pass to a function when we call it. They help the function work with different data.

Parameters vs Arguments (Important)

```
def add(a, b):   # a, b → PARAMETERS
    return a + b
add(10, 20)     # 10, 20 → ARGUMENTS
```

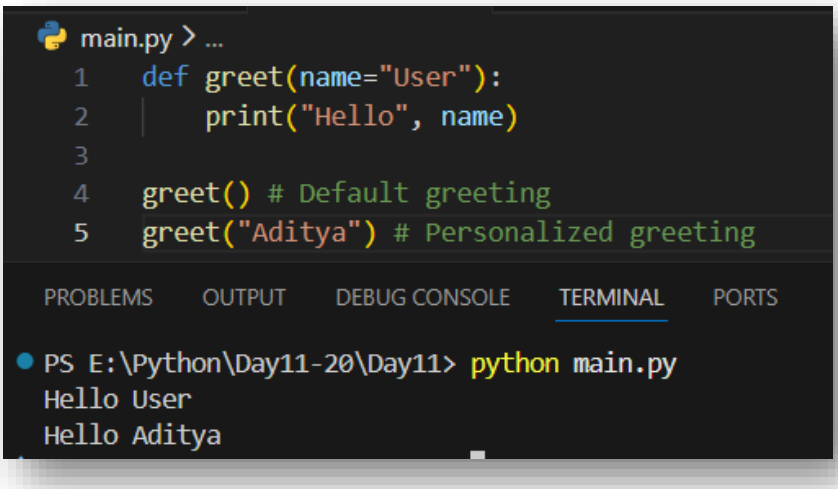
- Parameters → variables in function definition
- Arguments → actual values passed to function

Types of Function Arguments in Python:

Python mainly supports 5 types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable-length Arguments (*args)
5. Keyword Variable-length Arguments (**kwargs)

Default Arguments: A parameter that has a default value.



Example: default arguments another example.

```python
def avergge(a=9, b=1):
    print("The average is:", (a + b) / 2)

avergge() # Default average
avergge(5, 15) # Custom average
avergge(20) # Custom average with one default value of b
avergge(b=30) # Custom average with one default value of a
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day11-20\Day11> python main.py
The average is: 5.0
The average is: 10.0
The average is: 10.5
The average is: 19.5
```

Keyword Arguments: Arguments are passed using parameter names.

```python
def info(name, age):
    print(name, age)

info(age=21, name="Aditya")
info("Aditya", 21)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day11> python main.py
Aditya 21
Aditya 21
```

Required Arguments: Required arguments must always be passed when calling the function.

```python
def greet(name):
    print("Hello,", name)
greet("Alice") # Correct usage with one argument
greet() # Incorrect usage, will raise TypeError
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day11-20\Day11> python main.py
Hello, Alice
Traceback (most recent call last):
  File "E:\Python\Day11-20\Day11\main.py", line 24, in <module>
    greet() # Incorrect usage, will raise TypeError
    ^^^^^^^
TypeError: greet() missing 1 required positional argument: 'name'
```

Variable-length Arguments (*args): Used when we don't know how many arguments will be passed.

```python
def average(*numbers):
    if len(numbers) == 0:
        print("No numbers provided.")
        return
    total = sum(numbers)
    avg = total / len(numbers)
    print("The average is:", avg)
average(10, 20, 30) # Average of three numbers
average(5, 15) # Average of two numbers
average() # No numbers provided
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day11-20\Day11> python main.py
The average is: 20.0
The average is: 10.0
No numbers provided.
```

Keyword Variable-length Arguments (kwargs):** Used to pass many keyword arguments.

```python
def show_info(**kwargs):
    for key, value in kwargs.items():
        print(key, ":", value)
show_info(name="Alice", age=25, city="New York")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day11-20\Day11> python main.py
name : Alice
age : 25
city : New York
```

Example: basic example with 'return'

```python
def average(a, b):
    return (a + b) / 2

result = average(10, 20)
print(result)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR

PS E:\Python\Day11-20\Day11> python main.py
15.0
```

Example: above example without ‘return’

```
49  def average(a, b):
50      avg = (a + b) / 2
51      print(avg)
52  average(10, 20)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS E:\Python\Day11-20\Day11> python main.py
15.0
```

Summary:

- Arguments pass values to functions
- Positional → order matters
- Keyword → order doesn’t matter
- Default → fallback values
- *args → many positional arguments
- **kwargs → many keyword arguments
- Follow correct argument order

--The End--