Day 10: Functions in Python – A Deep Dive with Real-World Examples

1. Understanding Functions in Python

What is a Function?

A function is a reusable block of code that performs a specific task. It helps to **organize** the program, **avoid redundancy**, and **improve readability**.

Why Use Functions?

- **Modularity:** Breaks the program into smaller, manageable parts.
- **Code Reusability:** Write once, use multiple times.
- **Maintainability:** Easier to debug and update.
- **Readability:** Enhances code clarity.

2. Defining a Function in Python

Syntax:

```
def function_name(parameters):
    """Docstring: Briefly describes the function's purpose"""
    # Function body: Code to be executed
    return value  # (Optional)
```

Basic Example:

```
def greet(name):
    """This function greets the person passed as a parameter."""
    print(f"Hello, {name}!")

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

Real-World Example: Automated Welcome Message

```
def welcome_user(username):
    """Function to generate a welcome message for users logging in."""
    return f"Welcome, {username}! We're glad to have you."

message = welcome_user("John")
print(message)  # Output: Welcome, John! We're glad to have you.
```

---

3. Function Arguments and Return Values

Arguments:

- Values **passed to a function** when calling it.
- They allow us to provide input to the function.

Return Values:

- The **output** of a function.
- Functions can return:
    - **Single values**
    - **Multiple values** (using tuples, lists, or dictionaries)

Example: Basic Addition Function

```
def add_numbers(x, y):
    """This function adds two numbers and returns the result."""
    result = x + y
    return result

sum_result = add_numbers(5, 3)
print(sum_result)  # Output: 8
```

Real-World Example: Discount Calculation

```
def apply_discount(price, discount):
    """Calculates the discounted price of a product."""
    discounted_price = price - (price * discount / 100)
    return discounted_price

final_price = apply_discount(1000, 10)
print(f"Final price after discount: ${final_price}")  # Output: Final price after discount: $900.0
```

---

4. Default and Keyword Arguments

Default Arguments

- Arguments with **predefined default values**.
- If no value is passed during the function call, the default is used.

Keyword Arguments

- Arguments are passed using key=value pairs.
- Allows specifying arguments **in any order**.

Example: Greeting Function with Default & Keyword Arguments

```
def greet(name, greeting="Hello"):
    """Greets the person with a custom or default greeting."""
    print(f"{greeting}, {name}!")

greet("Bob")  # Output: Hello, Bob! (Uses default greeting)
greet("Charlie", greeting="Hi")  # Output: Hi, Charlie! (Keyword argument)
```

Real-World Example: Restaurant Order System

```
def order_food(item, quantity=1, drink="Water"):
    """Function to process a food order with default and keyword arguments."""
    print(f"Order: {quantity} x {item} with {drink}")

order_food("Burger")  # Output: Order: 1 x Burger with Water (Default values used)
order_food("Pizza", 2, drink="Soda")  # Output: Order: 2 x Pizza with Soda
```

---

5. Returning Multiple Values

A function can return multiple values using a **tuple** or **dictionary**.

Example: Student Report Generation

```
def student_report(name, math, science, english):
    """Returns a student's name and average marks."""
    avg_marks = (math + science + english) / 3
    return name, avg_marks
student, average = student_report("Alice", 85, 90, 80)
print(f"Student: {student}, Average Marks: {average}")
# Output: Student: Alice, Average Marks: 85.0
```

Real-World Example: Bank Account Balance Inquiry

```
def account_details(account_number):
    """Returns account holder name and balance."""
    accounts = {
        101: ("Alice Johnson", 5000),
        102: ("Bob Smith", 3200)
    }
    return accounts.get(account_number, ("Unknown", 0))

name, balance = account_details(101)
print(f"Account Holder: {name}, Balance: ${balance}")
# Output: Account Holder: Alice Johnson, Balance: $5000
```

---

6. Key Takeaways

- ✅ **Functions** make code modular, reusable, and easier to maintain.
- ✅ **Arguments & Return Values** allow functions to take input and provide output.
- ✅ **Default & Keyword Arguments** improve function flexibility.
- ✅ **Returning Multiple Values** is useful for handling complex data structures.