Python - Function

Shahriar's Intelligence Academy

Function

Definition:

A function is a **reusable block of code** designed to perform a specific task. It helps to **break a program into smaller, modular parts**, improve code readability and maintainability and **avoid repetition**.

Execution flow:

- **Define the Function**: Write the function using the def keyword.
- **Call the Function**: Execute the function by using its name followed by parentheses.
- **Process the Input (if any)**: The function processes the provided input (arguments).
- **Return the Result (if applicable)**: The function may return a value.

Example:

```
# Define a function
def greet(name):
    print(f"Hello, {name}!")
```

```
# Call the function
greet("Alice")
greet("Bob")
```

Function Development

Syntax:

```
def function_name(parameters):
    """
    Optional docstring (description of the function).
    """
    # Function body (statements to execute)
    return output  # Optional
```

Example:

```
# Function to calculate the square of a number
def square(number):
    return number * number

# Calling the function
result = square(5)
print(f"The square of 5 is {result}")
```

Steps:

- Use the def keyword followed by the function name and parentheses ().
- Include parameters inside the parentheses (optional).
- Write the logic inside the function body, indented properly.
- Use the return keyword to send a value back to the caller (if needed).

Function – Return v/s Print

Differences:

Aspect	`return`	`print`
Purpose	Returns a value to the caller for further use.	Displays a value to the console but does not return it.
Reusability	The returned value can be used in expressions or assigned to variables.	No value is sent back to the program.
Example Use Case	Performing calculations or data processing.	Debugging or displaying output.

Function – Return v/s Print

Examples:

```python
# Function with return
def add(a, b):
    return a + b
```

```python
# Usage
result = add(3, 4)  # Returns 7
print(result)       # Outputs: 7
```

```python
# Function with print
def display_sum(a, b):
    print(f"The sum is: {a + b}")
```

```python
display_sum(3, 4)   # Outputs: The sum is: 7
```

Function – Multiple Tasks

Example:

```python
# Function performing multiple tasks: validation and calculation
def process_temperature(temp):
    if temp < -273.15:
        return "Invalid temperature! Below absolute zero."
    elif temp > 100:
        return "Temperature exceeds boiling point!"
    else:
        return f"Temperature in Fahrenheit: {temp * 9/5 + 32}"

# Calling the function
print(process_temperature(-300))  # Invalid temperature!
print(process_temperature(50))   # Temperature in Fahrenheit: 122.0
```

Function Arguments

Definition:

Arguments are **inputs passed to a function** when it is called. Use them to make functions more flexible and dynamic.

.

When to use:

When a function **needs data from the caller** to perform its task.

Examples:

```
# Function with arguments
def greet_person(name, age):
    print(f"Hello, {name}! You are {age} years old.")

# Calling the function with arguments
greet_person("Alice", 30)
greet_person("Bob", 25)
```

Global variable

Definition:

Variables **declared outside any function** and accessible **throughout the program**.

When to use:

When data **needs to be shared and modified** across multiple functions.

Examples:

```
# Global variable
counter = 0

def increment():
    global counter
    counter += 1
    print(f"Counter incremented to {counter}")
```

```
def reset():
    global counter
    counter = 0
    print("Counter reset to 0")

# Using global variable
increment()  # Counter incremented to 1
increment()  # Counter incremented to 2
reset()      # Counter reset to 0
```

Local variable

Definition:

Variables **declared inside a function** and accessible **only within that function**.

When to use:

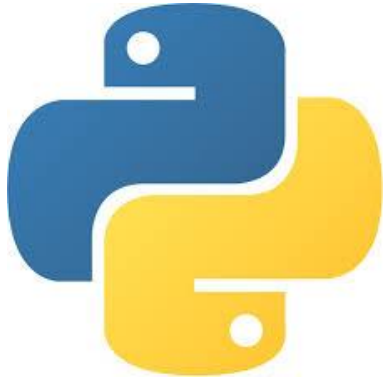
When **data is temporary** and only **relevant within the scope** of a specific function..

Examples:

```
def calculate_area(radius):
    pi = 3.14159  # Local variable
    return pi * radius ** 2

# Calling the function
area = calculate_area(5)
print(f"The area is: {area}")
# Note: `pi` is not accessible outside the function
```

Thank You!



Shahriar's Intelligence Academy