Chapter 08 Function

Python Notes by Wahab

Functions

Functions in Python are blocks of reusable code that execute a specific task. They help make code more modular, readable, and easier to maintain.

Syntax:

```python
def function_name(parameters):
    # Code block
    return result  # Optional
```

Example:

```python
def greet():
    print("Hello, world!")
```

Defining Functions

To create a function, use the def keyword, followed by the function name and parentheses. Any inputs go inside the parentheses, followed by a colon. The function's code block is indented.

```python
def function_name():
   # Code here
   pass
```

Function Call

In Python, a **function call** is how you execute a function that has been defined. It involves using the function name followed by parentheses, optionally passing arguments.

Syntax

```
function_name(arguments)
```

`function_name:` The name of the function you want to call.
`arguments:` Optional values passed to the function for processing.

Example

```python
def greet(name):
    print(f"Hello, {name}!")
```

```
# Function call
greet("mariyam")   #calling function by passing argument mariyam
```

Type of functions

In Python there are two type of function.

1. builtin function **(already present in python)**
2. user define function **(created by user)**
   Example of built infunctin are `len()` , `range()`, `print()` etc.
   Example of user define functin is `greet()` that we created above.

Function Arguments

Function arguments allow data to be passed into a function. In Python, there are several ways to handle arguments, making functions flexible and versatile.

Positional Arguments

Arguments are assigned based on their position.

```
def add(x, y):
    return x + y

result = add(5, 10)  # Output: 15
```

Default Arguments

Default values are used if no argument is provided by the caller.

```
def person_info(name, age):
    print(f"Name: {name}, Age: {age}")

person_info(age=18, name="Wahab")  # Output: Name: Wahab, Age: 18
```

Keyword Arguments

Arguments are specified by parameter name, allowing them to be in any order.

```
def person_info(name, age):
    print(f"Name: {name}, Age: {age}")

person_info(age=18, name="Wahab")  # Output: Name: Wahab, Age: 18
```

Arbitrary Positional Arguments (*args)

Accepts multiple positional arguments as a tuple, allowing functions to handle an unknown number of inputs.

```python
def add(*numbers):
    return sum(numbers)

result = add(5, 10, 15)  # Output: 30
```

Arbitrary Keyword Arguments (**kwargs)

Accepts multiple keyword arguments as a dictionary, which is useful when you don't know all possible argument names in advance.

```python
def display_info(**info):
    for key, value in info.items():
        print(f"{key}: {value}")

display_info(name="Wahab", age=18, city="Rawalpindi")
```

Summary Tabel

Argument Type	Syntax	Description
Positional	`def func(x, y)`	Arguments passed by position
Default	`def func(x=0)`	Arguments with default values
Keyword	`func(y=val)`	Arguments passed by parameter name
Arbitrary Positional	`def func(*args)`	Accepts multiple arguments as a tuple
Arbitrary Keyword	`def func(**kwargs)`	Accepts multiple arguments as a dictionary

Return Values

Functions can return a value using the return statement. The return statement also ends the function execution.
NOTE: Functions can return multiple values using tuples.

Example

```python
def divide_and_remainder(x, y):
    return x // y, x % y

quotient, remainder = divide_and_remainder(10, 3)  # Quotient: 3, Remainder: 1
```