Function (Part-2)

Advanced

Function Advanced

Part 1

ALREADY Covered

Part 2

Covering NOW

Table of contents

01

Quick Recap

02

Intermediate Stuff

01

Quick Recap

Quick Recap

Function: A reusable block of code that runs when called.

Inputs (Parameters): Functions can accept inputs to work with.

Output (Return): Functions can send a result back after execution.

Types:

- **Built-in Functions** (e.g., print(), len(), type())
- **User-defined Functions** (custom functions you create)

Why Use Functions:

- Code Reusability
- Modularity
- Code Organization
- Easier Debugging

Quick Recap

Defining a Function:

- Use **def** keyword with **function name** and **parameters**.
- (Optional) Add a **docstring** to describe the function.
- (Optional) Use **return** to send back a **result**.

Quick Recap

<table>
<tr><th>Python</th><th>Output</th></tr>
<tr><td><pre>
def greet(name):
    """
    This function greets the person passed as
    an argument.
    """
    return f"Hello, {name}!"

# Calling the function
print(greet(“Amar"))
</pre></td><td><pre>
Hello, Amar!
</pre></td></tr>
</table>

Quick Recap

Python

```python
def greet(name, age):
    """
    This function greets the person passed as an argument.
    """
    return f"Hello, {name}! You are {age} years old."

# Calling the function
print(greet(“Amar“, 25))
```

Output

```
Hello, Amar!
You are 25
years old.
```

02

Intermediate Stuff

Intermediate Stuff

- **Arguments** are the values you **pass inside** the **parentheses ()** when calling a function.

- **Parameters** are the variables you define **inside the function definition** to accept those values.

- **Simple Meaning:**
  • **Parameters** are like **empty containers**,
  • **Arguments** are the **actual items** you put inside when calling.

Python

```python
def greet(name): # 'name' is a parameter
    print(f"Hello, {name}!")

greet("Amar") # "Amar" is an argument
```

Types of Function Arguments

Type	Description
Positional Arguments	Passed by **order**. Must be in the correct position.
Keyword Arguments	Passed by **name**, so order doesn't matter.
Default Arguments	Parameters that already have a **default value** if no argument is provided.
Variable-length Arguments	Accepts **multiple values** using ***args** or **multiple keyworded values** using ****kwargs.**

Positional Arguments

- Arguments are **matched** to parameters by their **position**.
- Order **matters**!

Python

```python
def sub(a, b):
    return a - b

print(sub(5, 10))
```

Output

```
-5
```

(Here, a=3 and b=5 based on their positions.)

Keyword Arguments

- You can pass arguments by **specifying parameter names**.
- **Order doesn't matter** when using keyword arguments.

Python

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

introduce(age = 25, name = ”Amar”)
```

Output

```
My name is Amar
and I am 25 years
old.
```

(Since you used name= and age=, order doesn't matter.)

Default Arguments

- You can assign a **default value** to a **parameter**.
- If **no value** is provided during the function call, the **default value** is used.

Python

```python
def greet(name = ”Guest”):
    print(f"Hello, {name}. ")

greet()
greet(name = ”Amar”)
```

Output

```
Hello, Guest.
Hello, Amar.
```

Default Arguments

Important Rule:

- Parameters with default values **must come after** parameters without default values.

Python

```python
# Correct
def greet(name, age = 18):
    pass
```

```python
# Incorrect:
# SyntaxError: parameter without a default follows parameter with a default
def greet(age = 18, name):
    pass
```

Variable-length Arguments

- Sometimes you **don't** know **how many arguments** you'll **pass**.
- Python gives **two special ways**:
    - ***args** (Non-keyworded variable arguments)
    - ****kwargs** (Keyworded variable arguments)

Variable-length Arguments

***args (Non-keyworded variable arguments):**

- Accepts **multiple positional arguments** as a **tuple**.
- Used when you want to **accept many inputs**.

Python

```python
def add_numbers(*args):
    total = sum(args)
    return total

print(add_numbers(2, 4, 6, 8))
```

Output

```
20
```

Variable-length Arguments

**kwargs (Keyworded variable arguments)

- Accepts **multiple keyword arguments** as a **dictionary**.
- Good when you want **named values**.

Python

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

info(name="Amar", age=25, city="Delhi")
```

Output

```
name: Amar
age: 25
city: Delhi
```

Mixing Arguments

- You can **combine** all these types together while **defining a function**.

- **Order to remember:**
  Normal parameters → *args → default parameters → **kwargs

Python

```python
def demo(a, b, *args, city="Unknown", **kwargs):
    print(a, b)
    print(args)
    print(city)
    print(kwargs)

demo(1, 2, 3, 4, 5, city = "Delhi", country = "India",
pincode = 110001)
```

Output

```
1 2
(3, 4, 5)
Delhi
{'country': 'India',
'pincode': 110001}
```

Summary

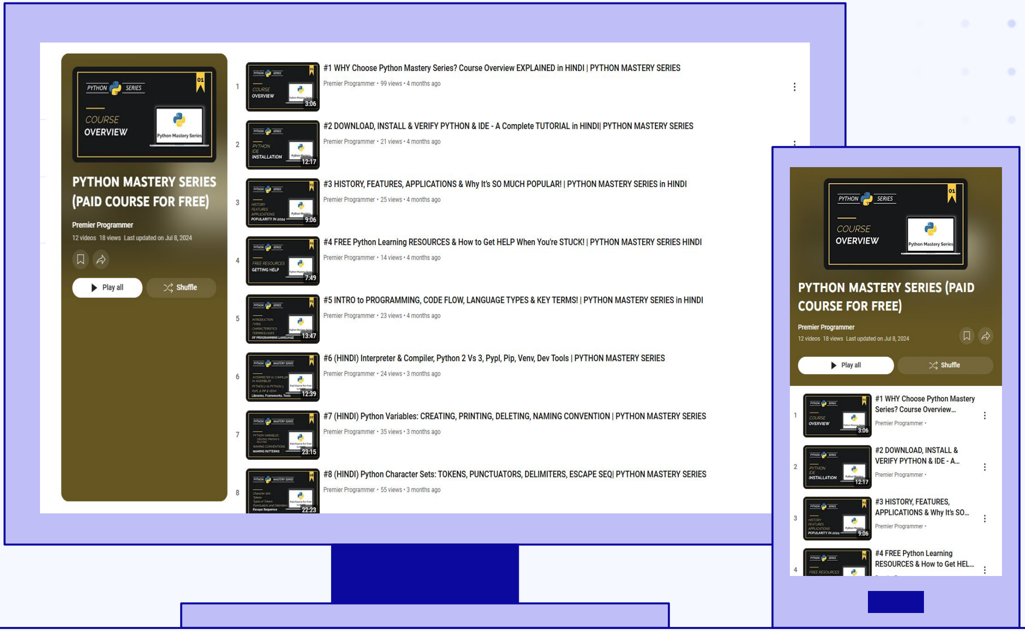
- **Positional arguments** must come **before** keyword arguments.
- **Default parameters** must be defined **after** normal parameters.
- ***args** collects extra **positional arguments** as a **tuple**.
- ****kwargs** collects extra **keyword arguments** as a **dictionary**.

Summary

Type	Definition	Calling
Positional Arguments	def f(a, b)	f(1, 2)
Keyword Arguments	def f(a, b)	f(b=2, a=1)
Default Arguments	def f(a, b=2)	f(1) or f(1, 3)
***args**	def f(*args)	f(1,2,3)
****kwargs**	def f(**kwargs)	f(name="Amar", age=25)

WATCH

Level up your coding with each episode in this focused Python series.



PYTHON MASTERY SERIES (PAID COURSE FOR FREE)

Premier Programmer

12 videos 18 views Last updated on Jul 8, 2024

▶ Play all | Shuffle

1 #1 WHY Choose Python Mastery Series? Course Overview EXPLAINED in HINDI | PYTHON MASTERY SERIES
Premier Programmer • 99 views • 4 months ago

2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A Complete TUTORIAL in HINDI| PYTHON MASTERY SERIES
Premier Programmer • 21 views • 4 months ago

3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO MUCH POPULAR! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 25 views • 4 months ago

4 #4 FREE Python Learning RESOURCES & How to Get HELP When You're STUCK! | PYTHON MASTERY SERIES HINDI
Premier Programmer • 14 views • 4 months ago

5 #5 INTRO to PROGRAMMING, CODE FLOW, LANGUAGE TYPES & KEY TERMS! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 23 views • 4 months ago

6 #6 (HINDI) Interpreter & Compiler, Python 2 Vs 3, Pypl, Pip, Venv, Dev Tools | PYTHON MASTERY SERIES
Premier Programmer • 24 views • 3 months ago

7 #7 (HINDI) Python Variables: CREATING, PRINTING, DELETING, NAMING CONVENTION | PYTHON MASTERY SERIES
Premier Programmer • 35 views • 3 months ago

8 #8 (HINDI) Python Character Sets: TOKENS, PUNCTUATORS, DELIMITERS, ESCAPE SEQ| PYTHON MASTERY SERIES
Premier Programmer • 55 views • 3 months ago

PYTHON MASTERY SERIES (PAID COURSE FOR FREE)

Premier Programmer

12 videos 18 views Last updated on Jul 8, 2024

Play all | Shuffle

1 #1 WHY Choose Python Mastery Series? Course Overview...
Premier Programmer •

2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A...
Premier Programmer •

3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO...
Premier Programmer •

4 #4 FREE Python Learning RESOURCES & How to Get HEL...

Next Video!

Function
Annotation