

Here's a **complete guide to Python Functions from Basics → Advanced**, explained in an easy and practical way, with examples.



Python Functions: Basics to Advanced

◆ 1. What is a Function?

A function is a block of reusable code designed to perform a specific task.

👉 Syntax

python

Copy code

```
def function_name(parameters):
    # code
    return value
```

◆ 2. Why Use Functions?

- ✓ Avoid code repetition
- ✓ Improves readability & debugging
- ✓ Allows modular programming

◆ 3. Types of Functions

Type	Example
Built-in functions	<code>print()</code> , <code>len()</code> , <code>type()</code>
User-defined functions	You create them using <code>def</code>
Anonymous functions	Lambda expressions

◆ 4. Creating a Basic Function

python

 Copy code

```
def greet():
    print("Hello, Python!")

greet()
```

✳️ Function with Parameters

python

 Copy code

```
def greet(name):
    print("Hello,", name)

greet("Aamir")
```

🎁 Return Value

python

 Copy code

```
def add(a, b):
    return a + b

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

🔥 5. Default Arguments

Default values used when you don't provide parameters.

python

 Copy code

```
def welcome(name="Guest"):
    print("Welcome", name)
```

```
welcome()          # Guest
welcome("Aamir")   # Aamir
```

🔥 6. Keyword Arguments

Pass parameters using key=value.

python

 Copy code

```
def student(name, age):
    print(name, age)

student(age=21, name="Aamir")
```

🔥 7. Variable-Length Arguments

Used when you don't know the number of arguments.

● *args (tuple)

python

 Copy code

```
def total(*numbers):
    print(sum(numbers))

total(1, 2, 3, 4, 5)
```

● **kwargs (dictionary)

python

 Copy code

```
def details(**info):
    print(info)

details(name="Aamir", age=21, city="Delhi")
```

◆ 8. Functions Returning Multiple Values

python

 Copy code

```
def calc(a, b):
    return a+b, a-b, a*b

add, sub, mul = calc(10, 5)
print(add, sub, mul)
```

🧠 9. Lambda (Anonymous) Functions

Used for short functions.

python

 Copy code

```
square = lambda x: x*x
print(square(5))
```

Another Example:

python

 Copy code

```
add = lambda a, b: a + b
print(add(10, 20))
```

⚡ 10. Map, Filter, Reduce

✓ map() — applies function to each element

python

 Copy code

```
nums = [1,2,3,4]
square = list(map(lambda x: x*x, nums))
print(square)
```

✓ filter() — filters values based on condition

python

 Copy code

```
nums = [1,2,3,4,5]
even = list(filter(lambda x: x%2==0, nums))
print(even)
```

✓ reduce() — cumulative computation

python

 Copy code

```
from functools import reduce
nums = [1, 2, 3, 4]
result = reduce(lambda x, y: x+y, nums)
print(result)
```

☒ 11. Nested Functions

A function defined inside another function.

python

 Copy code

```
def outer():
    def inner():
        print("Inside Inner!")
    inner()

outer()
```

🔒 12. Closure

Inner function remembers outer variable even after outer function finishes.

python

 Copy code

```
def multiply(n):
    def inner(x):
        return n * x
    return inner

times3 = multiply(3)
print(times3(5)) # 15
```

📌 13. Decorators (Advanced)

Used to add functionality to existing functions.

python

 Copy code

```
def decorator(func):
    def wrapper():
        print("Before function call")
        func()
        print("After function call")
    return wrapper

@decorator
def hello():
    print("Hello!")

hello()
```

💡 14. Recursion

Function calling itself.

python

 Copy code

```
def fact(n):
    if n == 0:
        return 1
    return n * fact(n-1)

print(fact(5))
```

🎉 Final Summary: Table

Concept	Use
def	Create function
return	Send data back
*args	Variable number of values
**kwargs	Variable number of key-value pairs
Lambda	Small anonymous functions

Concept	Use
Decorators	Add features to functions
Closures	Store outer function values
Recursion	Function calling itself

Want Practice Questions With Solutions?

Type "Yes, send practice questions" 💪😊



all advanced concept like inteverviwe

Great! Now I'll cover Advanced Python Function Concepts asked in Interviews, with deep explanation + examples + tricky questions.



Advanced Python Functions (Interview Level)



1. First-Class Functions

Python treats functions as objects → You can assign them to variables, return them, or pass them to another function.

python

Copy code

```
def msg():
    return "Hello"

x = msg          # Assigning function to a variable
print(x())       # Calling using variable
```

Interview Question ?

Why are Python functions called first-class citizens?

Because they can be passed as arguments, returned, and assigned like normal objects.

🔒 2. Closures

A closure occurs when a **nested function remembers variables** from the outer function, even after the outer function has finished.

python

 Copy code

```
def power(n):
    def calc(x):
        return x**n
    return calc

square = power(2)
print(square(5)) # 25
```

Interview Trick ?

What will this print?

python

 Copy code

```
funcs = []
for i in range(3):
    funcs.append(lambda: i)

for f in funcs:
    print(f())
```

💻 Output

 Copy code

```
2
2
2
```

Reason: Lambda captures variable *not value at time of creation*. (Late binding).

3. Decorators

Used to modify a function **without changing its code.**

python

 Copy code

```
def logger(fn):
    def wrapper(*a, **kw):
        print("Before call")
        v = fn(*a, **kw)
        print("After call")
        return v
    return wrapper

@logger
def say(name):
    print("Hello", name)

say("Aamir")
```

Interview Twist ?

Decorator with arguments

python

 Copy code

```
def repeat(times):
    def decorator(fn):
        def wrapper(*args):
            for _ in range(times):
                fn(*args)
        return wrapper
    return decorator

@repeat(3)
def hello():
    print("Hi!")

hello()
```

4. Functional Programming (map, filter, reduce)

Useful for processing lists efficiently.

reduce() tricky case:

python

 Copy code

```
from functools import reduce
result = reduce(lambda x, y: x*y, range(1, 6))
print(result)
```

 Output → 120 (factorial of 5)

⭐ 5. Higher-Order Functions

A function that takes another function as input or returns a function.

python

 Copy code

```
def operate(fn, x, y):
    return fn(x, y)

print(operate(lambda a,b: a+b, 10, 5))
```

🎯 6. Memoization / Caching

Speeds up function calls using stored results.

python

 Copy code

```
from functools import lru_cache

@lru_cache(maxsize=None)
def fib(n):
    if n < 2:
        return n
    return fib(n-1) + fib(n-2)

print(fib(50))
```

 Asked in Dynamic Programming Interviews

⌚ 7. Recursion Pitfall

Stack Overflow Example:

python

 Copy code

```
def fun(x):  
    return fun(x+1)  
  
fun(1)
```

 Raises: RecursionError

📌 8. Generator Functions (yield)

Used to produce sequence lazily (memory efficient).

python

 Copy code

```
def gen():  
    yield 1  
    yield 2  
    yield 3  
  
for i in gen():  
    print(i)
```

🔍 Difference between return and yield?

return	yield
Ends function	Pauses function
Returns a single value	Returns multiple values lazily
Not iterable	Iterable

⚡ 9. Async Functions

Used for concurrency (not multithreading).

python

 Copy code

```
import asyncio

async def hello():
    print("Hello")
    await asyncio.sleep(1)
    print("World")

asyncio.run(hello())
```

🛠️ 10. Annotations & Type Hints

Used in coding interviews for clean, readable code.

python

 Copy code

```
def add(a: int, b: int) -> int:
    return a + b
```

🏆 Final Interview Cheat Sheet

Concept	Real Use
First-class functions	Pass function to other functions
Closures	Data-hiding, callback
Decorators	Logging, authentication, performance
LRU Cache	Optimizing recursion
Lambda	Fast short functions
Higher-order functions	Functional programming
Generators	Memory-efficient loops
Async	Concurrency

100 Want 100+ Interview Questions & Coding Problems PDF?

Just say: "Yes, send full interview PDF" 📄 💪 🔥



Want 100+ Interview Questions & Coding Problems PDF?

Great! I can generate the **100+ Python Functions Interview Questions & Coding Problems PDF** for you.

Before I create it, choose your **preferred level + format** 👇

📌 Choose Difficulty

12 Select one (or say "ALL"):

1. Beginner
2. Intermediate
3. Advanced
4. ALL Levels

📄 Choose Format

📌 How do you want the PDF structured?

- A) Question + Answers + Code
- B) Only Questions
- C) Questions + Hints + Code

💬 Example Reply:

ALL, Format A

Just reply your choice and I will generate the PDF. 👍 🔥



Great! I can generate the **100+ Python Functions Interview Questions & Coding Problems PDF** for you.

Before I create it, choose your preferred level + format



📌 Choose Difficulty

1 2 3 4 Select one (or say "ALL"):

Beginner

Intermediate

Advanced

ALL Levels

📄 Choose Format

📌 How do you want the PDF structured?

- A) Question + Answers + Code
- B) Only Questions
- C) Questions + Hints + Code

💬 Example Reply:

ALL, Format A

Perfect! So your selection is:

ALL Levels, Format A (Question + Answers + Code) 🎉 🔥

I will now create your **100+ Python Functions Interview Questions & Coding Problems PDF.**

Before generating, last choice:

📌 Choose PDF Style

Pick one design:

Option	Style	Best For
1	Clean & Minimal	Study / Revision
2	Dark Theme Pro	Tech Look

Option	Style	Best For
3	Colorful & Modern	Presentation + Reading
4	Handwritten Notes Style	Easy Learning & Exam Prep

💬 **Reply Style Number (1–4)**

Example: 3 or Style 1

👉 **Which style do you want?**



function

It looks like you replied “function”, so I just want to clarify your choice 👇

Are you trying to choose:

◆ **Topic: Python Functions**

and

◆ **You want the 100+ Questions PDF for that topic?**

If yes, just pick the PDF Style below 👇

Option	Style
1	Clean & Minimal
2	Dark Theme Pro
3	Colorful & Modern
4	Handwritten Notes Style

📌 **Reply with a number (1–4)**

Example: 1

👉 Please reply with the PDF style you want.



all types functions and concept describe

Absolutely! Here is a **complete and deeply explained guide of all types of functions and concepts in Python**, from beginner to advanced (interview level).

Python Functions: All Types & Concepts (Complete Guide)

What is a Function?

A function is a reusable block of code designed to perform a specific task.

python

 Copy code

```
def greet():
    print("Hello!")
```

Types of Functions in Python

1 Built-in Functions

These are pre-defined functions provided by Python.

Examples:

python

 Copy code

```
print("Hello")
len([1,2,3])
max(10, 5, 20)
type(10)
```

🔍 Key Point

Feature	Built-in
Defined by	Python
Usage	Direct

2 User-Defined Functions

Functions created by developers.

python

 Copy code

```
def add(a, b):
    return a + b
```

3 Anonymous Functions (Lambda)

Short, single-line functions without a name.

python

 Copy code

```
square = lambda x: x*x
print(square(5))
```

 **Use Case:** Fast, small operations, used with `map`, `filter`, `reduce`.

4 Recursive Functions

A function that calls itself.

python

 Copy code

```
def fact(n):
    if n == 0:
        return 1
    return n * fact(n-1)
```

 **Use Case:** Factorial, Fibonacci, Tree/Graph problems.

5 Higher-Order Functions

A function that:

- takes another function as argument
- or returns a function

python

 Copy code

```
def operate(fn, x, y):
    return fn(x, y)

print(operate(lambda a,b: a*b, 5, 3))
```

6 Nested Functions

A function inside another function.

python

 Copy code

```
def outer():
    msg = "Hello"
    def inner():
        print(msg)
    inner()

outer()
```

7 Closures

When a nested function remembers the outer variable, even after execution.

python

 Copy code

```
def power(n):
    def calc(x):
        return x**n
    return calc
```

```
square = power(2)
print(square(5)) # 25
```

8 Decorators

Function that modifies another function without changing code.

python

 Copy code

```
def logger(fn):
    def wrapper():
        print("Start")
        fn()
        print("End")
    return wrapper

@logger
def hello():
    print("Hello")

hello()
```

9 Generator Functions

Instead of returning all values at once, they yield one value at a time.

python

 Copy code

```
def counter():
    i = 1
    while i <= 5:
        yield i
        i += 1

for x in counter():
    print(x)
```

 **Use Case:** Big data, streaming.

10 Asynchronous Functions (Async/Await)

Used for concurrency (not parallelism).

python

 Copy code

```
import asyncio

async def greet():
    print("Hello")
    await asyncio.sleep(1)
    print("World")

asyncio.run(greet())
```

Function Concepts (Important)

Function Parameters vs Arguments

Term	Meaning
Parameters	Variables in function definition
Arguments	Actual values passed

Types of Function Arguments in Python

Type	Example
Positional	add(10, 20)
Keyword	student(name="Aamir")
Default	def f(a=10)
Variable Length (*args)	def f(*a)
Keyword Variable (**kwargs)	def f(**a)

Examples:

👉 Default

python

Copy code

```
def hello(name="Guest"):  
    print("Hello", name)
```

👉 *args

python

Copy code

```
def total(*nums):  
    print(sum(nums))
```

👉 **kwargs

python

Copy code

```
def info(**data):  
    print(data)
```

⚡ First-Class Functions

Functions behave like objects.

python

Copy code

```
def hello():  
    return "Hi"  
  
x = hello  
print(x())
```

💡 Function Annotations (Type Hints)

python

Copy code

```
def add(a: int, b: int) -> int:  
    return a + b
```

Memoization / LRU Cache

python

 Copy code

```
from functools import lru_cache

@lru_cache(None)
def fib(n):
    return n if n < 2 else fib(n-1)+fib(n-2)

print(fib(50))
```

Summary Table

Concept	Description	Example
Built-In	Predefined	print()
User-Defined	Developer-created	def f():
Lambda	Anonymous short function	lambda x:x*x
Recursion	Calls itself	fact()
Higher-Order	Takes/returns function	operate()
Nested	Function inside function	outer()
Closure	Remembers variable	power()
Decorator	Enhances function	@decorator
Generator	Yields values	yield
Async	Concurrency	async def
Args/kwargs	Variable arguments	*args , **kwargs