**📘 Python Functions – A Clear Guide**

# ✅ What is a Function?

**A function is a reusable block of code/statements that performs a specific task. It helps organize code and avoid repetition.**

**The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing some code again and again for different inputs, we can do the function calls to reuse the code contained in it over and over again.**

**Example:  
-------------  
def my\_function():  
 print("Hello from a function")**

# ▶️ Calling a Function

**Once a function is defined, you can call it like this:**

**my\_function()**

# 🔙 Return Values

**A function can return data using return:**

**def add(a, b):  
 return a + b  
  
result = add(4, 5)  
print(result)**

# 🆎 Function Naming Rules

**- Function names follow variable naming rules**

**- Use lowercase letters and underscores (snake\_case) for readability**

# 📥 Function with Parameters

**You can pass information (parameters) into functions.**

**def greet(name):  
 print("Hello, " + name)  
  
greet("Alice")  
greet("Bob")**

**- name is a parameter (defined in the function)**

**- "Alice" and "Bob" are arguments (passed during call)**

# 🔢 Default Parameter Value

**If a parameter has a default value, it becomes optional.**

**def greet(name = "Guest"):  
 print("Hello, " + name)  
  
greet("Sam") # Hello, Sam  
greet() # Hello, Guest**

# 🧾 Number of Parameters

**Functions must be called with the correct number of arguments:**

**def add(a, b):  
 print(a + b)  
  
add(5, 3) # Correct  
add(5) # ❌ Error: Missing one argument**

# 🧵 Arbitrary Arguments (\*args)

**If you're not sure how many arguments will be passed, use \*args.**

**def my\_function(\*kids):  
 print("The youngest child is " + kids[2])  
  
my\_function("Ram", "Rahim", "Raju")**

**- \*kids collects arguments into a tuple.**

# 🗝️ Keyword Arguments

**You can also pass values with keys, known as keyword arguments.**

**def my\_function(child1, child2):  
 print("The youngest child is " + child2)  
  
my\_function(child1 = "Rahul", child2 = "Riya")**

# 🔐 Arbitrary Keyword Arguments (\*\*kwargs)

**Use \*\*kwargs if you don’t know how many keyword arguments you’ll receive.**

**def my\_function(\*\*kid):  
 print("His last name is " + kid["lname"])  
  
my\_function(fname = "John", lname = "Doe")**

# 🧠 Recursion

**A function can call itself – this is called recursion.**

**Example: Factorial using recursion**

**def factorial(n):  
 if n == 1:  
 return 1  
 else:  
 return n \* factorial(n-1)  
  
print(factorial(5)) # Output: 120**

# 📌 Summary Table

**| Concept | Syntax/Example |  
|--------------------------|------------------------------------------------|  
| Define function | def my\_function(): |  
| Call function | my\_function() |  
| Parameters | def greet(name): |  
| Default Parameter | def greet(name="Guest"): |  
| Return statement | return value |  
| Arbitrary args | def func(\*args): |  
| Arbitrary kwargs | def func(\*\*kwargs): |  
| Recursion | func() calls itself |**

**\*args--🡪assigning as tuple**

**\*\*args-🡪I will pass arguments with keys and without having any count.**

**A function with name is called as named function**

**If that function is not having any name then it is called as anonymous function**

**Lambda Functions in Python**

**What is a Lambda Function?**

* A **tiny, anonymous function** created using the lambda keyword.
* Unlike regular functions (def), **lambdas are used for short, one-time operations**.
* They do **not need a name**, making them quick and efficient

Here are the **proper syntax** and **examples** for **Lambda functions in Python**

**1. Lambda vs Regular Function (Squaring a Number)**

**Syntax**

lambda x: x \* x

* lambda – Keyword to define the function.
* x – Input argument.
* x \* x – Expression that calculates the square of x.

**Example**

**Regular function**

def square(x):

return x \* x

**Equivalent lambda function**

square\_lambda = lambda x: x \* x

print(square\_lambda(4)) # Output: 16

✅ **Lambda saves space** by performing the same task in one line.

**2. Lambda Function with Multiple Inputs (Adding Two Numbers)**

**Syntax**

lambda x, y: x + y

* x, y – Input arguments.
* x + y – Expression that adds both numbers.

**Example**

# Regular function

def add(x, y):

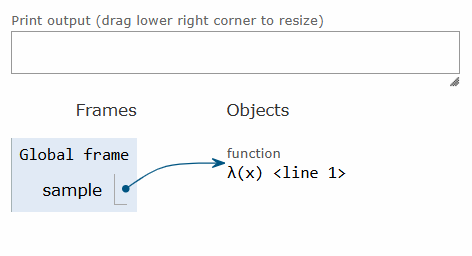
return x + y

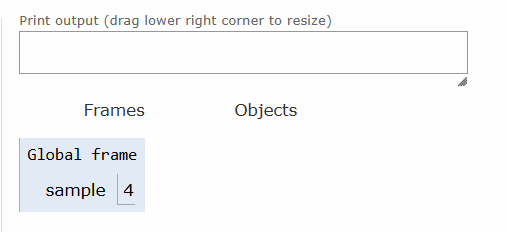
# Equivalent lambda function

add\_lambda = lambda x, y: x + y

print(add\_lambda(3, 5)) # Output: 8

✅ **Short & direct** compared to defining a full function.





1.if we assign a function expression as a value/object to any variable

2.if we pass a function expression as an argument to the other function

3.if a function returns another function

4.if u add a function as a list element

Then function can be said to be as a first-class function

1.function without parameters

2.with parameters(with positional arguments)

3.with default parameters

4.arbitrary arguments (\*args)

5.keyword arguments

6.arbitrarykeywordargs(\*\*kwargs)

7.lambda functions

8.defintion if firstclass functions

9.recursive function

SCOPES-🡪GLOBAL,LOCAL AND ENCLOSED SCOPE,built-in scope