**8. Functions**

1. **Defining functions in Python.**

1. What Is a Function?

* In programming, a function is a named block of code you can call multiple times, optionally with inputs, and which optionally returns an output
* In Python, you define one using def name(params): followed by an indented block. You call it by writing name(args)
* They support code reuse, avoiding duplication Mathematical Foundation
* Function theory in math treats functions as mappings from inputs to outputs.
* Python functions follow this model: they transform inputs into outputs, often without side effects

2. Functional Programming Principles

* In functional programming, pure functions have no side effects and always return the same output for the same input
* Python supports higher-order functions (functions that take or return other functions)—e.g., map, filter, lambda .
* You can perform function composition or currying—techniques from lambda calculus—though Python doesn’t enforce them by default .

3.Flexibility in Python

* Default parameters, \*args, \*\*kwargs for flexible signatures
* Positional-only (/) and keyword-only (\*) parameters for API clarity
* Anonymous functions via lambda—useful for short, inline logic .
* First-class treatment: assign them to variables, pass them to functions, or return them

1. **Different types of functions: with/without parameters, with/without return values**

1. Function with parameters & with return value

* Signature example (C/C++): int add(int a, int b)
* Purpose: Takes input values (arguments) and returns a result.
* Usage: Ideal for computations.
* Example :

def add(a, b):

return a + b

result = add(3, 4) # result == 7

2. No parameters, but returns a value

Example :

import math

def get\_pi():

return math.pi

π = get\_pi()

**Python-specific notes**

* Implicit None: If you omit return, or use return without a value, Python returns None by default
* Always returns something: Even if no return, every function outputs something—the None object .
* Mutable arguments: Python uses “call by object reference”—if you pass a list to a function and modify it, changes persist outside
* Flexible parameters:
  + \*args and \*\*kwargs for arbitrary extra inputs
  + default values, keyword-only and positional-only args are also supported

1. **Anonymous functions (lambda functions).**

What is a lambda function?

A lambda function is an anonymous, single-expression function defined at runtime. It can take any number of arguments but only one expression, and automatically returns the result of that expression

**Syntax :**

**lambda arg1, arg2, …: expression**

* Only one expression: You can’t put statements like if/for blocks (beyond inline expressions)
* Readability concerns: Many Pythonistas (~ including Guido) prefer def over lambdas for anything nontrivial
* Naming confusion: If you're assigning it to a variable and reusing it, just use def for clarity .