Functions: Reusable Blocks

Python for Absolute Beginners · Lesson 15

What you'll learn

- What a function is, and why programmers wrap code in one
- How to define a function with def
- How parameters carry values into a function
- The difference between printing a result and returning one
- How to reuse one function many times

A function is a named block of code

- A function is a block of code with a name, written once and run whenever you call that name
- You have already used functions all along: print(), len(), str()
- def says “here begins a function definition”; the name follows, then brackets
- The indented block underneath is the function's body
- Nothing happens when Python reads the definition — the body runs only when the function is called

Your first function

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

greet()
greet()

Hello, coder!
Hello, coder!
```

Defining is writing the recipe; calling is cooking the meal. Two calls, two runs.

Parameters: values a function receives

- A parameter is a name listed in the definition's brackets
- When you call the function, you hand it a value — called an argument
- The parameter holds that argument while the body runs
- A function can take several parameters, separated by commas
- Arguments match parameters in order: first to first, second to second

Greet with a parameter

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

greet("Mia")
greet("Sam")

Hello, Mia!
Hello, Sam!
```

Same function, different results, because the argument changes what name holds each time.

Two parameters

```
def describe_pet(name, animal):
    print(name + " is a " + animal + ".")

describe_pet("Biscuit", "dog")
describe_pet("Whiskers", "cat")

Biscuit is a dog.
Whiskers is a cat.
```

Read calls left to right, carefully: mix up the order and you get a wrong sentence.

Printing versus returning

- So far every function has printed its result — but often you want a function to hand a value back
- return does that: the caller can store the value or print it directly
- Contrast: once a printing function prints, the value is gone — displayed, but unusable
- Rule of thumb: print when a human needs to see something; return when another part of the program needs to use it
- Code after a return inside a function never runs — returning ends the function at once

Return a value

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

total = add(4, 3)
print(total)
print(add(10, 20))

7
30
```

Return means “the answer of this function is a + b.” Most real functions return.

Why functions matter

- Functions exist for the same reason loops do: write once, use many times
- But they add something loops cannot — a name for an idea
- Greeting becomes one idea, done three times, instead of three repeated lines
- When the greeting needs to change, you change it in one place, not three
- In big programs, that single-change property is what keeps code manageable

One idea, done three times

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

greet("Mia")
greet("Sam")
greet("Alex")

Hello, Mia!
Hello, Sam!
Hello, Alex!
```

Same output as three repeated print lines — but the greeting lives in one place.

Hands-on exercise

- Write a function say_hello(name) that prints a greeting, and call it with three different names
- Write a function double(number) that returns twice its argument; store double(7) in a variable and print it
- Write a function area(width, height) that returns a rectangle's area, and print the area of a 5 by 3 rectangle
- Explain in one sentence why double should return its answer instead of printing it

Key takeaways

- A function is a named block: def defines it, calling its name runs it.
- Parameters receive values; arguments are the values handed in, matched in order.
- print shows a value to a human; return hands it back to the program.
- Code after return never runs — returning ends the function.
- Functions name ideas, so one change fixes every use.

Take the quiz

Take the Lesson 15 quiz — 5 quick questions to check what you learned.

Next lesson

Mini-Project: The Number-Guessing Game