

Functions

Function Definition

Zero or more, comma-separated

def *function_name(parameters)* :
 body

↑
def keyword

↙
One or more statements

2

Form: zero or more, comma-separated

function_name(arguments)

How it's executed:

1. Each argument is an expression. Evaluate these expressions, in order. (The value of each expression is a memory address.)
2. Store those memory addresses in the corresponding parameters.
3. Execute the body of the function.

3

Form: **Return Statement**

return expression

How it's executed:

1. Evaluate the expression.
(The value of the expression is a memory address.)
2. Exit the function, using that memory address as the value of the function call.

NB: The function ends *immediately*. Any remaining statements in it are not executed.

4

If there is no return?

- Then the function ends when there are no statements left to execute.
- The function *does* return a value: the special value **None**.

5

Return vs print

- Return says “get me out now, and send back this value to whoever called me.”
- It does not print anything. But the caller may decide to print the returned value.
- Print says “print this value now.” It does not stop the function.

6

When to use return vs print

- If we give you a specification that says which to do, follow that!
- When you have the freedom to choose, consider how the function will be used.
Examples:
 - If you want to give the “caller” control over whether or not the result is printed, use return.
 - If you want to give the caller the option to use the result later, use return.

7

Namespaces

- A “namespace” keeps track of what names we know about (and can therefore talk about).
- A namespace is created when we begin in the shell or the main block.
- When we call a function, a new namespace is created.
- When we leave a function, its namespace is destroyed.

8

How Python looks up a name

1. Look in the namespace you're in (the "local" one).
If the name is defined there, it's the one!
2. Look in the namespace you started in (the "global" one), that is, the main or the shell.
If the name is defined there, it's the one!
3. If the name is found in neither of these, Python gives up.
What you've done is illegal.

9

How Python looks up a name

1. Look in the namespace you're in (the "local" one).
If the name is defined there, it's the one!
2. Look in the namespace you started in (the "global" one), that is, the main or the shell.
If the name is defined there, it's the one!
3. If the name is found in neither of these, Python gives up.
What you've done is illegal.

But don't access variables
this way: bad style!

10

Why use functions?

- So far, our examples have been mostly silly things, just to show how functions work.
- There are great reasons to use functions:
 - **Reuse**: You can do the same set of steps in multiple places without repeating code.
 - **Chunking**: Bundling up a set of steps and giving them a name makes code easier to understand. Worth doing even if you only call the function *once*.
(Think of 7 ± 2 in psychology.)

11

Function design

- How do you decide on the parameters?
Ask yourself what the function needs to **know** in order to do its job.
- How do you know whether it needs to return a value?
Ask yourself what the function needs to **tell back** (to the code that called it).

12

Function design

- How do you decide on the parameters?

Ask yourself what the function needs to know in order to do its job.

- How do you know whether it needs to return a value?

Ask yourself what the function needs to tell back
(to the code that called it).

Give it everything it needs this way,
rather than via names in the global
namespace