</talentlabs>

CHAPTER 4

Functions

</talentlabs>

AGENDA

- Functions Overview
- Writing Functions
- Using Functions
- Why Use Functions?
- Best Practices
- Variable Scopes

Functions Overview

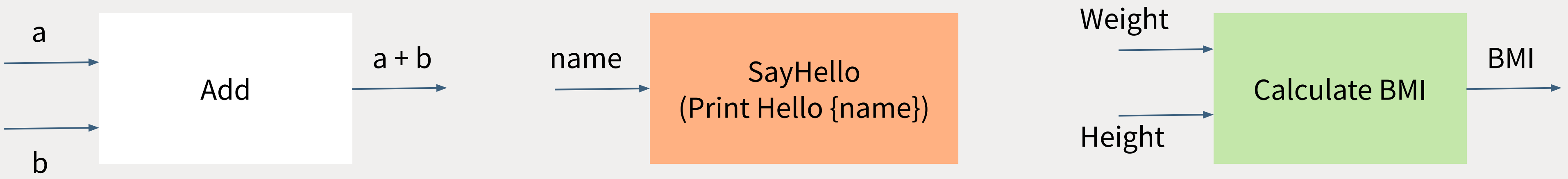
Function

- A piece of packaged code, or say a mini-program
- A programming component that performs a specific task
- An encapsulation of a piece of repeatedly used code
- In some other languages/scenarios, it is named as:

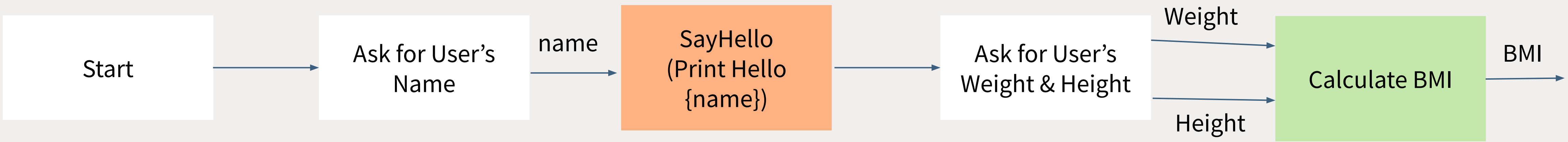
Method **Procedure** **Subroutine**

How Function Works:

Functions:



Main Program (BMI Calculator Program):

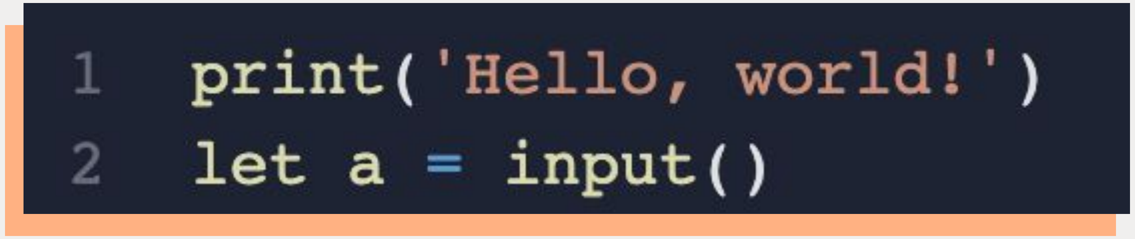


Built-in Functions

</talentlabs>

Python Built-in Function:

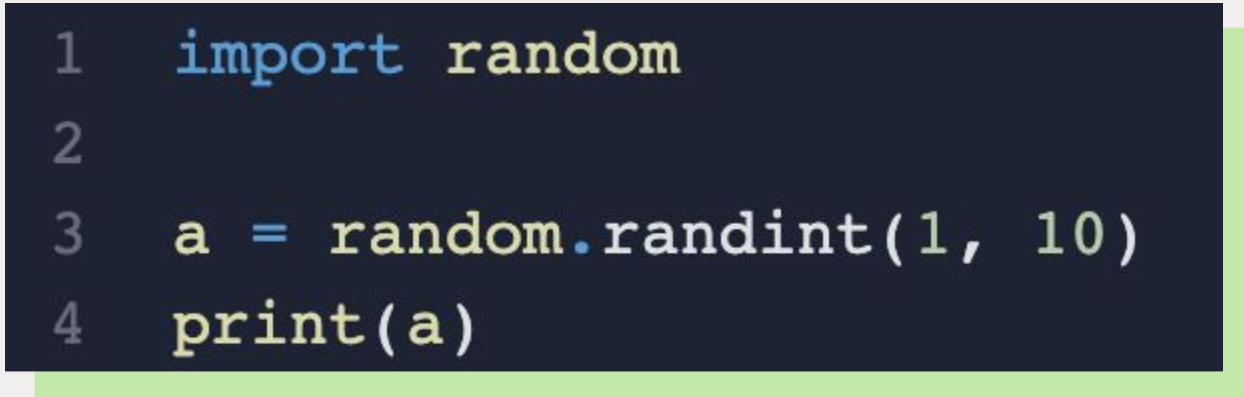
- Commonly used function are already built-in
- No special setup is required



```
print('Hello, world!')
let a = input()
```

Python Standard Libraries

- Commonly used function, also built-in
- But packaged as “Standard Library”
- If you want to use it, you will need to import it first



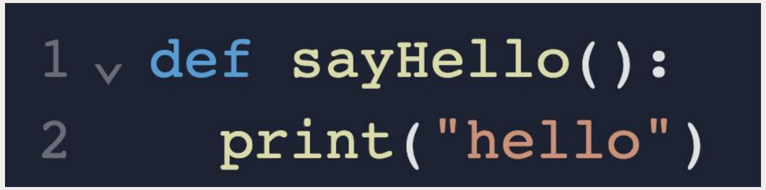
```python
import random

a = random.randint(1, 10)
print(a)
```

Writing Your Own Functions

</talentlabs>

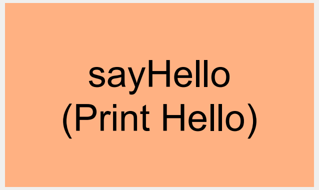
Let’s Write Some Functions

(**WITHOUT** Input, **WITHOUT** Returning Value)



```python
def sayHello():
    print("hello")
```

Python Code

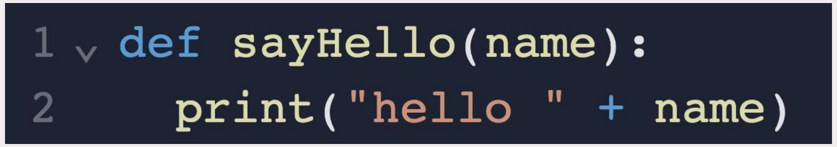


sayHello
(Print Hello)

Diagram View

Let’s Write Some Functions

(WITH Input, WITHOUT Returning Value)



```python
def sayHello(name):
    print("hello " + name)
```

Python Code

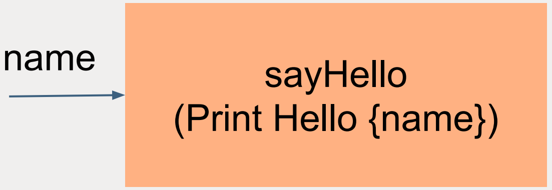
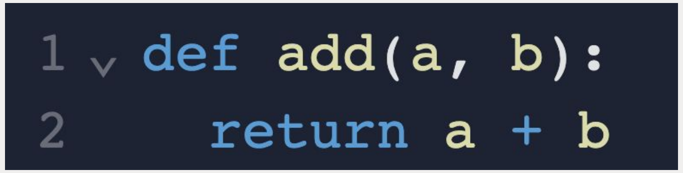


Diagram View

Let’s Write Some Functions

(WITH Input, WITH Returning Value)



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

Python Code

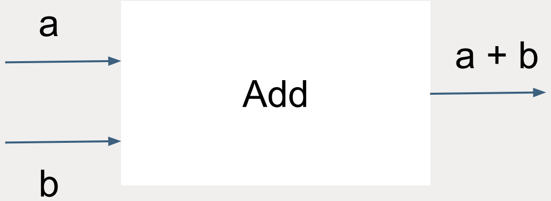
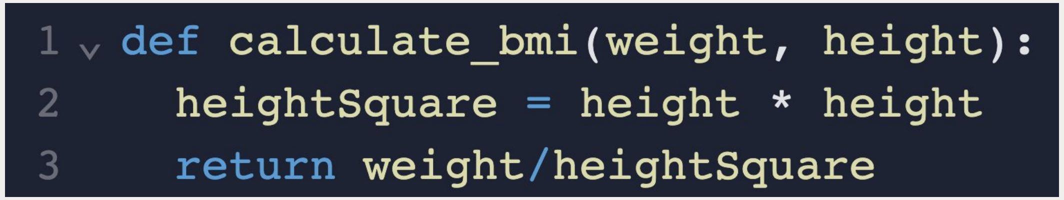


Diagram View

Let’s Write Some Functions

(WITH Input, WITH Returning Value)



```python
def calculate_bmi(weight, height):
    heightSquare = height * height
    return weight/heightSquare
```

Python Code

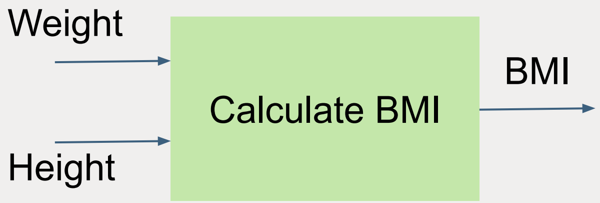
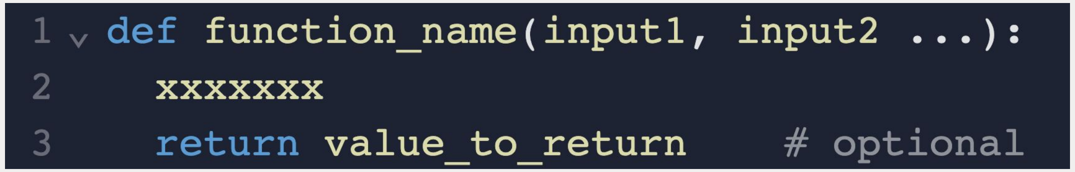


Diagram View

Note: BMI = weight (kg) ÷ height^2 (m)

Summary: How to Build a Function?



```python
def function_name(input1, input2 ...):
    xxxxxxx
    return value_to_return     # optional
```

Python Code

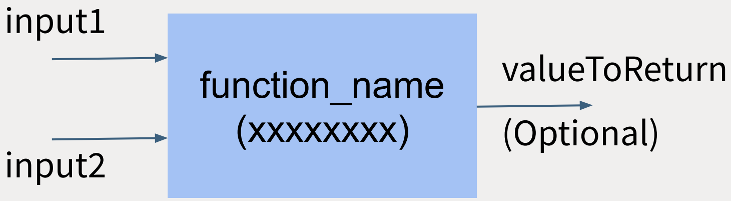
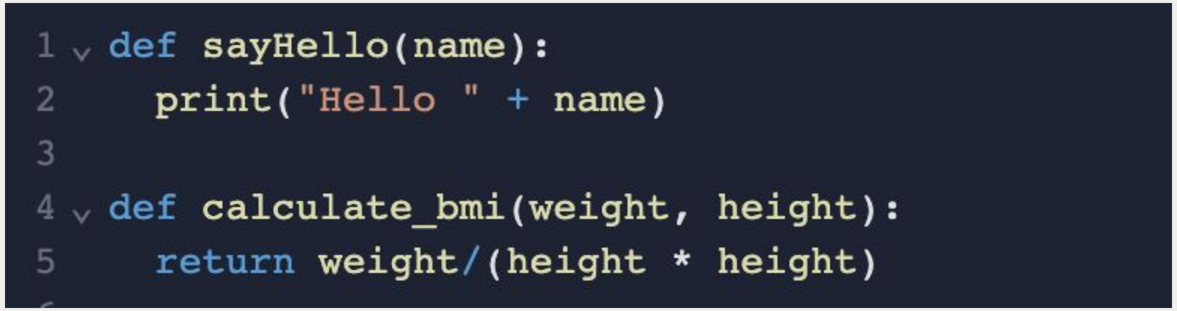


Diagram View

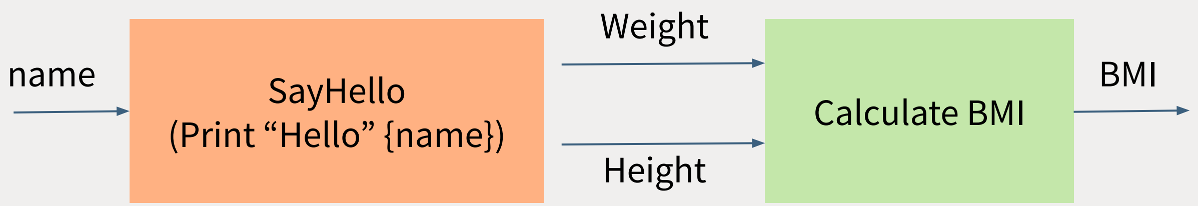
Using Functions

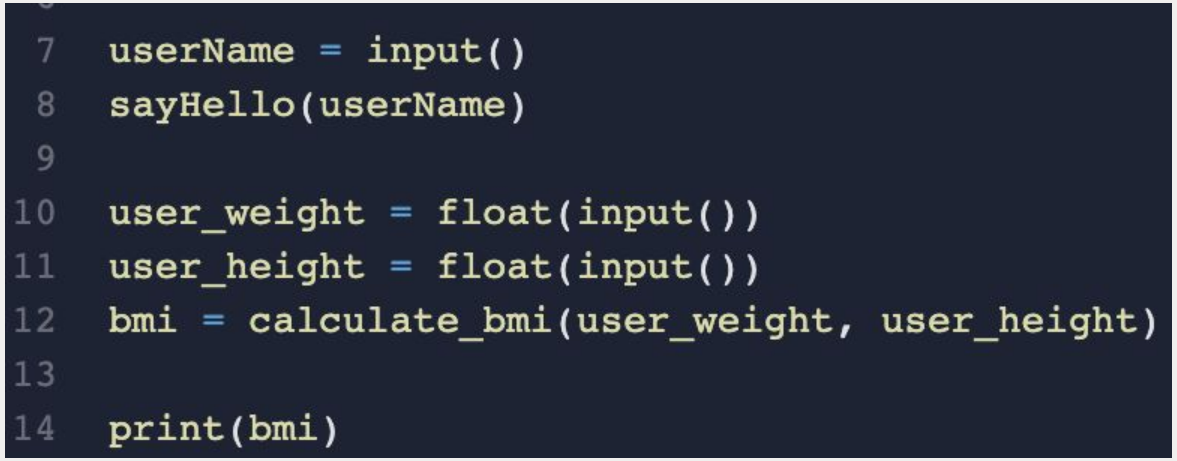
Let’s Use Some Functions



```python
def sayHello(name):
  print("Hello " + name)

def calculate_bmi(weight, height):
  return weight/(height * height)
```

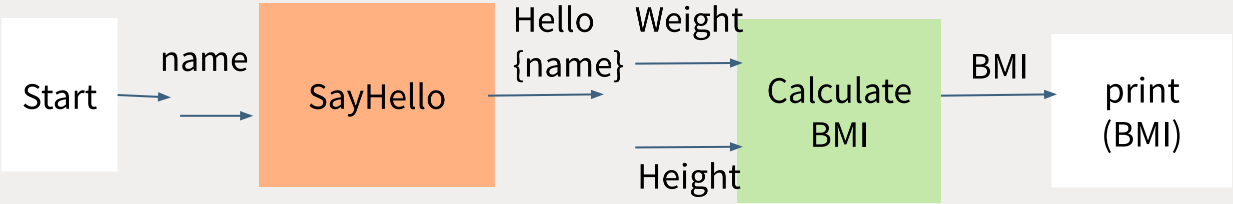
Functions:





```python
userName = input()
sayHello(userName)

user_weight = float(input())
user_height = float(input())
bmi = calculate_bmi(user_weight, user_height)

print(bmi)
```

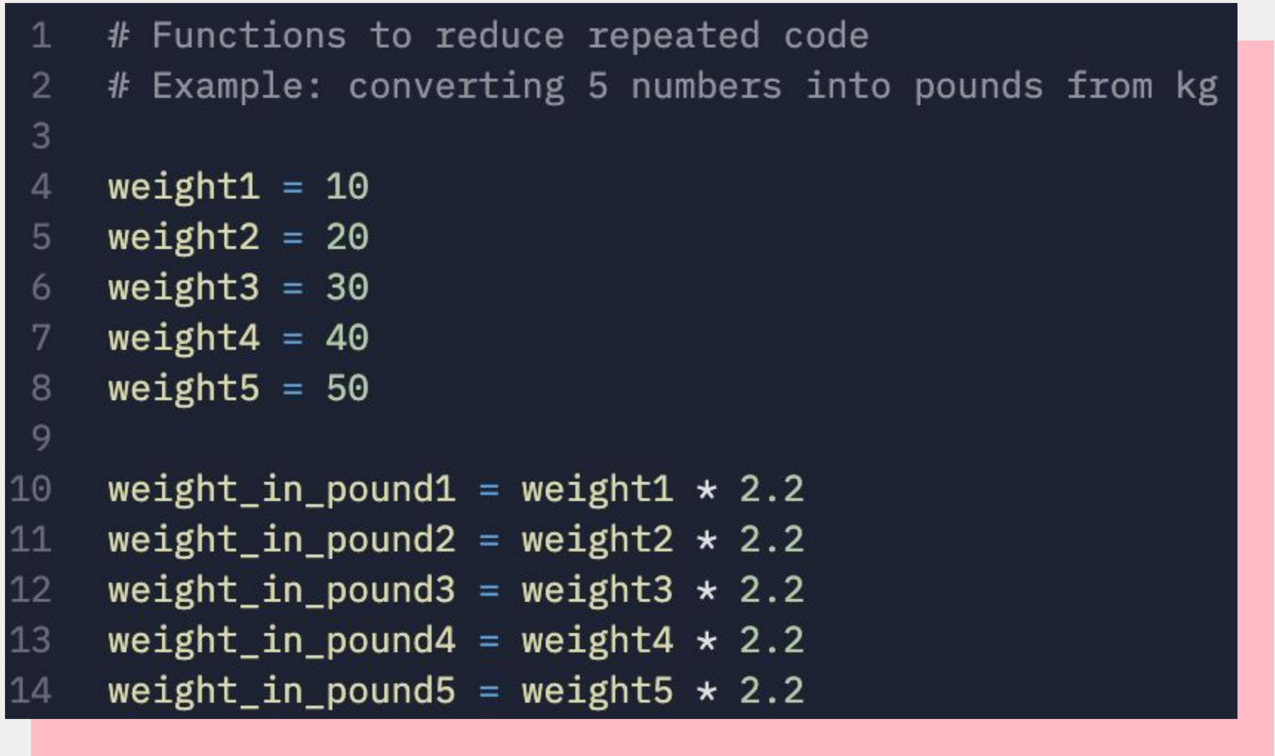
Main Program:



Why Use Functions

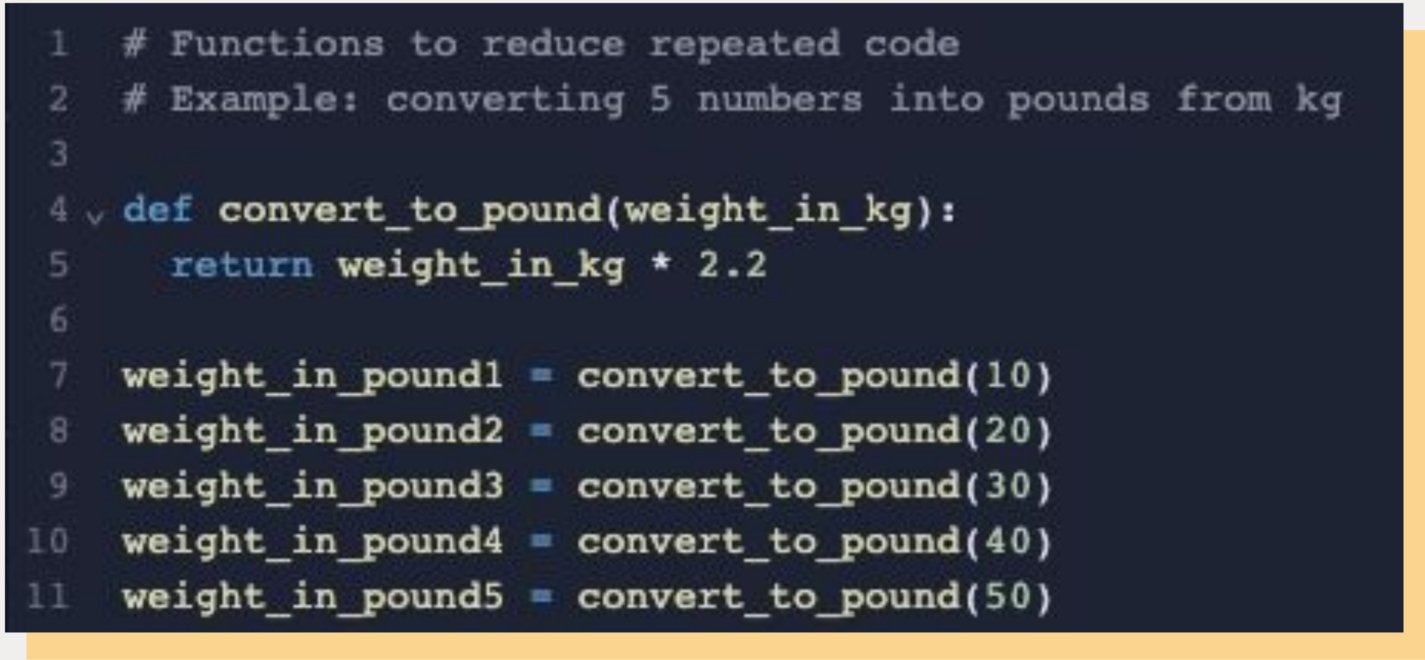
</talentlabs>

Why Build Functions?



```
# Functions to reduce repeated code
# Example: converting 5 numbers into pounds from kg

weight1 = 10
weight2 = 20
weight3 = 30
weight4 = 40
weight5 = 50

weight_in_pound1 = weight1 * 2.2
weight_in_pound2 = weight2 * 2.2
weight_in_pound3 = weight3 * 2.2
weight_in_pound4 = weight4 * 2.2
weight_in_pound5 = weight5 * 2.2
```

Why Build Functions?



```python
# Functions to reduce repeated code
# Example: converting 5 numbers into pounds from kg

def convert_to_pound(weight_in_kg):
  return weight_in_kg * 2.2

weight_in_pound1 = convert_to_pound(10)
weight_in_pound2 = convert_to_pound(20)
weight_in_pound3 = convert_to_pound(30)
weight_in_pound4 = convert_to_pound(40)
weight_in_pound5 = convert_to_pound(50)
```

Best Practices

function_name

- It should be a description of what the function does
- 👍 Example:

convert_kg_to_pound   calculate_taxi_fare   print_hello_world

- it should be meaningful and human-readable
- 👎 Example:

calculate   convert   printReport

</talentlabs>

Input / Parameter / Arguments

- Can have virtually unlimited parameters
- Again, it should be human-readable and meaningful
- 👍 Example:

weight_in_kg   name   age

- 👎 Example:

Input1   weight1, weight2, weight3

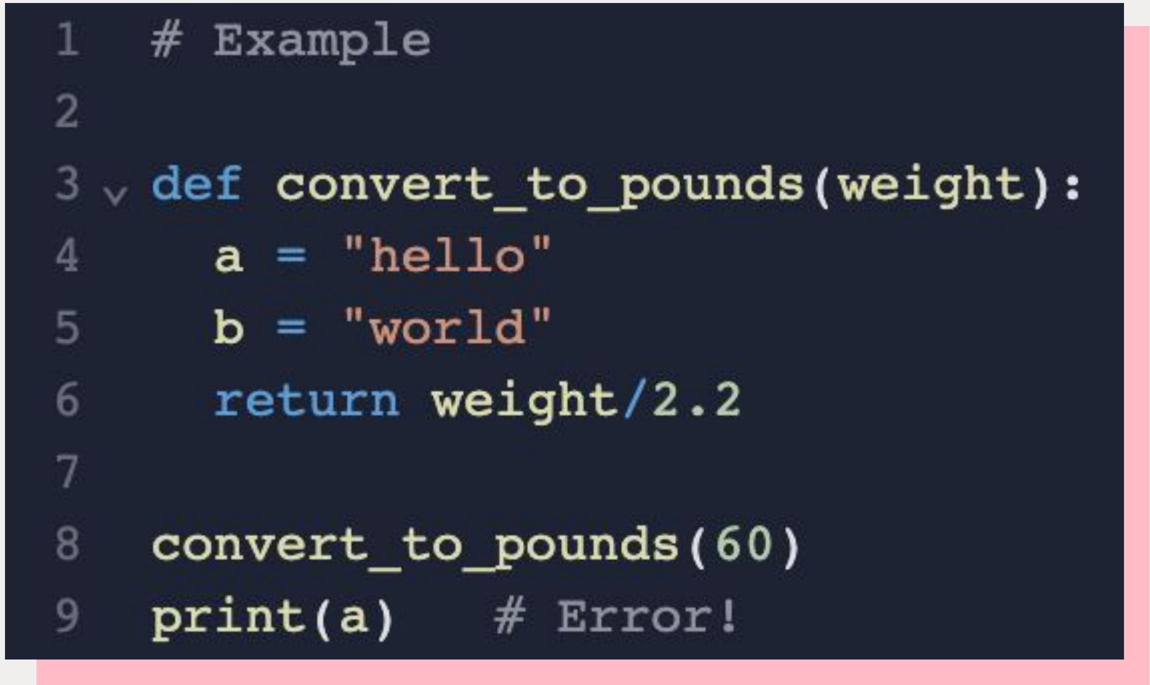
Return Value

- Optional for Python
- If your function is named as convert_to_pounds, the return value should be the **resulting number in pounds**

Variable Scopes

Local Scope

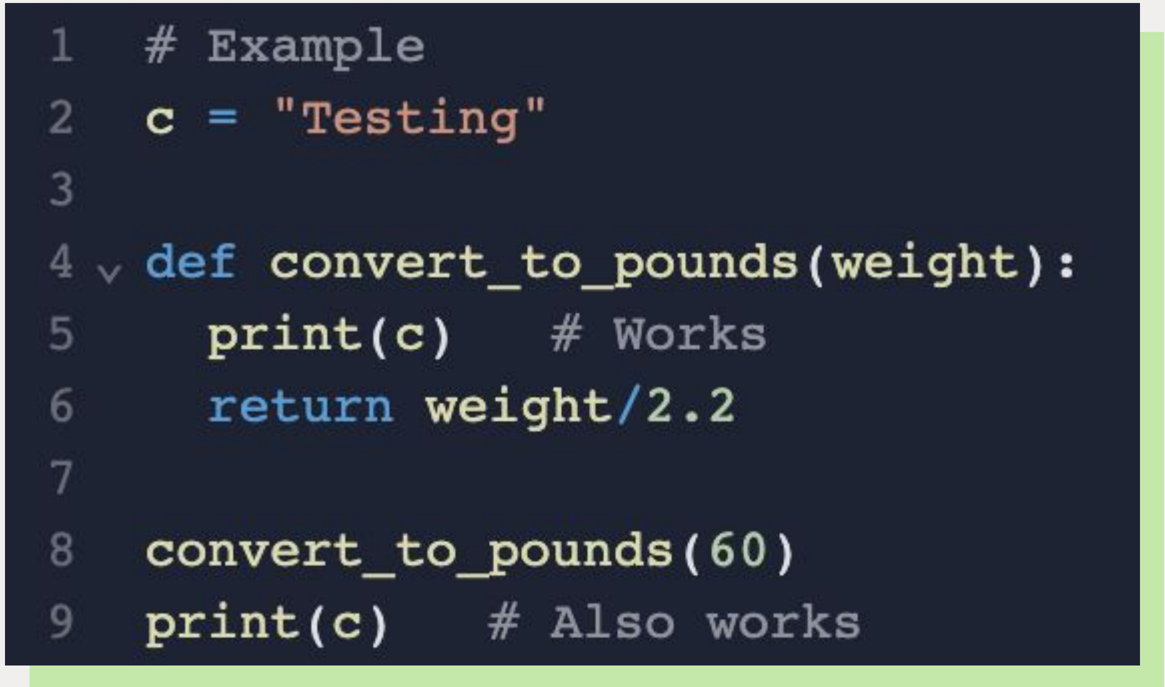
- What is created in the function could not be accessed outside of the function



```python
# Example

def convert_to_pounds(weight):
  a = "hello"
  b = "world"
  return weight/2.2

convert_to_pounds(60)
print(a)    # Error!
```

Global Scope

- What is created outside of the function **could be accessed** inside of the function
- Exception: If the name of the variable is the same, then it would take the value from the “inside” one



```python
# Example
c = "Testing"

def convert_to_pounds(weight):
  print(c)    # Works
  return weight/2.2

convert_to_pounds(60)
print(c)    # Also works
```