The Learning Hub

Python Functions

Education and Training Solutions 2023

The Learning Hub

Functions

The Learning Hub

Introduction to Function

- A function is a block of code, that only runs when it is called for execution.
- Those blocks can be reused to avoid repeating the same instructions.
- You can pass data known as parameters.
- A function can return data as an output.

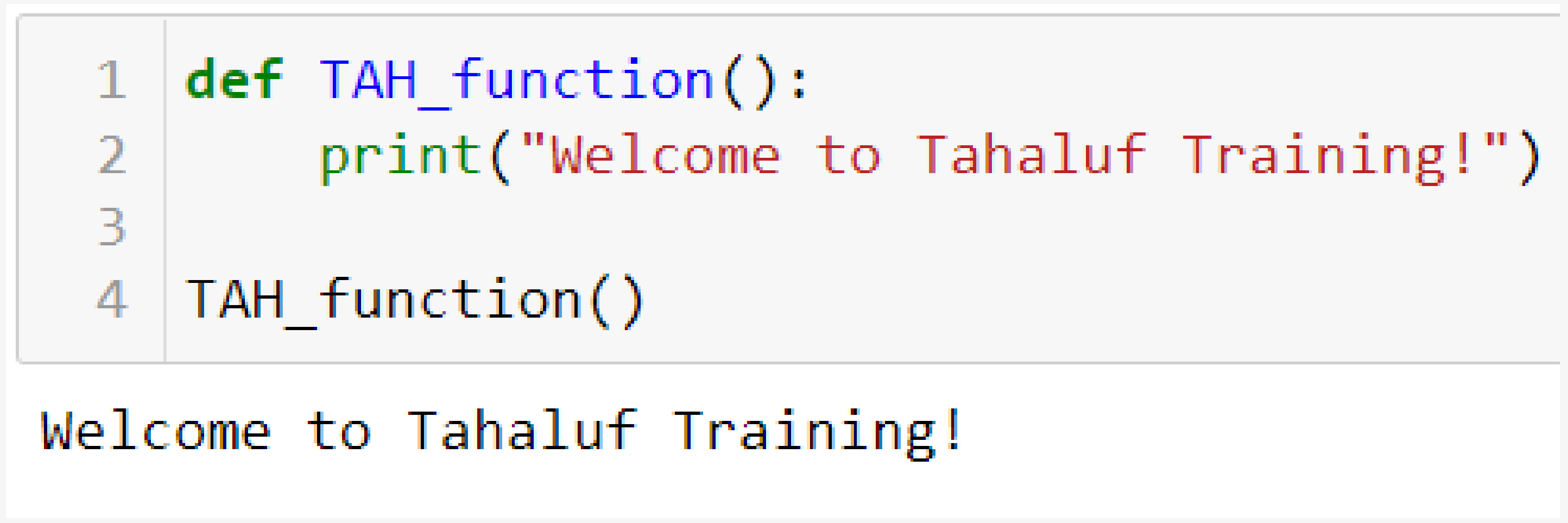
The Learning Hub

Function Indentation

- Indentation is used in python to identify blocks of code.
- Code within the same block should be indented at the same level.
- Python function is one type of code block. All code under a function declaration should be indented to identify it as part of the function.
- There can be additional indentation within a function to handle other statements such as for and if so long as the lines are not indented less than the first line of the function code.

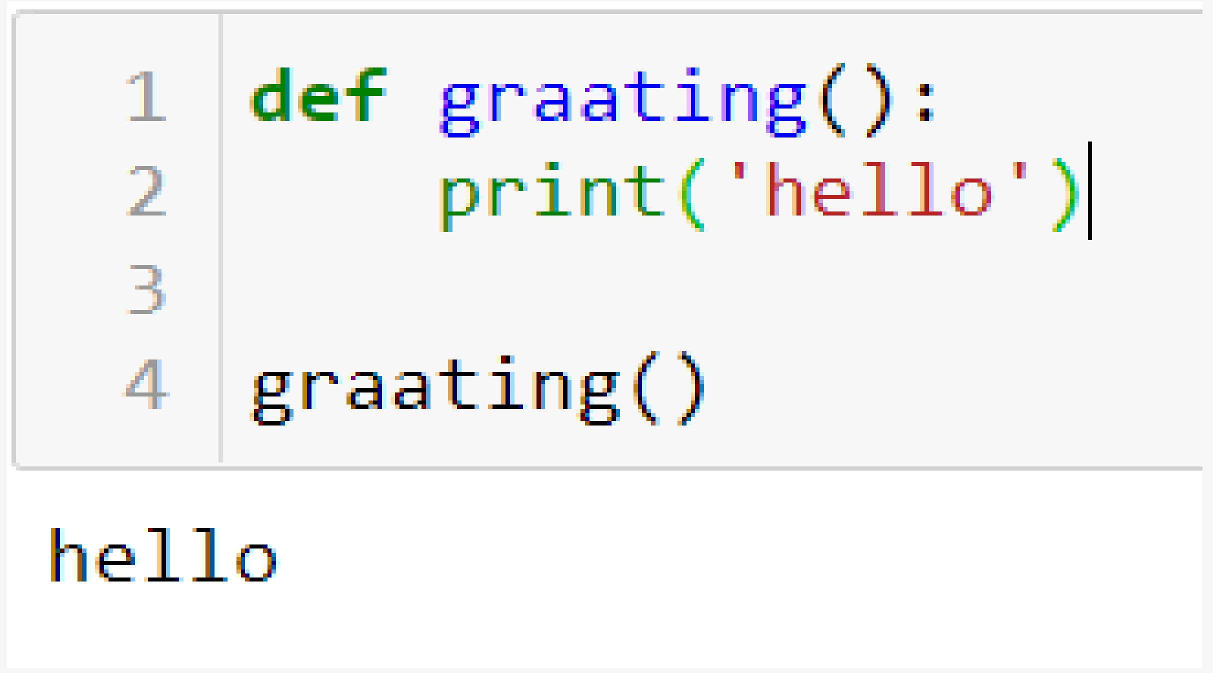
Creating Function

- In Python a function is defined using the def keyword followed by the function name and the parenthesized parameters.
- Example:



```python
def TAH_function():
    print("Welcome to Tahaluf Training!")

TAH_function()
```

```
Welcome to Tahaluf Training!
```



```python
def graating():
    print('hello')

graating()
```

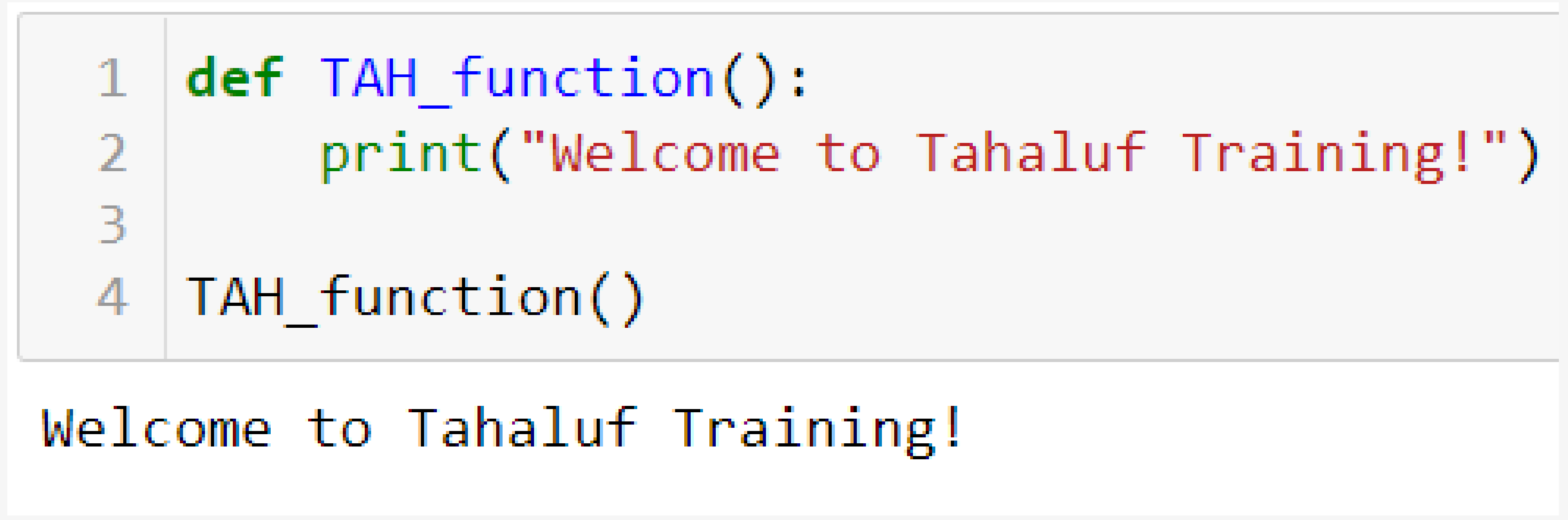
```
hello
```

The Learning Hub

Calling Function

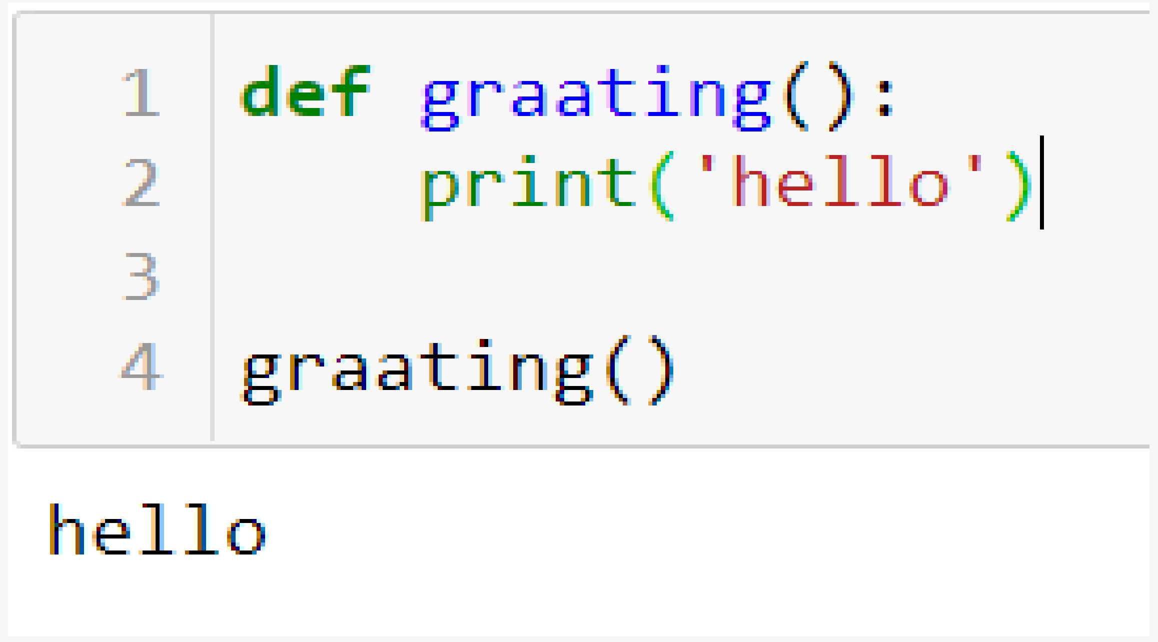
- Use the function name followed by parenthesis.

- **Example:**



```
def TAH_function():
    print("Welcome to Tahaluf Training!")

TAH_function()
```

```
Welcome to Tahaluf Training!
```



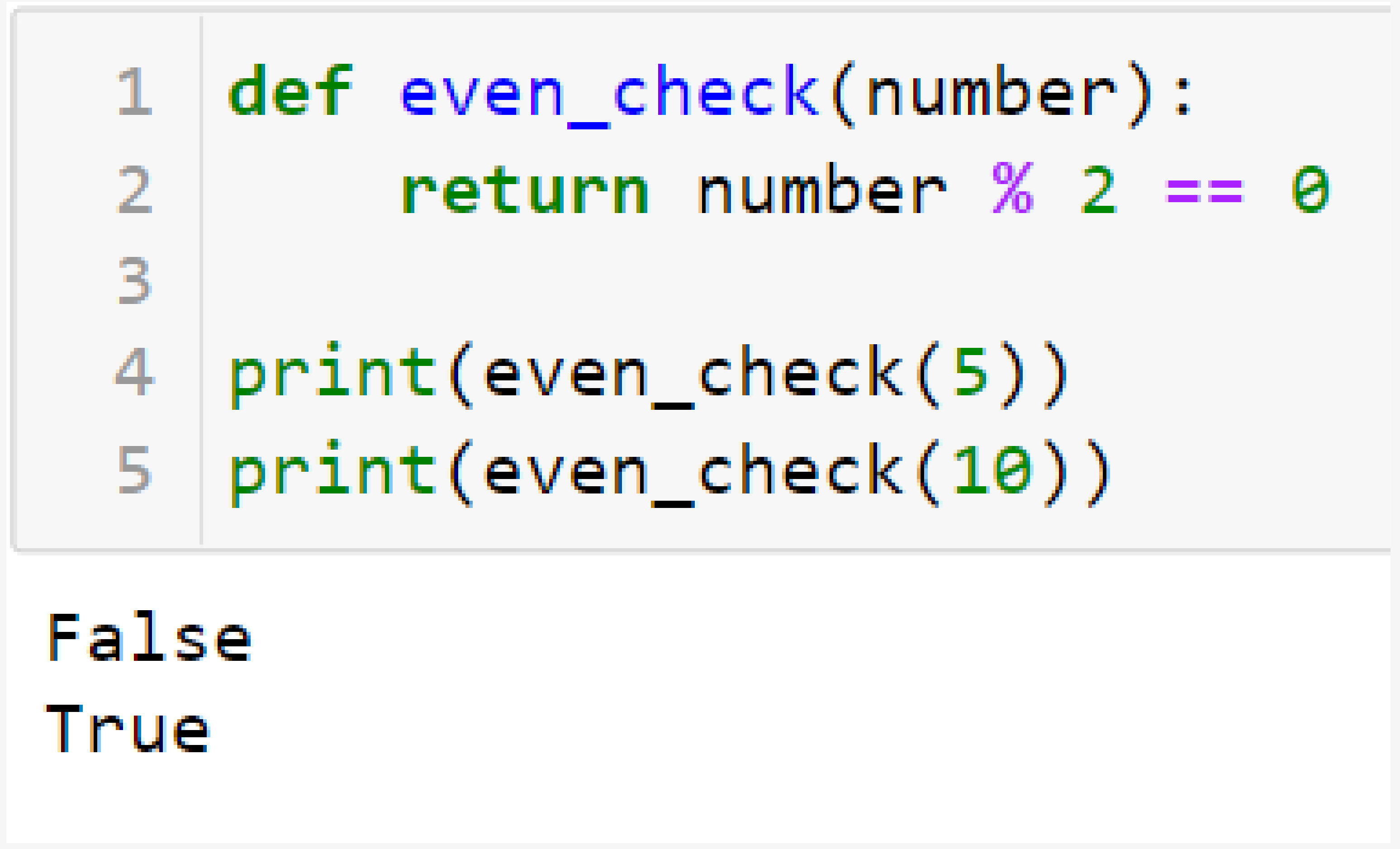
```
def graating():
    print('hello')

graating()
```

```
hello
```

Logic with Functions

- we can perform logical statements (such as if/else/if statements, for and while loops, checking if an item is in a list or not in a list)within a function.

- **Example:**



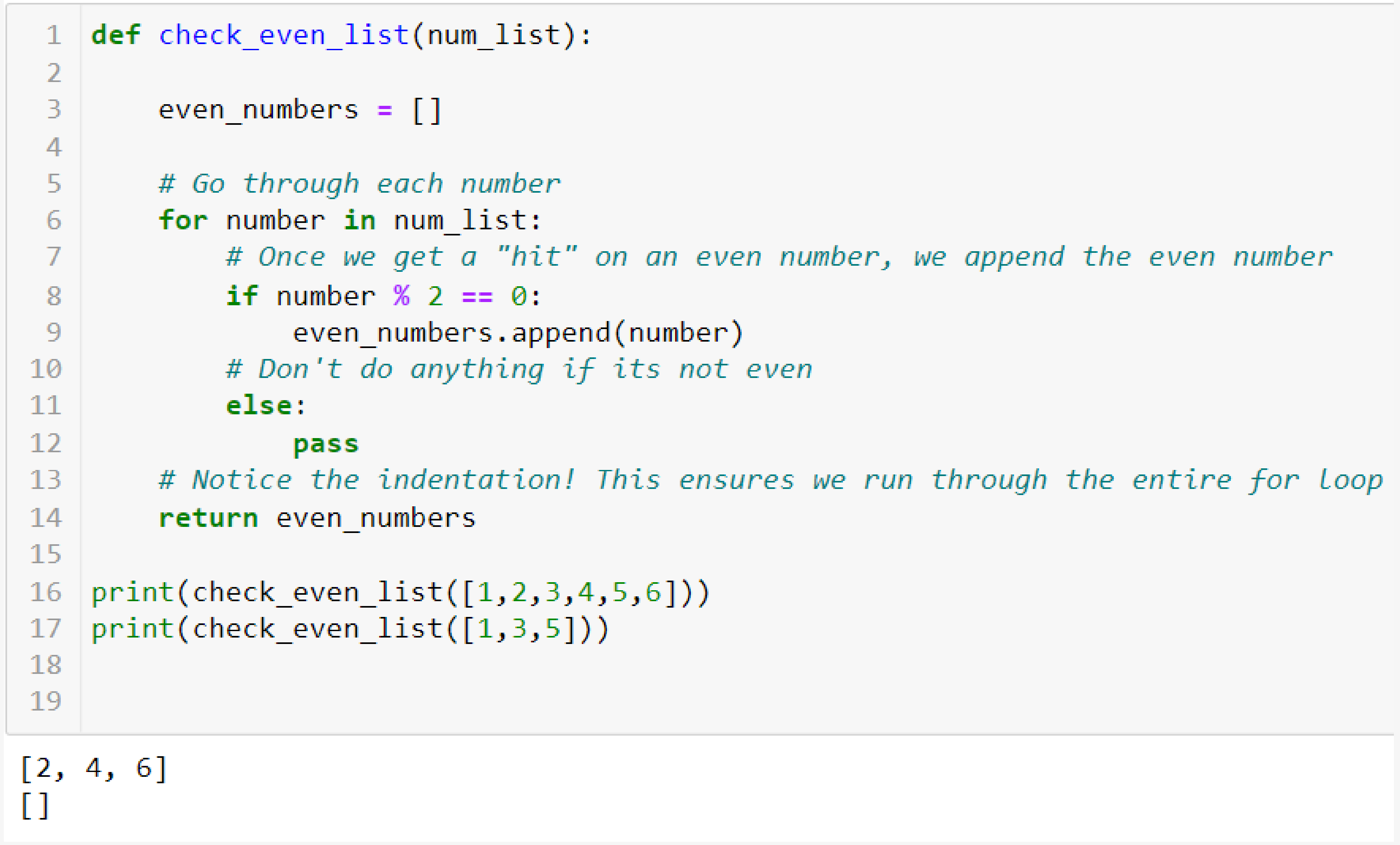
```python
def even_check(number):
    return number % 2 == 0

print(even_check(5))
print(even_check(10))
```

```
False
True
```

The Learning Hub

Logic with Functions

- **Example:** function returns all the even numbers in a list.

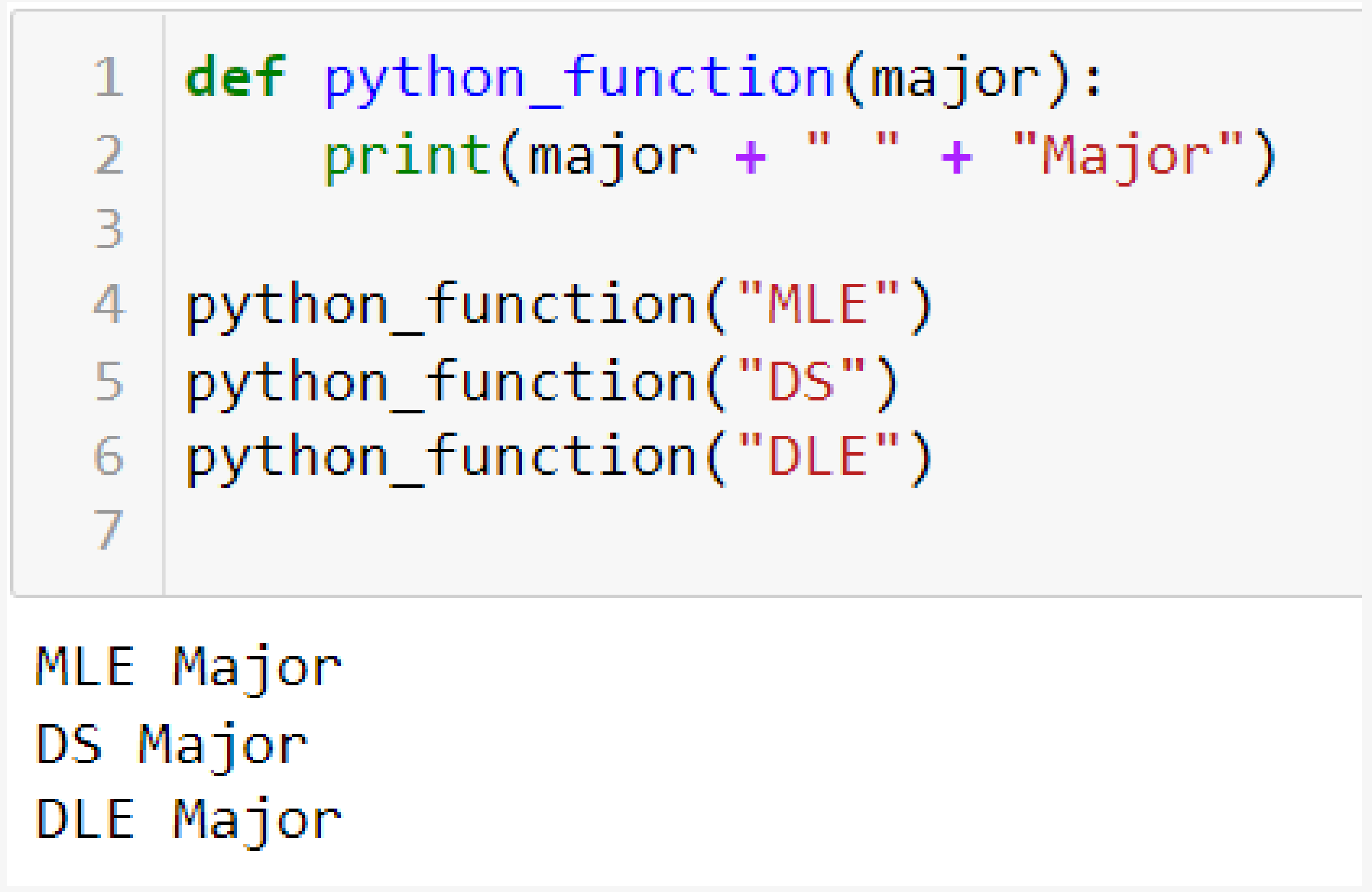


```python
def check_even_list(num_list):

    even_numbers = []

    # Go through each number
    for number in num_list:
        # Once we get a "hit" on an even number, we append the even number
        if number % 2 == 0:
            even_numbers.append(number)
        # Don't do anything if its not even
        else:
            pass
    # Notice the indentation! This ensures we run through the entire for loop
    return even_numbers

print(check_even_list([1,2,3,4,5,6]))
print(check_even_list([1,3,5]))


```

```
[2, 4, 6]
[]
```

Introduction to Arguments

- Information can be passed into functions as arguments.
- Arguments are specified after the function name, within the parentheses of the function definition.
- **Example:**



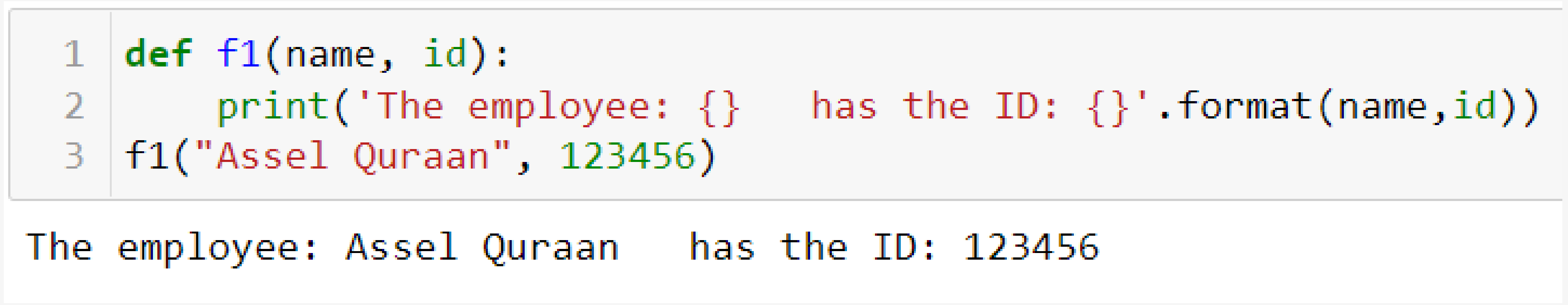
```
def python_function(major):
    print(major + " " + "Major")

python_function("MLE")
python_function("DS")
python_function("DLE")

```

```
MLE Major
DS Major
DLE Major
```

Number of Arguments

- Python functions can have multiple parameters.
- Function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

- **Example:**



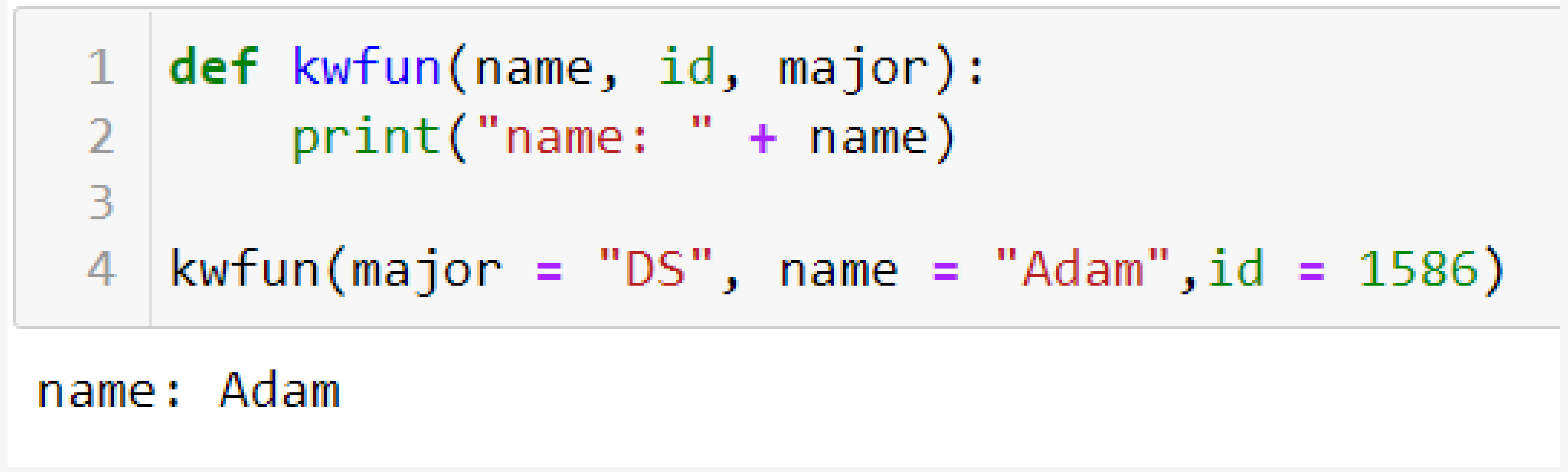
```python
def f1(name, id):
    print('The employee: {}   has the ID: {}'.format(name,id))
f1("Assel Quraan", 123456)
```

```
The employee: Assel Quraan   has the ID: 123456
```

The Learning Hub

Keyword Arguments

- You can send arguments with the key = value syntax.
- **Example:**



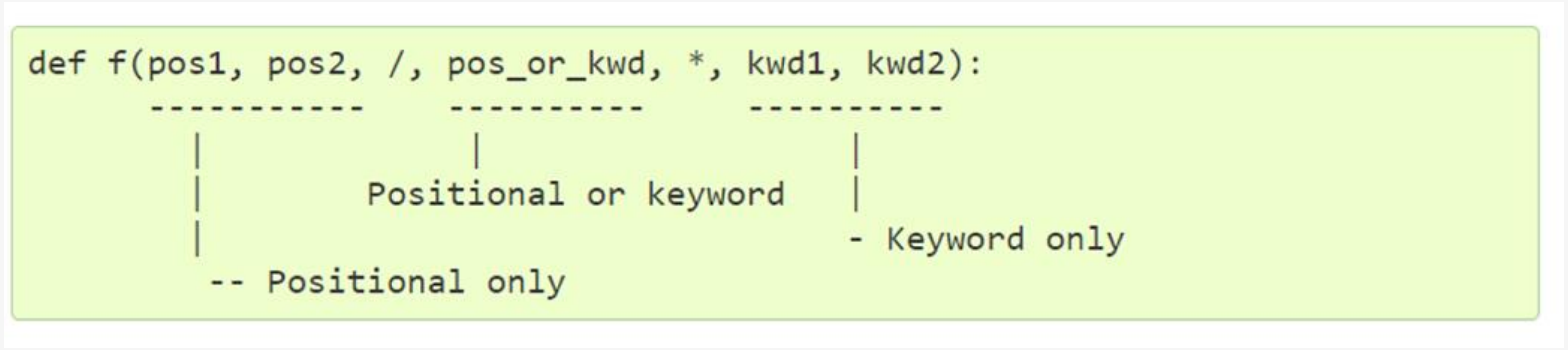
```python
def kwfun(name, id, major):
    print("name: " + name)

kwfun(major = "DS", name = "Adam",id = 1586)
```

```
name: Adam
```

The Learning Hub

Special Arguments

- *args and **kwargs can be passed into the same function, but *args have to appear before **kwargs

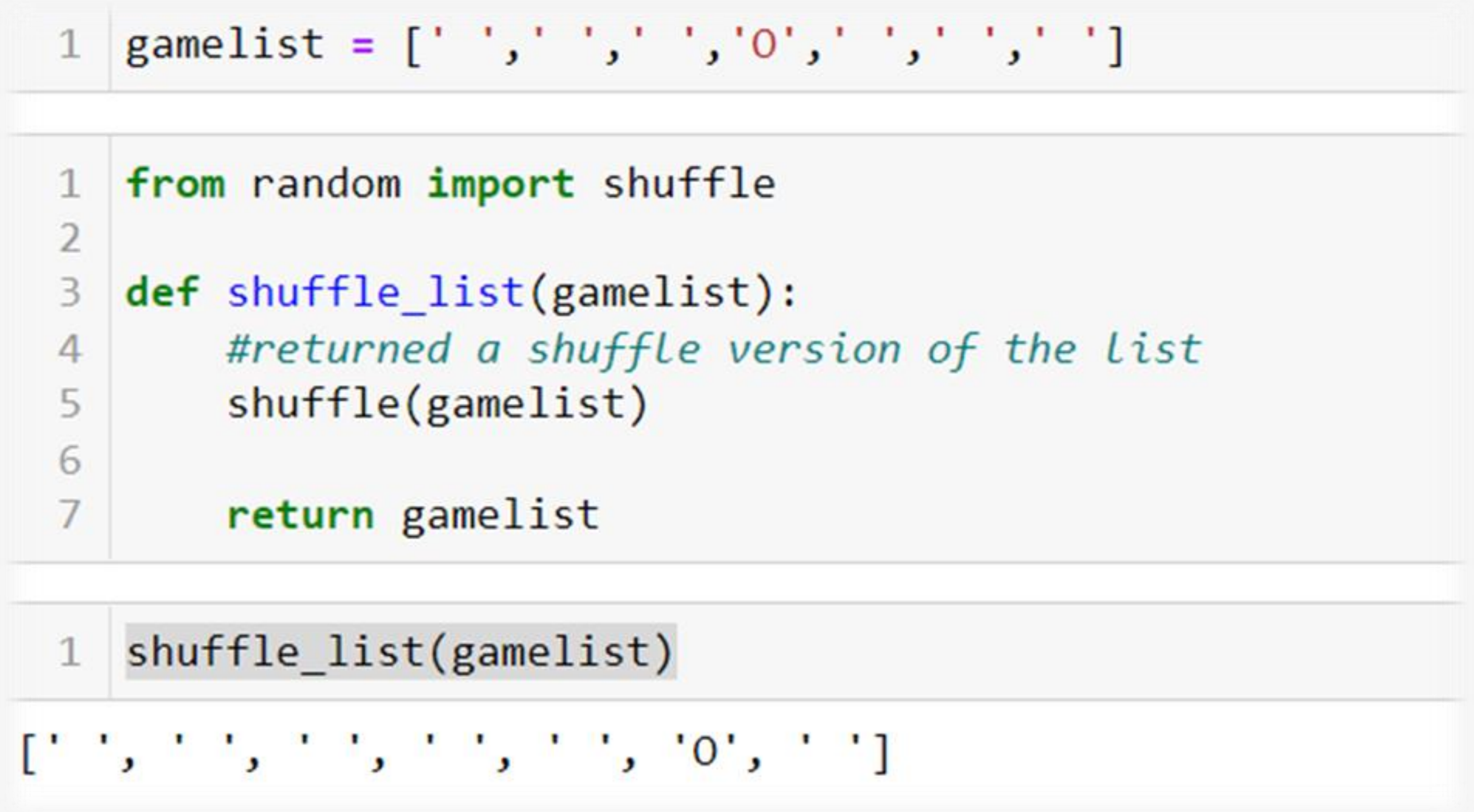


```
def f(pos1, pos2, /, pos_or_kwd, *, kwd1, kwd2):
      -----------    ----------     ----------
        |             |                  |
        |        Positional or keyword   |
        |                                - Keyword only
         -- Positional only
```

The Learning Hub

Interactions between Python Functions

- When using results from functions as input of another function.
- The best example of Interactions between Python Functions is a guessing game
- There will be 7 positions in the list, one of which is an 'O', a function will shuffle the list



```
gamelist = [' ',' ',' ','O',' ',' ',' ']
```

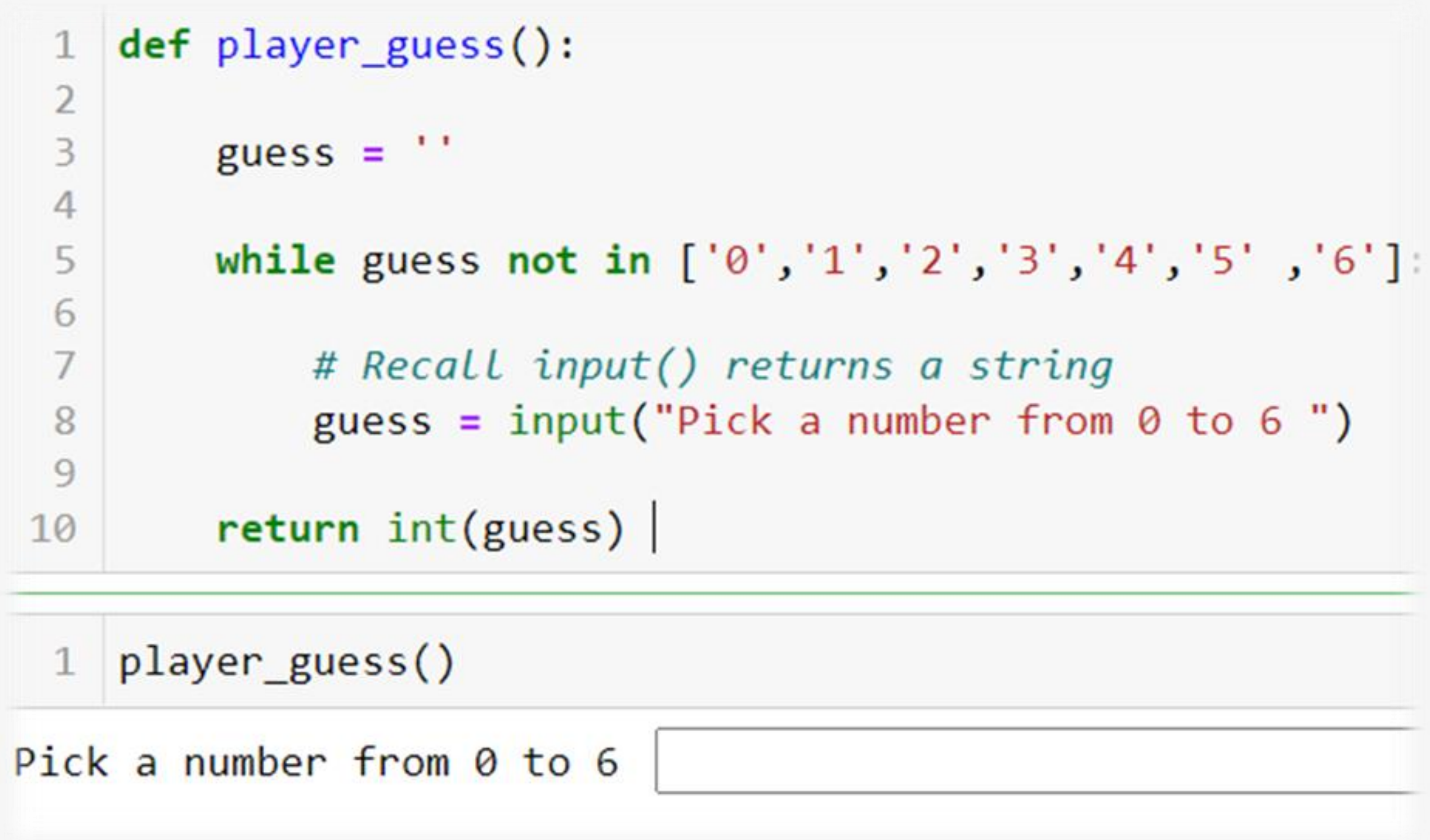
```
from random import shuffle

def shuffle_list(gamelist):
    #returned a shuffle version of the list
    shuffle(gamelist)

    return gamelist
```

```
shuffle_list(gamelist)
```

```
[' ', ' ', ' ', ' ', ' ', 'O', ' ']
```

Interactions between Python Functions

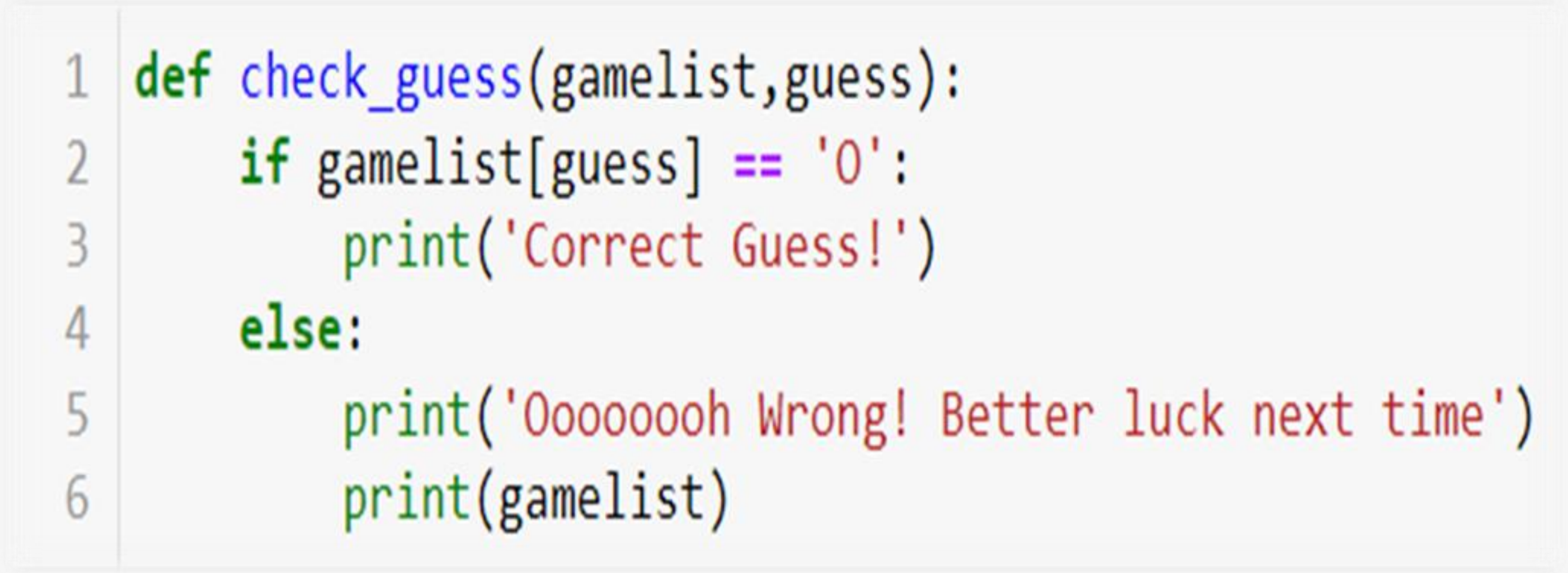
- Another function will take a player's guess
- And the third one will check if the guess is correct according on the output of the two previous functions.



```python
def player_guess():

    guess = ''

    while guess not in ['0','1','2','3','4','5' ,'6']:

        # Recall input() returns a string
        guess = input("Pick a number from 0 to 6 ")

    return int(guess)
```

```python
player_guess()
```

```
Pick a number from 0 to 6
```

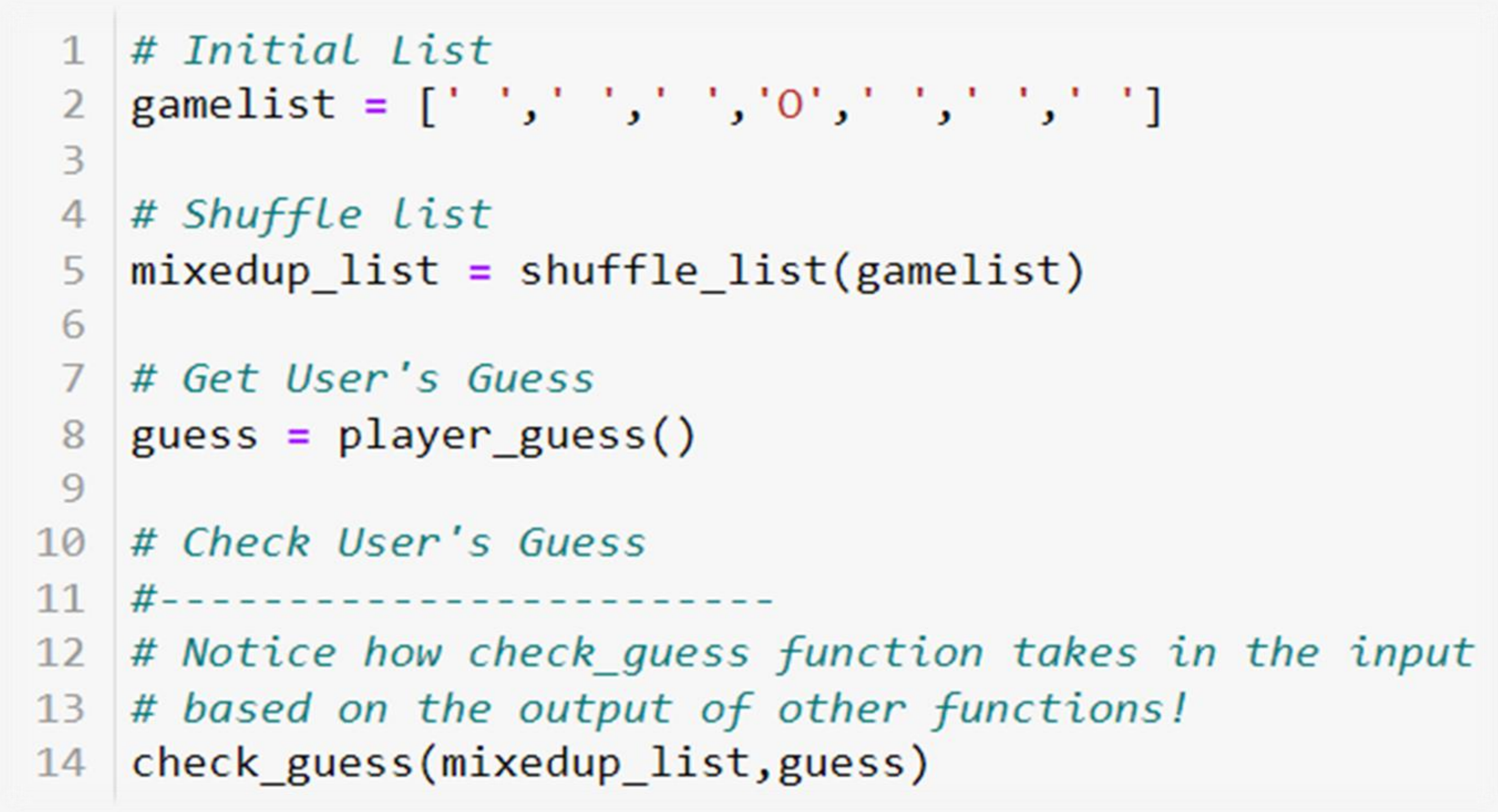


```python
def check_guess(gamelist,guess):
    if gamelist[guess] == 'O':
        print('Correct Guess!')
    else:
        print('Oooooooh Wrong! Better luck next time')
        print(gamelist)
```

The Learning Hub

Interactions between Python Functions

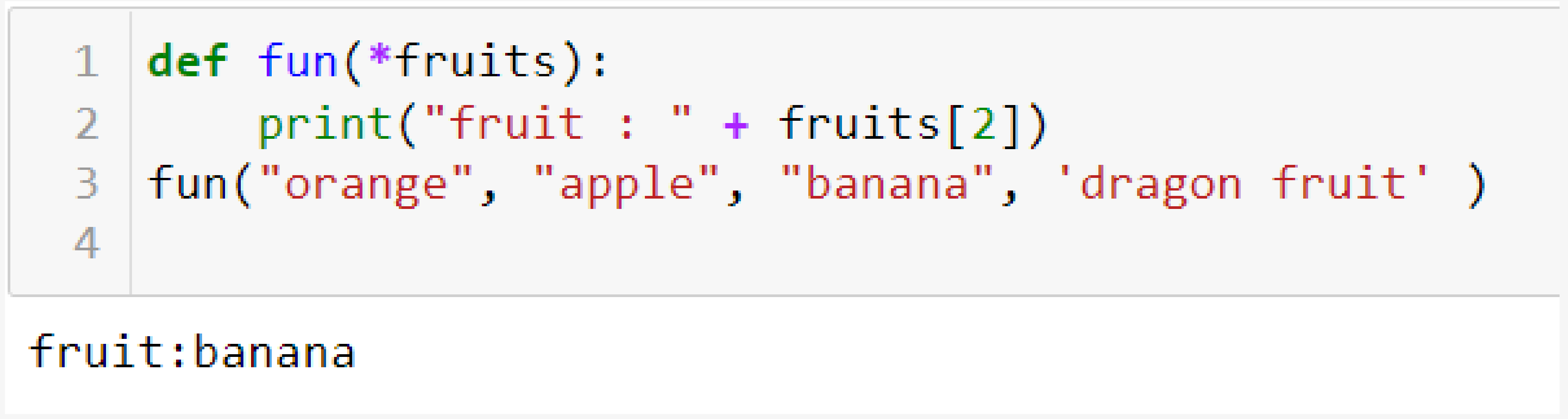
- Now we create a little setup logic to run all the functions. Notice how they interact with each other



```python
# Initial List
gamelist = [' ',' ',' ','O',' ',' ',' ']

# Shuffle list
mixedup_list = shuffle_list(gamelist)

# Get User's Guess
guess = player_guess()

# Check User's Guess
#------------------------
# Notice how check_guess function takes in the input
# based on the output of other functions!
check_guess(mixedup_list,guess)
```

Arbitrary Arguments (*args)

- If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.
- **Example:**



```
def fun(*fruits):
    print("fruit : " + fruits[2])
fun("orange", "apple", "banana", 'dragon fruit' )

```

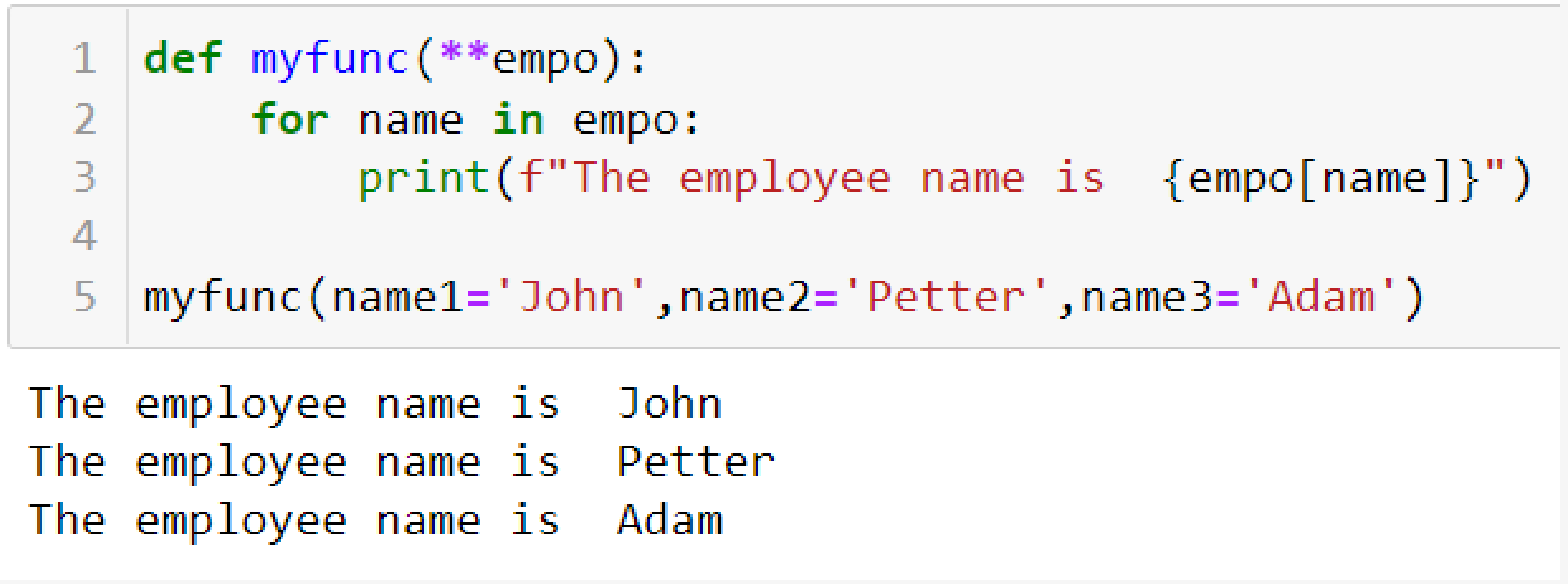
```
fruit:banana
```

The Learning Hub

Arbitrary Keyword Arguments (**kwargs)

- If you do not know how many keyword arguments will be passed into your function, add two asterisks: ** before the parameter name in the function definition.

- **Example:**

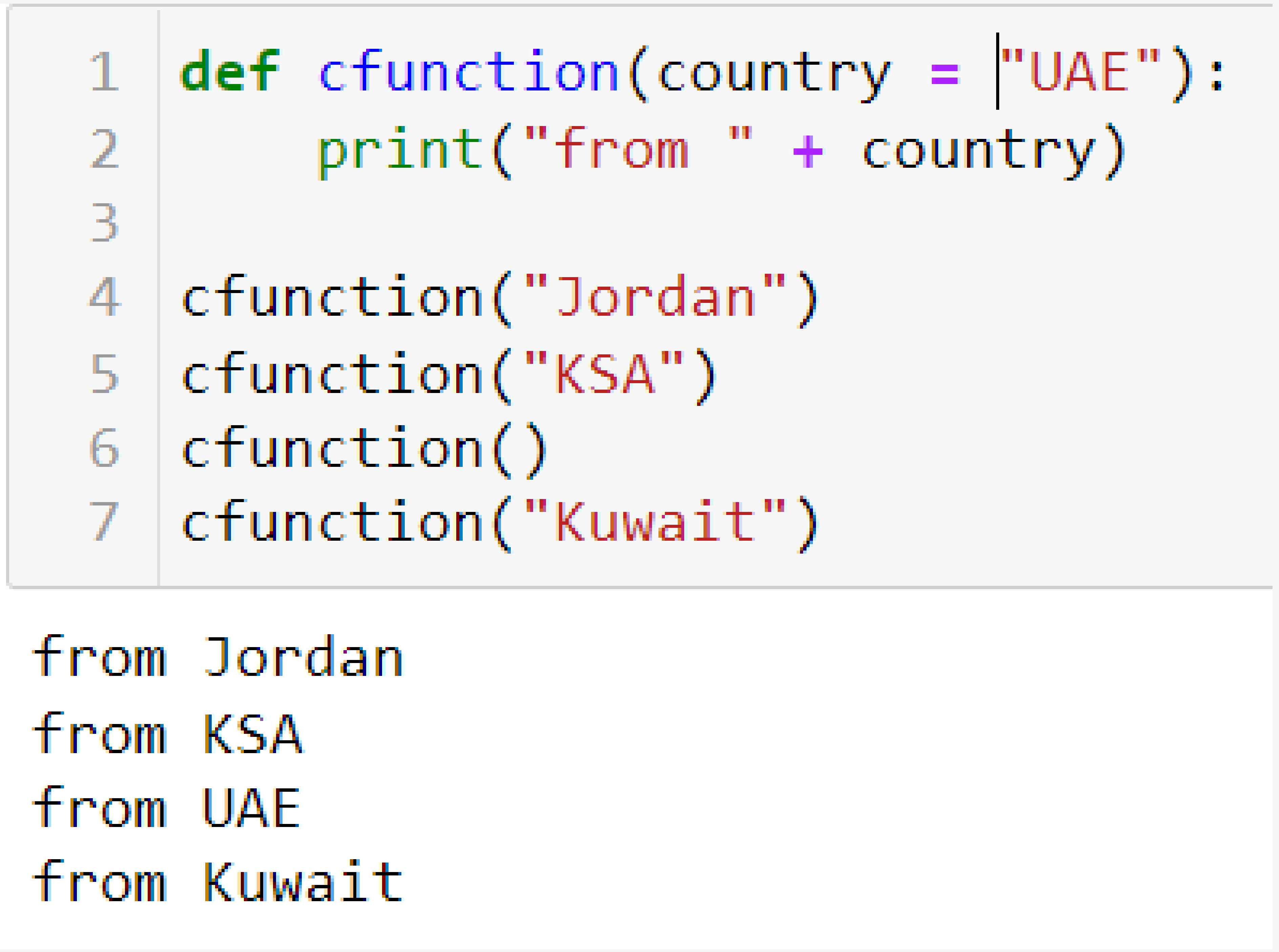


```python
def myfunc(**empo):
    for name in empo:
        print(f"The employee name is  {empo[name]}")

myfunc(name1='John',name2='Petter',name3='Adam')
```

```
The employee name is  John
The employee name is  Petter
The employee name is  Adam
```

The Learning Hub

Default Parameter Value

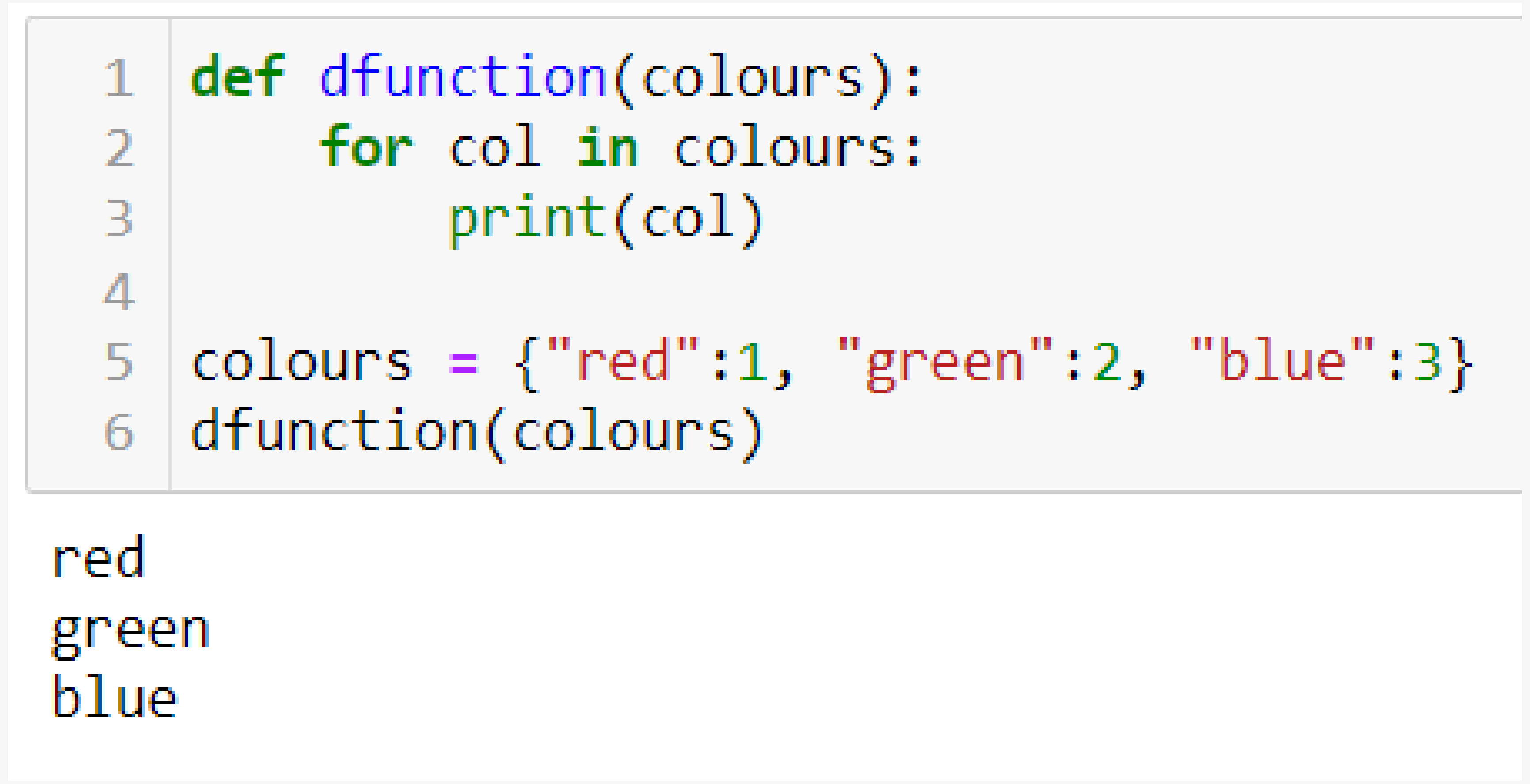


```python
def cfunction(country = "UAE"):
    print("from " + country)

cfunction("Jordan")
cfunction("KSA")
cfunction()
cfunction("Kuwait")
```

```
from Jordan
from KSA
from UAE
from Kuwait
```

Passing a dictionary as an Argument

- Send any data type of argument to a function (string, number, list, dictionary), and it will be treated as the same data type inside the function.
- **Example:**



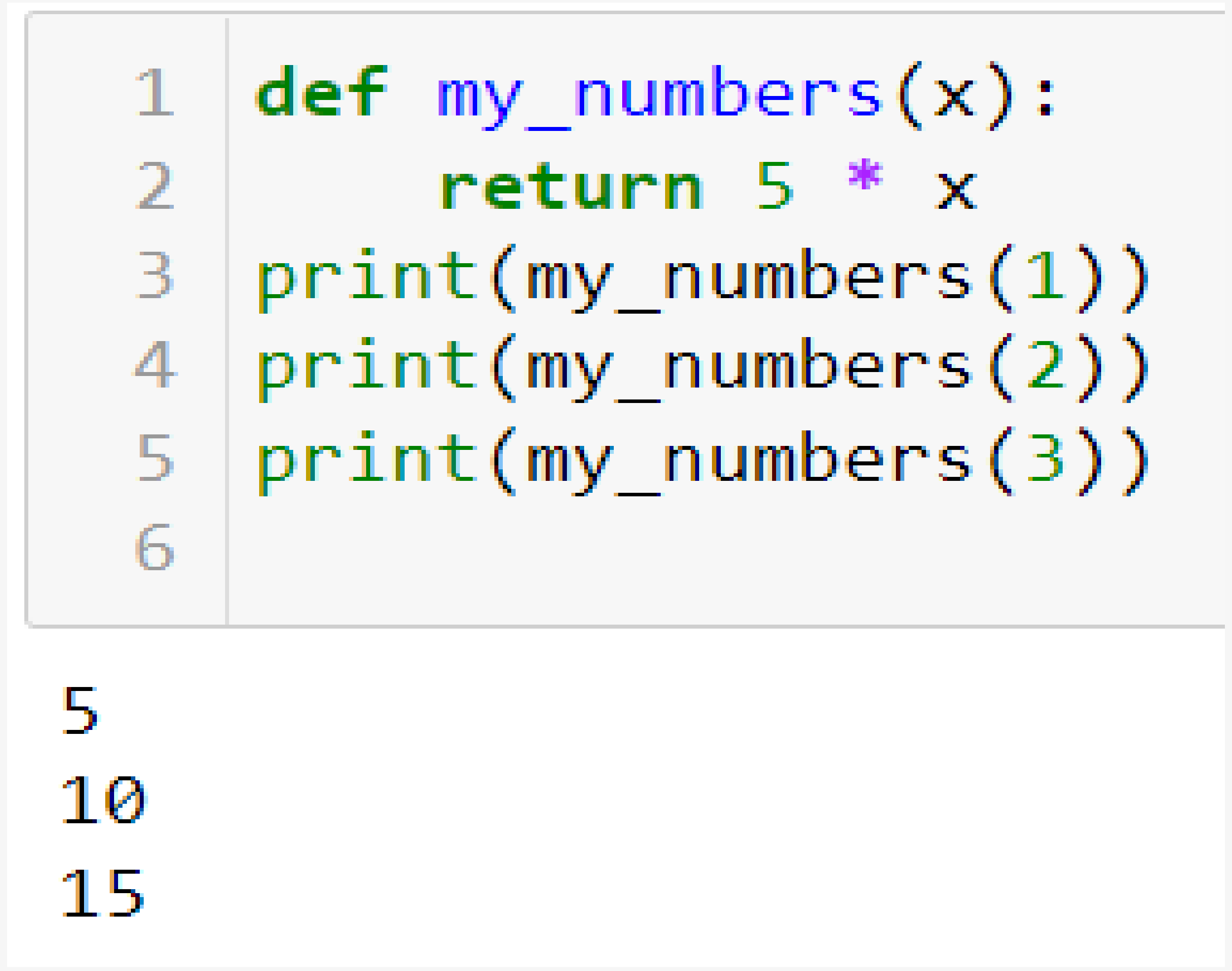
```python
def dfunction(colours):
    for col in colours:
        print(col)

colours = {"red":1, "green":2, "blue":3}
dfunction(colours)
```

```
red
green
blue
```

Return Values

- Return () statement can be used to let a function return a value that can then be stored as a variable, or used in whatever manner a user wants.

- Example:



```python
def my_numbers(x):
    return 5 * x
print(my_numbers(1))
print(my_numbers(2))
print(my_numbers(3))

```

```
5
10
15
```

The Learning Hub

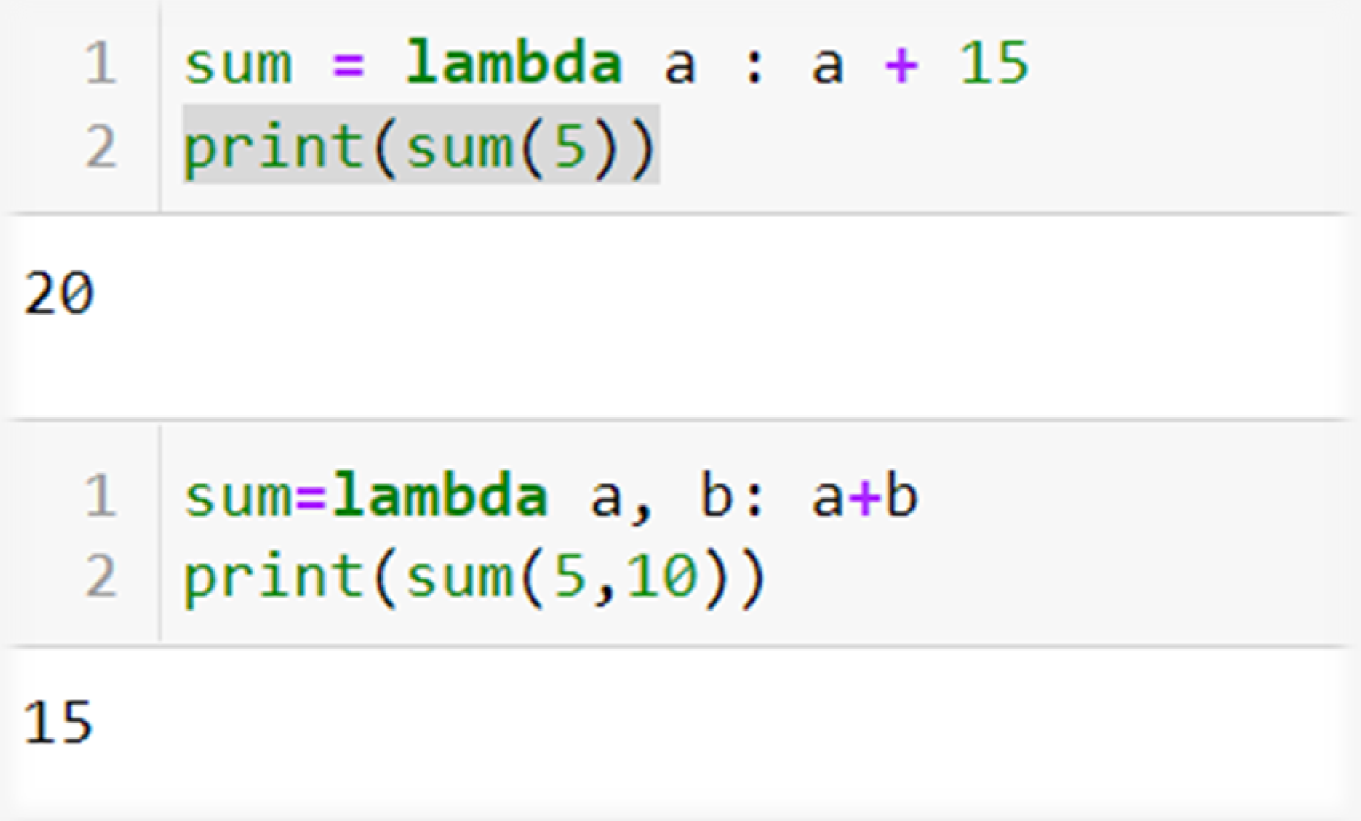
Lambda Expression

- A lambda function is a small anonymous function that can be created with the lambda keyword.
- A lambda function can take any number of arguments, but can only have one expression
- syntactically restricted to a single expression
- Syntax

➢ **lambda arguments : expression**

Lambda Expression

- Lambda functions can take any number of arguments:
- **Example:**

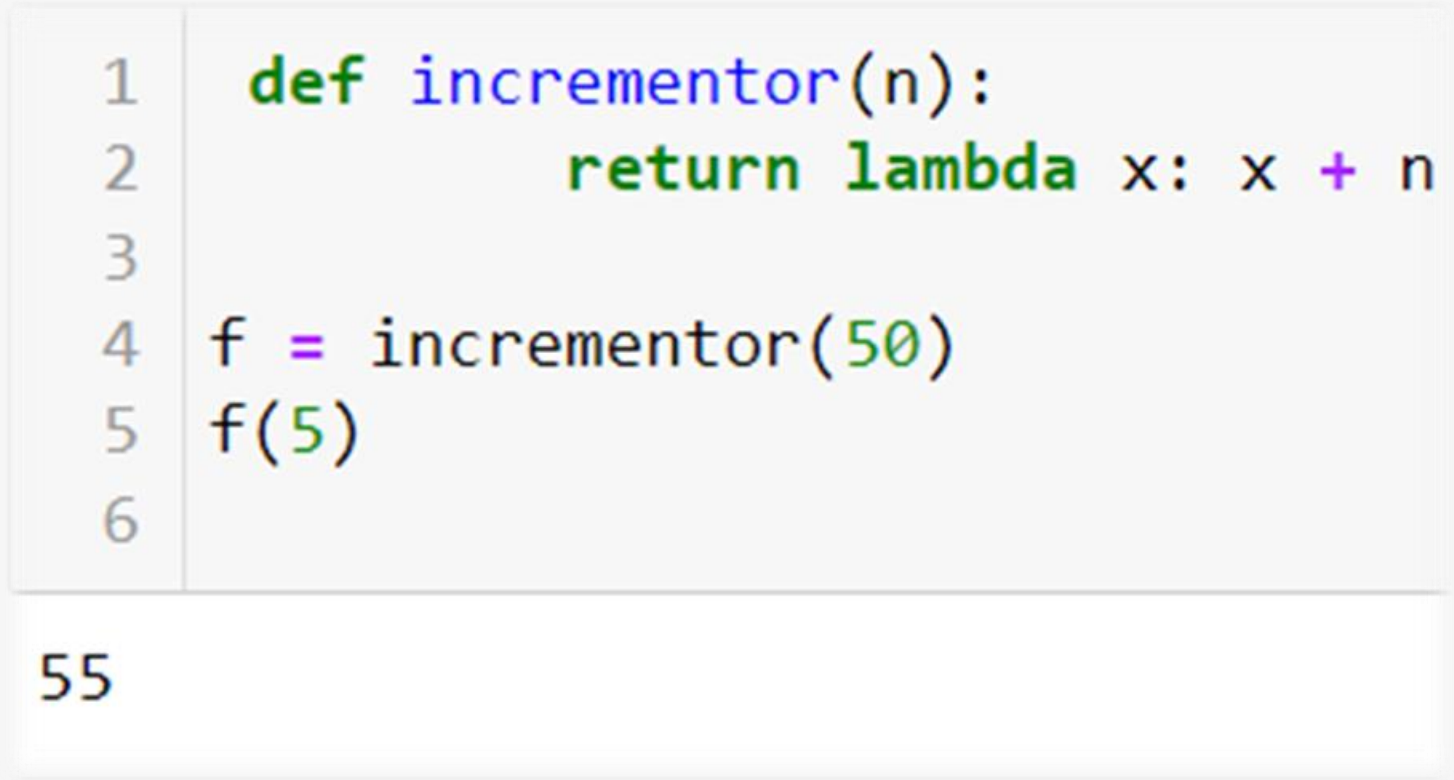


```
sum = lambda a : a + 15
print(sum(5))
```

```
20
```

```
sum=lambda a, b: a+b
print(sum(5,10))
```

```
15
```



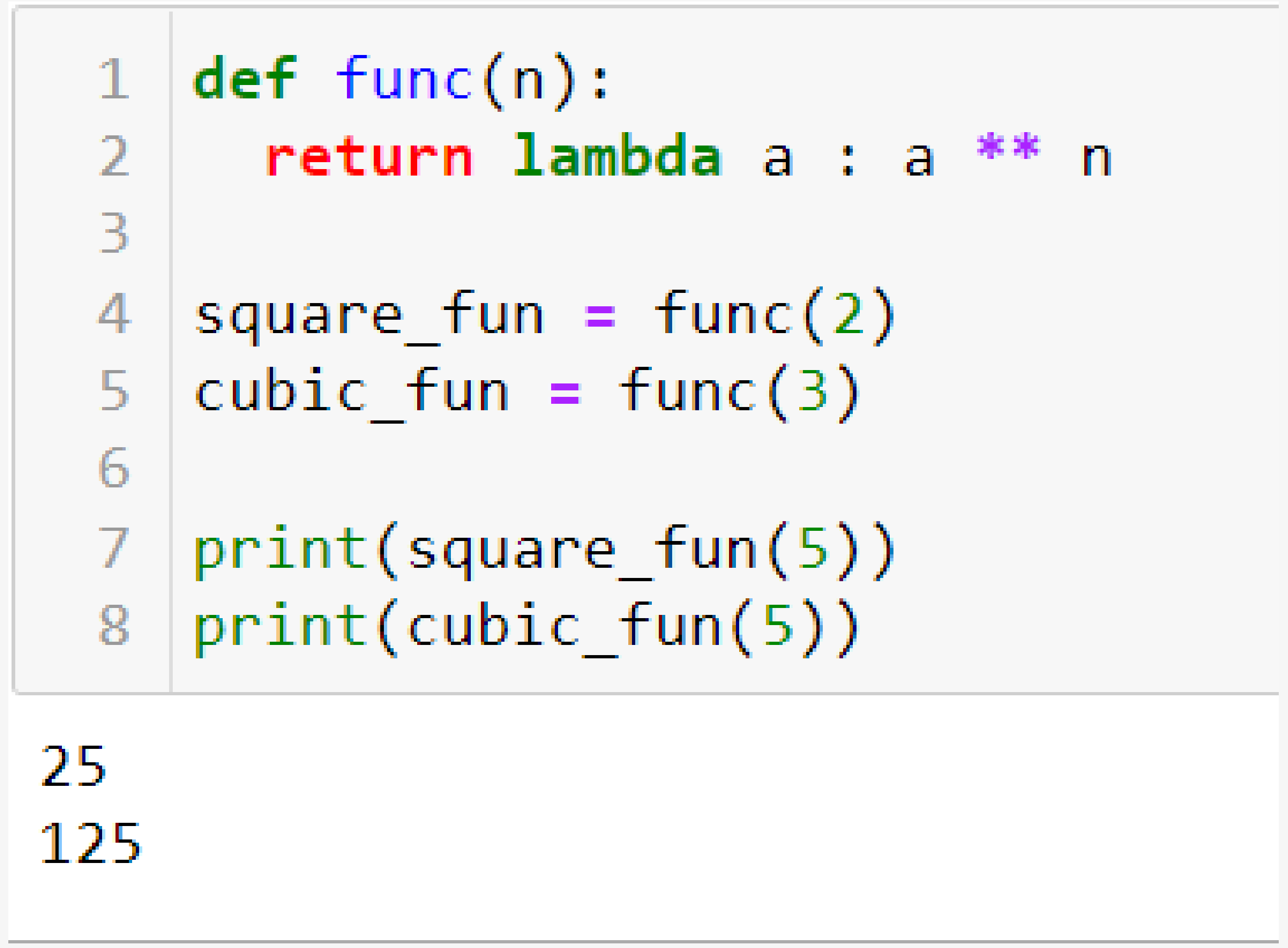
```
 def incrementor(n):
        return lambda x: x + n

f = incrementor(50)
f(5)

```

```
55
```

The Learning Hub

Lambda Expression

- Lambda functions can take any number of arguments:
- **Example:**



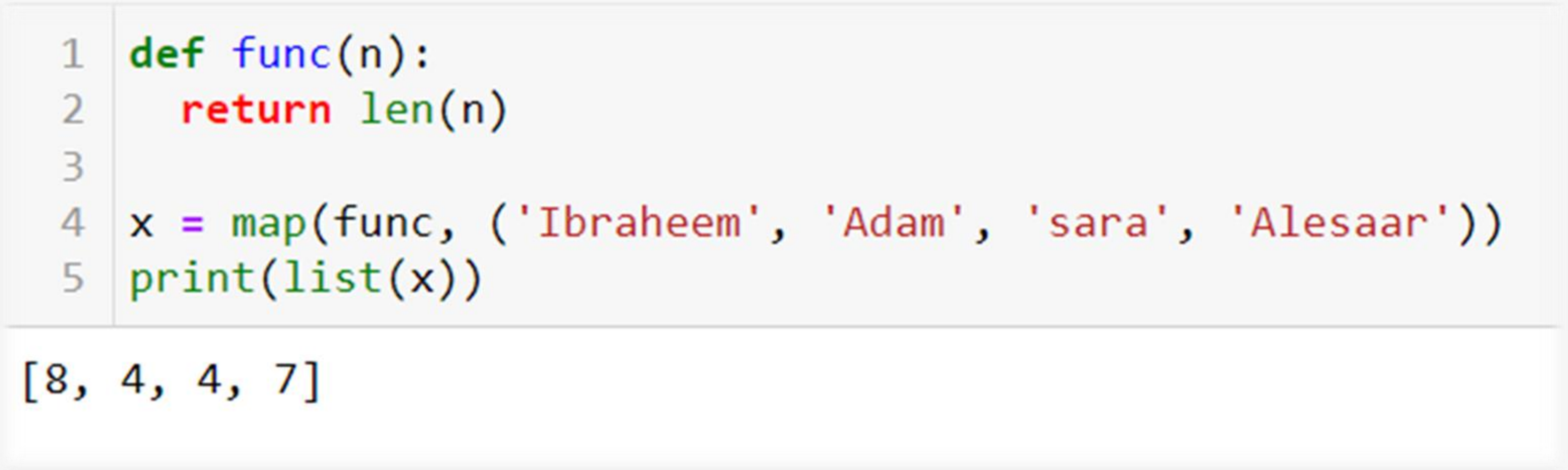
```
def func(n):
  return lambda a : a ** n

square_fun = func(2)
cubic_fun = func(3)

print(square_fun(5))
print(cubic_fun(5))
```

```
25
125
```

The Learning Hub

Map and Filter Functions

- **map()** function: built-in function returns the specified iterator with the results after applying the function to each item of a given iterable.
- **Example:**



```python
def func(n):
  return len(n)

x = map(func, ('Ibraheem', 'Adam', 'sara', 'Alesaar'))
print(list(x))
```

```
[8, 4, 4, 7]
```

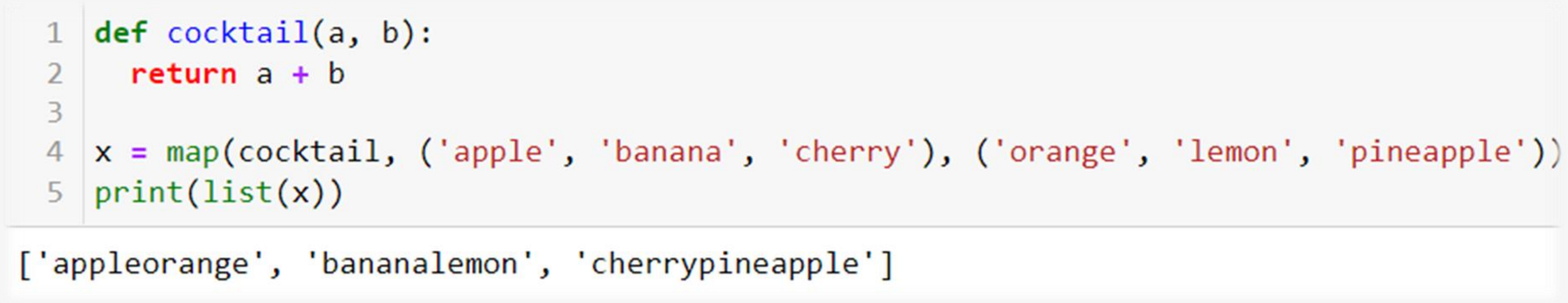
The Learning Hub

Map and Filter Functions

- **Syntax: map(function, iterable)**

items sent to the function as parameters

- **Example:**



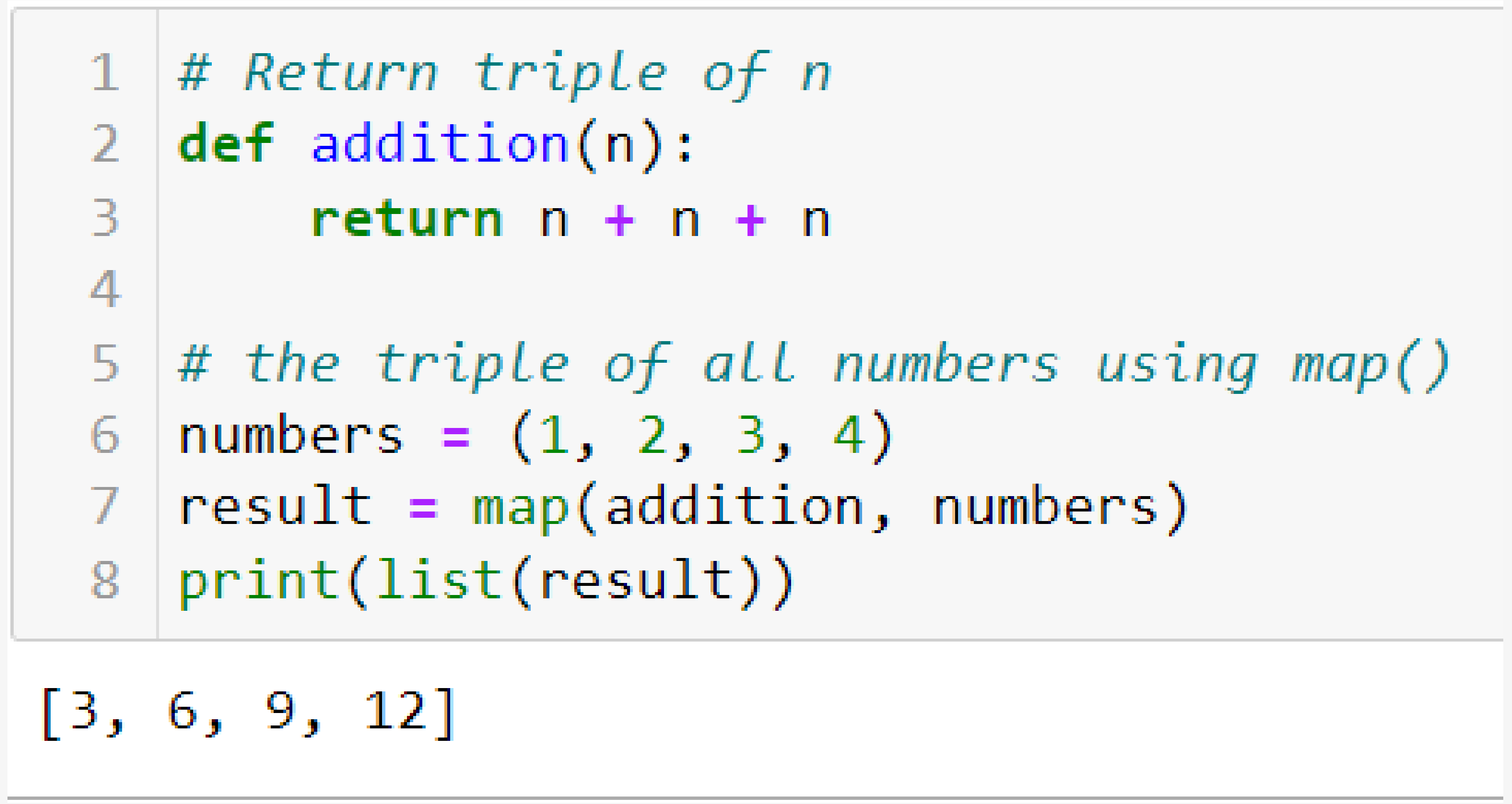
```
def cocktail(a, b):
  return a + b

x = map(cocktail, ('apple', 'banana', 'cherry'), ('orange', 'lemon', 'pineapple'))
print(list(x))
```

```
['appleorange', 'bananalemon', 'cherrypineapple']
```

The Learning Hub

Map and Filter Functions

- You can pass one or more iterable to the **map()** function
- **Example:**



```python
# Return triple of n
def addition(n):
    return n + n + n

# the triple of all numbers using map()
numbers = (1, 2, 3, 4)
result = map(addition, numbers)
print(list(result))
```

```
[3, 6, 9, 12]
```

Map and Filter Functions

- The filter(): Use a filter function to exclude each element in the iterable object to be true or not.

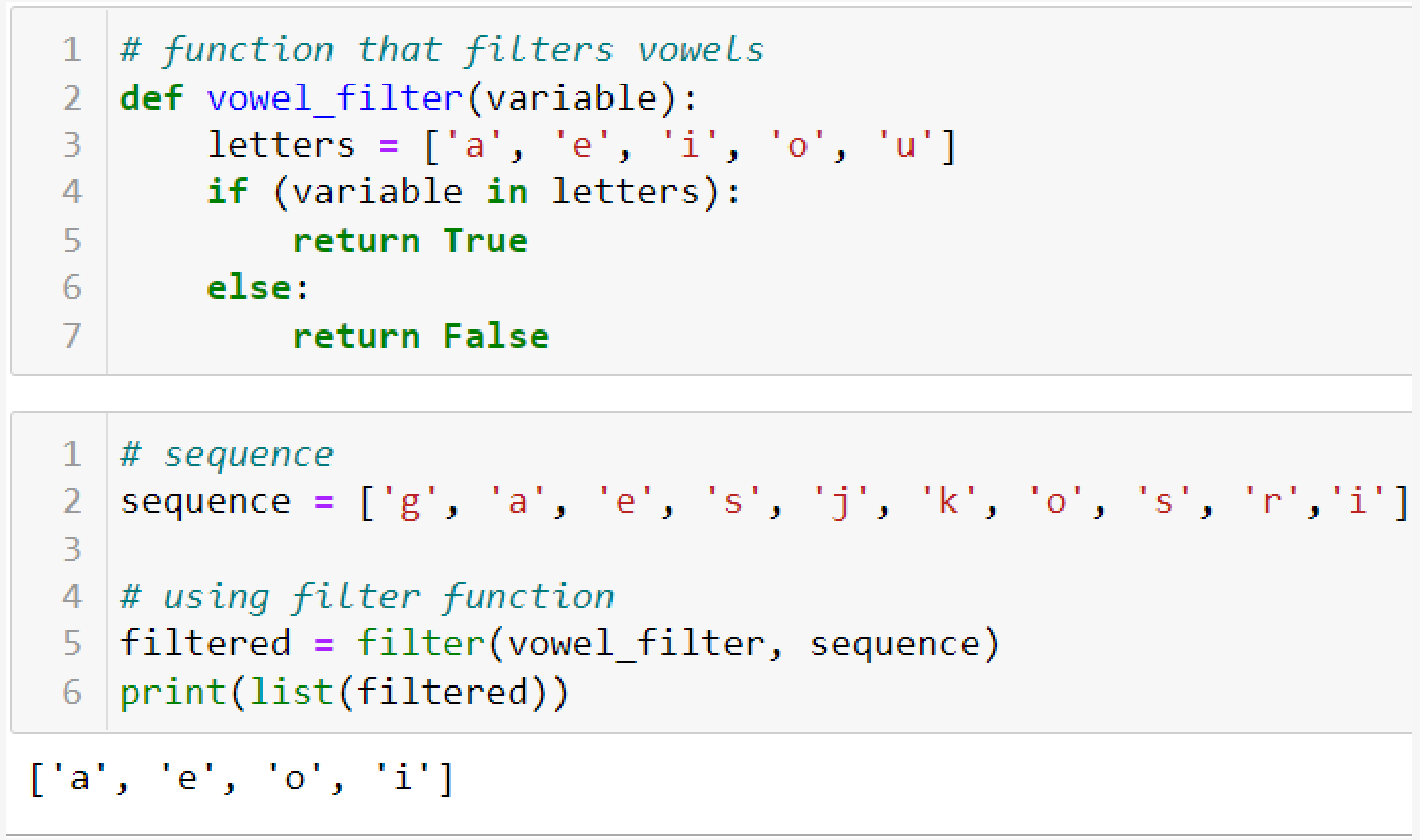
- Syntax: filter(function, iterable)

function: a function that tests if each element of a iterable is true or not.

iterable: iterable to be filtered

Map and Filter Functions

- Example: filter ()



```python
# function that filters vowels
def vowel_filter(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False
```

```python
# sequence
sequence = ['g', 'a', 'e', 's', 'j', 'k', 'o', 's', 'r','i']

# using filter function
filtered = filter(vowel_filter, sequence)
print(list(filtered))
```

```
['a', 'e', 'o', 'i']
```

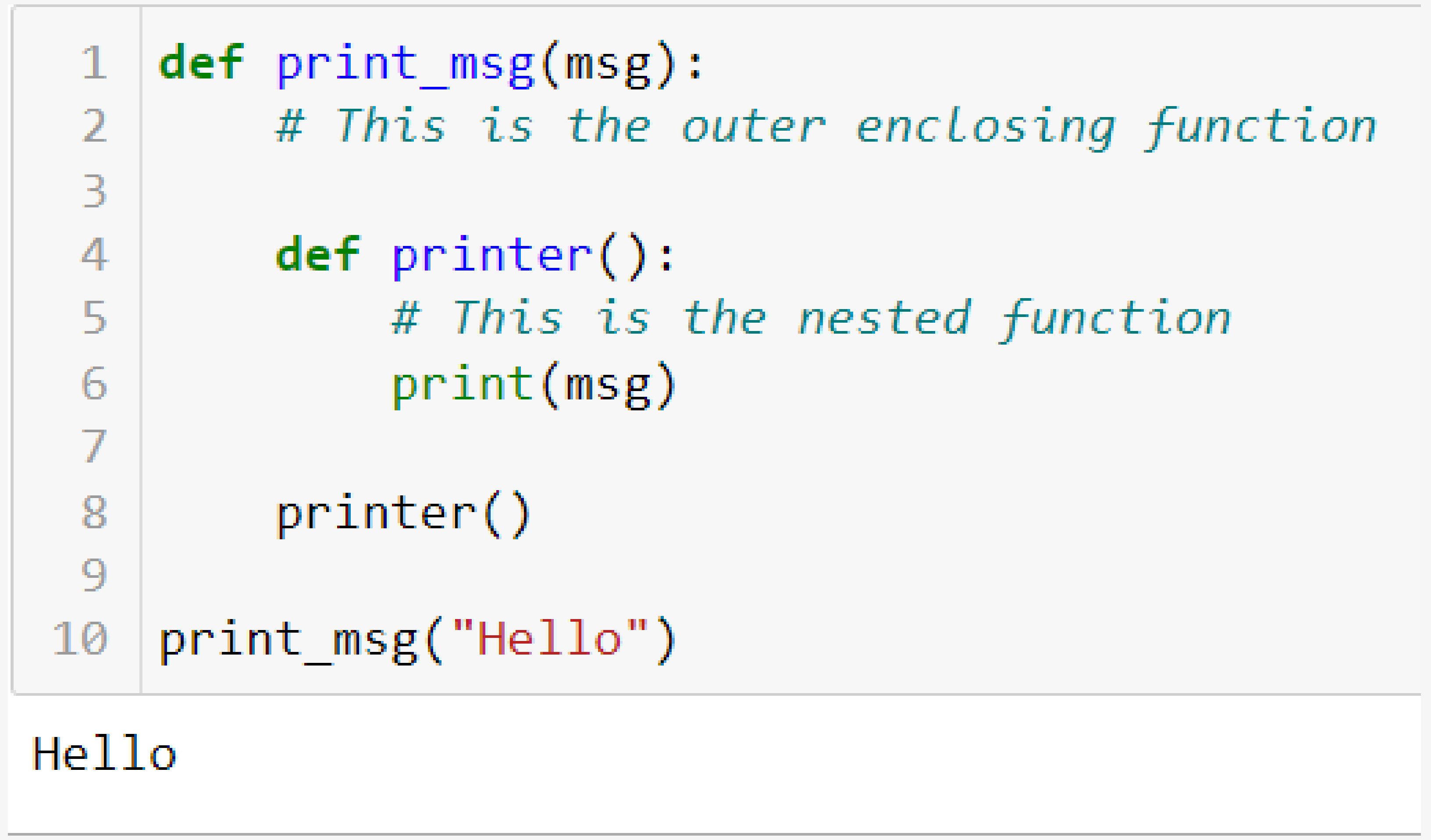
The Learning Hub

Nested Statements and Scope

- A function can be termed a Nested function when is defined inside another function.

  Nested functions can access variables of the enclosing scope.

- **Example :**



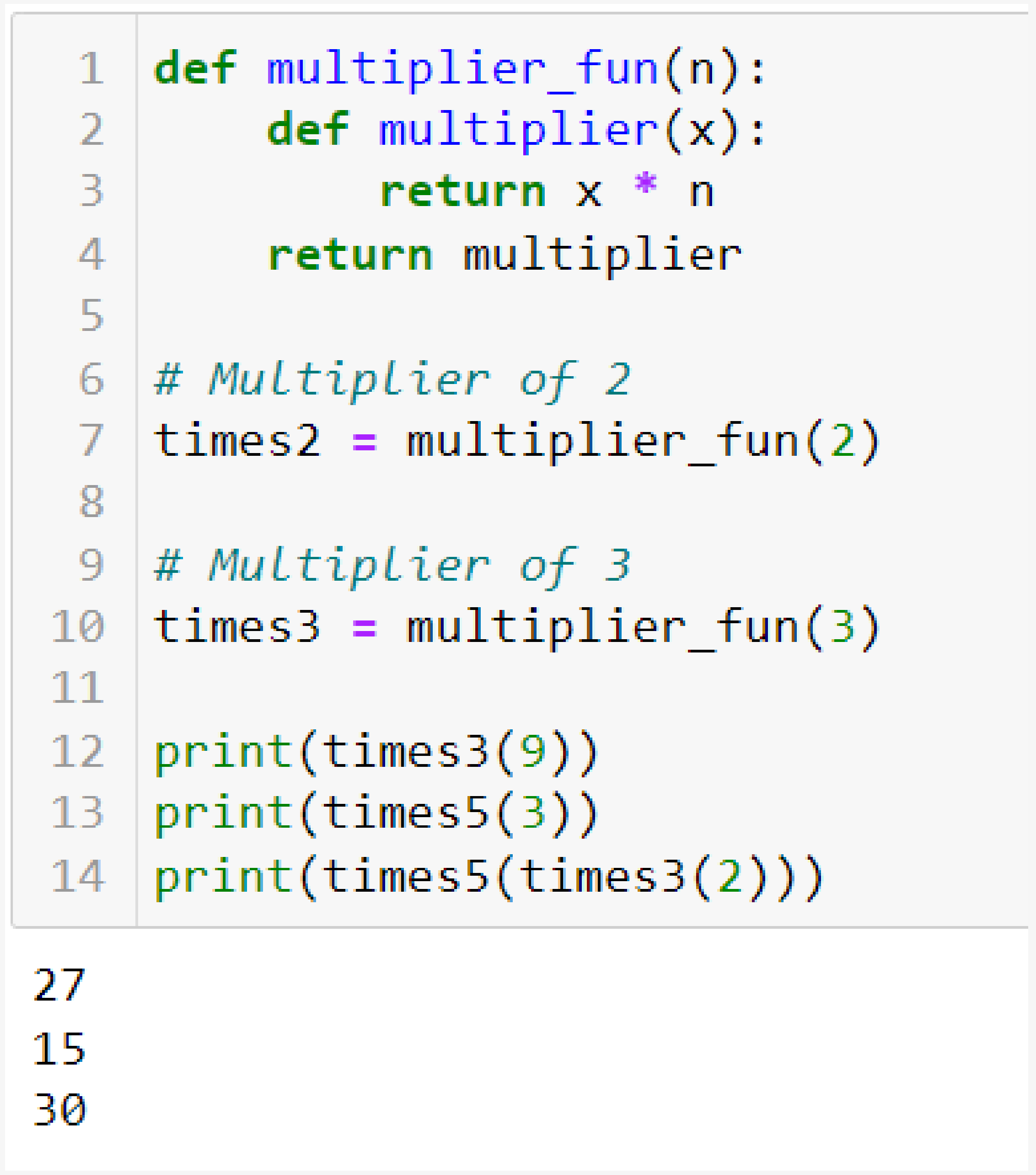
```
def print_msg(msg):
    # This is the outer enclosing function

    def printer():
        # This is the nested function
        print(msg)

    printer()

print_msg("Hello")
```

```
Hello
```

The Learning Hub

Nested Statements and Scope

- **Example:**



```python
def multiplier_fun(n):
    def multiplier(x):
        return x * n
    return multiplier

# Multiplier of 2
times2 = multiplier_fun(2)

# Multiplier of 3
times3 = multiplier_fun(3)

print(times3(9))
print(times5(3))
print(times5(times3(2)))
```

```
27
15
30
```

The Learning Hub

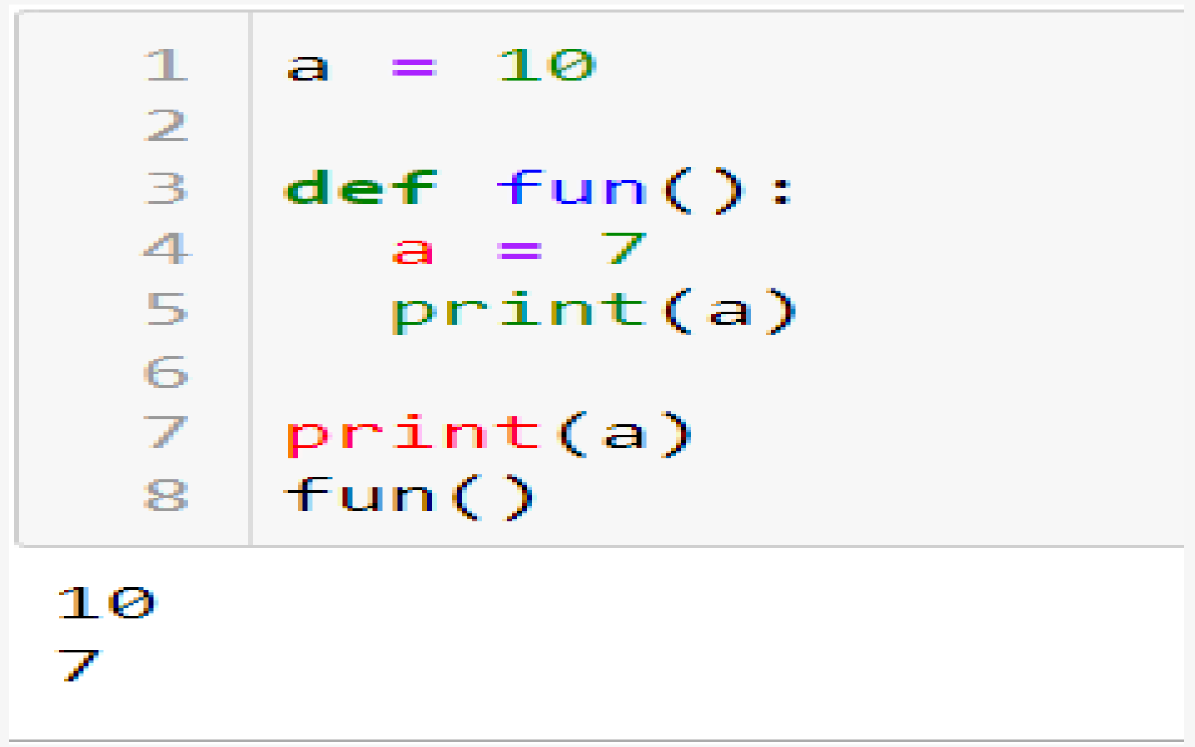
The Scope of Variables

- If you create a variable with the same name inside a function. This variable will be local, and can only be used inside the function.
- non-local variables are read-only by default and they must be declared explicitly as non-local (using nonlocal keyword) in order to modify them.

The Learning Hub

Local Variables

- In Python a local variable is a variable defined inside a function.
- local variables cannot be used outside of the scope of the function, And trying to do this without defining the variable outside the function will cause an error.
- **Example:**



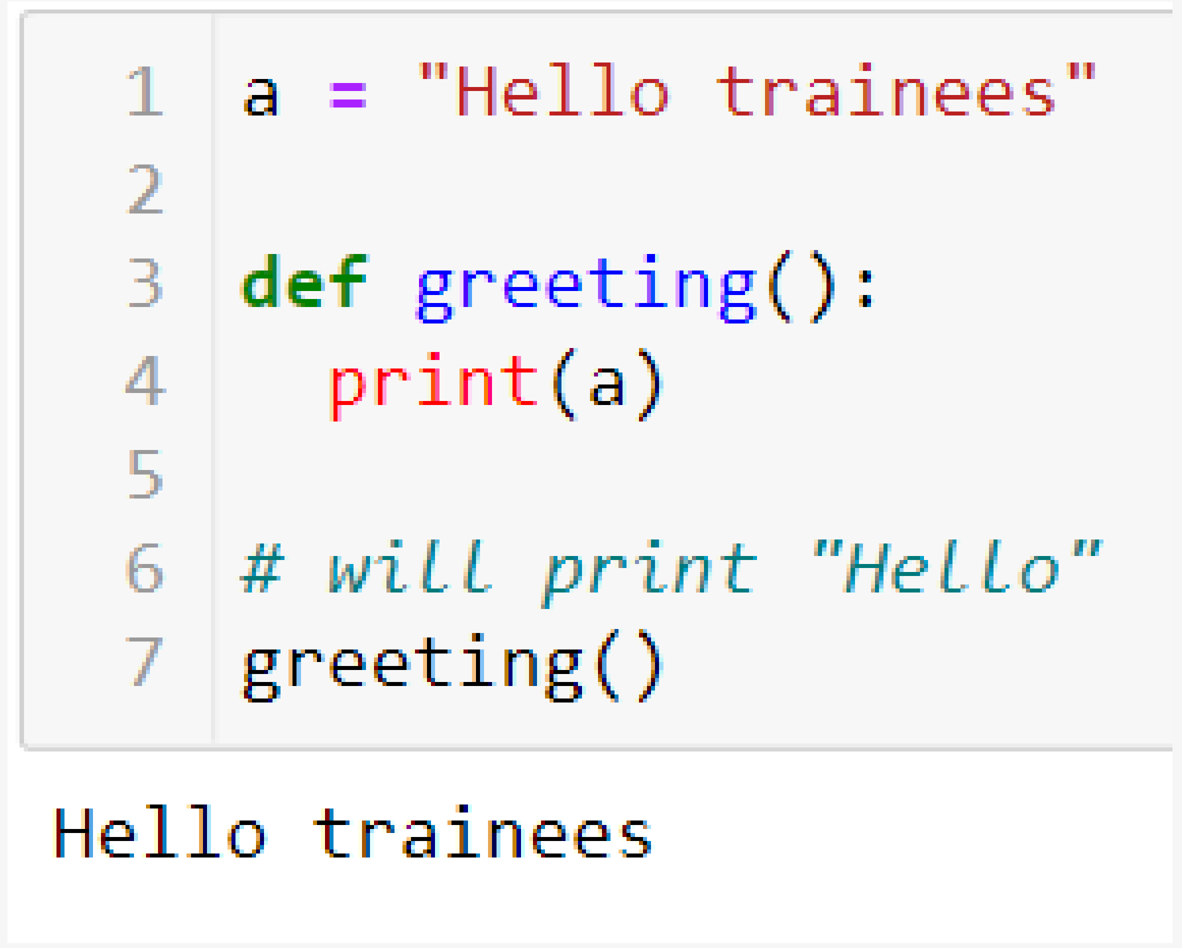
```
a = 10

def fun():
    a = 7
    print(a)

print(a)
fun()
```

```
10
7
```

The Learning Hub

Global Variables

- In Python a global variable is a variable defined inside a function.
- global variables can be used inside the scope of the function.
- **Example:**



```
a = "Hello trainees"

def greeting():
  print(a)

# will print "Hello"
greeting()
```

```
Hello trainees
```

The Learning Hub

References

- www.w3schools.com. (n.d.). W3Schools Free Online Web Tutorials. [online] Available at: http://W3schools.com. [Accessed 29 Sep. 2022].

- www.programiz.com. (n.d.). Python Lambda (Anonymous) Function. [online] Available at: https://www.programiz.com/python-programming/anonymous-function.

THANK YOU