Python Plus

- Functions
- Modules & Packages
- Working with Files
- Errors & Exceptions Handling

CLARUSWAY©
WAY TO REINVENT YOURSELF

Acquaintance with Functions

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

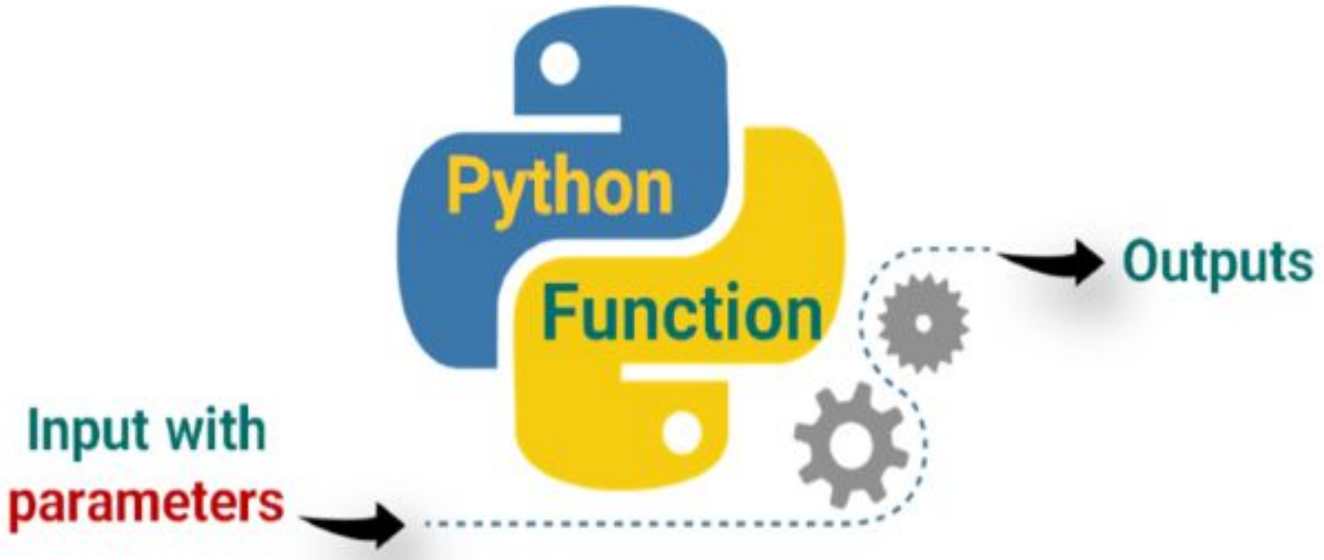
1 Introduction to Functions

What do you know about functions in Python?

Type at least 3 things...

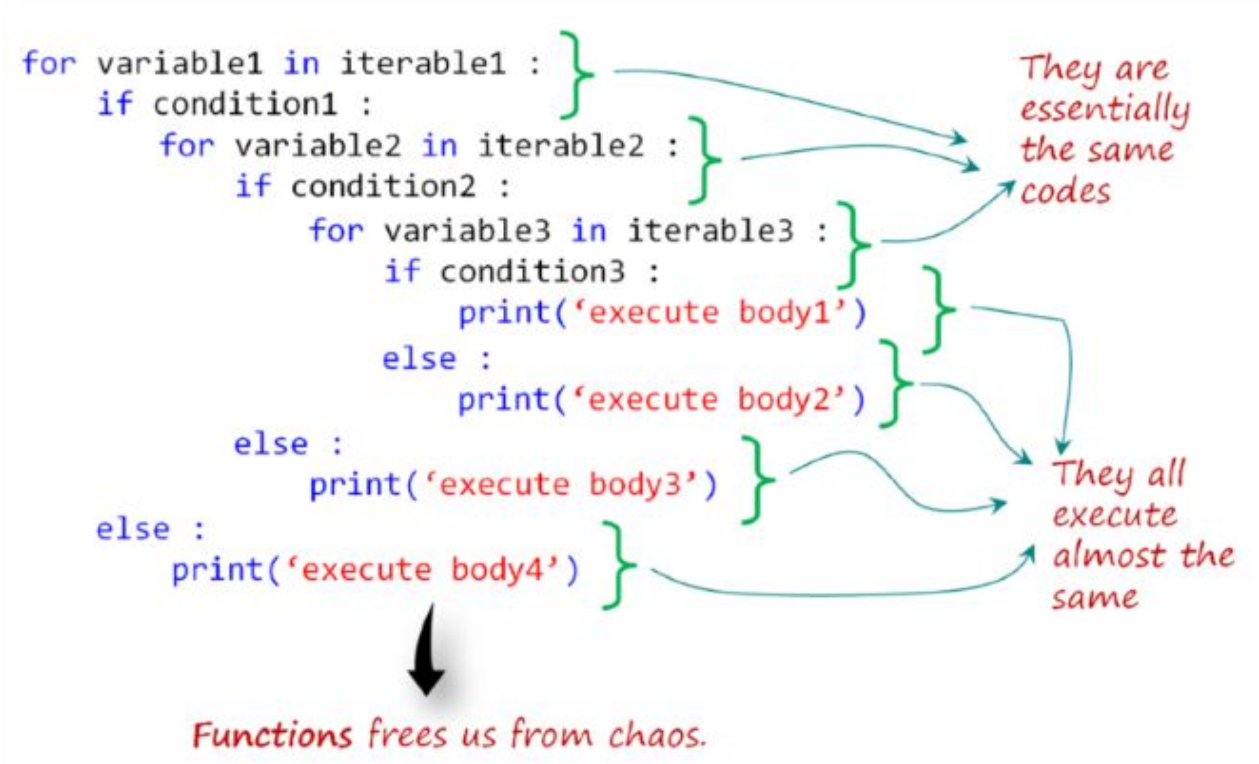
Students, write your response!

Introduction



Introduction (review)

- Functions free us from chaos.

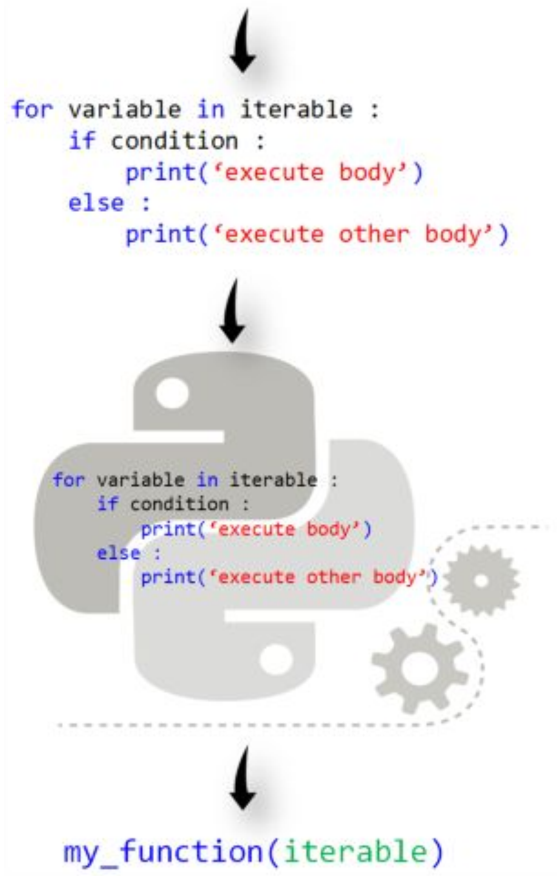


```
for variable1 in iterable1 :
    if condition1 :
        for variable2 in iterable2 :
            if condition2 :
                for variable3 in iterable3 :
                    if condition3 :
                        print('execute body1')
                    else :
                        print('execute body2')
            else :
                print('execute body3')
    else :
        print('execute body4')
```

They are essentially the same codes
They all execute almost the same
Functions frees us from chaos.

CLARUSWAY
WAY TO REINVENT YOURSELF

Introduction (review)



```
for variable in iterable :
    if condition :
        print('execute body')
    else :
        print('execute other body')
```

```
for variable in iterable :
    if condition :
        print('execute body')
    else :
        print('execute other body')
```

```
my_function(iterable)
```

You can choose a **piece of code** to convert into a function

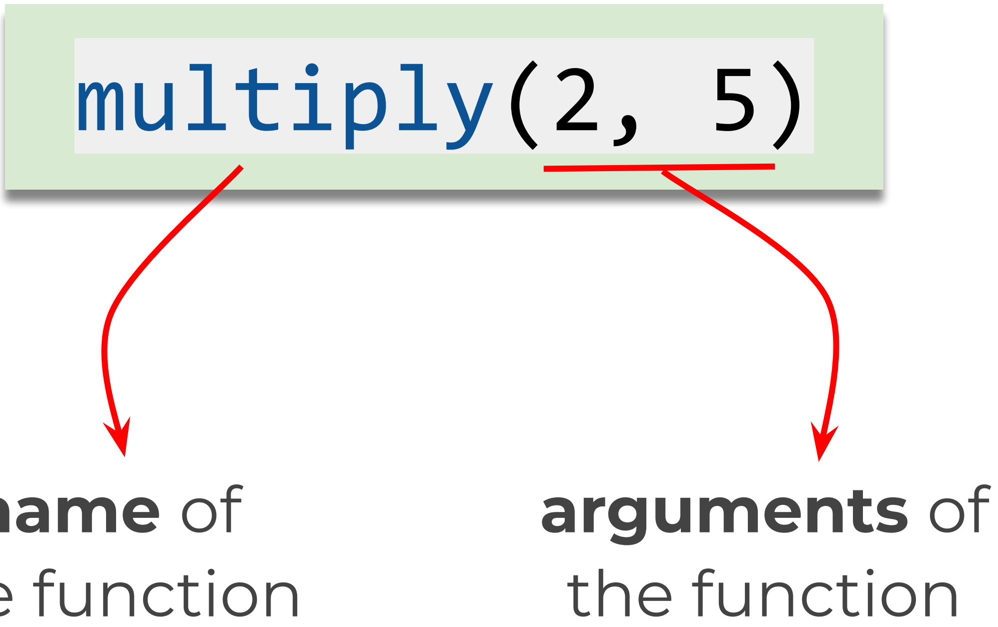
You can **create** a function which does what you want

You can **call** and **use** your function whenever and wherever you want

2 Calling a Function

Calling a Function Means Using It(review)

- Reading a function is very easy in Python.

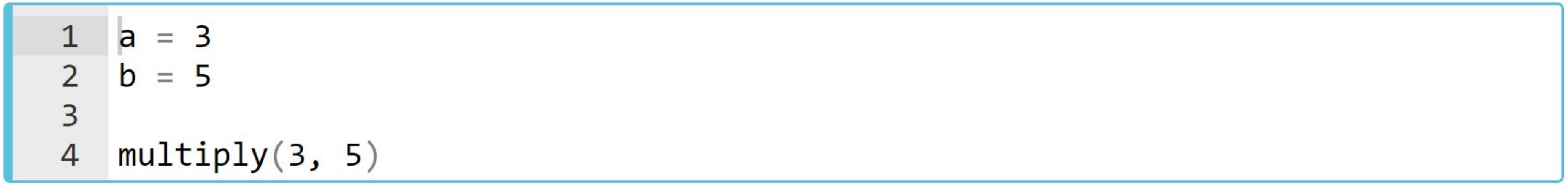


```python
multiply(2, 5)
```

Calling a Function Means Using It(review)

- Calling a function :

👇



```
a = 3
b = 5

multiply(3, 5)
```

Calling a Function Means Using It(review)

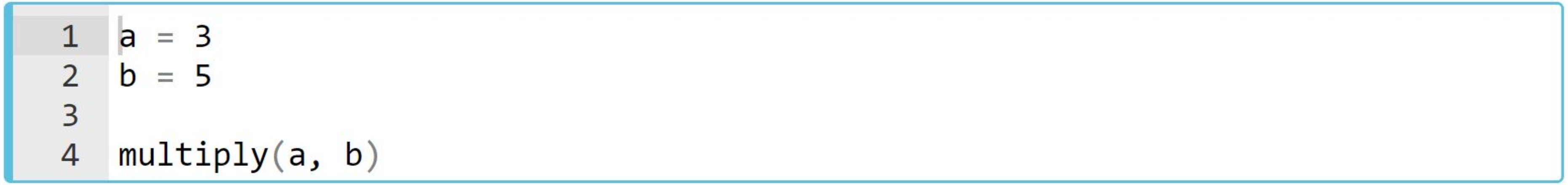
- Calling a function :

```python
a = 3
b = 5

multiply(3, 5)
```

```
15
```

Calling a Function Means Using It(review)

- Calling a function .



```python
a = 3
b = 5

multiply(a, b)
```

Calling a Function Means Using It(review)

▶

👇

```
a = 3
b = 5

multiply(a, b)
```

```
15
```

Calling `print()` Function (review)

- Calling `print()` function :

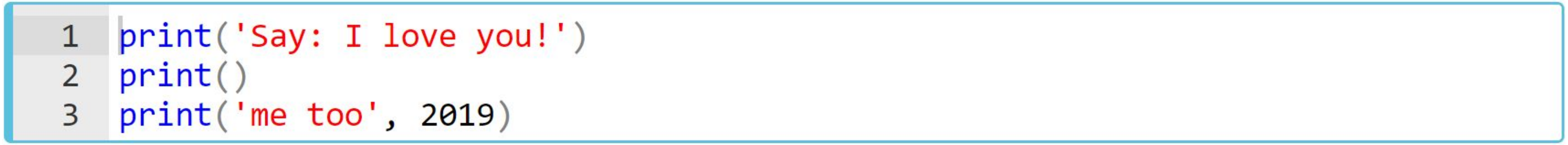
```python
print("Say : I love you!")
```

name of the function

argument of the function

Calling `print()` Function (review)

- Take a look at this pre-class example 👇



```python
print('Say: I love you!')
print()
print('me too', 2019)
```

Calling `print()` Function (review)

- Take a look at the example 👇

```python
print('Say: I love you!')
print()
print('me too', 2019)
```

```
Say: I love you!

me too 2019
```

3

Built-in Functions

Built-in Functions (review)

- The number of built-in functions :

In the latest version
Python 3.10

→ **71**

Built-in Functions (review)

- So far we have learned 👇

```
print(), int(), list(), input(), range()
```

- Some of them return bool type 👇

```
all(iterable), any(iterable), callable(object)
```

Built-in Functions (review)

- Some of them help you convert data types 👇

```
bool(), float(), int(), str()
```

- For creating and processing the collection types. 👇

```
 dict(), list(), tuple(), set(), len(), zip(),
filter(function, iterable), enumerate(iterable)
```

Built-in Functions (review)

- Some others tackle numbers. 👇

```
max(), min(), sum(), round()
```

- The others are built for special purposes. 👇

```
map(function, iterable, ...), eval(expression[,
globals[, locals]]), sorted(iterable), open(),
dir([object]), help([object])
```

As mentioned in the **pre-class** content, I took a look at the ***built-in functions*** in the official Python docs.

True

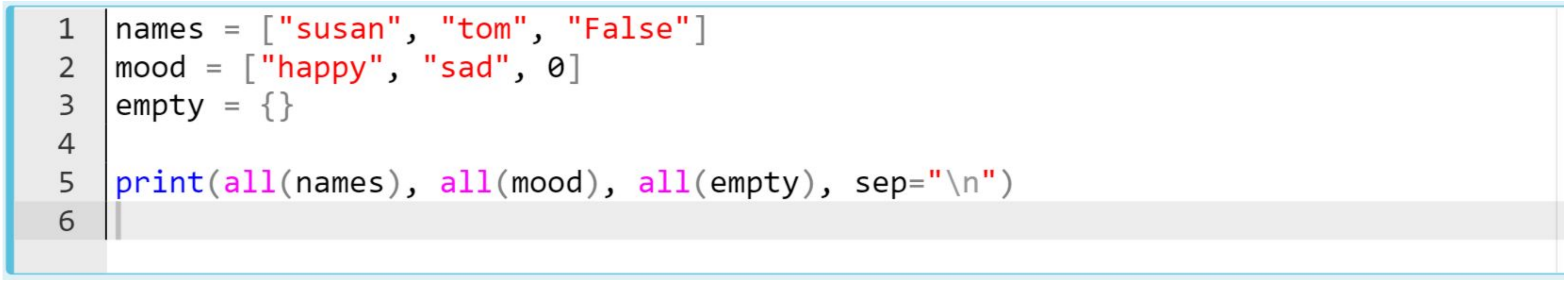
False

Built-in Functions

- Leaking snow water in your ear.

Built-in Functions

- `all()` function.



```
names = ["susan", "tom", "False"]
mood = ["happy", "sad", 0]
empty = {}

print(all(names), all(mood), all(empty), sep="\n")

```

What is the output? Try to figure out in your mind…

Students, write your response!

Built-in Functions

- `all()` function.

`all()` method returns:

Truth table for all()

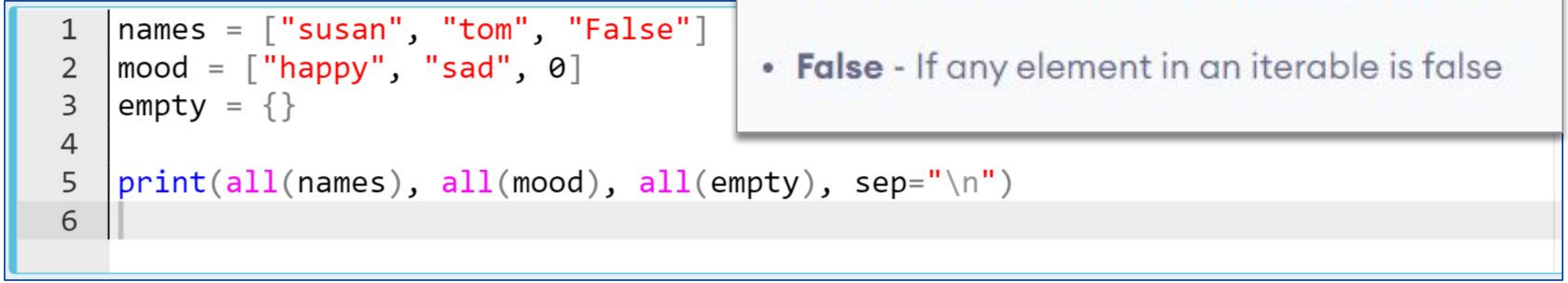
When	Return Value
All values are true	True
All values are false	False
One value is true (others are false)	False
One value is false (others are true)	False
Empty Iterable	True

Outp

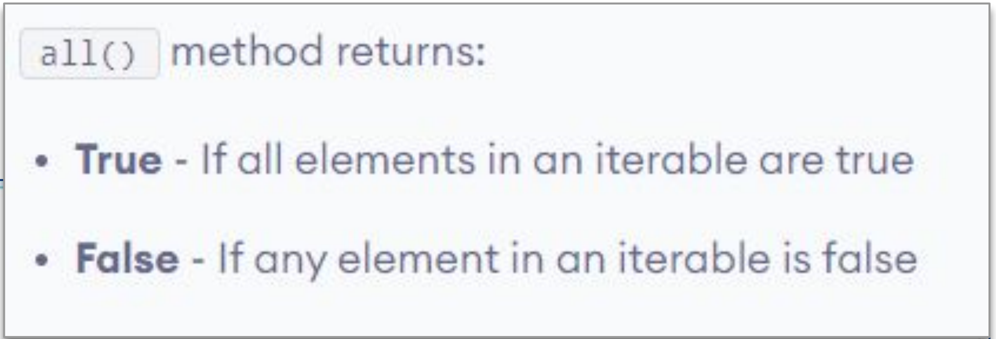
```
Tru
Fal
Tru
```

Built-in Functions

- `all()` function.

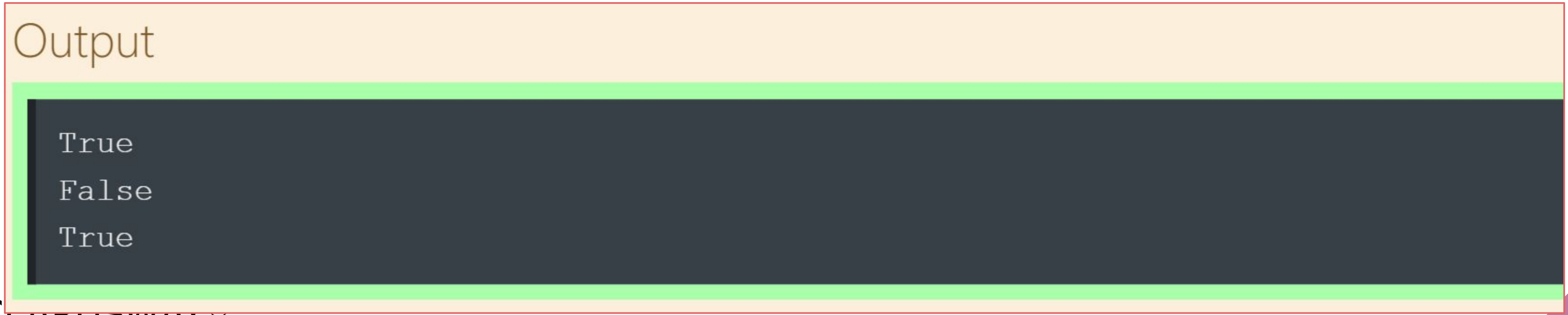


```python
names = ["susan", "tom", "False"]
mood = ["happy", "sad", 0]
empty = {}

print(all(names), all(mood), all(empty), sep="\n")
```



`all()` method returns:

- **True** - If all elements in an iterable are true
- **False** - If any element in an iterable is false

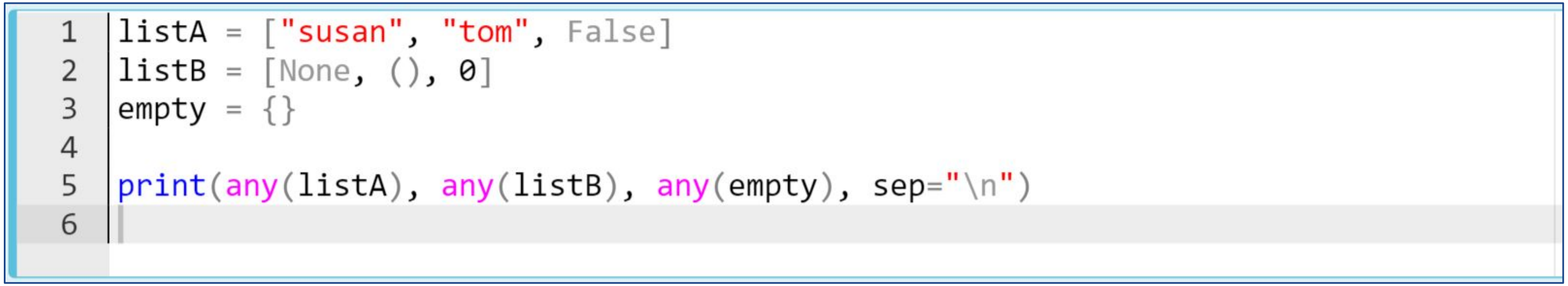


Output

```
True
False
True
```

Built-in Functions

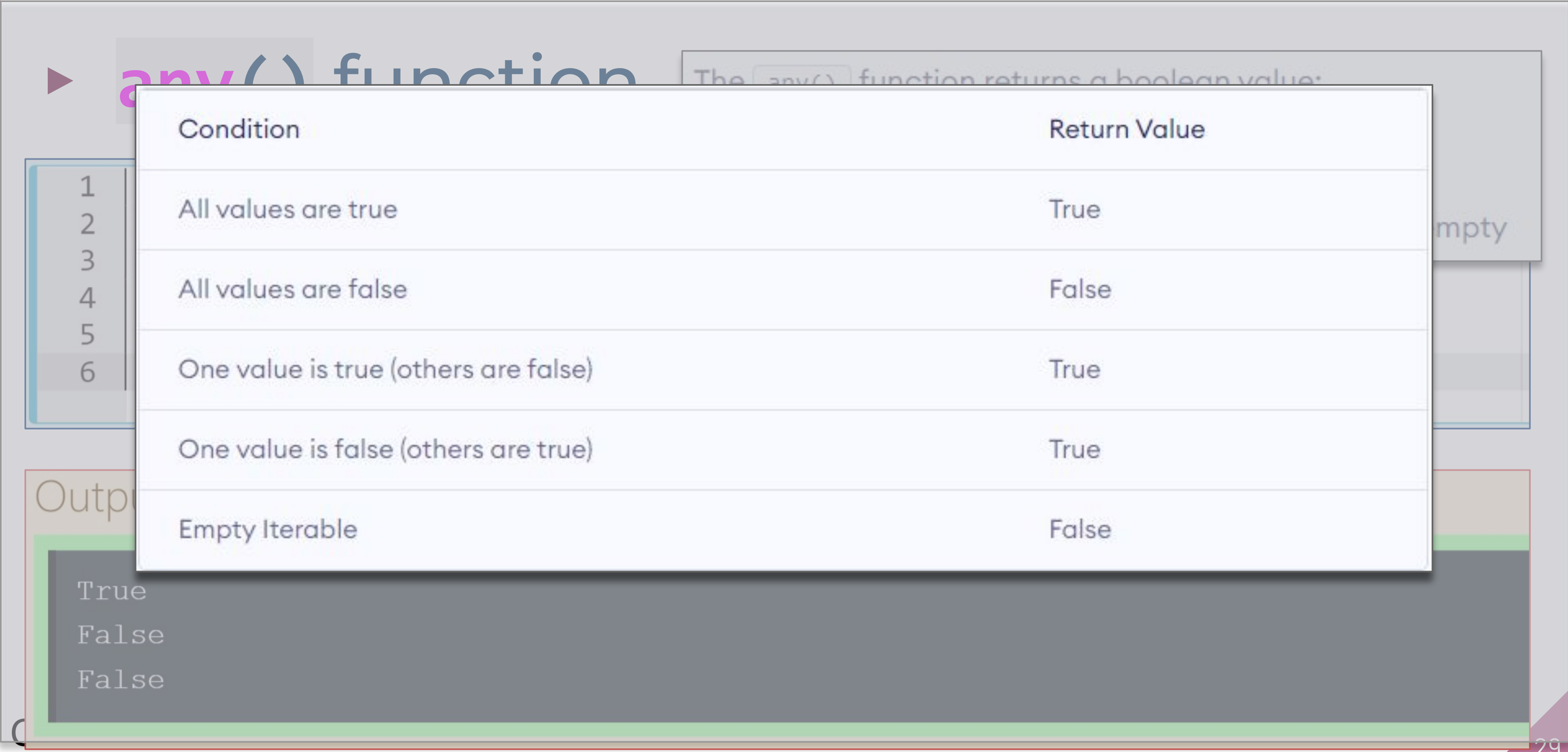
- `any()` function.



```python
listA = ["susan", "tom", False]
listB = [None, (), 0]
empty = {}

print(any(listA), any(listB), any(empty), sep="\n")

```

What is the output? Try to figure out in your mind...

Built-in Functions



any() function

The any() function returns a boolean value:

mpty

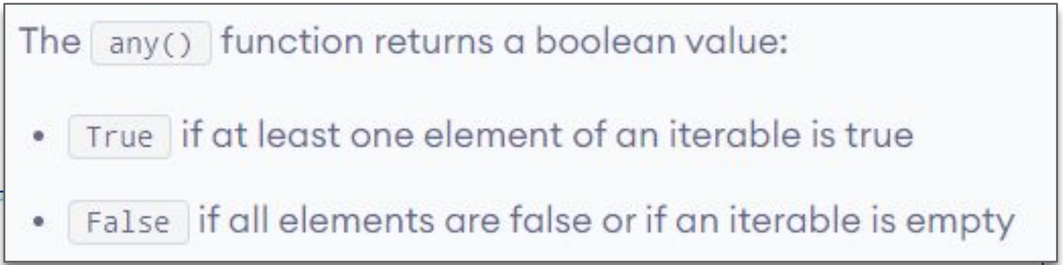
Condition	Return Value
All values are true	True
All values are false	False
One value is true (others are false)	True
One value is false (others are true)	True
Empty Iterable	False

Outp

```
True
False
False
```

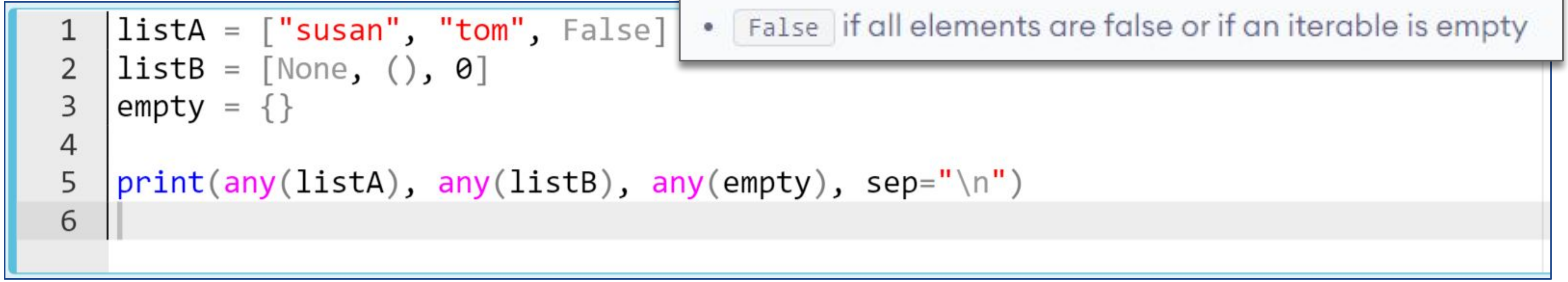
Built-in Functions

- `any()` function.

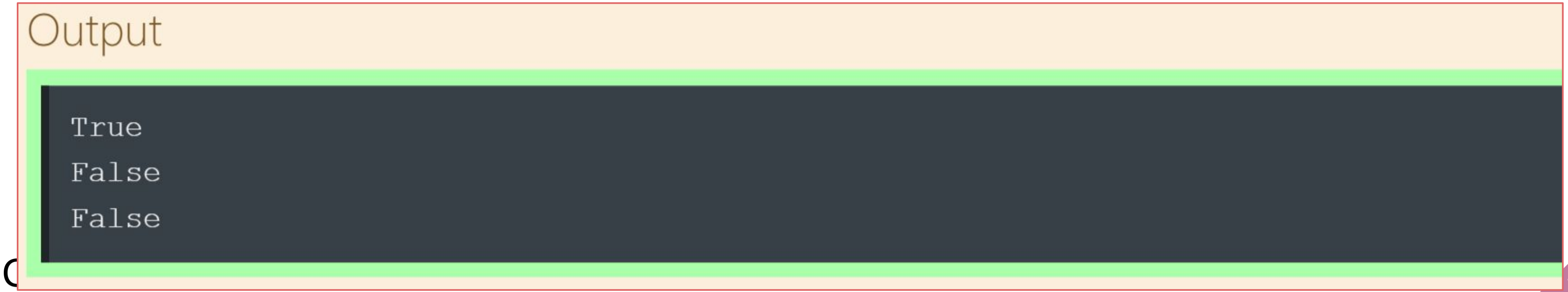


The `any()` function returns a boolean value:

- `True` if at least one element of an iterable is true
- `False` if all elements are false or if an iterable is empty



```python
listA = ["susan", "tom", False]
listB = [None, (), 0]
empty = {}

print(any(listA), any(listB), any(empty), sep="\n")
```

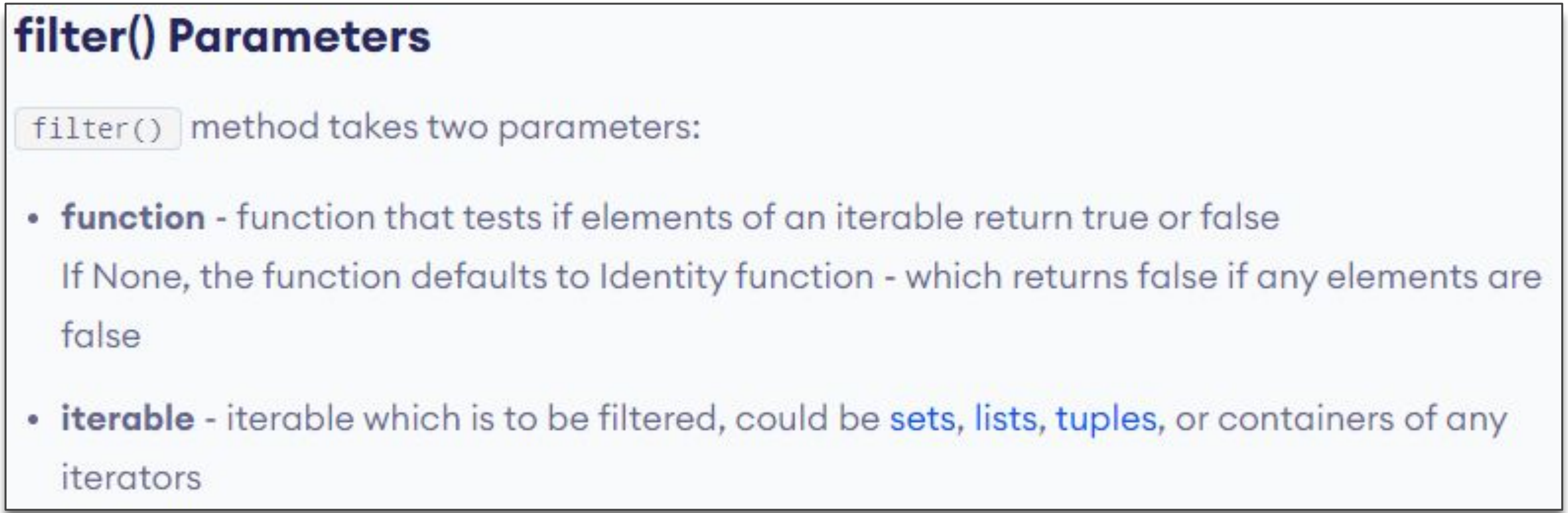


Output

```
True
False
False
```

Built-in Functions

- `filter(function, iterable)`.



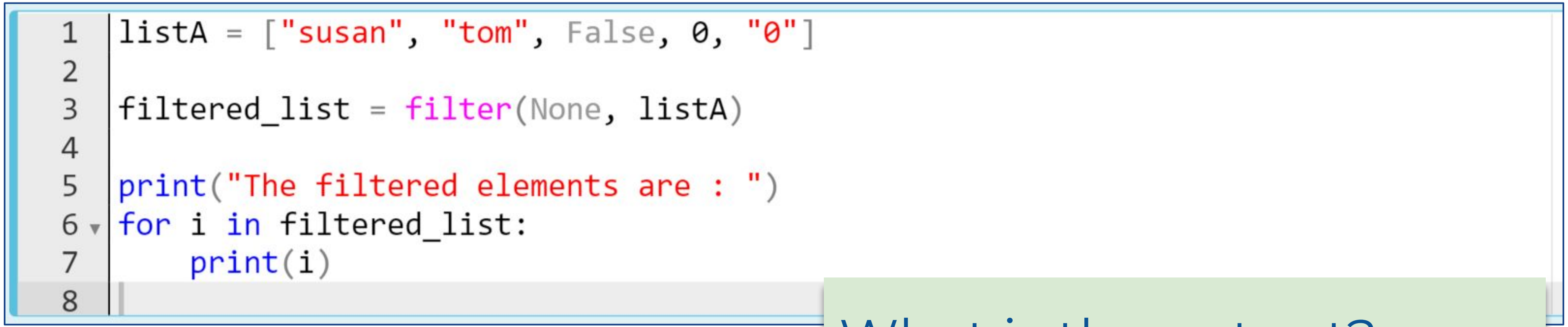
filter() Parameters

`filter()` method takes two parameters:

- **function** - function that tests if elements of an iterable return true or false
  If None, the function defaults to Identity function - which returns false if any elements are false
- **iterable** - iterable which is to be filtered, could be sets, lists, tuples, or containers of any iterators

Built-in Functions

- `filter(function, iterable)`.

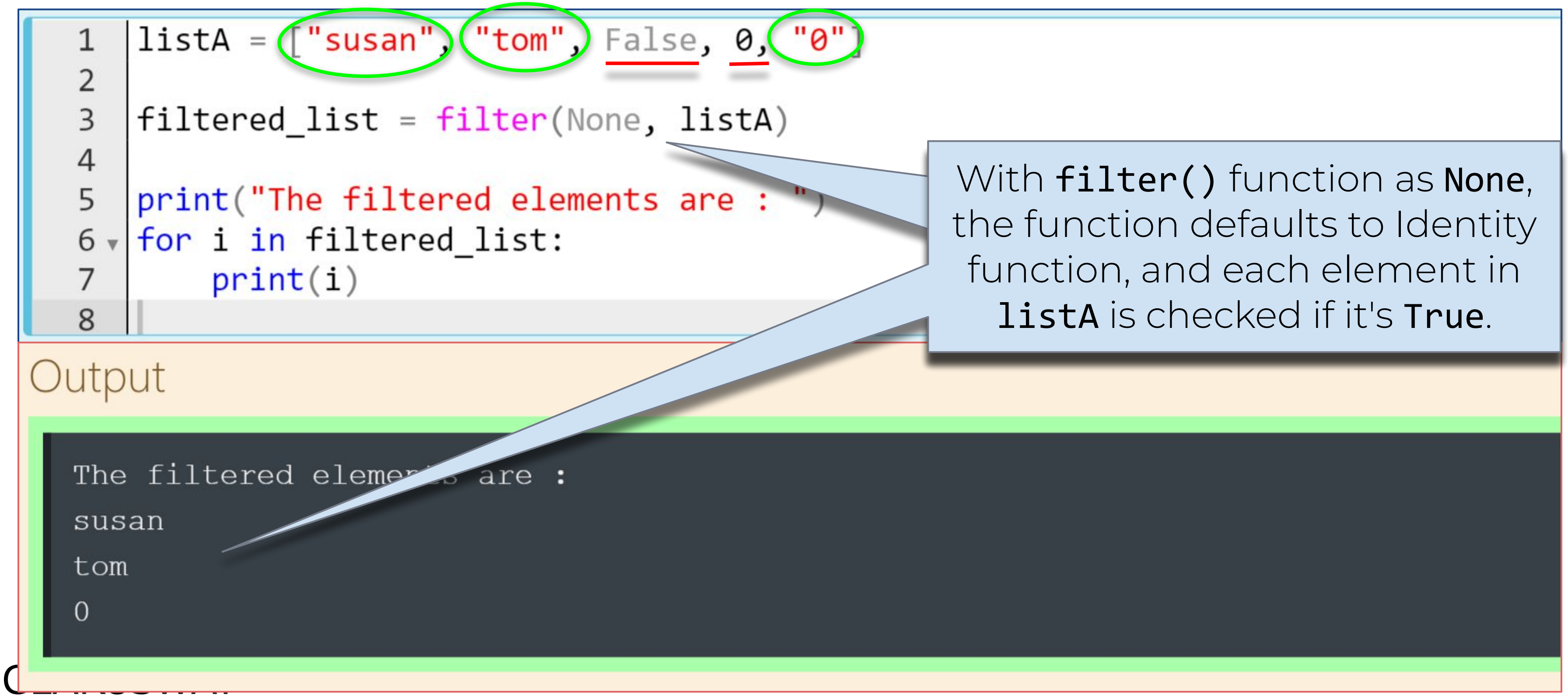


```python
listA = ["susan", "tom", False, 0, "0"]

filtered_list = filter(None, listA)

print("The filtered elements are : ")
for i in filtered_list:
    print(i)

```

What is the output? Try to figure out in your mind...

Built-in Functions

- `filter(function, iterable)`.



```python
listA = ["susan", "tom", False, 0, "0"]

filtered_list = filter(None, listA)

print("The filtered elements are : ")
for i in filtered_list:
    print(i)

```

Output

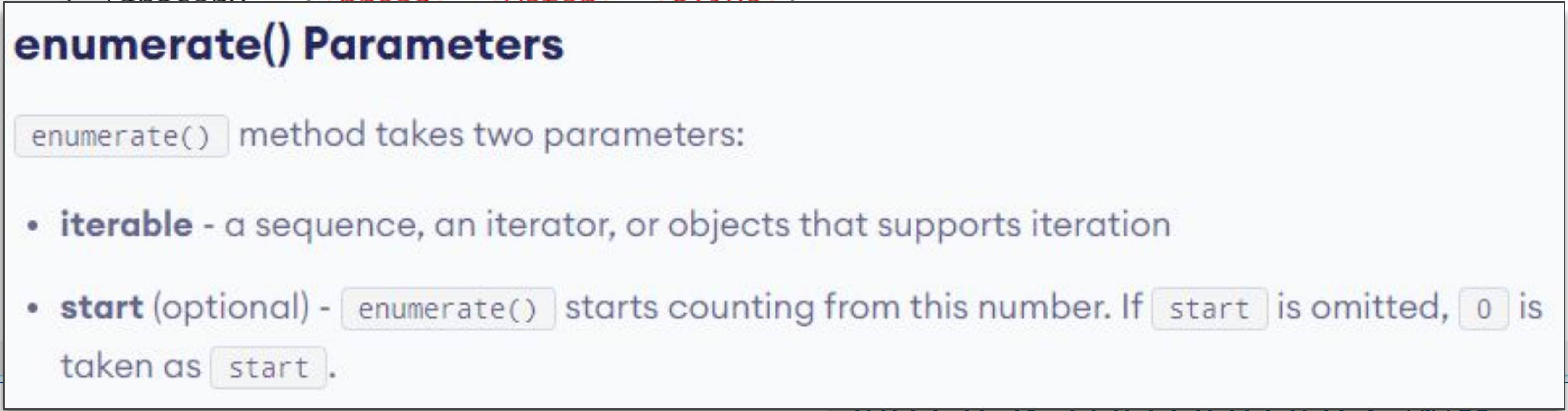
```
The filtered elements are :
susan
tom
0
```

With **filter()** function as **None**, the function defaults to Identity function, and each element in **listA** is checked if it's **True**.

Built-in Functions

- `enumerate(iterable, start=0)`.

```
grocery = ['bread', 'water', 'olive']
```



enumerate() Parameters

`enumerate()` method takes two parameters:

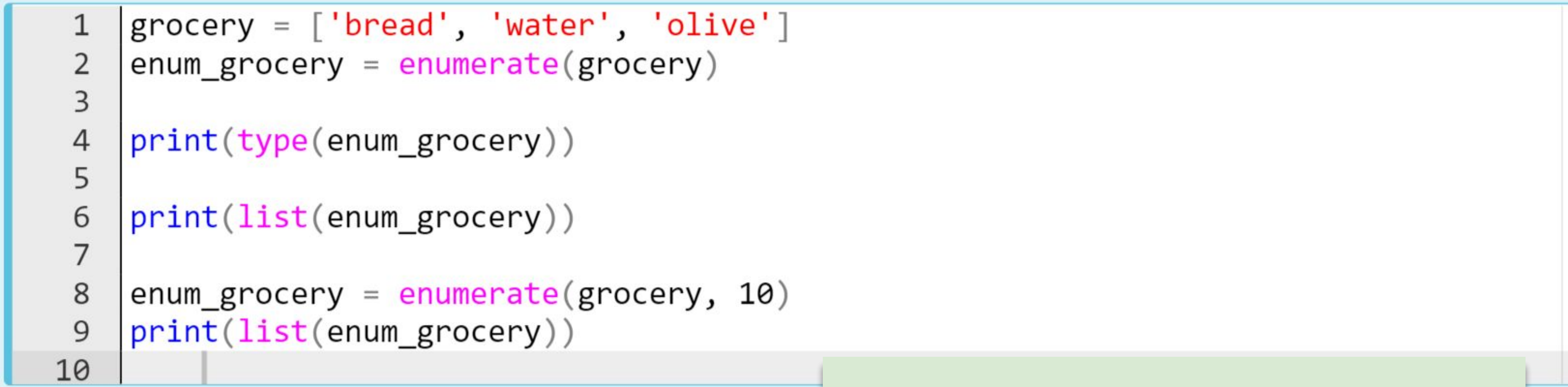
- **iterable** - a sequence, an iterator, or objects that supports iteration
- **start** (optional) - `enumerate()` starts counting from this number. If `start` is omitted, `0` is taken as `start`.

What is the output? Try to figure out in your mind...

CLARUSWAY©
WAY TO REINVENT YOURSELF

Built-in Functions

- `enumerate(iterable, start=0)`.



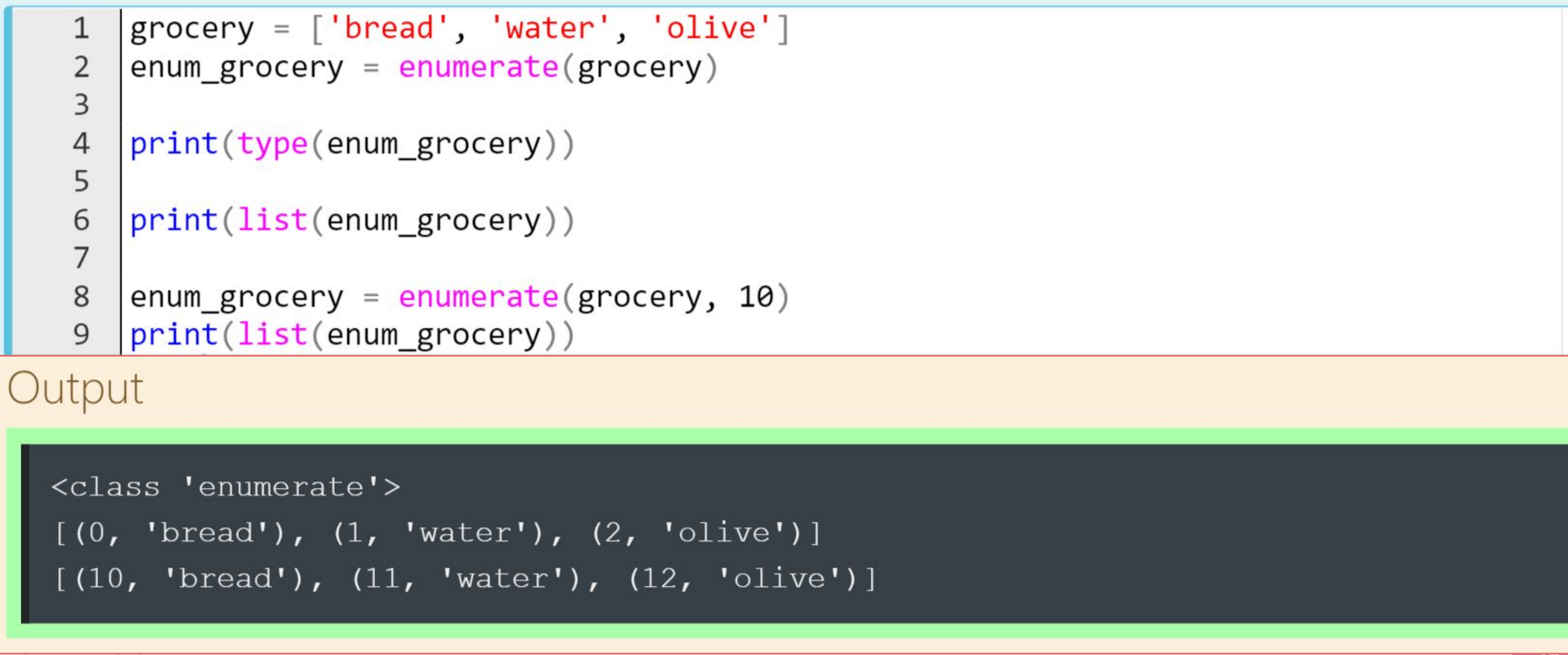
```
grocery = ['bread', 'water', 'olive']
enum_grocery = enumerate(grocery)

print(type(enum_grocery))

print(list(enum_grocery))

enum_grocery = enumerate(grocery, 10)
print(list(enum_grocery))

```

What is the output? Try to figure out in your mind...

CLARUSWAY©
WAY TO REINVENT YOURSELF

Built-in Functions

- `enumerate(iterable, start=0)`.



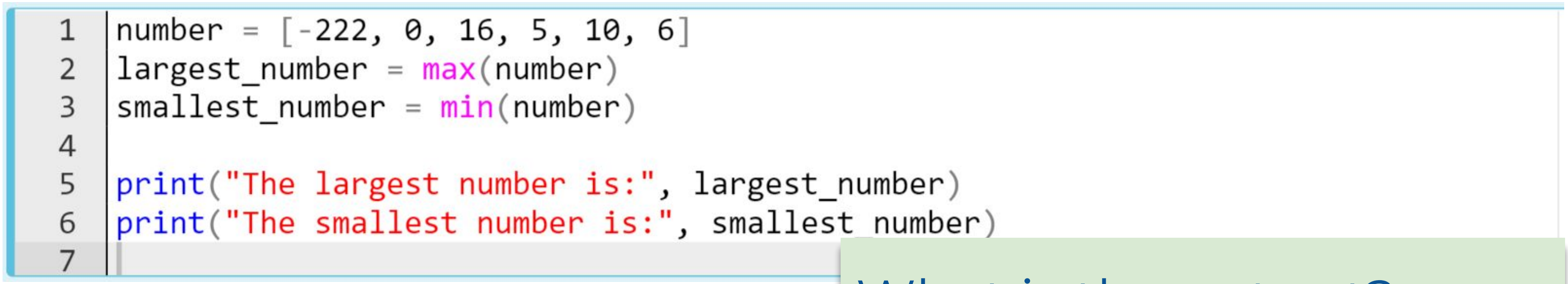
```python
grocery = ['bread', 'water', 'olive']
enum_grocery = enumerate(grocery)

print(type(enum_grocery))

print(list(enum_grocery))

enum_grocery = enumerate(grocery, 10)
print(list(enum_grocery))
```

Output

```
<class 'enumerate'>
[(0, 'bread'), (1, 'water'), (2, 'olive')]
[(10, 'bread'), (11, 'water'), (12, 'olive')]
```

Built-in Functions

- `max(iterable), min(iterable)`.

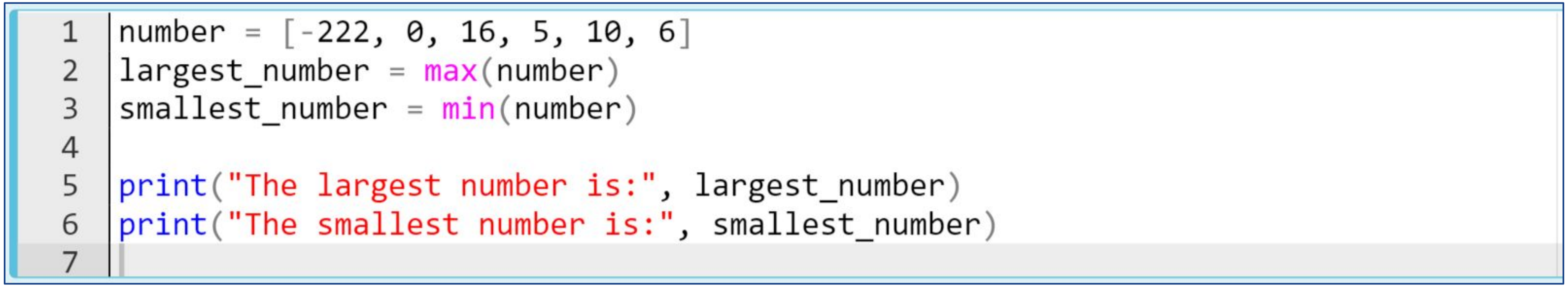


```python
number = [-222, 0, 16, 5, 10, 6]
largest_number = max(number)
smallest_number = min(number)

print("The largest number is:", largest_number)
print("The smallest number is:", smallest_number)
```

What is the output? Try to figure out in your mind...

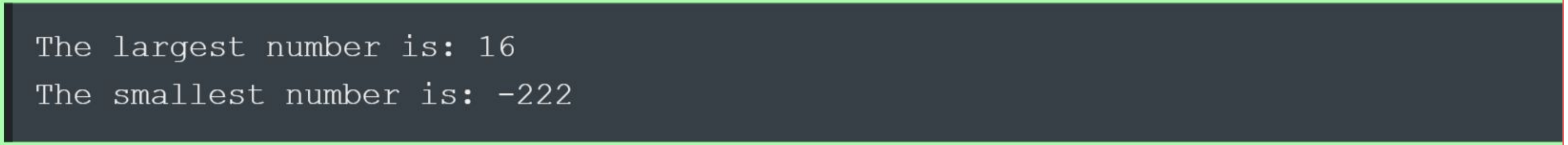
Built-in Functions

- `max(iterable), min(iterable).`



```python
number = [-222, 0, 16, 5, 10, 6]
largest_number = max(number)
smallest_number = min(number)

print("The largest number is:", largest_number)
print("The smallest number is:", smallest_number)
```

Output

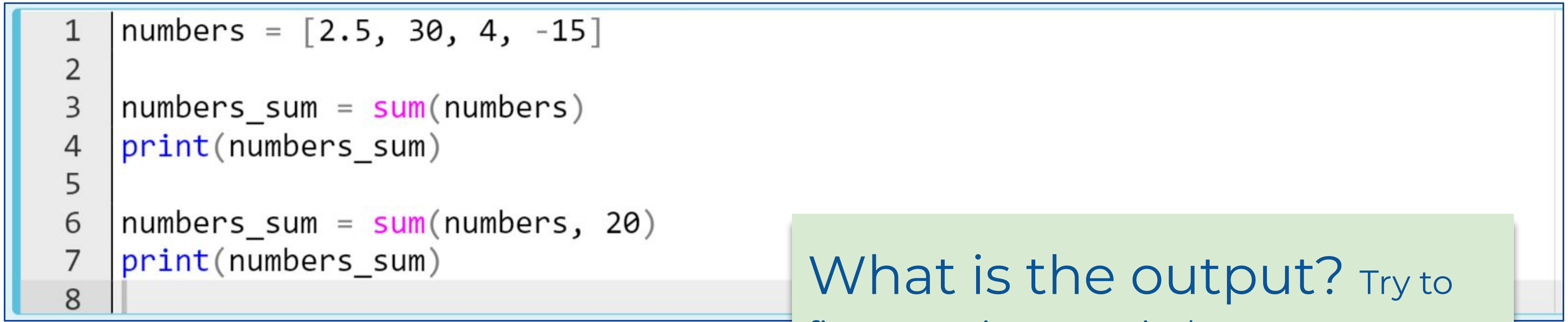


```
The largest number is: 16
The smallest number is: -222
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

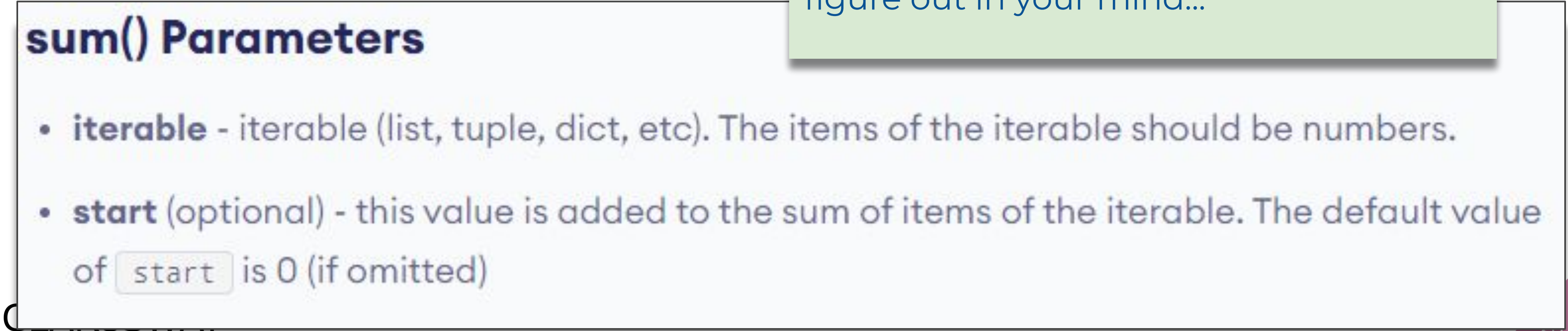
Built-in Functions

- `sum(iterable, start)`.



```
numbers = [2.5, 30, 4, -15]

numbers_sum = sum(numbers)
print(numbers_sum)

numbers_sum = sum(numbers, 20)
print(numbers_sum)

```

What is the output? Try to figure out in your mind...

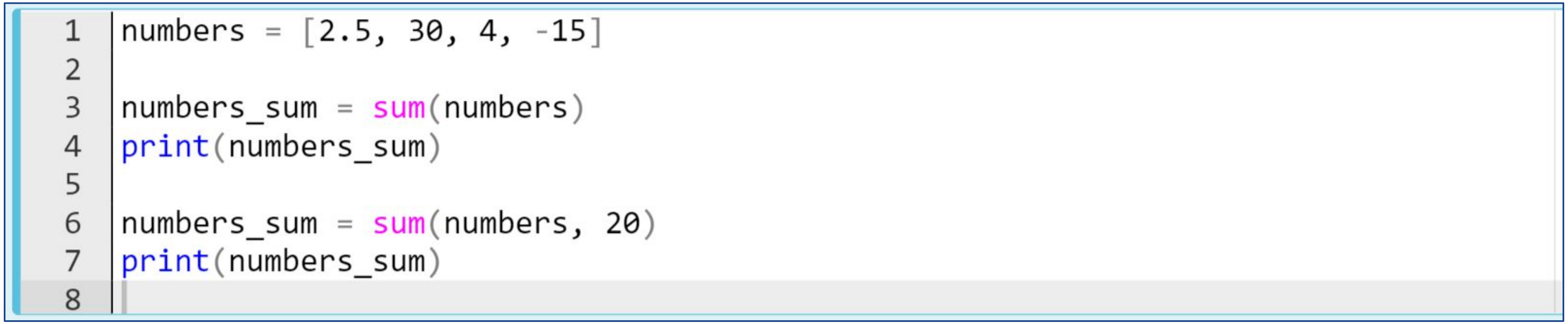


sum() Parameters

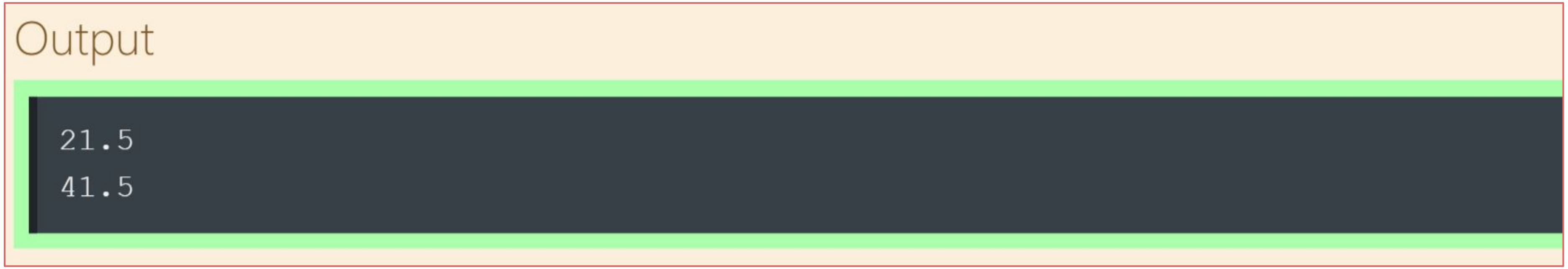
- **iterable** - iterable (list, tuple, dict, etc). The items of the iterable should be numbers.
- **start** (optional) - this value is added to the sum of items of the iterable. The default value of `start` is 0 (if omitted)

Built-in Functions

▸ `sum(iterable)`.



```python
numbers = [2.5, 30, 4, -15]

numbers_sum = sum(numbers)
print(numbers_sum)

numbers_sum = sum(numbers, 20)
print(numbers_sum)

```

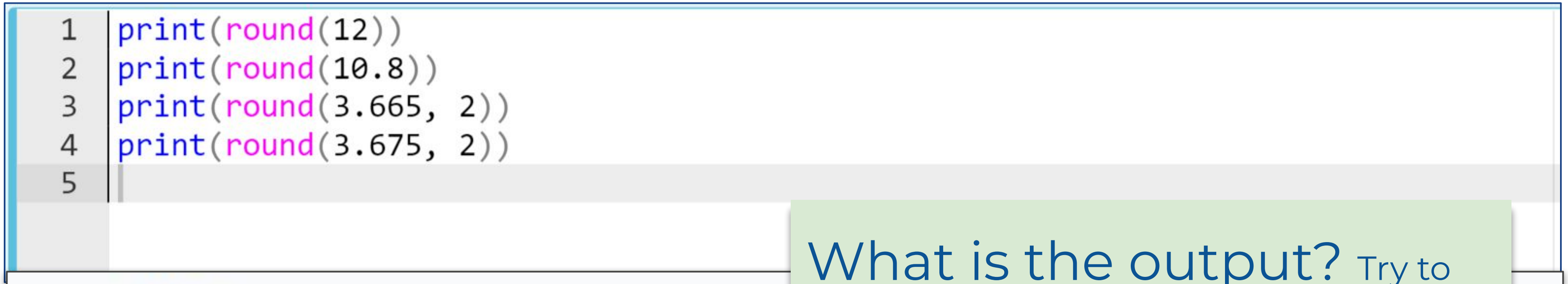


Output

```
21.5
41.5
```

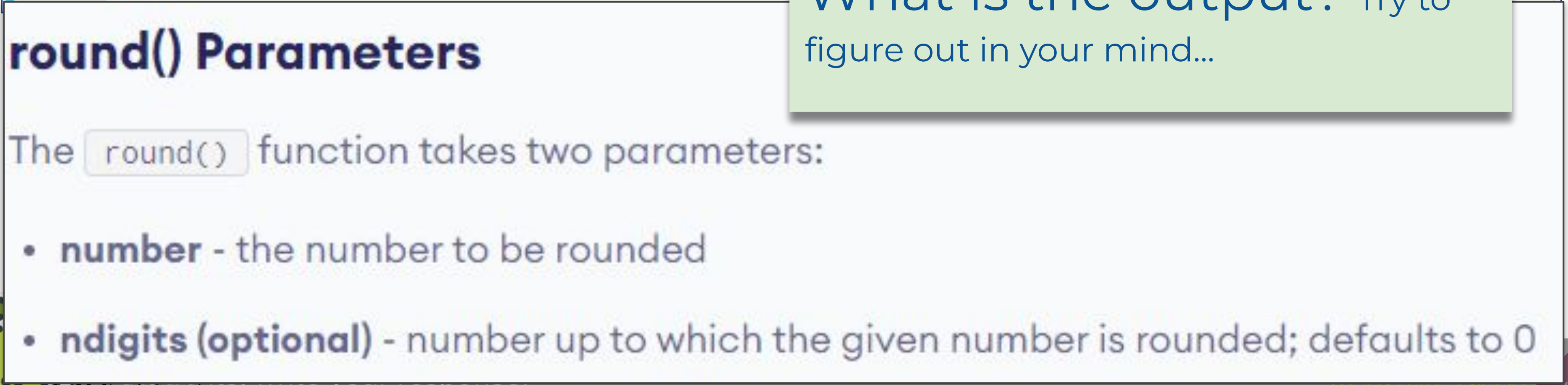
Built-in Functions

- `round(numbers, ndigits)`.



```python
print(round(12))
print(round(10.8))
print(round(3.665, 2))
print(round(3.675, 2))
```

What is the output? Try to figure out in your mind…



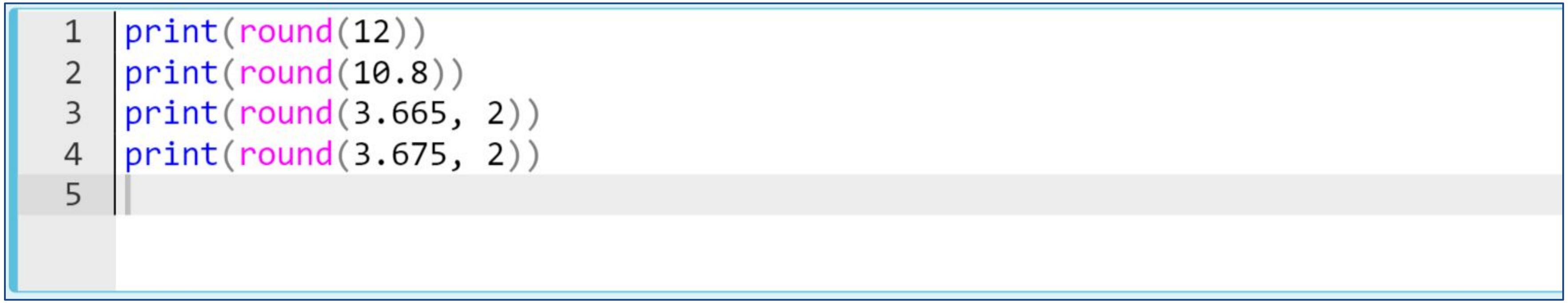
round() Parameters

The `round()` function takes two parameters:

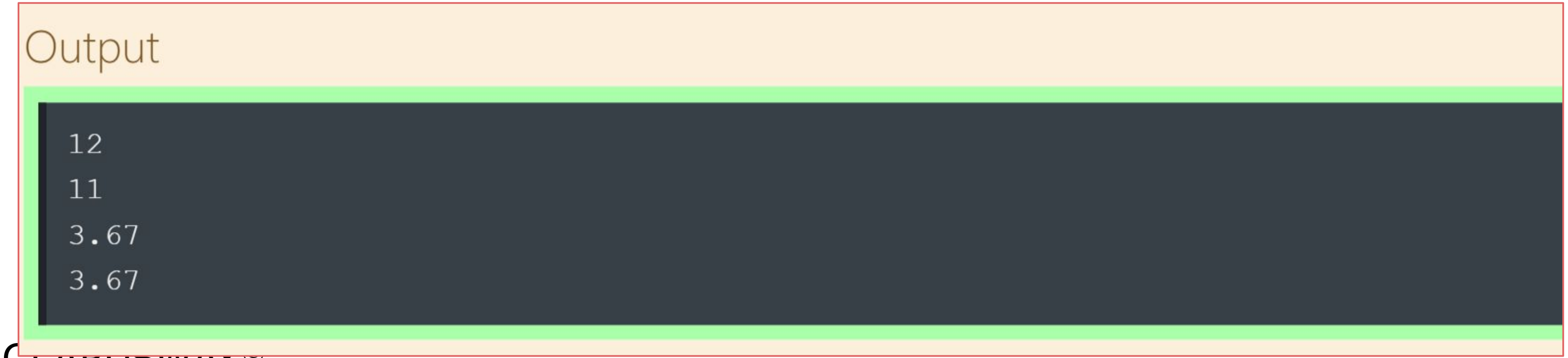
- **number** - the number to be rounded
- **ndigits (optional)** - number up to which the given number is rounded; defaults to 0

Built-in Functions

- `round(numbers, ndigits)`.



```
print(round(12))
print(round(10.8))
print(round(3.665, 2))
print(round(3.675, 2))
```



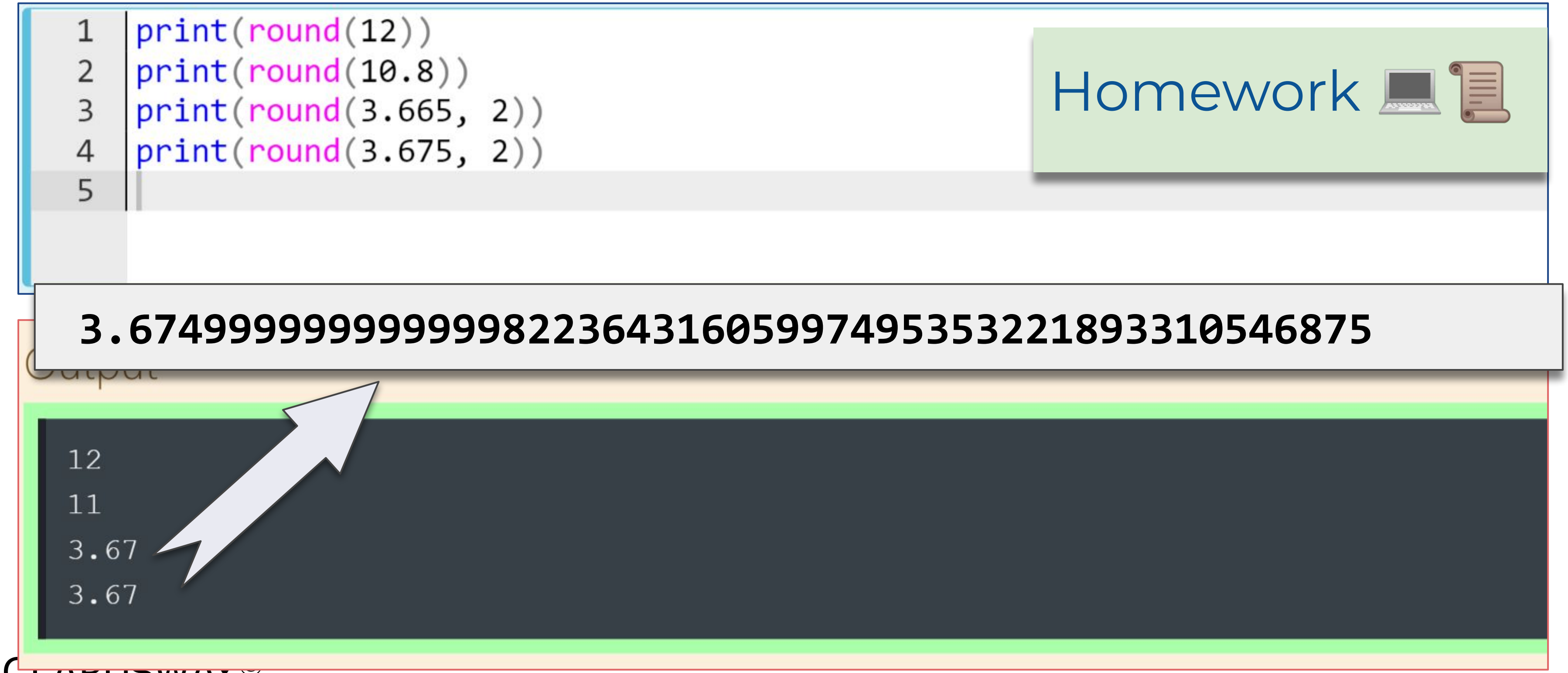
Output

```
12
11
3.67
3.67
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Built-in Functions

- `round(numbers, ndigits)`.



```python
print(round(12))
print(round(10.8))
print(round(3.665, 2))
print(round(3.675, 2))
```

Homework 💻📜

```
3.67499999999999982236431605997495353221893310546875
```

Output

```
12
11
3.67
3.67
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

THANKS!

End of the Lesson

(Acquaintance with Func)

next Lesson

[Defining a Function]

click above