FUNCTIONS (FÖLL)

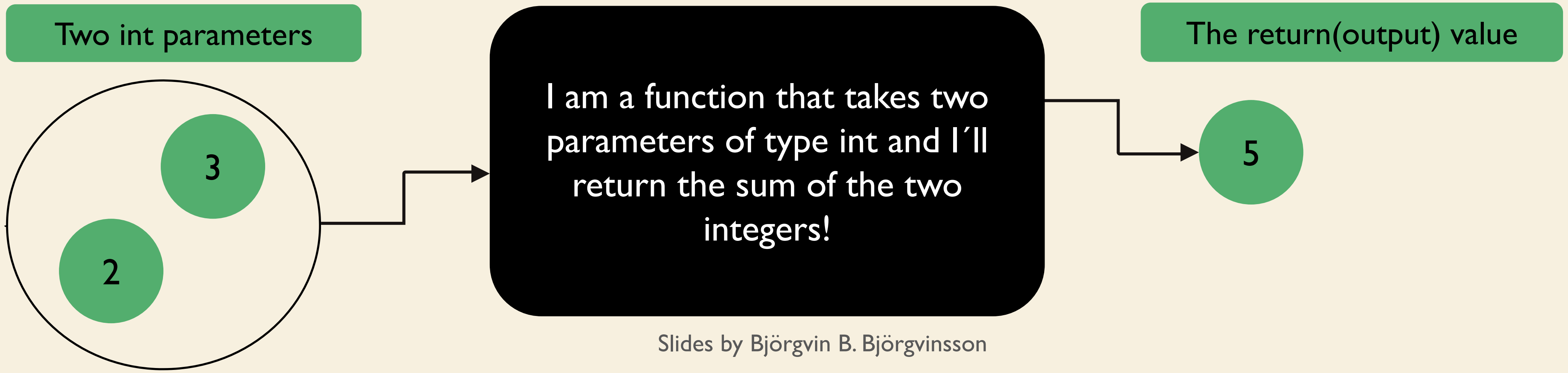
FUNCTIONS

- A Function is a "**self contained**" section of code that does a **specific task**.
- Functions can take in data through parameters, process it, and then "return" a result.
- One of the biggest benefits of using functions is that once it is written, it can be used over and over again.
- To use a function you must **"invoke"** or **"call"** it!
- You can think of a **function** as a tiny program or a sub-program that other programs can use!

FUNCTIONS

THE BLACK BOX ANALOGY

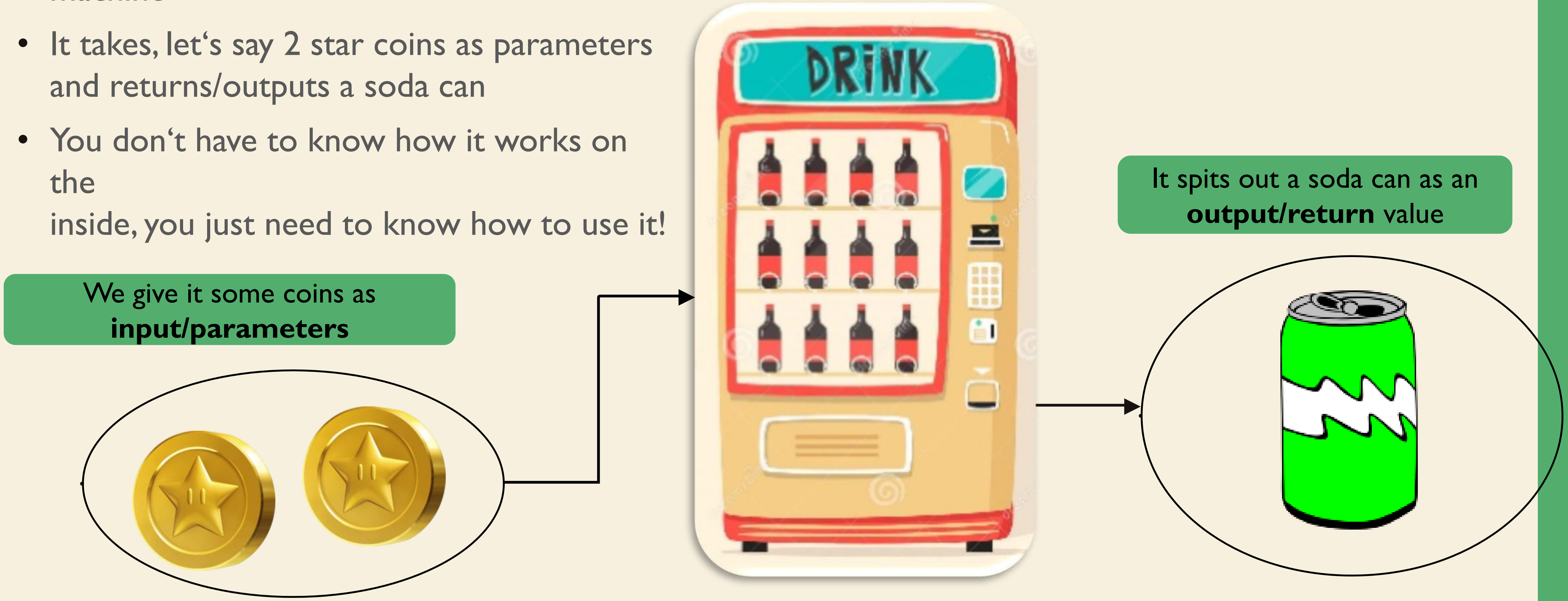
- You can think of a function as a **black box**
  - That means you don‘t need to know how the function is implemented
  - You just need to know it‘s name
  - You need to know how to use it
  - You need to know what parameters it takes in and what it outputs


Two int parameters
3
2
I am a function that takes two parameters of type int and I´ll return the sum of the two integers!
The return(output) value
5

FUNCTIONS

- A real life analogy might be a soda vending machine
- It takes, let‘s say 2 star coins as parameters and returns/outputs a soda can
- You don‘t have to know how it works on the
  inside, you just need to know how to use it!

We give it some coins as **input/parameters**



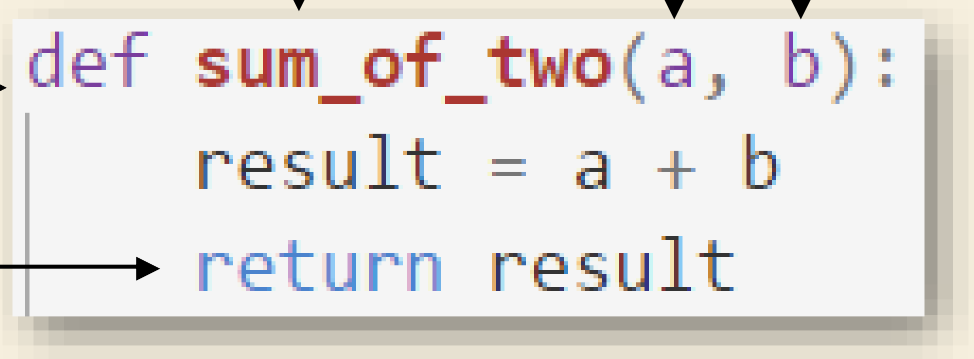
It spits out a soda can as an **output/return** value

FUNCTIONS

- You can think of functions as subtasks within your program or even a sub program
- Using functions makes programs
  - Easier to understand
  - Easier to change
  - Easier to write
  - Easier to test
  - Easier to debug
  - Easier for teams to develop

FUNCTIONS

- Let‘s see an example of a function
  - This is a function that returns the sum of the parameters
    - It takes two values as parameters



```
def sum_of_two(a, b):
    result = a + b
    return result
```

sum_of_two is the name of the function.

These are the **parameters**. They are listed between the parentheses and seperated by a comma.

def is a keyword that means: Here a function is being **defined**

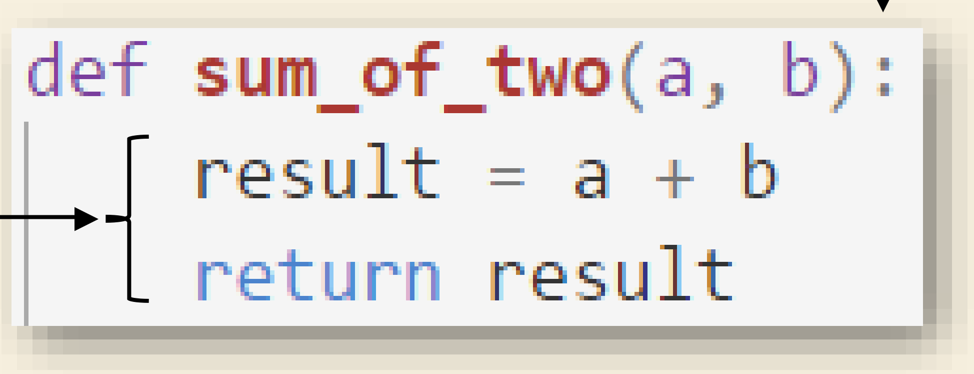
The **return** keyword followed by the **return value**. In this function the return value is the **value** of the variable sum!

FUNCTIONS

- Notice the colon and indentation
  - We use the colon and indentation just like in if statements and loops

The body of the function comes after the colon

This is the body of the function! We use indentation to indicate the body of the function



```
def sum_of_two(a, b):
    result = a + b
    return result
```

FUNCTIONS

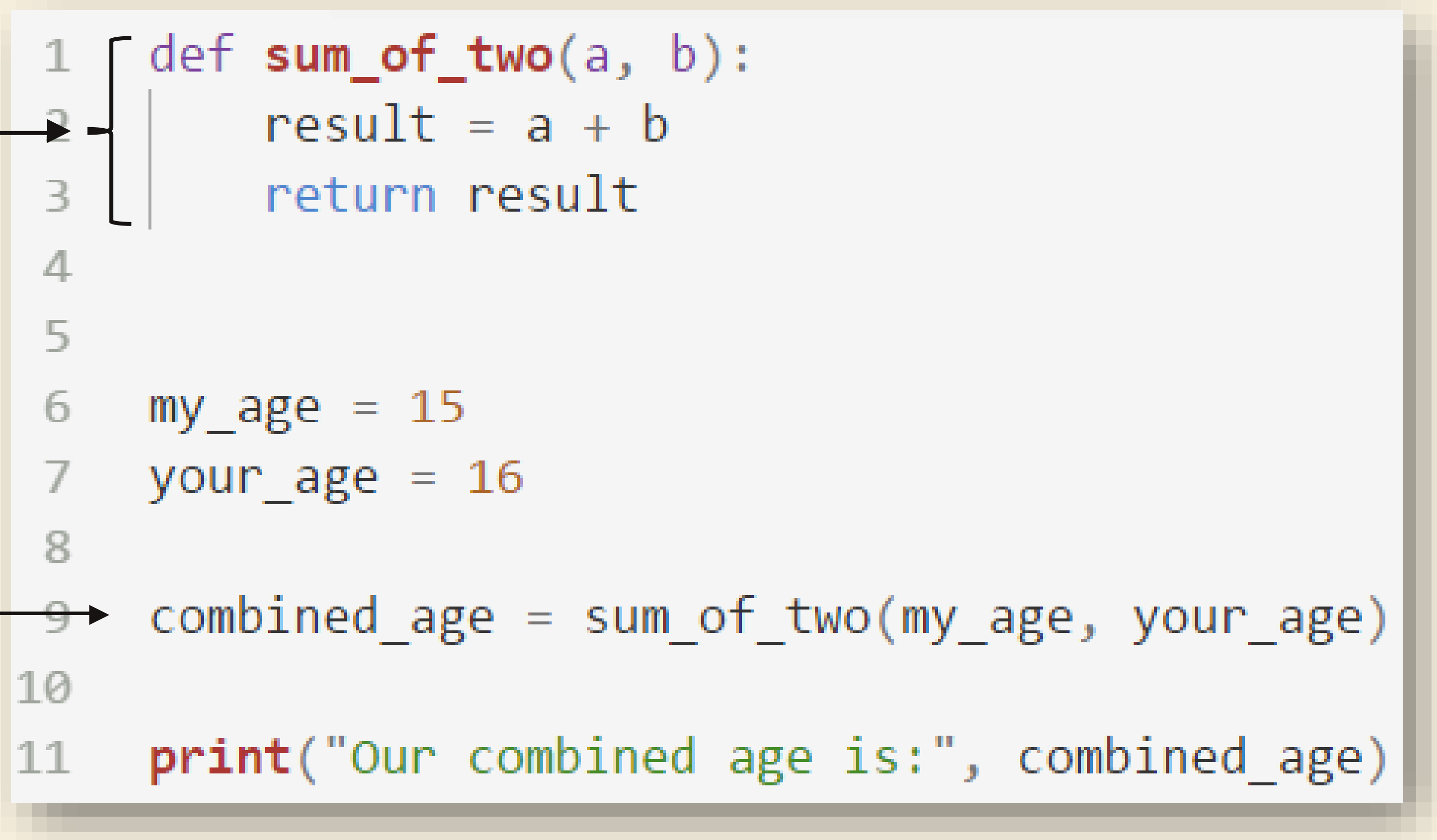
- The **return** statement indicates the value that is returned by the function.
- Do note that the **return** statement is optional.
  - A function can **return** nothing.
- If there is no **return** statement the function implicitly returns the value **None.**
  - You will learn more about **None** later.
- Also, functions that have no **return** statement are often called a procedure.
- Procedures are used to perform some task such as printing output, storing a file, etc.

FUNCTIONS

Here is the declaration and implementation of the function sum_of_two

Here the function is invoked/called. We send in the values of **my_age** as the first arguement and **your_age** as the second arguement.

The execution turns to the **function** on **line 1**. The value of the parameter **a** is now 15 and the value of the parameter **b** is 16. The variable result gets the value 31 and we **return that value**. The execution goes back to line 9 and the variable **combined_age** gets the value 31 that was returned from the function!

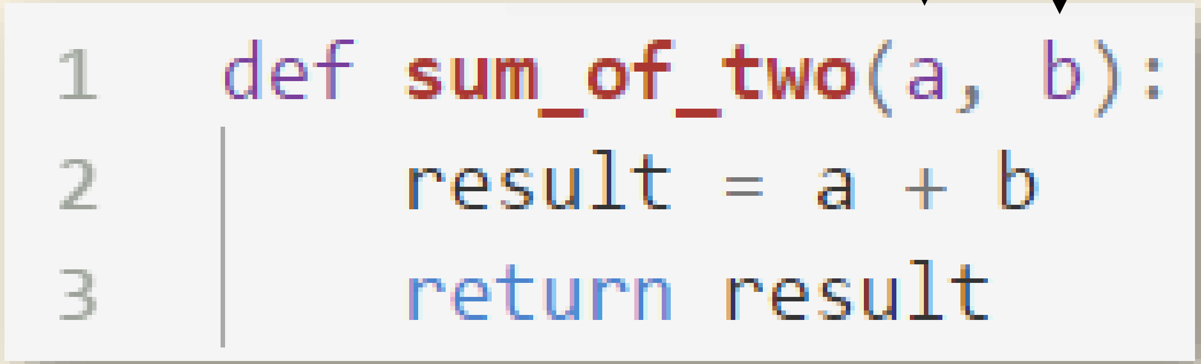


```
1   def sum_of_two(a, b):
2       result = a + b
3       return result
4
5
6   my_age = 15
7   your_age = 16
8
9   combined_age = sum_of_two(my_age, your_age)
10
11  print("Our combined age is:", combined_age)
```

FUNCTIONS

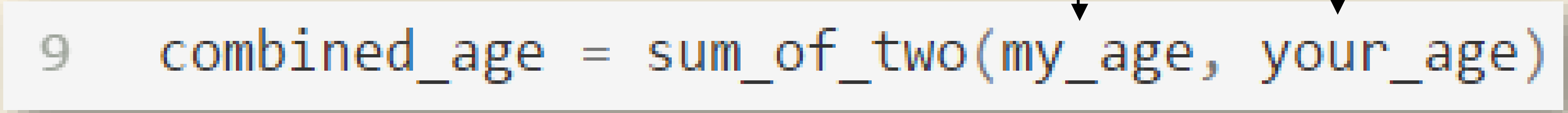
- Difference between parameters and arguments

These are the parameters of the function sum_of_two



```python
def sum_of_two(a, b):
    result = a + b
    return result
```

Arguments are the values that are passed to the function



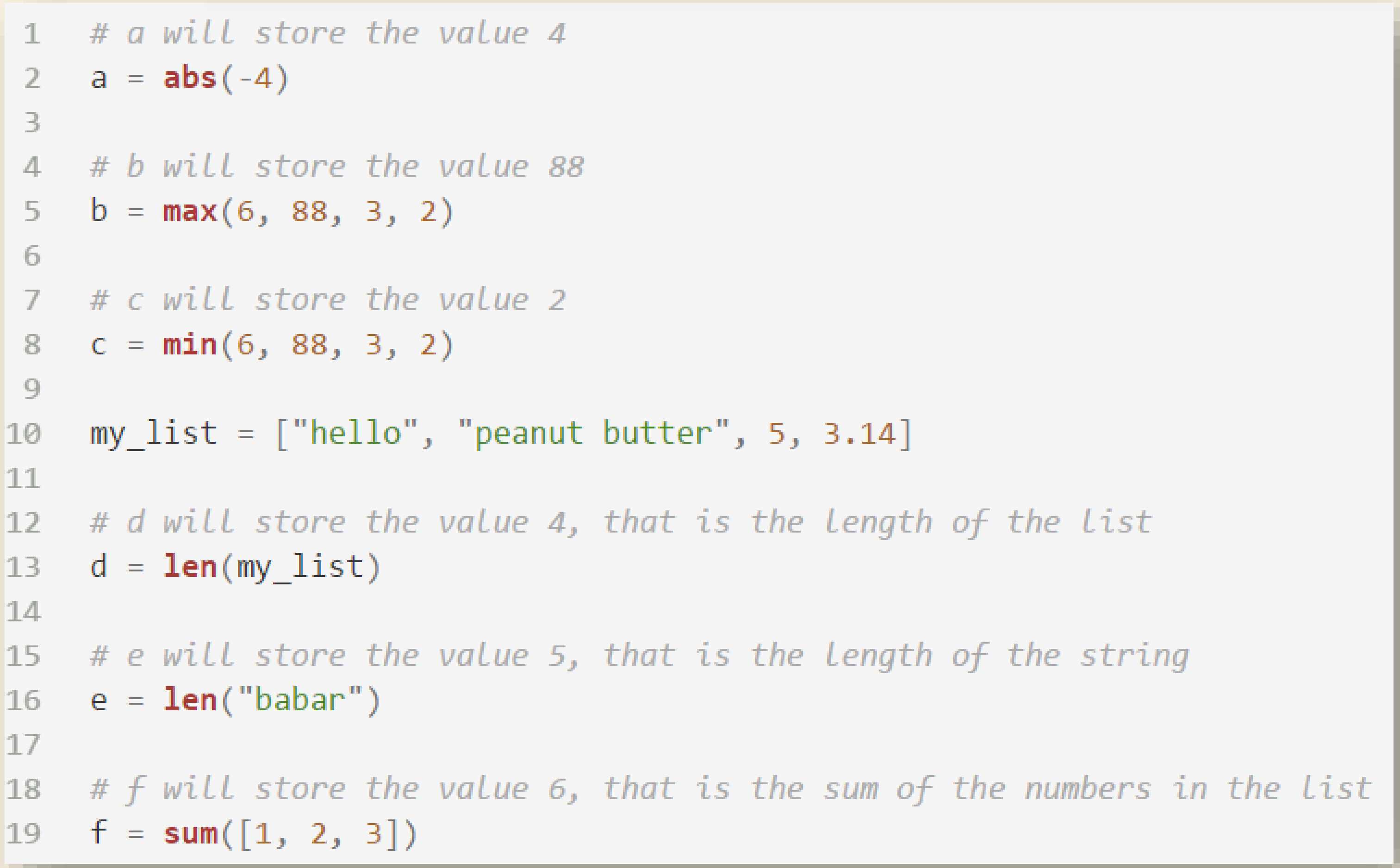
```python
combined_age = sum_of_two(my_age, your_age)
```

BUILT IN FUNCTIONS

- Python has many built in functions
  - You have already seen and used some of them such as int(), float(), input(), print()
- More commonly function are:
  - abs() -> returns the absolute value of a number
  - max() -> returns the highest value of the values that are passed to the function
  - min() -> returns the lowest value of the values that are passed to the function
  - len() -> returns the length of the object that is passed to the function
  - sum() -> returns the sum of items in the iterable(e.g list) that is passed to the function

FUNCTIONS

- Here you can see how some of these built in functions can be used



```python
# a will store the value 4
a = abs(-4)

# b will store the value 88
b = max(6, 88, 3, 2)

# c will store the value 2
c = min(6, 88, 3, 2)

my_list = ["hello", "peanut butter", 5, 3.14]

# d will store the value 4, that is the length of the list
d = len(my_list)

# e will store the value 5, that is the length of the string
e = len("babar")

# f will store the value 6, that is the sum of the numbers in the list
f = sum([1, 2, 3])
```

FUNCTIONS

- How to write a function
  - A function should do **one** thing. If it does too many things, it should be broken down into multiple functions
  - A function should be readable.
  - A function should be reusable.
    - If it does one thing well, then when a similar situation (maybe in another program) occurs, you can use it there as well.