MORE ON FUNCTIONS

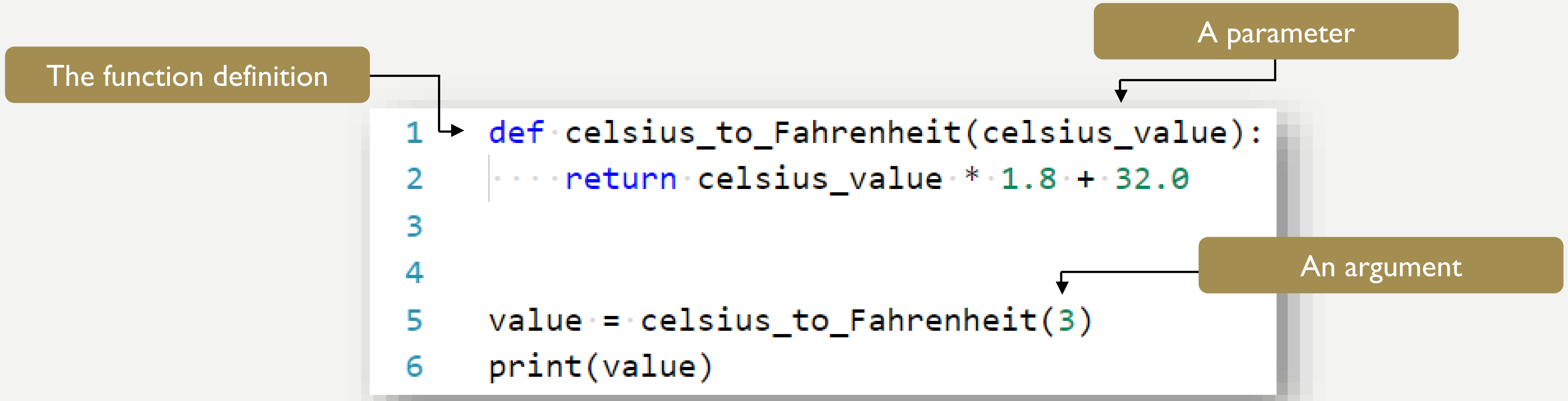
Slides by Björgvin B. Björgvinsson

FUNCTIONS
WHY HAVE THEM?

- They support the divide-and-conquer strategy
- They abstract the implementation of an operation
- They encourage reuse. Once it is written, it can be used again
- They encourage sharing. If well tested, other programmers can use them
- They give security. If well tested, then they are secure for reuse
- They simplify code and make it more readable

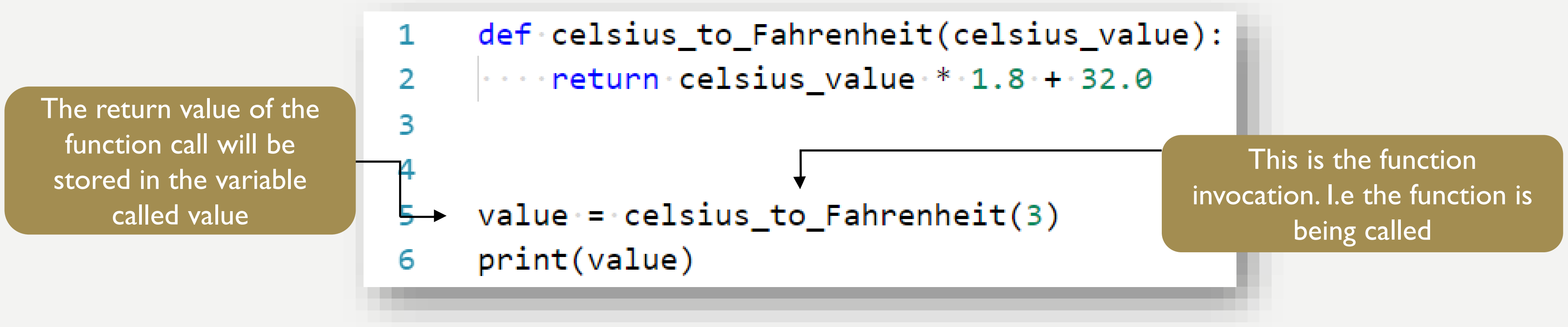
FUNCTIONS

- Consider a function which converts temperatures in Celsius to temperatures in Fahrenheit



```python
def celsius_to_Fahrenheit(celsius_value):
    return celsius_value * 1.8 + 32.0


value = celsius_to_Fahrenheit(3)
print(value)
```

FUNCTIONS



```python
def celsius_to_Fahrenheit(celsius_value):
    return celsius_value * 1.8 + 32.0


value = celsius_to_Fahrenheit(3)
print(value)
```

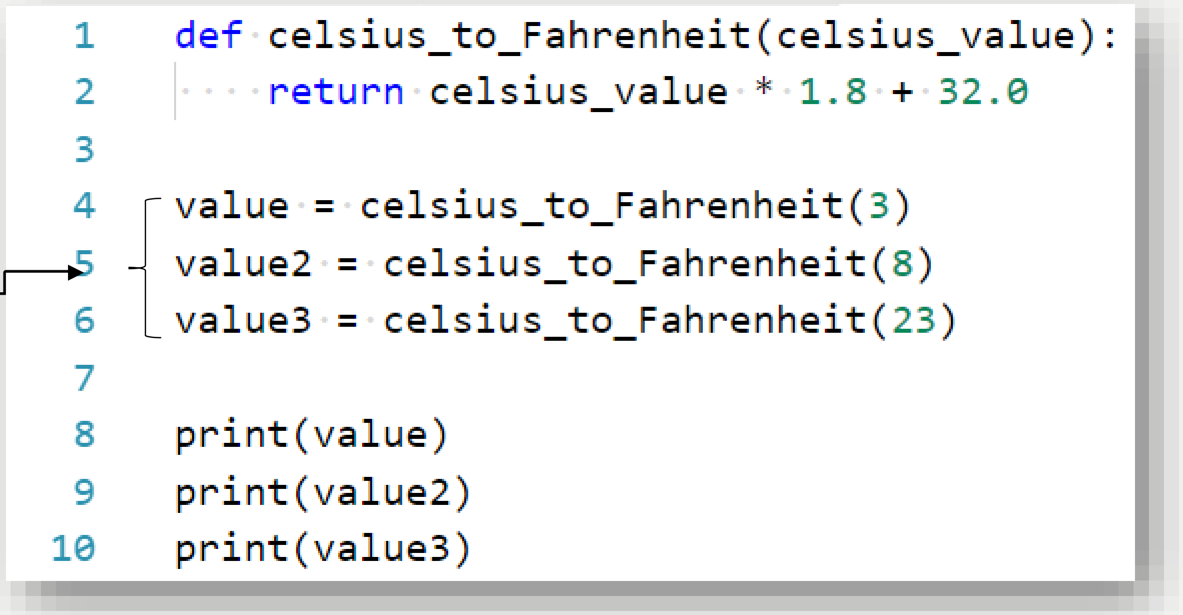
The return value of the function call will be stored in the variable called value

This is the function invocation. I.e the function is being called

FUNCTIONS

- Function are defined once!
  - But they can be invoked multiple times

Here the function **celsius_to_Fahrenheit** is invoked(called) three times. Each time a different argument is passed to the function. Three variables are created to store the return values from each function call.



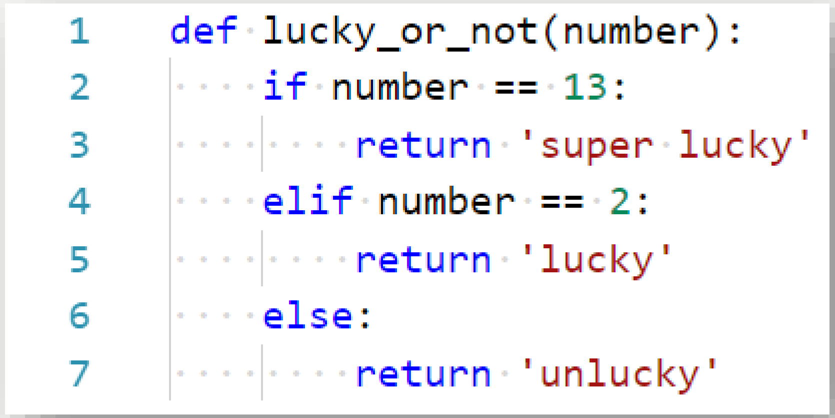
```python
def celsius_to_Fahrenheit(celsius_value):
    return celsius_value * 1.8 + 32.0

value = celsius_to_Fahrenheit(3)
value2 = celsius_to_Fahrenheit(8)
value3 = celsius_to_Fahrenheit(23)

print(value)
print(value2)
print(value3)
```

FUNCTIONS
THE RETURN STATEMENT

- The `return` statement indicates the value that is returned by the function
- The parameters indicate what data goes into the function and the return statement indicates the data that goes out
- The `return` statement is optional (the function can return nothing). If there is no `return`, the function is often called a procedure.
  - Do note that if there is no **explicit** return statement in a function the function will **implicitly** return the value **None**

MULTIPLE RETURNS IN A FUNCTION

- A function can have multiple `return` statements.
- The first `return` statement executed ends the function.
- Multiple returns can be confusing to the reader and should be used judiciously.



```python
def lucky_or_not(number):
    if number == 13:
        return 'super lucky'
    elif number == 2:
        return 'lucky'
    else:
        return 'unlucky'
```

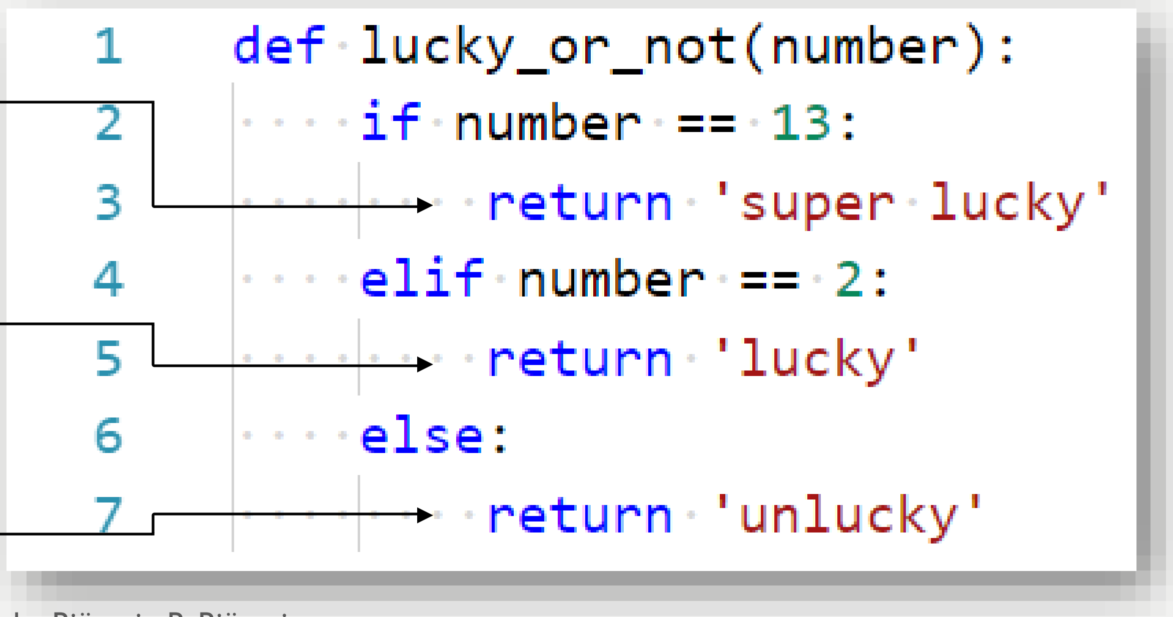
MULTIPLE RETURNS IN A FUNCTION

In this function the return value depends on the value passed to the function.

If the number 13 is passed to the function the string super lucky is returned and the rest of the function is not executed!

If the number 2 is passed to the function the string lucky is returned and the rest of the function is not executed!

If any number that is not 2 or 13 is passed to the function the string unlucky is returned.



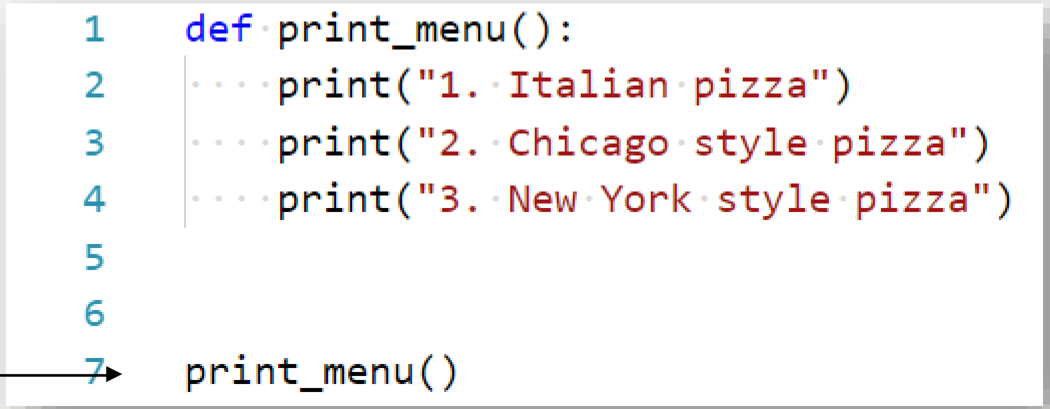
```python
def lucky_or_not(number):
    if number == 13:
        return 'super lucky'
    elif number == 2:
        return 'lucky'
    else:
        return 'unlucky'
```

Slides by Björgvin B. Björgvinsson

FUNCTIONS
THE RETURN STATEMENT

- Here is an example of a function that has no return statement
  - It's only purpose is to print a menu

Notice that there is no variable defined to store the return value. That is because the actual return value is None and we are not interested in using that!

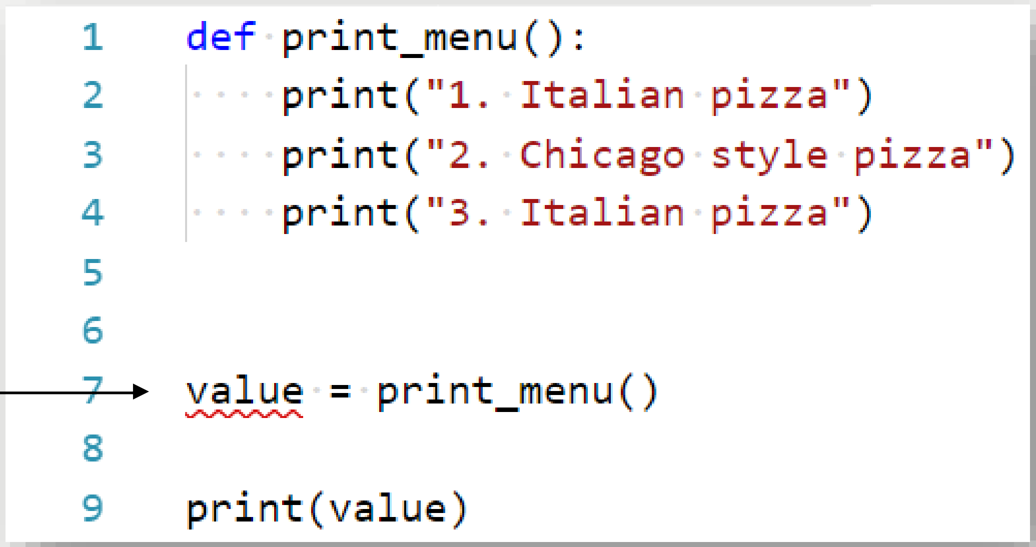


```python
def print_menu():
    print("1. Italian pizza")
    print("2. Chicago style pizza")
    print("3. New York style pizza")


print_menu()
```

FUNCTIONS
THE RETURN STATEMENT

- Do note that you can use the return value if you want to when using a function which has no explicit return statement but it is usually not done

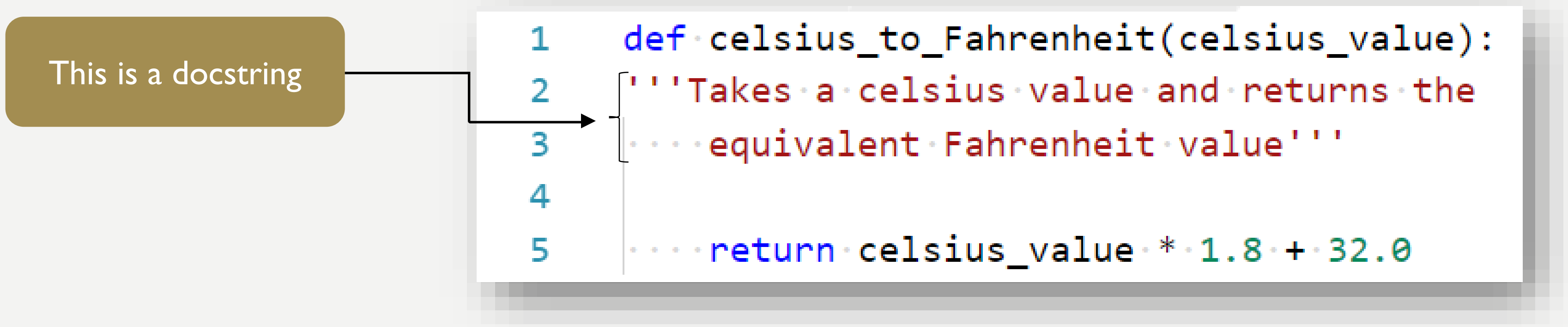
The variable value will store the value None and in line 9 the value None will be printed to the console.



```
1  def print_menu():
2      print("1. Italian pizza")
3      print("2. Chicago style pizza")
4      print("3. Italian pizza")
5
6
7  value = print_menu()
8
9  print(value)
```

FUNCTIONS
TRIPLE QUOTED STRING IN FUNCTION

- A triple quoted string just after the definition of a function is called a ***docstring***
- A docstring is a documentation of the function's purpose
  - It can be used by other tools to tell the user what the function is used for.



```python
def celsius_to_Fahrenheit(celsius_value):
'''Takes a celsius value and returns the
    equivalent Fahrenheit value'''

    return celsius_value * 1.8 + 32.0
```

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 (refactored)
- ***A function should be readable!***
  - You will read a lot more code than you will ever write ☺
- ***A function should be reusable***!
  - If it does one thing well, then when a similar situation occurs, use it there as well!
- ***A function should not be too long!***