# CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding

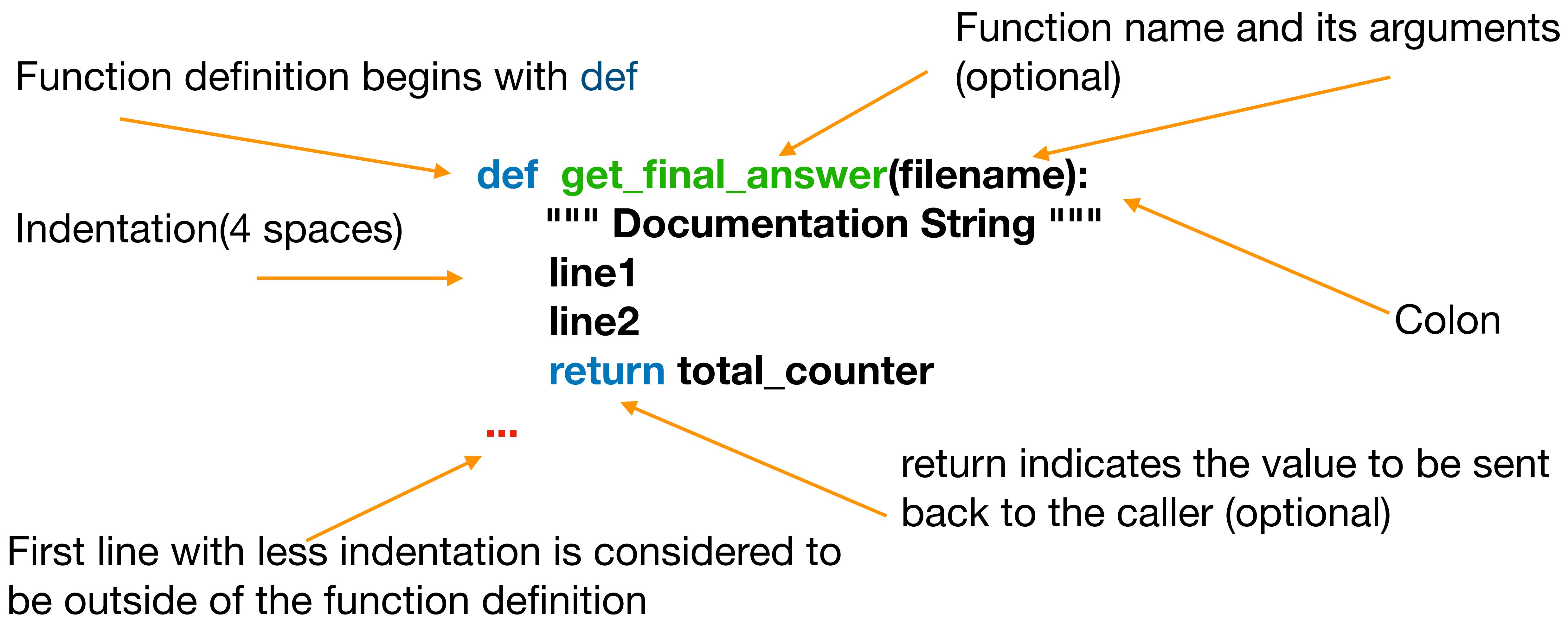
Sep 13 2019, Lecture 8

## Agenda

- Agenda:
    - Quick review of concepts from last lecture
    - Functions and modules

## Functions

- A function
  - A block of code which only runs when it is called
  - One way to organize and reuse code
  - You can pass information to a function
  - You can ask a function to return data

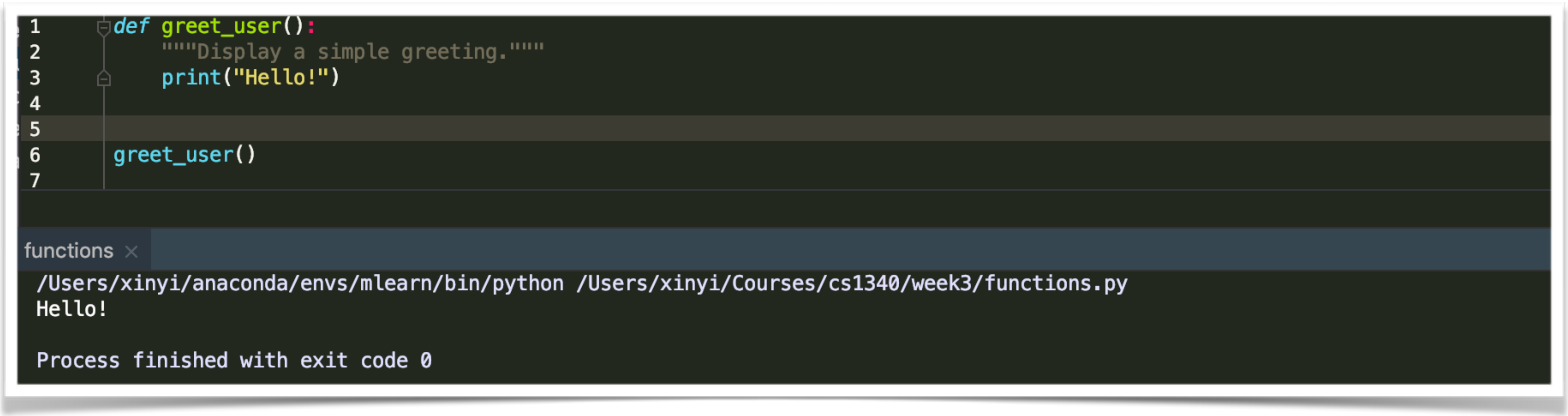


```
def  get_final_answer(filename):
     """ Documentation String """
     line1
     line2
     return total_counter
...
```

Function definition begins with def
Function name and its arguments (optional)
Indentation(4 spaces)
Colon
return indicates the value to be sent back to the caller (optional)
First line with less indentation is considered to be outside of the function definition

## Functions

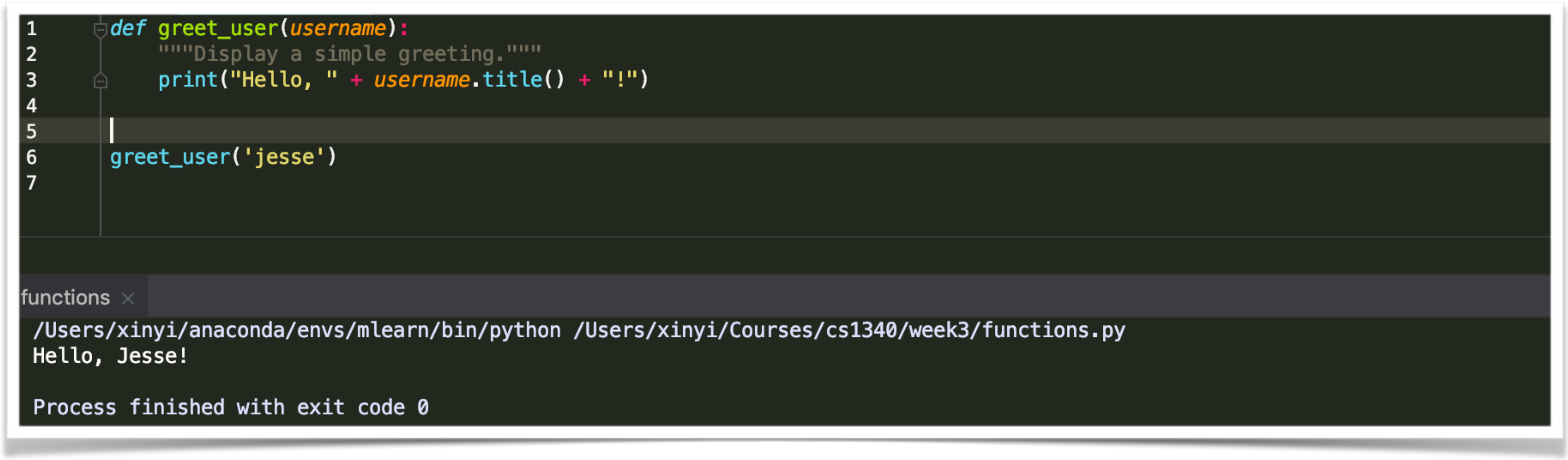
- An example



```
def greet_user():
    """Display a simple greeting."""
    print("Hello!")


greet_user()

```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello!

Process finished with exit code 0
```



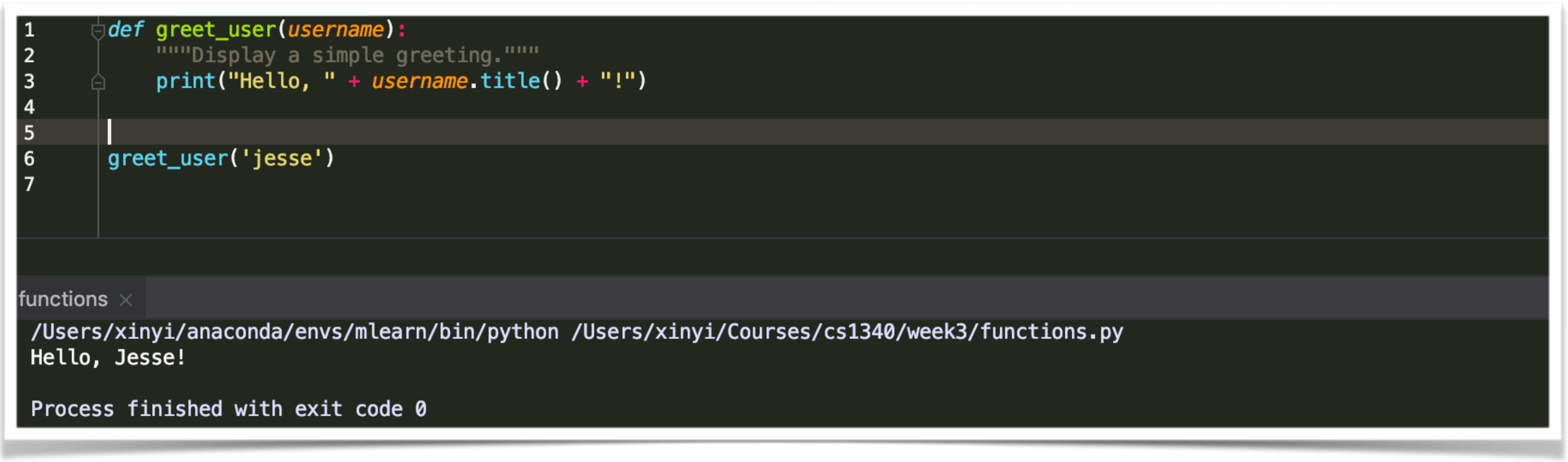
```
def greet_user(username):
    """Display a simple greeting."""
    print("Hello, " + username.title() + "!")


greet_user('jesse')

```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello, Jesse!

Process finished with exit code 0
```

# Functions

- Arguments and Parameters
  - The variable username in the definition of greet_user() is an example of a *parameter*
  - The value "jesse" in greet_user("jesse") is an example of an *argument*



```python
def greet_user(username):
    """Display a simple greeting."""
    print("Hello, " + username.title() + "!")


greet_user('jesse')

```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello, Jesse!

Process finished with exit code 0
```

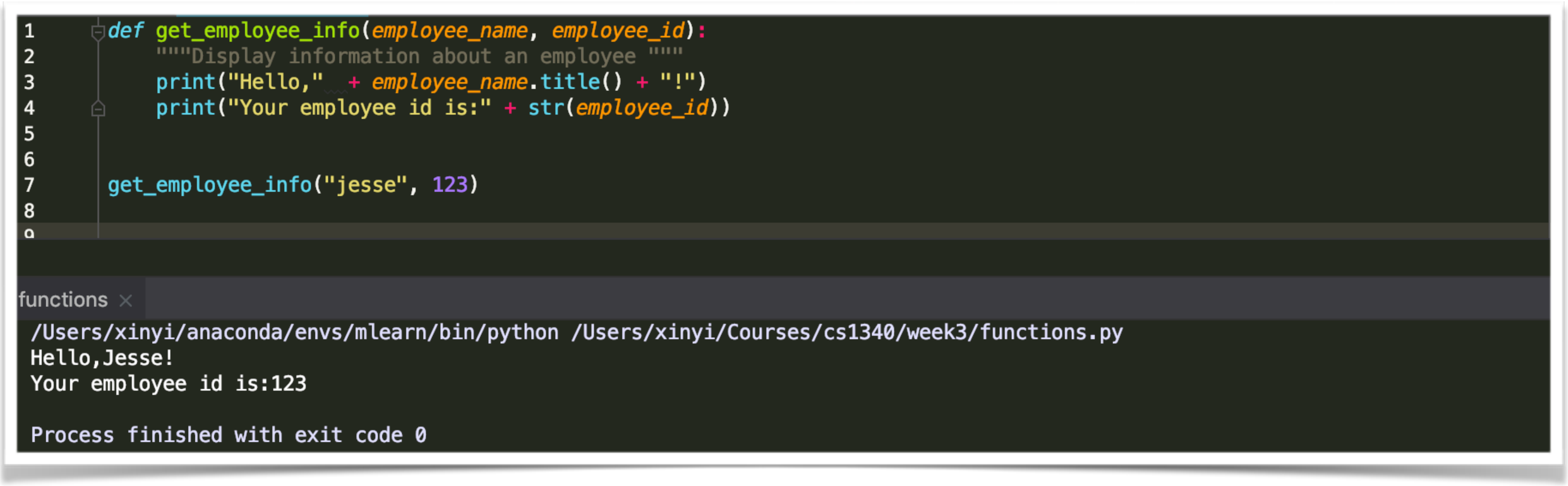
**Note: the fact is sometimes people speak of parameters and arguments interchangeably.**

## Functions

- Passing arguments
  - A function definition can have multiple parameters, a function call may need multiple arguments
  - Ways of passing arguments
    - positional arguments
      - need to be in the same order the parameters were written
    - keyword arguments
      - where each argument consists of a variable name and a value

## Functions

- Positional arguments
  - match each argument in the function call with a parameter in the function definition



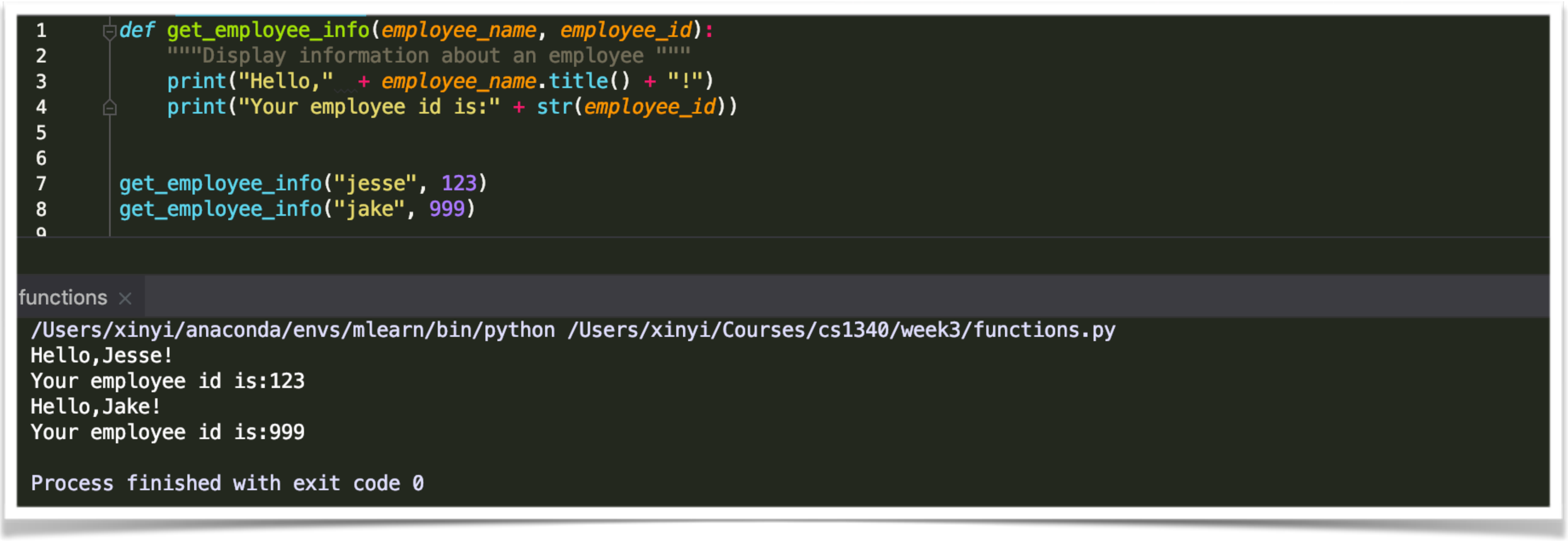
```
def get_employee_info(employee_name, employee_id):
    """Display information about an employee """
    print("Hello,"  + employee_name.title() + "!")
    print("Your employee id is:" + str(employee_id))


get_employee_info("jesse", 123)
```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello,Jesse!
Your employee id is:123

Process finished with exit code 0
```

# Functions

- Multiple function calls



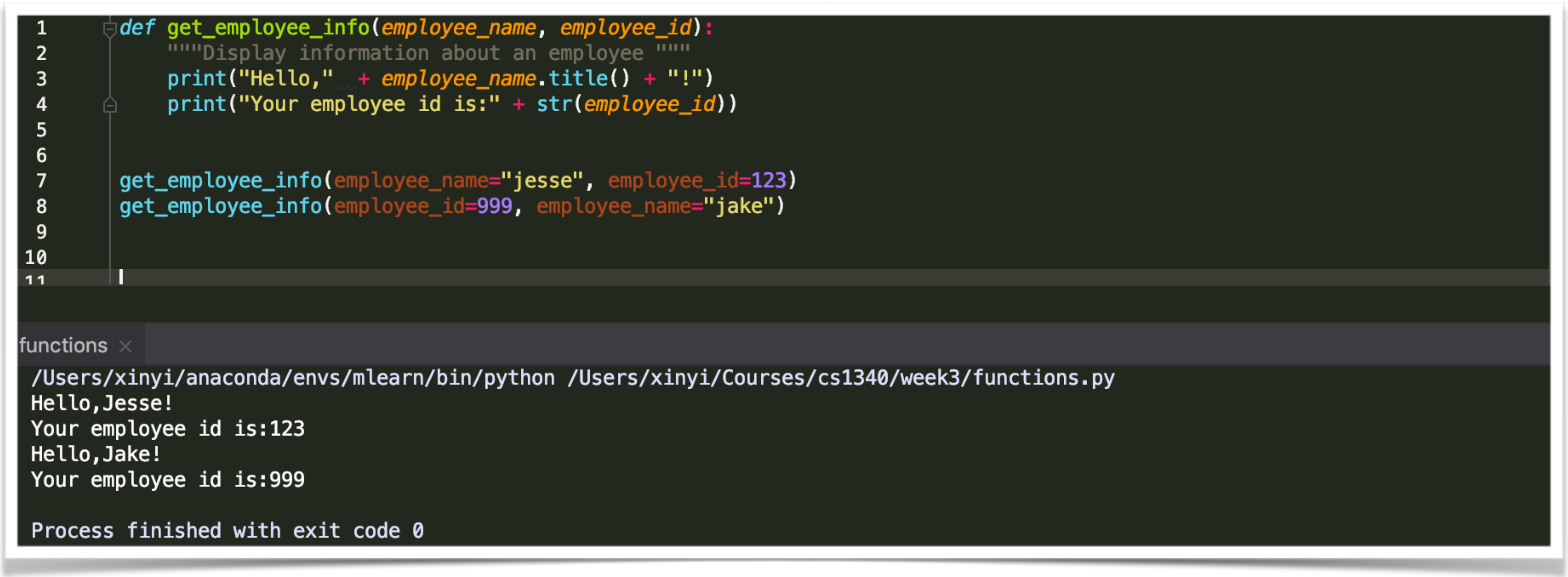
```python
def get_employee_info(employee_name, employee_id):
    """Display information about an employee """
    print("Hello,"  + employee_name.title() + "!")
    print("Your employee id is:" + str(employee_id))


get_employee_info("jesse", 123)
get_employee_info("jake", 999)
```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello,Jesse!
Your employee id is:123
Hello,Jake!
Your employee id is:999

Process finished with exit code 0
```

# Functions

- Keywords arguments
  - A name-value pair that you pass to a function



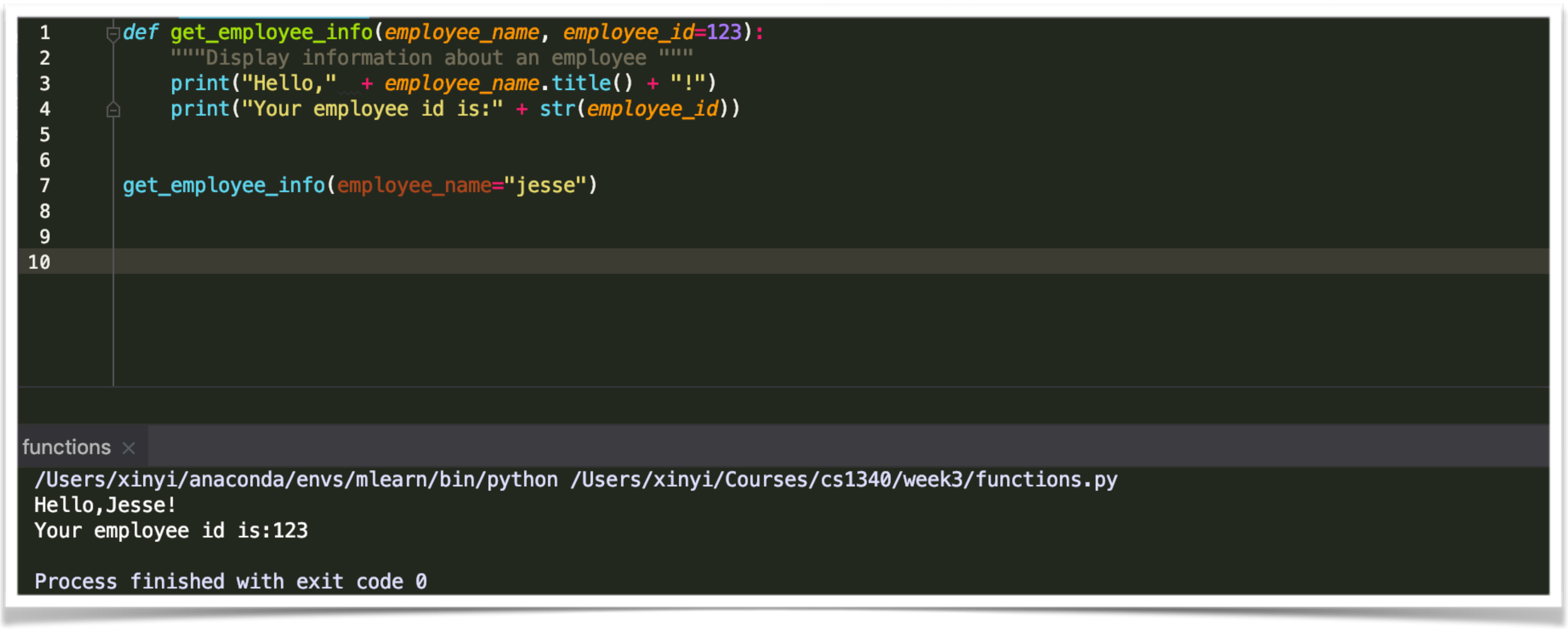
```python
def get_employee_info(employee_name, employee_id):
    """Display information about an employee """
    print("Hello,"  + employee_name.title() + "!")
    print("Your employee id is:" + str(employee_id))


get_employee_info(employee_name="jesse", employee_id=123)
get_employee_info(employee_id=999, employee_name="jake")
```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello,Jesse!
Your employee id is:123
Hello,Jake!
Your employee id is:999

Process finished with exit code 0
```

## Functions

- Default Values
  - You can define a default value for each parameter, if an argument for a parameter is provided in the function call, Python uses the argument value. If not, it



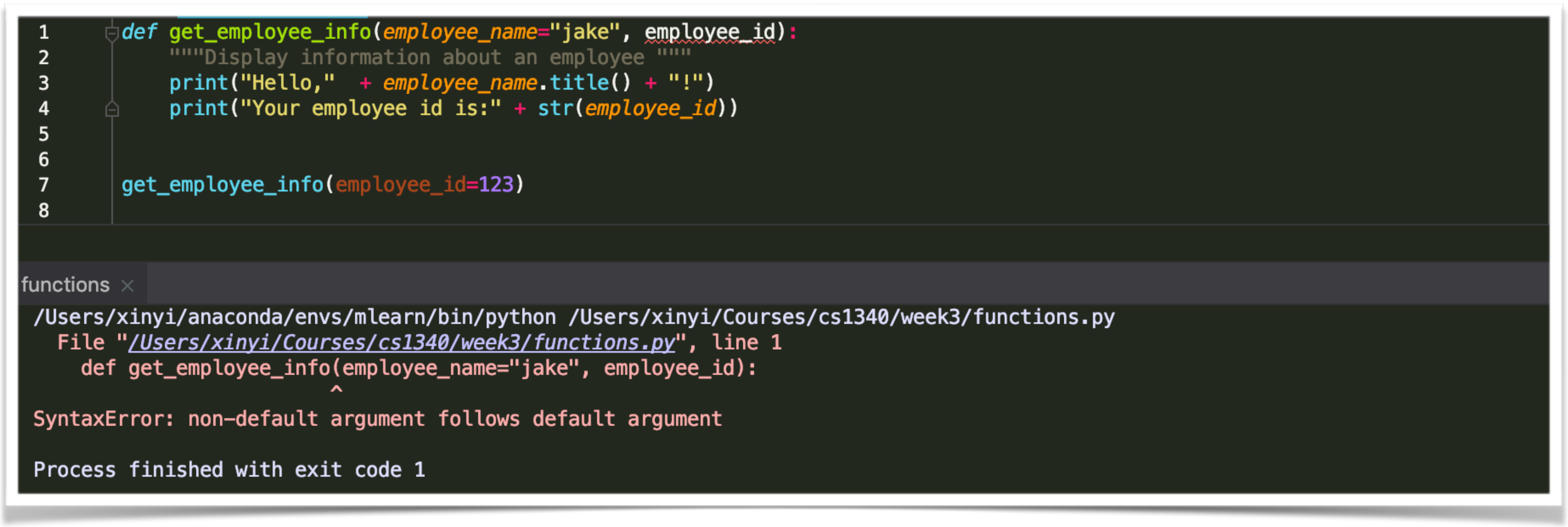
```python
def get_employee_info(employee_name, employee_id=123):
    """Display information about an employee """
    print("Hello," + employee_name.title() + "!")
    print("Your employee id is:" + str(employee_id))


get_employee_info(employee_name="jesse")
```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello,Jesse!
Your employee id is:123

Process finished with exit code 0
```

## Functions

- Default Values
  - any parameter with a default value needs to be listed after all the parameters that don't have default values

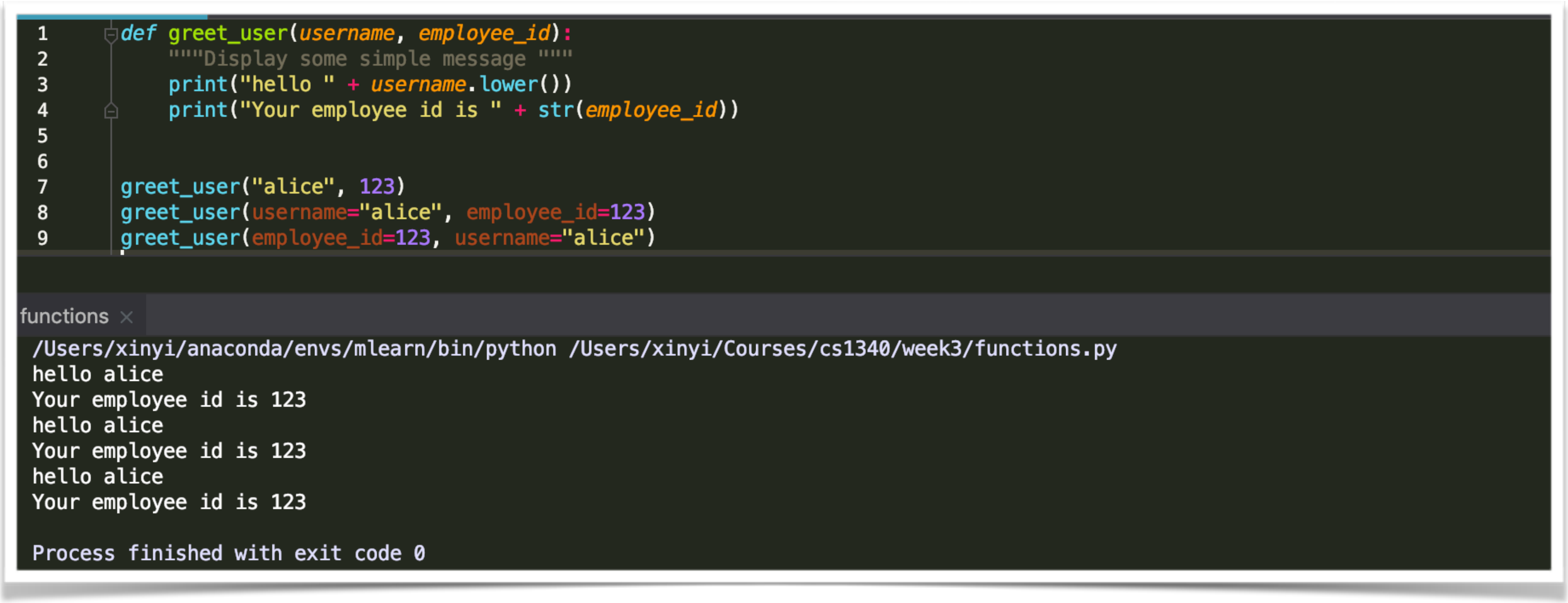


```
def get_employee_info(employee_name="jake", employee_id):
    """Display information about an employee """
    print("Hello,"  + employee_name.title() + "!")
    print("Your employee id is:" + str(employee_id))


get_employee_info(employee_id=123)

```

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
  File "/Users/xinyi/Courses/cs1340/week3/functions.py", line 1
    def get_employee_info(employee_name="jake", employee_id):
                                         ^
SyntaxError: non-default argument follows default argument

Process finished with exit code 1
```

# Functions

- Equivalent Function Calls



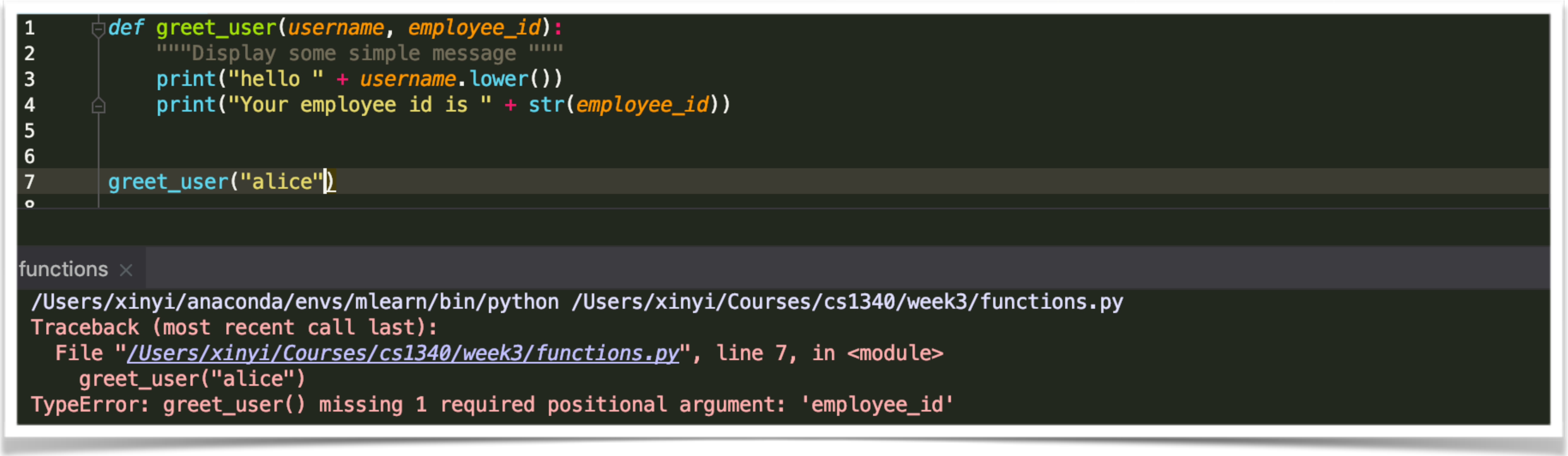
```python
def greet_user(username, employee_id):
    """Display some simple message """
    print("hello " + username.lower())
    print("Your employee id is " + str(employee_id))


greet_user("alice", 123)
greet_user(username="alice", employee_id=123)
greet_user(employee_id=123, username="alice")
```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
hello alice
Your employee id is 123
hello alice
Your employee id is 123
hello alice
Your employee id is 123

Process finished with exit code 0
```

# Functions

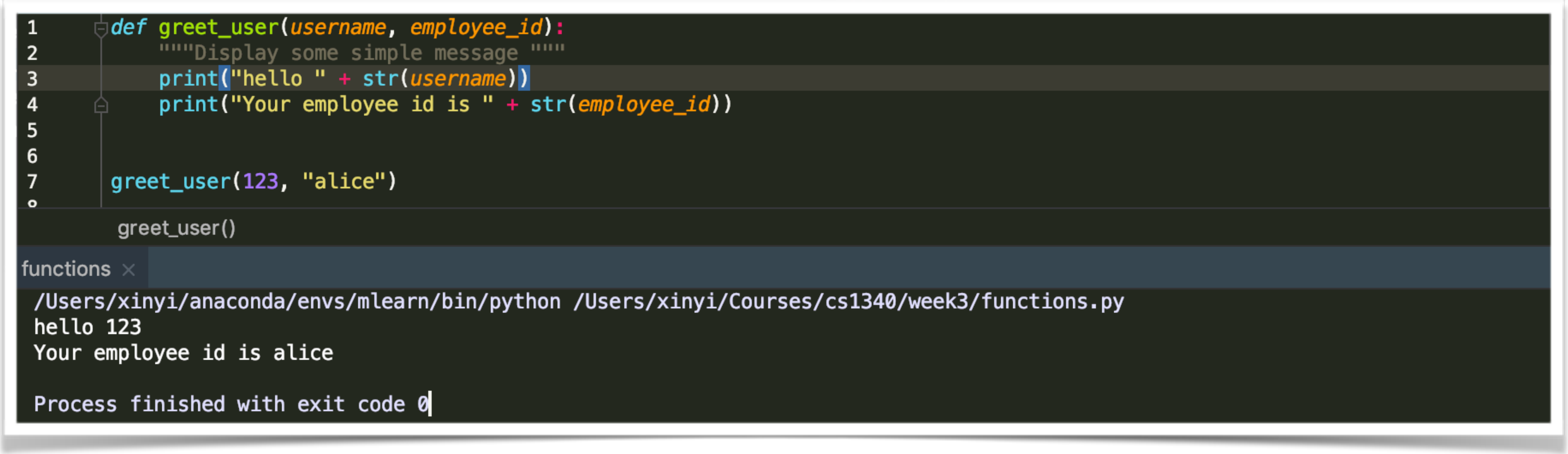
- Avoiding argument errors



```
def greet_user(username, employee_id):
    """Display some simple message """
    print("hello " + username.lower())
    print("Your employee id is " + str(employee_id))


greet_user("alice")
```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Traceback (most recent call last):
  File "/Users/xinyi/Courses/cs1340/week3/functions.py", line 7, in <module>
    greet_user("alice")
TypeError: greet_user() missing 1 required positional argument: 'employee_id'
```



```
def greet_user(username, employee_id):
    """Display some simple message """
    print("hello " + str(username))
    print("Your employee id is " + str(employee_id))


greet_user(123, "alice")
```

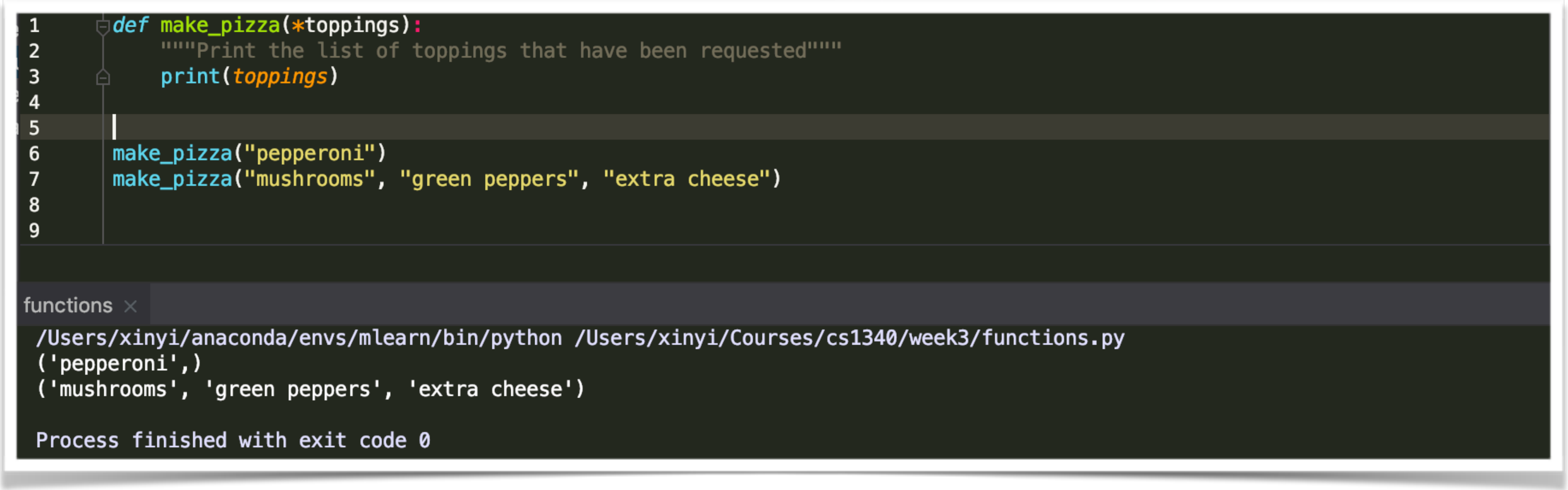
greet_user()

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
hello 123
Your employee id is alice

Process finished with exit code 0
```

## Functions

- Passing an arbitrary number of arguments
  - when you don't ahead of time how many arguments a function needs to accept.

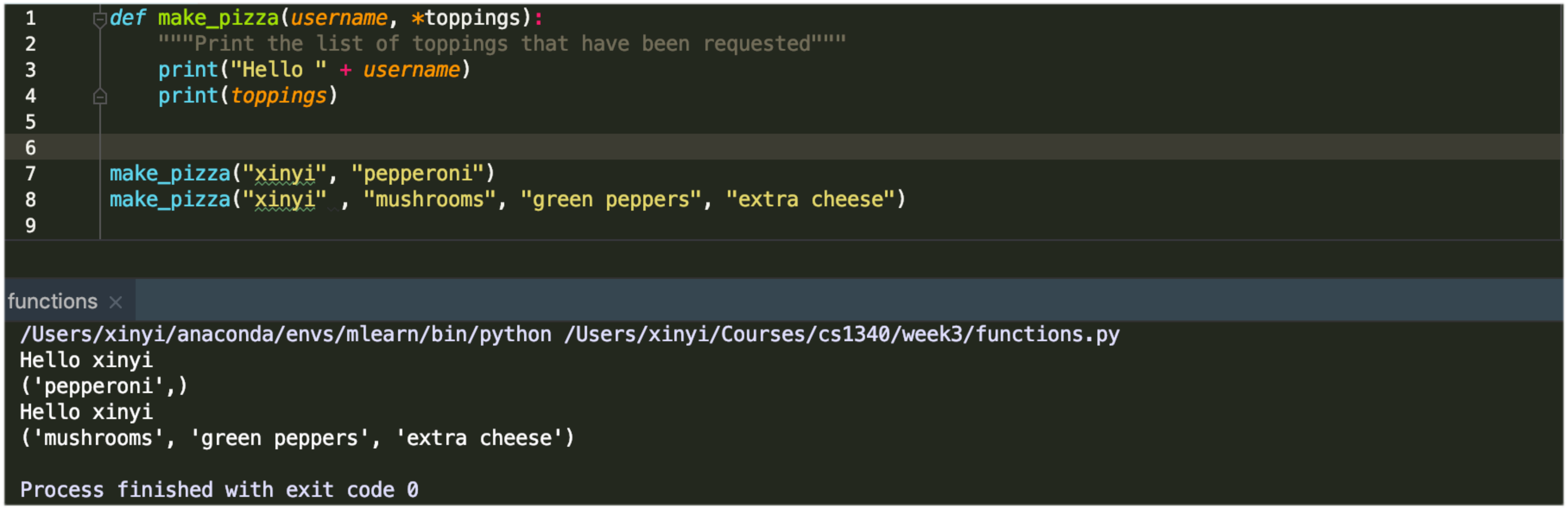


```
def make_pizza(*toppings):
    """Print the list of toppings that have been requested"""
    print(toppings)


make_pizza("pepperoni")
make_pizza("mushrooms", "green peppers", "extra cheese")
```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
('pepperoni',)
('mushrooms', 'green peppers', 'extra cheese')

Process finished with exit code 0
```

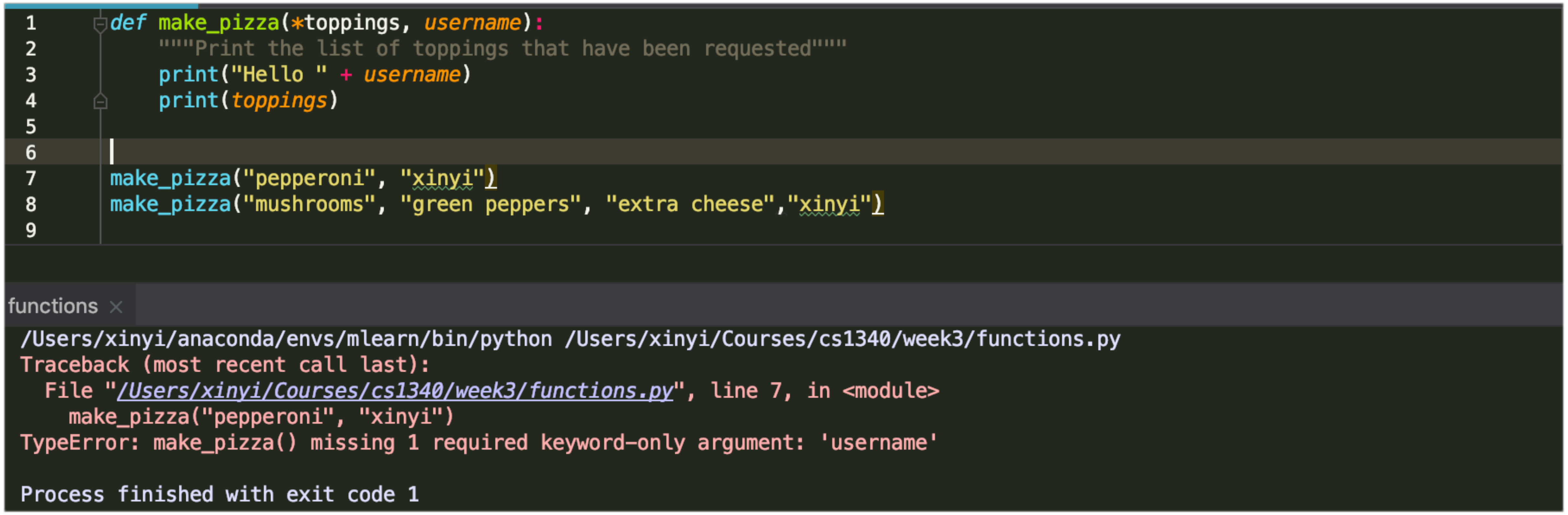
# Functions



```python
def make_pizza(username, *toppings):
    """Print the list of toppings that have been requested"""
    print("Hello " + username)
    print(toppings)


make_pizza("xinyi", "pepperoni")
make_pizza("xinyi" , "mushrooms", "green peppers", "extra cheese")

```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Hello xinyi
('pepperoni',)
Hello xinyi
('mushrooms', 'green peppers', 'extra cheese')

Process finished with exit code 0
```



```python
def make_pizza(*toppings, username):
    """Print the list of toppings that have been requested"""
    print("Hello " + username)
    print(toppings)


make_pizza("pepperoni", "xinyi")
make_pizza("mushrooms", "green peppers", "extra cheese","xinyi")

```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
Traceback (most recent call last):
  File "/Users/xinyi/Courses/cs1340/week3/functions.py", line 7, in <module>
    make_pizza("pepperoni", "xinyi")
TypeError: make_pizza() missing 1 required keyword-only argument: 'username'

Process finished with exit code 1
```

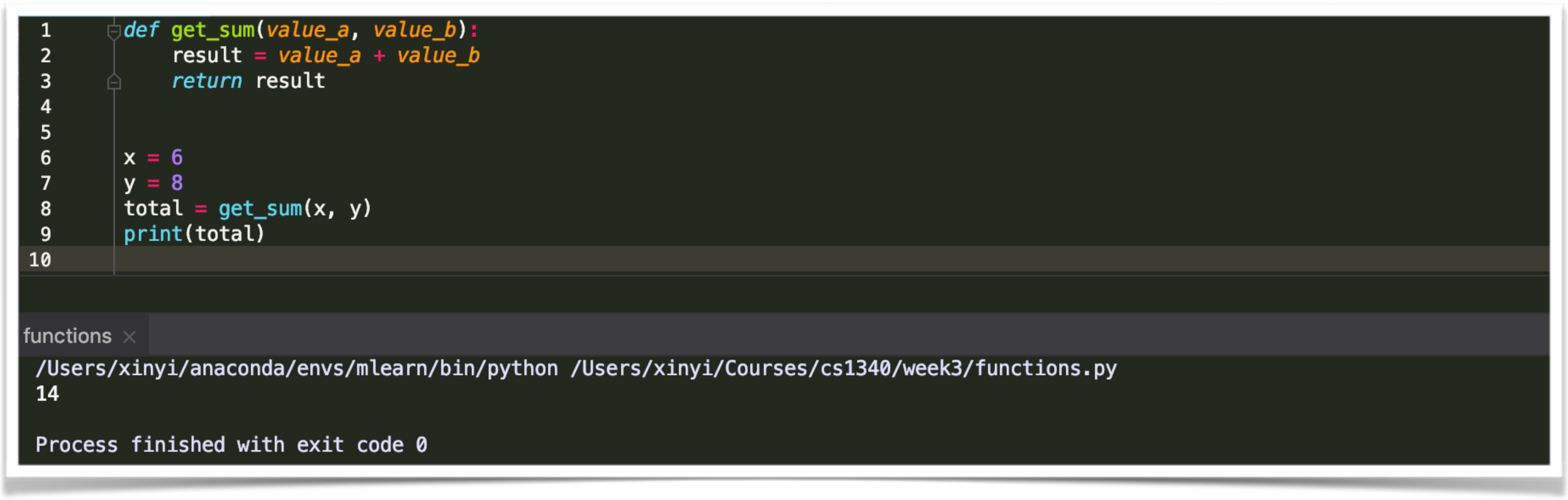
## Demo

## Functions

- Return values
  - A function doesn't always have to display its output directly. Instead, it can process some data and then return a value or set of values
  - The return statement takes a value from inside a function and sends it back to the line that called the function

## Functions

- Return a simple value, example 1



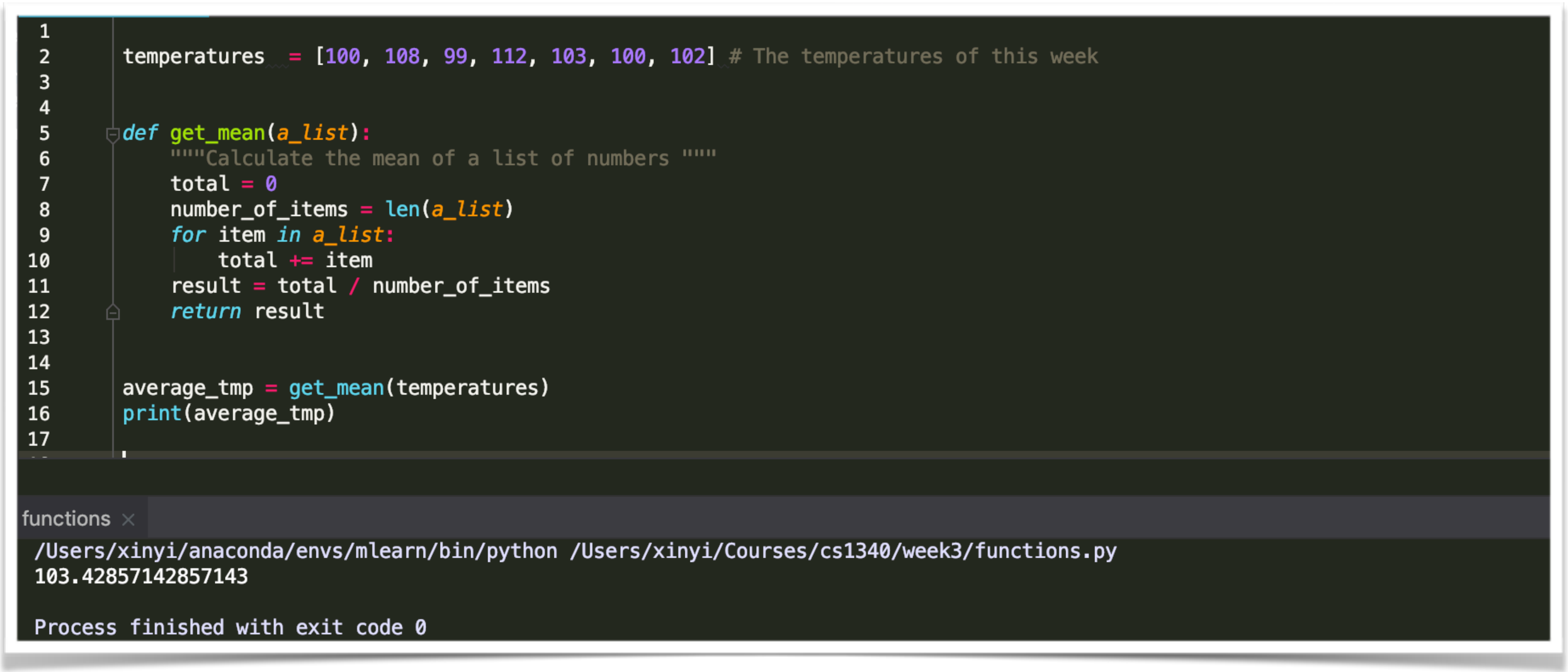
```
def get_sum(value_a, value_b):
    result = value_a + value_b
    return result


x = 6
y = 8
total = get_sum(x, y)
print(total)

```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
14

Process finished with exit code 0
```

## Functions

- Return a simple value, example 2

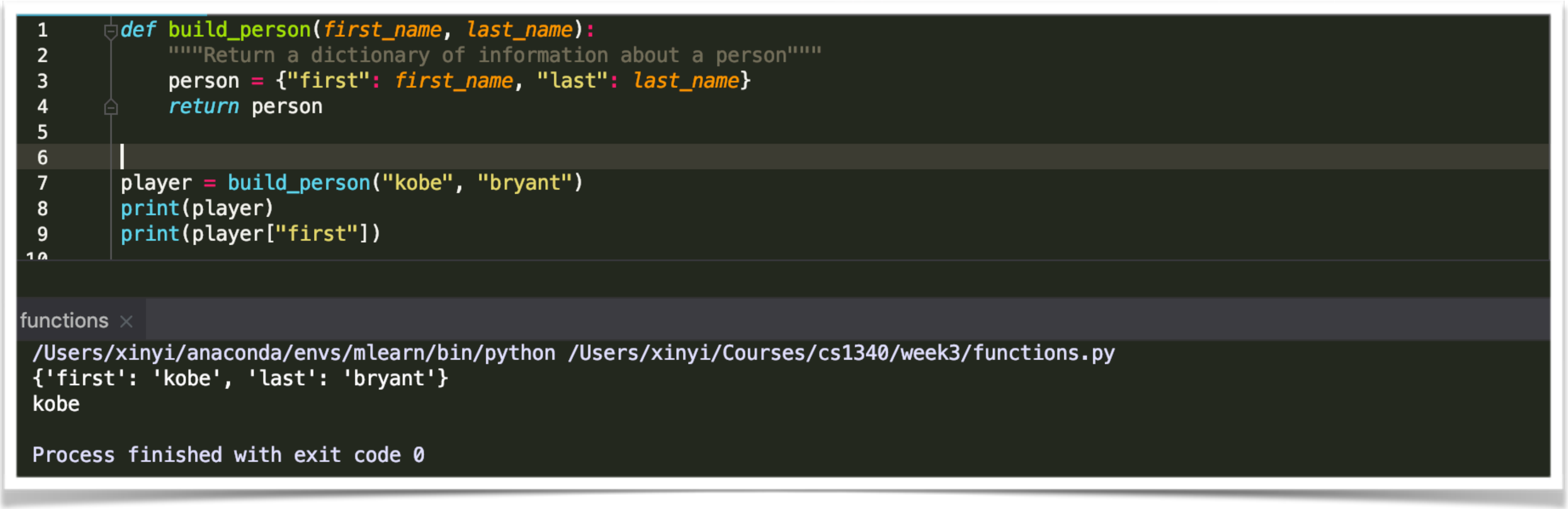


```

temperatures  = [100, 108, 99, 112, 103, 100, 102] # The temperatures of this week


def get_mean(a_list):
    """Calculate the mean of a list of numbers """
    total = 0
    number_of_items = len(a_list)
    for item in a_list:
        total += item
    result = total / number_of_items
    return result


average_tmp = get_mean(temperatures)
print(average_tmp)

```

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
103.42857142857143

Process finished with exit code 0
```

## Functions

- Return a dictionary



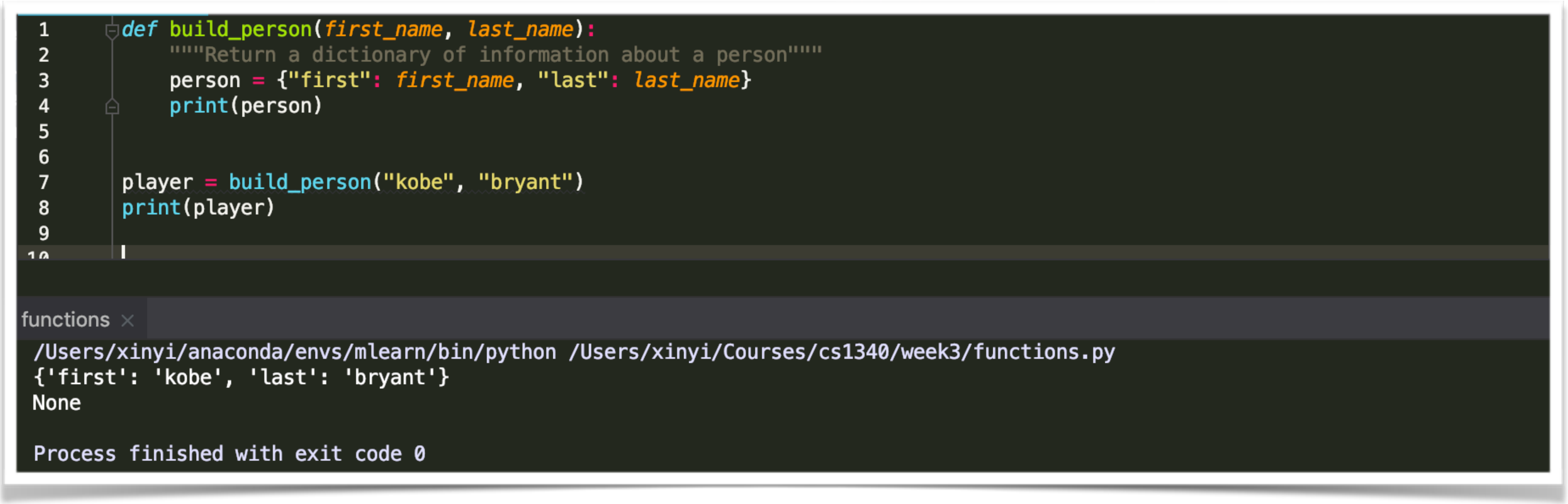
```python
def build_person(first_name, last_name):
    """Return a dictionary of information about a person"""
    person = {"first": first_name, "last": last_name}
    return person


player = build_person("kobe", "bryant")
print(player)
print(player["first"])
```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
{'first': 'kobe', 'last': 'bryant'}
kobe

Process finished with exit code 0
```

## Functions

- All functions in Python have a return value
  - even if no return line inside the code
- Functions without a return return the special value None
  - None is a special constant in the language.
  - None is also logically equivalent to False



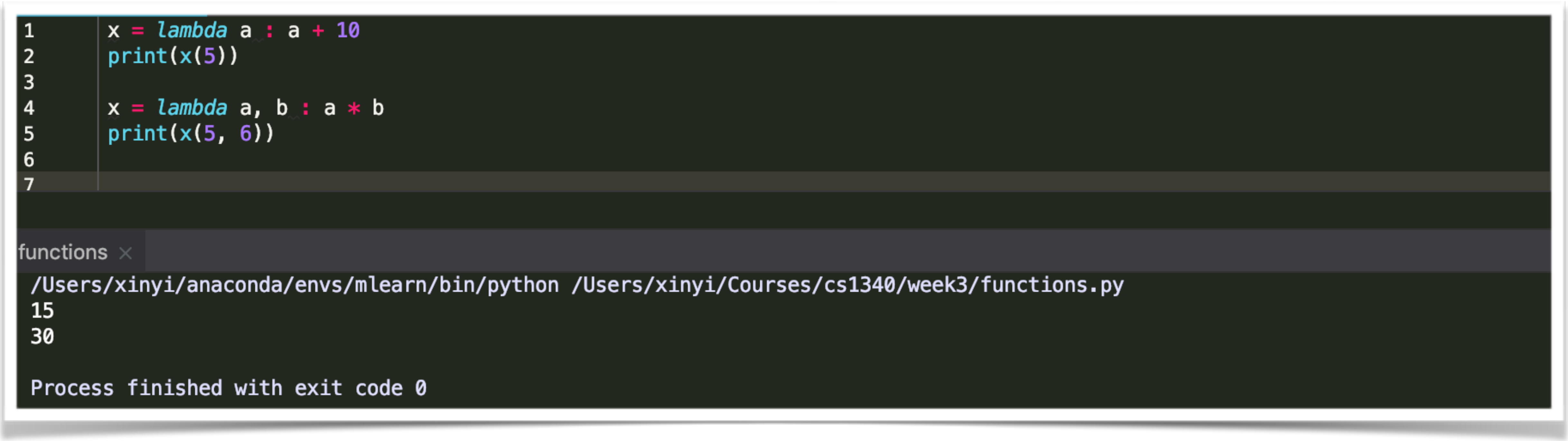
```python
def build_person(first_name, last_name):
    """Return a dictionary of information about a person"""
    person = {"first": first_name, "last": last_name}
    print(person)


player = build_person("kobe", "bryant")
print(player)

```

functions ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
{'first': 'kobe', 'last': 'bryant'}
None

Process finished with exit code 0
```

## Lambda function

- A lambda function is a small anonymous function
- A lambda function can take any number of arguments, but can only have one expression.
- Syntax: lambda **arguments : expression**



```
x = lambda a : a + 10
print(x(5))

x = lambda a, b : a * b
print(x(5, 6))

```

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
15
30

Process finished with exit code 0
```

## Demo

## Next Week

- A case study
- Organize code using Modules
- Inputs/outputs, File manipulations