Chapter 3
Functions

A. Downey, *Think Python: How to Think Like a Computer Science,* 3rd ed., O'Reilly, 2024.

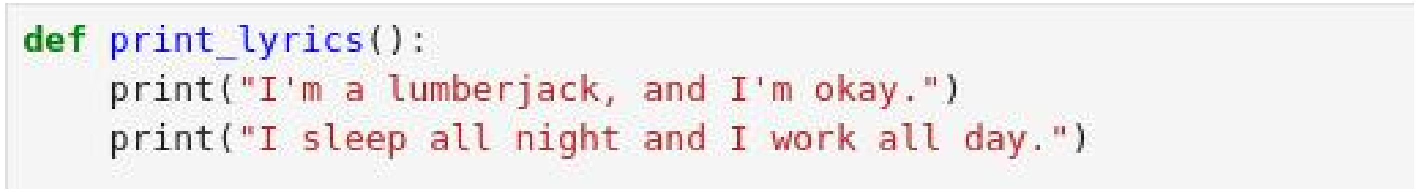
https://allendowney.github.io/ThinkPython/

Contents

- 1. Defining New Functions
- 2. Parameters
- 3. Calling Functions
- 4. Repetition
- 5. Variables and Parameters Are Local

1. Defining New Functions

- A **function definition** specifies the name of a new function and the sequence of statements that run when the function is called.

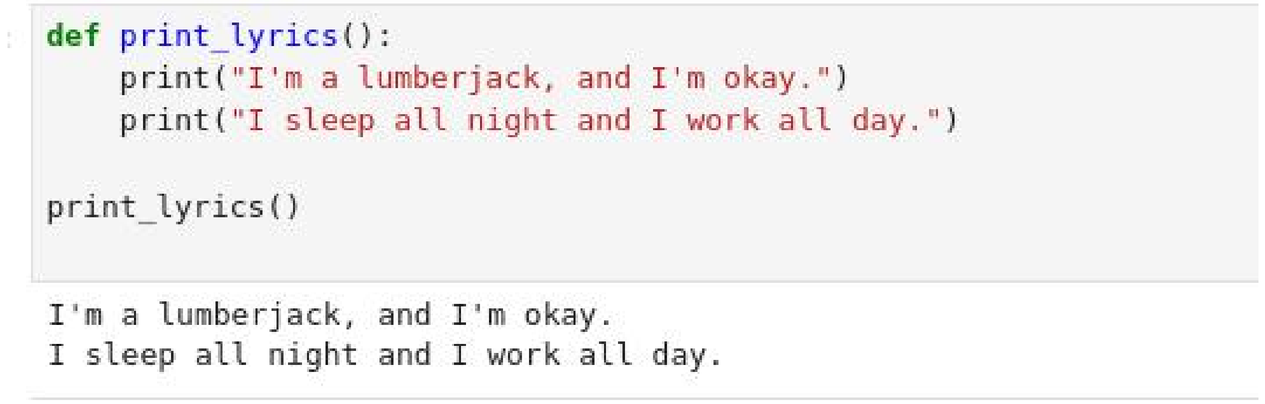


```python
def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print("I sleep all night and I work all day.")
```

- **def** is a keyword that indicates that this is a function definition. The name of the function is **print_lyrics**.
  - Anything that's a legal variable name is also a legal function name.
  - The empty parentheses after the name indicate that this function doesn't take any arguments.
  - The first line of the function definition is called the **header**—the rest is called the **body**.
  - **The header has to end with a colon and the body has to be indented.**
    - By convention, indentation is always four spaces.
  - The body of this function is two print statements; in general, the body of a function can contain any number of statements of any kind.

1. Defining New Functions

- We can call it the same way we call built-in functions. When the function runs, it executes the statements in the body.

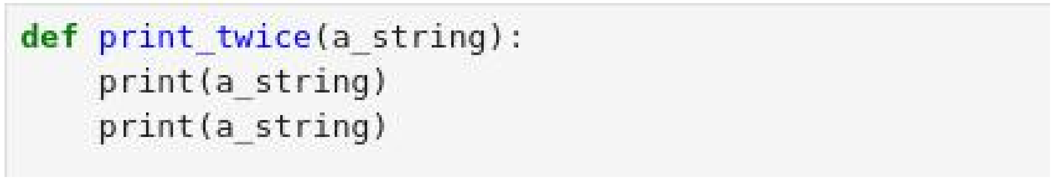


```
def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print("I sleep all night and I work all day.")

print_lyrics()
```

```
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
```

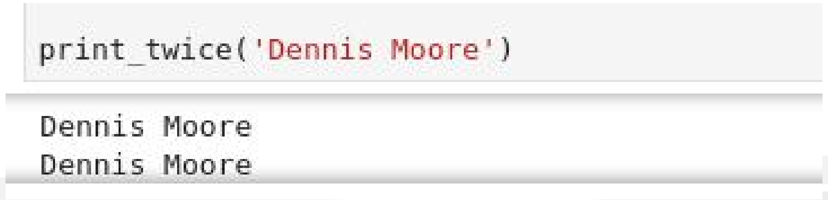
2. Parameters

- Some of the functions we have seen require arguments.
  - For example, when you call **abs** you pass a number as an argument.
- Some functions take more than one argument
  - For example, **math.pow** takes two, the base and the exponent.
- Here is a definition for a function that takes one argument:



```
def print_twice(a_string):
    print(a_string)
    print(a_string)
```

  - **a_string**: a **paremeter**
- When the function is called, the value of the **argument** is assigned to the parameter.



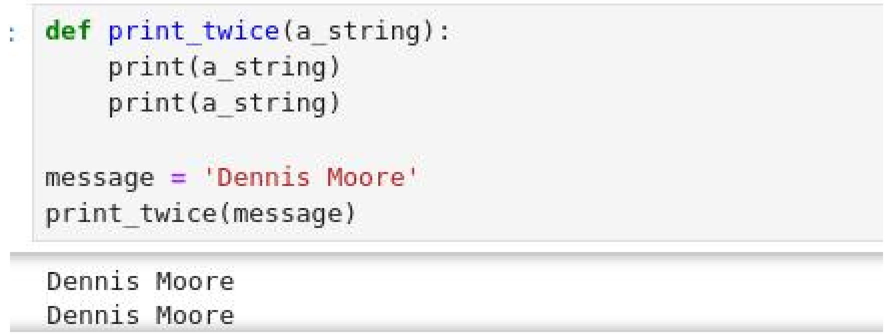
```
print_twice('Dennis Moore')
```

```
Dennis Moore
Dennis Moore
```

  - **'Dennis Moore'** : The argument which is passed to the parameter **a_string**
    - **'Dennis Moore'** gets assigned to the parameter **a_string**

2. Parameters

- You can also use a variable as an argument:



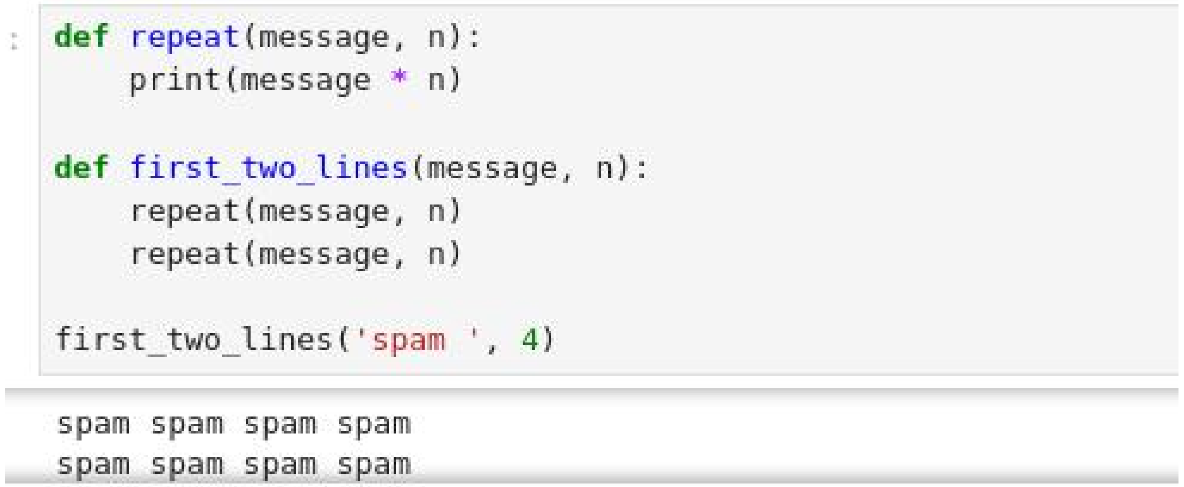
```
def print_twice(a_string):
    print(a_string)
    print(a_string)

message = 'Dennis Moore'
print_twice(message)
```

```
Dennis Moore
Dennis Moore
```

  – **message**: a variable **message** is the argument which is assigned to the parameter **a_string**.

3. Calling Functions

- Once you have defined a function, you can use it inside another function.

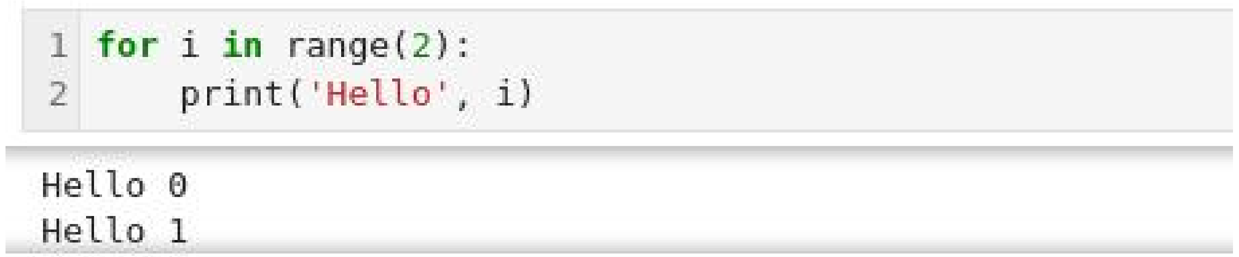


```
def repeat(message, n):
    print(message * n)

def first_two_lines(message, n):
    repeat(message, n)
    repeat(message, n)

first_two_lines('spam ', 4)
```

```
spam spam spam spam
spam spam spam spam
```

4. Repetition

- If we want to do something more than one time, we can use a **for** statement.



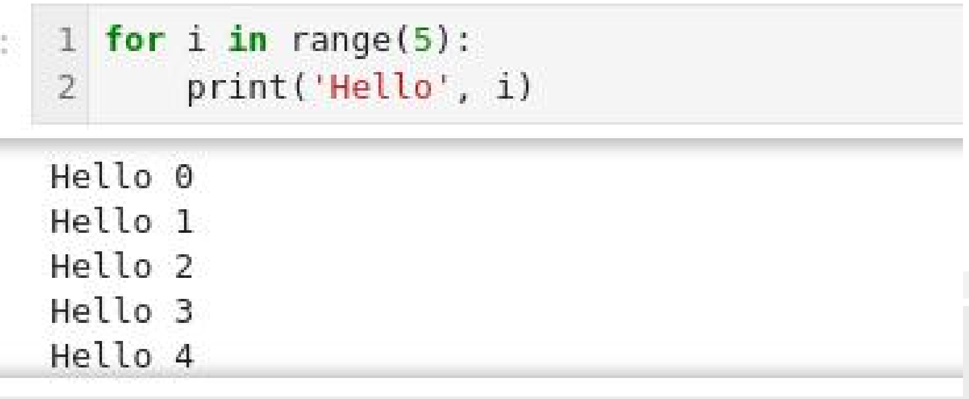
```
for i in range(2):
    print('Hello', i)
```

```
Hello 0
Hello 1
```

  - The first line is a header that ends with a colon. The second line is the body, which has to be indented.
  - The first line starts with the keyword **for**, a new variable named **i**, and another keyword, **in**. It uses the **range** function to create a sequence of two values, which are 0 and 1.
    - In Python, when we start counting, we usually start from 0.
  - When the for statement runs, it assigns the first value from **range** to **i** and then runs the **print** function in the body
  - When it gets to the end of the body, it loops back around to the header, which is why this statement is called a *loop*.
  - The second time through the loop, it assigns the next value from **range** to **i**, and then runs the **print** again. Then, because that’s the last value from **range** , the loop ends.

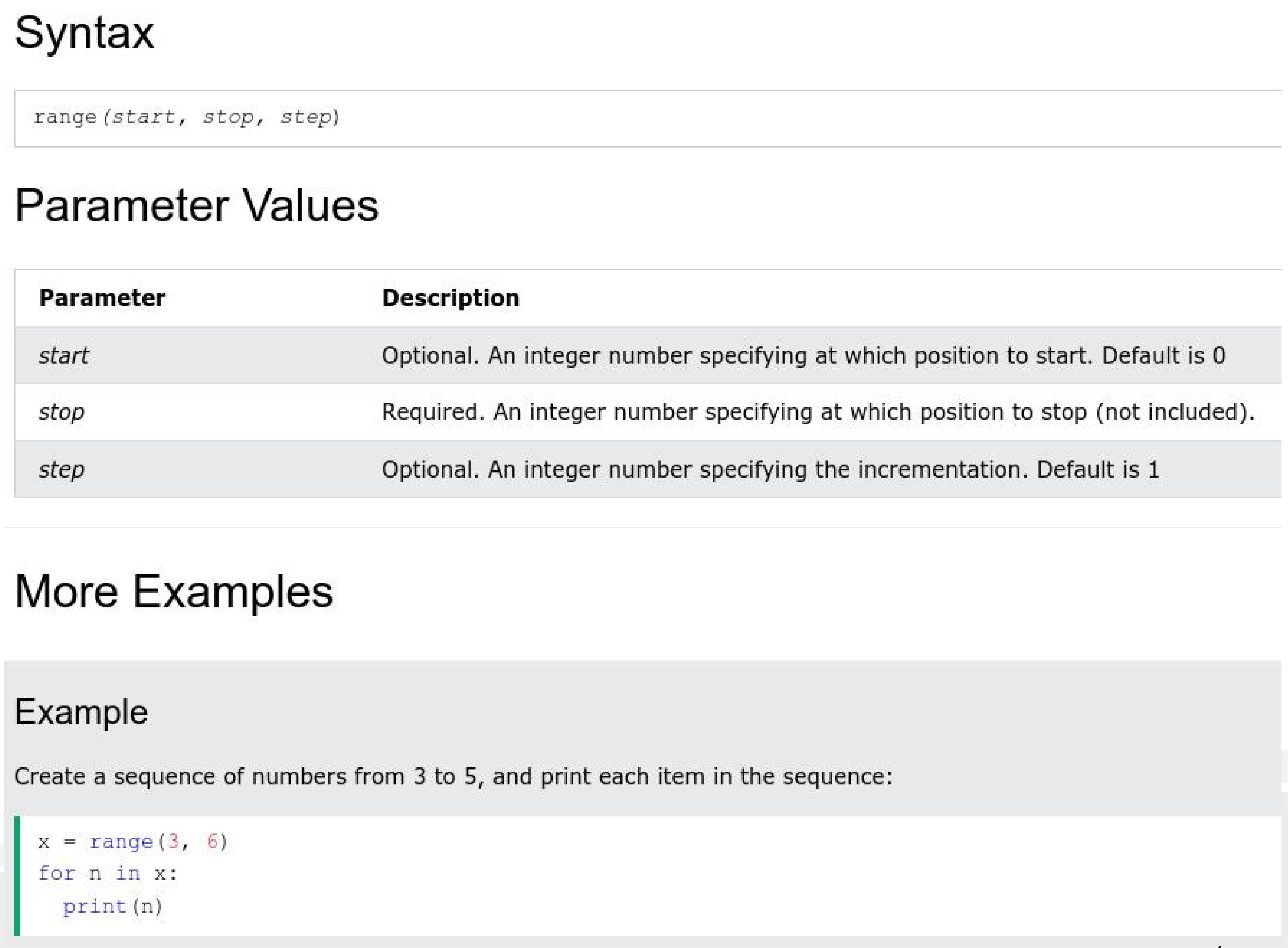
4. Repetition

- Python **range()** Function
  - The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.



```python
for i in range(5):
    print('Hello', i)
```

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```



Syntax

```
range(start, stop, step)
```

Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start. Default is 0
stop	Required. An integer number specifying at which position to stop (not included).
step	Optional. An integer number specifying the incrementation. Default is 1

More Examples

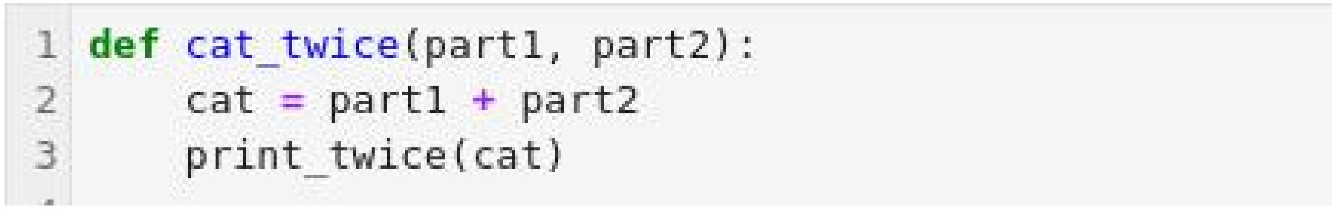
Example

Create a sequence of numbers from 3 to 5, and print each item in the sequence:

```python
x = range(3, 6)
for n in x:
  print(n)
```

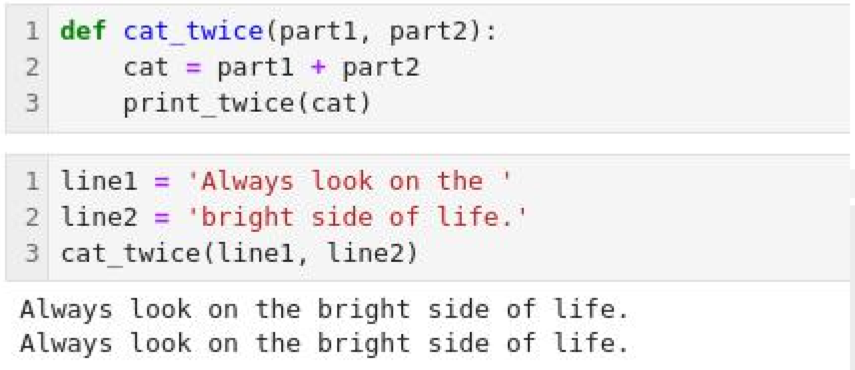
5. Variables Inside a Function and Parameters Are Local

- When you create a variable inside a function, it is **local**, which means that it only exists inside the function. For example, the following function takes two arguments, concatenates them, and prints the result twice:



```python
def cat_twice(part1, part2):
    cat = part1 + part2
    print_twice(cat)
```

- Here's an example that uses it:



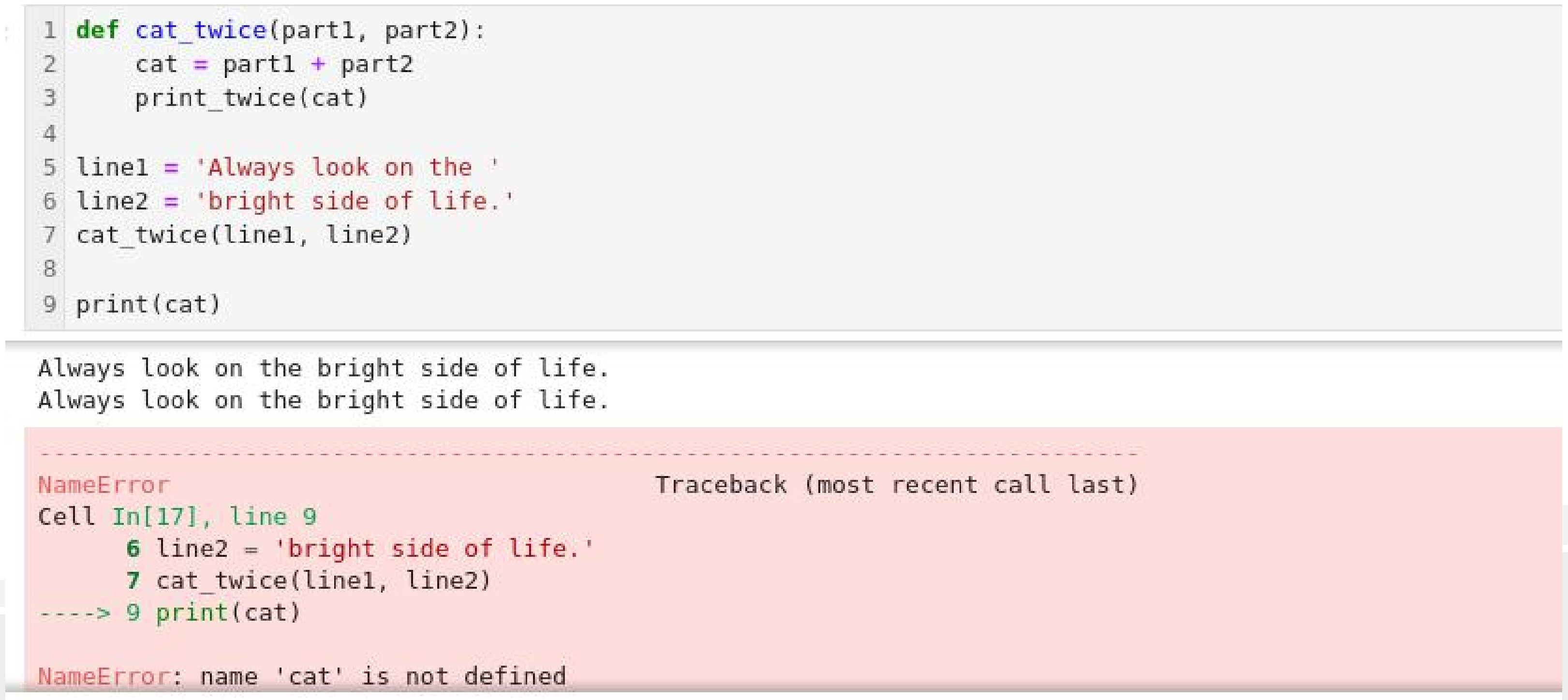
```python
def cat_twice(part1, part2):
    cat = part1 + part2
    print_twice(cat)
```

```python
line1 = 'Always look on the '
line2 = 'bright side of life.'
cat_twice(line1, line2)
```

```
Always look on the bright side of life.
Always look on the bright side of life.
```

5. Variables Inside a Function and Parameters Are Local

- When **cat_twice** runs, it creates a local variable named **cat**, which is destroyed when the function ends.
- If we try to display it outside **cat_twice**, we get a **NameError**:



```
def cat_twice(part1, part2):
    cat = part1 + part2
    print_twice(cat)

line1 = 'Always look on the '
line2 = 'bright side of life.'
cat_twice(line1, line2)

print(cat)
```

```
Always look on the bright side of life.
Always look on the bright side of life.
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 9
      6 line2 = 'bright side of life.'
      7 cat_twice(line1, line2)
----> 9 print(cat)

NameError: name 'cat' is not defined
```

5. Variables Inside a Function and Parameters Are Local

- Outside of the function, local variables are not defined.
- Parameters are also local.