Recursive functions

A recursive function is a [function](https://www.pythontutorial.net/python-basics/python-functions/) that calls itself until it doesn’t.

The following fn() function is a recursive function because it has a call to itself:

def fn():

*# ...*

fn()

In the fn() function, the #... means other code.

Also, a recursive function needs to have a condition to stop calling itself. So you need to add an [if statement](https://www.pythontutorial.net/python-basics/python-if/) like this:

def fn():

*# ...*

if condition:

*# stop calling itself*

else:

fn()

*# ...*

Typically, you use a recursive function to divide a big problem that’s difficult to solve into smaller problems that are easier to solve.

In programming, you’ll often find the recursive functions used in data structures and algorithms like trees, graphs, and binary searches.

Python recursive function examples

Let’s take some examples of using Python recursive functions.

1) A simple recursive function example in Python

Suppose you need to develop a countdown function that counts down from a specified number to zero.

For example, if you call the function that counts down from 3, it’ll show the following output:

3

2

1

The following defines the count\_down() function:

def count\_down(start):

""" Count down from a number """

print(start)

If you call the count\_down() function now:

count\_down(3)

…it’ll show only the number 3.

To show the numbers 3, 2, and 1, you need to:

* First, call the count\_down(3) to show the number 3.
* Second, call the count\_down(2) to show the number 2.
* Finally, call the count\_down(1) to show the number 1.

In order to do so, inside the count\_down() function, you’ll need to define a logic to call the function count\_down() with argument 2, and 1.

To do it, you need to make the count\_down() function recursive.

The following defines a recursive count\_down() function and calls it by passing the number 3:

def count\_down(start):

""" Count down from a number """

print(start)

count\_down(start-1)

count\_down(3)

If you execute the program, you’ll see the following error:

RecursionError: maximum recursion depth exceeded while calling a Python

The reason is that the count\_down() calls itself indefinitely until the system stops it.

Since you need to stop counting down the number reaches zero. To do so, you add a condition like this:

def count\_down(start):

""" Count down from a number """

print(start)

*# call the count\_down if the next*

*# number is greater than 0*

next = start - 1

if next > 0:

count\_down(next)

count\_down(3)

Output:

3

2

1

In this example, the count\_down() function only calls itself when the next number is greater than zero. In other words, if the next number is zero, it stops calling itself.

2) Using a recursive function to calculate the sum of a sequence

Suppose that you need to calculate a sum of a sequence e.g., from 1 to 100. A simple way to do this is to use a [for loop with the range() function](https://www.pythontutorial.net/python-basics/python-for-range/):

def sum(n):

total = 0

for index in range(n+1):

total += index

return total

result = sum(100)

print(result)

Output:

5050

To apply the recursion technique, you can calculate the sum of the sequence from 1 to n as follows:

* sum(n) = n + sum(n-1)
* sum(n-1) = n-1 + sum(n-2)
* …
* sum(0) = 0

The sum() function keeps calling itself as long as its argument is greater than zero.

The following defines the recursive version of the sum() function:

def sum(n):

if n > 0:

return n + sum(n-1)

return 0

result = sum(100)

print(result))

As you can see, the recursive function is much shorter and more readable.