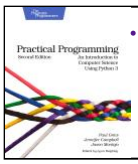
Visualizing Function Calls



- 3.5 Tracing Function Calls in the Memory Model

Optional reading

We can explore how Python manages function calls using the Python Visualizer. (See the Resources page.)

In the example below, function `convert_to_seconds` contains a call on `convert_to_minutes`.

```
def convert_to_minutes(num_hours):
    """(int) -> int
    Return the number of minutes there are in num_hours hours.
    >>> convert_to_minutes(2)
    120
    """
    result = num_hours * 60
    return result

def convert_to_seconds(num_hours):
    """(int) -> int
    Return the number of seconds there are in num_hours hours.
    >>> convert_to_seconds(2)
    7200
    """
    return convert_to_minutes(num_hours) * 60

seconds_2 = convert_to_seconds(4)
```

Here is what the memory model looks like just before the return statement inside function `convert_to_minutes` looks like:

```
1  def convert_to_minutes(num_hours):
2      '''(int) -> int
3      Return the number of minutes there are in num
4      >>> convert_to_minutes(2)
5      120
6      '''
7      result = num_hours * 60
8      return result
9
10 def convert_to_seconds(num_hours):
11     '''(int) -> int
12     Return the number of seconds there are in num
13     >>> convert_to_minutes(2)
14     7200
15     '''
16     return convert_to_minutes(num_hours) * 60
17
18 seconds_2 = convert_to_seconds(4)
```

Edit code

<< First | < Back | Step 6 of 8 | Forward > | Last >>

line that has just executed
next line to execute

Frames

Global variables
convert_to_minutes id1
convert_to_seconds id2

convert_to_seconds
num_hours id3

convert_to_minutes
num_hours id3
result id4

Objects

id1:function
convert_to_minutes(num_hours)

id2:function
convert_to_seconds(num_hours)

id3:int
4

id4:int
240

Note that there are three stack frames on the call stack: the main one, then underneath that a frame for the call on function `convert_to_seconds`, and underneath that the frame for the call on function `convert_to_minutes`.

Here is a link to the Python Visualizer at this stage of the execution so that you can explore this yourself. **We strongly encourage you to step backward and forward through this program until you understand every step of execution.**

When the return statement is executed, the call on `convert_to_minutes` exits. The bottom stack frame is removed, and execution resumes using the stack frame for `convert_to_seconds`:

```
1  def convert_to_minutes(num_hours):
2      '''(int) -> int
3      Return the number of minutes there are in num
4      >>> convert_to_minutes(2)
5      120
6      '''
7      result = num_hours * 60
8      return result
9
10 def convert_to_seconds(num_hours):
11     '''(int) -> int
12     Return the number of seconds there are in num
13     >>> convert_to_minutes(2)
14     7200
15     '''
16     return convert_to_minutes(num_hours) * 60
17
18 seconds_2 = convert_to_seconds(4)
```

Edit code

<< First | < Back | Step 8 of 8 | Forward > | Last >>

line that has just executed
next line to execute

Frames

Global variables
convert_to_minutes id1
convert_to_seconds id2

convert_to_seconds
num_hours id3
Return value id5

Objects

id1:function
convert_to_minutes(num_hours)

id2:function
convert_to_seconds(num_hours)

id3:int
4

id5:int
14400

Jennifer Campbell • Paul Gries
University of Toronto