Digital Career Institute

Python Course - Functions

Digital Career Institute

Recursive functions

Calling functions

From within the same function

```
>>> def my_function(global_var):
...     my_function(global_var)
...     # more instructions
...
>>> my_function(global_var)
```

Functions can also call themselves and become **recursive**.

‼ If we don't define a way to leave the loop, this would be iterating forever and it will never reach the lines after this instruction.

Python has a recursion limit.

```
>>> from sys import getrecursionlimit
>>> getrecursionlimit()
1000
```

Recursive functions *need a* ***halting condition*** *that guarantees the function will exit when required.*

Calling functions

Recursive functions

```
>>> def sum(list):
...     if not list:  # Base case
...         return 0
...     else:  # Recursive cases
...         first = list.pop(0)
...         return first + sum(list)
...
>>> print( sum([1, 3, 5, 9]) )
18
```

*We do all the calls and when we reach the **base case** we make the calculations **in reverse order**.*

A recursive function needs a way to detect the halting condition, or **Base case**. The base case will never include a recursive call and will finish the stack of calls.

The rest of the cases will include the recursive call and will indicate the operation that will need to be performed once all the calls are resolved.

Calling functions

Recursive functions phases

```
>>> def sum(list):
...     if not list:  # Base case
...         return 0
...     else:  # Recursive cases
...         first = list.pop(0)
...         return first + sum(list)
...
>>> print( sum([1, 3, 5, 9]) )
18
```

1. **Winding phase**. Drill down until the base case:

```
- sum([1, 3, 5, 9])
    - return 1 + sum([3, 5, 9])
        - return 3 + sum([5, 9])
            - return 5 + sum([9])
                - return 9 + sum([])
                    - return 0
```

2. **Unwinding phase**. Process the result:

```
                    - return 0
                - return 9 + 0 = 9
            - return 5 + 9 = 14
        - return 3 + 14 = 17
    - return 1 + 17
- 18
```

Calling functions

Recursive functions

```
>>> def sum(list):
...     if not list:
...         return 0
...     else:
...         first = list.pop(0)
...         return first + sum(list)
...
>>> print( sum([1, 3, 5, 9]) )
18
```

```
>>> def sum(list):
...     total = 0
...     for number in list:
...         total = total + number
...     return total
...
>>> print( sum([1, 3, 5, 9]) )
18
```

One same task can sometimes be done recursively and iteratively.

Recursive functions are often more costly and take more time to execute.

Calling functions

Digital Career Institute

Examples of recursive functions

Photo by Murad Swaleh on Unsplash

Tree data structures

- Directories & files
- Organization hierarchy
- Website pages
- DOM objects/XMLs

Network analysis

- Workflows
- Routing

In some other cases, recursive functions may be the best way to do a task.

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute