# Cheat Sheet

# Function Call Stack & Recursion

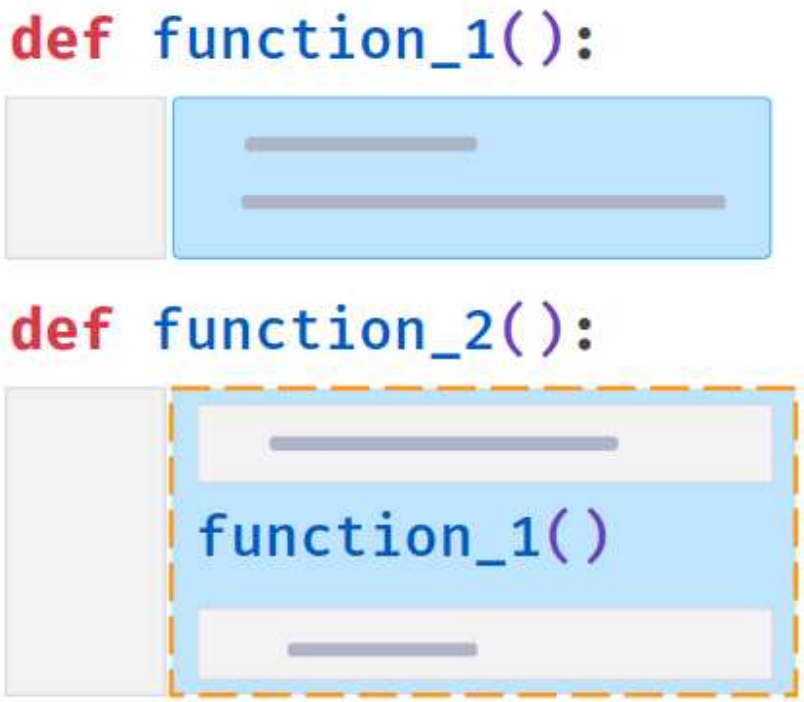
## Stack

Stack is a data structure that stores items in an Last-In/First-Out manner.

## Calling a Function

Calling

function_1() inside function_2()




### Code

PYTHON

```python
def get_largest_sqr(list_x):
    len_list = len(list_x)
    for i in range(len_list):
        x = list_x[i]
        list_x[i] = x * x
    largest = max(list_x)
    return largest

list_a = [1,-3,2]
result = get_largest_sqr(list_a)
```

Expand

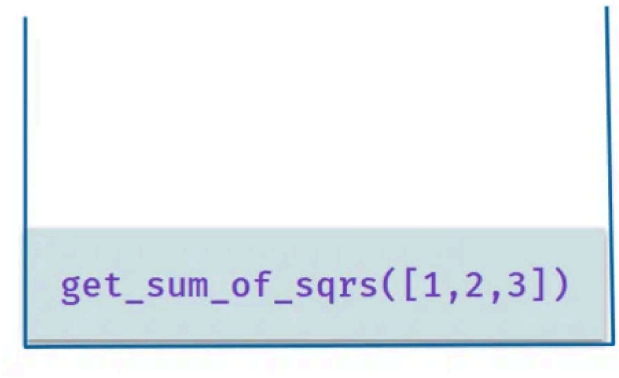
### Output

```
9
```

In the above code calling functions are

len() and max() inside get_largest_sqr()

## Sum of Squares of List Items




### Code

PYTHON

```python
def get_sqrd_val(x):
    return (x * x)

def get_sum_of_sqrs(list_a):
    sqrs_sum = 0
    for i in list_a:
        sqrs_sum += get_sqrd_val(i)
    return sqrs_sum

list_a = [1, 2, 3]
```

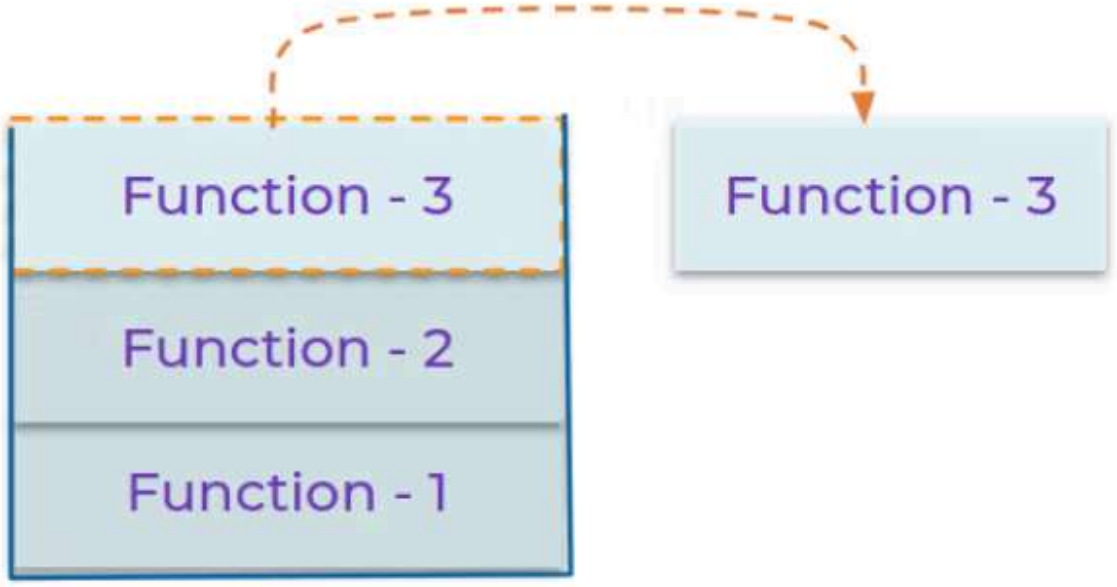
Expand

### Output

14

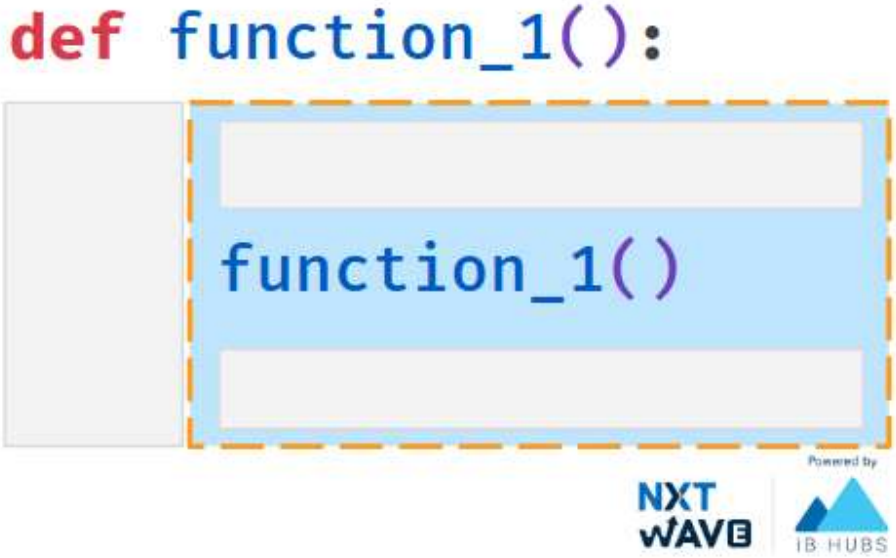
# Function Call Stack

Function Call Stack keeps track of function calls in progress




# Recursion

A function calling itself is called a **Recursion**




Let's understand recursion with a simple example of multiplying N numbers

## Multiply N Numbers

PYTHON

```python
def factorial(n):  # Recursive Function
    if n == 1:  # Base Case
        return 1
    return n * factorial(n - 1)  # Recursion
```

```
num = int(input())
result = factorial(num)
print(result)
```

### Base Case

A recursive function terminates when base condition is met

#### Input

```
3
```

#### Output

```
6
```

## Without Base case

#### Code

PYTHON

```python
def factorial(n):
    return n * factorial(n - 1)
num = int(input())
result = factorial(num)
print(result)
```

#### Input

```
3
```

#### Output

```
RecursionError: maximum recursion depth exceeded
```

Submit Feedback