

[Image Source](#)

# How to make logs useful?

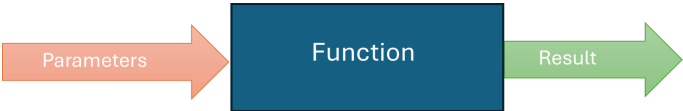
As developers, logs are the expressive part of our application. They allow us to capture the activity of an application.

In computer programming, a developer can decide whether to display logs in his application, which can be a traceability aid in the event of errors.

The problem now is to know what to display and where, hence the purpose of this article.

## Logs in a single function

By definition, a function receives one or more parameters as input, then performs operations and returns a result.



In a simple function, the useful information's to be displayed in the logs are:

- The parameter(s) the function receives.

- Result after operations (optional, useful when you have a sequence of function calls).

For example, for a function that takes no parameters and returns the message 'Hello' we would have in Python:

```
1 def hello():
2     print("hello()::called")
3     return "Hello"
```

Suppose the function took a name as parameter and returned "Hello username", we'd have:

```
1 def hello(name):
2     print(f"hello()::called with parameter name='{name}'")
3     return f"Hello {name}"
```

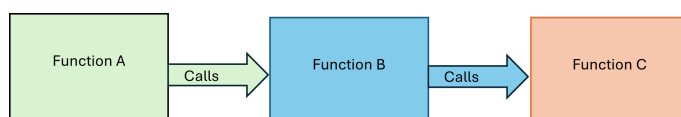
If we want to display the result of the function in the logs we would have:

```
1 def hello(name):
2     print(f"hello()::called with parameter name='{name}'")
3     response = f"Hello {name}"
4     print(f"hello()::executed with result:{response}")
5     return response
```

**When parameters contain confidential information such as email and password, these should not be displayed in the logs and should be replaced by \*\*.**

## Logs in a sequence of functions

A computer program is a sequence of functions and methods that call each other. Let's assume a program consisting of 3 functions: *function A*, *function B* and *function C*, with the following stack:

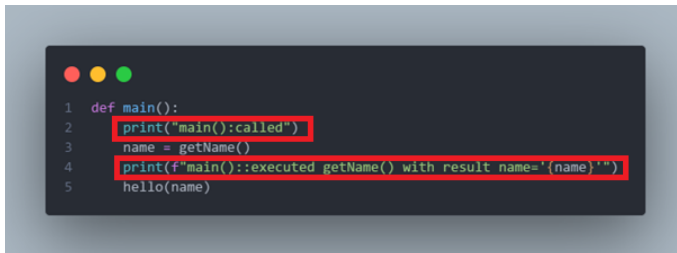


When you have several functions, the logs should look like this:

- The calling function must display its parameters when called.
- After executing another function, the calling function must display the result obtained.
- The calling function can display the returned result.

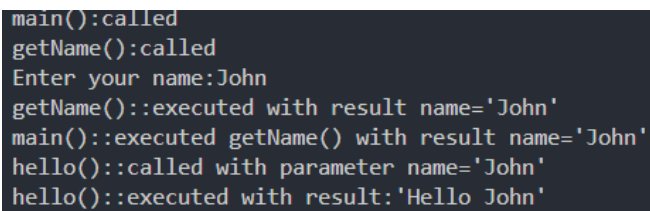
Let's say a **main** function calls a **getName** function and, with the name entered by the user, calls a **hello** function to display the message 'Hello username'.

If we want to apply the logs in this sequence of functions, we would have:



```
1 def main():
2     print("main():called")
3     name = getName()
4     print(f"main():executed getName() with result name='{name}'")
5     hello(name)
```

Execution would give :



```
main():called
getName():called
Enter your name:John
getName()::executed with result name='John'
main()::executed getName() with result name='John'
hello()::called with parameter name='John'
hello()::executed with result:'Hello John'
```

In the previous examples, we used the print function to display logs. In your projects, you should use log modules from your own language. In Python, you can use [Logging](#).

The techniques described above can be adapted to suit your needs.

When parameters contain confidential information such as email and password, these should not be displayed in the logs and should be replaced by \*\*.