Hallym LIKELION 11th Backend Study 2nd Week

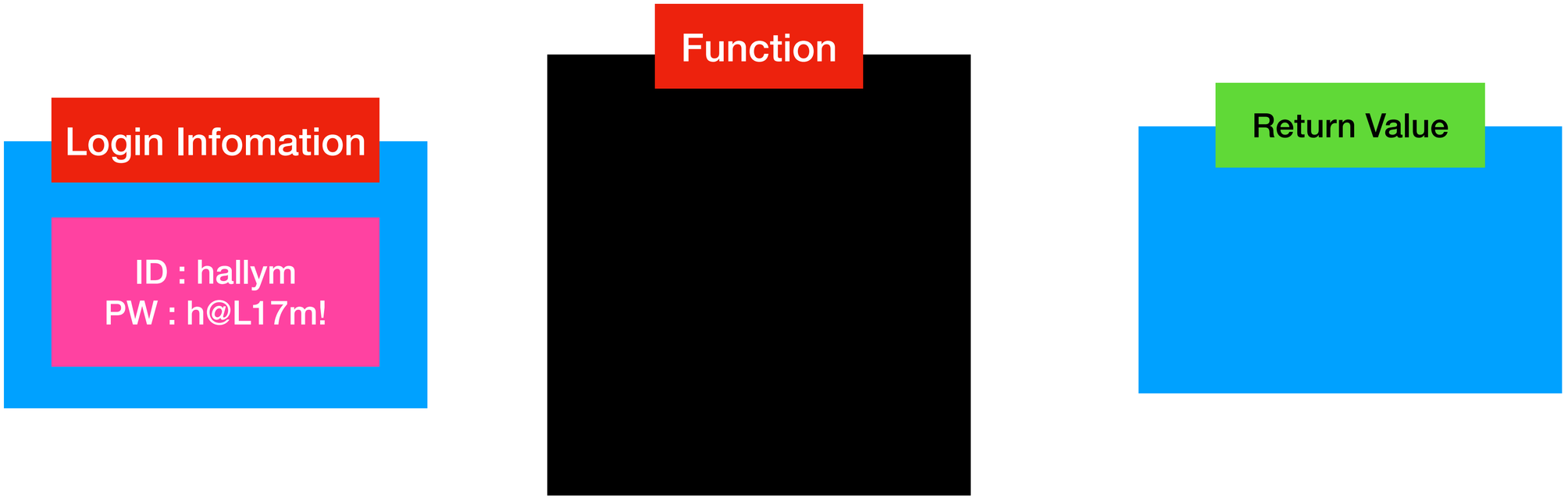
한림대학교 멋쟁이사자처럼 11기 백엔드 스터디 2주차

- Python

Bahk InSung

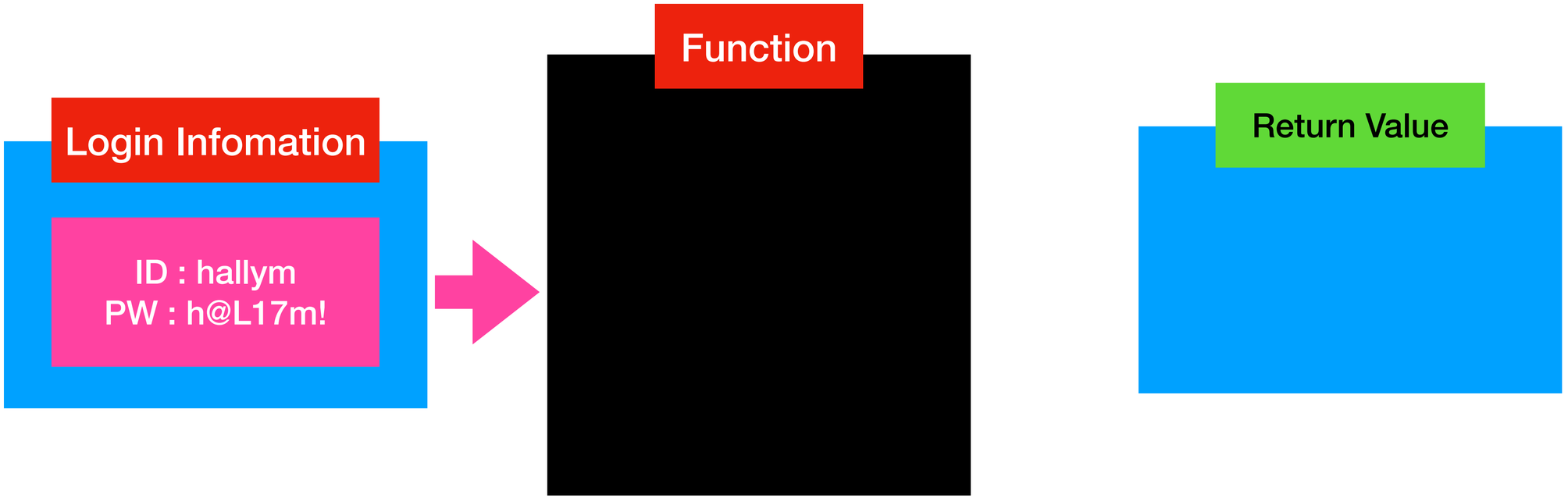
함수

Function in Python



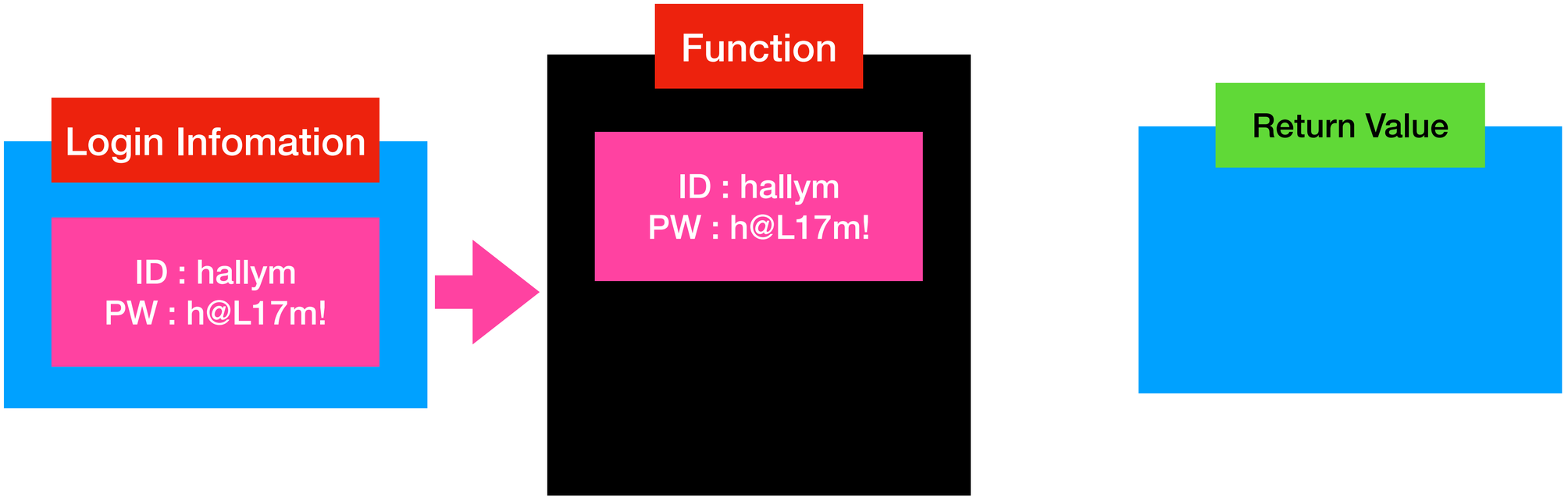
함수

Function in Python



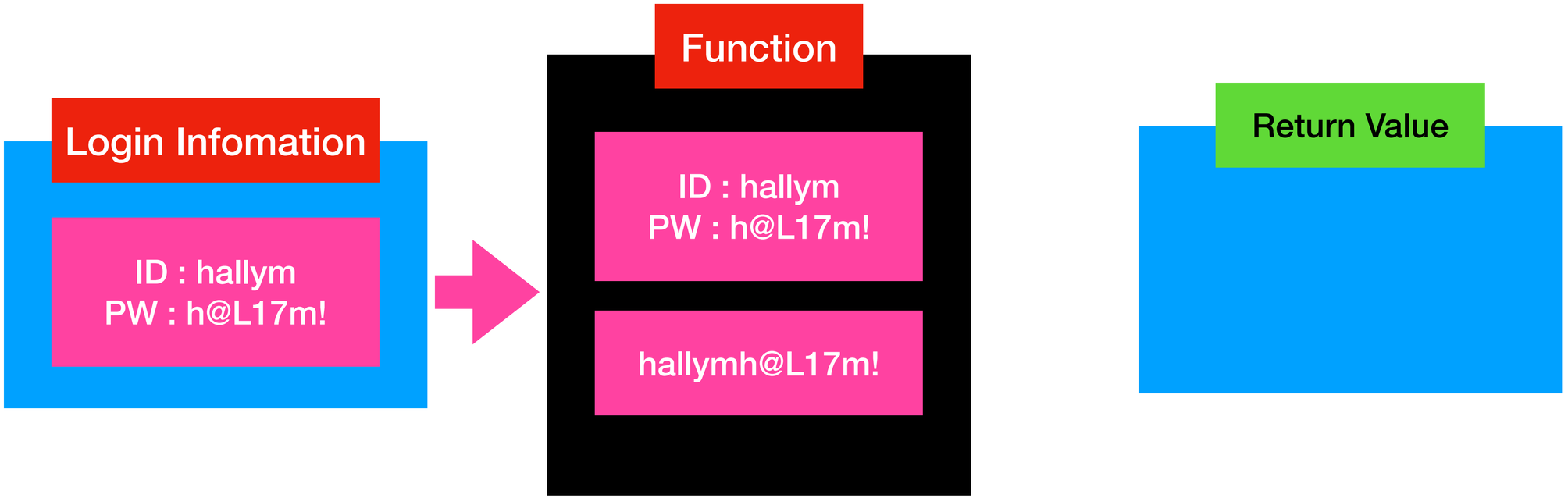
함수

Function in Python



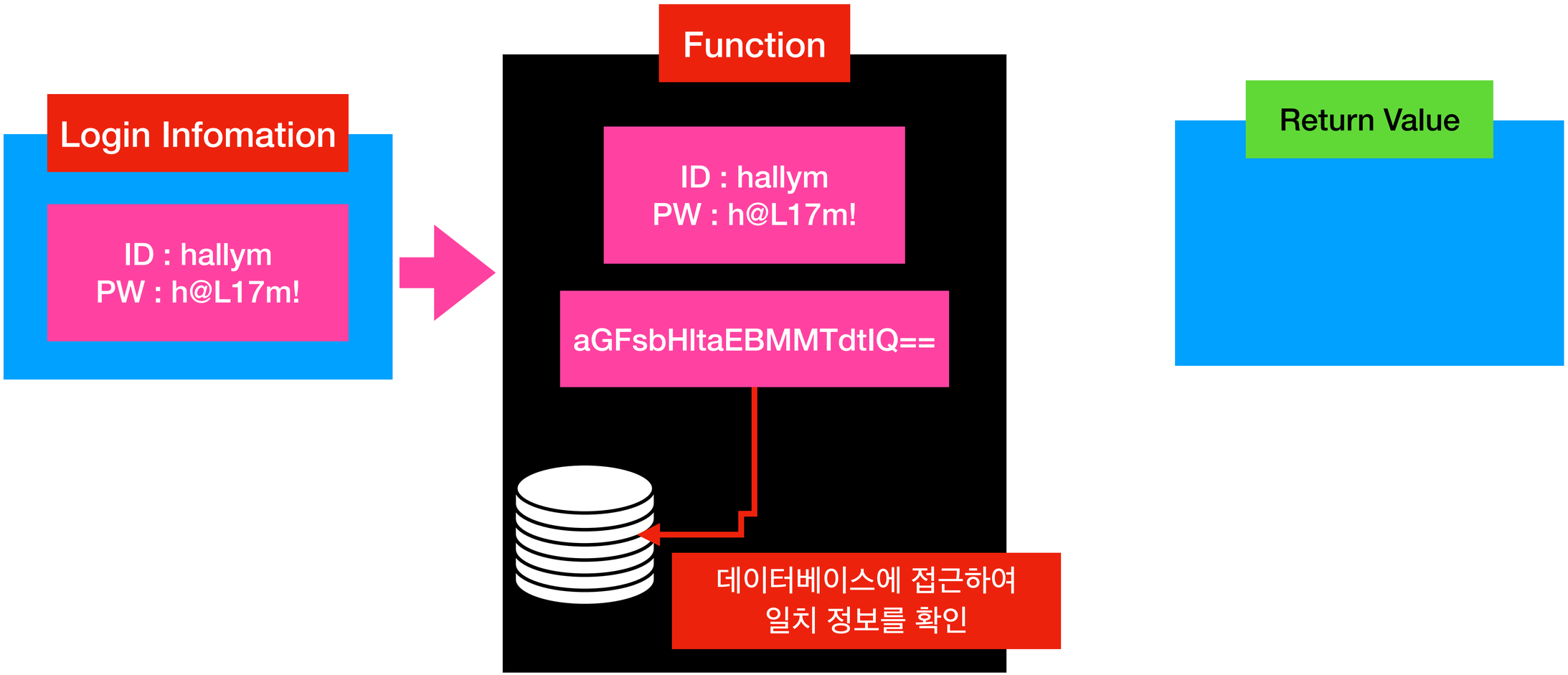
함수

Function in Python



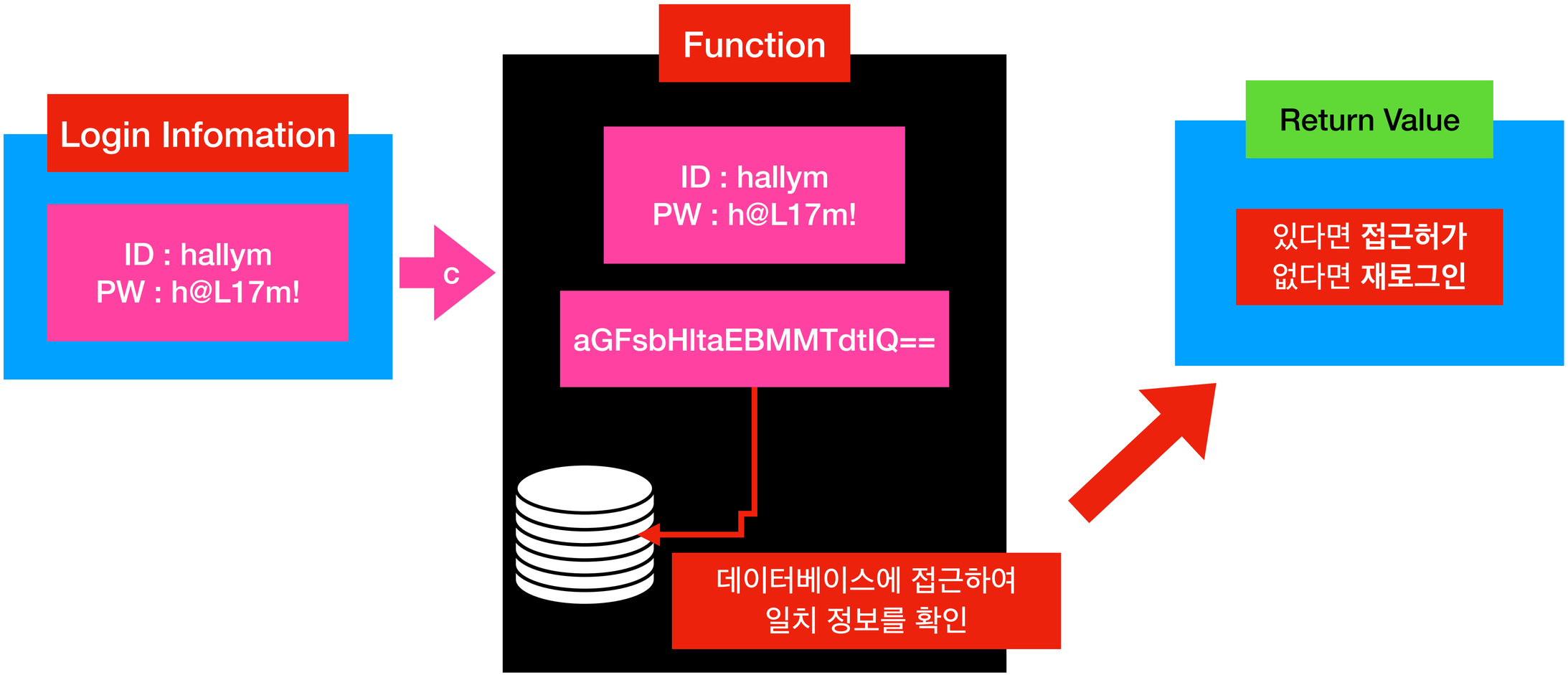
함수

Function in Python



함수

Function in Python



함수

Function in Python

Login Infomation

ID : hallym
PW : h@L17m!

User Infomation

user.json

함수

Function in Python

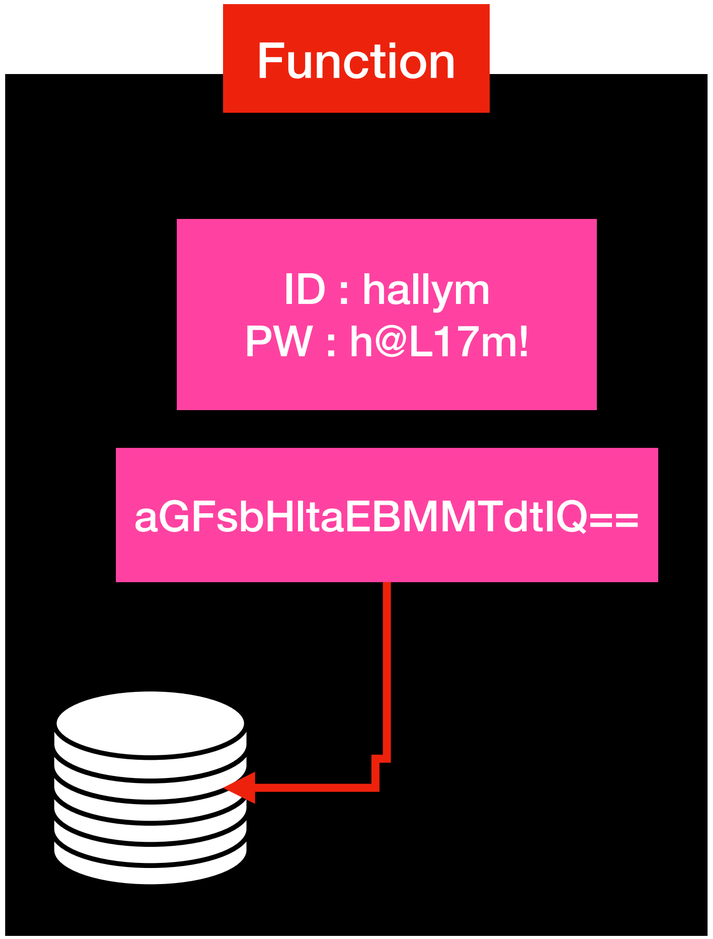
User Infomation

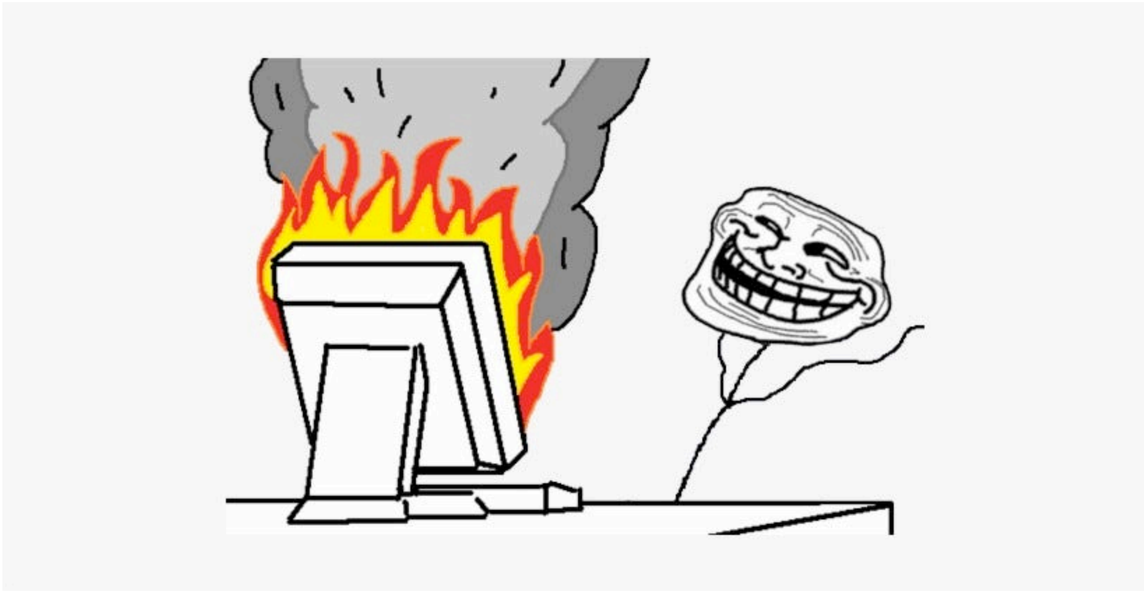
database.json

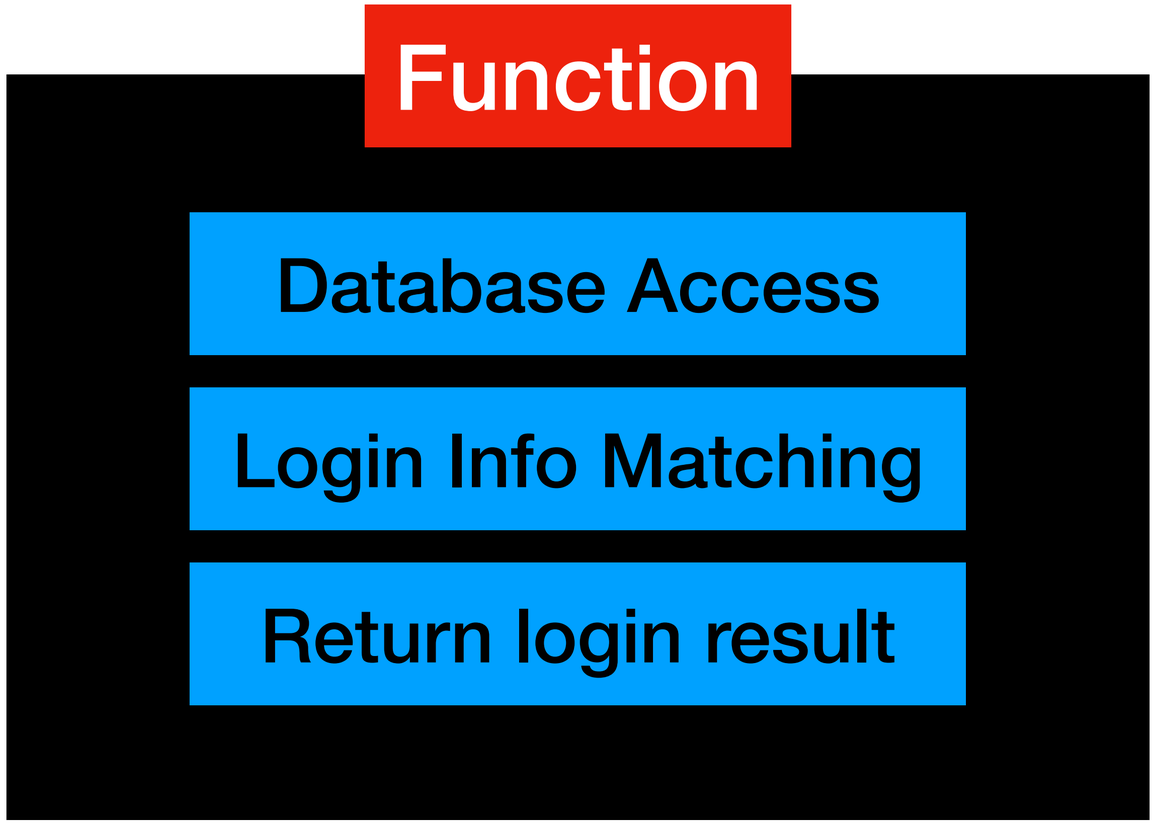
```
{
    "users": {
        "ID" : "hallym",
        "PW" : "p@ssw0rd"
    }
}
```

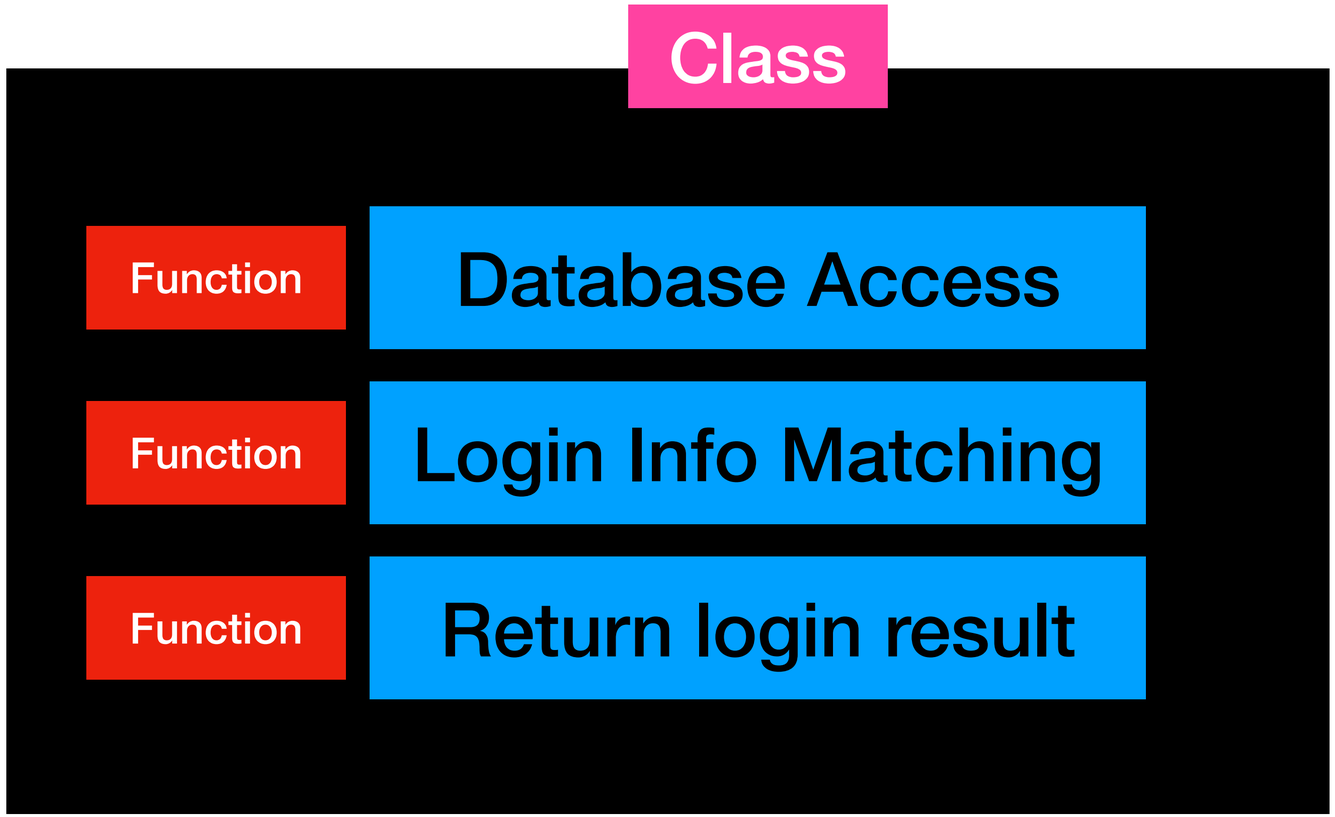
함수

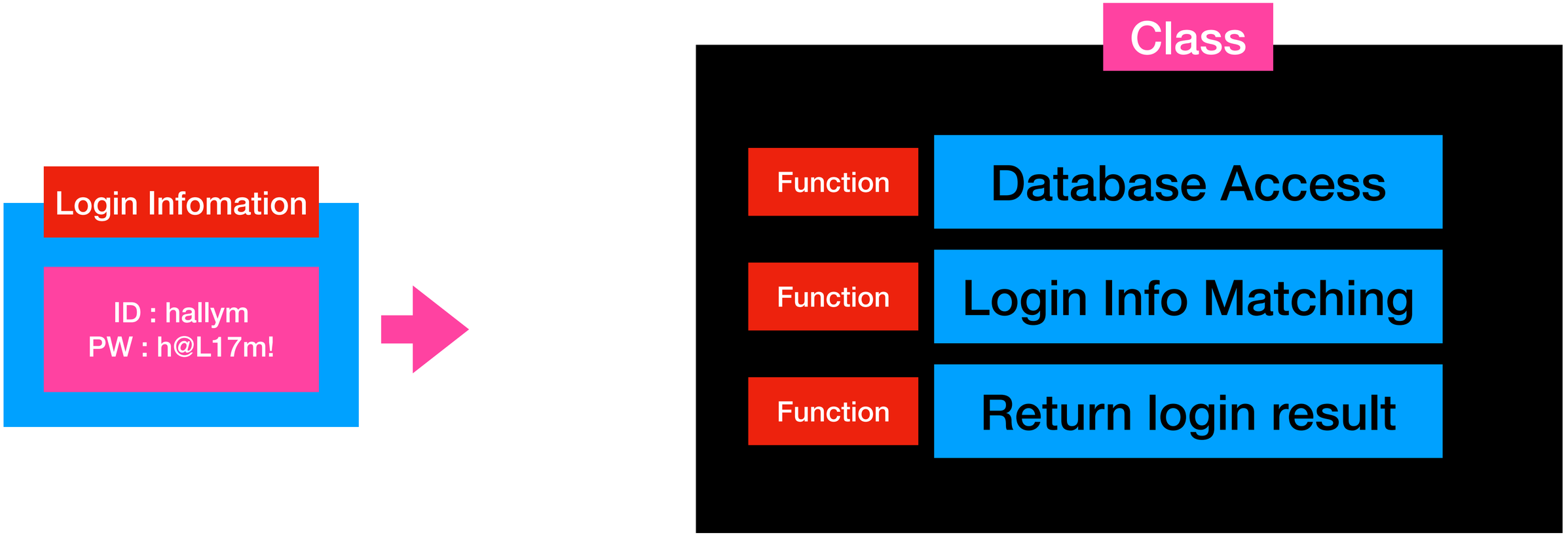
Function in Python

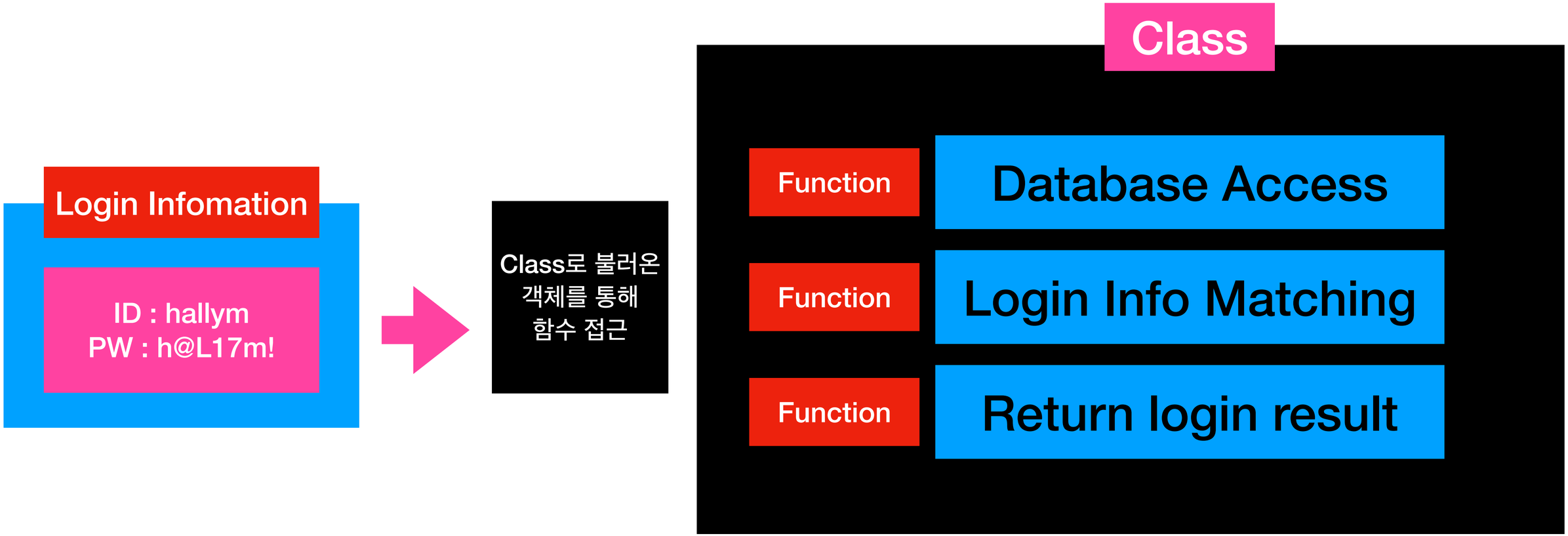





Function
Database Access
Login Info Matching
Return login result

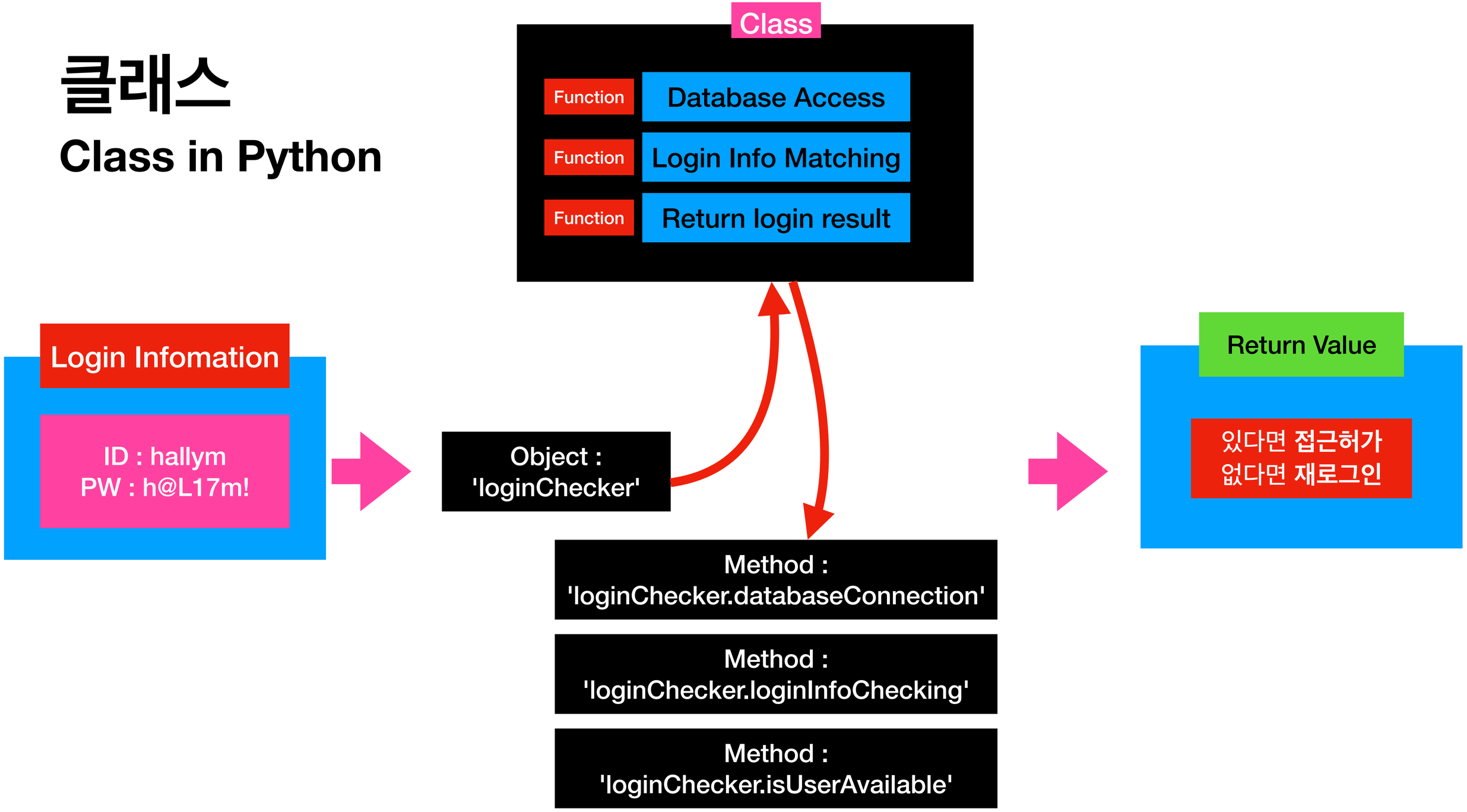

Class
Function
Database Access
Function
Login Info Matching
Function
Return login result


Login Infomation
ID : hallym
PW : h@L17m!
Class
Function
Database Access
Function
Login Info Matching
Function
Return login result


Login Infomation
ID : hallym
PW : h@L17m!
Class로 불러온
객체를 통해
함수 접근
Class
Function
Database Access
Function
Login Info Matching
Function
Return login result

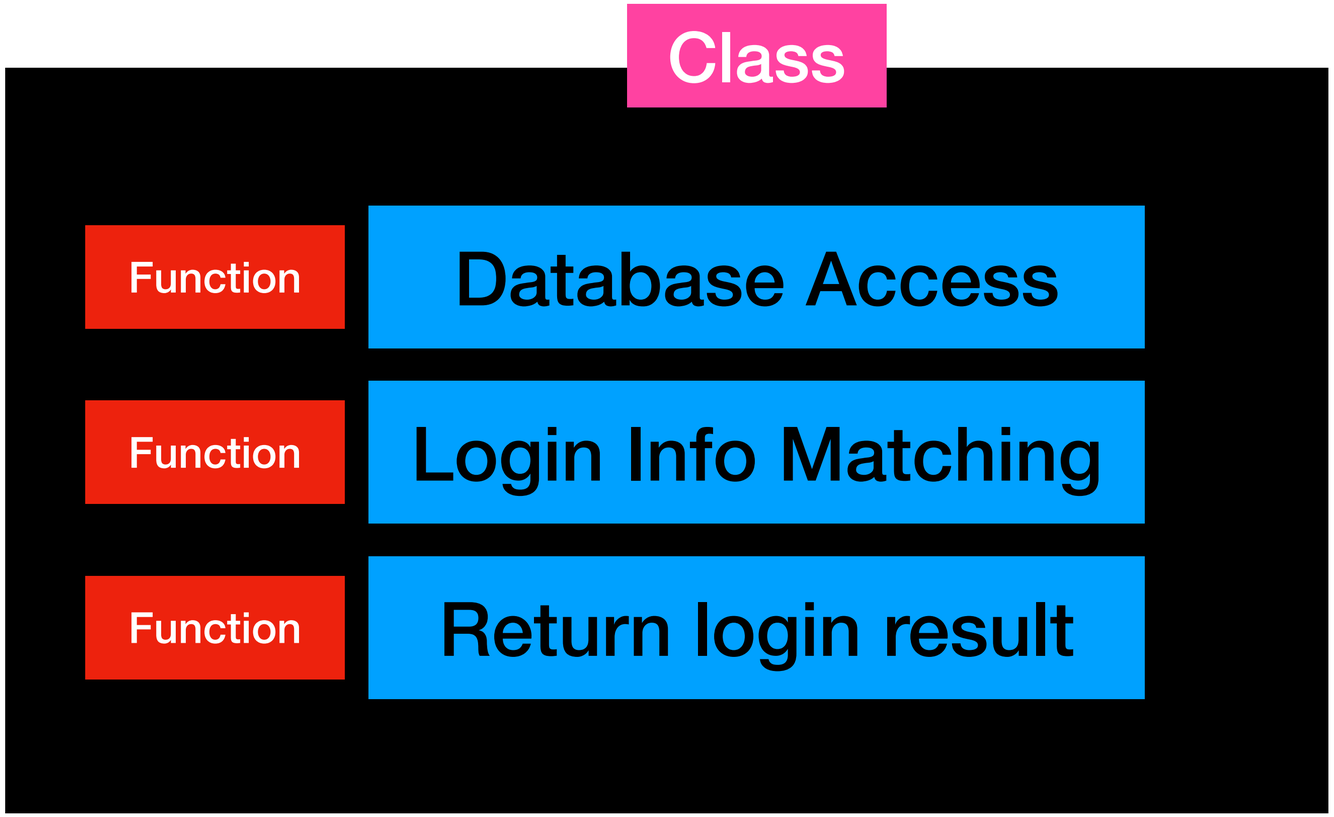
클래스

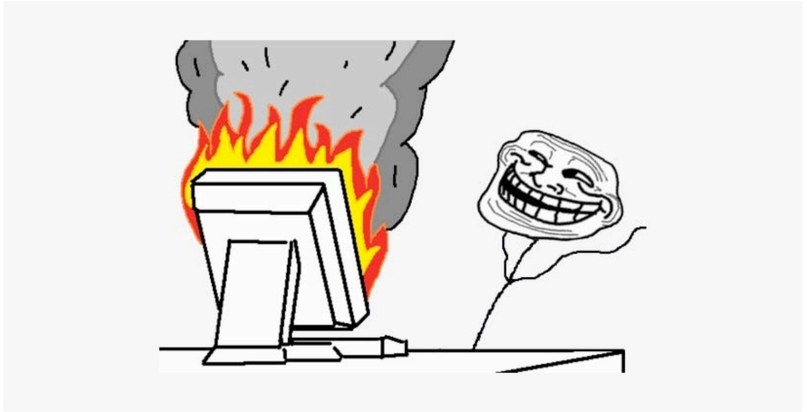
Class in Python



클래스

Class in Python





예외처리

Exception Handling in Python

예외처리

Exception Handling in Python

특정 구문을 시도 했을 경우 (try)
에러가 나는 경우에 예외처리 (except)

예외처리

Exception Handling in Python

Built-in Exceptions

In Python, all exceptions must be instances of a class that derives from `BaseException`. In a `try` statement with an `except` clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived). Two exception classes that are not related via subclassing are never equivalent, even if they have the same name.

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an "associated value" indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class's constructor.

User code can raise built-in exceptions. This can be used to test an exception handler or to report an error condition "just like" the situation in which the interpreter raises the same exception; but beware that there is nothing to prevent user code from raising an inappropriate error.

The built-in exception classes can be subclassed to define new exceptions; programmers are encouraged to derive new exceptions from the `Exception` class or one of its subclasses, and not from `BaseException`. More information on defining exceptions is available in the Python Tutorial under User-defined Exceptions.

Exception context

When raising a new exception while another exception is already being handled, the new exception's `__context__` attribute is automatically set to the handled exception. An exception may be handled when an `except` or `finally` clause, or a `with` statement, is used.

This implicit exception context can be supplemented with an explicit cause by using `from` with `raise`:

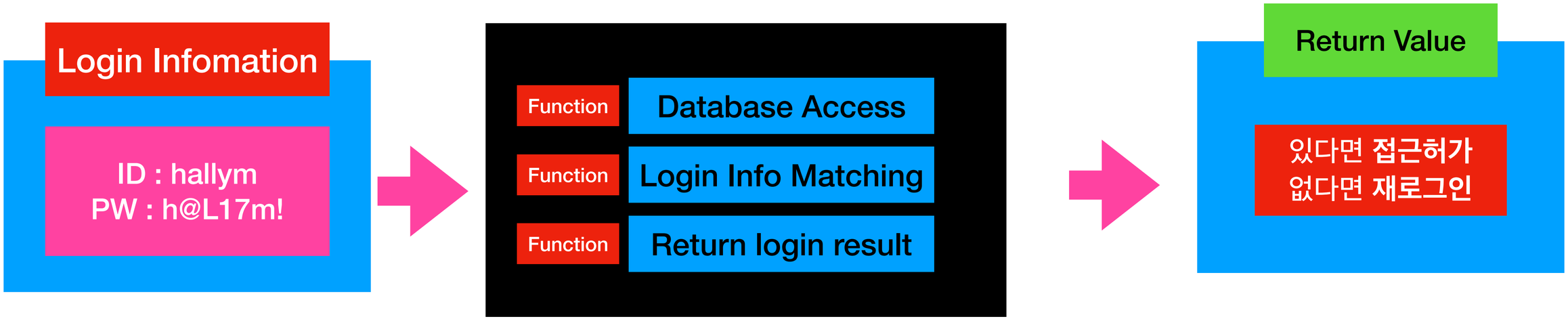
```
raise new_exc from original_exc
```

The expression following `from` must be an exception or `None`. It will be set as `__cause__` on the raised exception. Setting `__cause__` also implicitly sets the `__suppress_context__` attribute to `True`, so that using `raise new_exc from None` effectively replaces the old exception with the new one for display purposes (e.g. converting `KeyError` to `AttributeError`), while leaving the old exception available in `__context__` for introspection when debugging.

The default traceback display code shows these chained exceptions in addition to the traceback for

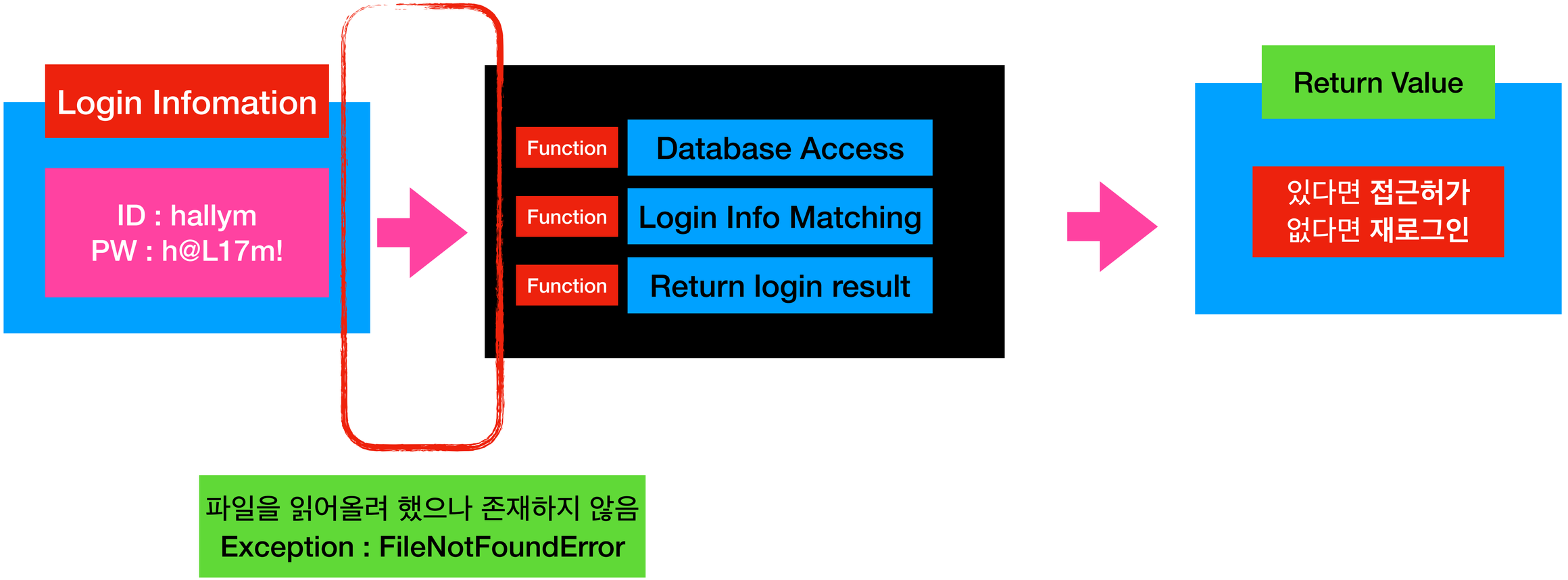
예외처리

Exception Handling in Python



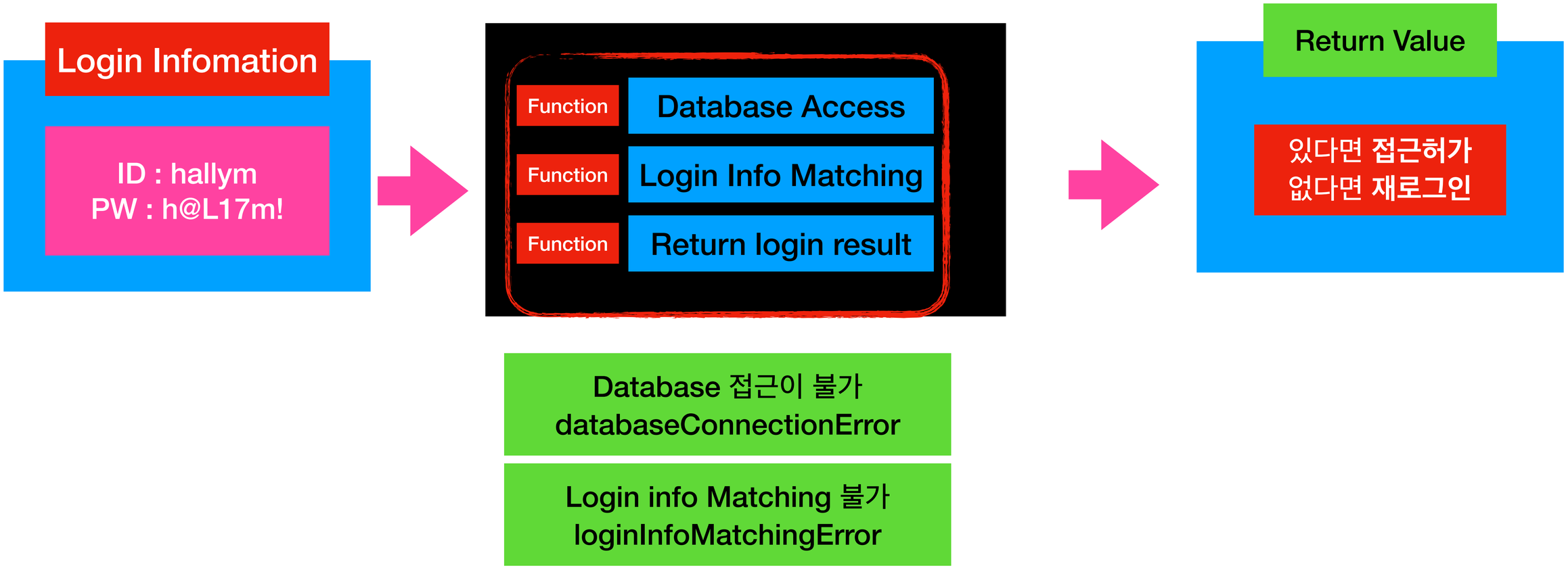
예외처리

Exception Handling in Python



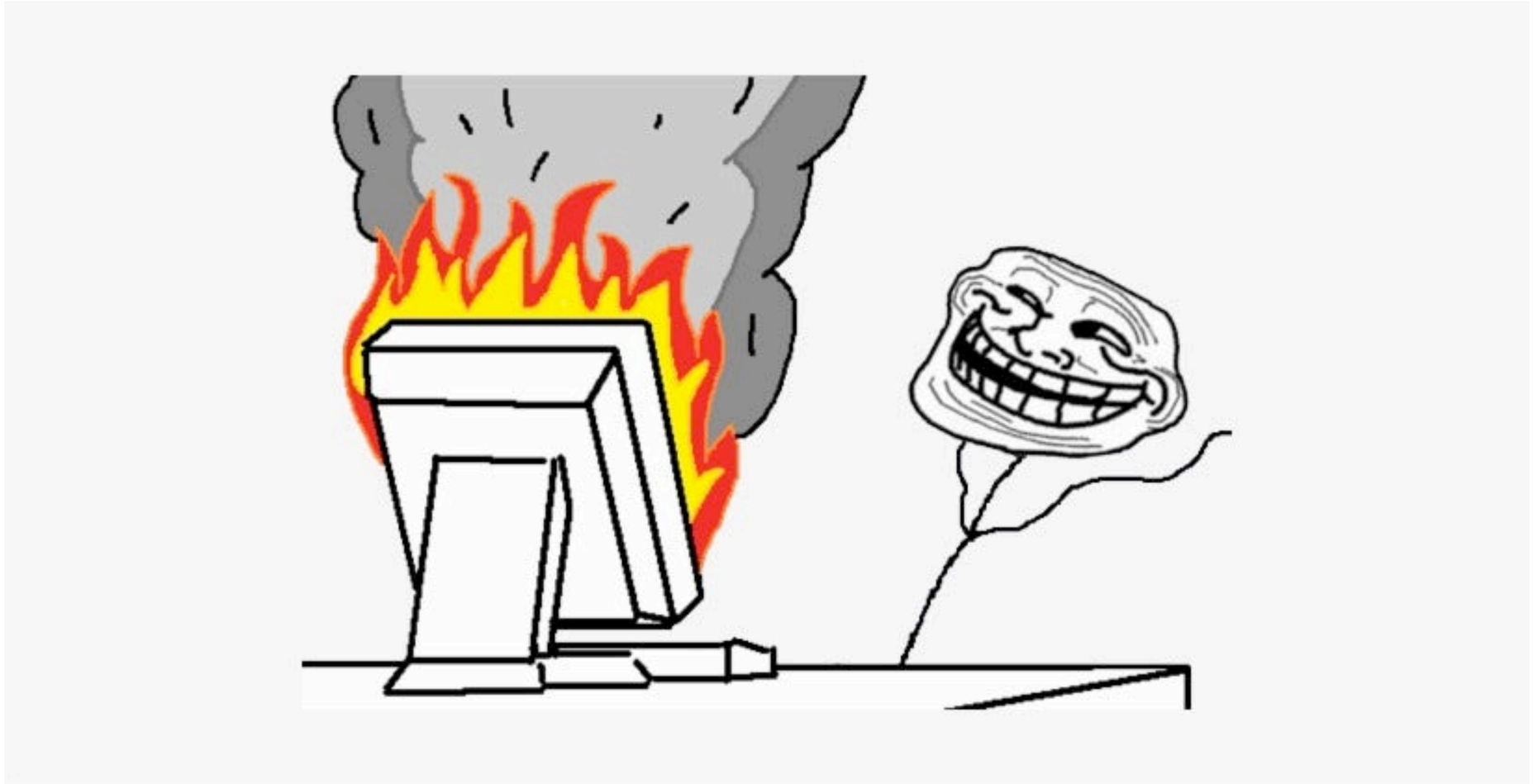
예외처리

Exception Handling in Python



예외처리

Exception Handling in Python



시험 화이팅