Oneday, if we have an program (hw0306_sample.c).
In main function, we call func_a, then call func_b in func_a.
However, if we want to debug in func_b, such as num can't smaller than zero.
In func_b, we need to return an error number to func_a, then func_a return errno number to main function.
It is too complex. And setjmp and lonjmp can deal with this problem.

For instance (hw0306_example.c), first, we declare a "jmp_buf" type varible to record the information of jumping.
Using "setjmp" to tag the jumping destination, and using "longjmp" at the program which need to jump.
And declare a number varible(jmpval) to record the error number.
When setjmp excute, jmpval will be set 0.
When longjmp excute, program will jump to setjmp function positon, and jmpval will be set val parameter of longjmp.

By doing so, if any function wnat to return error number and stop previous function, setjmp and longjmp can help us.

In addition,
in manual C11 7.13 Nonlocal jumps.2

> The type declared is "jmp_buf", which is an array type suitable for holding the information needed to restore a calling environment.

Acutally, jmp_buf is a array to store the information.