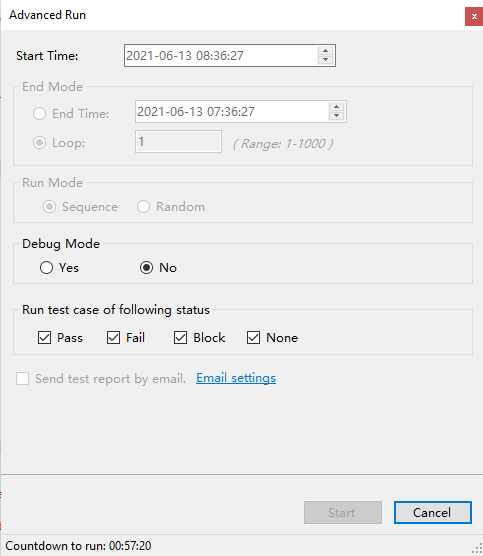
# AutoTest Studio Getting started tutorials 4: Task Scheduling

AutoTest Studio has a powerful task scheduling engine that allows users to flexibly run test tasks, such as running tasks regularly, running tasks quantitatively, and running test cases in specified states.

For demonstration purposes, we will copy 5 copies of the test cases in the second section, and modify the caseID to: TEST-1, TEST-2, TEST-3, TEST-4, TEST-5.

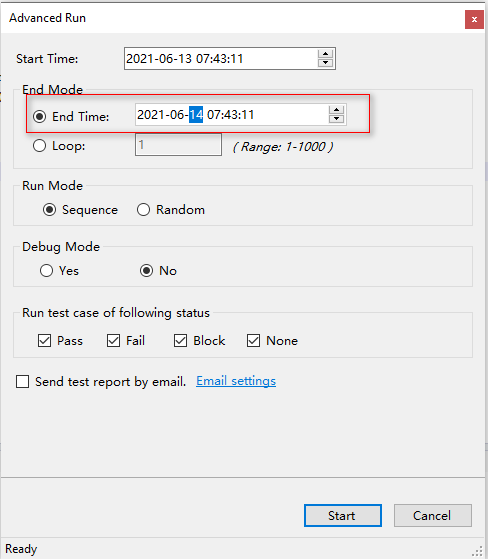
**Timed Run**

When the set Start Time is greater than the current time, AutoTest Stduio will enter a countdown and will not run until the specified time is reached. If the start time is less than the current time, the task will be run immediately.



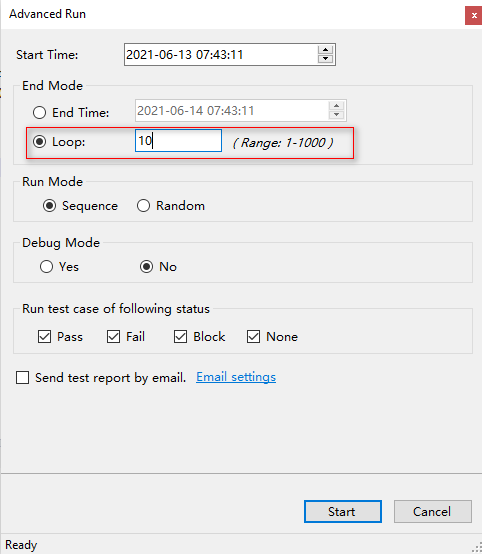
**Timing ends**

AutoTest Studio allows you to set the end time. When the end time is reached, the program immediately terminates the task. Here you don’t have to worry that AutoTest Studio will end the task violently. In fact, after the end time is reached, AutoTest Studio will run the current test case in Terminate the task.



**Specify the number of runs**

In addition to providing a specified time to end the task, AutoTest Studio also provides a specified number of runs. As shown in the figure below, AutoTest Studio will terminate the task after running all the use cases in the task 10 times.



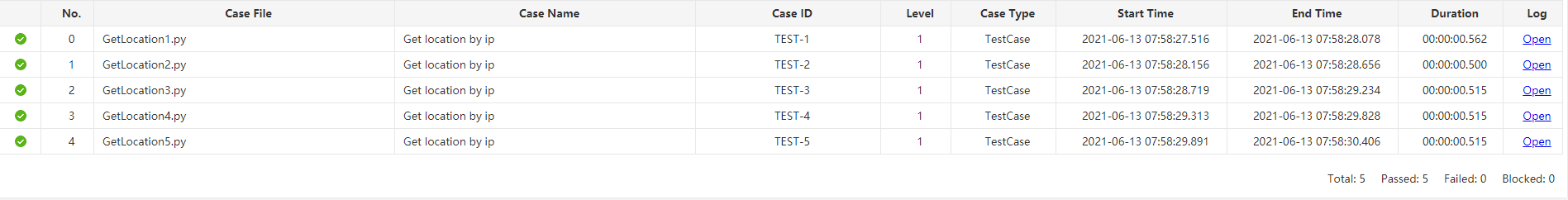
**Operating mode**

AutoTest Studio will run all use cases sequentially by default, but it also provides a random way to run use cases.

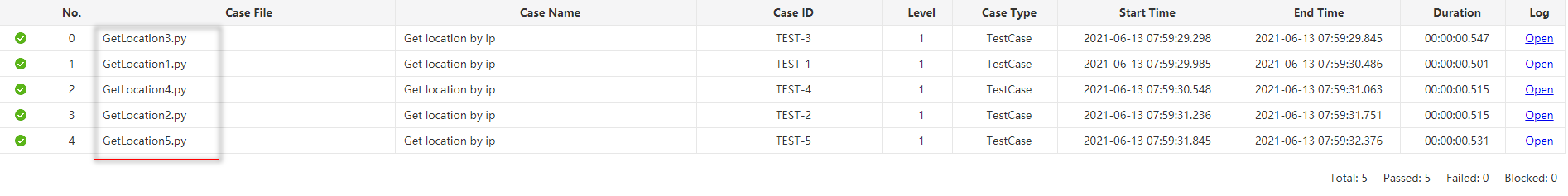
**Sequence**, run the use cases sequentially;

**Random**, run the use cases in a random manner.

The result of the sequential operation:



Random run results

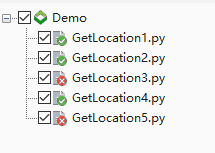


**Tuning operation**

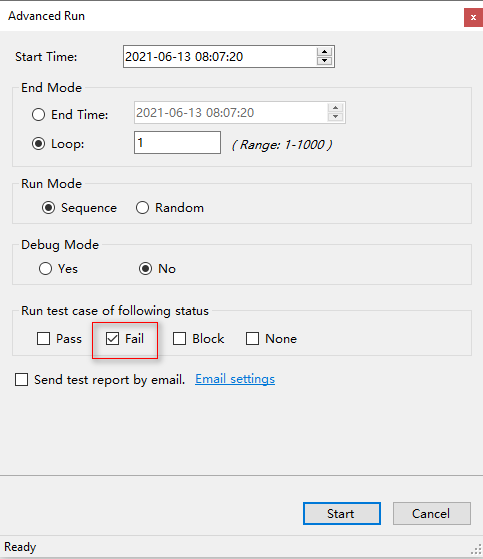
When Debug Mode is set to "Yes", AutoTest Studio can run test cases in a debug mode. This function is very useful when we locate some "strange" problems. We can let the use case enter a breakpoint when it runs to a special condition.

**Run the use case in the specified state**

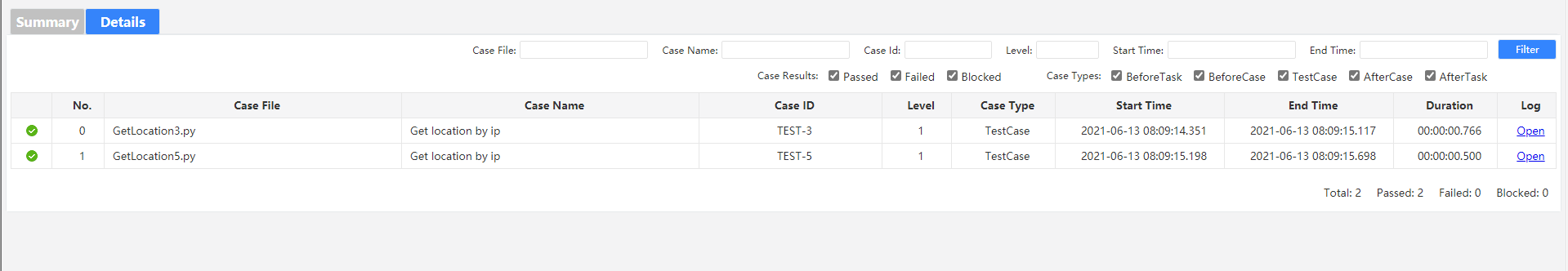
AutoTest Studio can run use cases in a specified state, such as the use cases that run the previous round of failures. We assume that the results of the previous run are as shown in the figure below.



Set the "**Run test case of following status**" of "**Advanced Run**" to run only the use cases with the "Fail" status.

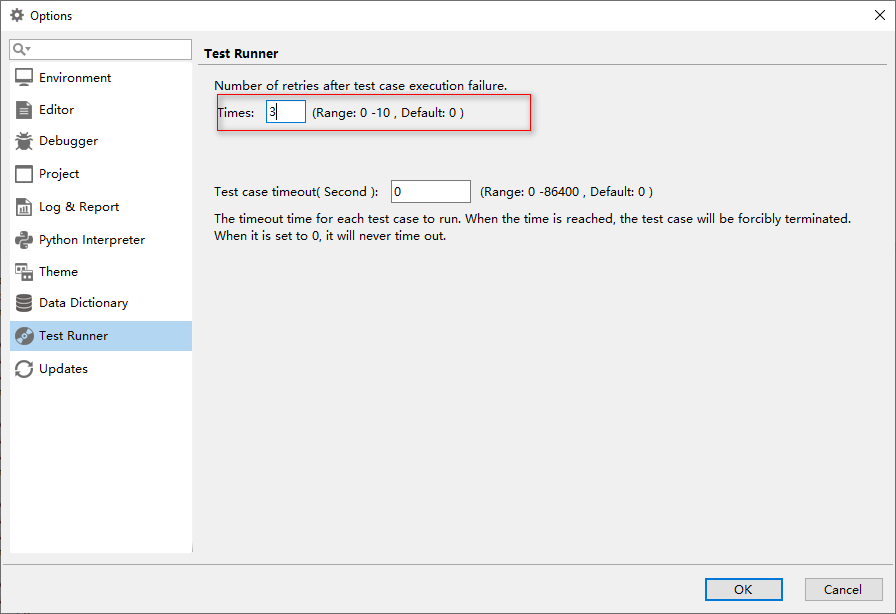


The following is the result of the operation. You can see that AutoTest Studio only ran "GetLocation3.py" and "GetLocation5.py", which are the use cases that failed in the previous run.



**Rerun after failure**

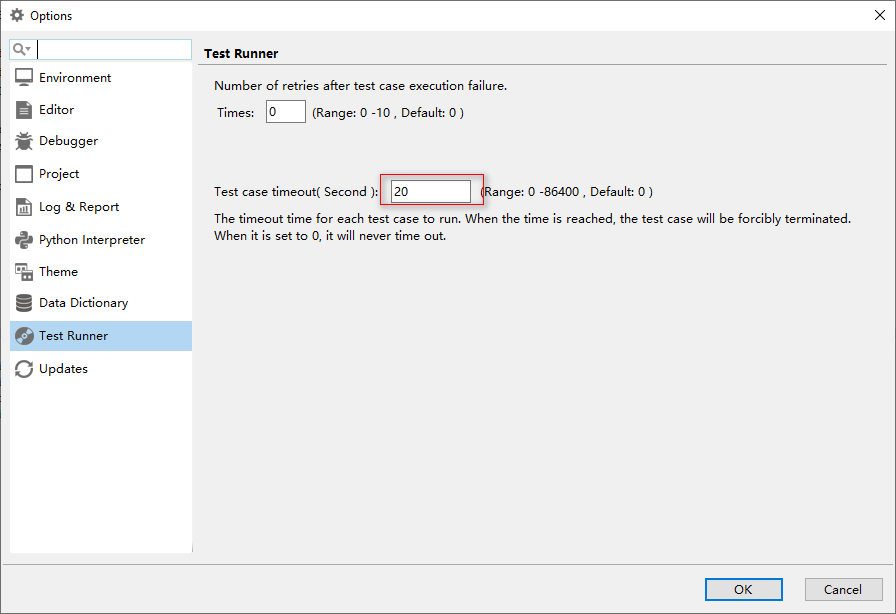
AutoTest Studio provides a global configuration parameter that allows the use case to run again after a failure, and can specify the number of repeated runs. The default is 0, that is, no repetition.



**Use case run timeout setting**

In many cases, the test case may be blocked due to some abnormal reasons, but we do not want the entire task to be blocked due to the blocking of this use case. Therefore, we can set the test case timeout period to force the suspension of the blocked test case and continue to run other cases. Use case, but it should be noted here that the timeout period cannot be set too small, because even when the use case is running normally, if the timeout period exceeds this timeout period, the operation will also be terminated.

Suppose we set the timeout period of the use case to 20 seconds, and we use time.sleep(25) to block the use case in the use case.



**Sample code:**

import time

time.sleep(25)

Run log results:

