https://tenthousandmeters.com/blog/python-behind-the-scenes-13-the-gil-and-its-effects-on-python-multithreading/

If the OS schedules threads, not processes, for execution, then
1 process
can spawn n threads
each of which can run on a different core!

However, per process, the python GIL prevents the use of > 1 CPU core to run a python thread. So "a multi-threaded program is not faster than its single-threaded equivalent even on a multi-core architecture".
See the example of performing 100MM decrements.
You can run decrement(n=100MM) on one thread.
Or decrement(n=50MM) on two threads.
If the two threads ran in parallel, you would expect the total time for all decrements to be completed to be halved in the latteer case for the same total number of operations.
However, you see that the time to completion will be the same in both scenarios. Because there is no parallelism -- only concurrency

However, for IO bound tasks, python threads voluntarily release the GIL so we can achieve multipl threads running on different cores here