Logger Module

Requirements:

1. A call to Write on the interface must be as fast as possible so the calling application can get on with its work without waiting for the log to be written to a file.
    a. Achieved through spawning a Process for each write to the file, each of them appends to the log file.
2. If we cross midnight a new file with a new timestamp must be made
    a. Log file is updated each time the module is used to write a log
3. It must be possible to stop the component in two ways:
    a. Ask it to stop right away and if any outstanding logs they are not written
        i. Achieved by running a Process.terminate() on all write processes
    b. Ask it to stop by the wait for it to finish writing outstanding logs if any
        i. Achieved by running a Process.join() before program is stopped
4. If an error occurs the calling application must not be put down. It is more important for the application to continue to run than lines not being written to log
    a. Each write process is bound by a try-except decorator function so all errors would be handled without the program crashing

Known Issues/ Improvement Scope:

The module does some processing to check if midnight has passed/ log text generation before spawning a process, this may affect the program flow of the script using this module. Could be fixed by pushing more of the processing on to the spawned process. A type of shared memory could also be used so that checks like whether midnight have passed could be shifted to another process instead of checking it each time.

There is also a bug in the code right now where there are too many open locks on the file. This can be remedied by setting ulimit -n unlimited

My goal was to push all of these write tasks into a processing queue, where worker threads would pick up these jobs one by one. But I ran out of time/ couldn't research into this enough.

Unit Testing:

For testing if a file is written, the unit test simply checks if a log file has been written to the temp directory

For testing the stopping function,

To check if all logs that are supposed to be written, are written after the logger is stopped, we check if the number of lines in the log file is the same as the number of logs we actually sent from our program

To check if outstanding log write requests are ignored, we check if the number of lines in the log files is lower than the number of logs we actually sent from our program

Unfortunately, I ran out of time to test the midnight function and leave commenting on my code