Bonus Questions:

1. How would you optimize the `get_jobs` function?
    a. This could be done using better data_structure , instead of iterating the entire list.
    b. When using a data structure that searches pending jobs in O(1).
    If we are using an external database, we can index jobs based on status.
    c. To obtain a fair scheduling we can also use Multi-level-feedback Queue.

2. Are there tools that you would use instead of writing this script to manage the job scheduling? How would the entire solution change to adopt them?
    a. Cron Job - Linux has a cron job utility where we can schedule different tasks, also Kubernetes cron is an excellent choice for such a system
3. What would you do differently if the job was CPU-bound rather than IO-bound? Particularly since Python is not a parallel language (i.e. GIL).
    a. Since Python is not a parallel language multi-threading does not make sense in CPU Bound processes. We can instead use Multi-Processors where each core has its own GIL.
    We would also need to take care of how graceful shutdowns are handled on each core.
4. How should someone deploying a scheduler-powered job determine their value for `--max-concurrent-jobs`?
    a. It should depend on number of cores available to run CPU-Bound process in a multi-processing environment.