Università degli Studi di Napoli Federico II
Advanced Computer Programming

Creare un'applicazione Python multi-processo caratterizzata dalle seguenti classi:

- **Generator**: classe che rappresenta un processo, il cui metodo run è in grado di generare 10 messaggi di log testuali di due tipi: INFO-{numero casuale} oppure ERROR-{numero casuale}. Dopo la generazione, il messaggio viene inserito in una Queue condivisa con il Log Manager (classe *LogManager*).

- **LogManager**: classe che rappresenta un processo, il cui metodo run analizza i messaggi ricevuti attraverso la Queue condivisa con i Generator. Ogni messaggio prelevato dalla Queue, viene inserito in una coda condivisa con il FileWriter. Inoltre se il messaggio contiene la stringa ERROR, tale messaggio viene inviato all'Error Server (classe *ErrorServer*) attraverso socket TCP.

- **FileWriter**: classe che rappresenta un processo, il cui metodo run preleva i dati dalla coda condivisa con il LogManager e li scrive (in modalità append) su un file events.log.

- **ErrorServer**: applicazione Python a se stante, che riceve, tramite socket TCP, i messaggi inviati dal LogManager, e li scrive su un file errors.log. Ad ogni ricezione, l'Error server stampa a video il numero di messaggi ricevuti fino a quel momento.

Realizzare il software come descritto utilizzando il pattern Proxy-Skeleton, avviando 2 generatori, 1 log manager, 1 file writer, 1 error server.

Provare a realizzare anche una variante che sfrutta le pipeline (**Pipe**) invece della Queue e socket UDP invece di quelle TCP.

Provare a realizzare una ulteriore variante dove i processi sono sostituiti da thread, e le Queue da liste, e prevedere i necessari meccanismi di sincronizzazione tra i thread.