## Task 1

### a

**add_entry**

- on initialization of a board, an indexed_entries dict is defined
- on call of add_entry: self and an entry is passed and added to indexed_entry dict by id

**class: entry**

- each entry has an id and a value
- entries are stored in a dict with their is and their value using **to_dict** function

**create_entry_request**

- first function checks, whether the server is crashed
- then it gets the next id value for a new entry
- after that it sends a message to the first server (coordinator) in the server list (function: send_message)
- this message contains as a type add entry and as the entry value the previously defined entry_value

**send_message**

- coordinator propagates the message (thus the new entry) to all other servers, including itself
- if a server receives an add entry-message, it will add the entry to the board

TODO proper explanation

### b

Since there's one coordinator that always notifies every other server of new entries to the board, which then add that entry to itself, this way, assuming no messages get lost and always arrive in the same order, guarantees consistency within each server.

### c

This implementation is advantageous as it is very easy to implement and doesn't require as many messages to be sent over the network (1 (user adds entry) + 1 (coordinator is notified) + n (propagate the new entry to all servers including itself) = n+2 messages with n being the number of servers).
Downsides of this implementation includes poor scalability: One coordinator being responsible for propagating new entries to every server might work fine for a small number of servers, but gets a bit tricky, when there's a huge

number of servers involved. Additionally, when the need for a huge number of servers arises, one might assume, there is also heavy use of the system to be seen, which might overload the coordinator quickly, resulting in poor performance, as there is only so much a single coordinator might able to handle in a timely manner.
Also, in the real world we cannot always guarantee that the coordinator doesn't fail, meaning, if the single coordinator fails, no more entries will be propagated and consistency is lost quickly. The same thing applies for lost messages: If the notification of a new entry doesn't arrive at coordinator side, it will not be propagated, thus resulting again in inconsistency. The same applies for propagation messages that are lost.