keepalive message implementation

Every node starts a new thread and opens a server using the backend port we get from the configuration file. The server continues to listen and handle different message.
Every node starts a new thread to send liveness message to its neighbors. The node uses a global dictionary to record all the possible neighbors and its connection status. For every 3 second, search for all neighbors in the global dictionary. For each neighbor, start a new thread to send a “Are you alive? “message. And wait 0.5 second for the confirmation message. If the node miss confirmation message for 3 times, we set this neighbor as disconnected.
When server receive a ‘Are you alive? ‘message, send a confirmation message to who send this message.

link-state advertisement implementation.

Use a dictionary called mapdict to store the map information.
Define a function called updatemystate(), this function will update the information which is directly related to the node.
The node also define a function to broadcast its own state, which includes its distance with its alive neighbors. Also, there is a function to broadcast delete message to all its neighbor.
Every time the server receives a update state message or delete message, it will use it to update its own map and forward the information to all its alive neighbors except the node who send the message.
Whenever there is a change in connection which we get by keepalive message. The node will update its state and broadcast it. Also, if a neighbor is defined to disconnect, its neighbor will broadcast the delete message.

Libraries used

Socket: Define a UDP socket to connect different node.
Threading: Manage a new thread
Time: Make the program to wait.
Argparse: Get the config file path
Os: Exit the program