Five in a row game with networking

Game overview:

Two players play the game on an rectangular board. The players agree on who to make the first move, then he/she places a colored stone anywhere on the board. After the players alternately place a stone. The one who manages to place N stones in a row (column or diagonal) wins the game. If the board gets full and nobody managed to have N in the row, the game is tie.

Game config:

- board size, grid style
- number of stones in line a to win
- colors
- connection port, timeout
- Encryption key size
- player names, colors

Extra functionality:

- Background, move, game end music effects. All mutable by pressing m.
- Game timer

API description:

Examples provided in server.py and client.py

```
#imports game module
from fiveinarow.fiveinarow import FiveInaRow

# creates game instance in SERVER mode
fir = FiveInaRow(FiveInaRow.SERVER)

# starts config connection screen
fir.start()

# sets player name and that she moves first and starts game
fir.set_player('Kata', FiveInaRow.FIRSTMOVE)
fir.start_game()
```

After the game ends, prints game over or winner message and a button for playing new game. Both players have to push this button to start the new game.

API internals:

Detailed info for each function is available in the source code

FiveInaRow class:

- Main game initializer and controller
- Configuration loader and validator
- Starting connection
- Setting player name
- Starting game

Game_board module: Grid, Board, Player classes:

- Grid:
    - Responsible for drawing a grid with the given parameters
    - Processing mouse clicks to game coordinates
    - Draws all previous moves of the players
- Board:
    - Storing players’ moves
    - Checking if move is valid (empty space)
    - Checking if player has won
- Player:
    - Players’ info, id
    - Name
    - If player is on move

Communicator class (comm):

- Provides an interface to the communication
- manages encryption initialisation

Encrypted communicator class (encomm):

- Manages encryption and decryption of messages
- Generates encryption keys

Low level communicator class (llcomm):

- Responsible for data transmission
- Connection initialisation in desired mode (server, or client)

PG button and PG text input:

- Pygame extension modules for text input boxes and push buttons

Communication overview:

The communication is divided into separate levels (encryption, llcomm). The encryption is based on a symmetric key algorithm from cryptography.Fernet package. As soon as the client connects generates a random key. In the meanwhile he server then generates an asymmetric key-pair, and sends the public key to the client. The client encrypts the common symmetric key with the received public key and sends back to the server. From than the communication if secure.

Before the start of the game instances request the other players information (name). After a start_game request is performed the game play starts.

If the player on turn makes a valid move, sends it to the partner. The partner also checks if it is a valid move checking his own board.

If a player wins the game updates his stats and broadcasts it to the partner for the scoreboard.

If the player presses the new game button, a request of new game is sent. If this request is received from the partner too, board is cleared and a new match is started.

Refs:

https://www.pygame.org/docs/ref/mixer.html
https://cryptography.io/en/latest/
https://stuvel.eu/rsa
http://pyzmq.readthedocs.io/en/latest/api/zmq.html
https://stackoverflow.com/questions/46390231/how-to-create-a-text-input-box-with-pygame
https://stackoverflow.com/questions/36653519/how-do-i-get-the-size-width-x-height-of-my-py game-window
https://github.com/tigeroses/FIR