Консольное клиент-серверное приложение для игры в «крестики-нолики»

Приложение должно быть построено по архитектурной модели *Model View Controller* и может быть как сервером, так и клиентом — это определяется аргументами командной строки. Примеры запуска:

```
> python xo.py
```
запуск в режиме клиента, адрес сервера по-умолчанию localhost

```
> python xo.py 192.168.1.98
```
запуск в режиме клиента, адрес сервера 192.168.1.87

```
> python xo.py server X
```
запуск в режиме сервера, выбранная масть: `**X**`, адрес localhost

```
> python xo.py server O 192.168.1.98
```
запуск в режиме сервера, выбранная масть: `**O**`, сетевой адрес 192.168.1.98

Обмен данными между клиентом и сервером должен происходить в формате JSON по сетевому протоколу UDP.

1. Структура

Исходный код приложения должен содержать, как минимум, следующие компоненты (классы):

- **`XoApplication`** — реализует логику приложения и игры (предоставлен вам в готовом виде в файле **`app.py`**);
- **`XoModel`** — реализует работу с данными матрицы игрового поля;
- **`XoView`** — ведёт диалог с пользователем в командной строке;
- **`XoServer`** — реализует диалог с другим игроком по сети в режиме сервера;
- **`XoClient`** — реализует диалог с другим игроком по сети в режиме клиента.

2. Класс XoModel

Конструктор принимает в качестве аргументов размер игрового поля и создаёт внутреннее представление этого поля — матрицу указанного размера.
Класс содержит методы, позволяющие:

- проверить, является ли ячейка поля пустой (`is_empty`);
- преобразовать внутреннее представление в красивый текстовый вид с нумерацией строк и колонок (`__str__`);
- записать в указанную ячейку любой символ (`set`);
- получить кортеж из элементов в указанной строке (`row`);
- получить кортеж из элементов в указанной колонке (`col`);
- получить кортеж из элементов главной или побочной диагонали (`diag`).

3. Класс XoView

Конструктор этого класса можно оставить пустым.
Класс содержит методы, позволяющие:

- отобразить любой текст (`show`);
- отобразить сообщение о выигрыше (`show_win`);
- отобразить сообщение о проигрыше (`show_loose`);
- сообщить пользователю какой мастью он играет и принять его очередной ход (`input`).

Примечание: ход пользователя должен быть буквенно-цифровым, где буква означает строку, а цифра — номер колонки (как в игре «морской бой»), например: `**b2**` означает, что масть, которой играет пользователь, нужно поставить во второй строке и второй колонке (т.е. в центре поля в классических «крестиках-ноликах»). Метод должен проверять корректность ввода и запрашивать ход повторно, если ввод не соответствует формату.

4. Класс XoServer

Конструктор класса должен создать серверный UDP сокет и принимат обязательные параметры для этого: имя хоста и номер порта.
Класс содержит методы, позволяющие:

- отправить ответ на запрос масти от клиента (`char_response`);
- отправить клиенту информацию о только что сделанном ходе (`send`);
- дождаться хода клиента и получить информацию о нём, т.е. строку и столбец (`receive`).

5. Класс XoClient

Конструктор класса должен создать клиентский сокет для коммуникации с сервером, и принимает обязательные параметры для этого: имя хоста сервера и номер порта.
Класс содержит методы, позволяющие:

- отправить запрос серверу «какой мастью мне играть?» (`char_request`);
- отправить серверу информацию о только что сделанном ходе (`send`);
- дождаться хода сервера и получить информацию о нём, т.е. строку и столбец (`receive`).

6. Проблема класса XoApplication

В логике класса не учитывается случай ничьи в игре — когда никто не выиграл, а на игровом поле не осталось свободного места. Исправьте эту проблему.