بازی دوز با رابط گرافیکی

1) عنوان پروژه

🎯 بازی دوز (Tic-Tac-Toe) با رابط گرافیکی پایتون

یک بازی دوز ۳×۳ با دو حالت:

- 👥 **دو نفره (Player vs Player)**
- 🤖 **بازی با سیستم (Player vs Computer)**

با یکی از کتابخانه‌های مرسوم پایتون برای UI:

- گزینه‌ی پیشنهادی (ساده و استاندارد): **Tkinter** (پیش‌فرض پایتون، نصب اضافی لازم ندارد) 🧩

2) امکانات نسخه‌ی ساده (MVP)

- ✅ نمایش صفحه بازی ۳×۳ با دکمه‌ها
- ✅ انتخاب حالت بازی در ابتدای اجرا (دو نفره / با سیستم)
- ✅ نمایش نوبت (X یا O)
- ✅ جلوگیری از کلیک روی خانه‌ی پر
- ✅ تشخیص برد و مساوی
- ✅ دکمه‌ی «شروع مجدد»
- ✅ در حالت بازی با سیستم: حرکت سیستم **تصادفی از خانه‌های خالی** (بدون Minimax، برای ساده‌ماندن) 🤖

3) ساختار پروژه (چند فایل + import) 🔗

پروژه در یک پوشه با نام `dooz_simple` سازمان‌دهی می‌شود و شامل چند فایل و دو زیرپوشه‌ی اصلی است:

- فایل `main.py` : نقطه‌ی شروع اجرای برنامه (Entry Point). این فایل فقط برنامه را اجرا می‌کند و UI را بالا می‌آورد.
- فایل `README.md` : توضیحات پروژه، نحوه‌ی اجرا، و معرفی اجزای اصلی.

در پوشه‌ی `game` (منطق و هسته‌ی بازی) این فایل‌ها قرار می‌گیرند:

- فایل `game/__init__.py` : برای تبدیل پوشه‌ی `game` به یک پکیج قابل import.
- فایل `game/logic.py` : شامل کلاس منطق بازی (مدیریت برد، نوبت‌ها، ثبت حرکت، تشخیص برد/مساوی، ریست).
- فایل `game/ai.py` : شامل کلاس سیستم بازی‌کننده (Computer) با الگوریتم ساده (انتخاب تصادفی از خانه‌های خالی).

در پوشه‌ی `ui` (رابط کاربری گرافیکی) این فایل‌ها قرار می‌گیرند:

- فایل `ui/__init__.py` : برای تبدیل پوشه‌ی `ui` به یک پکیج قابل import.
- فایل `ui/app.py` : شامل کلاس رابط کاربری با Tkinter (ساخت پنجره، دکمه‌های برد، نمایش وضعیت بازی، و اتصال رویدادها به منطق بازی).

در این ساختار، فایل `main.py` از کلاس UI استفاده می‌کند و UI نیز برای انجام عملیات بازی، کلاس‌های داخل پوشه‌ی `game` را import می‌کند. ✅

4) شی‌گرایی: کلاس‌ها و مسئولیت‌ها (ساده و کافی)

🧠 منطق بازی (game/logic.py)

- `TicTacToeGame`
  - نگهداری برد ۳×۳ (لیست ۹تایی یا ماتریس)
  - نگهداری بازیکن فعلی (X/O)
  - متدها:
    - `make_move(index)`
    - `check_winner()`
    - `is_draw()`
    - `reset()`

🤖 هوش مصنوعی ساده (game/ai.py)

- `RandomComputerPlayer`

- فقط یک کار انجام می‌دهد:
  - `choose_move(board)` → انتخاب تصادفی از خانه‌های خالی 🎲

🖥️ رابط کاربری (ui/app.py)

- `TicTacToeApp`
  - ساخت پنجره
  - ساخت ۹ دکمه
  - نمایش وضعیت (نوبت/برنده/مساوی)
  - اتصال رویداد کلیک‌ها به منطق بازی
  - اگر حالت «با سیستم» باشد: بعد از حرکت کاربر، حرکت سیستم را اجرا و UI را به‌روزرسانی کند

5) جریان اجرای بازی (Workflow)

۱. برنامه اجرا می‌شود.

۲. یک پنجره کوچک باز می‌شود و کاربر حالت بازی را انتخاب می‌کند:

- 👥 دو نفره
- 🤖 با سیستم

۳. برد نمایش داده می‌شود.

۴. با هر کلیک:

- حرکت ثبت می‌شود
- برد/مساوی بررسی می‌شود
- در صورت ادامه بازی، نوبت تغییر می‌کند
- اگر PvC باشد و نوبت سیستم برسد، سیستم یک حرکت تصادفی می‌زند

۵. در پایان: پیام نتیجه و امکان شروع مجدد.

6) تحویل نهایی (Deliverables)

- ✅ کد چندفایلی طبق ساختار بالا
- ✅ `README.md` شامل:
  - نحوه اجرا (`python main.py`)
  - توضیح کوتاه کلاس‌ها

- توضیح دو حالت بازی

7) الزام پایانی ارائه

در پایان پروژه، **حتماً یک ویدیو از اجرای برنامه با مدت حداکثر ۲ دقیقه ضبط کنید و آپلود کنید**؛ داخل ویدیو نشان دهید:

- انتخاب حالت دو نفره و یک برد سریع
- انتخاب حالت با سیستم و چند حرکت
- دکمه شروع مجدد و نمایش نتیجه 🎥✅