TANK ARENA · FILES & WORKFLOW

檔案關係與流程：從規則策略到天梯

1 寫規則策略

自己電腦／Colab

strategy.py
你寫：if-else 戰術，看得到完整狀態

strategy_lib.py
函數庫：Me、Memory、Behavior、Action

sandbox.py
離線小場地：不用連網站先測試

***.ipynb**
課堂實作、脫困練習

→ 上傳 strategy.py

2 試跑與錄資料

繳交網站「② 策略與錄製」

試跑一局
看重播，確認策略沒問題

錄製服務
讓 strategy.py 打上千局，記下每一步

dataset.zip
每一步：91 個觀測 → 策略的 5 個動作

→ 下載 dataset.zip

3 訓練神經網路

自己電腦／Colab

agent.py
你改：網路設計圖＋訓練設定（MODEL、HIDDEN、LAYERS、EPOCHS…）

train.py
不用改：讀資料、照 agent.py 訓練

model.pt
訓練好的權重（還不能上傳）

→ model.pt ＋ agent.py

4 匯出

自己電腦／Colab

export_model.py
打包網路＋權重，連同 strategy.py、agent.py

scripted_agent.py
比對匯出前後動作完全相同

submit/model.pt
要上傳的就是這個

→ 上傳 submit/model.pt

5 繳交與比賽

繳交網站 → 天梯 → 觀戰網站

網站檢查
格式、推論速度、抄襲比對

天梯
只跑 model.pt，每人 200 場，算分

arena-tv
觀戰：轉播、重播、排行榜

期中報告.pdf
4 個表格，也在網站上傳

strategy.py 裡面：系統每一步呼叫 act()

```python
class Strategy:
    def reset(self):                          # 每局開始：換一本新的筆記本
        self.memory = Memory()
    def act(self, state):                     # 每一步（1/20 秒）：第 3 層 戰術
        me = Me(state, memory=self.memory)
        move = Behavior.keep_distance(me, e, 230) # 第 2 層 行為
        aim = Action.aim_turret_at(me, e)         # 第 1 層 基本動作
        return combine(move, aim, fire)  # → 5 個整數
```

自己寫行為：`@Behavior.custom`（名字不能和內建相同）。要跨步記住的東西放 `me.mem`，並在 **reset()** 給初始值。

誰改什麼

你寫／你改	strategy.py、agent.py、期中報告
不用改	strategy_lib.py、train.py、export_model.py、scripted_agent.py、sandbox.py
網站做	試跑、錄資料、檢查、天梯、轉播

✕ 上傳 train.py 存的 model.pt → 要上傳 export 後的 **submit/model.pt**

✕ 改自己電腦的 strategy_lib.py → 網站用的是原版，要改請寫在 strategy.py 裡

規則策略 vs 模型

規則策略
看**完整狀態**（所有坦克、子彈位置）
只用來**錄示範資料**
比賽時**不執行**

神經網路模型
只看 **91 個觀測**
模仿規則策略的 5 個動作
比賽時**只有它上場**

規則策略做決定的理由，模型不一定看得到：讓示範「看得到的畫面 → 該做的動作」對得起來，模型才學得好。