

פרויקט צוללות (שחקן יחיד)

- לוח: מטריצה בגודל $X \times X$ (ערך X נקבע מראש במשתנה; למתקדמים — קלט מ־CLI/קובץ).
- אתחול לוח: כל התאים מאותחלים ל-0 (או None).
- פרמטרים נוספים:
 - מספר צוללות למקם (למתקדמים — קלט מ־CLI/קובץ).
 - מספר יריות מותר (למתקדמים — קלט מ־CLI/קובץ).
- יש למקם את כל הצוללות באופן אקראי על הלוח. במקום שנמצאת בו צוללת יש להחליף את ה-0 (או NONE ב-1)
- תצוגה למסך (בזמן המשחק):
 - תא שלא ירו עליו: מציגים O.
 - תא שירו עליו ואין בו צוללת: מציגים X.
 - תא שירו עליו ויש בו צוללת: מציגים V.
- תור השחקן (כל עוד נותרו יריות ולא כל הצוללות טובעו):
 - הצגת הלוח - כולל O X V קודמים.
 - בחירת קואורדינטות X,Y
 - לוודא שהקלט בתחום המותר ושטרם נורה על תא זה.
 - ירי.
 - הודעה: פגיעה/פספוס.
 - עדכון הלוח: X לפספוס, V לפגיעה
 - להדפיס סטטוס: מספר יריות שנותרו, מספר צוללות שנותרו.
- סיום:
 - ניצחון: כל הצוללות טובעו לפני שנגמרו היריות.
 - הפסד: נגמרו היריות ועדיין נותרו צוללות; מציגים לוח סופי עם מיקומי הצוללות חשופים (1 היכן שיש צוללת, יחד עם ה-X וה-V וה-O הקודמים).

מבנה תיקיות (אופציונלי)

```
submarines_project/
├── main.py          # נקודת כניסה (CLI)
├── submarines/
│   ├── __init__.py
│   ├── board.py     # יצירה/עדכון/תצוגת לוח
│   ├── placement.py # מיקום צוללות אקראי
│   ├── game.py      # לוגיקת משחק וחוקי עצירה
│   └── io.py         # קלט/פלט (CLI)
└── tests/
    └── test_game.py  # בדיקות יחידה בסיסיות (לא חובה)
```

אפשר כמובן לאחד קבצים או לוותר על תיקיות — זה רק סדר מוצע.

ממשק פונקציות (אופציונלי)

ייצוג נתונים מוצע

כדי לתמוך גם בתצוגה במהלך המשחק וגם בחשיפה בסוף:

- ships: list[list[int]] — (צוללת=1/0) (מטריצה של 0/1).
- shots: list[list[bool]] — (האם נורה על התא) False/True מטריצה של.
- תצוגה נגזרת מחיתוך שני הלוחות:
 - אם $\text{shots}[y][x] == \text{False} \rightarrow O$
 - אם $\text{shots}[y][x] == \text{True}$ ו- $\text{ships}[y][x] == 0 \rightarrow X$
 - אם $\text{shots}[y][x] == \text{True}$ ו- $\text{ships}[y][x] == 1 \rightarrow V$

submarines/[board.py](#)

- `create_matrix(size: int, fill: int = 0) -> list[list[int]]`
ממלא מטרצה בגודל size מחזיר מטרצה מרובעת בגודל fill.
- `create_bool_matrix(size: int, fill: bool = False) -> list[list[bool]]`
- `in_bounds(size: int, x: int, y: int) -> bool`
בדיקת גבולות.
- `count_remaining_ships(ships: list[list[int]], shots: list[list[bool]]) -> int`
כמה תאי צוללת (1) טרם נחשפו בירי.
- `render_public(ships: list[list[int]], shots: list[list[bool]]) -> str`
O/X/V). מחזיר מחרוזת לתצוגת המשחק (עם).
- `render_reveal(ships: list[list[int]], shots: list[list[bool]]) -> str`
(מחזיר מחרוזת לוח סופי עם חשיפת כל הצוללות (1) במיקומים שיש צוללת).

submarines/[placement.py](#)

- `place_random_ships(ships: list[list[int]], n: int) -> None`

submarines/[game.py](#)

`init_game(size: int, n_ships: int, max_shots: int, *, rng: random.Random | None = None) -> dict`

יוצר מצב משחק:

```
{
  "size": int,
  "ships": list[list[int]],
  "shots": list[list[bool]],
  "n_ships": int,
  "max_shots": int,
  "shots_used": int
}
```

- `shoot(state: dict, x: int, y: int) -> tuple[bool, str]`
(is_hit, message). מבצע ירי אם לא נורה קודם ועל הגבולות; מחזיר
(רק בניסיון ירי תקין (לא ירי כפול/מחוץ לגבול shots_used מעדכן).
- `is_won(state: dict) -> bool`
כל תאי ה-1 נורו.

- `is_lost(state: dict) -> bool`
ועדיין יש תאי צוללת לא-פגועים `shots_used >= max_shots`.
- `shots_left(state: dict) -> int`
- `remaining_ships(state: dict) -> int`

submarines/io.py

- `parse_coords(raw: str, *, one_based: bool = True) -> tuple[int, int] | None`
לקואורדינטות, עם אפשרות 1 x y ממיר קלט משתמש `based`.
- `print_status(state: dict) -> None`
יריות שנותרו + צוללות שנותרו + `(render_public)` מדפיס לוח ציבורי.
- `print_end(state: dict, won: bool) -> None`
`(render_reveal)` מדפיס תוצאה; אם הפסד — מציג חשיפה.

main.py

- `play(size: int = 6, n_ships: int = 6, max_shots: int = 10, *, one_based: bool = True) -> None`
לולאת המשחק הראשית:
 1. (אתחול מצב; 2) כל עוד לא ניצחון/הפסד ויש יריות — קלט קואורדינטות, ירי, הודעה.
 2. בסיום — תוצאה וחשיפה אם צריך.
- `if __name__ == "__main__":`
size, n_ships, max_shots, לקריאת `argparse` — לטעון פרמטרים קבועים; למתקדמים `--seed` ואופציונלית.

כללי ולידציה והתנהגות (מומלץ)

- קלט:
 - קואורדינטות בתחום (לפי 0-based/1-based שנקבע).
 - דחיית ירי לתא שכבר נורה — לא מפחיתים יריות במקרה כזה, ומבקשים קלט אחר.
- מיקום צוללות:
 - ללא חפיפה; אם `n_ships > size*size` — להעלות שגיאה.

- ספירת יריות:
 - סופרים רק ירי חוקי (בתחום ועל תא שלא נורה).
 - הודעות:
 - פגיעה/פספוס; סטטוס אחרי כל ירי.
 - סיום:
 - ניצחון מיידי ברגע שכל הצוללות טובעו.
 - הפסד כשנגמרו היריות ועדיין נותרו צוללות; חשיפת 1 בכל מיקום צוללת.
-

תרחישי קצה (מומלץ לבדיקה)

- ירי כפול על אותו תא — לא להפחית יריות; להתריע ולבקש קלט חדש.
- קואורדינטות מחוץ לטווח — דחייה ולא להפחית יריות.
- $n_ships == 0$ (ניצחון מיידי) / $max_shots == 0$ (הפסד מיידי אם יש צוללות).
- לוח קטן ($2 \times 1 / 2 \times 1$) ומספר צוללות קיצוני.
- רנדומליות דטרמיניסטית עם seed לצורך בדיקות.