# Battleship Lite

Krisztina Tesenyi

# Code overview

```python
from tabulate import tabulate

# First a Game map class is created, properties rows, cols also a game_map
# list is initialised which will contain the map. The map will be a state map
# it is filled with zeros as a start, zero means no ship.


class Game_map:
    def __init__(self, rows=10, cols=10) -> None:
        self.rows = rows
        self.cols = cols
        self.game_map = []

    def create_game_map(self):
        self.game_map = [[0]*self.cols for _ in range(self.rows)]

    def print_game_map(self):
        headers = 'ABCDEFGHIJ'
        print(tabulate(self.game_map, headers=headers, tablefmt='fancy_grid',
              showindex=range(1, self.rows + 1)))


def main():
    game_map = Game_map()
    game_map.create_game_map()
    game_map.print_game_map()


if __name__ == "__main__":
    main()
```

Result of the code

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

First, the Game_map class is created. Properties: rows (10), cols (10), also a game_map list is initialised. The list will contain the map, which will be a state map.

- create_game_map method builds up the map, using a combination of value repetition and a for loop. It puts zeros 10 times (which is how many columns are in the map ([0]*self.cols), in all 10 rows (for _ in range(self.rows)) in the 2D list. The map is filled with zeros as a start, zero means nothing (no ship, no shot, no hit) is on a certain coordinate.
- print_game_map method prints out the map using the tabulate package. A header and index are added to the map, and a tablemft format is fancy_gird.
- main function will include code that is relevant for controlling the game. A Game_map object was created called game_map and methods were called to build and print the game_map for test.
- if __name__ == "__main__":
  main()

  This code block allows execution of the file when it runs as a script, but not when it is imported as a module.

# Future development ideas

- Ask user input for the size of the map (rows, cols), they are defined as parameters of the Game_map class, so it can be easily added as an option in the future.