grid.py
print_grid()
add_vessel()
has_overlap()

GRID_WIDTH
GRID_HEIGHT
NUM_VESSELS
B
VESSEL_NAMES[]
VESSEL_SIZES[]

human.py
get_location()
get_choice()

grid_defend[]
gird_attack[]

import grid

ai.py
get_location()
get_choice()

grid_defend[]
grid_attack[]

import grid

battleship.py
main()

Game loop

import human
import ai
import grid
grid.py

• has_overlap(grid, row, column, size, direction)
  – Check direction
  – Single for loop (size), if detects collision, return true
  – If every cell is checked without collision, return false
• add_vessel(grid, row, column, size, direction)
  – Check direction
  – Single for loop (size), modifies grid[row][column]
• print_grid(grid)
  – Two nested for loops

• Does operations to the grid parameter (no global grids)
human.py & ai.py

- **get_location(index)**
  - Depending on human or ai: ask the user or choose random coordinates and direction
  - Row/column validation (row/col are on the grid?)
  - Ship size validation (does ship fit on the grid?)
  - Overlap check (calls `has_overlap` from grid.py)
  - Recursive if any of these checks fail
  - Finally: adds ship to defend grid (calls `add_vessel`
battleship.py

• main()
  – Human and ai position their ships: get_location
  – while loop keeps going until either human’s ships sunk or ai’s ships sunk (two calls to all_ships_sunk)
  – Human or ai get_choice
  – drop_bomb
  – print_grid
human.py & ai.py

• `get_choice()`
  – Depending on human or ai: ask the user or choose a coordinate to drop a bomb
  – Row/column validation (row/col are on the grid?)
  – Check `grid_attack` if we’ve already bombed there
  – Recursive if any of these checks fail
  – Finally: returns the row and column to bomb
  – `return row, column`
grid.py

• drop_bomb(grid_attack, grid_defend, row, column)
  – Checks grid_defend if we hit a ship, prints HIT or MISS
  – Modifies grid_attack[row][column] with bomb placement
  – Modifies grid_defend[row][column] with bomb placement
  – Changes grid cell to ‘o’ for miss, ‘X’ for hit

• all_ships_sunk(grid_defend)
  – Checks the entire grid for ANY presence of a ship (5, 4, 3, or 2)
  – Nested for loop (WIDTH/HEIGHT)
  – If a ship is found, return false
  – If both loops are finished and no ship is found, return true
human.py

ai.py

generate_location
has_overlap
add_vessel(grid_defend)

generate_choice()
check grid_attack

battleship.py

get_location called for human and ai

while(human ships not sunk AND ai ships not sunk)

generate_choice

drop_bomb
print_grid

grid.py

has_overlap
add_vessel grid_defend
print_grid
drop_bomb
all_ships_sunk