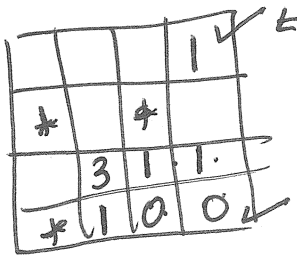
# OOP Python 6/16/26

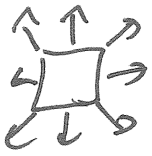
## Minesweeper




r x c board
squares covered
some bombs hidden
user clicks square to uncover
GOAL: To click all non-bomb squares

## problems to solve

#1 computing # of adjacent bombs



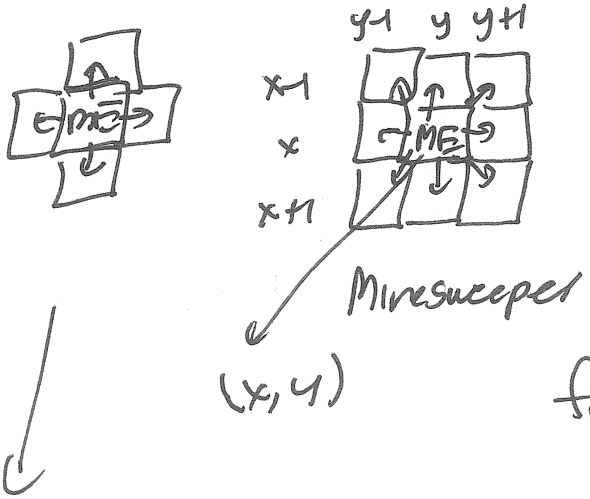
#2 recursive clear

if player clicks on a '0' square, then all adjacent squares are automatically safe, so the game clears those squares + if any those are "0"s it keeps on clearing until no new 0 squares are clears.

recursion is a function that sometimes calls itself

move (r, c)
might call
move (r-1, c) for example.

# DR/DC
# DX/DY arrays




i 0 1 2 3 4 5 6 7

DX = [-1, -1, -1, 0, 0, 1, 1, 1]

DY = [-1, 0, 1, -1, 1, -1, 0, 1]

DX = [-1, 0, 0, 1]

DY = [0, -1, 1, 0]

```
for i in range(len(DX)):
    nX = x + DX[i]
    nY = y + DY[i]
```

| | |
|---|---|
| i = 0 | (x-1, y-1) |
| i = 1 | (x-1, y) |
| i = 2 | (x-1, y+1) |
| i = 3 | (x, y-1) |
| i = 4 | (x, y+1) |
| i = 5 | (x+1, y-1) |
| i = 6 | (x+1, y) |
| i = 7 | (x+1, y+1) |

---

## Problem #2

move (x,y)

```
if (no bombs adj x,y)
    for i in range(len(DX)):
        move (x + DX[i], y + DY[i])
```

Skip previously cleared squares, out of bounds

# 3 Files

minemain.py – Runs the Program
Displays the only screen
Drawing to the surface!

minesweeper.py → mineclass defines the class that has all the game logic.

minefunc.py

DRAWING
converting the backend data stored in the object to the appropriate drawings

## OBJECT

board

| | | | |
|---|---|---|---|
| 1 | 2 | * | 2 |
| 1 | * | 3 | * |
| 1 | 1 | 2 | 1 |
| 0 | 0 | 0 | 0 |

show

| | | | |
|---|---|---|---|
| T | F | F | F |
| F | F | F | F |
| F | F | F | F |
| F | F | F | F |

~~left~~ need | 12 |