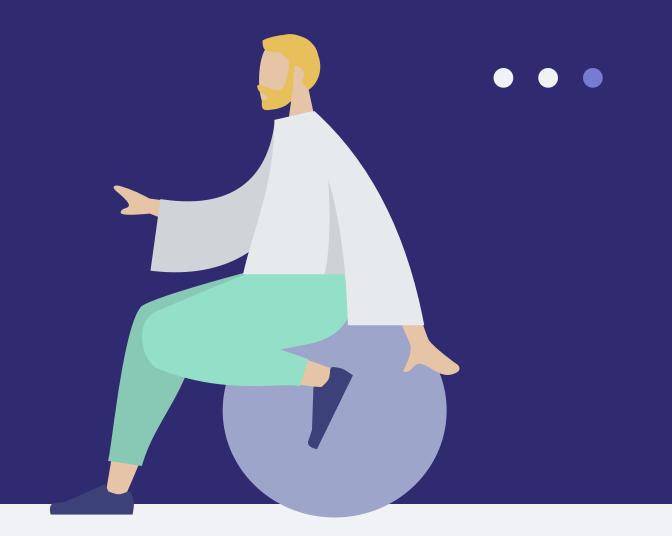


### Rules Recap

How do we play minesweeping game in the childhood?



#### Input

User clicks the left/right of the mouse (input 0/1 in our system), start sweeping!

#### Position + Dig

If there is a mine, GAME OVER!

#### Position + Dig

If there's no mine, then search through the surrounding 8 boxes, if no mine again, search till the non-zero number exist.

#### Position + Place Flag

Place a flag at the input position/



### Video Demo

https://www.bilibili.com/video/BV19h41 1Z73c

# Core Code of Minesweeping

The core code we use to implement the Minesweeping

## Single.py

The main body of minesweeping, including functions for "accepting input", "judging result", "setting board"

```
Single.py X
C: > Users > gzLij > Desktop > Solutions for UP3 > 🕏 Single.py > ...
       from translate board import encrypted board
       # set up the board, randomly placing the mines
  5 > def set_board(r, c, n): ···
  21
       class Solution:
           def __init (self, board, click, numFlags, endGame = False): ...
  28
           def updateBoard(self): ...
  29 >
           def set click(self,new click): ...
  75 >
           def get flag(self): ...
  77 >
           def get_board(self): ...
  79 >
           def get_game_status(self): ...
  81 >
  83
           def play(self): ...
 84 >
 102
       # original settings (Will be commented out in online mode)
 103
       row = 16
       column = 16
      num_of_mines = 16
      Flags = num of mines
       bo = set board(row, column, num of mines)
 109
 110
 111
      a = Solution(bo, [], Flags)
      a.play()
```

# Core Code of Minesweeping



The core code we use to implement the Minesweeping Game

# Single.py

Recursion

```
def dfs(self,i, j):
   direction = ((1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (-1, 1), (-1, -1), (1, -1))
   self.m, self.n = len(self.board), len(self.board[0])
   cnt = 0
   for x, y in direction:
       x, y = x + i, y + j
       if 0 <= x < self.m and 0 <= y < self.n and self.board[x][y] == 'M':
           cnt += 1
   if cnt == 0:
       self.board[i][j] = 'B'
       for x, y in direction:
           x, y = x + i, y + j
           if 0 <= x < self.m and 0 <= y < self.n and self.board[x][y] == 'E':
                self.dfs(x, y)
    else:
        self.board[i][j] = str(cnt)
```

# Core Code of Minesweeping

The core code we use to implement the Minesweeping

### Translate\_board.py

A side-function designed for clients' view. Hide the locations of mines from client by using another set of representation

```
translate_board.py X
C: > Users > gzLij > Desktop > Solutions for UP3 > 🕏 translate_board.py
      import copy
       '''这个file用来加密每个需要发给用户的board,使得雷对于他们不直接可见'''
      # Printing the Minesweeper Layout
      def encrypted board(lst):
           11st = copy.deepcopy(1st)
          string = ''
          for i in range(len(llst)):
              for j in range(len(llst[i])):
                  if llst[i][j] == 'E':
                       llst[i][j] = '| |'
  10
                  elif llst[i][j] == 'M':
 11
 12
                       llst[i][j] = '|_|'
 13
                  elif llst[i][j] == 'e':
 14
                       llst[i][j] = '|F|'
 15
                   elif llst[i][j] =='X':
 16
                       llst[i][j] = '|F|'
 17
                   elif llst[i][j] == 'm':
 18
                       llst[i][j] = '|X|'
                  elif llst[i][j] == 'B':
  19
                       llst[i][j] = '|0|'
  20
 21
  22
                      llst[i][j] = '|'+ llst[i][j]+ '|'
 23
 24
          for n in llst:
 25
               string_col = ''
 26
              for m in n:
 27
                   string col += m
              string += string_col + '\n'
 28
  29
           return string
```

### Chat\_Server.py

Memorize the game information set up in the first round (initiation)

Collect and distribute information back and forward

```
if msg["message"] == 'g':
    row = 16
    column = 16
                                                        intializing
    num of mines = 16
    Flags = 16
                                                        game
   bo = set_board(row, column, num_of_mines)
    self.play once = Solution(bo, Flags, [])
   board translated = encrypted board(self.play once.get board())
    msg["message"] += '\nCurrent Board Looks like this: \n'
    msg["message"] += '\n' + board_translated + '\n'
    msg["message"] += 'next player please input a clicker'
else:
    try:
        clicker = msg["message"].split(',')
                                                        play game
        for i in range(len(clicker)):
            clicker[i] = int(clicker[i])
        a = self.play once
        a.set_click(clicker)
        if a.get_game_status() == False:
           board_translated = encrypted_board(a.updateBoard())
           flags_left = a.get_flag()
           msg["message"] += "\nCurrent Board Looks like this: \n"
           msg["message"] += board_translated
           msg["message"] += "\nNumber of flags remained: " + str(flags_left)
            msg["message"] += "\nGame Over!"
        msg["message"] += " "
```

# Core Code in Connecting

The main changes we made in the chat\_system structure

#### Chat\_Server.py

THE SERVER PLAYS THE GAME!

(instead of the client)

---Same for other online platforms

### Client\_State\_Machine.py

We plan to build our game under the S\_Chatting State, hoping to increase interaction between the players.

The click that the user inputs during the game can also be found in the chat-history.



# Core Code in Connecting

The main changes we made in the chat\_system structure

## Class Knowledge Application



#### OOP

Visualize your competitive advantages using a quadrant for easy scanning.

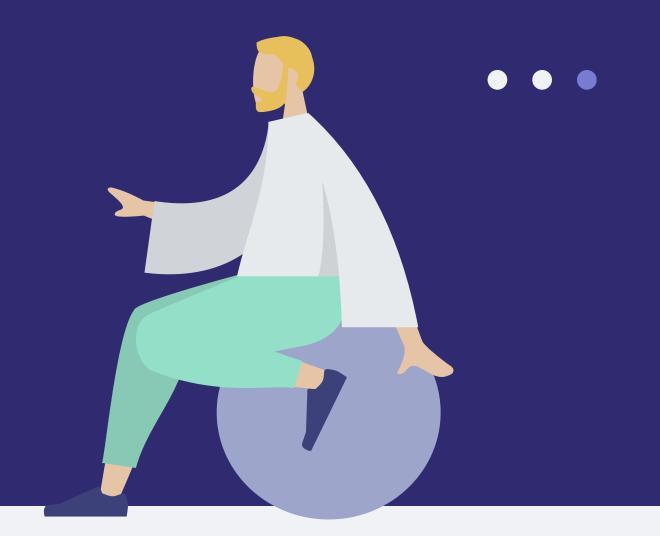
# Utilize Class/ Function from other Python files

Import class from file.py from file.py import \*

#### Recursion - Single.py

Each round when searching for the boundary, we use recursion to ensure finding a box with a number that is not zero.

#### **Problems Encountered**



# Being confused by local/global variables

OOP Nested function, clarifying the scopes

#### Multi-player Display

Manipulate the "msg +=" statement, accept any input, use the "try...Except" statement to avoid invalid input

#### Variable changes

Solvined Using OOP in chat\_server.py

#### "User Interface"

encrypting the message sent to users



### Thank You!

Questions Welcomed!