```
2차원 리스트(1)
                                                                                                                                                                                                                                                               구현(1)
        Bop 4
                             hei_list — 1 5 14 ··· 31
                           수 있는 데이터 다입
                               단일 식별자로 연속된 저장
공간 접근 수단 제공
          2차원 리스트
                                                                                                                                                                                                                                                                                            lass Tic_Tac_Toe
                                                                                                                                                                                                                                                                                              # 제임판 생성
def __init__(self):
self.board = []
                           ↑ 리스트에 리스트가 내포된,
리스트의 리스트
                                                ∰2차원 리스트
                                                    ♂ 리스트에 리스트가 내포된 즉, 리스트의 리스트
2차의
                                           # 첫 플레이어 선택
def select_first_player(self):
if random.randint(0, 1) == 0:
return 'X'
else:
return '0'
        2차원 리스트 구조
                                                                                                                                                                                                                                                                                              # 기호 표시
def mark_spot(self, row, col, player):
self.board[row-1][col-1] = player
                             (翻 2차원 리스트 구조
열면보
                                                                                                                                                                                                                                                                                          Bell Doard(I)(W-I)(Dor-I) = pix

변수 15 win(pelf, player):

n = len(pelf board)

### (Perf in range(n):

win = True

for in range(n):

if self.board(I)(I) != player:

win = False

break

if win == True :

return win
                                           # g Reg
for i in range(n):
win = True
for j in range(n):
if self.board[j[i] != player:
win = False
break
if win == True :
return win
          2차원 리스트 접근
                           return win

# OPE (Table 100

win = Tive

for i in range(i) != player :
    if self loose(i)[i] != player :
    break

if win == Tive :
    if win == Tive :
    if self loose(i)[i] != i - 1] != player :
    break

if win == Tive

break

if win == Tive

return win

retur
        2차원 리스트 명성
                                   2차원 리스트 생성
                                                      구문형식

Bid (필소사원스1),
[일소사원스1],
[일소사원스2],
(필소사원스1),
[일소사원스3])
                                                                                                                                                                                                                                                                                                  # 이외의 경우 승리상황이 아니므로 False
return False
                                                                                                                                                                                                                                                                                              #편의 변한 역부 확인
def is_board_full(self):
for row in self.board:
for item in row:
if item == "":
return False
return True
        2차원 리스트 등적 생성
                                                                                                                                                                                                                                                                                            # 필진이에 변경
def next_player(self, player):
if player == 'O':
return 'X'
else:
return 'O'
                           □ 2차원 리스트 동직 생성
                                   | Insect random size of | O of |
| Insect random size of |
| 3 m (Mg / Pris where, Mg / Principlane) |
| 5 m is in expectations) |
| 6 m is | 1 |
| 7 m is manufactions |
| 8 m is manufaction |
| 9 m is manufaction |
| 10 m is mission |
| 10 m
                                                                                                                                                                                                                                                                                                # return 'X' if player == '0' else '0'
                                                                                                                                                                                                                                                                                              # 전체 개용한 상태 문제
def show_board(self):
for row in self.board:
for item in row:
print(item, end = "")
print()
        2차원 리스트 순회

    ②차원 리스트 순회
    ○ 각치원 순치적으로 순회하기 위한 2개의
반복 구조가 중첩
출발 5시 분수

                                                                                                                                                                                                                                                                                                    # 첫 플레이어 선택
player = self.select_first_player()
                                                                                                                                                                                                                                                                                                  # 개명 루프 시작
while True:
if player == "X":
print("컴퓨터 자래입니다.")
else:
print("사용자 자래입니다.")
                                                                                                                                                                                                                                                                                                      # 현재 게임만 상태 클릭
self.show_board()
                                                                                                                                                                                                                                                                                                  cise:

row, col = list(map(int, input) 선택된 변변의 대체를 함께 int 등으로 인배팅, list로 제한 pint(li, 1837 형 " + sti(cow) + ", 명" + sti(col) + "한 반짝합니다.")

pint(l)
                                                                                                                                                                                                                                                                                                    # row, col 값이 각각 0이면 개임 무프 중요
if row == 0 and col == 0 :
break
                                                                                                                                                                                                                                                                                                    # 압력된 위치 표시
self.mark_spot(row, col -1, player)
self.show_board()
                                                                                                                                                                                                                                                                                                    표 현재 출경에어가 어전는지 확인
if self.is_win(player) == True:
if player == 'X':
print("명류다가 어떤습니다. 다시 도전에세요.")
else:
print ("사용자가 어떤습니다. 국예합니다.")
break
                                                                                                                                                                                                                                                                                                    # 개명한 가득당 확인, 년한 여부 확인
if self.is_board_full() == True:
print("무승부입니다. 다시 도전하세요.")
break
                                                                                                                                                                                                                                                                                                  # 웰레이어 변경
player = self.next_player(player)
                                                                                                                                                                                                                                                                                              # 최용 게임판 클릭
print()
self.show_board()
                                                                                                                                                                                                                                                                                 #계명 생성
TTT = Tic_Tac_Toe()
                                                                                                                                                                                                                                                                                    #계임 시작
TTT.start()
```

```
# GPT o3-mini-high
import random
Class Tio_Tac_Toe:
# 개편한 영화 포크 설명
def_indt_(sedf):
self board = [] # 개편만 (2자용 리스트)
self_players = ["X", "O"] # 휴데에 가요 (X 사용자, O: 명류터)
self_current_player = None # 한국 문화에
self_current_player = None # 한국 문화에
self_current_player = None # 한국 문화에
    self.create_board() # **GOL #**(3x3 보드, 번만은 ***호 표시)
def create_board(self):
self.board = [[*** for _ in range(3)] for _ in range(3)]
    # 첫 콜레이어 선역 (유작회교 선역)
def select_first_player(self):
self.current_player = random.choice(self.players)
print("선용 클레이아.", self.current_player)
    # ਸਭਲ ਗੁਸ਼ਕ ਭੁਗਰਲਤ ਸ਼ਹ ਸ਼ਮ (ਅਧ ਸਭਸ਼ ਸੁਸ਼ਕ ਸ਼ਹਿਤ ਦਾ ਦੇ
def mark spoticeff. row, col. physyr;
if self board(row)[col] == ***
self board(row)[col] = player
return True
else:
              rse:
print("해당 위치는 이미 사용 중입니다. 다시 선택해주세요.")
return False
    # 송리 상략 확인 (명, 명, 두 덕작선)
def is, win(self, player):
# 명 확인
for row in self.board:
if all(spot == player for spot in row):
return True
        # 열 확인
for cot in range(3):
if all(self.board[row][cot] == player for row in range(3)):
return True
        # 대격선 확인 (좌성→우하)
if all(self.board[i][i] == player for i in range(3)):
return True
         return False
     # 클래에에 전경 (단적 클래에에가 X에진 O, O에면 X로 변경)
def next_player(self, player):
if player == self, players[0]:
return self.players[1]
else:
return self.players[0]
    # 현재 개성한 상태 출력
def show_board(self):
for row in self.board:
print(" | ".join(row))
print() # 울탁공
    # 개명 무료 시작
def start(self):
# # 개명한 생성
self.create_board()
# 첫 콜레이어 선택
self.select_first_player()
          while True:
print(f'현재 필리이어: {self.current_player}*)
self.show board()
           # 합력된 위치가 유요한지 체크
if row < 0 or row >= 3 or col < 0 or col >= 3:
print("흥분은 위치입니다. 다시 영역해주세요.")
              # 입력된 위치에 가요 표시
if not self.mark_spot(row, col, self.current_player):
continue # 이미 채워진 건이면 다시 입력 반응
            # 전략 클레이어 송리 역부 확인
if self is_win(self.current_player):
self.show_board()
print(f*(self.current_player) 클레이어가 송리됐습니
break
           # ᡓরণণ তর
self.current_player = self.next_player(self.current_player)
# 제임 명성 및 시작
TTT = Tic_Tac_Toe()
TTT.start()
```