

# 파이썬 게임 Tic Tac Toe 제작 실습

Cursor AI 단계별 실습

# 1단계: 텍스트 기반 기본 게임

## 질의문:

"텍스트 기반의 2인용 Tic Tac Toe 게임을 파이썬으로 만들어줘.

보드는 콘솔에 출력되고, 사용자는 좌표를 입력해 X와 O를 번갈아 두도록 해줘."

텍스트 기반의 2인용 Tic Tac Toe 게임을 파이썬으로 만들어드리겠습니다. 콘솔에 보드를 출력하고, 사용자가 좌표를 입력해서 X와 O를 번갈아 두는 게임입니다.

tic\_tac\_toe.py +105 ✓

완벽한 Tic Tac Toe 게임을 만들었습니다! 이 게임의 주요 특징은 다음과 같습니다:

### ◆◆ 게임 특징

1. **아름다운 보드 출력:** 유니코드 문자를 사용해 깔끔한 3x3 보드를 표시
2. **좌표 입력 시스템:** 행과 열 번호(1-3)를 입력해서 위치 선택
3. **2인 플레이:** X와 O가 번갈아가며 수를 둠
4. **승리 조건 확인:** 가로, 세로, 대각선 3개 연속 확인
5. **무승부 처리:** 보드가 가득 찰 경우 무승부 판정
6. **게임 재시작:** 게임 종료 후 다시 플레이 가능

플레이어 X의 차례입니다.

행 번호를 입력하세요 (1-3): 1

열 번호를 입력하세요 (1-3): 2

	1	2	3
1		X	
2			
3			

## 2단계: 사용자 vs AI 컴퓨터

### 질의문:

"기존 텍스트 기반 Tic Tac Toe 게임을 수정해서, 사용자가 X, 컴퓨터가 O를 두도록 바꿔줘. 컴퓨터는 랜덤으로 빈 칸에 수를 두도록 해줘."

완벽하게 수정되었습니다! 이제 게임이 사용자(X) vs 컴퓨터(O) 모드로 변경되었습니다.

### ◆◆ 주요 변경사항

1. 플레이어 변경: 2인용 → 사용자(X) vs 컴퓨터(O)
2. 컴퓨터 AI 추가: 랜덤으로 빈 칸을 선택해서 수를 둠
3. 게임 흐름: 사용자 → 컴퓨터 → 사용자 → 컴퓨터 순서로 진행
4. 새로운 함수들:
  - `get_empty_cells()`: 빈 칸들의 좌표를 찾음
  - `computer_move()`: 컴퓨터가 랜덤으로 수를 둠

### 🎯 게임 진행 방식

1. 사용자가 X를 놓을 위치 선택 (1-3 행/열 입력)
2. 컴퓨터가 자동으로 O를 랜덤 위치에 배치

올바른 숫자를 입력해주세요.

행 번호를 입력하세요 (1-3): 1

열 번호를 입력하세요 (1-3): 1

	1	2	3
1	0	x	
2			
3			

플레이어 x의 차례입니다.

행 번호를 입력하세요 (1-3):

### 3단계: AI 전략 개선 (간단한 룰 적용)

#### 질의문:

"컴퓨터가 랜덤이 아니라 간단한 전략을 사용하도록 개선해줘.

예를 들어, 이길 수 있는 자리가 있으면 그곳을 두고, 막아야 할 자리가 있으면 막도록 해줘."

컴퓨터가 랜덤이 아닌 간단한 전략을 사용하도록 개선하겠습니다. 이길 수 있는 자리가 있으면 그곳을 두고, 상대방이 이길 수 있는 자리가 있으면 막도록 구현하겠습니다.

tic\_tac\_toe.py +41 -6 ✓

완벽하게 개선되었습니다! 이제 컴퓨터가 랜덤이 아닌 전략적으로 수를 두게 됩니다.

#### 💡 컴퓨터 AI 전략 (우선순위 순)

##### 1. 승리 기회 포착 🎯

- 이길 수 있는 자리가 있으면 즉시 그곳에 수를 둠
- 3개 연속을 만들 수 있는 마지막 자리 확인

##### 2. 방어 전략 🛡️

- 상대방이 이길 수 있는 자리를 미리 막음

컴퓨터 (O)의 차례입니다...

컴퓨터가 (3, 1)에 O를 놓아 승리했습니다!

	1	2	3
1	X	X	O
2	X	O	
3	O		

😄 컴퓨터 (O)가 승리했습니다! 😄

게임을 다시 하시겠습니까? (y/n):

## 4단계: 게임 승패/무승부 판정 강화

### 질의문:

“게임이 끝났을 때, 승리자 또는 무승부 여부를 출력하도록 코드를 개선해줘.  
게임이 끝나면 다시 시작할 수 있도록 해줘.”

완벽하게 개선되었습니다! 이제 게임이 훨씬 더 체계적이고 재미 있어졌습니다.

### 💡💡 주요 개선사항

#### 1. 게임 구조 개선 💡💡

- `play_game()` 함수로 한 게임 진행 로직 분리
- `main()` 함수에서 게임 통계 관리 및 재시작 처리

#### 2. 상세한 결과 출력 💡💡

- 게임 종료 시: 승리자 또는 무승부 명확하게 표시
- 매 게임 후: 현재까지의 통계 출력
  - 사용자 승리 횟수
  - 컴퓨터 승리 횟수
  - 무승부 횟수

1		X		X		O	
2		X		O			
3		O					

😄 컴퓨터 (O)가 승리했습니다! 😄

📊 현재까지의 게임 결과:

사용자 (X) 승리: 0회

컴퓨터 (O) 승리: 1회

무승부: 0회

총 게임 수: 1회

게임을 다시 하시겠습니까? (y/n):



## 5단계: 코드 구조화 (함수 분리)

### 질의문:

“현재 코드를 함수 단위로 구조화해줘.

예: `print_board()`, `check_winner()`, `player_move()`, `computer_move()` 등으로 나눠줘.”

현재 코드를 함수 단위로 구조화해줘.

예: `print_board()`, `check_winner()`, `player_move()`, `computer_move()` 등으로 나눠줘.

- `display_statistics()` - 현재 통계 출력
- `display_final_results()` - 최종 결과 출력
- `ask_play_again()` - 재시작 여부 확인

### 6. 메인 함수

- `main()` - 전체 게임 흐름 관리

### 🎯 구조화의 장점

1. **가독성 향상**: 각 함수가 하나의 명확한 역할만 담당
2. **유지보수성**: 특정 기능 수정 시 해당 함수만 수정하면 됨
3. **재사용성**: 함수들을 다른 곳에서도 활용 가능
4. **테스트 용이성**: 각 함수를 독립적으로 테스트 가능
5. **코드 중복 제거**: 반복되는 로직을 함수로 분리

이제 코드가 훨씬 더 전문적이고 체계적으로 구성되었습니다!



```
def print_board(board):
    """게임 보드를 콘솔에 출력합니다."""
    print("\n")
    print("    1    2    3")
    print("  _____")
    for i in range(3):
        print(f" {i+1} | {board[i][0]} | {board[i][1]} | {board[i][2]} |")
        if i < 2:
            print("  _____")
    print("  _____")
    print()

def check_winner(board, player):
    """승리 조건을 확인합니다."""
    # 가로 확인
    for row in board:
        if all(cell == player for cell in row):
            return True

    # 세로 확인
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True

    # 대각선 확인
    if all(board[i][i] == player for i in range(3)):
        return True
```

## 6단계: 클래스 기반 리팩토링

### 질의문:

“지금까지 만든 Tic Tac Toe 게임을 클래스(TicTacToe)로 리팩토링해줘.

메서드로 보드 출력, 수 두기, 승패 판정 등을 구현하고 play() 메서드로 게임이 진행되도록 해줘.”

완벽하게 클래스 기반으로 리팩토링되었습니다! 이제 TicTacToe 클래스로 모든 기능이 체계적으로 구성되었습니다.

### ◆◆ 클래스 구조

#### TicTacToe 클래스의 주요 구성

##### 1. 초기화 (\_\_init\_\_)

- 게임 보드 초기화
- 승리/패배/무승부 통계 변수 초기화

##### 2. 보드 관련 메서드

- print\_board() - 보드 출력
- is\_board\_full() - 보드 가득참 확인
- get\_empty\_cells() - 빈 칸 좌표 반환

##### 3. 게임 로직 메서드

```
import random

class TicTacToe:
    def __init__(self):
        """TicTacToe 게임을 초기화합니다."""
        self.board = [[" " for _ in range(3)] for _ in range(3)]
        self.user_wins = 0
        self.computer_wins = 0
        self.draws = 0

    def print_board(self):
        """게임 보드를 콘솔에 출력합니다."""
        print("\n")
        print("    1    2    3")
        print("  _____")
        for i in range(3):
            print(f" {i+1} | {self.board[i][0]} | {self.board[i][1]} |")
            if i < 2:
                print("  _____")
        print()

    def check_winner(self, player):
        """승리 조건을 확인합니다."""
        # 가로 확인
        for row in self.board:
            if all(cell == player for cell in row):
                return True
```

## 7단계: 간단한 GUI (Tkinter)

### 질문:

“지금까지 만든 Tic Tac Toe를 Tkinter GUI로 만들어줘.

3x3 버튼으로 보드를 표시하고, 사용자가 클릭하면 X가 표시되고, 이후 컴퓨터가 O를 두도록 해줘.”

지금까지 만든 Tic Tac Toe 게임을 Tkinter GUI로 만들어드리겠습니다. 3x3 버튼으로 보드를 표시하고, 사용자 클릭 시 X가 표시되고 컴퓨터가 O를 두도록 구현하겠습니다.

tic\_tac\_toe\_gui.py +245 ✓

완벽한 Tkinter GUI 버전의 Tic Tac Toe 게임을 만들었습니다!



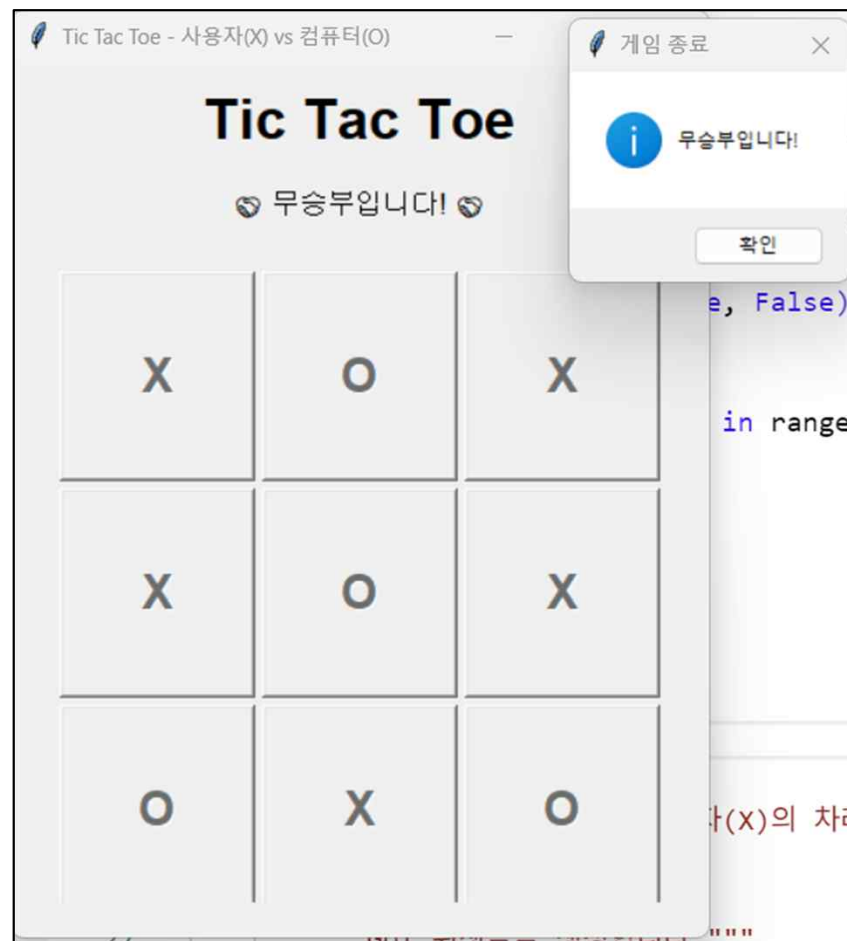
### GUI 주요 특징

#### 1. 시각적 구성

- 3x3 버튼 그리드: 클릭 가능한 9개 버튼으로 게임 보드 구성
- 색상 구분: 사용자 X는 파란색, 컴퓨터 O는 빨간색
- 깔끔한 레이아웃: 400x500 크기의 고정된 창

#### 2. 게임 진행

- 사용자 차례: 빈 칸을 클릭하면 X가 표시됨





## 8단계: GUI 개선 - 게임 종료 메시지

### 질의문:

"Tkinter GUI 버전에 게임 종료 시 팝업 메시지를 띄우도록 해줘.

예: '사용자 승리', '컴퓨터 승리', '무승부' 등 메시지를 보여주고 게임을 다시 시작할 수 있도록 해줘."

완벽하게 개선되었습니다! 이제 Tkinter GUI 버전에 게임 종료 시 상세한 팝업 메시지와 재시작 기능이 추가되었습니다.

### ◆◆ 새로 추가된 기능

#### 1. 향상된 게임 종료 팝업

- 사용자 승리: "🎉 축하합니다! 사용자(X)가 승리했습니다!"
- 컴퓨터 승리: "😞 아쉽습니다! 컴퓨터(O)가 승리했습니다!"
- 무승부: "🤝 이번 게임은 무승부입니다!"

#### 2. 스마트한 재시작 시스템

- 즉시 재시작: 팝업에서 "예"를 선택하면 새 게임 자동 시작
- 게임 종료: "아니오"를 선택하면 게임 보드 비활성화
- 수동 재시작: "새 게임" 버튼으로 언제든지 재시작 가능

#### 3. 개선된 사용자 경험



## 9단계: GUI 디자인 개선

### 질의문:

"Tkinter GUI에 색상과 버튼 크기를 조정해서 보기 좋게 디자인해줘.  
X는 파란색, O는 빨간색으로 표시해줘."

완벽하게 디자인이 개선되었습니다! 💎💎💎 이제 Tkinter GUI가 훨씬 더 보기 좋고 전문적으로 보입니다.

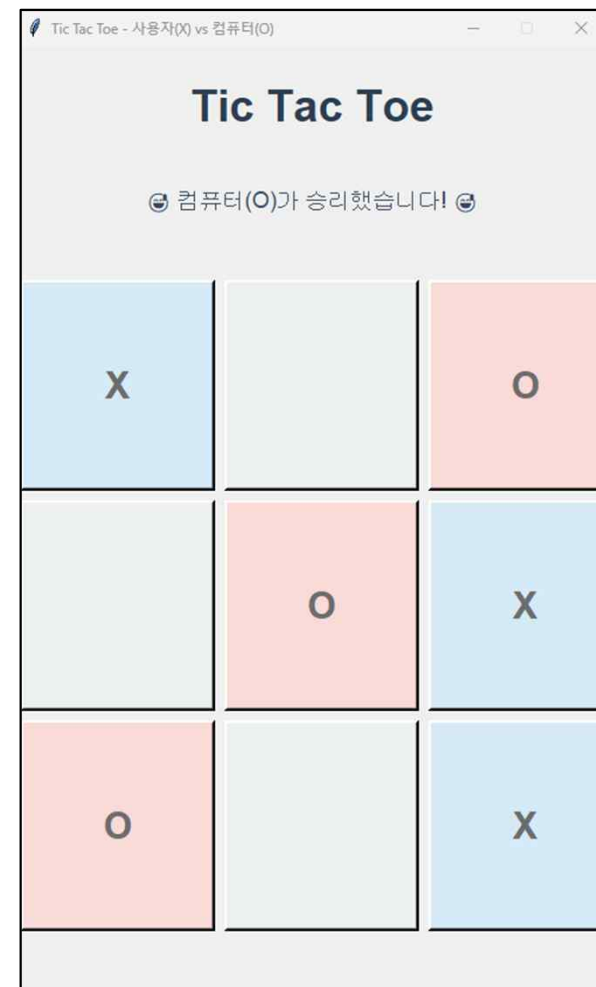
### 💎💎 디자인 개선사항

#### 1. 색상 테마 시스템

- 배경색: 밝은 회색 (#f0f0f0) - 눈에 편안함
- 제목 색상: 진한 파란색 (#2c3e50) - 전문적인 느낌
- X 색상: 파란색 (#3498db) - 사용자 표시
- O 색상: 빨간색 (#e74c3c) - 컴퓨터 표시
- 상태 메시지: 진한 회색 (#34495e) - 가독성 향상

#### 2. 버튼 디자인 개선

- 크기: 8x4로 확대하여 클릭하기 편함
- 폰트: 24pt 굵은 글씨로 명확한 가독성
- 테두리: 3px 두께의 입체적 테두리



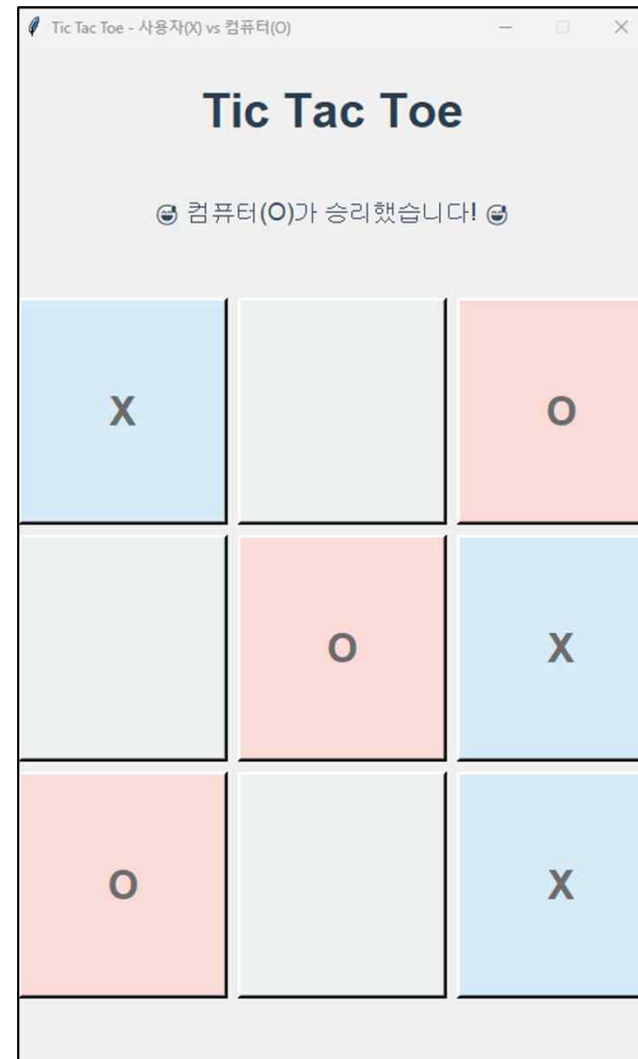
# 게임 화면 하단이 잘려서 나올 경우 추가 질의:

게임 화면 하단이 잘려서 나와. 소스 수정해줘.

<변경전>



<변경후>



# 10단계: 고급 AI (Minimax 알고리즘)

## 질의문:

"GUI 버전에 적용할 수 있도록 Minimax 알고리즘을 사용한 AI를 만들어줘.  
컴퓨터가 최적의 수를 두도록 해줘."

완벽하게 Minimax 알고리즘이 적용되었습니다! 이제 컴퓨터는 절대 지지 않는 완벽한 AI가 되었습니다.

## 💡💡 Minimax 알고리즘 적용 완료

### 1. 핵심 알고리즘

- Minimax: 모든 가능한 게임 상황을 탐색하여 최적의 수를 찾음
- Alpha-Beta 가지치기: 불필요한 탐색을 줄여 성능 최적화
- 깊이 제한: 최대 8단계까지 탐색하여 계산 시간 제한

### 2. 점수 시스템

- 컴퓨터 승리:  $10 - \text{depth}$  (빠른 승리일수록 높은 점수)
- 사용자 승리:  $\text{depth} - 10$  (늦은 패배일수록 높은 점수)
- 무승부:  $0$  (중립적)

### 3. 최적화 전략



# 11단계: 질의

## Minimax 알고리즘이 뭐야?

Minimax 알고리즘이 뭐야?

### 🎯 Minimax 알고리즘이란?

Minimax는 게임 이론에서 사용되는 의사결정 알고리즘입니다. 특히 **제로섬 게임**(한 명이 이기면 다른 한 명이 지는 게임)에서 최적의 전략을 찾는 데 사용됩니다.

### 💡💡 핵심 개념

#### 1. 기본 원리

- **Maximizer**: 컴퓨터(O) - 최대한 높은 점수를 얻으려 함
- **Minimizer**: 사용자(X) - 최대한 낮은 점수를 만들려 함
- **게임 트리**: 모든 가능한 게임 상황을 나무 구조로 표현

#### 2. 작동 방식

≡ text

컴퓨터 차례 (Maximizer) → 가장 높은 점수 선택  
↓  
사용자 차례 (Minimizer) → 가장 낮은 점수 선택  
↓  
컴퓨터 차례 (Maximizer) → 가장 높은 점수 선택  
↓  
... (게임 끝까지 반복)

### 🎮 Tic Tac Toe에서의 예시

상황: 컴퓨터가 수를 둘 차례

≡ text

보드 상태:

```
x | o |  
-----  
  | x |  
-----  
  |  | o
```

### Minimax 알고리즘 실행 과정

1. 빈 칸들 확인: (0,2), (1,0), (1,2), (2,0), (2,1)

2. 각 빈 칸에 대해 점수 계산:

- (0,2)에 O를 놓아보기 → 점수 계산
- (1,0)에 O를 놓아보기 → 점수 계산
- (1,2)에 O를 놓아보기 → 점수 계산
- ...

3. 가장 높은 점수 선택: 컴퓨터는 최고 점수를 가진 수를 선택



# Thank You!!

