**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Icon

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC: TRÍ TUỆ NHÂN TẠO**

**Đề bài: GAME PLAYING**

Mã học phần: INT3401\_20

Giảng viên hướng dẫn: TS.Trần Hồng Việt

Thành viên nhóm:

|  |  |
| --- | --- |
| Lê Văn Đức | 21021291 |
| Phạm Tiến Đạt | 21021285 |

**MỤC LỤC**

[**I.** **Mở đầu** 2](#_Toc167909836)

[1. Giới thiệu Game playing 2](#_Toc167909837)

[**II.** **Thuật toán Minimax** 2](#_Toc167909838)

[1. Khái niệm thuật toán Minimax 2](#_Toc167909839)

[2. Cách hoạt động của thuật toán Minimax 3](#_Toc167909840)

[3. Ưu điểm 3](#_Toc167909841)

[4. Nhược điểm 3](#_Toc167909842)

[5. Ví dụ về thuật toán 3](#_Toc167909843)

[6. Nhận xét 4](#_Toc167909844)

[**III.** **Thuật toán Anpha- Beta pruning** 5](#_Toc167909845)

[1. Khái niệm 5](#_Toc167909846)

[2. Mục đích 5](#_Toc167909847)

[3. Cách thức hoạt động 5](#_Toc167909848)

[4. Ưu điểm 6](#_Toc167909849)

[5. Nhược điểm 6](#_Toc167909850)

[6. Ví dụ thuật toán 6](#_Toc167909851)

[7. Di chuyển thứ tự trong anpha- beta pruning 9](#_Toc167909852)

[8. Các quy tắc để tìm thứ tự tốt 10](#_Toc167909853)

[**IV.** **Ứng dụng của thuật toán trong game playing** 10](#_Toc167909854)

[**V.** **Demo ứng dụng** 10](#_Toc167909855)

[1. Chương trình code tictactoe cho yêu cầu ma trận 9x9 mà chỉ cần 5 quân thẳng hàng là thắng 10](#_Toc167909856)

[2. Chương trình code XO số quân bằng số kích thước ma trận thì thắng 12](#_Toc167909857)

1. **Mở đầu**

Game playing là một nhánh quan trọng trong Trí tuệ nhân tạo (Artificial Intelligence) tập trung vào việc phát triển các thuật toán quan trọng như A\*, Minimax hay Anpha- Beta pruning để tạo ra các trò chơi từ cơ bản đến nâng cao.

Game playing có lẽ có ứng dụng khá thực tế và dễ nhận ra như: tạo ra các game hay các nhân vật trong game.

Ngoài ra khi mở rộng vấn đề, giả sử ta kết hợp game playing với học máy có giám sát( Supervised learning) thì có thể tạo ra các chương trình thông minh với độ chính xác cao.

Do vậy tỉ lệ khi chơi game với máy có thể tăng tính cạnh tranh cao hơn.

1. Giới thiệu Game playing

* Khái niệm: Game playing được định nghĩa chính thức là một loại tìm kiếm với các thành phần như sau:

+, Trạng thái ban đầu: là một tập hợp toán tử xác định các nước đi mà người chơi có thể thực hiện .

+, Trạng thái kết thúc: Được xác định khi trò chơi kết thúc

+, Một hàm tiện ích hay hoàn trả: đưa ra một giá trị số cho kết quả một trò chơi

1. **Thuật toán Minimax**
2. Khái niệm thuật toán Minimax

* Thuật toán Minimax hay còn gọi là Minmax là một phương pháp được sử dụng trong lý thuyết quyết định với mục đích:

+, Tối thiếu hóa vật chất

+, Tối đa hóa lợi ích

* Đây là một thuật toán được xác định giá trị max và quyết định nước đi tốt nhất cho bài toán cần giải quyết
* Tuy nhiên, với mỗi bài toán khác nhau mà yêu cầu và đích của bài toán là khác nhau. Có bài toán yêu cầu giá trị nhỏ nhất, có bài toán lại yêu cầu giá trị lớn nhất. Do vậy , tùy vào yêu cầu bài toán mà tìm phương án giải quyết phù hợp nhất

1. Cách hoạt động của thuật toán Minimax

* Bước 1: xây dựng cây trò chơi . Xây dựng cây trò chơi để biểu diễn tất cả các trạng thái có thể xảy ra trong trò chơi , mỗi nút đại diện cho một trạng thái có thể xảy ra. Tùy vào từng bài toán mà số lượng nút và lá là khác nhau . Cần xác định đầy đủ trạng thái của trò chơi để tránh xảy ra sai sót trong quá trình tối ưu và tìm kiếm nước đi
* Bước 2: gán giá trị cho nút. Giá trị được gán cho mỗi nút để thể hiện vị trí của người chơi hiện tại. Max là giá trị lớn nhất trong số các giá trị của nút . Min là giá trị nhỏ nhất trong các giá trị của nút. Việc xác định và định trị giá trị cho các nút hay nhánh lá là vô cùng quan trọng vì nó có thể ảnh hưởng trực tiếp của bài toán. Chỉ một sai sót nhỏ tronng quá trình định trị cũng có thể làm thuật toán dẫn đến kết quả sai.
* Bước 3: tìm kiếm tối ưu. Dựa theo thuật toán tìm ra giá trị tốt nhất hoặc xấu nhất cho bài toán. Xét xem phương án đó tối ưu nhất chưa. Nếu có sai sót xem lại các bước trên.

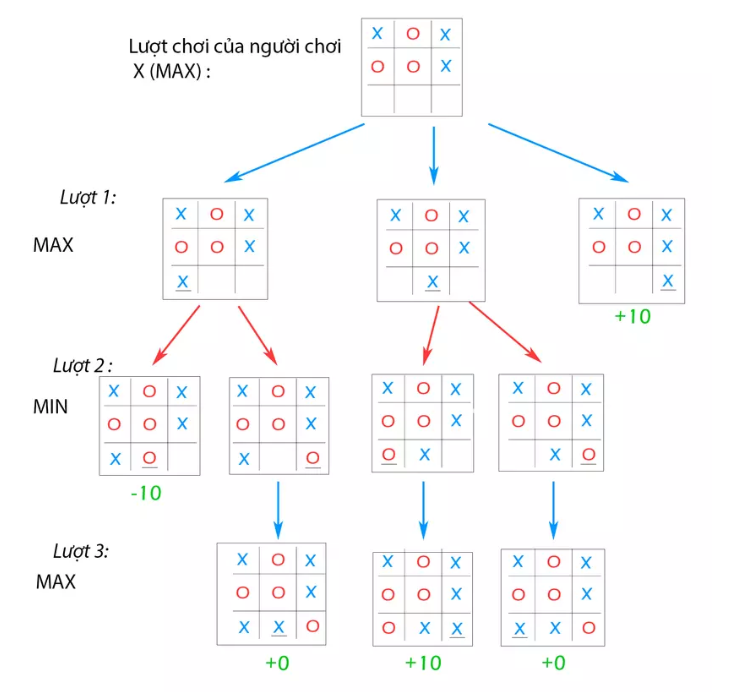
1. Ưu điểm

* Đơn giản, dễ hiểu và khá dễ sử dụng vì chỉ cần xét giá trị của các lá và gán cho các nút tương ứng với các giá trị tìm được
* Hiệu quả với các trò chơi có số lượng không gian trạng thái hạn chế

1. Nhược điểm

* Tốn thời gian và bộ nhớ cho các trò chơi có số lượng không gian trạng thái lớn . Trong một số trường hợp , nếu số lượng không gian trạng thái quá lớn có thể gây tràn bộ nhớ và thuật toán có thể bị treo và không dùng được
* Không hiệu quả với các trò chơi có thông tin không đầy đủ hoặc yếu tố ngẫu nhiên

1. Ví dụ về thuật toán



Như hình trên ta thấy là trạng thái hiện tại của game đang đến lượt đánh của người chơi X đại diện cho MAX. Ta tạm quy định giá trị MAX lúc game thắng cho X = +10 và MIN lúc game thua cho X = -10 và lúc game hòa = 0. Lúc này ở lượt 1, MAX có thể đi được 1 trong 3 nước như hình. Vậy làm sao để chọn 1 trong 3 nước đó nước nào là tốt nhất để đi. Chúng ta dựa vào giá trị của từng nước để chọn nước tốt nhất, như ở đây 3 node đó thuộc lớp MAX nên chọn giá trị lớn nhất. Chúng ta bắt đầu tìm giá trị của từng node đó. Ở lớp MAX trong lượt 1, thì ta có node 1,2,3 được đánh số từ trái sáng phải như hình. Node 3 chúng ta đã là node lá (X win game ) và có giá trị là +10. Còn 2 node 1,2 thì chưa biết giá trị của nó tại lượt 1 nên chúng ta dựa vào giá trị của các node con để định giá trị và bằng giá trị bé nhất của các node con ở lớp MIN tại lượt 2. Cứ tiếp tục tương tự như vậy đến lúc gặp node lá thì từ node lá đó ta suy ngược lại và ta tính được node 1 có giá trị là -10 và node 2 là 0. Vậy nước đi tốt nhất ở đây là như node 3 có giá trị lớn nhất là +10. 5

1. Nhận xét

* Minimax là một thuật toán đệ quy (recursion) giúp lựa chọn và tìm kiếm bước đi dẫn đến trạng thái tốt nhất cho người chơi hoặc tệ nhất cho đối thủ
* Vì nso một thuật toán đệ quy nên tính ứng dụng của nó khá cao. Hơn nữa, việc sử dụng thuật toán cũng khá đẽ dàng và dễ sử dụng
* Có một lưu ý là khi xây dựng công thức hàm tuyến tính, trước tiên ta phải chọn các tính năng , sau đó điều chỉnh trọng số cho đến khi chương trình chạy tốt
* Tiếp đó có thể tự động hóa bằng cách cho chương trình chơi nhiều trò chơi với chính nó nhưng hiện tại không ai có ý tưởng hay về cách tự động chọn các tính năng tốt. Cách tiếp cận đơn giản nhất để kiểm soát lượng tìm kiếm là dặt đôj sâu cố định
* Có thể đặt độ sâu giới hạn sao cho thử nghiệm giới hạn thành công đối với tất cả các nút ở độ sâu hoặc dưới độ sâu d. Độ sâu được chonnj rằng lượng thời gian sử dụng sẽ không vượt quá quy định của trò chơi cho phép. Thêm chút nữa , cách tiếp cận mạnh mẽ là áp dụng phương pháp đào sâu lặp đi lặp lại.

1. **Thuật toán Anpha- Beta pruning**
2. Khái niệm

* Thuật toán Anpha- Beta pruning là một thuật toán cắt tỉa được sử dụng để tối ưu hóa thuật toán Minimax trong các trò chơi hai người

1. Mục đích

* Giảm thiểu số lượng nút cần khám phá trong cây trò chơi , do đó tiết kiệm thời gian và bộ nhớ.
* Tìm ra nước đi tốt nhất và tối ưu chi phí đến mức tối đa

1. Cách thức hoạt động

* Giới thiệu hai giá trị:

+, Anpha: giới hạn tối đa có thể đạt được cho người chơi hiện tại ở một nhánh cụ thể của cây trò chơi

+, Beta: giới hạn tối thiểu có thể đạt được cho người chơi đối thủ ở một nhánh cụ thể

* Quá trình cắt tỉa:

+, Đối với nút max: Nếu giá trị của một nút vượt quá giá trị của anpha hiện tại , cắt tỉa toàn bộ nhánh con của nút đó . Cập nhập giá trị anpha thành giá trị lớn nhất của nút con đã khám phá

+, Đối với nút min: Nếu giá trị của một nút con nhỏ hơn giá trị của hiện tại , tỉa toàn bộ nhánh con của nút đó. Cập nhập giá trị beta thành giá trị nhỏ nhất của nút con đã khám phá

* Thuật toán tiếp tục duyệt các nút còn lại cho đến khi đạt nút gốc

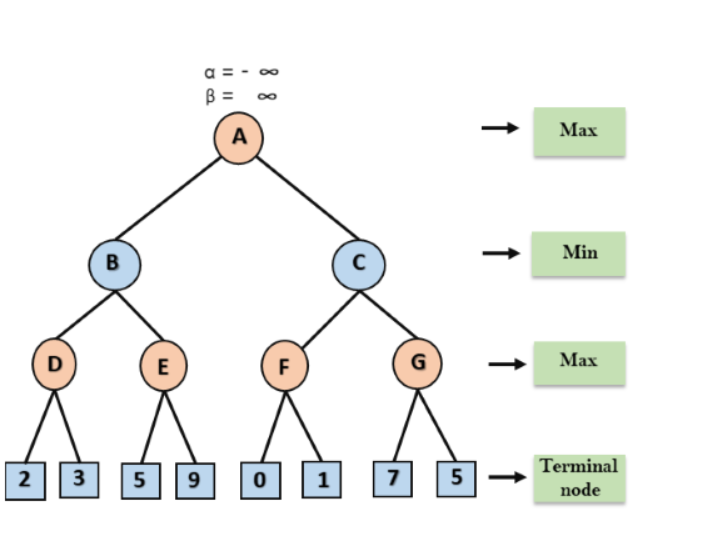
1. Ưu điểm

* Hiệu quả hơn so với thuật toán Minimax , đặc biệt là đối với các trò chơi có số lượng không gian trạng thái lớn
* Giảm thời gian và bộ nhớ cần thiết để tìm kiếm bước đi tốt nhất

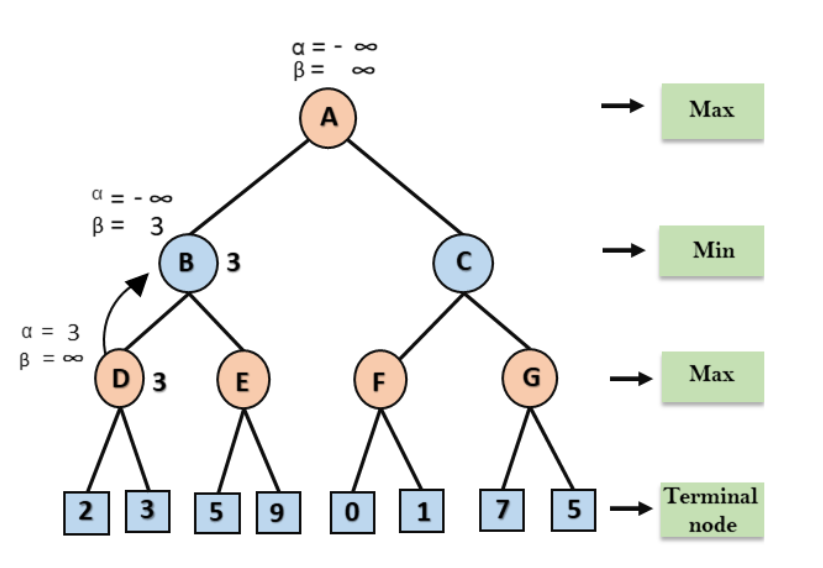
1. Nhược điểm

* Phức tạp hơn so với thuật toán Minimax do phải xét từ vô cùng cho giá trị các lá và thường xuyên cập nhập trạng thái của anpha và beta nếu giá trị đó tối ưu hơn các giá trị của nút ở thời điểm hiện tại
* Có thể bỏ lỡ một số bước đi tốt trong một số trường hợp hiếm gặp

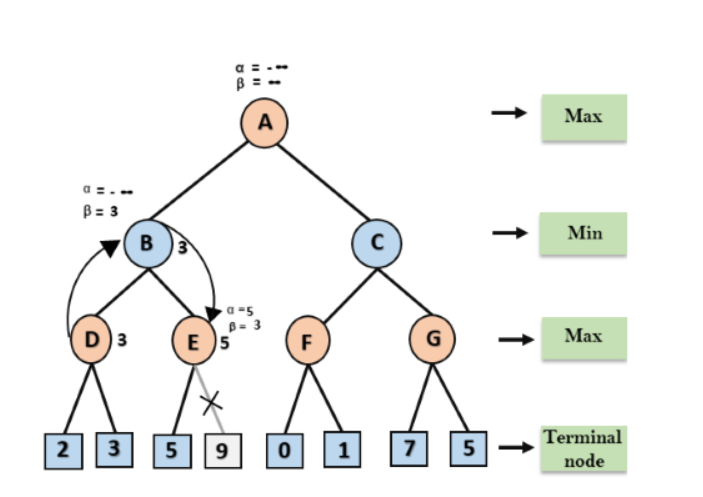
1. Ví dụ thuật toán



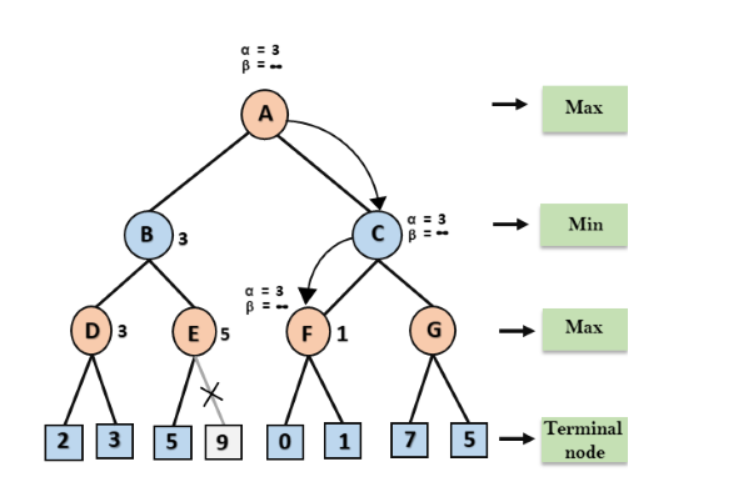
* **Bước 1**: Ở bước đầu tiên, người chơi Max sẽ bắt đầu di chuyển đầu tiên từ nút A nơi α = -∞ và β = + ∞, những giá trị alpha và beta này được truyền lại cho nút B nơi lại α = -∞ và β = + ∞ và Nút B chuyển cùng một giá trị cho nút con D của nó.
* **Bước 2**: Tại nút D, giá trị của α sẽ được tính theo lượt của nó cho Max. Giá trị của α được so sánh với đầu tiên là 2 và sau đó là 3, và max (2, 3) = 3 sẽ là giá trị của α tại nút D và giá trị của nút cũng sẽ là 3.
* **Bước 3**: Bây giờ thuật toán quay ngược lại nút B, trong đó giá trị của β sẽ thay đổi vì đây là lượt của Min, Bây giờ β = + ∞, sẽ so sánh với giá trị của các nút tiếp theo có sẵn, tức là min (∞, 3) = 3, do đó tại nút B bây giờ α = -∞ và β = 3.



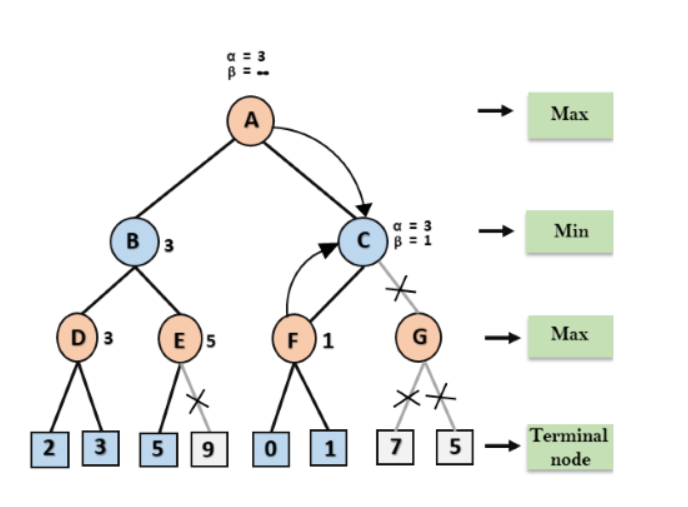
* Trong bước tiếp theo, thuật toán duyệt qua nút kế tiếp tiếp theo của nút B là nút E, và các giá trị của α = -∞ và β = 3 cũng sẽ được chuyển.
* **Bước 4**: Tại nút E, Max sẽ đến lượt, và giá trị của alpha sẽ thay đổi. Giá trị hiện tại của alpha sẽ được so sánh với 5, vì vậy max (-∞, 5) = 5, do đó tại nút E α = 5 và β = 3, trong đó α> = β, vì vậy kế thừa bên phải của E sẽ bị lược bỏ, và thuật toán sẽ không đi qua nó, và giá trị tại nút E sẽ là 5.



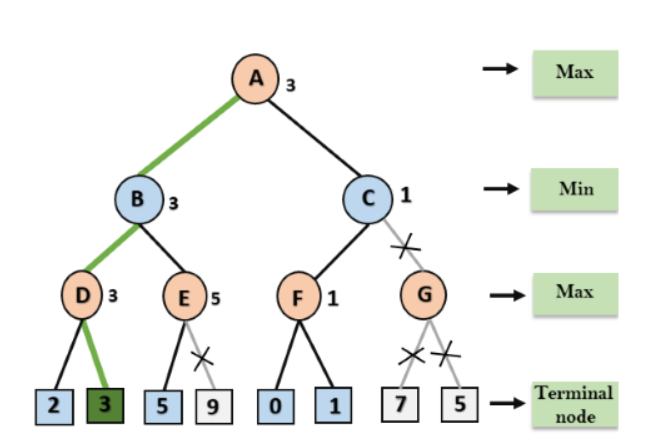
* **Bước 5**: Ở bước tiếp theo, thuật toán lại chiếu ngược cây, từ nút B đến nút A. Tại nút A, giá trị của alpha sẽ được thay đổi giá trị lớn nhất có sẵn là 3 như max (-∞, 3) = 3 và β = + ∞, hai giá trị này bây giờ được chuyển đến người kế nhiệm bên phải của A là Nút C.
* Tại nút C, α = 3 và β = + ∞, và các giá trị tương tự sẽ được chuyển cho nút F.
* **Bước 6**: Tại nút F, một lần nữa giá trị của α sẽ được so sánh với nút con bên trái là 0 và max (3,0) = 3, sau đó so sánh với nút con bên phải là 1 và max (3,1) = 3 vẫn còn α vẫn là 3, nhưng giá trị nút của F sẽ trở thành 1.



* **Bước 7**: Nút F trả về giá trị nút 1 cho nút C, tại C α = 3 và β = + ∞, ở đây giá trị beta sẽ được thay đổi, nó sẽ so sánh với 1 nên min (∞, 1) = 1. Bây giờ tại C, α = 3 và β = 1, và một lần nữa nó thỏa mãn điều kiện α> = β, vì vậy con tiếp theo của C là G sẽ bị lược bớt, và thuật toán sẽ không tính toàn bộ cây con G.



* **Bước 8**: C bây giờ trả về giá trị từ 1 đến A ở đây giá trị tốt nhất cho A là max (3, 1) = 3. Sau đây là cây trò chơi cuối cùng là hiển thị các nút được tính toán và các nút chưa bao giờ tính toán. Do đó, giá trị tối ưu cho bộ cực đại là 3 cho ví dụ này.



1. Di chuyển thứ tự trong anpha- beta pruning

* Hiệu quả của thuật toán phụ thuộc nhiều vào thứ tự mỗi nút được kiểm tra. Thứ tự di chuyển là một khía cạnh quan trọng của việc anpha-beta pruning
* Nó có thể có hai loại:
* **Worst ordering**: Trong một số trường hợp, thuật toán Alpha-Beta Pruning không cắt tỉa bất kỳ lá nào của cây và hoạt động chính xác như thuật toán minimax. Trong trường hợp này, nó cũng tiêu tốn nhiều thời gian hơn vì các yếu tố alpha-beta, động thái cắt tỉa như vậy được gọi là thứ tự tồi tệ nhất. Trong trường hợp này, chuyển động tốt nhất xảy ra ở phía bên phải của cây. Độ phức tạp về thời gian cho một lệnh như vậy là O (bm).
* **Ideal ordering**: Thứ tự lý tưởng để Alpha-Beta Pruning xảy ra khi nhiều lần cắt tỉa diễn ra trên cây và các động thái tốt nhất xảy ra ở phía bên trái của cây. Chúng tôi áp dụng DFS do đó nó tìm kiếm đầu tiên bên trái của cây và đi sâu gấp đôi so với thuật toán minimax trong cùng một khoảng thời gian. Độ phức tạp trong thứ tự lý tưởng là O (bm / 2).

1. Các quy tắc để tìm thứ tự tốt

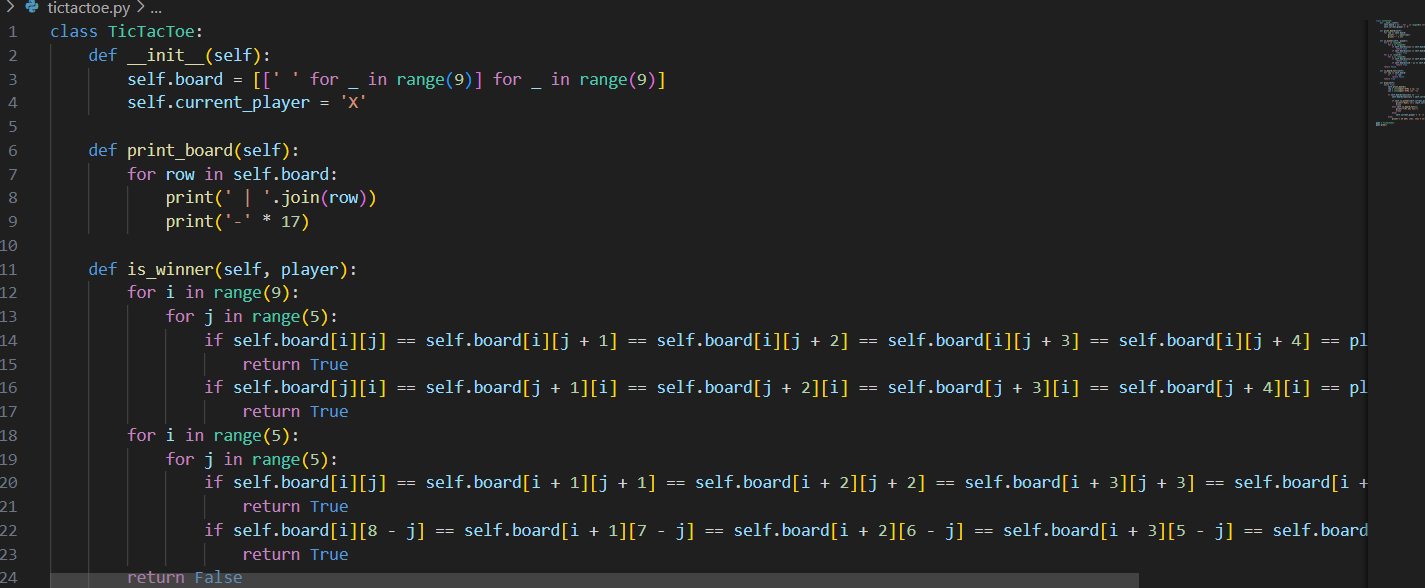
Sau đây là một số quy tắc để tìm thứ tự tốt trong thuật toán cắt tỉa anpha-beta:

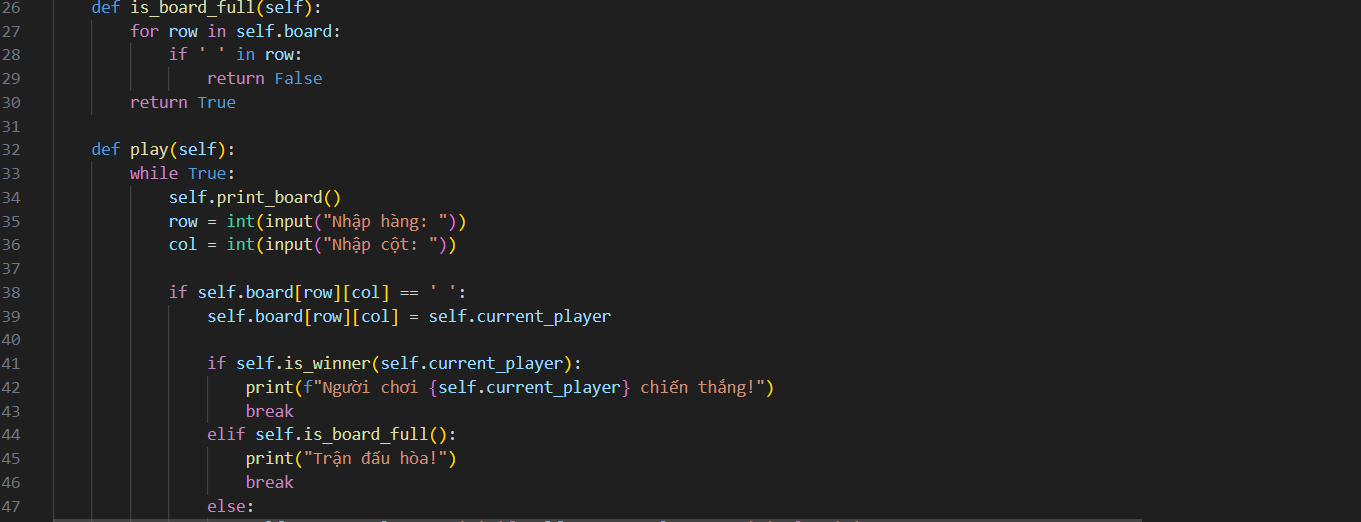
* Xảy ra nước đi tốt nhất từ nút nông nhất
* Thứ tự các nút trong cây sao cho các nút tốt nhất được kiểm tra trước
* Sử dụng kiến thức miền trong khi tìm ra đường đi tốt nhấ. Ví dụ đối với cờ vua , hãy thử thứ thứ tự : bắt trước , sau đó đe dọa , sau đó tiến lên, lùi lại
* Ghi lại các trạng thái, vì các trạng thái có thể lặp đi lặp lại

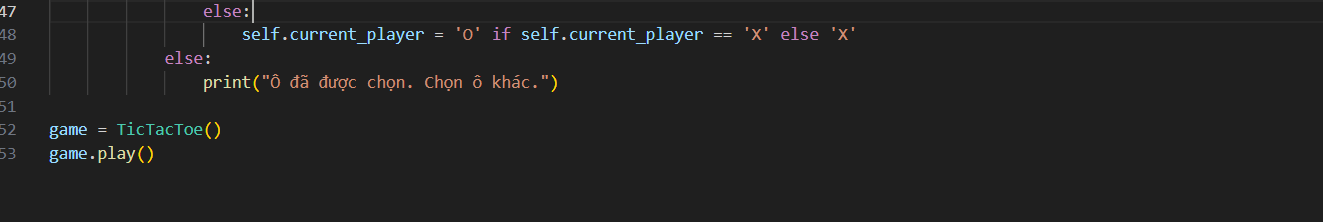
1. **Ứng dụng của thuật toán trong game playing**

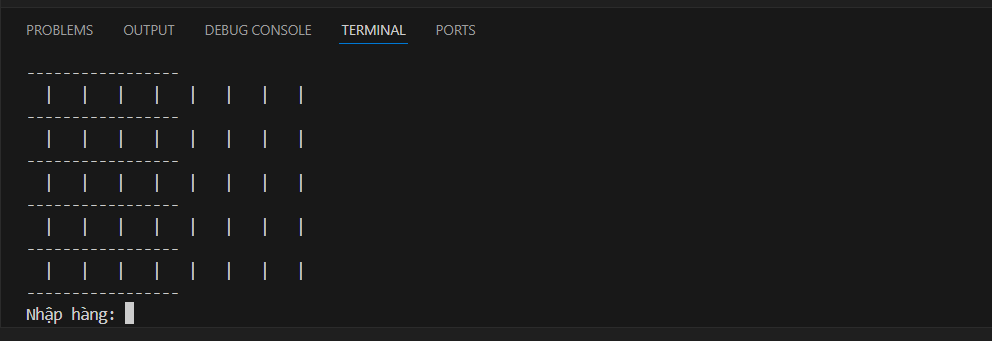
* Tìm kiếm nước đi tốt nhất
* Tiết kiệm thời gian , chi phí
* Tạo ra các trò chơi và các nhân vật game

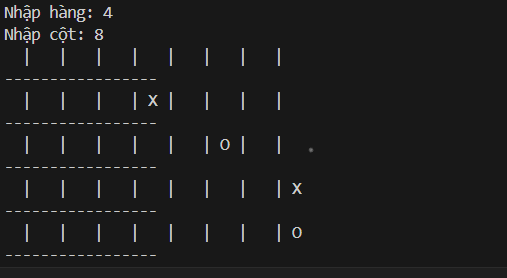
1. **Demo ứng dụng**
2. Chương trình code tictactoe cho yêu cầu ma trận 9x9 mà chỉ cần 5 quân thẳng hàng là thắng



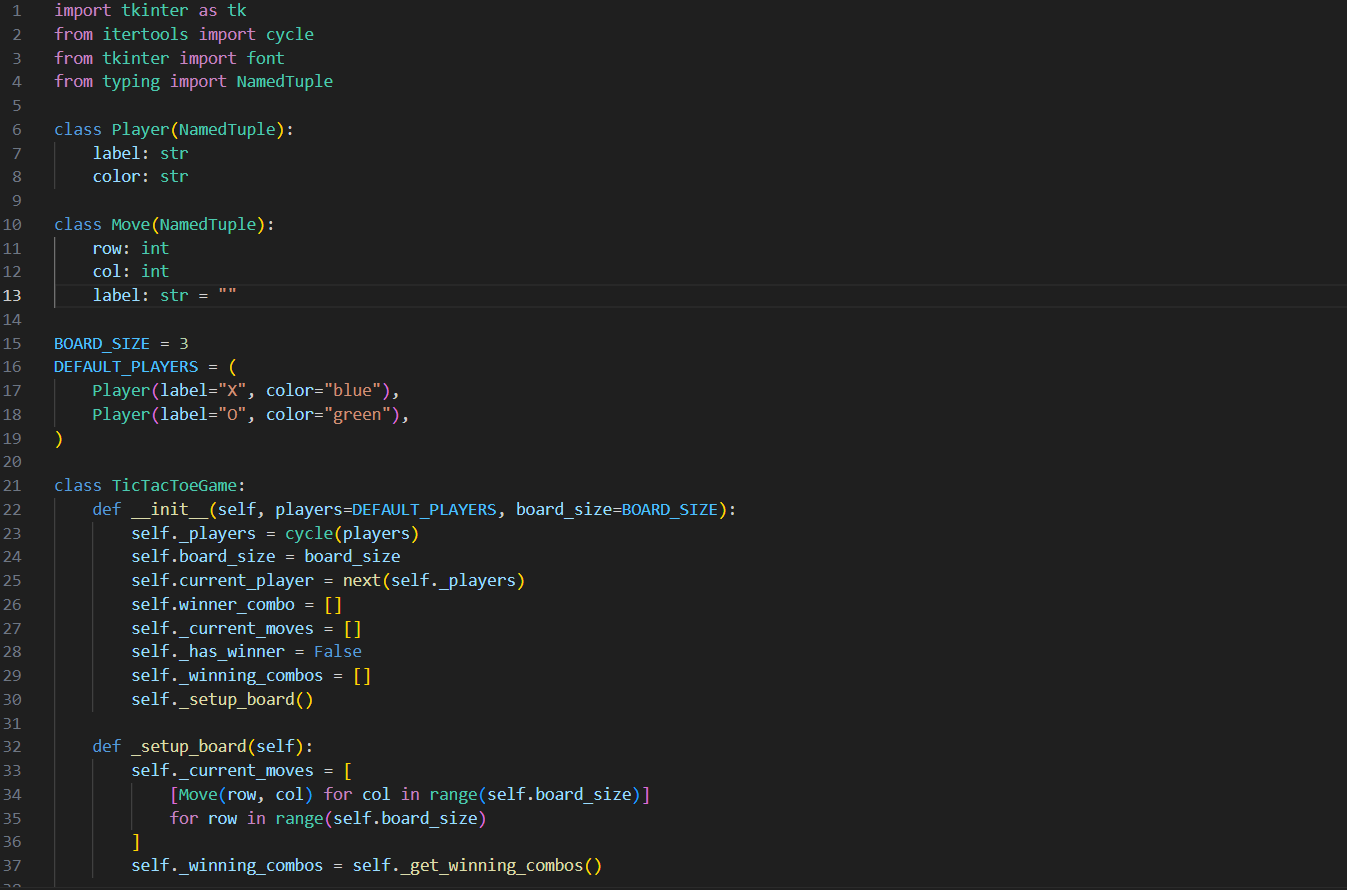


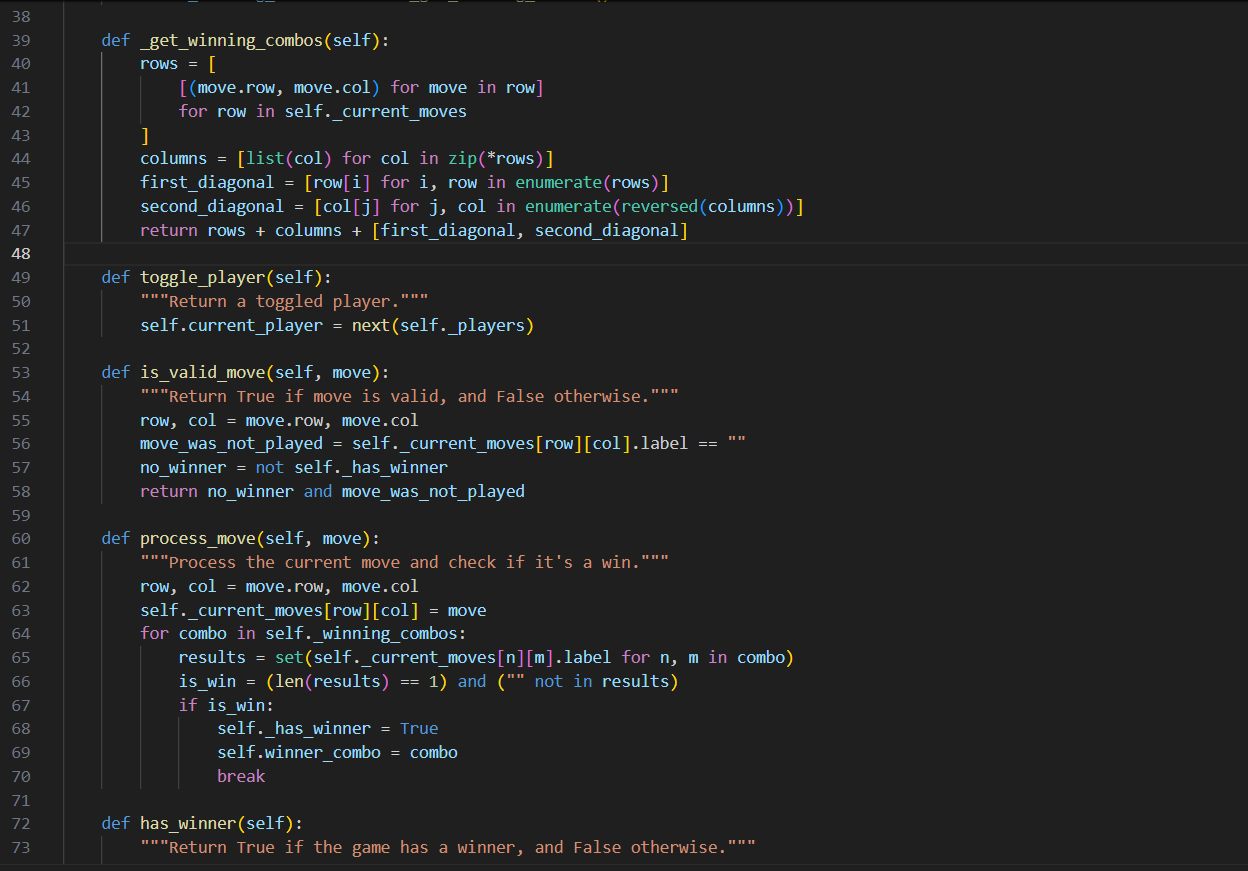


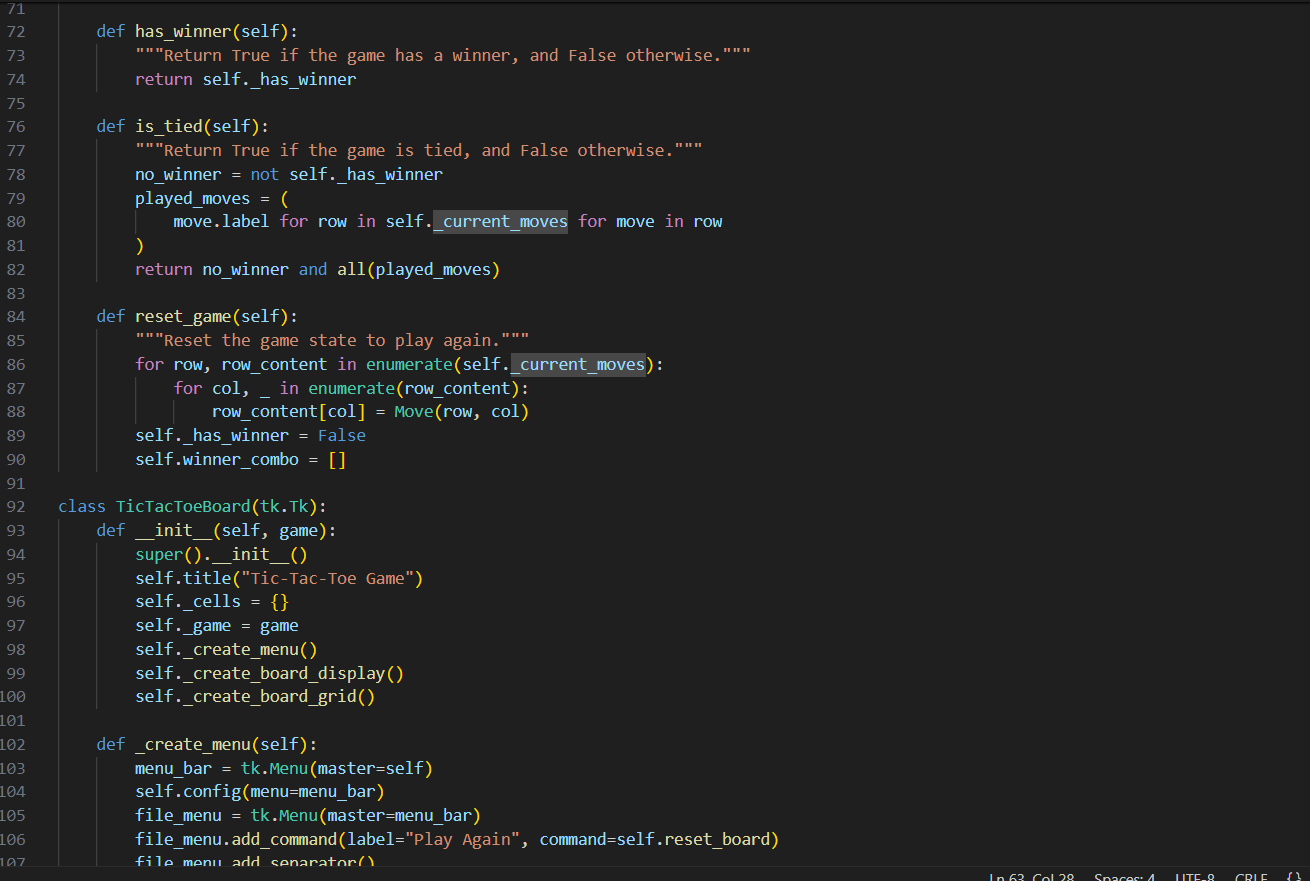


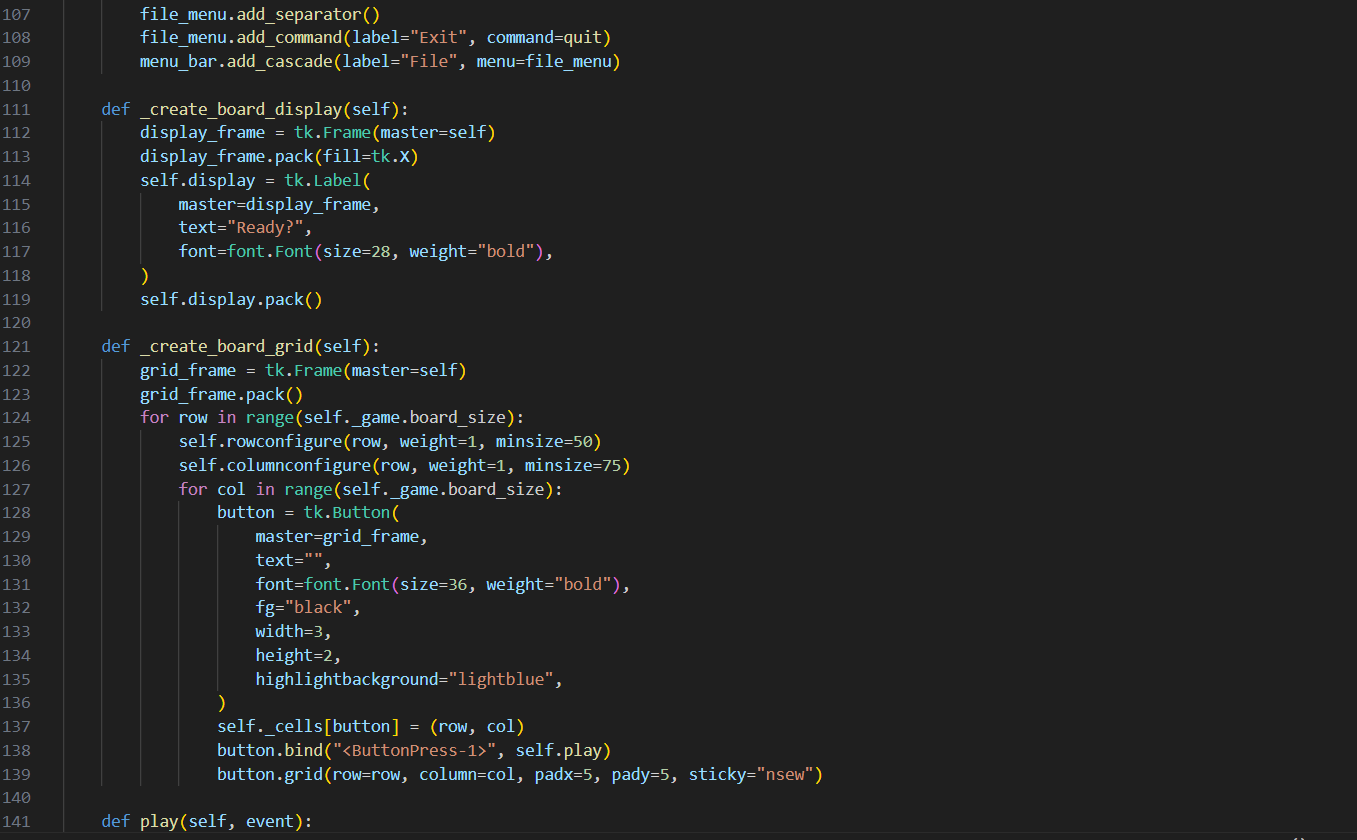


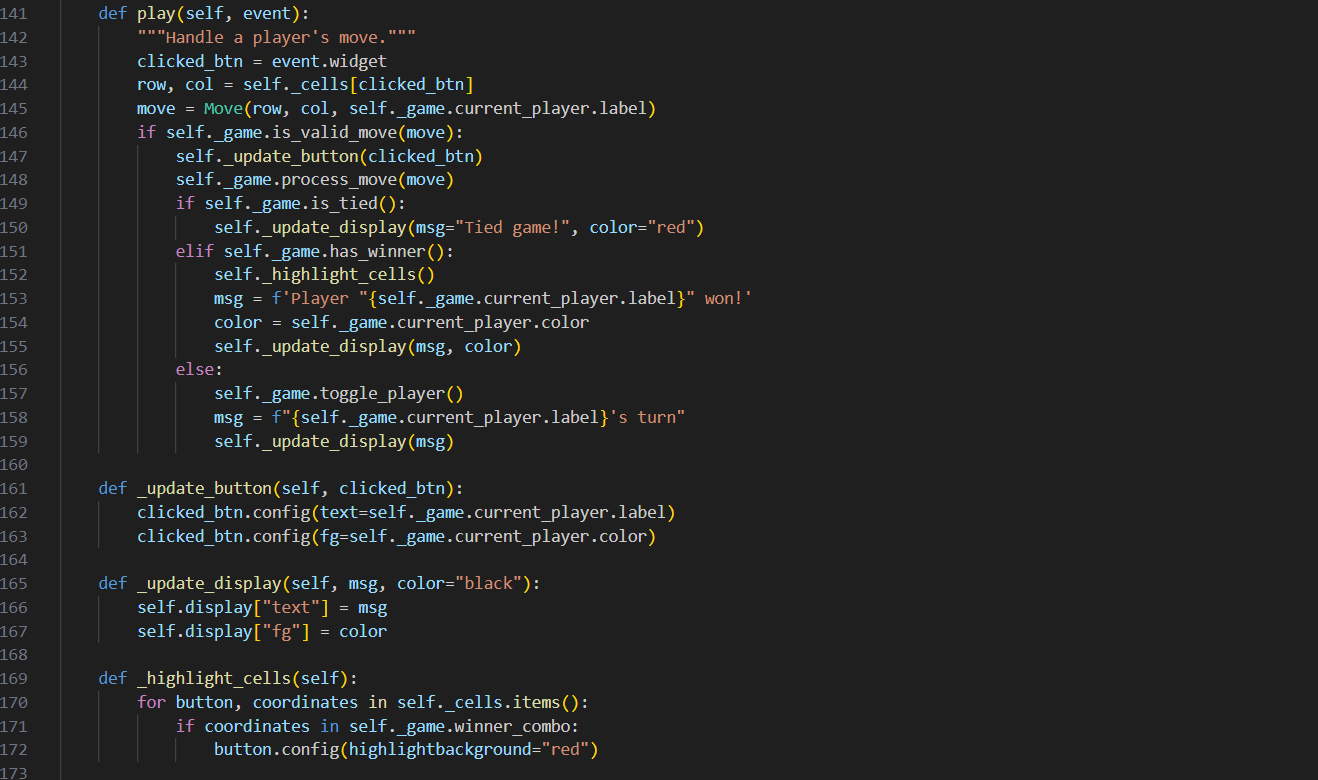
1. Chương trình code XO số quân bằng số kích thước ma trận thì thắng

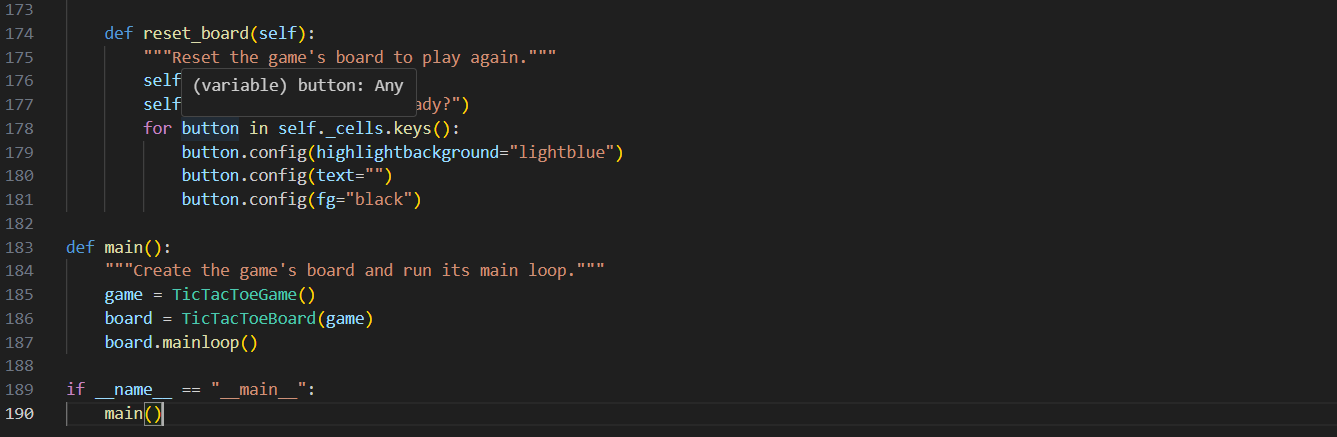












* Kết quả chạy:



