```python
import numpy as np
import random


tablero=np.zeros((4,4),dtype=str)


def imprimir_tablero():
    return print(f"""
                0   1   2  3
            0   {tablero[0,0]} | {tablero[0,1]} | {tablero[0,2]} | {tablero[0,3]}
            1   {tablero[1,0]} | {tablero[1,1]} | {tablero[1,2]} | {tablero[1,3]}
            2   {tablero[2,0]} | {tablero[2,1]} | {tablero[2,2]} | {tablero[2,3]}
            3   {tablero[3,0]} | {tablero[3,1]} | {tablero[3,2]} | {tablero[3,3]}
        """)
```

Declaración del tablero y defino la función imprimir_tablero para visualizar por terminal el tablero con las jugadas

```python
def  juego_humano_vs_ia():

    if condiciones():
        pass
    else:
        x=int(input("En que fila quieres colocar la 'X': "))
        y=int(input("En que columna quieres colocar la 'X': "))
        if tablero[x,y]=='':
            tablero[x,y]='X'
            imprimir_tablero()
        else:
            print("Poscion ocupada")
            juego_humano_vs_ia()

        #turno de la ia
        mejor_valor = -float('inf')
        mejor_jugada = None
        for x in range(4):
            for y in range(4):
                if tablero[x,y]=='':
                    tablero[x, y] = 'O'
                    valor = minimax(tablero, 0, False, -float('inf'), float('inf'))
                    tablero[x, y] = ''
                    if valor > mejor_valor:
                        mejor_valor = valor
                        mejor_jugada = (x, y)

        if mejor_jugada:
            tablero[mejor_jugada[0], mejor_jugada[1]] = 'O'
            imprimir_tablero()
        juego_humano_vs_ia()
```

Definimos la función de juego_humano_vs_ia donde comenzamos evaluando las condiciones de victoria si nadie a ganado pasamos al else donde seleccionaremos nuestras posiciones primero el humano escoge la posición donde pondrá la X si la posición esta vacia se coloca la X en la posición y se imprime el tablero de no ser así dice que la posición esta ocupada y se vuelve a llamar para poner la x de nuevo.

Ahora el turno de la ia se hace un ciclo for anidado para recorrer el tablero y se coloca en una posición para posterior mente llamar a mínimax y con los valores obtenidos de llamarlo decidimos cual fue la mejor posición y por ultimo si la mejor_jugada fue TRUE se coloca en el tablero original y volver a llamar a la función para seguir con el juego.

```
def juego_humano_vs_humano():

    if condiciones():
        pass
    else:

        x=int(input("En que fila quieres colocar la 'X': "))
        y=int(input("En que columna quieres colocar la 'X': "))
        if tablero[x,y]=='':
            tablero[x,y]='X'
            imprimir_tablero()
        else:
            print("Poscion ocupada")
            juego_humano_vs_humano()
        x=int(input("En que fila quieres colocar la 'O': "))
        y=int(input("En que columna quieres colocar la 'O': "))
        if tablero[x,y]=='':
            tablero[x,y]='O'
            imprimir_tablero()
        else:
            print("Poscion ocupada")
            juego_humano_vs_humano()

        juego_humano_vs_humano()
```

Definimos la función juego_humano_vs_humano donde comenzamos evaluando si alguien ya a ganado y empieza X el cual selecciona una posición donde será colocada y si el lugar esta vacio lo colocamos e imprimimos el tablero.

Pasamos a que O seleccione el lugar donde será colocado y si no se encuentra ocupado lo colocamos en el lugar e imprimimos el tablero para después llamar a la función para comenzar el siguiente turno

```python
def juego_ia_vs_ia():
    turno = True  # True para IA con 'O', False para IA con 'X'

    while True:
        imprimir_tablero()

        if turno:
            # Turno de la IA 'O'
            mejor_valor = -float('inf')
            mejor_jugada = None
            for i in range(4):
                for j in range(4):
                    if tablero[i, j] == '':
                        tablero[i, j] = 'O'
                        valor = minimax(tablero, 0, False, -float('inf'), float('inf'))
                        tablero[i, j] = ''
                        if valor > mejor_valor:
                            mejor_valor = valor
                            mejor_jugada = (i, j)

            if mejor_jugada:
                tablero[mejor_jugada[0], mejor_jugada[1]] = 'O'
                imprimir_tablero()
            if condiciones():
                break
            turno = False  # Cambiar turno a IA 'X'
```

definimos la funcion juego_ia_vs_ia donde asignaremos el turno mediante un valor boolean y entramos a un ciclo while imprimimos el tablero y entramos al if si es TRUE que es el turno de O se hace un ciclo for anidado para recorrer el tablero y se coloca en una posición para posterior mente llamar a mínimax y con los valores obtenidos de llamarlo decidimos cual fue la mejor posición y por ultimo si la mejor_jugada fue TRUE se coloca en el tablero original y volver a llamar a la función para seguir con el juego y colocamos el turno en FALSE

```
else:
    # Turno de la IA 'X'
    mejor_valor = float('inf')
    mejor_jugada = None
    for i in range(4):
        for j in range(4):
            if tablero[i, j] == '':
                tablero[i, j] = 'X'
                valor = minimax(tablero, 0, True, -float('inf'), float('inf'))
                tablero[i, j] = ''
                if valor < mejor_valor:
                    mejor_valor = valor
                    mejor_jugada = (i, j)

    if mejor_jugada:
        tablero[mejor_jugada[0], mejor_jugada[1]] = 'X'
        imprimir_tablero()
    if condiciones():
        break
    turno = True  # Cambiar turno a IA 'O'
```

Como el turno es FALSE ahora pasamos al else donde será el turno de X y hacemos lo mismo que estamos haciendo con las IA's se hace un ciclo for anidado para recorrer el tablero y se coloca en una posición para posterior mente llamar a mínimax y con los valores obtenidos de llamarlo decidimos cual fue la mejor posición y por ultimo si la mejor_jugada fue TRUE se coloca en el tablero original y volver a llamar a la función para seguir con el juego y colocamos el turno en TRUE.

```
def condiciones():
    if '' not in tablero:
        print("La partida termino en empate")
        return True
    elif np.all(np.diag(tablero)=='X'):
        print("!!Ganaste X¡¡")
        imprimir_tablero()
        return True
    elif np.all(np.diag(tablero)=='O'):
        print("!!Ganaste O¡¡")
        imprimir_tablero()
        return True
    elif np.all(np.diag(np.fliplr(tablero))=='X'):
        print("!!Ganaste X¡¡")
        imprimir_tablero()
        return True
    elif np.all(np.diag(np.fliplr(tablero))=='O'):
        print("!!Ganaste O¡¡")
        imprimir_tablero()
        return True
    else:
        for i in range(4):
            fila=tablero[i, :]
            if np.all(fila=='X'):
                print("!!Ganaste X¡¡")
                imprimir_tablero()
```

La función de condición es donde se revisan las condiciones de victoria, derrota y empata en esta se comprueba si el tablero esta lleno lo cual significa que hubo empate, se comprueba fila por fila y columna por columna si alguna tiene todas iguales que imprime el mensaje de victoria y quien fue el ganador

```python
cache = {}

def minimax(tablero, profundidad, es_max, alpha, beta):
    estado = tablero.tostring()  # Convertir tablero en una cadena para usarla como clave
    if estado in cache:
        return cache[estado]

    elif profundidad == 0 or condiciones():  # Evaluar victoria o empate
        evaluacion = evaluar_tablero(tablero)
        cache[estado] = evaluacion
        return evaluacion

    elif es_max:
        max_eval = -float('inf')
        for i in range(4):
            for j in range(4):
                if tablero[i, j] == '':
                    tablero[i, j] = 'O'
                    evaluacion = minimax(tablero, profundidad - 1, False, alpha, beta)
                    tablero[i, j] = ''
                    max_eval = max(max_eval, evaluacion)
                    alpha = max(alpha, evaluacion)
                    if beta <= alpha:
                        break
        cache[estado] = max_eval
        return max_eval
    else:
        min_eval = float('inf')
        for i in range(4):
            for j in range(4):
                if tablero[i, j] == '':
                    tablero[i, j] = 'X'
                    evaluacion = minimax(tablero, profundidad - 1, True, alpha, beta)
                    tablero[i, j] = ''
                    min_eval = min(min_eval, evaluacion)
                    beta = min(beta, evaluacion)
                    if beta <= alpha:
                        break
        cache[estado] = min_eval
        return min_eval
```

cache = {} se define un diccionario vacío llamado `cache` para almacenar los estados de los tableros ya evaluados y sus correspondientes valores. Esto permite reutilizar evaluaciones previas, evitando cálculos repetitivos y mejorando la eficiencia def minimax(tablero, profundidad, es_max, alpha, beta) esta es la definición de la función minimax, que toma como parámetros el tablero actual, la profundidad de búsqueda, un booleano es_max (que indica si es el turno del jugador maximizador), y los valores de alpha y beta para la poda Alfa-Beta.

Estado = tablero.tostring() convierte el tablero en una cadena que puede usarse como clave en el diccionario cache. De esta manera, cada estado del tablero se puede almacenar y reutilizar if estado in cache Verifica si el estado actual del tablero ya fue evaluado y guardado en la caché. Si es así, se retorna ese valor, evitando calcular

nuevamente la evaluación para ese estado elif profundidad == 0 o condiciones() esta condición indica el final de la búsqueda. Si se ha alcanzado la profundidad máxima o el juego ha terminado (victoria, derrota o empate), se detiene la búsqueda evaluacion = evaluar_tablero(tablero) si la búsqueda ha terminado, se llama a una función para evaluar el tablero actual, asignando un valor basado en el resultado (por ejemplo, 1 para victoria, -1 para derrota, 0 para empate) cache[estado] = evaluacion se almacena el valor de la evaluación en la caché utilizando el estado del tablero como clave. Esto permite acceder rápidamente a este valor si el mismo estado del tablero se vuelve a evaluar más tarde return evaluacion retorna el valor de la evaluación para ese estado terminal del tablero.

elif es_max si es_max es True, significa que es el turno del jugador maximizador (la IA). Aquí comienza el bloque de código para la IA. max_eval = -float('inf') se inicializa max_eval con el valor más bajo posible, -infinito, ya que el objetivo del maximizador es encontrar el valor más alto posible (for i in range(4) y for j in range(4)) estos bucles anidados recorren todas las posiciones del tablero (de tamaño 4x4) buscando celdas vacías (representadas por '').

if tablero[i, j] == '' verifica si la celda en la posición (i, j) está vacía. Si lo está, la IA puede hacer un movimiento ahí tablero[i, j] = 'O' la IA (maximizador) coloca su ficha (`'O'`) en la celda vacía evaluacion = minimax(tablero, profundidad - 1, False, alpha, beta) se llama recursivamente a la función `minimax` con la profundidad disminuida en 1. El turno cambia a `False`, lo que significa que ahora le toca al jugador minimizador (el humano).

Tablero[i, j] = '' deshacer el movimiento realizado por la IA, devolviendo la celda a su estado vacío para permitir la evaluación de otras posibles jugadas max_eval = max(max_eval, evaluacion) actualiza el valor de max_eval con el mayor valor entre el valor actual y la evaluación obtenida del movimiento simulado. alpha = max(alpha, evaluacion) se actualiza el valor de alpha con el valor máximo entre el valor actual de alpha y la evaluación obtenida. Este es un paso clave en la poda Alfa-Beta, ya que alpha representa el mejor valor que el maximizador puede obtener.

if beta <= Alpha si beta es menor o igual que alpha, se rompe el bucle. Esto significa que el jugador minimizador ya tiene una mejor opción y no hay necesidad de seguir evaluando más movimientos para este jugador maximizador. Este es el proceso de poda que reduce significativamente el número de nodos que se exploran. cache[estado] =max_eval Se almacena el valor máximo encontrado en la caché, utilizando el estado del tablero como clave. return max_eval Retorna el valor máximo obtenido para el jugador maximizador. else Si es_max es False, significa que es el

turno del jugador minimizador (el humano). Aquí comienza el bloque de código para el jugador minimizador. min_eval = float('inf') Se inicializa min_eval con el valor más alto posible, +infinito, ya que el objetivo del minimizador es encontrar el valor más bajo posible.

El bucle doble se repite de la misma forma que en el bloque del maximizador, pero ahora el jugador minimizador X coloca su ficha, y la lógica es inversa. El minimizador intenta minimizar el valor, así que se actualiza min_eval con el valor mínimo y se actualiza beta con el valor mínimo entre beta y la evaluación. if beta <= alpha Se realiza la misma verificación para la poda. Si beta es menor o igual que alpha, se rompe el bucle, podando el árbol de búsqueda. Al final del bloque del minimizador, se almacena min_eval en la caché y se retorna ese valor.

```python
def evaluar_tablero(tablero):
    # Revisar filas y columnas
    for i in range(4):
        if np.all(tablero[i, :] == 'X'):
            return -10  # Ganó el jugador humano
        if np.all(tablero[i, :] == 'O'):
            return 10  # Ganó la IA
        if np.all(tablero[:, i] == 'X'):
            return -10
        if np.all(tablero[:, i] == 'O'):
            return 10

    # Revisar diagonales
    if np.all(np.diag(tablero) == 'X'):
        return -10
    if np.all(np.diag(tablero) == 'O'):
        return 10
    if np.all(np.diag(np.fliplr(tablero)) == 'X'):
        return -10
    if np.all(np.diag(np.fliplr(tablero)) == 'O'):
        return 10

    # Verificar si es empate
    if '' not in tablero:
        return 0

    # No hay ganador ni empate
    return 0
```

La función evaluar_tablero se encarga en que momento se a empatado, ganado, perdido o no a pasado nada y dependiendo del lo que pase devuelve un valor 10 si se gana, -10 si se pierde 0 si se empeta y 0 si no a pasado nada

```python
def seleccionar_modalidad():
    modalidad = int(input("Selecciona la modalidad: \n1. Humano vs Humano\n2. Humano vs IA\n3. IA vs IA\n"))

    if modalidad == 1:
        imprimir_tablero()
        juego_humano_vs_humano()
    elif modalidad == 2:
        imprimir_tablero()
        juego_humano_vs_ia()
    elif modalidad == 3:
        juego_ia_vs_ia()
    else:
        print("Opción no válida")


if __name__=="__main__":
    seleccionar_modalidad()
```

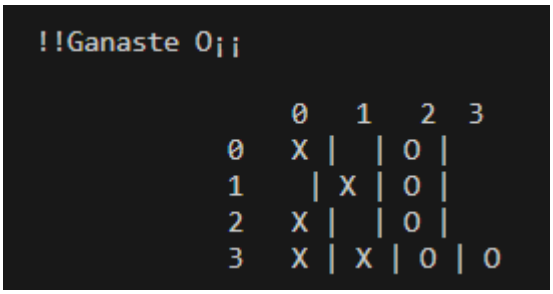
ya por ultimo definimos una función para seleccionar el modo de juego y la mandamos a llamar donde se iniciara todo.

Resultados

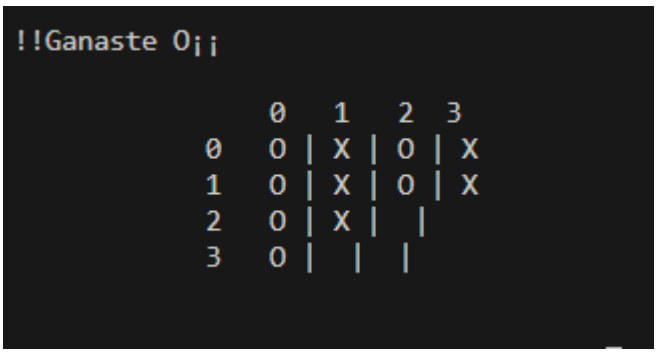
humano vs IA

```
    0   1   2  3
0   X | O | O | O
1   X | O | O | O
2   X | X | O | X
3     | X | X | X


    0   1   2  3
0   X | O | O | O
1   X | O | O | O
2   X | X | O | X
3   O | X | X | X

La partida termino en empate
```

Humano contra humano


!!Ganaste O¡¡
0 1 2 3
0 X | | O |
1 | X | O |
2 X | | O |
3 X | X | O | O

IA vs IA


!!Ganaste O¡¡
0 1 2 3
0 O | X | O | X
1 O | X | O | X
2 O | X | |
3 O | | |