Nim (21 Sticks Variant)

Setup

- Start with **21 sticks** (or counters, stones, matches, etc.).
- Two players take turns.

Rules

1. On your turn, you must take **1, 2, or 3 sticks** from the pile.
2. Players alternate turns.
3. **The player forced to take the last stick loses.**

Example Play

- Start: 21 sticks.
- Player A takes 2 → 19 left.
- Player B takes 3 → 16 left.
- Player A takes 1 → 15 left.
- ... and so on, until one player is forced to take the last stick and loses.

In [1]:

```
from Game import *
```

```
Version:  0.3.14
```

- What is the state? (how are they represented?)
  - state = number of sticks remaining (integer)
- What is a move?
  - (integer) number of sticks taken (1, 2, or 3)

In [2]:

```
def initial_state():
    return 21
```

In [3]:

```
initial_state()
```

Out[3]:

```
21
```

In [4]:

```
def show_state(state,player):
    print("Player",player)
    print(f"Sticks remaining: {state}")
```

In [5]:

```
state=initial_state()
show_state(state,1)
```

```
Player 1
Sticks remaining: 21
```

In [6]:

```
def valid_moves(state,player):
    # return a **list** of moves that are valid

    if state==1:
        return [1]
    elif state==2:
        return [1,2]
    else:
        return [1,2,3]

```

In [7]:

```
valid_moves(4,1)
```

Out[7]:

```
[1, 2, 3]
```

In [8]:

```
def update_state(state,player,move):
    new_state = state - move
    return new_state
```

In [9]:

```
update_state(10,1,3)
```

Out[9]:

```
7
```

In [10]:

```
def win_status(state,player):
    # return None if the game is not over
    # return 'win' if player has won
    # return 'lose' if player has lost
    # return 'stalemate' if player has stalemate

    if player==1:
        other_player=2
    else:
        other_player=1

    if state==0:
        return 'lose'
    else:
        return None
```

In [11]:

```
win_status(3,1)
```

Agents

In [12]:

```
def human_move(state,player):
    move = int(input("Enter your move (1, 2, or 3): "))
    while move not in valid_moves(state,player):
        print("Invalid move. Try again.")
        move = int(input("Enter your move (1, 2, or 3): "))
    return move

human_agent=Agent(human_move)
```

In [13]:
```
def random_move(state,player):
    return random.choice(valid_moves(state,player))

random_agent=Agent(random_move)
```

In [14]:
```
from Game.minimax import *
```

In [18]:
```
state=7
minimax_values(state,1,display=False)
```

Out[18]:
```
([1, -1, -1], [2, 3, 1])
```

In [21]:
```
def minimax_move(state,player):
    values,actions = minimax_values(state,player,display=False)
    return top_choice(actions,values)
minimax_agent=Agent(minimax_move)
```

In [ ]:

In [ ]:

Running the Game

In [22]:
```
g=Game()
g.run(minimax_agent,random_agent)
```

```
====
Game  1
Player 1
Sticks remaining: 21
Player 1 moves 1
Player 2
Sticks remaining: 20
Player 2 moves 3
Player 1
Sticks remaining: 17
Player 1 moves 2
Player 2
Sticks remaining: 15
Player 2 moves 2
Player 1
Sticks remaining: 13
Player 1 moves 1
Player 2
Sticks remaining: 12
Player 2 moves 2
Player 1
Sticks remaining: 10
Player 1 moves 1
Player 2
Sticks remaining: 9
Player 2 moves 2
Player 1
Sticks remaining: 7
Player 1 moves 2
Player 2
Sticks remaining: 5
Player 2 moves 3
Player 1
Sticks remaining: 2
Player 1 moves 1
Player 2
Sticks remaining: 1
Player 2 moves 1
Player 2
Sticks remaining: 0
Player  1 won.
```

```
Out[22]: [1]
```

```
In [ ]:
```