MINMAX-ALGORITHM

```
import os
import time
from random import choice
+-*
HUMAN = -1
COMP = +1
board = [
[0, 0, 0],
[0, 0, 0],
[0, 0, 0]
]

def evaluate(state):
if wins(state, COMP):
return +1
elif wins(state, HUMAN):
return -1
else:
return 0

def wins(state, player):
win_state = [
[state[0][0], state[0][1], state[0][2]],
[state[1][0], state[1][1], state[1][2]],
[state[2][0], state[2][1], state[2][2]],
[state[0][0], state[1][0], state[2][0]],
[state[0][1], state[1][1], state[2][1]],
[state[0][2], state[1][2], state[2][2]],
[state[0][0], state[1][1], state[2][2]],
[state[2][0], state[1][1], state[0][2]],
]
return [player, player, player] in win_state

def game_over(state):
return wins(state, HUMAN) or wins(state, COMP)

def empty_cells(state):
cells = []
for x, row in enumerate(state):
for y, cell in enumerate(row):
if cell == 0:
cells.append([x, y])
return cells

def valid_move(x, y):
return [x, y] in empty_cells(board)

def set_move(x, y, player):
if valid_move(x, y):
board[x][y] = player
return True
return False
```

```
def minimax(state, depth, player):
if player == COMP:
best = [-1, -1, -float('inf')]
else:
best = [-1, -1, +float('inf')]
if depth == 0 or game_over(state):
score = evaluate(state)
return [-1, -1, score]
for cell in empty_cells(state):
x, y = cell[0], cell[1]
state[x][y] = player
score = minimax(state, depth - 1, -player)
state[x][y] = 0
score[0], score[1] = x, y
if player == COMP:
if score[2] > best[2]:
best = score
else:
if score[2] < best[2]:
best = score
return best

def clean():
os.system('cls' if os.name == 'nt' else 'clear')

def render(state):
chars = {
-1: 'X',
+1: 'O',
0: ' '
}
str_line = '---------------'
print('\n' + str_line)
for row in state:
for cell in row:
symbol = chars[cell]
print(f'| {symbol} ', end='')
print('|')
print(str_line)

def ai_turn(c_choice, h_choice):
depth = len(empty_cells(board))
if depth == 0 or game_over(board):
return
clean()
print(f'Computer turn [{c_choice}]')
render(board)
if depth == 9:
x = choice([0, 1, 2])
y = choice([0, 1, 2])
set_move(x, y, COMP)

# Main game loop
if __name__ == '__main__':
human_choice = 'X'
computer_choice = 'O'
while True:
ai_turn(computer_choice, human_choice)
render(board)
```

```
if game_over(board):
break
x, y = map(int, input("Your turn (row column): ").split())
set_move(x, y, HUMAN)
render(board)
if game_over(board):
break
```

OUTPUT:

Here’s the initial game board:

```
1 | 2 | 3
---|---|---
4 | 5 | 6
---|---|---
7 | 8 | 9

Your turn! Please enter your move (row column):

5

1 | 2 | 3
---|---|---
4 | X | 6
---|---|---
7 | 8 | 9



O | 2 | 3
---|---|---
4 | X | 6
---|---|---
7 | 8 | 9

Your turn! Please enter your move (row column):
2 1


O | X | 3
---|---|---
4 | X | 6
---|---|---
7 | 8 | 9


O | X | O
---|---|---
4 | X | 6
---|---|---
7 | 8 | 9

Your turn! Please enter your move (row column):
7 3


O | X | O
---|---|---
4 | X | 6
---|---|---
```

```
X | 8 | 9



O | X | O
---|---|---
O | X | 6
---|---|---
X | 8 | 9

Your turn! Please enter your move (row column):
8 1

O | X | O
---|---|---
O | X | 6
---|---|---
X | 8 | 9


O | X | O
---|---|---
O | X | O
---|---|---
X | 8 | 9

Your turn! Please enter your move (row column):
8 2


O | X | O
---|---|---
O | X | O
---|---|---
X | X | 9


O | X | O
---|---|---
O | X | O
---|---|---
X | X | O

Your turn! Please enter your move (row column):
9 1


O | X | O
---|---|---
O | X | O
---|---|---
X | X | O


O | O | O
---|---|---
O | X | O
---|---|---
X | X | O
```

Congratulations! I win!