README.md

Vier Gewinnt – Mattis Schulte, Sajan Sivapatham | GitHub

Aufgabenstellung

Am Anfang hat der Lehrer das "Vier Gewinnt"-Spiel erklärt. Dabei hatte ein Schüler ein Protokoll angefertigt, wo alle benötigten Methoden und deren Erklärungen festgehalten wurden. Als das Protokoll fertiggestellt wurde, haben die Schüler die noch leeren Methoden übernommen, die man für das Spiel benötigt. Die Schüler hatten dann nur die notwendigen leeren Methoden und die dazugehörigen Erklärungen parat, um das Spiel zu entwickeln.

- Die vorgegebenen leeren Methoden des Lehrers mussten benutzt werden.
- Unten Links muss der Startpunkt sein (x=0/y=0)(wie im Koordinatensystem).
- Ein Konsolenspiel mit zwei Spielern ist möglich.
- Ausnahmen im Programmablauf werden sauber abgewickelt und führen nicht zum Programmabbruch.
- Fehlerhafte Eingaben werden abgefangen.
- Am Ende wird der Gewinner angezeigt und welche Steine gesetzt worden sind, die zum Gewinn des Spielers führten.
- Die verschiedenen Spielstände werden alle korrekt erkannt.

Erweiterungen für eine bessere Punktzahl:

- Neben der Auswahl zum Zweispielermodus auch ein KI-Modus einbauen, sodass man alleine gegen einen künstlichen Spieler spielen kann.
- Einen Unittest entwickeln, der mindestens 15 Tests beinhaltet und welche, die das Gewinnen überprüfen.

Liste der verwendeten Ausstattung

- Computer mit Windows 10 (21H2) / macOS Monterey 12.2.1 (21D62)
- PyCharm 2021.3.2 (Professional Edition)
- Python 3.10
- Replit (Webseite mit einer Online-IDE, Editor, Compiler und Interpreter)
- Das Protokoll des Schülers

Spielregeln

Die Spielregeln von "Vier Gewinnt", sind sehr einfach erklärt. Der Spieler kann gewinnen, indem er mit seinen zugehörigen Spielsteinen abwechselnd eine Reihe mit mindestens 4 Spielsteinen legt. Er kann diese Reihe vertikal, horizontal oder diagonal verlegen. Das Einzige, was man beachten muss, ist, dass diese vierer Reihe nicht durch einen gegnerischen Spielstein belegt wird. So kann man das Spiel spielen, bis man selbst gewonnen hat, der Gegner gewonnen hat oder bis das Spielbrett voll ist.

Programm

Benutzeranleitung:

Wenn man das Spiel gestartet hat, kommt man in das Spielmenü, wo der Benutzer auswählen kann, in welchen Spielmodus er spielen will:

```
Wählen Sie einen Spielmodus:
(1) Einzelspieler (KI-Modus)
(2) Zweispieler
>>
```

Dabei hat er die Auswahl zwischen einem primitiven Computergegner oder er kann das Spiel mit einem menschlichen Gegenspieler starten.

(1) Einzelspieler (KI-Modus)

Wenn der Benutzer den Einzelspielermodus (also gegen den Computergegner) wählt, muss der Benutzer seinen Benutzernamen eingeben:

```
Bitte geben Sie ihren Benutzernamen ein!
>>
```

Nachdem der Benutzer seinen Benutzernamen eingegeben hat, wird man zur Farbauswahl gebeten:

```
Bitte wählen Sie ihre Farbe (Rot, Grün, Gelb oder Blau)!
>>
```

Der Benutzer muss dann seine gewünschte Spielsteinfarbe eingeben. Nachdem der Spieler eine Farbe ausgewählt hat, wird er zum Spielfeld weitergeleitet, wo der Benutzer denn ersten Spielstein setzen darf. Dabei darf er nur die Spalte auswählen (A-G), wo er denn Spielstein einwerfen möchte:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Spieler 1 (Rot) ist am Zug >> a
```

Nachdem der Benutzer die Spalte eingegeben hat, wird der Spielstein gesetzt und der Computergegner setzt dann sein Spielstein auch automatisch ein (Der Computergegner hat nach Zufallsprinzip die Farbe Gelb ausgewählt):

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|  🔴  |      |      |  🟡  |      |      |      |
```

```
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Spieler 1 (Rot) ist am Zug >>
```

Danach geht das Spiel weiter, bis Spielbrett voll ist oder jemand gewonnen hat.

Der Benutzer hat gewonnen:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|  ●   |  ●   |  ●   |  ●   |  ●   |  ●   |  ●   |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Spieler 1 (Rot) hat mit folgenden Steinen gewonnen: (1|0) (2|0) (3|0) (4|0)
```

Die primitive KI hat gewonnen:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |  ●   |      |      |
+------+------+------+------+------+------+------+
|  ●   |  ●   |  ●   |  ●   |  ●   |  ●   |  ●   |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Primitive KI (Gelb) hat mit folgenden Steinen gewonnen: (1|0) (2|0) (3|0) (4|0)
```

Unentschieden:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|  🟡  |  🔴  |  🔴  |  🔴  |  🟡  |  🟡  |  🔴  |
+------+------+------+------+------+------+------+
|  🟡  |  🔴  |  🟡  |  🟡  |  🔴  |  🔴  |  🔴  |
+------+------+------+------+------+------+------+
|  🟡  |  🟡  |  🔴  |  🔴  |  🔴  |  🟡  |  🟡  |
+------+------+------+------+------+------+------+
|  🔴  |  🔴  |  🔴  |  🟡  |  🟡  |  🔴  |  🔴  |
+------+------+------+------+------+------+------+
|  🟡  |  🟡  |  🔴  |  🔴  |  🔴  |  🟡  |  🟡  |
+------+------+------+------+------+------+------+
|  🟡  |  🟡  |  🟡  |  🔴  |  🟡  |  🔴  |  🟡  |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Das Spiel ist unentschieden!
```

(2) Zweispieler

Der Zweispielermodus ist so ähnlich aufgebaut wie der Einzelspielermodus. Wenn der Benutzer denn Zweispielermodus gewählt hat, wird dieser gebeten, denn Namen des ersten Spielers zu wählen.

```
Spieler 1:
Bitte geben Sie ihren Benutzernamen ein!
>>
```

Auch hier wird abgefragt, welche Farbe der Spieler möchte.

```
Spieler 1:
Bitte wählen Sie ihre Farbe (Rot, Grün, Gelb oder Blau)!
>>
```

Beim zweiten Spieler sind die Abfragen identisch, außer bei der Wahl der Farbe. Denn die Farbe, die der erste Spieler gewählt hat, wird dann dem zweiten Spieler nicht mehr zur Verfügung stehen.

Nachdem beide Spieler ihren Benutzernamen und Farben ausgewählt haben, wird das Spiel gestartet und das Spielbrett erscheint.

```
VIER GEWINNT

+------+------+------+------+------+------+------+
```

```
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Spieler 1 (Rot) ist am Zug >>
```

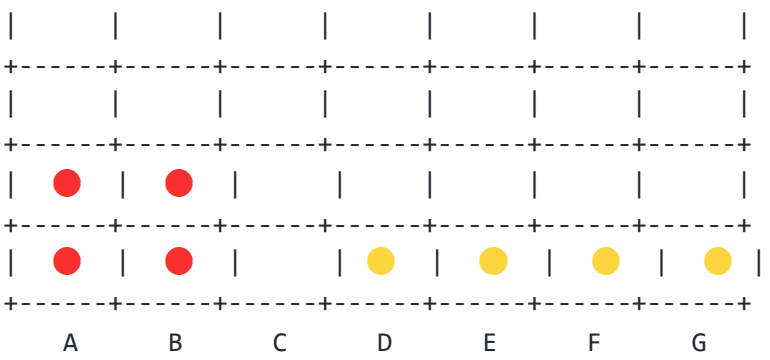
Danach werden abwechselnd zwischen den zwei Spielern die Spielsteine eingeworfen, bis jemand gewonnen hat oder das Spiel unterschieden verläuft bzw. das Spielbrett voll ist.

Der Spieler 1 hat gewonnen:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|  🔴  |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|  🔴  |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|  🔴  |      |      |      |      |  🟡  |      |
+------+------+------+------+------+------+------+
|  🔴  |      |      |      |  🟡  |  🟡  |      |
+------+------+------+------+------+------+------+
    A      B      C      D      E      F      G

Spieler 1 (Rot) hat mit folgenden Steinen gewonnen: (0|0) (0|1) (0|2) (0|3)
```

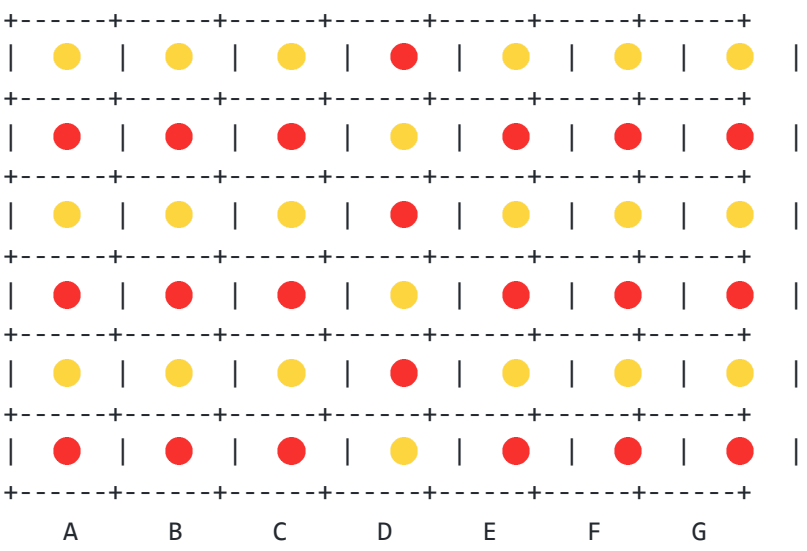
Der Spieler 2 hat gewonnen:

```
VIER GEWINNT

+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+
```



```
    A      B      C      D      E      F      G

Spieler 2 (Gelb) hat mit folgenden Steinen gewonnen: (3|0) (4|0) (5|0) (6|0)
```

Unentschieden bzw. das Spielbrett ist voll:



```
VIER GEWINNT

    A      B      C      D      E      F      G

Das Spiel ist unentschieden!
```

Programmteile erklärt:

Hauptteil (main.py)

```
exec(clear_cmd)
usr_name = 'Spieler 1'
print('Bitte geben Sie ihren Benutzernamen ein!')
usr_input = input('>> ')
if usr_input != '':
    usr_name = usr_input
exec(clear_cmd)

# Select color
print(f'Bitte wählen Sie ihre Farbe ({" oder ".join([", ".join(color_helper[key] for
usr_input = input('>> ')
while True:
```

```
    try:
        if usr_input.upper() in dict((v.upper(), k) for k, v in color_helper.items())
            usr_input = dict((v.upper(), k) for k, v in color_helper.items())[usr_inp
            if usr_input not in valid_colors.colors:
                raise WrongColError('Duplicated color!')
        else:
            raise WrongColError('Invalid color!')

        p1 = Player(usr_name, usr_input)
        valid_colors.colors.remove(usr_input)
        p2 = Player('Primitive KI', choice(valid_colors.colors))
        Game = ConnectFourGame(p1, p2, Board, 1)
        break
    except WrongColError:
        print('Fehlerhafte Auswahl!')
        usr_input = input('>> ')
```

In diesem Teil des Codes wird der Benutzer nach seinem Namen und seiner Farbe gefragt, falls der Benutzer keinen Namen angegeben hat, wird der Standardname also hier z. B. Spieler 1 gewählt. Außerdem kann der Benutzer nur zwischen den erlaubten Farben `valid_colors` wählen, wenn er eine korrekte Farbe ausgewählt hat, wird dann eine Spieler-Instanz für den Benutzer und eine für die KI erstellt – der Name der KI lautet "Primitive KI" und die Farbe der KI wird zufällig gewählt. Aufgrund dessen entfernen wir auch vorher die vom Benutzer gewählte Farbe aus der Liste der erlaubten Farben, damit die KI nicht dieselbe Farbe wählen kann. Außerdem erstellen wir hier auch schon die Instanz für das eigentliche Spiel, der wir die beiden Spieler-Instanzen, das Board und den Spielmodus (1 oder 2) übergeben: `Game = ConnectFourGame(p1, p2, Board, 2)`

```
# Running the actual game
while not (Board.is_board_full()) and not (Board.get_winning_positions()):
    Game.play()
else:
    Board.print_board(p1.color, p2.color)
    if Board.get_winning_positions(Board.field):
        winner_token_owner = Board.get_token(Board.get_winning_positions()[0][0], Boa
        exec('winner_name = p' + str(winner_token_owner) + '.name')
        exec('winner_color = p' + str(winner_token_owner) + '.color')
        winner_color = color_helper[winner_color]
        print(f'{winner_name} ({winner_color}) hat mit folgenden Steinen gewonnen: ',
        [print(f'({"|".join(str(x) for x in item)})', end=' ') for item in Board.get_
        print()
    elif Board.is_board_full():
        print('Das Spiel ist unentschieden!')
```

In diesem Abschnitt wird, solange bis das Spielbrett voll ist oder jemand gewonnen hat, das eigentliche Spiel laufen gelassen. Dafür rufen wir die Methode `play` der `Game` Instanz auf und übergeben dieser die Instanzen der beiden Spieler. Zudem geben wir in diesem Teil auch, falls jemand gewonnen hat, den Namen des Gewinners, seine Farbe und die Steine aus, mit denen er gewonnen hat. Falls das Spiel unentschieden ist, wird das ebenfalls hier ausgegeben.

Die Spiele-Klasse (connect_four_game.py)

In dieser Klasse findet das eigentliche Spiel statt, so werden in dieser Klasse die Spielzüge durchgeführt, bestimmt, wer am Zug ist und der Algorithmus der KI ist ebenfalls in dieser Klasse.

```
def set_ai(self):
    """ The AI algorithm, it wants to either win or avoid losing if neither is possit
    valid_columns = [x for x in range(0, len(self.board.field)) if 0 in self.board.fi

    # Check if the game can be won or a loss avoided
    for win in reversed(range(0, 2)):
        for column in sample(valid_columns, len(valid_columns)):
            test_board = deepcopy(self.board.field)
            # Prioritize winning over avoiding losing
            if win:
                test_board[column][self.board.field[column].index(0)] = 2
            else:
                test_board[column][self.board.field[column].index(0)] = 1

            if self.board.get_winning_positions(test_board):
                self.board.set_token(column, 2)
                self.active_player = 1
                return

    # Check to prevent a random token from being placed in the opponent's favour
    for random_column in sample(valid_columns, len(valid_columns)):
        test_board = deepcopy(self.board.field)
        test_board[random_column][self.board.field[random_column].index(0)] = 2
        if 0 in test_board[random_column]:
            test_board[random_column][self.board.field[random_column].index(0) + 1] =
            if not self.board.get_winning_positions(test_board):
                self.board.set_token(random_column, 2)
                self.active_player = 1
                return
        else:
            self.board.set_token(random_column, 2)
            self.active_player = 1
            return

    # Choose random column
    self.board.set_token(choice(valid_columns), 2)
    self.active_player = 1
```

Dies ist die Methode des KI-Algorithmus, als Erstes werden alle möglichen Spalten ermittelt. Im ersten Abschnitt prüft die KI dann mit der Methode `get_winning_positions` der "Board-Klasse", ob es eine Spalte gibt, die die KI nutzen könnte, um zu gewinnen oder den Nutzer zumindest am Gewinnen zu hindern (dabei, wird der Sieg natürlich bevorzugt). Gibt es keine Spalte, in der dies der Fall ist, wird eine zufällige Spalte ausgewählt. Es wird vorher allerdings noch geprüft, ob der Zug vorteilhaft für den Gegner ist, also ob man durch seinen Zug den Gegner es ermöglicht vier nebeneinander zulegen (dafür ist der zweite Abschnitt da), falls dies nicht der Fall ist oder es keine andere Möglichkeit gibt, legt die KI.

```
def play(self):
    color_helper = {'RED': 'Rot', 'GREEN': 'Grün', 'YELLOW': 'Gelb', 'BLUE': 'Blau'}
    self.board.print_board(self.p1.color, self.p2.color)
    if self.active_player == 1:
        # Player one's turn
        while True:
            try:
                self.set_player1(input(f'{self.p1.name} ({color_helper[self.p1.color]
                break
            except(ValueError, IndexError):
                print('Fehlerhafte Auswahl!')
            except ColumnFullError:
                print('Diese Spalte ist schon voll!')
    elif self.active_player == 2 and self.game != self.AI:
        # Player two's turn
        while True:
            try:
                self.set_player2(input(f'{self.p2.name} ({color_helper[self.p2.color]
                break
            except(ValueError, IndexError):
                print('Fehlerhafte Auswahl!')
            except ColumnFullError:
                print('Diese Spalte ist schon voll!')
    elif self.active_player == 2 and self.game == self.AI:
        # AI's turn
       self.set_ai()
```

Dies ist die Spiele Methode, in dieser werden die eigentlichen Spielzüge durchgeführt oder die KI aufgerufen. Zudem werden hier auch Fehler wie eine fehlerhafte Eingabe der Spalte oder die Eingabe einer vollen Spalte behandelt und eine Fehlermeldung für den Benutzer ausgegeben.

Die Spielfeld-Klasse (connect_four_board.py)

Dies ist die Klasse für das Spielfeld, sie enthält Methoden zum setzten und Auslesen von Spielsteinen, zum Ausgeben des Spielfelds, zum zurücksetzten des Spielfelds und zum Überprüfen.

Als Erstes werden die Konstanten festgesetzt, die das Spielbrett aufnehmen soll. Also die Spieler während des Spiels definiert die Größe des Spielfeldes und welche Eingaben akzeptiert werden soll.

```
class ConnectFourBoard:
    """ Board to play ConnectFour. Pos(x,y) | pos(0,0) lower left-hand corner | xmax
    EMPTY = 0
    PLAYER1 = 1
    PLAYER2 = 2
    X_MAX = 7  # 7 columns
    Y_MAX = 6  # 6 rows

    identifier = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N
```

Als Nächstes wird definiert, wie das Spielbrett zurückgesetzt wird. Dabei wird beim zurücksetzten bzw. beim Starten die Spielfelder des Spielbrettes mit Nullen gefüllt.

```
def __init__(self):
    self.reset_board()

def reset_board(self):
    """ Write zeros into every field """
    self.field = [[0 for _ in range(self.Y_MAX)] for _ in range(self.X_MAX)]
```

Dann wird definiert, was als Spielbrett ausgegeben bzw. geprintet wird, hier werden denn verschiedenen Farben auch das passende Emoji zugewiesen.

```
def print_board(self, p1_color='RED', p2_color='YELLOW'):
    """ Show the board in a console grid """
    exec(clear_cmd)
    print('\nVIER GEWINNT\n')

    sep = '+'
    for _ in range(self.X_MAX):
        sep += '------+'
    print(sep)

    translated_colors = {'RED': '🔴', 'GREEN': '🟢', 'YELLOW': '🟡', 'BLUE': '🔵'}

    for row in reversed(range(self.Y_MAX)):
        for column in range(self.X_MAX):
            if self.field[column][row] == self.EMPTY:
```

```
                print('|    ', end='  ')
            elif self.field[column][row] == self.PLAYER1:
                print(f'|  {translated_colors[p1_color]}', end='  ')
            elif self.field[column][row] == self.PLAYER2:
                print(f'|  {translated_colors[p2_color]}', end='  ')
        print('|\n' + sep)

    # Print identifiers
    for i in range(self.X_MAX):
        print('    ' + self.identifier[i], end='  ')

    print('\n')
```

Mit diesen beiden Methoden werden die Spielsteine ausgelesen oder gesetzt, beim setzten eines Spielsteines werden außerdem die passenden Methoden zum Überprüfen, ob die Eingabe korrekt ist aufgerufen.

```
def set_token(self, column: int, token: int):
    """ Check for validity and set the token """
    if self.is_board_full():
        raise BoardFullError('The board is completely filled!')
    elif column > self.X_MAX:
        raise IndexError() from None
    elif self.is_col_full(column):
        raise ColumnFullError('This column is already full!')
    else:
        self.field[column][self.field[column].index(0)] = token

def get_token(self, x_pos, y_pos) -> int:
    """ Get token at specified position """
    return self.field[x_pos][y_pos]
```

Dies sind die Methoden zum Überprüfen, ob das Spielbrett bzw. die Spalte voll ist, dies geschieht durch das überprüfen, ob Nullen in der jeweiligen Spalten oder dem Spielbrett vorhanden sind.

```
def is_board_full(self) -> bool:
    """ Check, if the board is full """
    if not any(self.EMPTY in i for i in self.field):
        return True
    else:
        return False

def is_col_full(self, x_pos) -> bool:
    """ Tell if column is fully occupied """
    if self.EMPTY in self.field[x_pos]:
        return False
```

```
        else:
            return True
```

In diesen letzten beiden Methoden der Klasse wird überprüft, ob jemand gewonnen hat und mit welchen Steinen gewonnen wurde. Das Überprüfen funktioniert, in dem jedes Feld auf horizontale, vertikale und diagonale Gewinnfälle überprüft werden. Die Methode überprüft nur die Felder, in den die unterschiedlichen Gewinnmöglichkeiten möglich sind, zudem ist es auch möglich, andere Spielbretter zu überprüfen, indem man ein passendes Spielfeld der Methode übergibt, dies ist für die KI von Relevanz.

```
    def is_winning(self) -> bool:
        """ Tell if winning was achieved by a player """
        if self.get_winning_positions(self.field):
            return True
        else:
            return False

    def get_winning_positions(self, board=None) -> tuple:
        if board is None:
            board = self.field

        """ Check, if there is a win-situation and return the position """
        # Check horizontally
        for column in range(self.X_MAX - 3):
            for row in range(self.Y_MAX):
                if board[column][row] == board[column + 1][row] == board[column + 2][row]
                    return ((column, row), (column + 1, row), (column + 2, row), (column

        # Check vertically
        for row in range(self.Y_MAX - 3):
            for column in range(self.X_MAX):
                if board[column][row] == board[column][row + 1] == board[column][row + 2]
                    return ((column, row), (column, row + 1), (column, row + 2), (column,

        # Skip diagonal checks if column count is less than 4
        if self.X_MAX < 4:
            return False

        # Check up-diagonally
        for column in range(self.X_MAX - 3):
            for row in range(self.Y_MAX - 3):
                if board[column][row] == board[column + 1][row + 1] == board[column + 2][[
                    return ((column, row), (column + 1, row + 1), (column + 2, row + 2),

        # Check down-diagonally
        for column in range(self.X_MAX - 3):
            for row in range(3, self.Y_MAX):
                if board[column][row] == board[column + 1][row - 1] == board[column + 2][[
                    return ((column, row), (column + 1, row - 1), (column + 2, row - 2),
```

```
    return False
```

Fazit

Abschließend kann man sagen, dass das Projekt "Vier Gewinnt" eines der Projekte ist, welches man als fertiges Spiel später auch öfters zum Spielen benutzen kann. Das Entwickeln des Spiels hat einige Zeit in Anspruch genommen, jedoch zum Schluss beim Testen des Spiels uns besonders viel Spaß bereitet.