Python, základní kameny až skály II

Tomáš Svoboda

B4B33RPH, 2019-10-15

funkce pravé a modifikátory

Write code in Python 3.3

(drag lower right corner to resize code editor)

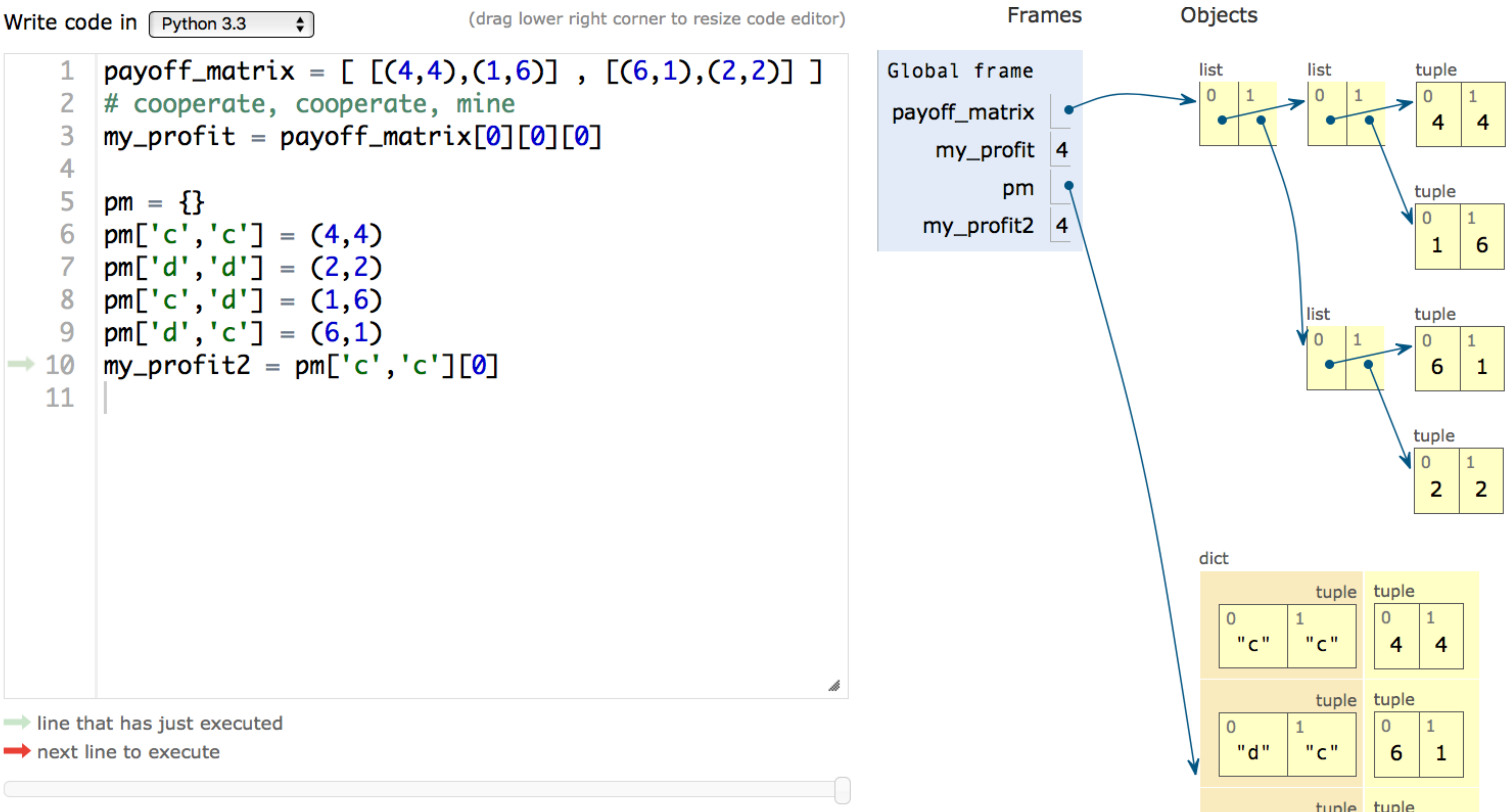
```
def increment_pure_function(x):
    v = []
    for item in x:
        v.append(item+1)
    return(v)

def increment_modifier(x):
    for i in range(len(x)):
        x[i] = x[i]+1
    return(x)

a = [1,2,3]
b = increment_pure_function(a)
print(a,',',b)
c = increment_modifier(a)
print(a,',',b,',',c)
```

Print output (drag lower right corner to resize)

```
[1, 2, 3] , [2, 3, 4]
[2, 3, 4] , [2, 3, 4] , [2, 3, 4]
```

Frames
Objects
Global frame
increment_pure_function
increment_modifier
a
b
c
function
increment_pure_function(x)
function
increment_modifier(x)
list
0 1 2
2 3 4
list
0 1 2
2 3 4

dict - tuples as keys



Write code in Python 3.3 (drag lower right corner to resize code editor)

```python
payoff_matrix = [ [(4,4),(1,6)] , [(6,1),(2,2)] ]
# cooperate, cooperate, mine
my_profit = payoff_matrix[0][0][0]

pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)
my_profit2 = pm['c','c'][0]
```

line that has just executed
next line to execute

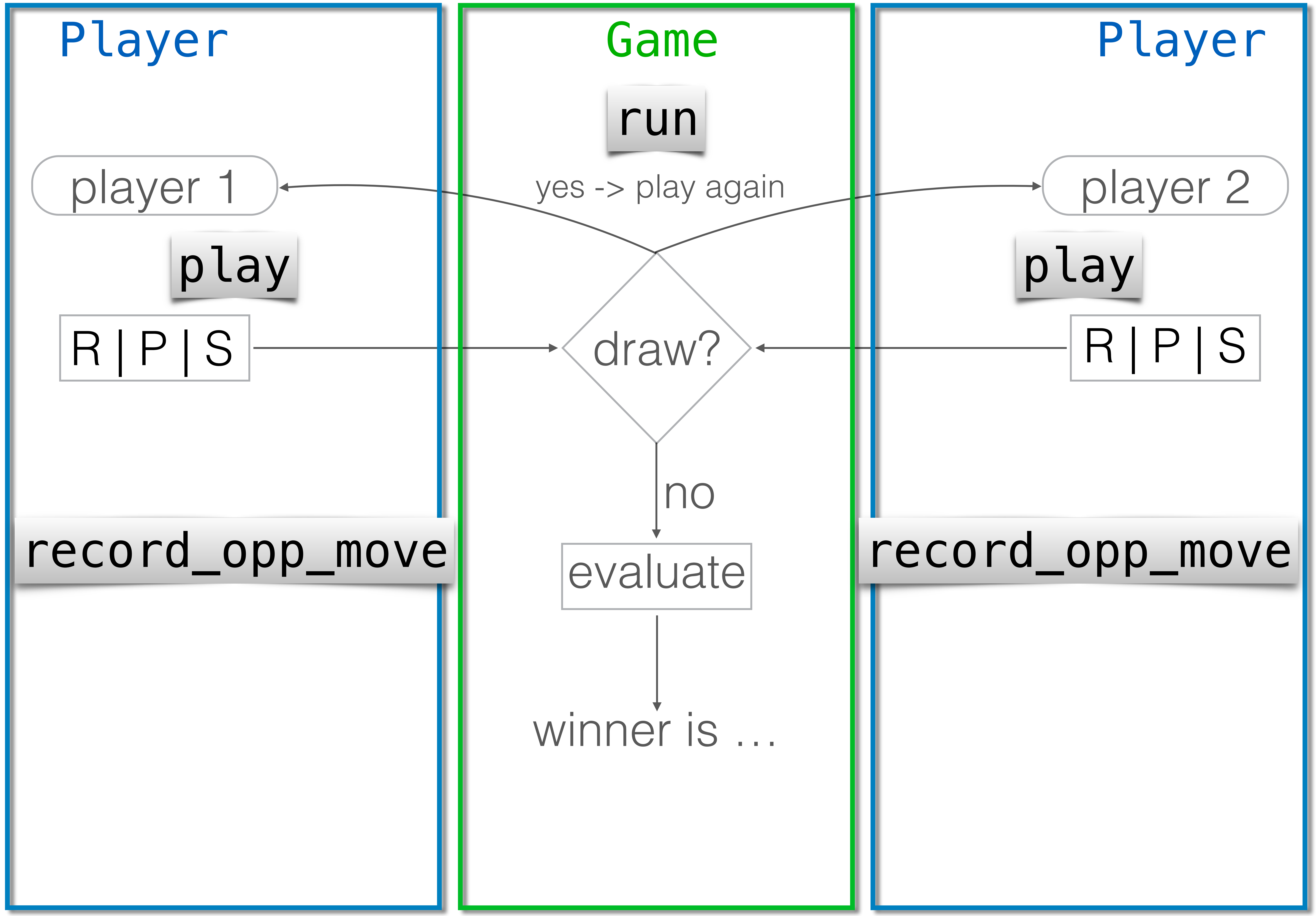
dictionary loops …

```python
pm = {}
pm['c','c'] = (4,4)
pm['d','d'] = (2,2)
pm['c','d'] = (1,6)
pm['d','c'] = (6,1)

for key in pm:
    print(key, pm[key])

for key,value in pm.items():
    print(key, value)
```

http://openbookproject.net/thinkcs/python/english3e/dictionaries.html

skládání objektů, dědění

- vylepšíme trochu hráče R-P-S
- ukážeme si na příkladu hráče piškvorek (tic-tac-toe)
- live-coding-session


Player
Game
Player
run
player 1
yes -> play again
player 2
play
play
R | P | S
draw?
R | P | S
no
record_opp_move
evaluate
record_opp_move
winner is …

vylepšení hráče R-P-S

- Společné do základní třídy BasePlayer
- Rozdílné strategie jako různí hráči
- Oddělení (zapozdření) technikalit správy hráčovy paměti
- A také vylepšíme hru na typ 2 za 3
- a ukážeme možnost jak řídit ukecanost běhu hry

Skládání objektů

```python
class Memory:
    def __init__(self, size=None):
        self.cyclic = size is not(None)
        if self.cyclic:
            self.data = size*[None]
        else:
            self.data = []
        self.frequencies = dict()
        for move in POSSIBLE_MOVES:
            self.frequencies[move] = 0

    def update(self, move):
        assert move in POSSIBLE_MOVES, str(move)+' is not acceptable'
        self.frequencies[move] += 1
        if self.cyclic:
            del self.data[-1]
        self.data.insert(0, move)

    def get_most_frequent(self):
        return max(self.frequencies, key=self.frequencies.get)

class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

```

Skládání objektů

```python
class Memory:
    def __init__(self, size=None):
        self.cyclic = size is not(None)
        if self.cyclic:
            self.data = size*[None]
        else:
            self.data = []
        self.frequencies = dict()
        for move in POSSIBLE_MOVES:
            self.frequencies[move] = 0

    def update(self, move):
        assert move in POSSIBLE_MOVES, str(move)+' is not acceptable'
        self.frequencies[move] += 1
        if self.cyclic:
            del self.data[-1]
        self.data.insert(0, move)

    def get_most_frequent(self):
        return max(self.frequencies, key=self.frequencies.get)

class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

```

Dědění

nahrazení metody

```python
class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

    def find_move(self):
        raise NotImplementedError

    def __str__(self):
        return 'Player '+ self.__class__.__name__

class RandomPlayer(BasePlayer):
    def find_move(self):
        return random.choice(POSSIBLE_MOVES)
```

Dědění

nahrazení metody

```python
class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

    def find_move(self):
        raise NotImplementedError

    def __str__(self):
        return 'Player '+ self.__class__.__name__

class RandomPlayer(BasePlayer):
    def find_move(self):
        return random.choice(POSSIBLE_MOVES)
```

Dědění

nahrazení metody

```python
class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

    def find_move(self):
        raise NotImplementedError

    def __str__(self):
        return 'Player '+ self.__class__.__name__

class RandomPlayer(BasePlayer):
    def find_move(self):
        return random.choice(POSSIBLE_MOVES)
```

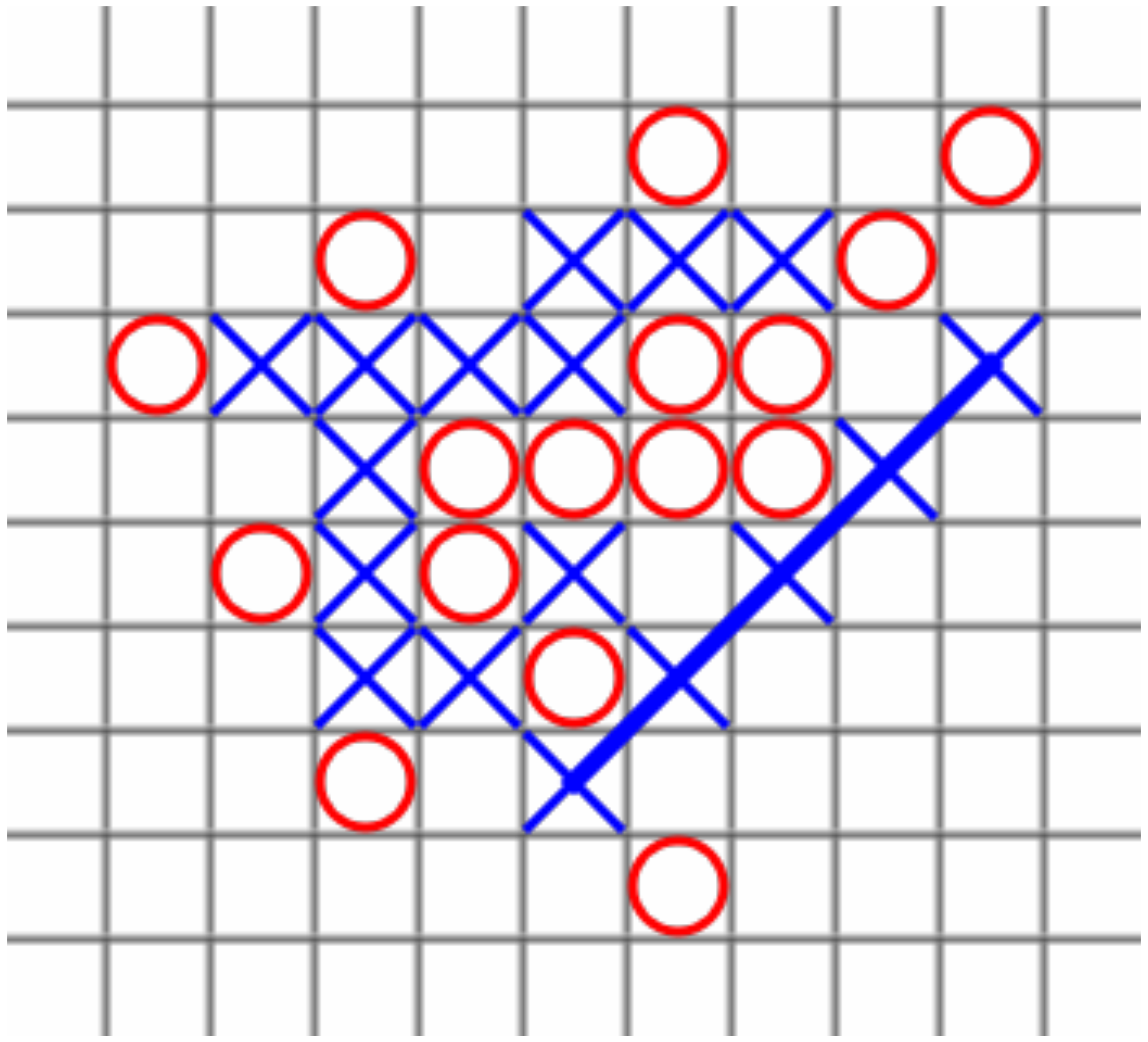
Volání metod rodičovské třídy/objektu

```python
class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

    def find_move(self):
        raise NotImplementedError

    def __str__(self):
        return 'Player '+ self.__class__.__name__

class SkewedRandom(BasePlayer):
    def __init__(self, weights):
        super().__init__()
        self.weights = weights

    def find_move(self):
        return random.choices(POSSIBLE_MOVES, self.weights)[0]
```

Volání metod rodičovské třídy/objektu

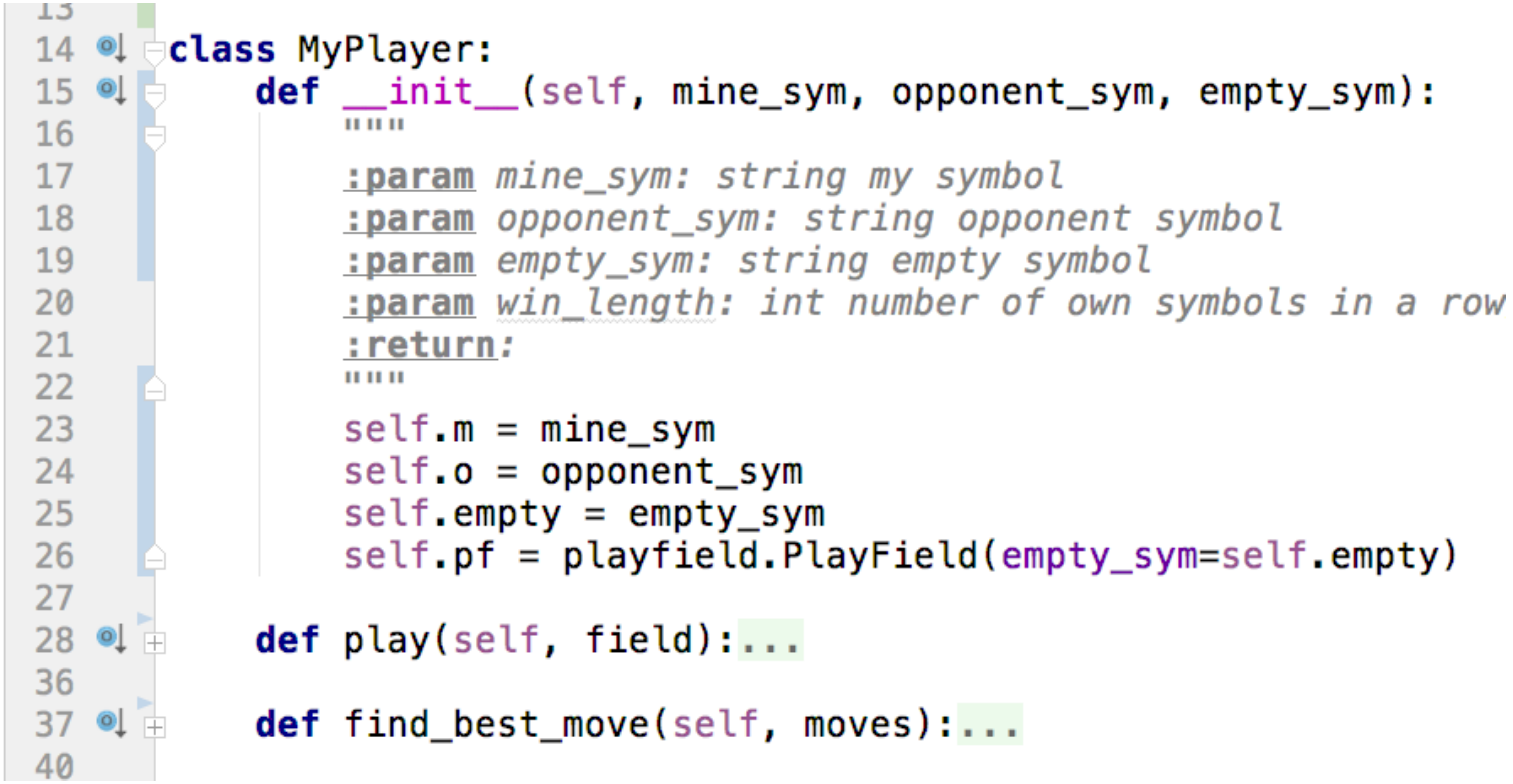
```python
class BasePlayer:
    def __init__(self):
        self.my_moves = Memory(100)
        self.opp_moves = Memory(100)

    def record_opp_move(self, move):
        self.opp_moves.update(move)

    def play(self):
        my_move = self.find_move()
        self.my_moves.update(my_move)
        return my_move

    def find_move(self):
        raise NotImplementedError

    def __str__(self):
        return 'Player '+ self.__class__.__name__

class SkewedRandom(BasePlayer):
    def __init__(self, weights):
        super().__init__()
        self.weights = weights

    def find_move(self):
        return random.choices(POSSIBLE_MOVES, self.weights)[0]
```

Piškvorky, tic-tac-toe, m,n,k-game



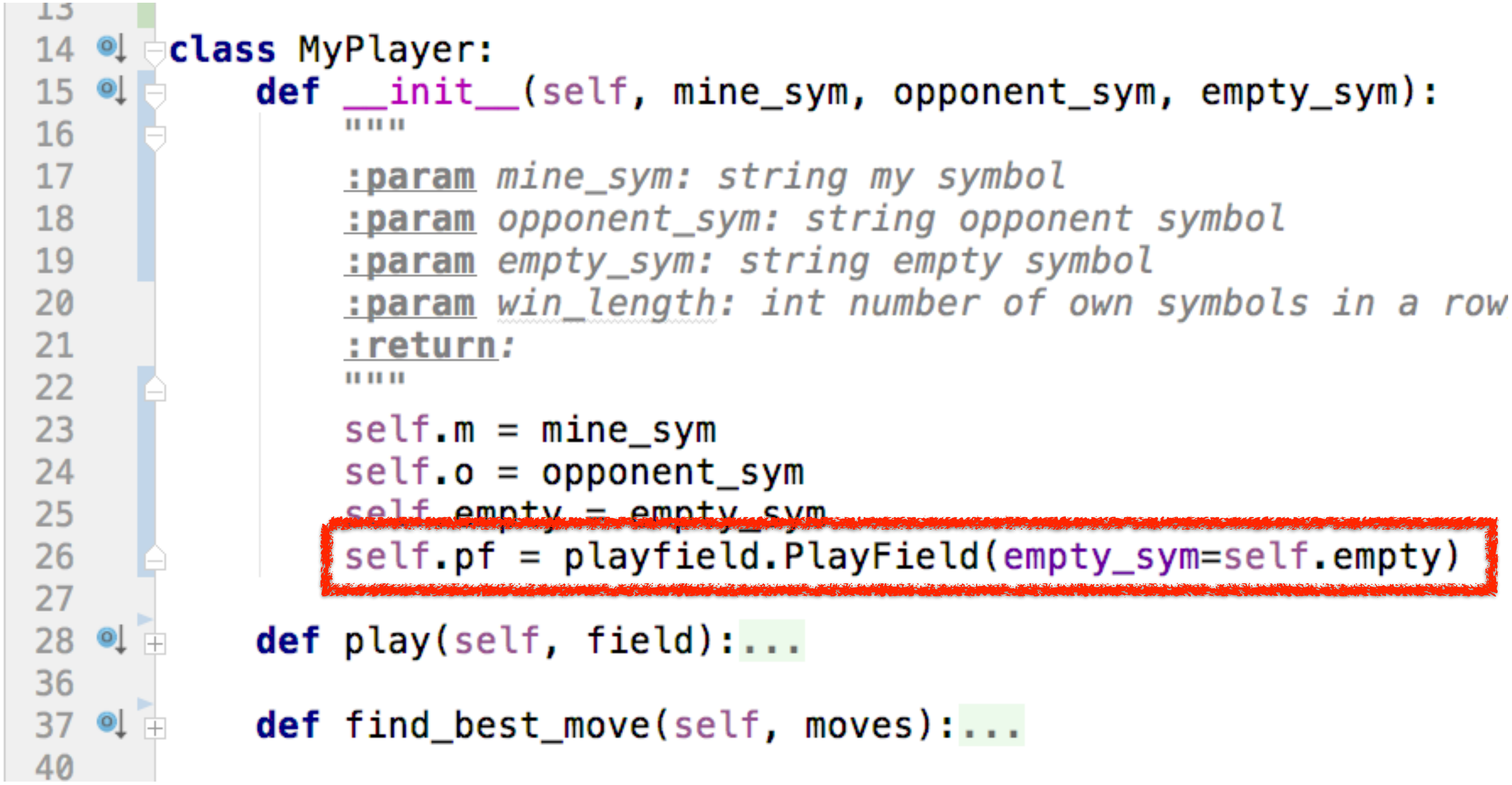
By DaBler - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5506040

skládání



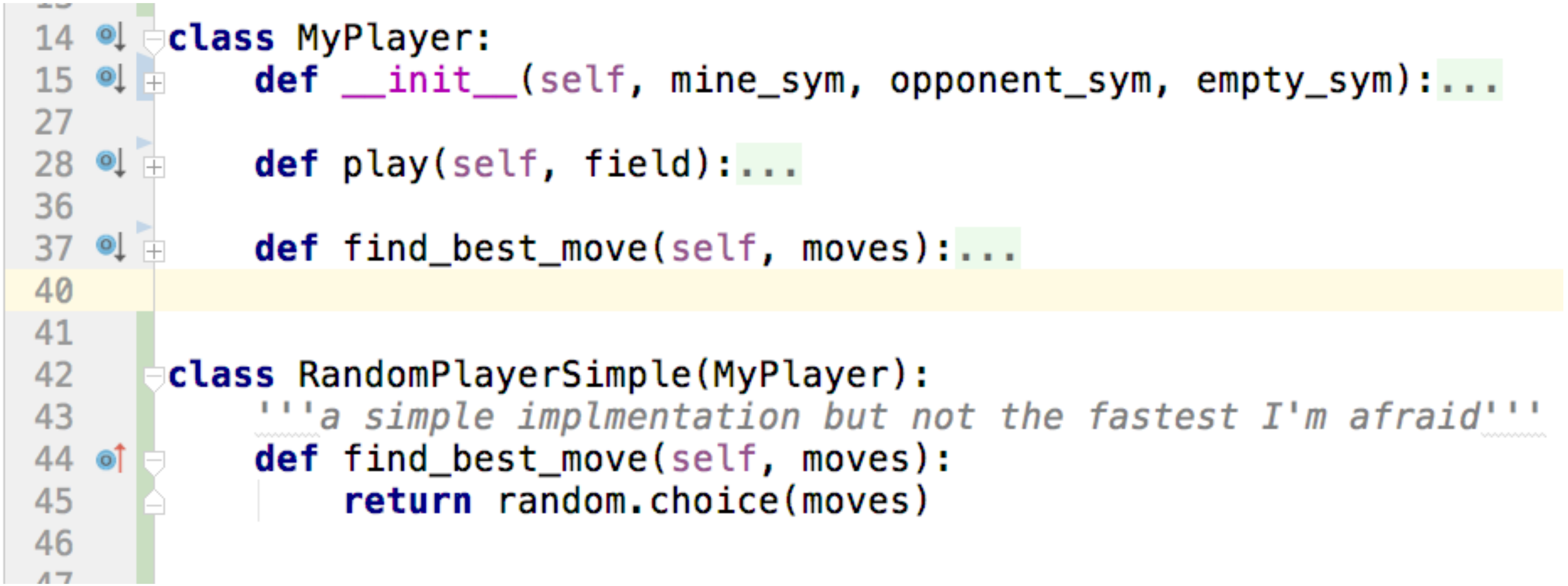
```python
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):
        """

        :param mine_sym: string my symbol
        :param opponent_sym: string opponent symbol
        :param empty_sym: string empty symbol
        :param win_length: int number of own symbols in a row
        :return:
        """
        self.m = mine_sym
        self.o = opponent_sym
        self.empty = empty_sym
        self.pf = playfield.PlayField(empty_sym=self.empty)

    def play(self, field):...

    def find_best_move(self, moves):...
```

skládání



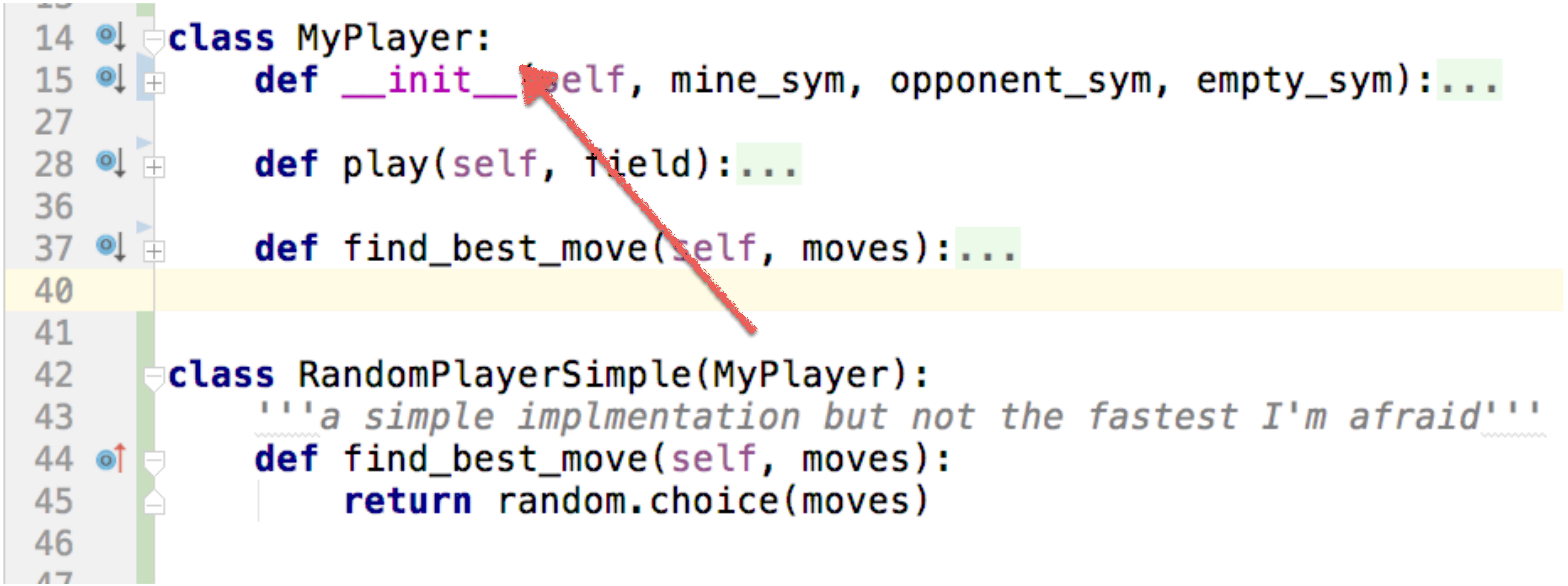
```
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):
        """

        :param mine_sym: string my symbol
        :param opponent_sym: string opponent symbol
        :param empty_sym: string empty symbol
        :param win_length: int number of own symbols in a row
        :return:
        """

        self.m = mine_sym
        self.o = opponent_sym
        self.empty = empty_sym
        self.pf = playfield.PlayField(empty_sym=self.empty)

    def play(self, field):...

    def find_best_move(self, moves):...
```

dědění



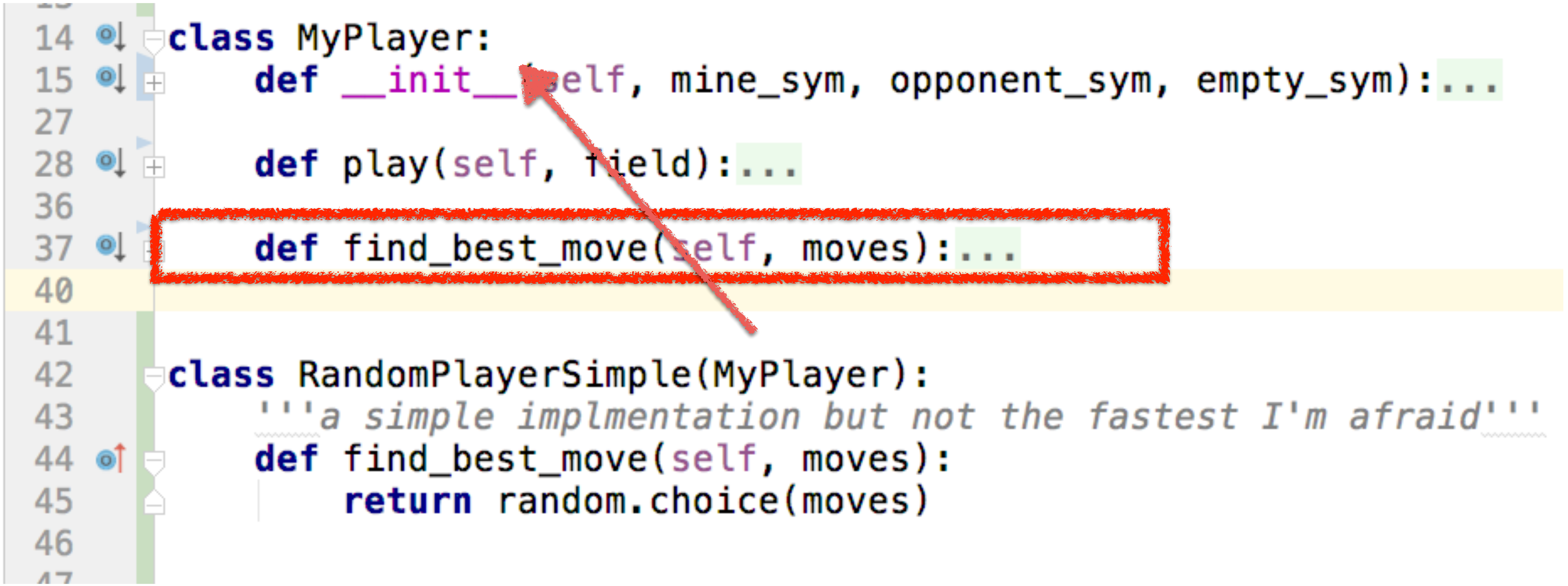
```
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):...

    def play(self, field):...

    def find_best_move(self, moves):...


class RandomPlayerSimple(MyPlayer):
    '''a simple implmentation but not the fastest I'm afraid'''
    def find_best_move(self, moves):
        return random.choice(moves)

```

dědění



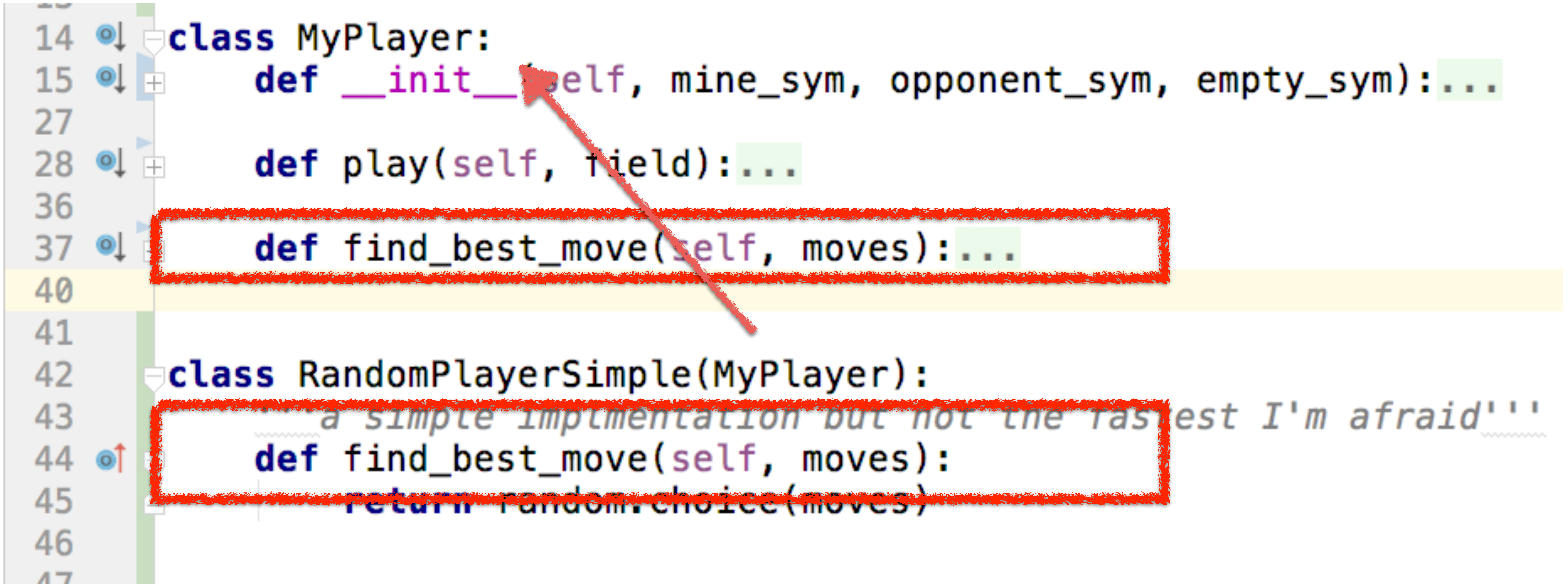
```
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):...

    def play(self, field):...

    def find_best_move(self, moves):...


class RandomPlayerSimple(MyPlayer):
    '''a simple implmentation but not the fastest I'm afraid'''
    def find_best_move(self, moves):
        return random.choice(moves)

```

dědění



```
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):...

    def play(self, field):...

    def find_best_move(self, moves):...


class RandomPlayerSimple(MyPlayer):
    '''a simple implmentation but not the fastest I'm afraid'''
    def find_best_move(self, moves):
        return random.choice(moves)

```

dědění



```
class MyPlayer:
    def __init__(self, mine_sym, opponent_sym, empty_sym):...

    def play(self, field):...

    def find_best_move(self, moves):...


class RandomPlayerSimple(MyPlayer):
    '''a simple implmentation but not the fastest I'm afraid'''
    def find_best_move(self, moves):
        return random.choice(moves)
```