

Python, základní kameny až skály II

Tomáš Svoboda

B4B33RPH, 2020-10-13, 2020-10-20

slovníky, dictionary, dict()

```
1 d = {}
2 d[1] = 'a'
3 d[0] = 'b'
4 d[2] = 'c'
5
6 print('for key in d:')
7 for key in d:
8     print(key, d[key])
9
10 print('for key, value in d.items():')
11 for key, value in d.items():
12     print(key, value)
13
14 print('for key in sorted(d.keys()):')
15 for key in sorted(d.keys()):
16     print(key, d[key])
```

dict - tuples as keys

Write code in Python 3.3

(drag lower right corner to resize code editor)

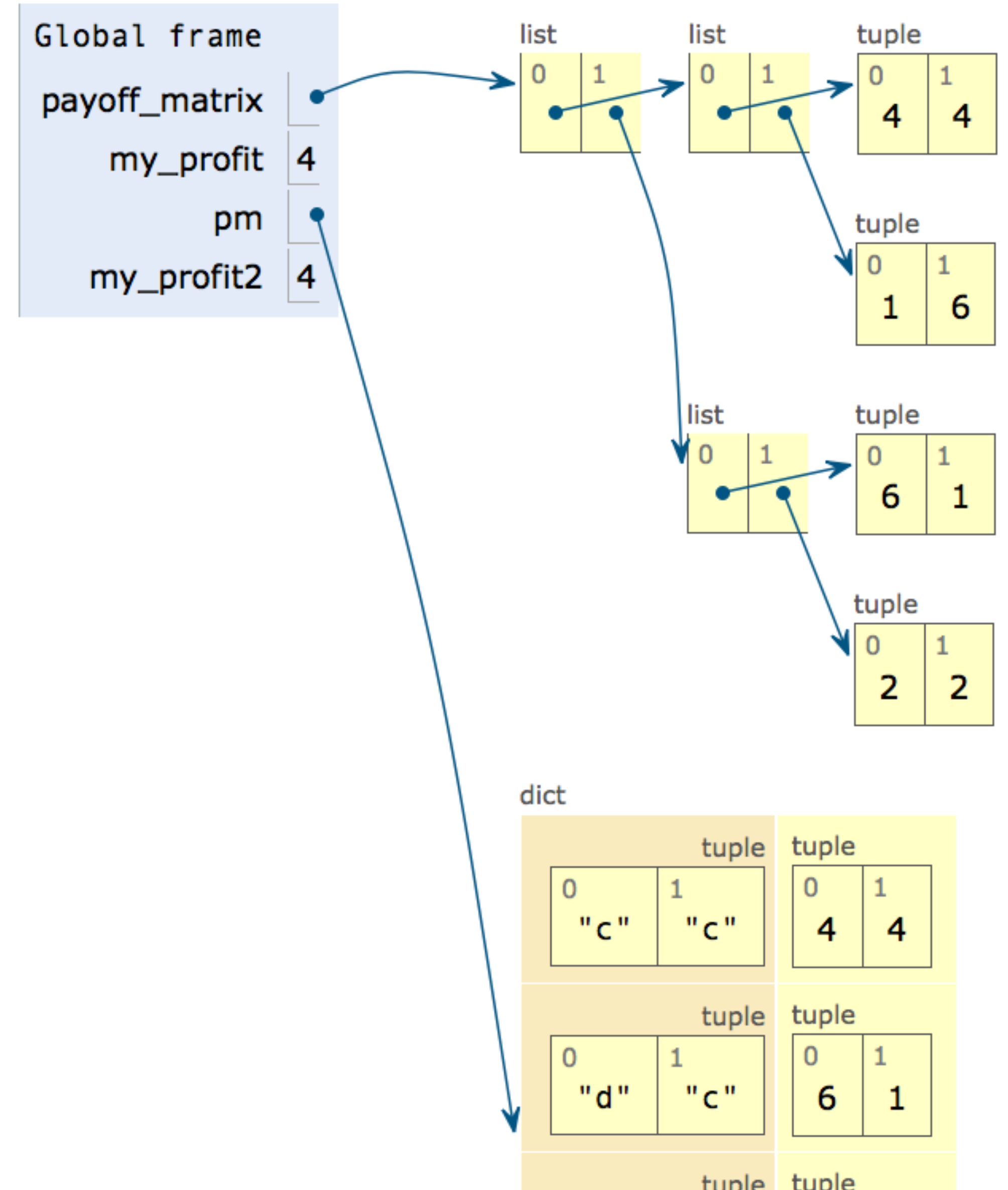
```
1 payoff_matrix = [ [(4,4),(1,6)] , [(6,1),(2,2)] ]
2 # cooperate, cooperate, mine
3 my_profit = payoff_matrix[0][0][0]
4
5 pm = {}
6 pm['c','c'] = (4,4)
7 pm['d','d'] = (2,2)
8 pm['c','d'] = (1,6)
9 pm['d','c'] = (6,1)
10 my_profit2 = pm['c','c'][0]
11
```

→ line that has just executed

→ next line to execute

Frames

Objects

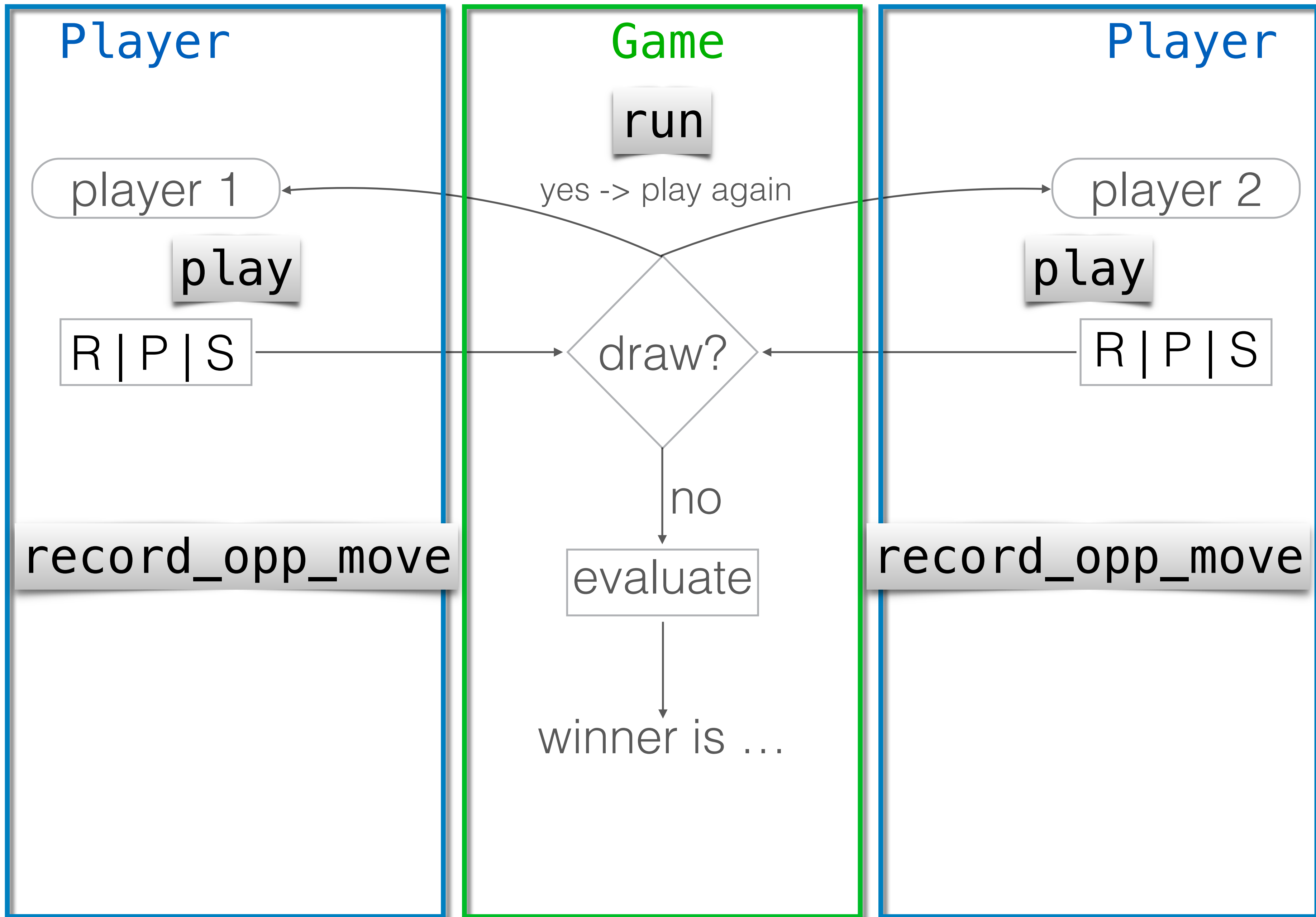


dictionary loops ...

```
1 pm = {}
2 pm['c', 'c'] = (4, 4)
3 pm['d', 'd'] = (2, 2)
4 pm['c', 'd'] = (1, 6)
5 pm['d', 'c'] = (6, 1)
6
7 for key in pm:
8     print(key, pm[key])
9
10 for key, value in pm.items():
11     print(key, value)
```

skládání objektů, dědění

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



vylepšení hráče R-P-S

- Společné do základní třídy BasePlayer
- Rozdílné strategie jako různí hráči
- Oddělení (zapouzdření) technikalit správy hráčovy paměti
- A také vylepšíme hru na typ 2 za 3
- a ukážeme možnost jak řídit ukecanost běhu hry
- *# bude se hodit při implementaci Reversi hráče*

BasePlayer, RandomPlayer ...

- `play()`
- `record_opp_move(move)`
- `ConstantPlayer`, `RandomPlayer`
- `SkewedPlayer`
- `SmartPlayer`

Lepší paměť

- `__init__(size)` # chceme omezit velikost
- `update(element)` # remember?
- kontrola přípustnosti vkládaného prvku
- `get_most_frequent()` # jaký je nejčastější prvek

Skládání objektů

```
1 class Memory:
2     def __init__(self, size=None):
3         self.cyclic = size is not None
4         if self.cyclic:
5             self.data = size*[None]
6         else:
7             self.data = []
8         self.frequencies = dict()
9         for move in POSSIBLE_MOVES:
10            self.frequencies[move] = 0
11
12    def update(self, move):
13        assert move in POSSIBLE_MOVES, str(move)+' is not acceptable'
14        self.frequencies[move] += 1
15        if self.cyclic:
16            del self.data[-1]
17            self.data.insert(0, move)
18
19    def get_most_frequent(self):
20        return max(self.frequencies, key=self.frequencies.get)
21
22 class BasePlayer:
23     def __init__(self):
24         self.my_moves = Memory(100)
25         self.opp_moves = Memory(100)
26
27     def record_opp_move(self, move):
28         self.opp_moves.update(move)
29
30     def play(self):
31         my_move = self.find_move()
32         self.my_moves.update(my_move)
33         return my_move
34
```

Skládání objektů

```
1 class Memory:
2     def __init__(self, size=None):
3         self.cyclic = size is not None
4         if self.cyclic:
5             self.data = size*[None]
6         else:
7             self.data = []
8         self.frequencies = dict()
9         for move in POSSIBLE_MOVES:
10            self.frequencies[move] = 0
11
12    def update(self, move):
13        assert move in POSSIBLE_MOVES, str(move)+' is not acceptable'
14        self.frequencies[move] += 1
15        if self.cyclic:
16            del self.data[-1]
17            self.data.insert(0, move)
18
19    def get_most_frequent(self):
20        return max(self.frequencies, key=self.frequencies.get)
21
22 class BasePlayer:
23     def __init__(self):
24         self.my_moves = Memory(100)
25         self.opp_moves = Memory(100)
26
27    def record_opp_move(self, move):
28        self.opp_moves.update(move)
29
30    def play(self):
31        my_move = self.find_move()
32        self.my_moves.update(my_move)
33        return my_move
34
```

Dědění

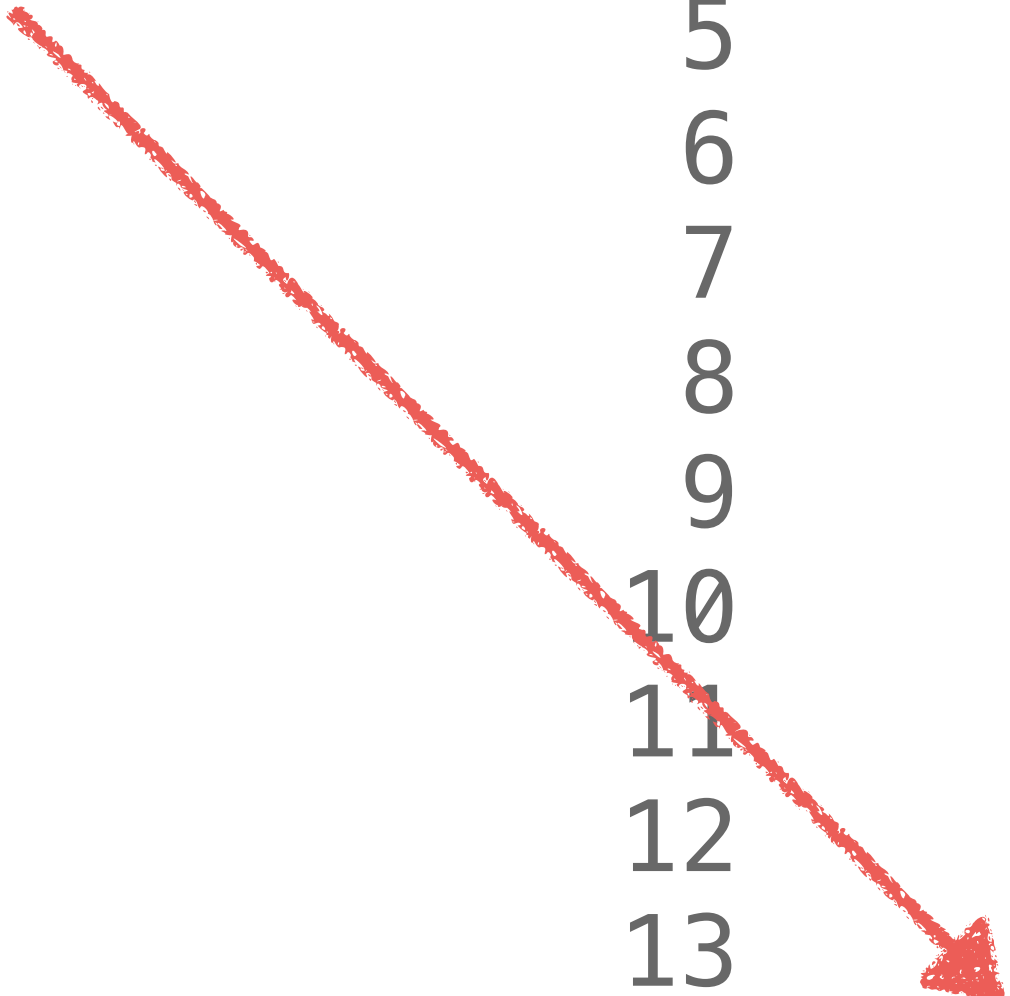
nahrazení metody

```
1 class BasePlayer:
2     def __init__(self):
3         self.my_moves = Memory(100)
4         self.opp_moves = Memory(100)
5
6     def record_opp_move(self, move):
7         self.opp_moves.update(move)
8
9     def play(self):
10        my_move = self.find_move()
11        self.my_moves.update(my_move)
12        return my_move
13
14    def find_move(self):
15        raise NotImplementedError
16
17    def __str__(self):
18        return 'Player ' + self.__class__.__name__
19
20 class RandomPlayer(BasePlayer):
21     def find_move(self):
22         return random.choice(POSSIBLE_MOVES)
```

Dědění

nahrazení metody

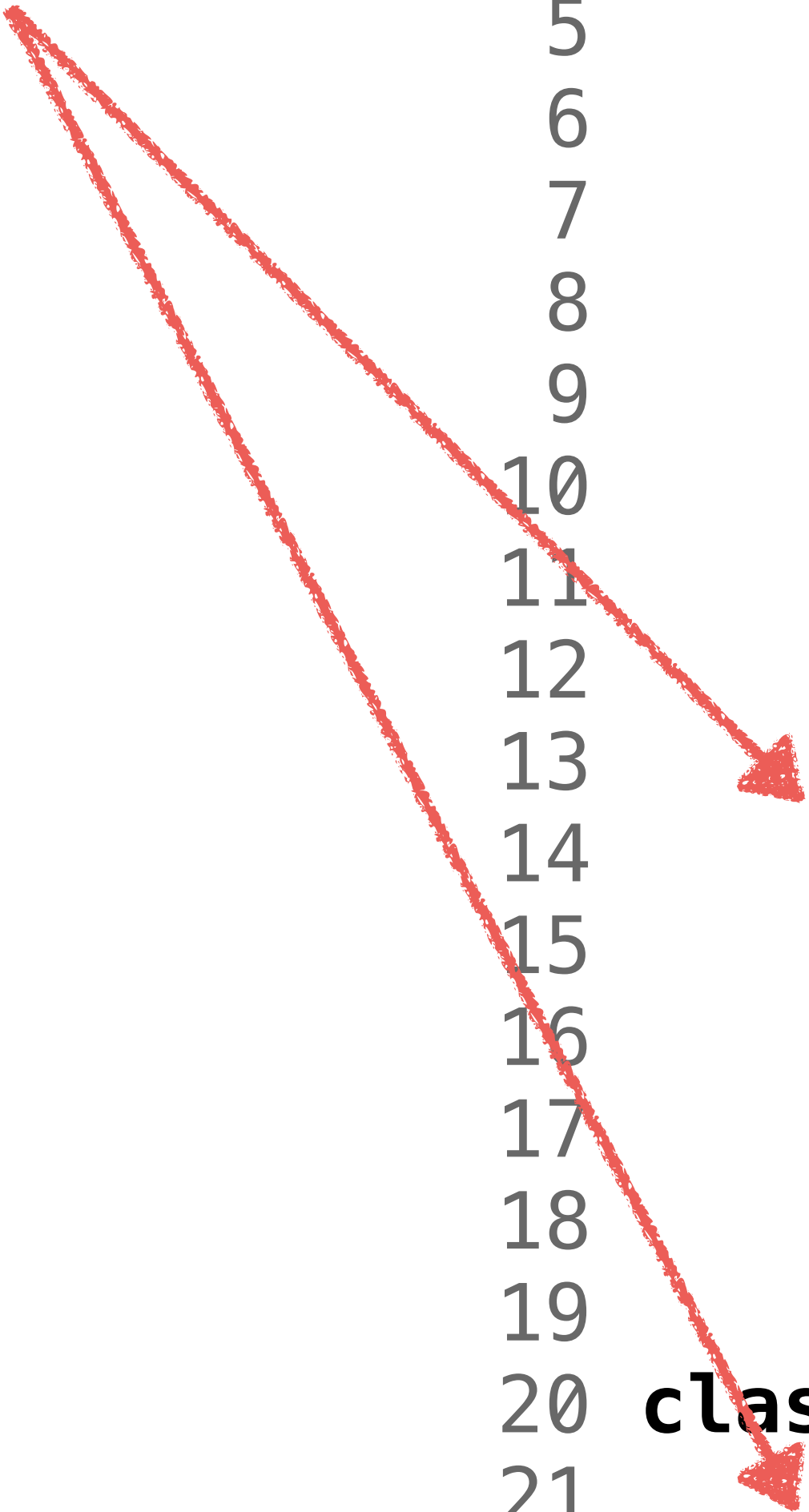
```
1 class BasePlayer:
2     def __init__(self):
3         self.my_moves = Memory(100)
4         self.opp_moves = Memory(100)
5
6     def record_opp_move(self, move):
7         self.opp_moves.update(move)
8
9     def play(self):
10        my_move = self.find_move()
11        self.my_moves.update(my_move)
12        return my_move
13
14    def find_move(self):
15        raise NotImplementedError
16
17    def __str__(self):
18        return 'Player ' + self.__class__.__name__
19
20 class RandomPlayer(BasePlayer):
21     def find_move(self):
22         return random.choice(POSSIBLE_MOVES)
```



Dědění

nahrazení metody

```
1 class BasePlayer:
2     def __init__(self):
3         self.my_moves = Memory(100)
4         self.opp_moves = Memory(100)
5
6     def record_opp_move(self, move):
7         self.opp_moves.update(move)
8
9     def play(self):
10        my_move = self.find_move()
11        self.my_moves.update(my_move)
12        return my_move
13
14    def find_move(self):
15        raise NotImplementedError
16
17    def __str__(self):
18        return 'Player ' + self.__class__.__name__
19
20 class RandomPlayer(BasePlayer):
21     def find_move(self):
22         return random.choice(POSSIBLE_MOVES)
```



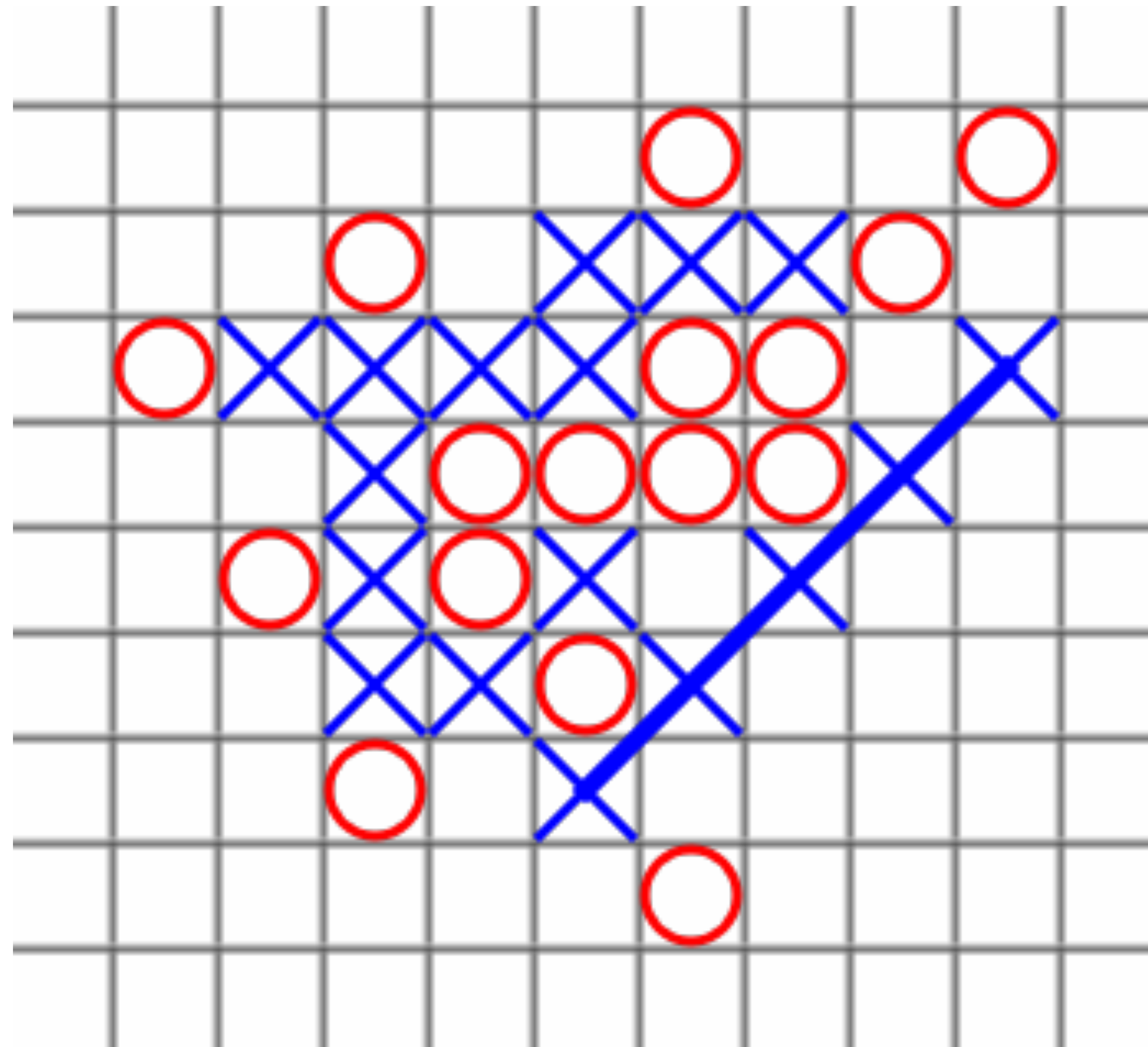
Volání metod rodičovské třídy/objektu

```
1 class BasePlayer:
2     def __init__(self):
3         self.my_moves = Memory(100)
4         self.opp_moves = Memory(100)
5
6     def record_opp_move(self, move):
7         self.opp_moves.update(move)
8
9     def play(self):
10        my_move = self.find_move()
11        self.my_moves.update(my_move)
12        return my_move
13
14    def find_move(self):
15        raise NotImplementedError
16
17    def __str__(self):
18        return 'Player ' + self.__class__.__name__
19
20 class SkewedRandom(BasePlayer):
21    def __init__(self, weights):
22        super().__init__()
23        self.weights = weights
24
25    def find_move(self):
26        return random.choices(POSSIBLE_MOVES, self.weights)[0]
```

Volání metod rodičovské třídy/objektu

```
1 class BasePlayer:
2     def __init__(self):
3         self.my_moves = Memory(100)
4         self.opp_moves = Memory(100)
5
6     def record_opp_move(self, move):
7         self.opp_moves.update(move)
8
9     def play(self):
10        my_move = self.find_move()
11        self.my_moves.update(my_move)
12        return my_move
13
14    def find_move(self):
15        raise NotImplementedError
16
17    def __str__(self):
18        return 'Player ' + self.__class__.__name__
19
20 class SkewedRandom(BasePlayer):
21     def __init__(self, weights):
22         super().__init__()
23         self.weights = weights
24
25     def find_move(self):
26         return random.choices(POSSIBLE_MOVES, self.weights)[0]
```


Piškvorky, tic-tac-toe, m,n,k-game



skládání

```
13
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym):
16         """
17         :param mine_sym: string my symbol
18         :param opponent_sym: string opponent symbol
19         :param empty_sym: string empty symbol
20         :param win_length: int number of own symbols in a row
21         :return:
22         """
23         self.m = mine_sym
24         self.o = opponent_sym
25         self.empty = empty_sym
26         self.pf = playfield.PlayField(empty_sym=self.empty)
27
28     def play(self, field):...
36
37     def find_best_move(self, moves):...
40
```

skládání

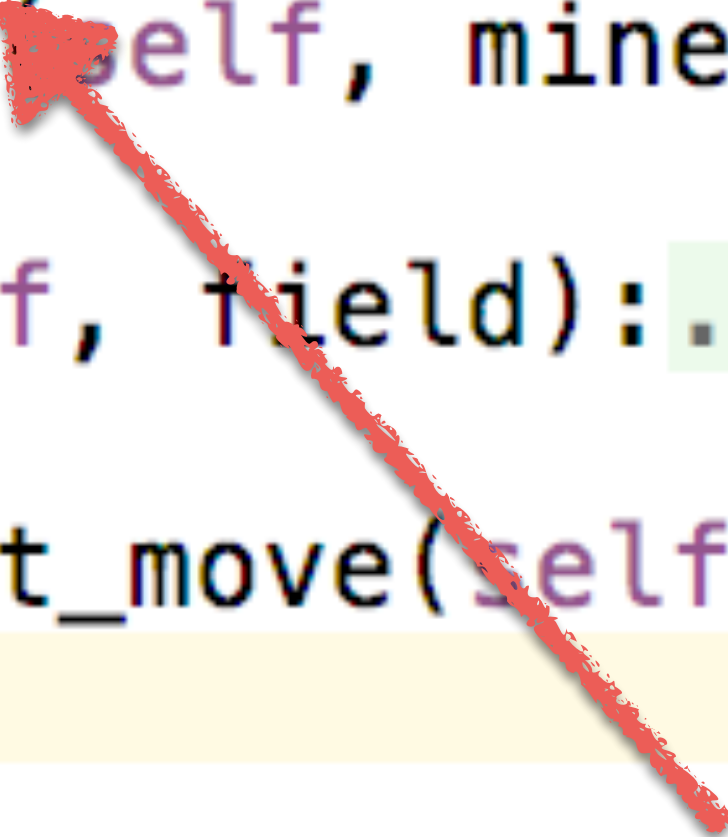
```
13
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym):
16         """
17         :param mine_sym: string my symbol
18         :param opponent_sym: string opponent symbol
19         :param empty_sym: string empty symbol
20         :param win_length: int number of own symbols in a row
21         :return:
22         """
23         self.m = mine_sym
24         self.o = opponent_sym
25         self.empty = empty_sym
26         self.pf = playfield.PlayField(empty_sym=self.empty)
27
28     def play(self, field):...
36
37     def find_best_move(self, moves):...
40
```

dědění

```
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym): ...
27
28     def play(self, field): ...
36
37     def find_best_move(self, moves): ...
40
41
42 class RandomPlayerSimple(MyPlayer):
43     '''a simple implmentation but not the fastest I'm afraid'''
44     def find_best_move(self, moves):
45         return random.choice(moves)
46
47
```

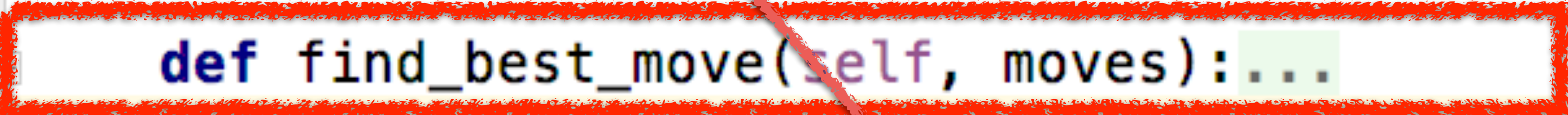
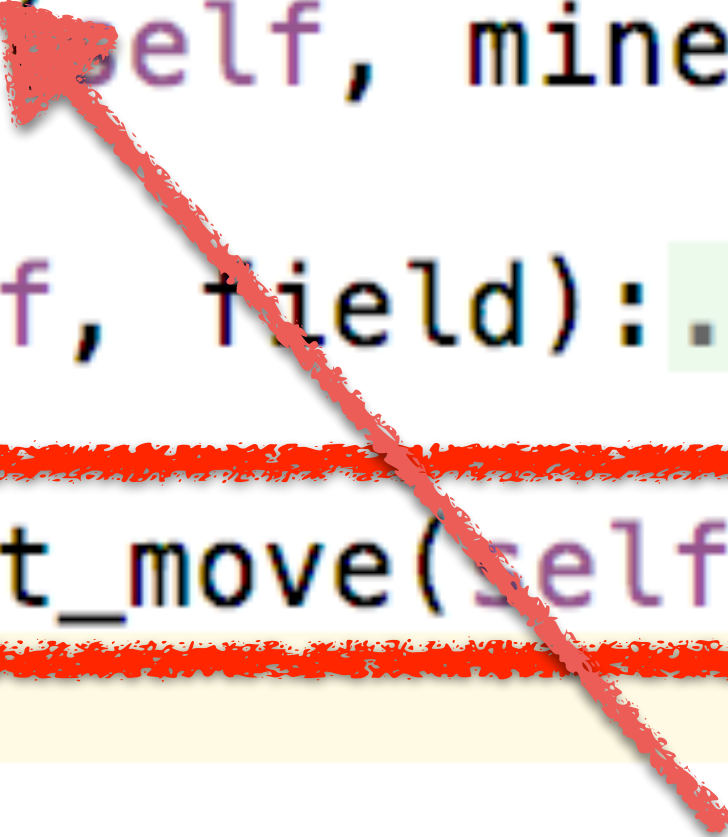
dědění

```
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym): ...
27
28     def play(self, field): ...
36
37     def find_best_move(self, moves): ...
40
41
42 class RandomPlayerSimple(MyPlayer):
43     '''a simple implmentation but not the fastest I'm afraid'''
44     def find_best_move(self, moves):
45         return random.choice(moves)
46
47
```



dědění

```
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym): ...
27
28     def play(self, field): ...
36
37     def find_best_move(self, moves): ...
40
41
42 class RandomPlayerSimple(MyPlayer):
43     '''a simple implmentation but not the fastest I'm afraid'''
44     def find_best_move(self, moves):
45         return random.choice(moves)
46
47
```



dědění

```
14 class MyPlayer:
15     def __init__(self, mine_sym, opponent_sym, empty_sym): ...
27
28     def play(self, field): ...
36
37     def find_best_move(self, moves): ...
40
41
42 class RandomPlayerSimple(MyPlayer):
43     """a simple implementation but not the fastest I'm afraid"""
44     def find_best_move(self, moves):
45         return random.choice(moves)
46
47
```

