Python, základní kameny až skály III

Tomáš Svoboda

B4B33RPH, 2018-10-30

live coding sessions

Dnes …

- slovníky, malá ukázka grafického výstupu
- generátorová notace (list comprehensions)
- logické funkce
- generátory
- set, frozen set

slajdy nejsou vše

- klíčové podněty
- kódy z přednášky na hraní - refaktorujte, zlepšujte, přidávejte funkcionalitu
- http://cw.fel.cvut.cz/wiki/courses/b4b33rph/literatura, https://stackoverflow.com,
- kolega Mareda čeká úterky podvečer
- Programujte!

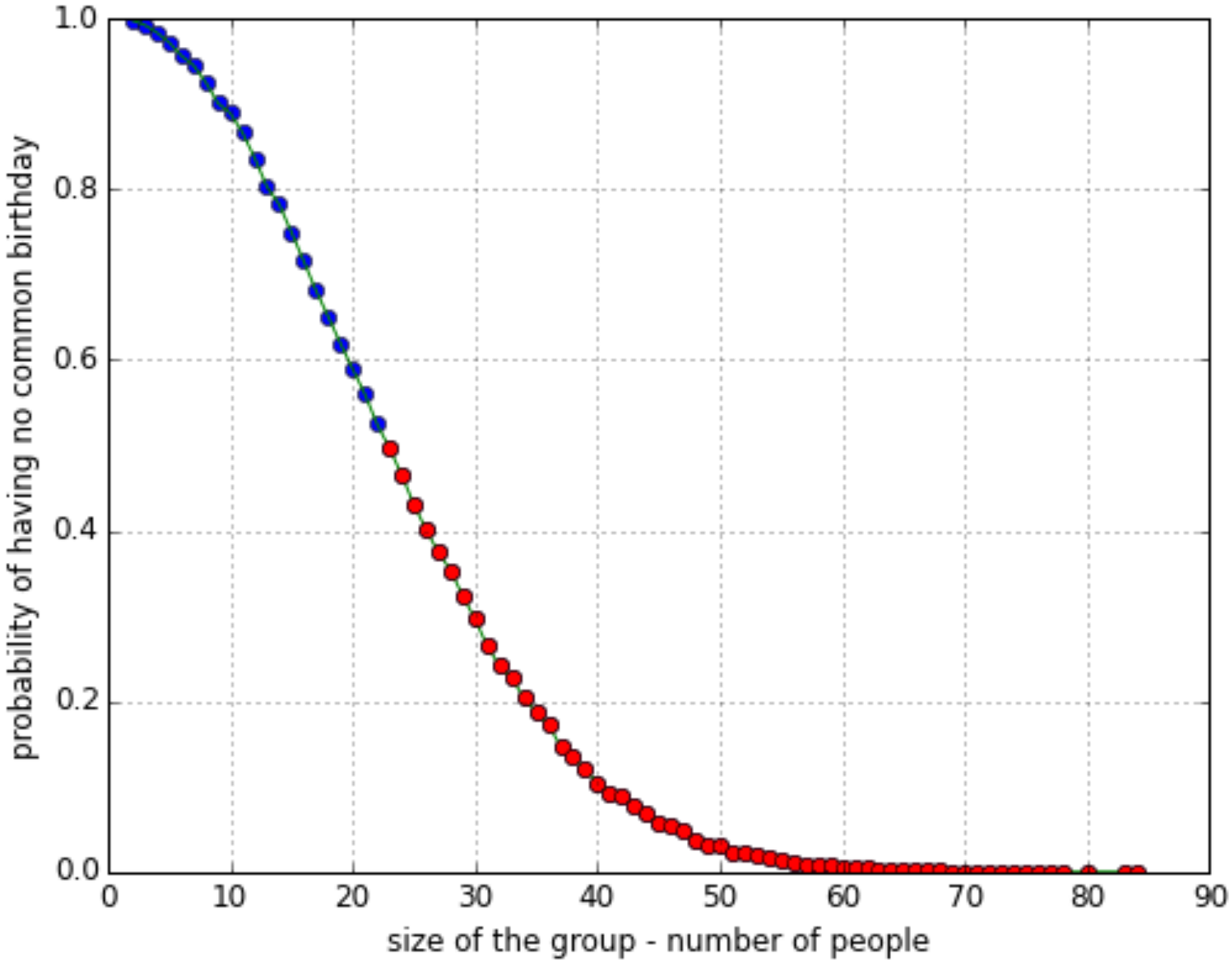
slovníky key:value

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

birthday problem

- Skupina N osob
- Jak velká je pravděpodobnost že alespoň jedno datum narození není unikátní?
- Pro jak velké N začne být pravděpodobnější, že alespoň jedny narozeniny jsou společné?

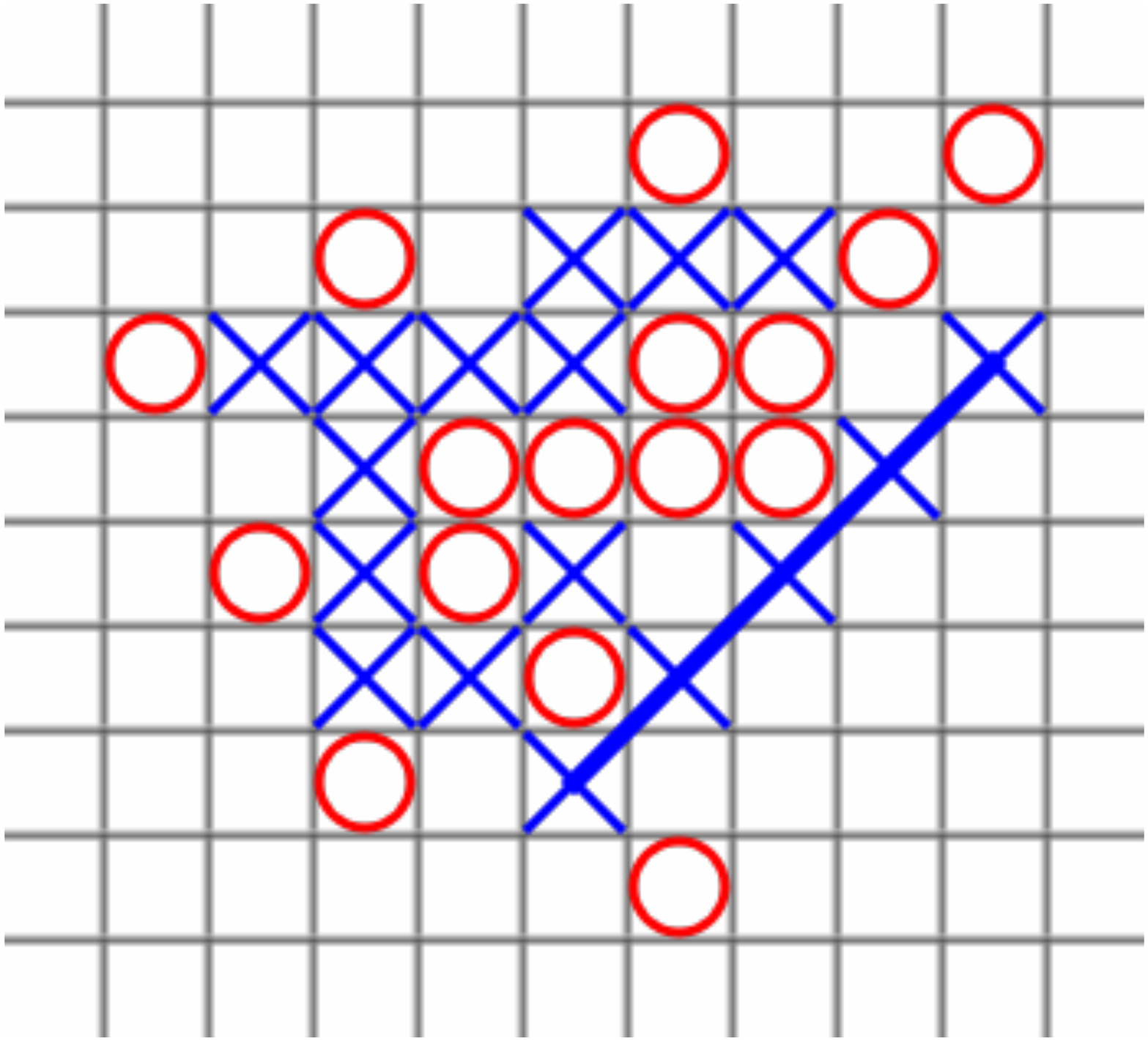
matplotlib.pyplot



generátorová notace

```
lp = {key:value for key,value in nc.items() if value < 0.5}
```

zpátky k piškvorkám



Logické funkce

- `is_inside`
- `is_winning`
- `is_empty`
- `is_full`

- Vracejí `True` nebo `False`

- Zpřehledňují hlavní ideu algoritmu

yield, generátory - proč?

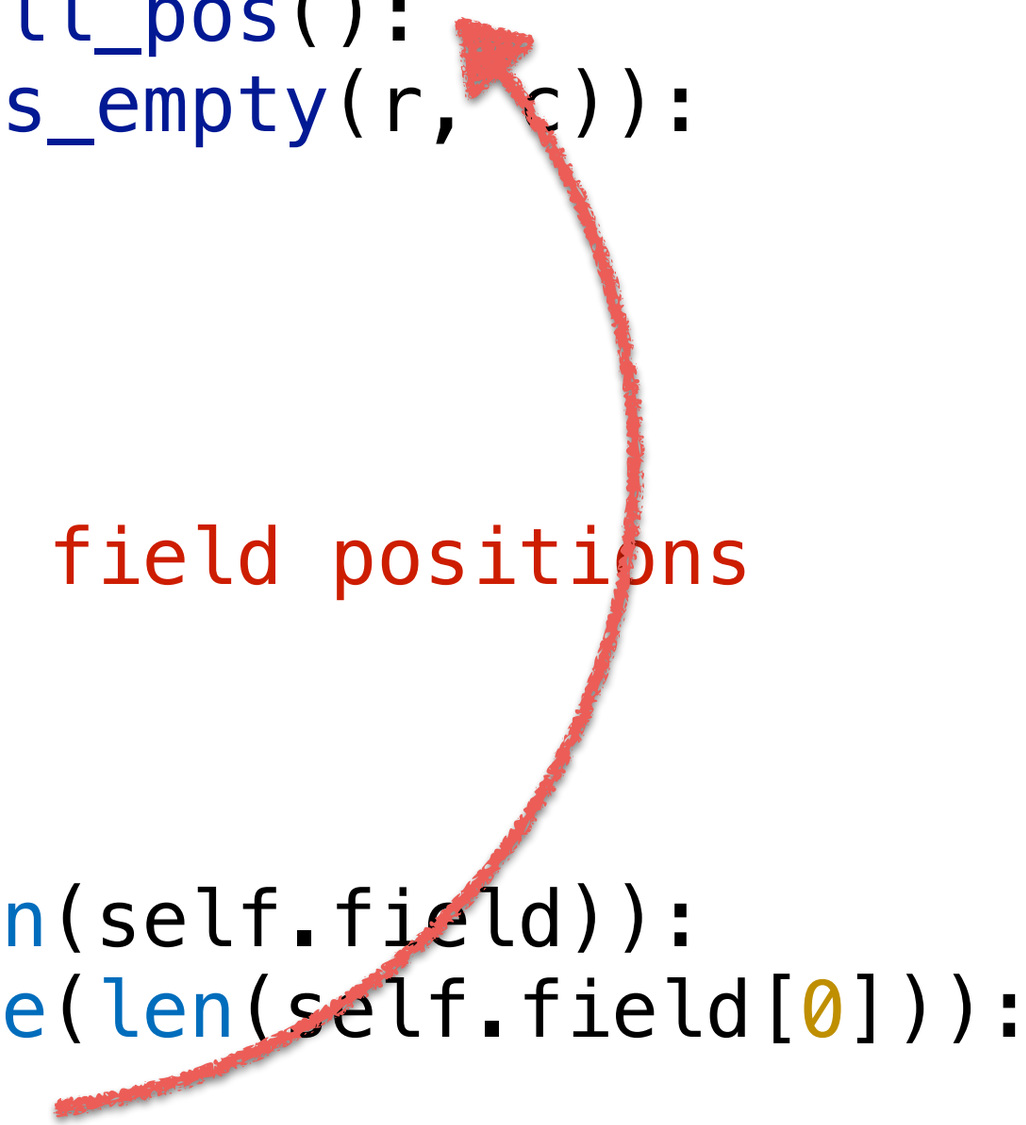
```
1 def is_winning_too_generous(self):
2     for r in range(self.size):
3         for c in range(self.size):
4             if self.is_empty(r,c):
5                 continue
6             for direction in self.directions[0:4]:
7                 if self.is_dir_winning(r,c,direction):
8                     return True
9     return False
```

pro všechny neprázdné pozice

```
1 def is_winning_too_generous(self):
2     for r,c in self.non_empty_pos():
3         for direction in self.directions[0:4]:
4             if self.is_dir_winning(r,c,direction):
5                 return True
6     return False
```

yield místo return

```
1  def non_empty_pos(self):
2      """generate all non-empty positions"""
3      for r,c in self.all_pos():
4          if not(self.is_empty(r, c)):
5              yield r,c
6
7  def all_pos(self):
8      """
9      generator for all field positions
10     :param self:
11     :yield: r,c
12     """
13     for r in range(len(self.field)):
14         for c in range(len(self.field[0])):
15             yield r,c
```



`range` is not a generator

- Často ho tak používáme
- `<class 'range'>`
- Zkusme si ho představit jako *líný* seznam (lazy list)

set, frozenset

```
1 >>> a = set([1,2,3,4,5,4])
2 >>> b = set([1,2,3,6])
3 >>> a | b
4 {1, 2, 3, 4, 5, 6}
5 >>> a & b
6 {1, 2, 3}
```

set - mutable

frozenset - immutable

sekvence pozic v jednom směru

```
1 def get_seq(self, start_pos, direction, sym):
2     r,c = start_pos
3     positions = [(r,c)]
4     r,c = r+direction['r'], c+direction['c']
5     while self.is_inside(r,c) and sym==self.field[r][c]:
6         positions.append((r, c))
7         r,c = r+direction['r'], c+direction['c']
8     return frozenset(positions)
```

kompletní sekvence

```
1  def get_all_sequences(self, pos, sym):
2      """
3      :param pos: r,c tuple
4      :param sym: symbol of the player that moves
5      :return: sequence:lenght dictionary
6      """
7      forward_sequences = []
8      sequences = {}
9      for direction in self.directions[0:4]: # forward
10         seq = self.get_seq(pos, direction, sym)
11         forward_sequences.append(seq)
12     for i in range(4): # backward directions
13         seq = self.get_seq(pos, self.directions[i + 4], sym)
14         complete_sequence = forward_sequences[i] | seq
15         sequences[complete_sequence] = len(complete_sequence)
16     return sequences
17
```

sjednocení sekvence vpřed a vzad