

Kámen-nůžky-papír

Tomáš Svoboda

Centrum strojového vnímání, Katedra kybernetiky
Fakulta elektrotechnická, České vysoké učení technické

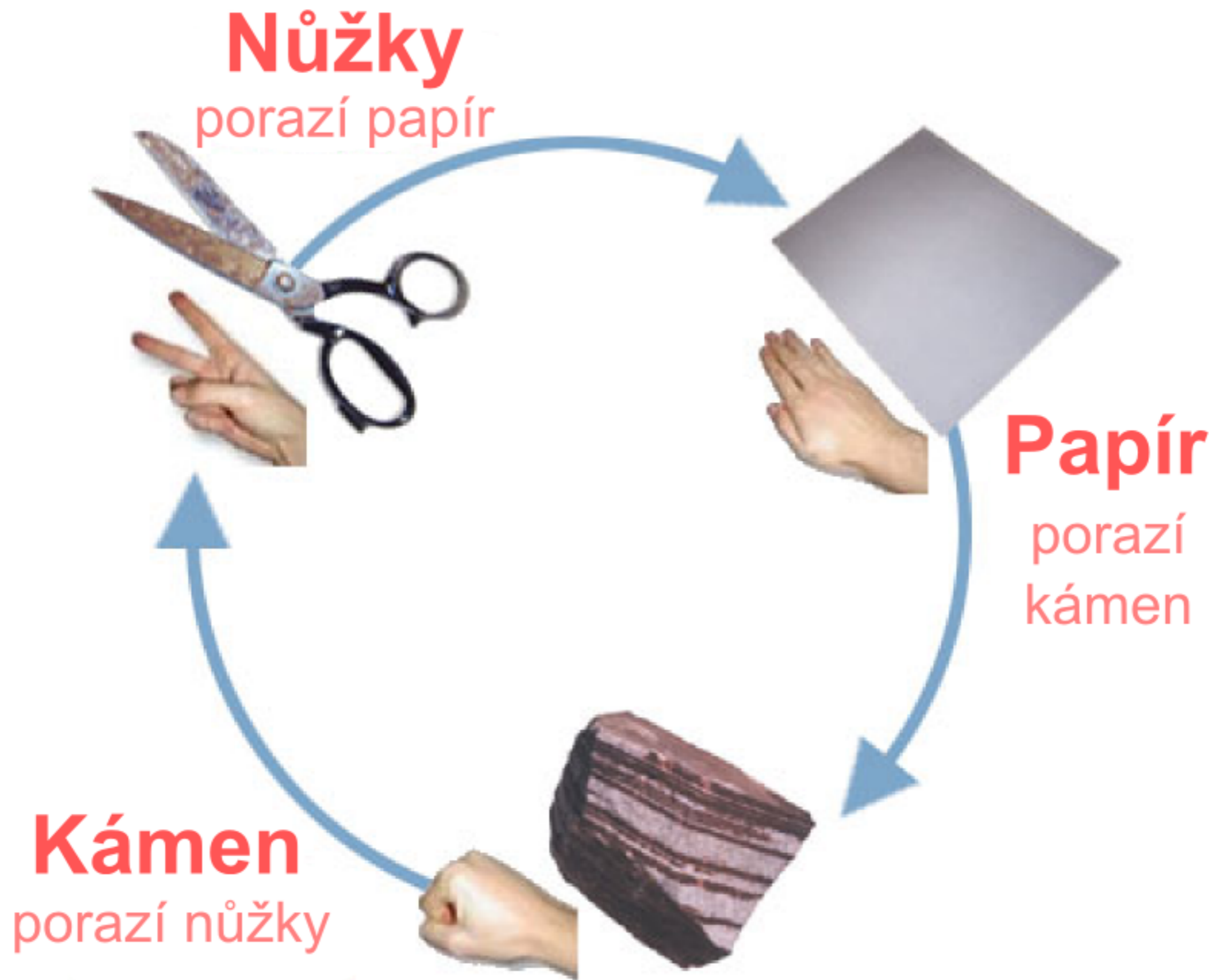
Studijní program: [Otevřená informatika](#)
2017-10-10

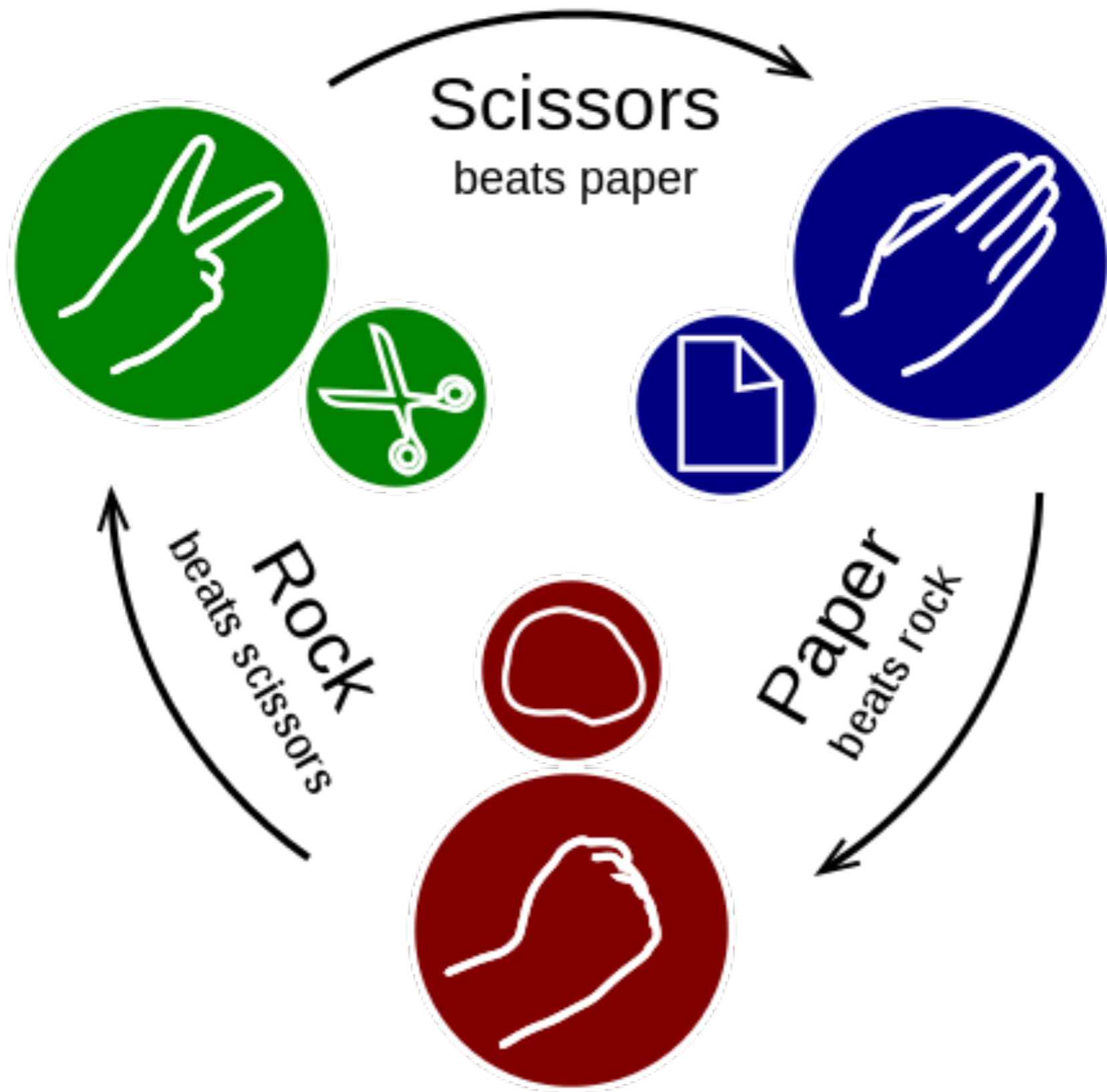
než začneme ...

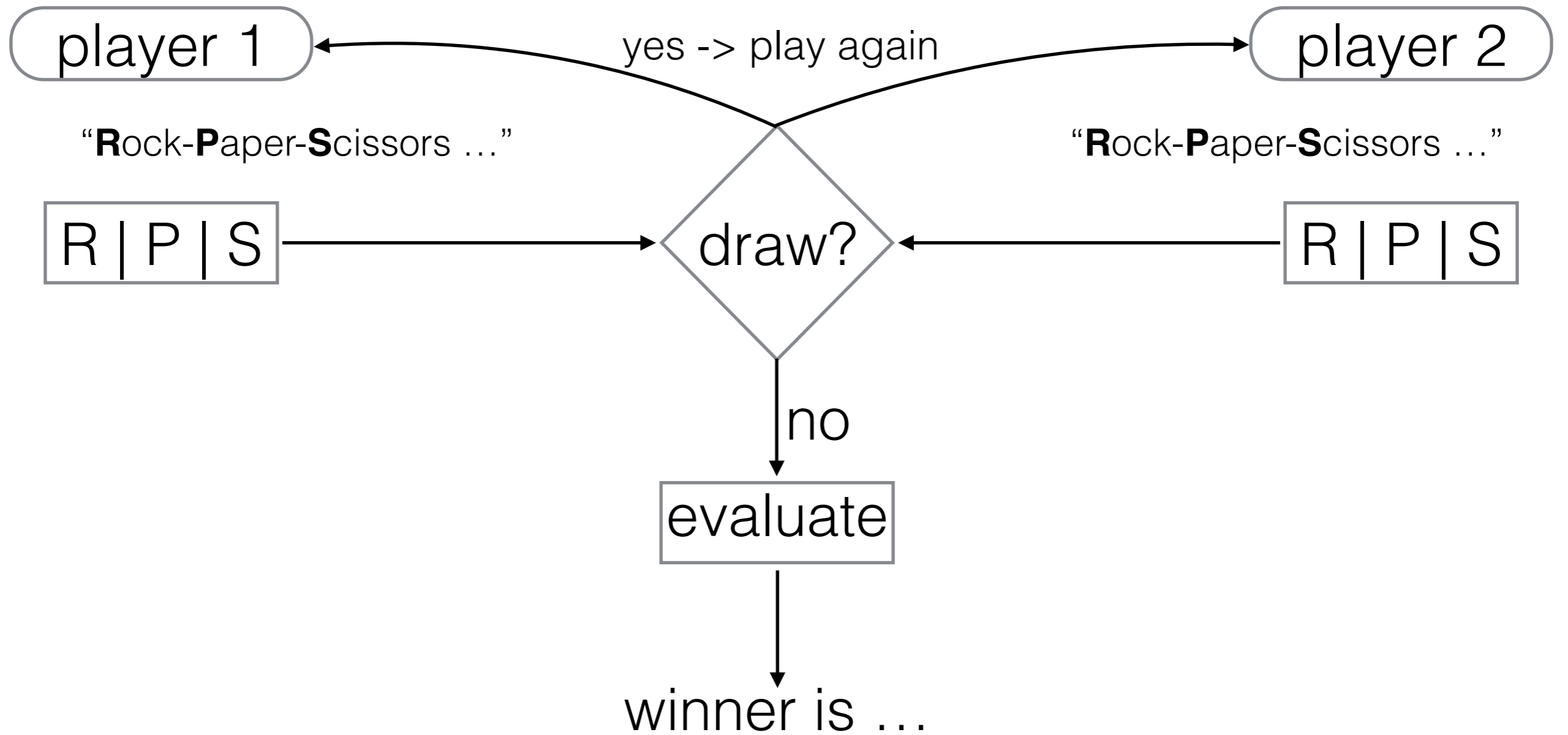
- dívka jménem Lenka
- doplňková cvičení, úterky 13:00-14:20, KN:E-220, kolega D. Mareša

Lenka - výsledky

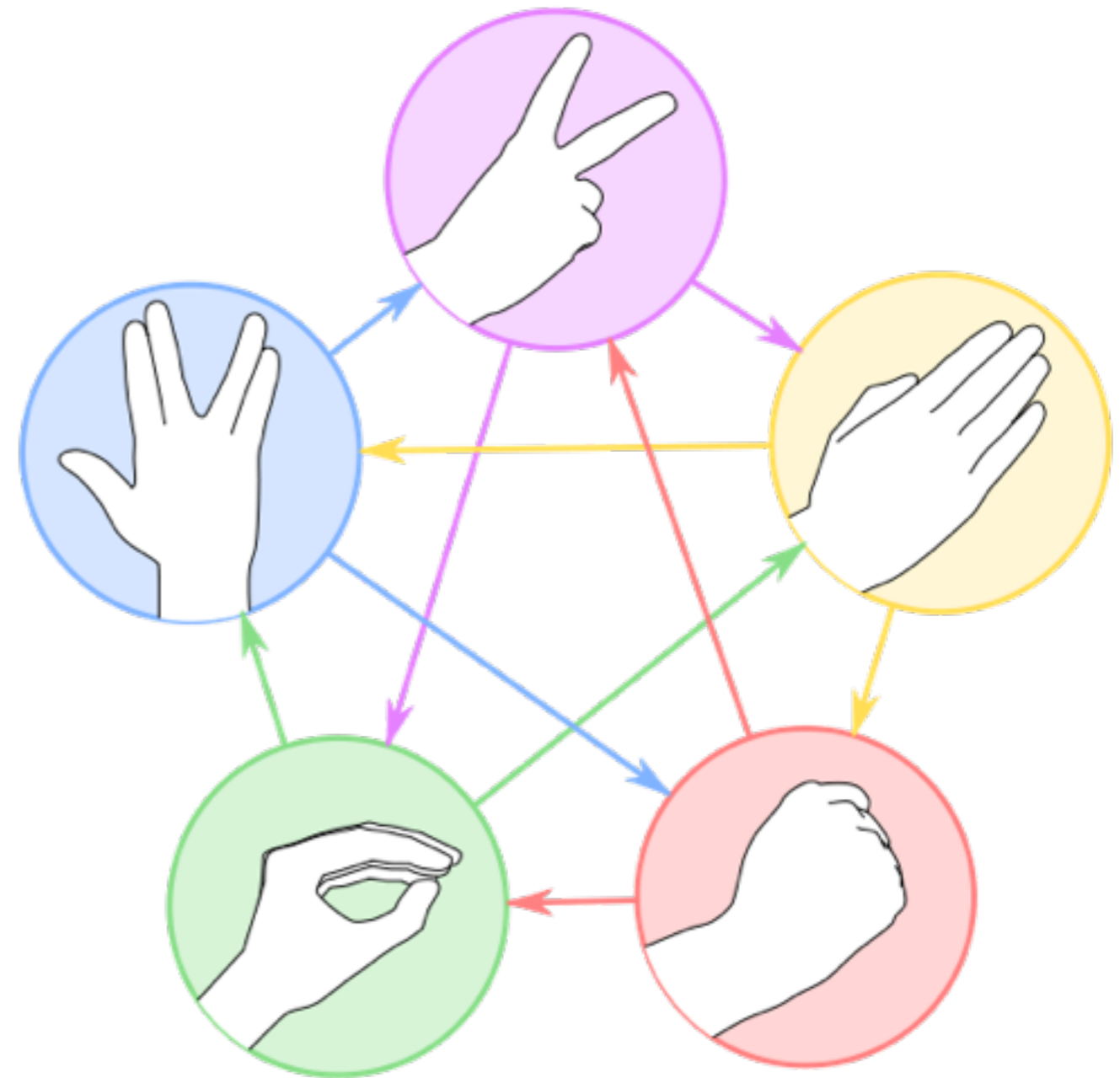
- Odevzdaných řešení: 185
- Syntakticky správně: 183
 - velkými písmeny: 36
 - malými písmeny: 147
- Správná odpověď: 86







Vyšší level



Player

player 1

play

R | P | S

Game

run

yes -> play again

draw?

no

evaluate

winner is ...

Player

player 2

play

R | P | S

playerdummy.py

obecná definice hráče

```
1 class MyPlayer:
2     '''A very dummy player (always returns R)'''
3
4     def play(self):
5         return 'R'
6
7 if __name__ == "__main__":
8     p = MyPlayer() # creating a player
9     print(p.play()) # showing what it plays
```

docstring - popis hráče/třídy

vždy vrat' 'R'

vykonej, pokud je spuštěno jako hlavní program

vytvoření konkrétního hráče podle vzoru

konkrétní hráč hraje

Co když potřebujeme chování hráče měnit?

playerdummys.py

```
1 class MyPlayer:
2     '''A dummy player on steroids'''
3     def __init__(self, answer='R'):
4         self.answer = answer
5
6     def play(self):
7         return self.answer
8
9 if __name__ == "__main__":
10     p1 = MyPlayer() # creating a default player
11     print(p1.play()) # showing what it plays
12     p2 = MyPlayer('P') # a better player?
13     print(p2.play()) # showing what it plays
14     # oops changed mind
15     p1.answer = 'S'
16     print(p1.play())
17
```

konstruktor objektu

přiřazení atributu objektu

konkrétní hráč bude hrát tak, jak má předepsáno

vlastnosti objektu mohu měnit

podruhé už hraje jinak

krokujte, vizualizujte

Python 3.6

```
1 class MyPlayer:
2     def __init__(self, answer):
3         self.answer = answer
4     def play(self):
5         return self.answer
6
7 if __name__ == "__main__":
8     p1 = MyPlayer('P')
9     p2 = MyPlayer('S')
10    move1 = p1.play()
11    move2 = p2.play()
```

Frames

Objects

Global frame

MyPlayer

p1

p2

play

self

MyPlayer class

[hide attributes](#)

__init__	function __init__(self, answer)
play	function play(self)

MyPlayer instance

answer	"P"
--------	-----

MyPlayer instance

answer	"S"
--------	-----

<http://pythontutor.com/>

playerdummysplusplus.py (dummy s pameti)

```
1 class MyPlayer:
2     '''A dummy player on steroids'''
3     def __init__(self, answer='R'):
4         self.answer = answer
5         self.history = [] vytvoření prázdného seznamu
6
7     def play(self):
8         return self.answer
9
10    def record(self, move):
11        self.history.append(move) přidej tah na konec seznamu
12        append je metoda pro seznamy
13 if __name__ == "__main__":
14     p1 = MyPlayer() # creating a default player
15     print(p1.play()) # showing what it plays
16     p2 = MyPlayer('P') # a better player?
17     print(p2.play()) # showing what it plays
18     # oops changed mind
19     p1.answer = 'S'
20     print(p1.play())
21     # just check the record function
22     p1.record('S')
23     print(p1.history)
24
```

Player

player 1

play

R | P | S

Game

run

yes -> play again

draw?

no

evaluate

winner is ...

Player

player 2

play

R | P | S

Jak hrát hru?

```
p1 = Player
```

```
p2 = Player
```

```
draw = True
```

```
while draw:
```

```
    move1 = p1.play
```

```
    move2 = p2.play
```

```
    draw = (move1 == move2)
```

```
result = evaluate(move1, move2)
```

Třída Game

```
1 class Game:
```

```
2     def __init__(self, p1, p2):
```

```
3         self.p1 = p1
```

Předání hráčů do hry

```
4         self.p2 = p2
```

```
5         self.winner = None
```

Na začátku žádný vítěz není

```
7     def run(self):
```

```
8         draw = True
```

Dokud nehrají, tak je remíza

```
9         while draw:
```

a dokud je remíza, tak hrajte dál

```
10            move1 = self.p1.play()
```

```
11            move2 = self.p2.play()
```

```
12            draw = (move1 == move2)
```

```
13            result = evaluate_moves([move1, move2])
```

```
14            if result[0] > result[1]:
```

```
15                self.winner = self.p1
```

Porovnej tahy podle pravidel
a přiřad' vítěze

```
16            else:
```

```
17                self.winner = self.p2
```

porovnej tahy

```
1 def evaluate_moves(moves):
2     '''
3     compares moves (plays) and decides about the winner
4     :param moves: 1x2 list of valid moves
5     :return: 1x2 list with points [1,0] or [0,1]
6     depending on who is winner
7     '''
8     if moves in [['P', 'R'], ['S', 'P'], ['R', 'S']]:
9         return [1, 0]
10    else:
11        return [0, 1]
12
```

Paper > Rock, Scissors > Paper,
Rock > Scissors

hlavní program game.py

```
1 import playertom  
2 import playerdummy
```

importuj hráče z modulů - souborů
playertom.py a playerdummy.py

```
3 tady patří class Game: a spol. ...
```

```
4 if __name__ == "__main__":
```

```
5     p1 = playertom.MyPlayer() vytvoř hráče
```

```
6     p2 = playerdummy.MyPlayer()
```

```
7     g = Game(p1, p2) inicializuj hru, předej hráče
```

```
8     g.run() hraj
```

```
9     print('Winner is:', g.winner.__doc__)
```

```
10         oznam vítěze
```

Pro pozdější analýzu výsledku, navrátíme index vítěze

```
1 class Game:
2     def __init__(self, p1, p2):
3         self.p1 = p1
4         self.p2 = p2
5         self.winner = None
6
7     def run(self):
8         draw = True
9         while draw:
10            move1 = self.p1.play()
11            move2 = self.p2.play()
12            draw = (move1 == move2)
13            result = evaluate_moves([move1, move2])
14            if result[0] > result[1]:
15                self.winner = p1
16                return 0
17            else:
18                self.winner = p2
19                return 1
```

Návratová hodnota podle toho,
kdo vyhrál

opakovaná hra

```
1 def compute_stats(winners):  
2     wins = [0, 0] počet výher každého hráče  
3     for winner in winners: pro každého vítěze v seznamu  
4         wins[winner] = wins[winner] + 1  
5     return wins  
6  
7 if __name__ == "__main__":  
8     p1 = playertom.MyPlayer()  
9     p2 = playerdummy.MyPlayer()  
10    winners = [] inicializuj prázdný seznam  
11    for i in range(10): mohl bych inicializaci hry přesunout před cyklus  
12        g = Game(p1, p2) pak bych ale musel doplnit "vynulování"  
13        winners.append(g.run()) self.winner i do těla metody run. Rozmyslete.  
14        print('Winner is:', g.winner.doc) přidej výsledek na konec seznamu  
15    wins = compute_stats(winners) analyzuj celkové výsledky  
16    print(p1.doc, 'won %d times' % wins[0])  
17    print(p2.doc, 'won %d times' % wins[1])
```

iterative game

```
48 class IterativeGame:
49     def __init__(self, p1, p2, runs=1):
50
57         self.runs = runs
58         self.p = [p1, p2]
59         self.profits = [0, 0]
60
61     def run(self):
62         for k in range(self.runs):
63             draw = True
64             while draw:
65                 moves = [None, None] # init moves
66                 for i in range(2):
67                     moves[i] = self.p[i].play()
68                     if not(is_valid_move(moves[i])):
69                         raise RuntimeError
71                 draw = (moves[0] == moves[1])
72             profit_increments = evaluate_moves(moves)
73             for i in range(2):
74                 self.profits[i] += profit_increments[i]
```

celkový výsledek hry bude v proměnné objektu