B4B33RPH: Rešení problémů a hry

Kámen-nůžky-papír

Tomáš Svoboda, Petr Pošík, **Petr Štibinger**

stibipet@fel.cvut.cz

4.10.2022

Katedra kybernetiky
Fakulta elektrotechnická
České vysoké učení technické v Praze

Než začneme...

Doplňková cvičení

- Každý čtvrtek, 16:30 – 17:30
- Místnost KN:E-310
- Jakub Kislinger, `kislijak@fel.cvut.cz`
- Konzultace, řešení problémů

Na přemýšlení

Představte si ženu jménem Lenka. Je jí 33, svobodná, upřímná, přímočará a velmi bystrá. Vystudovala filozofii na vysoké škole. Během studia se velmi zajímala o problémy diskriminace, sociální spravedlnost a účastnila se demonstrací proti atomovým zbraním.

Vaším úkolem je odhadnout, co dělá teď. Zkuste seřadit následující možnosti od nejvíce k nejméně pravděpodobné. Lenka je:

(a) aktivní feministka

(b) bankovní úřednice a aktivní feministka

(c) bankovní úřednice

Na přemýšlení

Dilema

Pokud **všichni** studenti zvolí pro svou odpověď **velká písmena**, dostane každý navíc **1 bod**. Pokud bude **alespoň jedna** odpověď tvořena **malými písmeny**, všichni, kteří odpověděli velkými písmeny, žádné další body nedostanou. Ti, kteří použili malá písmena, dostanou navíc pouze **0.5 bodu**.

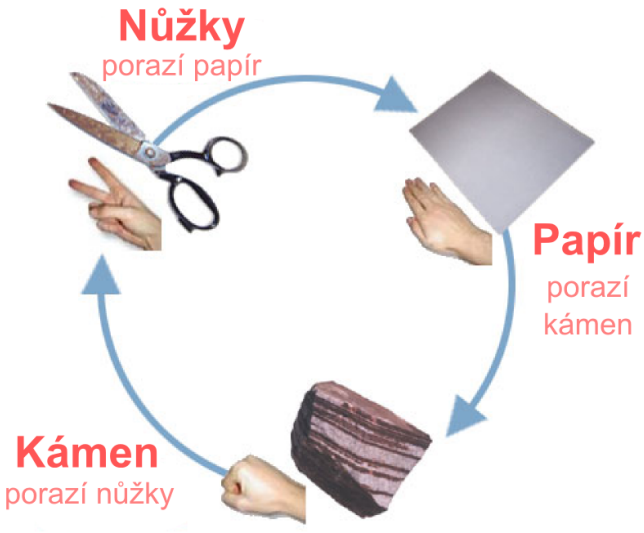
Poznámka k programovacím úlohám

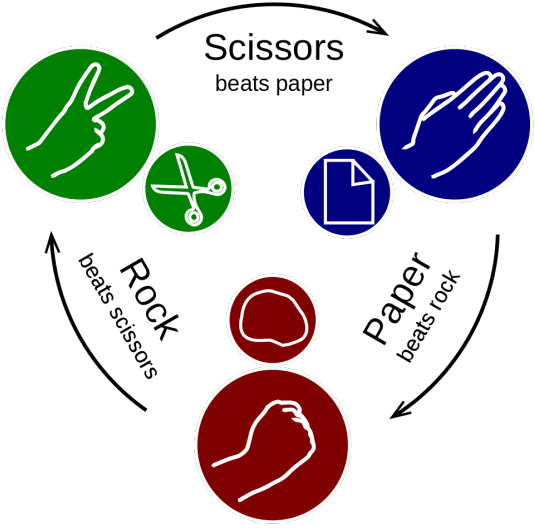
- Nutná vlastní práce
- Úlohy navrženy tak, aby šly vyřešit vlastními silami
- Základní koncepty z přednášek a cvičení
- Zapomínací test
- Pravidla samostatné práce

Kámen nůžky papír

- Jemný úvod do objektového programování
- Málokdy existuje **jediné správné** řešení

- Kódy budou po přednášce ke stažení na CourseWare
- Uvítáme bug-reporty, tipy pro zlepšení
- Kódy ve slajdech se mohou lišit od kódů ke stažení


Nůžky
porazí papír
Papír
porazí
kámen
Kámen
porazí nůžky

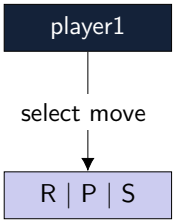

Scissors
beats paper
Paper
beats rock
Rock
beats scissors

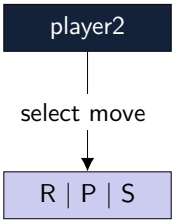
Průběh hry

player1

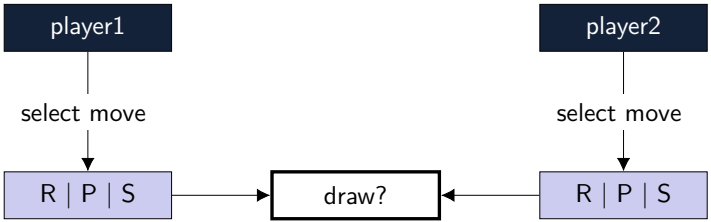
player2

Průběh hry

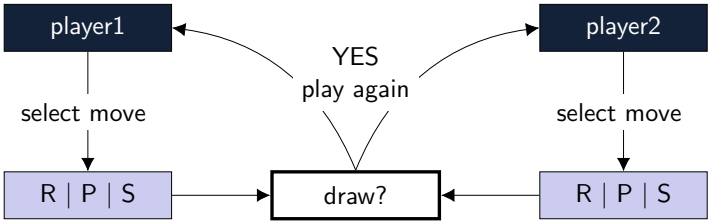




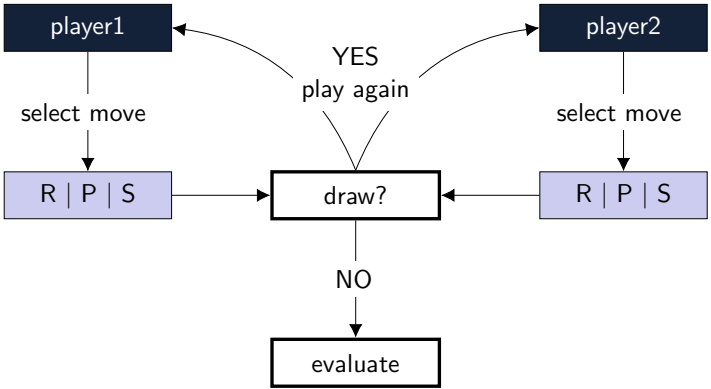
Průběh hry



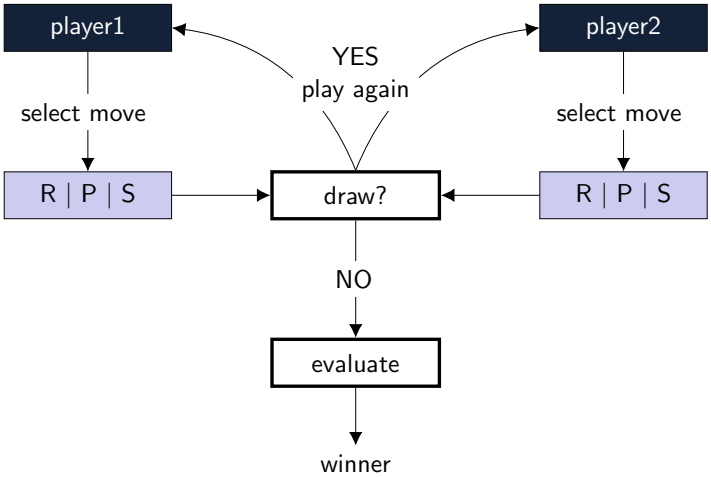
Průběh hry


player1
player2
YES
play again
select move
select move
R | P | S
draw?
R | P | S

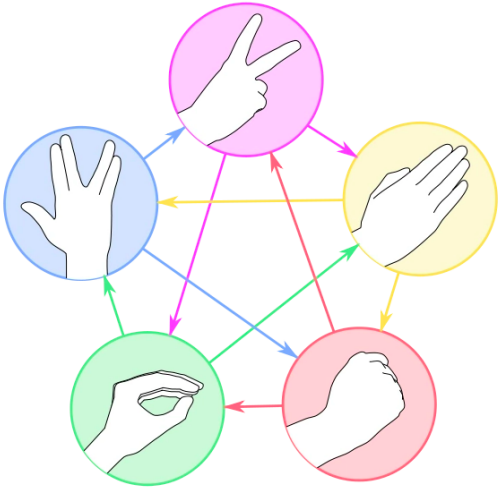
Průběh hry

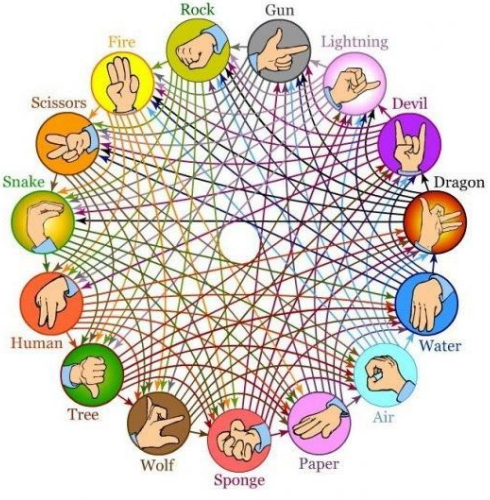

player1
player2
YES
play again
select move
select move
R | P | S
draw?
R | P | S
NO
evaluate

Průběh hry


player1
player2
YES
play again
select move
select move
R | P | S
draw?
R | P | S
NO
evaluate
winner

Vyšší level




Rock
Gun
Fire
Lightning
Scissors
Devil
Snake
Dragon
Human
Water
Tree
Air
Wolf
Paper
Sponge

Python vsuvka

- Paměťová místa s daty jsou **odkazována** (reference)
- Typ proměnné (a odpovídající paměťové místo) je přiděleno dynamicky
- Operátor **přiřazení** $=$
- Při ztrátě ukazatele (smazání, konec platnosti...) se paměť čistí (garbage collector)
- **Téměř vše** v Pythonu je **objekt**
- V Pythonu můžeme odkazovat/ukazovat **na všechno**

Příklady z Python konzole

```
>>> a = 1
>>> b = 2.5
>>> c = a+b
```

(a) skončí chybou

(b) **c** je typu **int**

(c) **c** je typu **float**

Příklady z Python konzole

```
>>> a = [1,2,3]
>>> b = [1,1,1]
>>> c = a+b
```

(a) **c** je rovno **[2,3,4]**

(b) skončí chybou

(c) **c** je rovno **[1,2,3,1,1,1]**

Příklady z Python konzole

```
>>> a = 1
>>> b = -a
>>> c = abs
>>> d = c(b)
```

(a) **d** je rovno **1**
(b) **d** je rovno **-1**
(c) **d** je **True**
(d) skončí chybou

Pokud si nejsme jisti, Python konzole to jistí

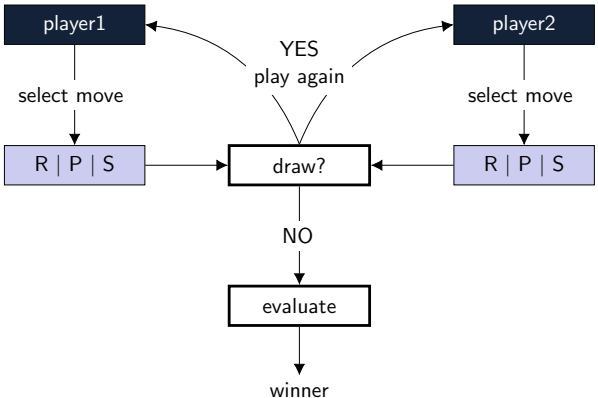
- `type(variable_name)`
- `isinstance(variable_name, data_type)`
- `help(variable_name)`

- klávesa Tab – autocomplete (ve většině IDE)

- Vyhledávání na internetu (pozor na plagiáty!)
- Python3 dokumentace

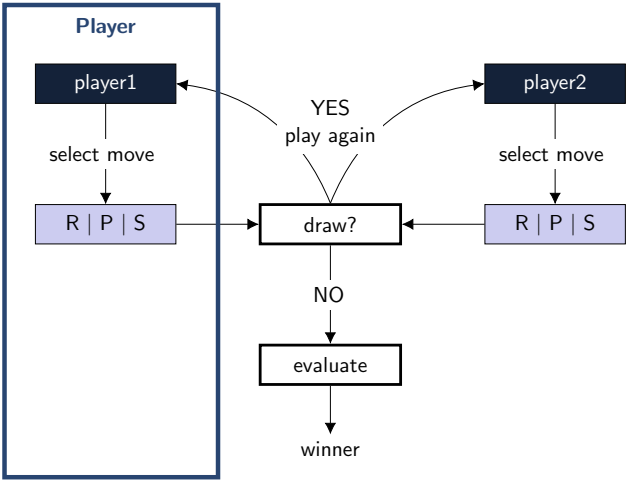
Řízení hry

- Kdo má tyto akce na starost?



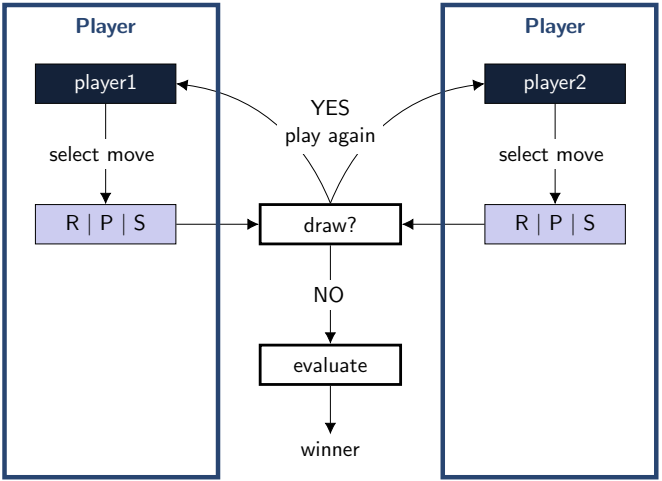
Řízení hry

- Kdo má tyto akce na starost?



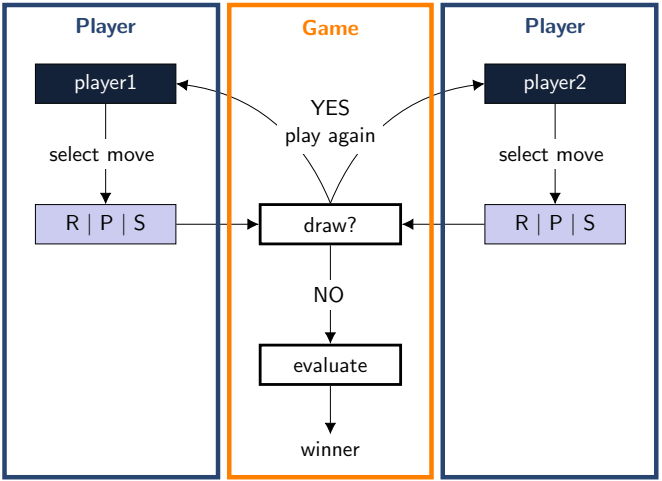
Řízení hry

- Kdo má tyto akce na starost?



Řízení hry

- Kdo má tyto akce na starost?



Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

docstring - popis chování

Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

docstring - popis chování

metoda - to co má na starost hráč

Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

docstring - popis chování

metoda - to co má na starost hráč

proveď, pokud je spuštěno jako hlavní program

Jednoduchý hráč

`playerdummy.py` - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

docstring - popis chování

metoda - to co má na starost hráč

proveď, pokud je spuštěno jako hlavní program

vytvoř **konkrétního** hráče

Jednoduchý hráč

playerdummy.py - hráč, co vždy hraje 'R'

obecná definice hráče

```
1  class MyPlayer:
2      '''A simple player that always returns R'''
3
4      def play(self):
5          return 'R'
6
7  if __name__ == '__main__':
8      p = MyPlayer() # creating one player
9      print(p.play()) # showing what they played
```

docstring - popis chování

metoda - to co má na starost hráč

proveď, pokud je spuštěno jako hlavní program

vytvoř **konkrétního** hráče

konkrétní hráč hraje

Jednoduchý hráč

playerdummy_plus.py - hraje vždy to, co mu na začátku nastavíme

```
1  class MyPlayer:
2      '''A simple player that always returns the same answer'''
3
4      def __init__(self, answer='R'):
5          self.answer = answer
6
7      def play(self):
8          return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer()      # creating a default player
12     print(p1.play())     # showing answer of p1
13     p2 = MyPlayer('P')   # a better player?
14     print(p2.play())     # showing answer of p2
15     p1.answer = 'S'      # p1 changed his mind
16     print(p1.play())
```

Jednoduchý hráč

playerdummy_plus.py - hraje vždy to, co mu na začátku nastavíme

```
1  class MyPlayer:
2      '''A simple player that always returns the same answer'''
3
4      def __init__(self, answer='R'):
5          self.answer = answer
6
7      def play(self):
8          return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer()      # creating a default player
12     print(p1.play())     # showing answer of p1
13     p2 = MyPlayer('P')   # a better player?
14     print(p2.play())     # showing answer of p2
15     p1.answer = 'S'      # p1 changed his mind
16     print(p1.play())
```

konstruktor - při vytvoření objektu

Jednoduchý hráč

playerdummy_plus.py - hraje vždy to, co mu na začátku nastavíme

```
1  class MyPlayer:
2      '''A simple player that always returns the same answer'''
3
4      def __init__(self, answer='R'):
5          self.answer = answer
6
7      def play(self):
8          return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer()      # creating a default player
12     print(p1.play())     # showing answer of p1
13     p2 = MyPlayer('P')   # a better player?
14     print(p2.play())     # showing answer of p2
15     p1.answer = 'S'      # p1 changed his mind
16     print(p1.play())
```

konstruktor - při vytvoření objektu

přiřazení **atributu** objektu

Jednoduchý hráč

playerdummy_plus.py - hraje vždy to, co mu na začátku nastavíme

```
1  class MyPlayer:
2      '''A simple player that always returns the same answer'''
3
4      def __init__(self, answer='R'):
5          self.answer = answer
6
7      def play(self):
8          return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer()      # creating a default player
12     print(p1.play())     # showing answer of p1
13     p2 = MyPlayer('P')   # a better player?
14     print(p2.play())     # showing answer of p2
15     p1.answer = 'S'      # p1 changed his mind
16     print(p1.play())
```

konstruktor - při vytvoření objektu

přiřazení **atributu** objektu

každý hráč bude hrát podle **svého** nastavení

Jednoduchý hráč

playerdummy_plus.py - hraje vždy to, co mu na začátku nastavíme

```
1  class MyPlayer:
2      '''A simple player that always returns the same answer'''
3
4      def __init__(self, answer='R'):
5          self.answer = answer
6
7      def play(self):
8          return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer()      # creating a default player
12     print(p1.play())     # showing answer of p1
13     p2 = MyPlayer('P')   # a better player?
14     print(p2.play())     # showing answer of p2
15     p1.answer = 'S'
16     print(p1.play())
```

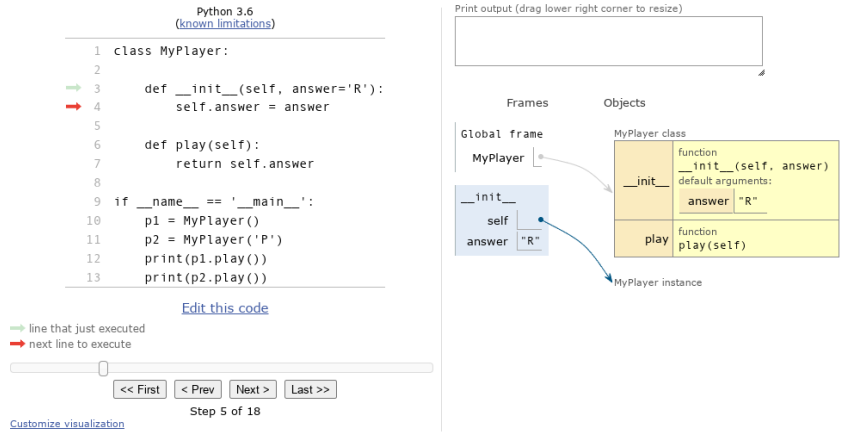
konstruktor - při vytvoření objektu

přiřazení **atributu** objektu

každý hráč bude hrát podle **svého** nastavení

atributy existujícího objektu můžu měnit

Krokování, vizualizace



https://pythontutor.com

Hráč s pamětí

playerdummy_plus_mem.py - navíc s pamětí tahů

```
1  class MyPlayer:
2      '''A dummy player on steroids'''
3      def __init__(self, answer='R'):
4          self.answer = answer
5          self.history = []
6
7      def play(self):
8          return self.answer
9
10     def record(self, move):
11         self.history.append(move)
12
13 if __name__ == '__main__':
14     p1 = MyPlayer()      # creating a default player
15     p2 = MyPlayer('P')  # a better player?
16     answer2 = p2.play()
17     p1.record(answer2)  # p1 records what p2 played
18     print(answer2)
19     print(p1.history) # check that recording works
```

Hráč s pamětí

playerdummy_plus_mem.py - navíc s pamětí tahů

```
1  class MyPlayer:
2      '''A dummy player on steroids'''
3      def __init__(self, answer='R'):
4          self.answer = answer
5          self.history = []
6
7      def play(self):
8          return self.answer
9
10     def record(self, move):
11         self.history.append(move)
12
13 if __name__ == '__main__':
14     p1 = MyPlayer()      # creating a default player
15     p2 = MyPlayer('P')   # a better player?
16     answer2 = p2.play()
17     p1.record(answer2)   # p1 records what p2 played
18     print(answer2)
19     print(p1.history) # check that recording works
```

každý hráč bude mít svůj prázdný **seznam** (list)

Hráč s pamětí

playerdummy_plus_mem.py - navíc s pamětí tahů

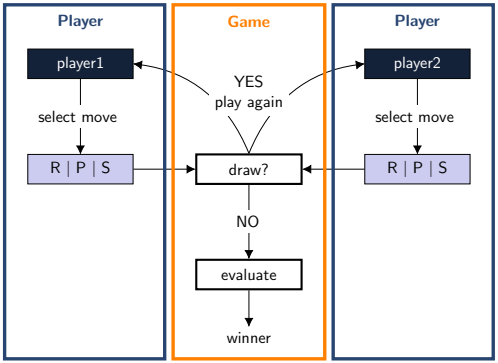
```
1  class MyPlayer:
2      '''A dummy player on steroids'''
3      def __init__(self, answer='R'):
4          self.answer = answer
5          self.history = []
6
7      def play(self):
8          return self.answer
9
10     def record(self, move):
11         self.history.append(move)
12
13 if __name__ == '__main__':
14     p1 = MyPlayer()      # creating a default player
15     p2 = MyPlayer('P')   # a better player?
16     answer2 = p2.play()
17     p1.record(answer2)   # p1 records what p2 played
18     print(answer2)
19     print(p1.history) # check that recording works
```

každý hráč bude mít svůj prázdný **seznam** (list)

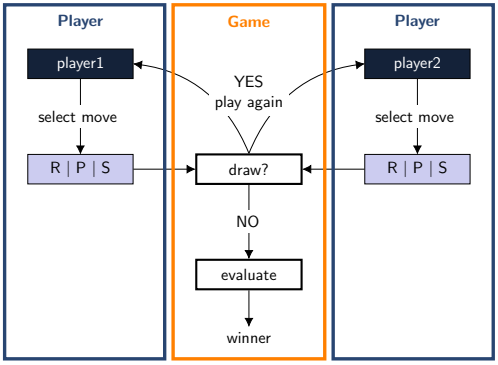
přidej *move* na konec seznamu
append je metoda seznamu

Řízení hry

- Jak převedeme schéma do kódu?

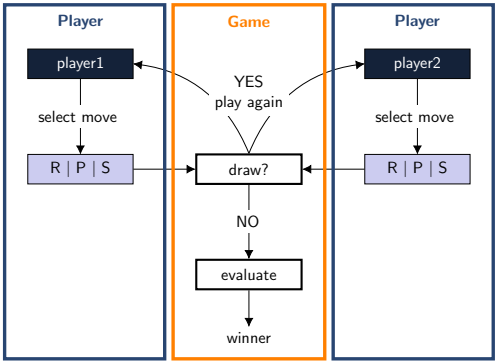


Řízení hry



pseudokód

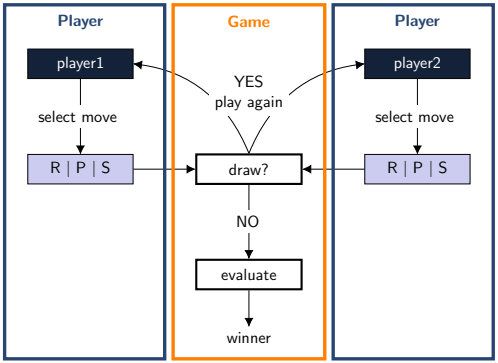
Řízení hry



pseudokód

```
p1 = Player
p2 = Player
```

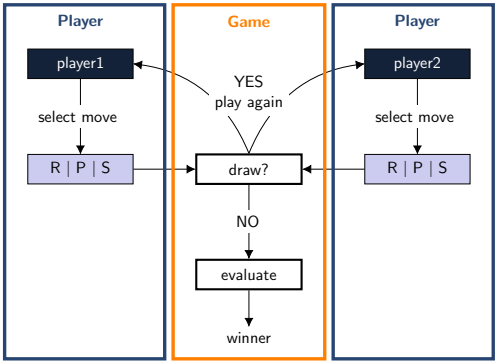
Řízení hry



pseudokód

```
p1 = Player
p2 = Player
draw = True
```

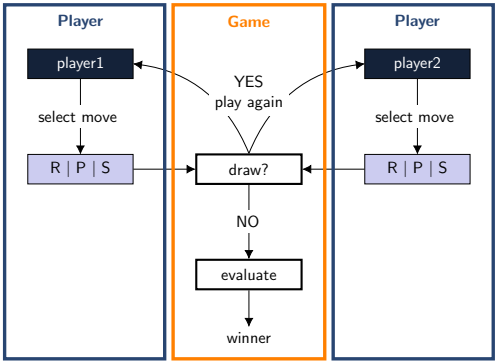
Řízení hry



pseudokód

```
p1 = Player
p2 = Player
draw = True
while draw:
```

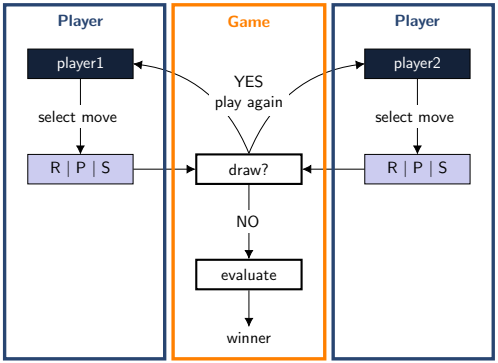
Řízení hry



pseudokód

```
p1 = Player
p2 = Player
draw = True
while draw:
    move1 = p1.play
    move2 = p2.play
```

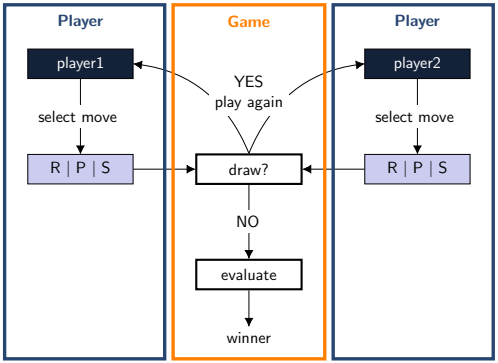
Řízení hry



pseudokód

```
p1 = Player
p2 = Player
draw = True
while draw:
    move1 = p1.play
    move2 = p2.play
    draw = (move1 == move2)
```

Řízení hry



pseudokód

```
p1 = Player
p2 = Player
draw = True
while draw:
    move1 = p1.play
    move2 = p2.play
    draw = (move1 == move2)
winner = evaluate(move1, move2)
```

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
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
14         self.winner = self.evaluate(move1, move2)
```

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
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
14         self.winner = self.evaluate(move1, move2)
```

předání hráčů do hry

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
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
14         self.winner = self.evaluate(move1, move2)
```

předání hráčů do hry

na začátku hry není vítězem nikdo

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
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
14         self.winner = self.evaluate(move1, move2)
```

předání hráčů do hry

na začátku hry není vítězem nikdo

dokud nezačnou hrát, je remíza

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
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
14         self.winner = self.evaluate(move1, move2)
```

předání hráčů do hry

na začátku hry není vítězem nikdo

dokud nezačnou hrát, je remíza

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
1  class Game:
2
3      ...
4
5      def evaluate(self, move1, move2):
6          '''
7          compare moves (plays) of the players and return the score for each player
8          does not consider draw
9          :param move1: move of p1
10         :param move2: move of p2
11         :return: reference to the winning player
12         '''
13         if (move1, move2) in (('R', 'S'), ('S', 'P'), ('P', 'R')):
14             return self.p1
15         else:
16             return self.p2
```

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
1  class Game:
2
3      ...          pokračování kódu z předchozí stránky
4
5      def evaluate(self, move1, move2):
6          '''
7          compare moves (plays) of the players and return the score for each player
8          does not consider draw
9          :param move1: move of p1
10         :param move2: move of p2
11         :return: reference to the winning player
12         '''
13         if (move1, move2) in (('R', 'S'), ('S', 'P'), ('P', 'R')):
14             return self.p1
15         else:
16             return self.p2
```

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
1  class Game:
2
3      ...
4
5      def evaluate(self, move1, move2):
6          '''
7          compare moves (plays) of the players and return the score for each player
8          does not consider draw
9          :param move1: move of p1
10         :param move2: move of p2
11         :return: reference to the winning player
12         '''
13         if (move1, move2) in (('R', 'S'), ('S', 'P'), ('P', 'R')):
14             return self.p1
15         else:
16             return self.p2
```

pokračování kódu z předchozí stránky

všechny dvojice tahů, které vyhraje první hráč

Třída Game

game_simple.py - jednoduchá hra s jedním kolem

```
1  class Game:
2
3      ...
4
5      def evaluate(self, move1, move2):
6          '''
7          compare moves (plays) of the players and return the score for each player
8          does not consider draw
9          :param move1: move of p1
10         :param move2: move of p2
11         :return: reference to the winning player
12         '''
13         if (move1, move2) in (('R', 'S'), ('S', 'P'), ('P', 'R')):
14             return self.p1
15         else:
16             return self.p2
```

pokračování kódu z předchozí stránky

všechny dvojice tahů, které vyhraje první hráč

pokud nevyhrál první hráč, vyhraje druhý

Třída Game

run_game.py - hlavní spouštěcí program

```
1  import playerdummy_plus
2  import game_simple
3
4  if __name__ == '__main__':
5      p1 = playerdummy_plus.MyPlayer('R')
6      p2 = playerdummy_plus.MyPlayer('S')
7      g = game_simple.Game(p1,p2)
8      g.run()
9      print('Winner is:', g.winner)
```

Třída Game

run_game.py - hlavní spouštěcí program

```
1  import playerdummy_plus
2  import game_simple
3
4  if __name__ == '__main__':
5      p1 = playerdummy_plus.MyPlayer('R')
6      p2 = playerdummy_plus.MyPlayer('S')
7      g = game_simple.Game(p1,p2)
8      g.run()
9      print('Winner is:', g.winner)
```

import **modulů** - ze souborů playerdummy_plus.py a game_simple.py

Třída Game

run_game.py - hlavní spouštěcí program

```
1  import playerdummy_plus
2  import game_simple
3
4  if __name__ == '__main__':
5      p1 = playerdummy_plus.MyPlayer('R')
6      p2 = playerdummy_plus.MyPlayer('S')
7      g = game_simple.Game(p1,p2)
8      g.run()
9      print('Winner is:', g.winner)
```

import **modulů** - ze souborů playerdummy_plus.py a game_simple.py

vytvářám hráče (přístup ke kódu z modulu)

Třída Game

run_game.py - hlavní spouštěcí program

```
1  import playerdummy_plus
2  import game_simple
3
4  if __name__ == '__main__':
5      p1 = playerdummy_plus.MyPlayer('R')
6      p2 = playerdummy_plus.MyPlayer('S')
7      g = game_simple.Game(p1,p2)
8      g.run()
9      print('Winner is:', g.winner)
```

import **modulů** - ze souborů playerdummy_plus.py a game_simple.py

vytvářám hráče (přístup ke kódu z modulu)

vytvářím hru, přidávám hráče

Třída Game

run_game.py - hlavní spouštěcí program

```
1  import playerdummy_plus
2  import game_simple
3
4  if __name__ == '__main__':
5      p1 = playerdummy_plus.MyPlayer('R')
6      p2 = playerdummy_plus.MyPlayer('S')
7      g = game_simple.Game(p1,p2)
8      g.run()
9      print('Winner is:', g.winner)
```

import **modulů** - ze souborů playerdummy_plus.py a game_simple.py

vytvářám hráče (přístup ke kódu z modulu)

vytvářím hru, přidávám hráče

odehraj jedno kolo, vypiš vítěze

Iterativní hra - na N vítězných

```
1   class Game:
2       def __init__(self, p1, p2, min_wins):
3           self.p1 = p1
4           self.p2 = p2
5           self.min_wins = min_wins
6           self.score = [0, 0]
7           self.winner = None
8
9       def evaluate(self, move1, move2):
10          if (move1, move2) in (('R', 'S'), ('P', 'R'), ('S', 'P')):
11              return 1, 0
12          elif (move1, move2) in (('S', 'R'), ('R', 'P'), ('P', 'S')):
13              return 0, 1
14          else:
15              return 0,0
16
17      def run(self):
18          while max(self.score) < self.min_wins:
19              round_score = self.evaluate(self.p1.play(), self.p2.play())
20              self.score[0] += round_score[0]
21              self.score[1] += round_score[1]
22
23          if self.score[0] > self.score[1]:
24              self.winner = self.p1
25          else:
26              self.winner = self.p2
```

Iterativní hra - na N vítězných

```
1   class Game:
2       def __init__(self, p1, p2, min_wins):
3           self.p1 = p1
4           self.p2 = p2
5           self.min_wins = min_wins
6           self.score = [0, 0]
7           self.winner = None
8
9       def evaluate(self, move1, move2):
10          if (move1, move2) in (('R', 'S'), ('P', 'R'), ('S', 'P')):
11              return 1, 0
12          elif (move1, move2) in (('S', 'R'), ('R', 'P'), ('P', 'S')):
13              return 0, 1
14          else:
15              return 0,0
16
17      def run(self):
18          while max(self.score) < self.min_wins:
19              round_score = self.evaluate(self.p1.play(), self.p2.play())
20              self.score[0] += round_score[0]
21              self.score[1] += round_score[1]
22
23          if self.score[0] > self.score[1]:
24              self.winner = self.p1
25          else:
26              self.winner = self.p2
```

hra si uchovává aktuální skóre

Iterativní hra - na N vítězných

```
1   class Game:
2       def __init__(self, p1, p2, min_wins):
3           self.p1 = p1
4           self.p2 = p2
5           self.min_wins = min_wins
6           self.score = [0, 0]
7           self.winner = None
8
9       def evaluate(self, move1, move2):
10          if (move1, move2) in (('R', 'S'), ('P', 'R'), ('S', 'P')):
11              return 1, 0
12          elif (move1, move2) in (('S', 'R'), ('R', 'P'), ('P', 'S')):
13              return 0, 1
14          else:
15              return 0,0
16
17      def run(self):
18          while max(self.score) < self.min_wins:
19              round_score = self.evaluate(self.p1.play(), self.p2.play())
20              self.score[0] += round_score[0]
21              self.score[1] += round_score[1]
22
23          if self.score[0] > self.score[1]:
24              self.winner = self.p1
25          else:
26              self.winner = self.p2
```

hra si uchovává aktuální skóre

evaluate spočítá skóre pro 1 kolo

Iterativní hra - na N vítězných

```
1   class Game:
2       def __init__(self, p1, p2, min_wins):
3           self.p1 = p1
4           self.p2 = p2
5           self.min_wins = min_wins
6           self.score = [0, 0]
7           self.winner = None
8
9       def evaluate(self, move1, move2):
10          if (move1, move2) in (('R', 'S'), ('P', 'R'), ('S', 'P')):
11              return 1, 0
12          elif (move1, move2) in (('S', 'R'), ('R', 'P'), ('P', 'S')):
13              return 0, 1
14          else:
15              return 0,0
16
17      def run(self):
18          while max(self.score) < self.min_wins:
19              round_score = self.evaluate(self.p1.play(), self.p2.play())
20              self.score[0] += round_score[0]
21              self.score[1] += round_score[1]
22
23          if self.score[0] > self.score[1]:
24              self.winner = self.p1
25          else:
26              self.winner = self.p2
```

hra si uchovává aktuální skóre

evaluate spočítá skóre pro 1 kolo

hraje se dokud jeden z hráčů nemá potřebný počet vítězství

Díky za pozornost

Prostor pro dotazy

Na cvičení – první testík