

B4B33RPH: Řešení problémů a hry

Kámen-nůžky-papír

Tomáš Svoboda, Petr Pošík, Petr Štibinger

svobodat@fel.cvut.cz

11. října 2023



Katedra kybernetiky
Fakulta elektrotechnická
České vysoké učení technické v Praze

Než začneme...

Doplňková cvičení

- Každý čtvrtek, 17:00 – 18:00
- Místnost KN:E-307
- Patrik Kolář, kolarpat@fel.cvut.cz
- Konzultace, řešení problémů

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

Školák vs. autobus, pozdě či včas?

- Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi 6:58-7:02.
- Přes veškerou snahu přichází školák na zastávku kdykoli mezi 6:55-7:01.
- Jak často přijde školák pozdě do školy?

Školák vs. autobus, pozdě či včas?

- Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi 6:58-7:02.
 - Přes veškerou snahu přichází školák na zastávku kdykoli mezi 6:55-7:01.
 - Jak často přijde školák pozdě do školy?
-
- Klíčová data?

Školák vs. autobus, pozdě či včas?

- Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi **6:58-7:02**.
 - Přes veškerou snahu přichází školák na zastávku kdykoli mezi **6:55-7:01**.
 - Jak často přijde školák pozdě do školy?
-
- Klíčová data?

ChatGPT/Bing pokusy text

* GPT-4



školák má problém, že často dorazí do školy pozdě. autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi 6:58-7:02. přes veškerou snahu přichází školák na zastávku kdykoli mezi 6:55-7:01. jak často přijde školák pozdě do školy?



Abychom zjistili, jak často školák přijde pozdě do školy, musíme se podívat na různé situace, kdy autobus přijede dříve než školák a hned odjede.



Rozmezí, kdy může autobus přijet, je 4 minuty (od 6:58 do 7:02).
Rozmezí, kdy může školák přijít, je 6 minut (od 6:55 do 7:01).

Nyní analyzujeme, kdy školák přijde pozdě:

1. Pokud autobus přijede v 6:58, školák bude pozdě, pokud přijde mezi 6:59 a 7:01. To je 2 minuty z celkových 6 minut, tedy $2/6 = 1/3$ šance.
2. Pokud autobus přijede v 6:59, školák bude pozdě, pokud přijde mezi 7:00 a 7:01. To je 1 minuta z celkových 6 minut, tedy $1/6$ šance.
3. Pokud autobus přijede v 7:00, školák nikdy nebude pozdě, protože vždy dorazí nejpozději v 7:01.
4. Pokud autobus přijede v 7:01 nebo 7:02, školák nikdy nebude pozdě, protože už bude na zastávce.

Sčítáme pravděpodobnosti:

$$1/3 + 1/6 = 3/6 = 1/2.$$

Tedy, školák má 50% šanci, že přijde pozdě do školy.

ChatGPT/Bing pokusy text



Školák má problém, že často dorazí do školy pozdě. Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi 6:58-7:02. Přes veškerou snahu přichází školák na zastávku kdykoli mezi 6:55-7:01. Jak často přijde školák pozdě do školy?



Chápu, že chcete zjistit pravděpodobnost, s jakou školák přijde pozdě na zastávku a nestihne autobus.

Autobus může přijet kdykoli v 5-minutovém intervalu (6:58 - 7:02). Podobně školák může přijít na zastávku v 7-minutovém intervalu (6:55 - 7:01).

Nakreslíme si časovou osu pro oba tyto intervaly, které se překrývají:

```
mezifile Copy code
Autobus: |---|---|---|---|
Čas:     6:58 6:59 7:00 7:01 7:02

Školák:  |---|---|---|---|
Čas:     6:55 6:56 6:57 6:58 6:59 7:00 7:01
```

Kdy školák nestihne autobus?

1. Pokud školák přijde na zastávku v 6:55, 6:56 nebo 6:57, vždy stihne autobus, protože autobus přijde nejdříve v 6:58.
2. Pokud školák přijde v 6:58, stihne autobus, pouze pokud autobus nepřijede přesně v 6:58. Pokud autobus přijede v 6:58, školák nestihne autobus. Pravděpodobnost, že autobus přijede v jakékoli jednotlivé minutě mezi 6:58 a 7:02 je 1/5.
3. Pokud školák přijde v 6:59, nestihne autobus, pokud autobus přijede v 6:58 nebo 6:59. Tedy pravděpodobnost, že školák nestihne autobus, je 2/5.
4. Pokud školák přijde v 7:00, nestihne autobus, pokud autobus přijede v 6:58, 6:59 nebo 7:00. Tedy pravděpodobnost, že školák nestihne autobus, je 3/5.
5. Pokud školák přijde v 7:01, vždy nestihne autobus.

Nyní spočítáme celkovou pravděpodobnost:

Pravděpodobnost, že školák přijde v konkrétní minutu mezi 6:55 a 7:01, je 1/7.

Celková pravděpodobnost, že školák nestihne autobus, je:

```
Copy code
0 * 1/7 + 1/5 * 1/7 + 2/5 * 1/7 + 3/5 * 1/7 + 1 * 1/7 = 6/35 + 2/35 + 6/35
```

Tedy školák nestihne autobus s pravděpodobností 22/35, což je přibližně 62,86%.

ChatGPT/Bing pokusy text

Školák má problém, že často dorazí do školy pozdě. Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi 6:58-7:02. Přeš veškerou snahu přichází školák na zastávku kdykoli mezi 6:55 a 7:01. Jak často přijde školák pozdě do školy?

✓ Hledá se: „how to solve probability problems“

✓ Generování odpovědi pro vás...

Toto je otázka o pravděpodobnosti, kterou lze vyřešit pomocí následujícího postupu:

- Nejprve musíme určit, jaké jsou možné případy, kdy školák přijde pozdě do školy. To se stane, pokud autobus přijede a odjede dříve, než školák dorazí na zastávku. Například, pokud autobus přijede v 6:58 a školák v 6:59, nebo pokud autobus přijede v 7:00 a školák v 7:01.
- Potom musíme spočítat, kolik je takových případů. To můžeme udělat pomocí tabulky, kde na jedné ose budou časy příjezdu autobusu a na druhé ose časy příjezdu školáka. Každá buňka v tabulce bude reprezentovat jeden možný scénář. Pokud je buňka pod diagonálou, znamená to, že školák přijde pozdě do školy. Pokud je buňka nad diagonálou nebo na ní, znamená to, že školák stihne autobus.

Autobus \ Školák	6:55	6:56	6:57	6:58	6:59	7:00	7:01
6:58	N	N	N	A	P	P	P
6:59	N	N	N	N	A	P	P
7:00	N	N	N	N	N	A	P
7:01	N	N	N	N	N	N	A
7:02	N	N	N	N	N	N	N

V tabulce jsou použity následující symboly:

- **N** znamená, že školák nepříje na zastávku (např. autobus přijede v 7:02 a školák v 6:55).
- **A** znamená, že školák stihne autobus (např. autobus přijede v 7:00 a školák v 7:00).
- **P** znamená, že školák přijde pozdě do školy (např. autobus přijede v 6:58 a školák v 6:59).
- Nakonec musíme spočítat pravděpodobnost, že školák přijde pozdě do školy. To uděláme tak, že vydělíme počet případů, kdy se to stane, celkovým počtem možných případů. V naší tabulce je celkem **35** buněk a **9** z nich obsahuje symbol **P**. Pravděpodobnost tedy je:

$$\frac{9}{35} \approx 0.257$$

To znamená, že školák má asi **25.7%** šanci, že přijde pozdě do školy.

Pokud chcete více informací o tom, jak řešit problémy s pravděpodobností, můžete si prohlédnout některé zdroje na internetu [1](#) [2](#) [3](#) [4](#). Doufám, že jsem vám pomohl s vaší otázkou. 😊

Další informace: [1. effortlessmath.com](#) [2. bing.com](#) [3. youtube.com](#) [+6 další](#)

1 z(e) 10

ChatGPT/Bing pokusy text

Používám tebou doporučený modul random, v tom chyba není. Ale stále dostávám hodnotu kolem 0.17, což se spíš blíží hodnotě 6/35 než tebou vypočtené hodnotě 9/35. Kde je chyba?

- ✓ Hledá se „how to debug python code“
- ✓ Generování odpovědi pro vás...

Dobrý den, děkuji za vaši zprávu. Po dalším prozkoumání vašeho problému jsem zjistil, že chyba je v tom, že jste použil nesprávný vzorec pro výpočet pravděpodobnosti. Vzorec, který jste použil, je platný pouze pro případ, kdy autobus i školák přicházejí na zastávku v pevných intervalech. Například, pokud by autobus přijížděl každou minutu a školák každých 10 sekund. V tomto případě by pravděpodobnost byla rovna počtu možných kombinací, kdy školák přijde pozdě, děleno celkovým počtem možných kombinací. To je 6/35.

Avšak ve vaší úloze je situace jiná. Autobus i školák přicházejí na zastávku v náhodných časech v určitých mezích. Například, autobus může přijet kdykoli mezi 6:58 a 7:02 a školák kdykoli mezi 6:55 a 7:01. V tomto případě musíme použít jiný vzorec, který bere v úvahu délku těchto rozmezí a jejich vzájemný poměr. Tento vzorec je následující:

$$P = \frac{L_1 - L_2}{L_1 + L_2}$$

Kde P je pravděpodobnost, že školák přijde pozdě do školy, L_1 je délka rozmezí pro příjezd autobusu a L_2 je délka rozmezí pro příjezd školáka. V naší úloze jsou tyto délky následující:

$$L_1 = 7:02 - 6:58 = 4 \text{ minuty}$$

$$L_2 = 7:01 - 6:55 = 6 \text{ minut}$$

Dosažením do vzorce dostaneme:

$$P = \frac{4 - 6}{4 + 6} = -\frac{1}{5} \approx -0.2$$

To znamená, že pravděpodobnost, že školák přijde pozdě do školy, je záporná! To je možné? Ano, je to možné, pokud si uvědomíme, co tento výsledek znamená. Znamená to, že školák má v průměru více času na to, aby stihl autobus, než aby ho nestihl. Jinými slovy, školák má v průměru více štěstí než smůly. To je dáno tím, že rozmezí pro příjezd školáka je delší než rozmezí pro příjezd autobusu.

Pokud bychom chtěli zjistit pravděpodobnost opačného jevu, tedy že školák stihne autobus, museli bychom vzorec upravit takto:

$$P' = \frac{L_2 - L_1}{L_1 + L_2}$$

Kde P' je pravděpodobnost, že školák stihne autobus. Dosažením do vzorce dostaneme:

$$P' = \frac{6 - 4}{4 + 6} = \frac{1}{5} \approx 0.2$$

To znamená, že pravděpodobnost, že školák stihne autobus, je kladná a rovná 0.2. To odpovídá tomu, co jste pozorovali ve své simulaci.

Doufám, že jsem vám objasnil rozdíl mezi dvěma vzorci pro výpočet pravděpodobnosti a kde byla chyba. Pokud máte další dotazy, rád vám odpovím. 🟡

Bing highlight! Žádný problém se zápornou pravděpodobností.

Dosazením do vzorce dostaneme:

$$P = \frac{4 - 6}{4 + 6} = -\frac{1}{5} \approx -0.2$$

To znamená, že pravděpodobnost, že školák přijde pozdě do školy, je záporná! To je možné? Ano, je to možné, pokud si uvědomíme, co tento výsledek znamená. Znamená to, že školák má v průměru více času na to, aby stihl autobus, než aby ho nestihl. Jinými slovy, školák má v průměru více štěstí než smůly. To je dáno tím, že rozmezí pro příjezd školáka je delší než rozmezí pro příjezd autobusu.

Školák vs. autobus, pravděpodobnost pozdního příchodu?

- Autobus má odjíždět ze zastávky v 7:00, ale může přijet (a hned odjet) kdykoli mezi **6:58-7:02**.
- Přes veškerou snahu přichází školák na zastávku kdykoli mezi **6:55-7:01**.
- Jak často přijde školák pozdě do školy?

[A] 6/35

[B] 6/48

[C] 9/48

[D] 9/35

Simulace, začněme od GPT výtvoru

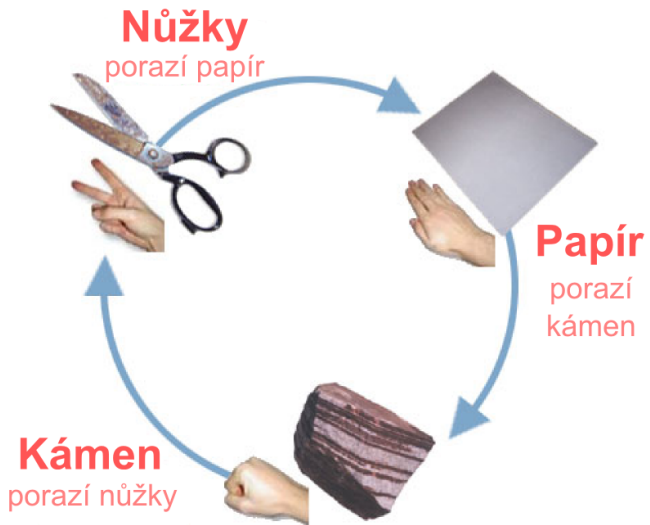
```
Users > svoboda > svoboda-git-clones > rph-felgit > RPH-student-materials > python-snippets > late_to_school_GPT_live
1 import random
2
3 def does_student_miss_bus(trials=10000):
4     missed_count = 0
5     for _ in range(trials):
6         bus_arrival = random.randint(0, 4)
7         student_arrival = random.randint(0, 6)
8
9         if student_arrival > bus_arrival:
10            missed_count += 1
11
12    return missed_count / trials
13
14 miss_probability = does_student_miss_bus()
15 print(f"Pravděpodobnost, že školák nestihne autobus je: {miss_probability:.4f}")
```

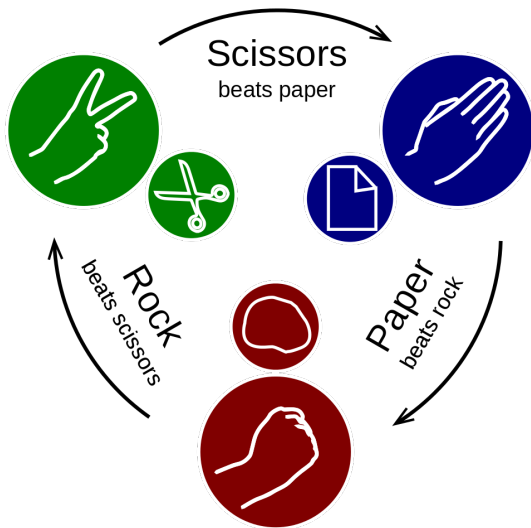
PROBLEMS DEBUG CONSOLE ... Filter (e.g. text, lexclude)

Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.11.1 64-bit

Kámen nůžky papír

- Jemný úvod do objektového programování
- Málokdy existuje **jediné správné** řešení
- Kódy budou po přednášce ke stažení na [CourseWare](#)
- Uvítáme bug-reports, tipy pro zlepšení
- Kódy ve slajdech se mohou lišit od kódů ke stažení



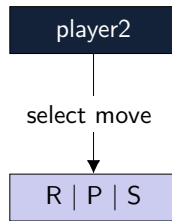
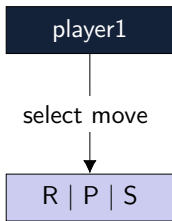


Průběh hry

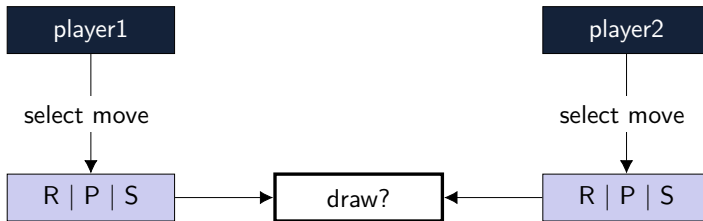
player1

player2

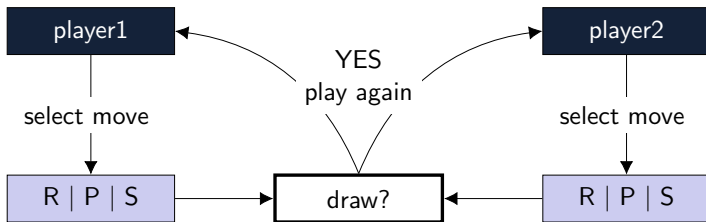
Průběh hry



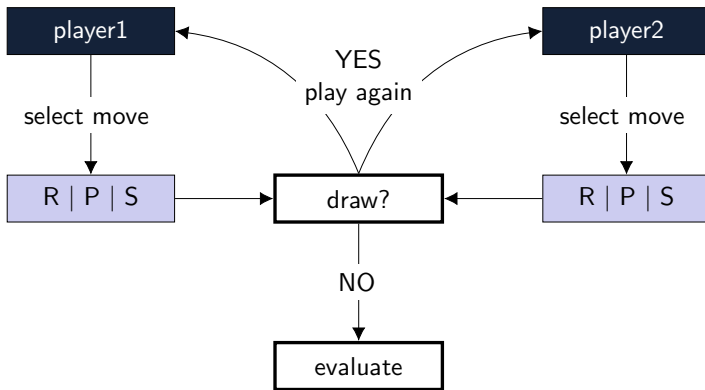
Průběh hry



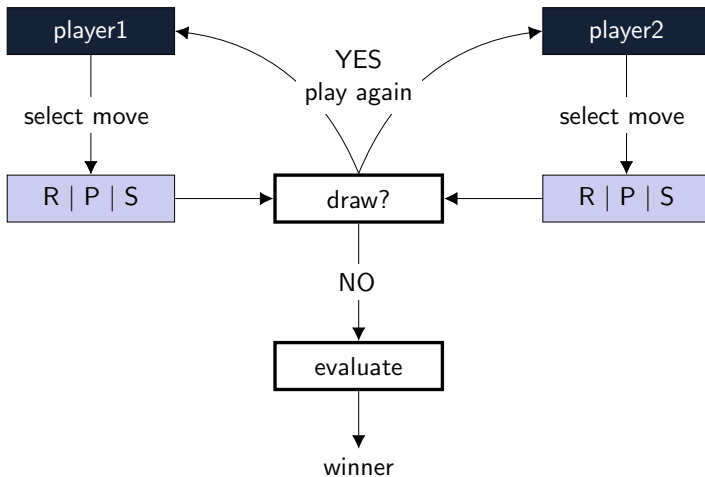
Průběh hry



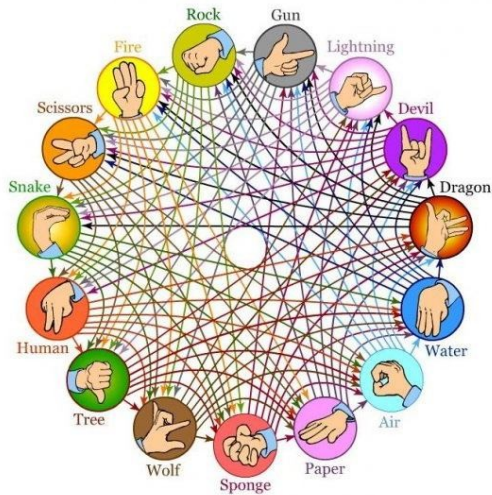
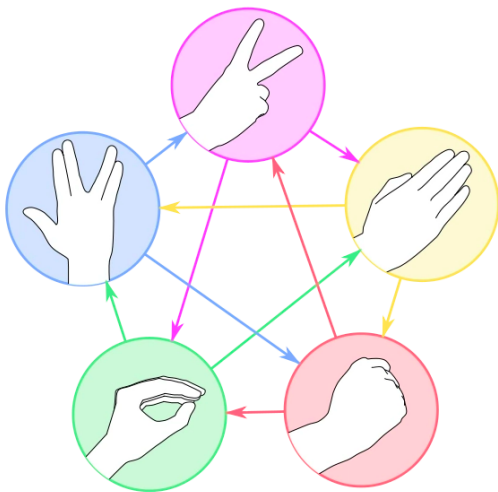
Průběh hry



Průběh hry



Vyšší level



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**

Typ proměnné (a odpovídající paměťové místo) je přiděleno dynamicky

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

- (a) skončí chybou
- (b) **c** je typu **int**
- (c) **c** je typu **float**

Téměř vše v Pythonu je objekt a chování operací na tom závisí

```
>>> 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]

V Pythonu můžeme odkazovat/ukazovat na všechno

```
>>> 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, i Python konzole)

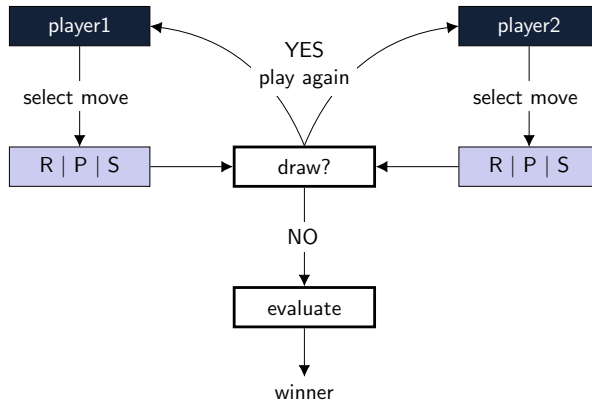
- Vyhledávání na internetu (pozor na plagiáty!)
- **Python3 dokumentace**

Šance uvolnit napětí na slido.com #2210 950

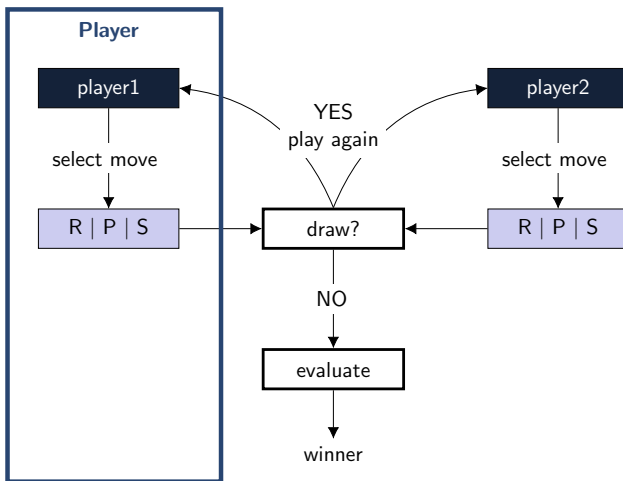


Clipdrop
by stability.ai

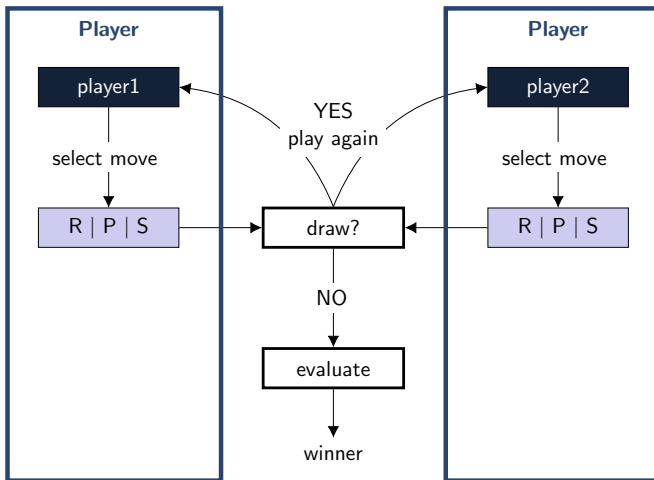
Řízení hry. Kdo má tyto akce na starost?



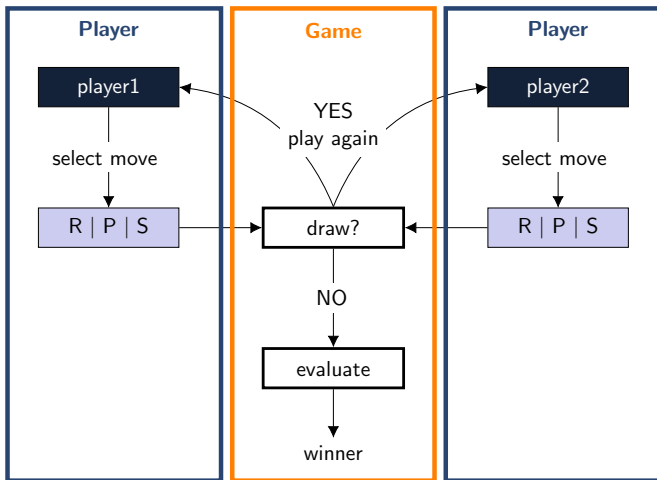
Řízení hry. Kdo má tyto akce na starost?



Řízení hry. Kdo má tyto akce na starost?



Řízení hry. Kdo má tyto akce na starost?



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

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

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

obecná definice hráče

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

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

obecná definice hráče

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

docstring - popis chování

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

obecná definice hráče

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

docstring - popis chování

metoda - to co má na starost 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 select_move(self):
5         return 'R'
6
7 if __name__ == '__main__':
8     p = MyPlayer() # creating one player
9     print(p.select_move()) # showing what they played
```

docstring - popis chování

metoda - to co má na starost hráč

proved', pokud je spuštěno jako hlavní program

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

obecná definice hráče

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

docstring - popis chování

metoda - to co má na starost hráč

proved', pokud je spuštěno jako hlavní program

vytvoř konkrétního hráče

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

obecná definice hráče

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

docstring - popis chování

metoda - to co má na starost hráč

proved', pokud je spuštěno jako hlavní program

vytvoř konkrétního hráče

konkrétní hráč hraje

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 select_move(self):
8         return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer() # creating a default player
12     print(p1.select_move()) # showing answer of p1
13     p2 = MyPlayer('P') # a better player?
14     print(p2.select_move()) # showing answer of p2
15     p1.answer = 'S' # p1 changed his mind
16     print(p1.select_move())
```

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 select_move(self):
8         return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer() # creating a default player
12     print(p1.select_move()) # showing answer of p1
13     p2 = MyPlayer('P') # a better player?
14     print(p2.select_move()) # showing answer of p2
15     p1.answer = 'S' # p1 changed his mind
16     print(p1.select_move())
```

konstruktor - při vytvoření objektu

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 select_move(self):
8         return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer() # creating a default player
12     print(p1.select_move()) # showing answer of p1
13     p2 = MyPlayer('P') # a better player?
14     print(p2.select_move()) # showing answer of p2
15     p1.answer = 'S' # p1 changed his mind
16     print(p1.select_move())
```

konstruktor - při vytvoření objektu

přiřazení atributu objektu

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 select_move(self):
8         return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer() # creating a default player
12     print(p1.select_move()) # showing answer of p1
13     p2 = MyPlayer('P') # a better player?
14     print(p2.select_move()) # showing answer of p2
15     p1.answer = 'S' # p1 changed his mind
16     print(p1.select_move())
```

konstruktor - při vytvoření objektu

přiřazení atributu objektu

každý hráč bude hrát podle svého nastavení

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 select_move(self):
8         return self.answer
9
10 if __name__ == '__main__':
11     p1 = MyPlayer() # creating a default player
12     print(p1.select_move()) # showing answer of p1
13     p2 = MyPlayer('P') # a better player?
14     print(p2.select_move()) # showing answer of p2
15     p1.answer = 'S'
16     print(p1.select_move())
```

konstruktor - při vytvoření objektu

přiřazení atributu objektu

každý hráč bude hrát podle svého nastavení

atributy existujícího objektu můžu měnit

Krokování, vizualizace

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

Python 3.6
[\(known limitations\)](#)

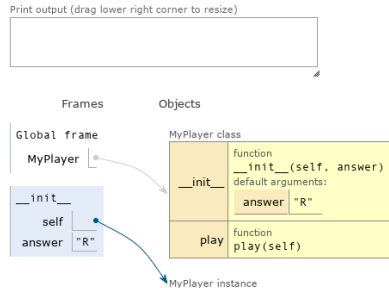
```
1 class MyPlayer:
2
3     def __init__(self, answer='R'):
4         self.answer = answer
5
6     def play(self):
7         return self.answer
8
9 if __name__ == '__main__':
10     p1 = MyPlayer()
11     p2 = MyPlayer('P')
12     print(p1.play())
13     print(p2.play())
```

[Edit this code](#)

→ line that just executed
→ next line to execute

Step 5 of 18

[Customize visualization](#)



<https://pythontutor.com>

Hráč s pamětí – playerdummy_plus_mem.py

```
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 select_move(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.select_move()
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

```
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 select_move(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.select_move()
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)

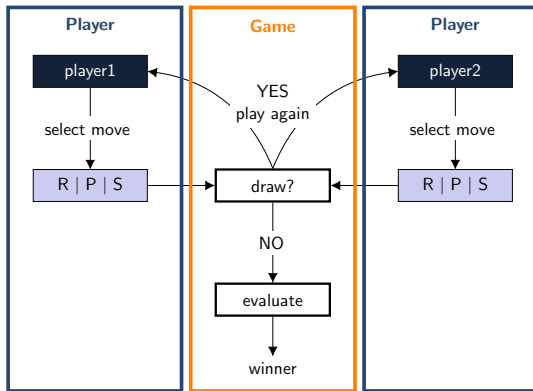
Hráč s pamětí – playerdummy_plus_mem.py

```
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 select_move(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.select_move()
17        p1.record(answer2) # p1 records what p2 played
18        print(answer2)
19        print(p1.history) # check that recording works
```

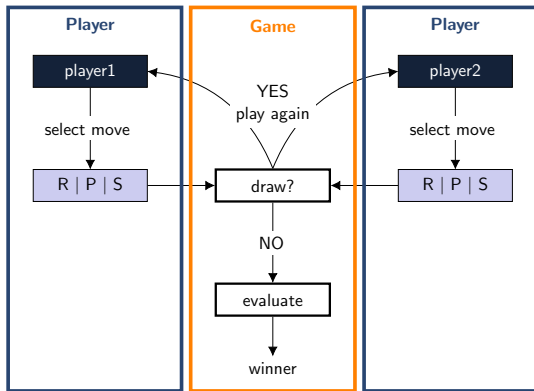
každý hráč bude mít svůj prázdný seznam (list)

přidej *move* na konec seznamu
append je metoda seznamu

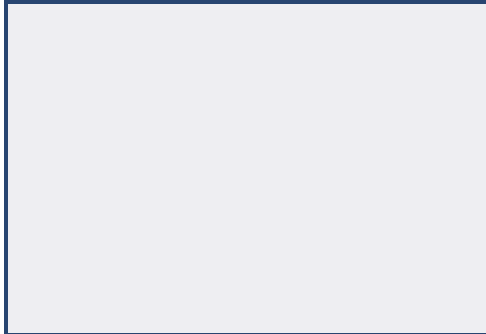
Řízení hry. Jak převedeme schéma do kódu?



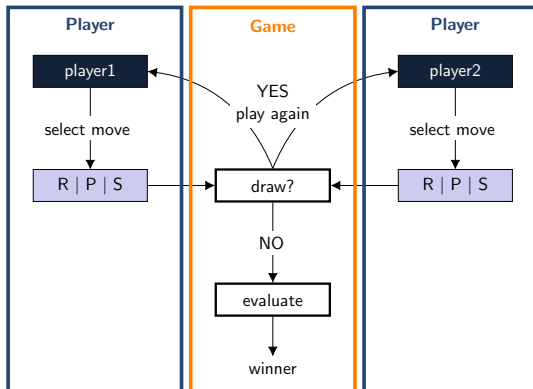
Řízení hry. Jak převedeme schéma do kódu?



pseudokód



Řízení hry. Jak převedeme schéma do kódu?

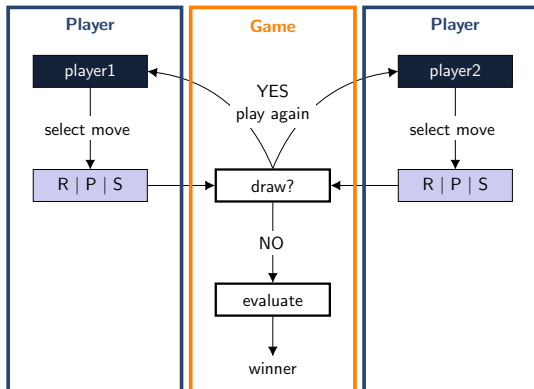


pseudokód

```
p1 = Player
```

```
p2 = Player
```

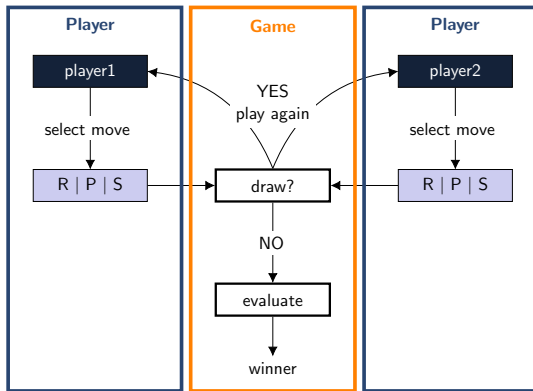
Řízení hry. Jak převedeme schéma do kódu?



pseudokód

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

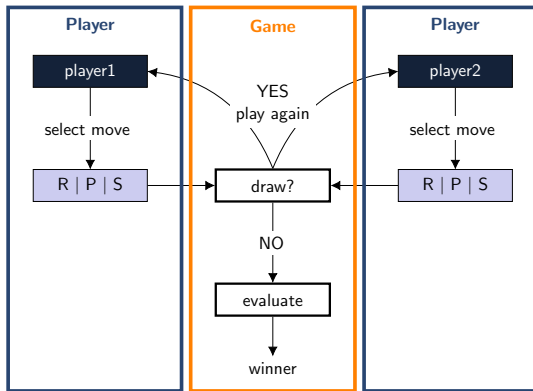
Řízení hry. Jak převedeme schéma do kódu?



pseudokód

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

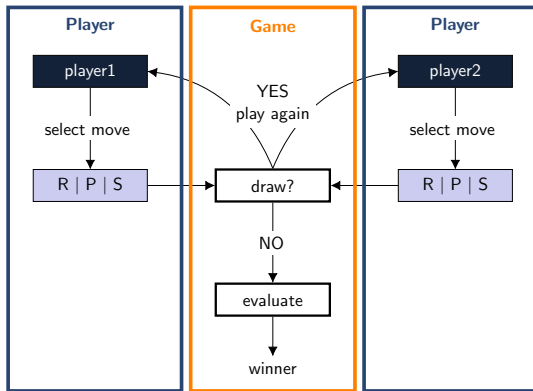
Řízení hry. Jak převedeme schéma do kódu?



pseudokód

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

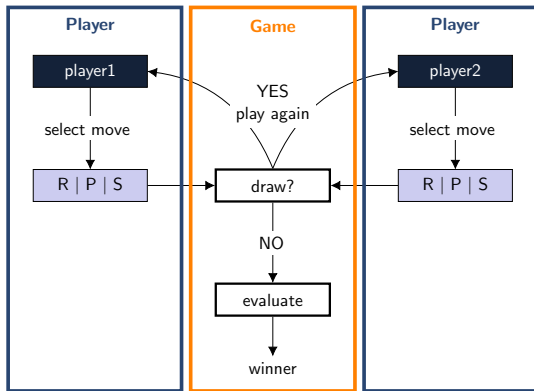
Řízení hry. Jak převedeme schéma do kódu?



pseudokód

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


Řízení hry. Jak převedeme schéma do kódu?



pseudokód

```
p1 = Player
p2 = Player
draw = True
while draw:
    move1 = p1.select_move
    move2 = p2.select_move
    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.select_move()
11            move2 = self.p2.select_move()
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.select_move()
11            move2 = self.p2.select_move()
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.select_move()
11            move2 = self.p2.select_move()
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.select_move()
11            move2 = self.p2.select_move()
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.select_move()
11            move2 = self.p2.select_move()
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

vytvářím 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

vytvářím 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

vytvářím hráče
(přístup ke kódu z modulu)

vytvářím hru, přidávám hráče

odehraj jedno kolo

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

vytvářím hráče
(přístup ke kódu z modulu)

vytvářím hru, přidávám hráče

odehraj jedno kolo

vypiš vítěze

Opakovaná 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
```

Opakovaná hra - na N vítězých

```
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

Opakovaná hra - na N vítězých

```
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

Opakovaná hra - na N vítězých

```
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

Jaká byla dnes rychlost výkladu?

- (A) Příště přidejte.
- (B) Tak akorát.
- (C) Prosím zpomalte.
- (D) Ona už přednáška skončila?

Díky za pozornost

Dnešní látka pro vás byla:

- (A) Lehká až triviální.
- (B) Tak akorát.
- (C) Těžká, ztrácel/a jsem se.
- (D) ...