

# Kámen-nůžky-papír

Tomáš Svoboda

Katedra kybernetiky

Fakulta elektrotechnická, České vysoké učení technické

Studijní program: [Otevřená informatika](#)

2021-10-05

# než začneme ...

- dívka jménem Lenka
- doplňková cvičení, čtvrtky 16:30-17:30, KN:E-310, kolega Ondřej Kyzr, kyzrondr@fel.cvut.cz

## 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

Vaše pořadí odevzdejte pomocí [brute](https://cw.felk.cvut.cz/brute)<sup>3</sup>. Poradí vám na prvním cvičení. K otázce se vrátíme ještě na příští přednášce.

---

<sup>3</sup><https://cw.felk.cvut.cz/brute>

Pokud všichni studenti zvolí pro svou odpověď **velká písmena**, dostane každý student ještě další jeden bod. Pokud bude odpověď alespoň jednoho studenta tvořena **malými písmeny**, všichni, kdo odpověděli velkými písmeny, nedostanou žádné další body a ti, kteří použili malá písmena, dostanou navíc jen 0.5 bodů. Volby jednotlivců nebudou zveřejněny. Výsledek se dozvíte opět později.

# Lenka - výsledky

# Lenka - výsledky

- Odevzdaných řešení: 156

# Lenka - výsledky

- Odevzdaných řešení: 156
- Syntakticky správně: 154

# Lenka - výsledky

- Odevzdaných řešení: 156
- Syntakticky správně: 154
- (Sémanticky) Správná odpověď: 46



# Lenka - výsledky

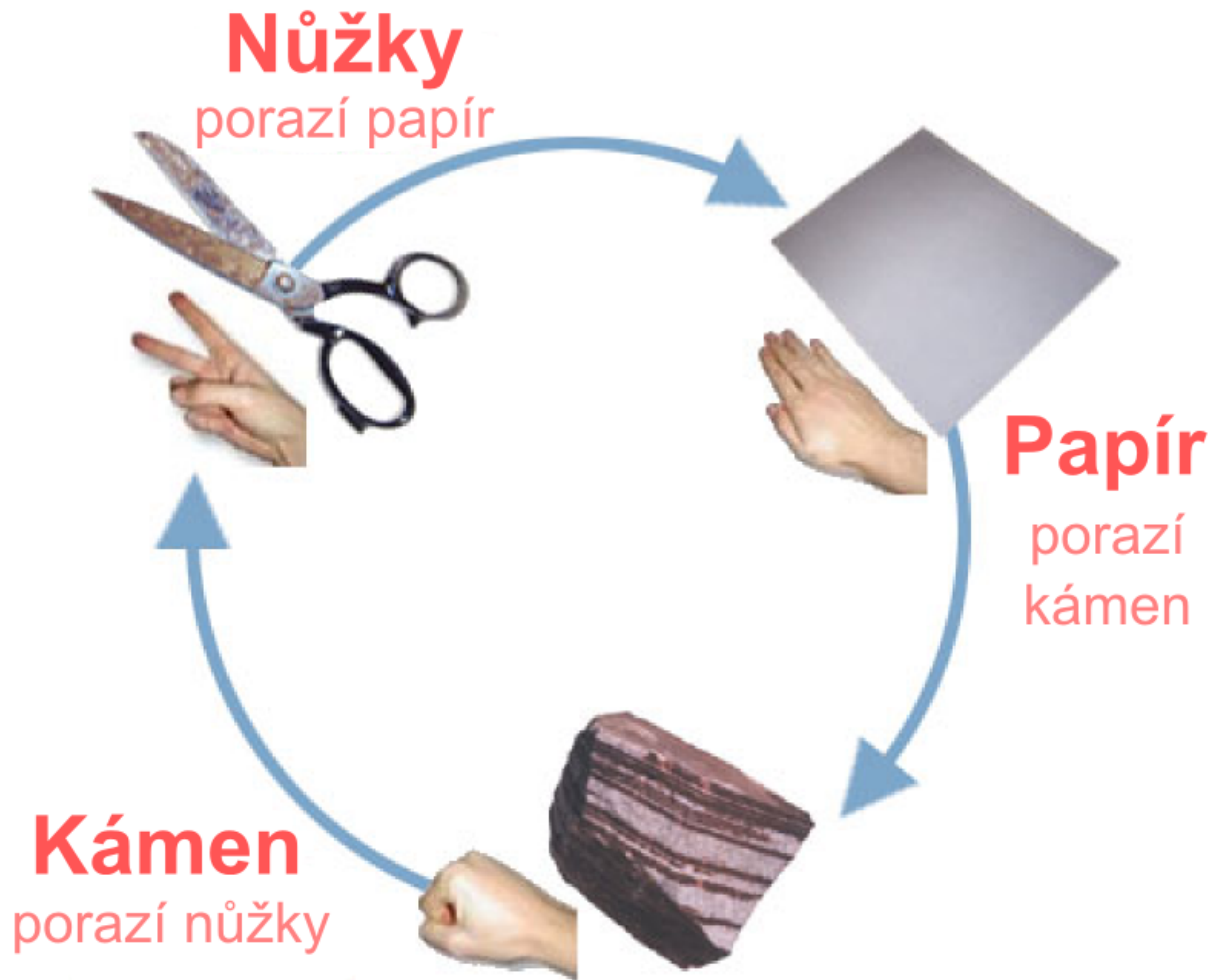
- Odevzdaných řešení: 156
- Syntakticky správně: 154
- (Sémanticky) Správná odpověď: 46
- velkými písmeny: 40

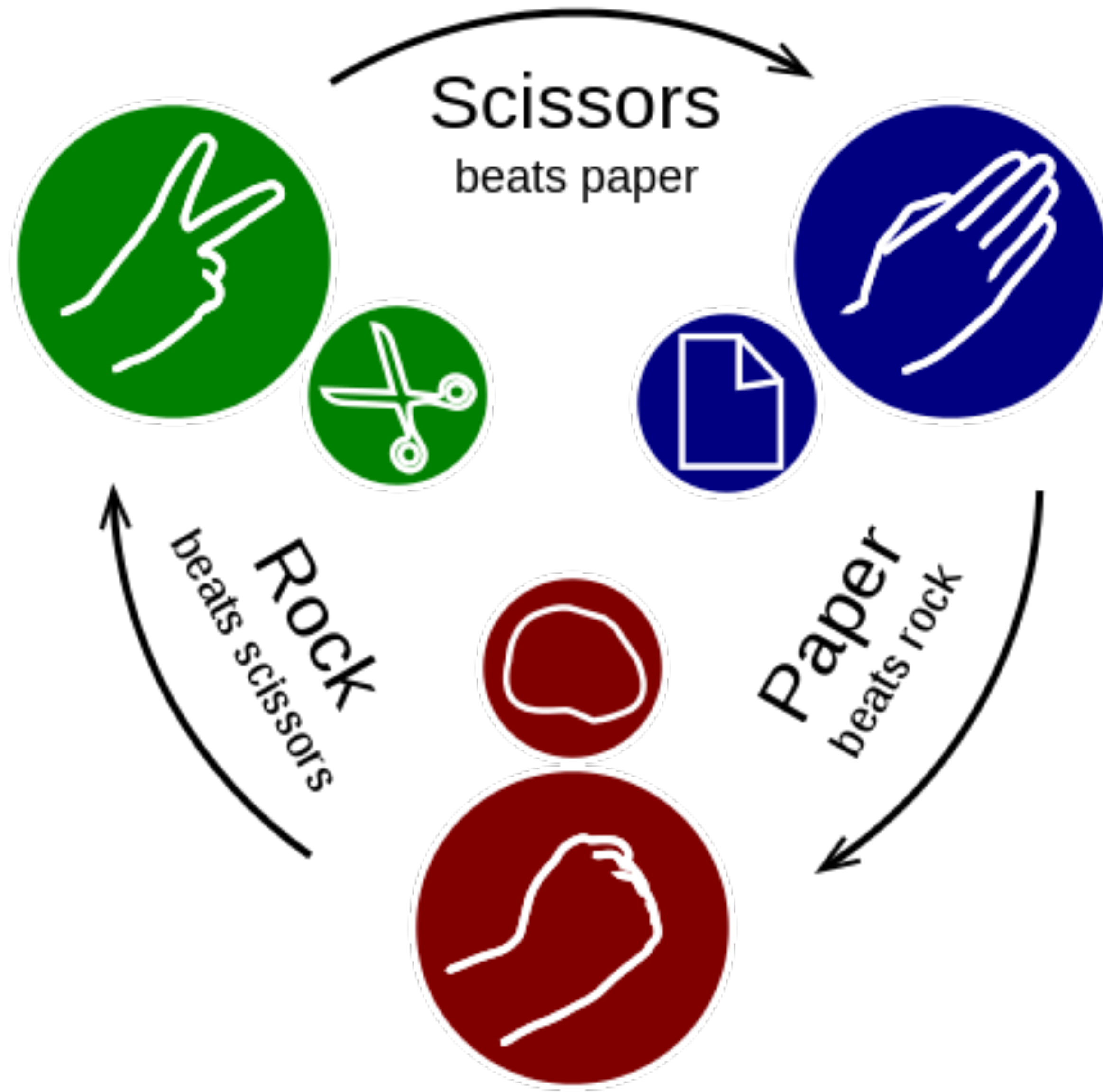
# Poznámka k programovacím úlohám

- nutná vlastní práce
- úlohy voleny tak, aby dovolily naprogramování vlastními silami
- zapomínací test
- [https://cw.fel.cvut.cz/wiki/help/common/plagiaty\\_opisovani](https://cw.fel.cvut.cz/wiki/help/common/plagiaty_opisovani)

# Kámen-nůžky-papír

- kódy budou po přednášce ke stažení
- velejemný úvod do objektů
- málokdy existuje **jediné správné** řešení
- vítame: bug-reports, doporučení ke zlepšení
- kódy v přednášce odpovídají starší verzi





player 1

player 2

player 1

“**R**ock-**P**aper-**S**cissors ...”

player 2

“**R**ock-**P**aper-**S**cissors ...”

player 1

“**R**ock-**P**aper-**S**cissors ...”

R | P | S

player 2

“**R**ock-**P**aper-**S**cissors ...”

R | P | S



player 1

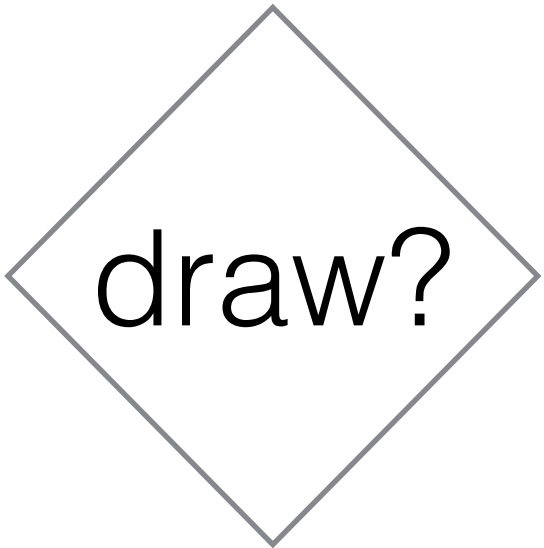
“**R**ock-**P**aper-**S**cissors ...”

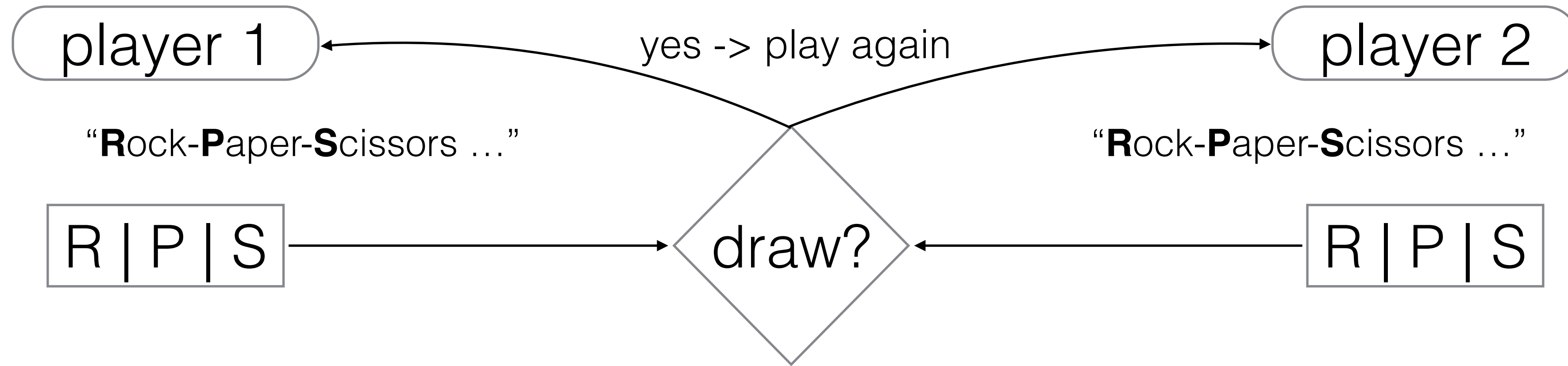
R | P | S

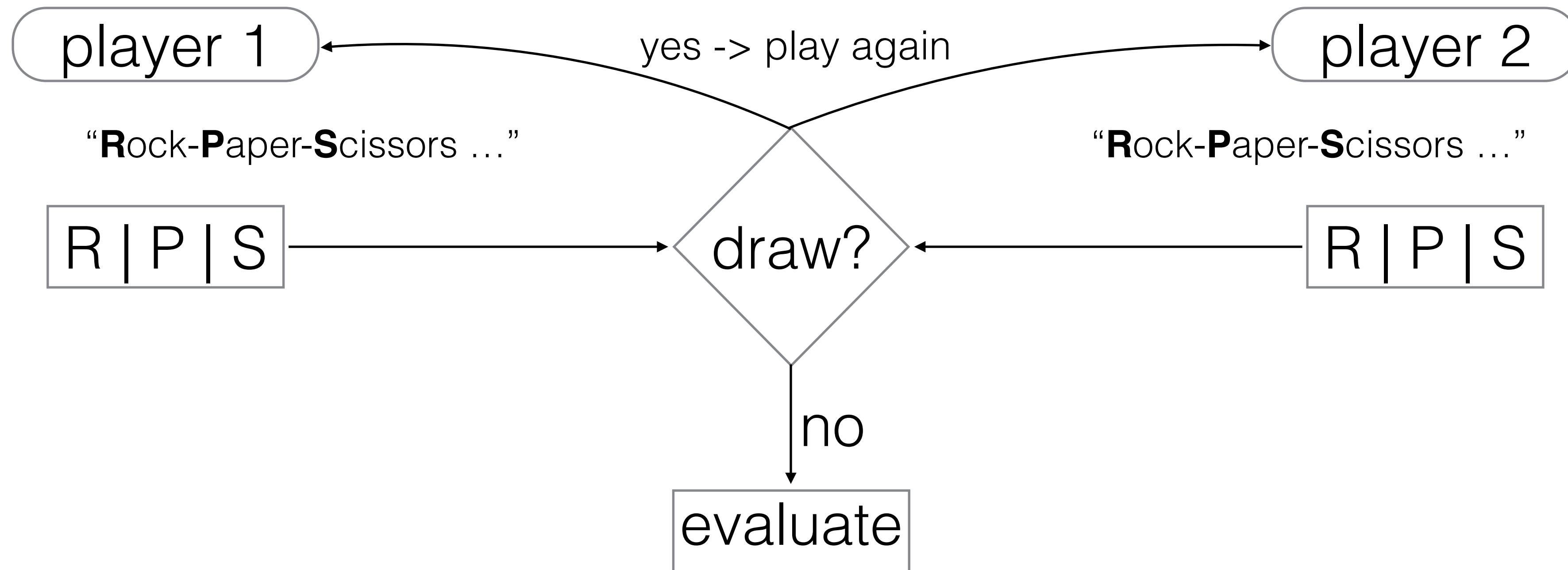
player 2

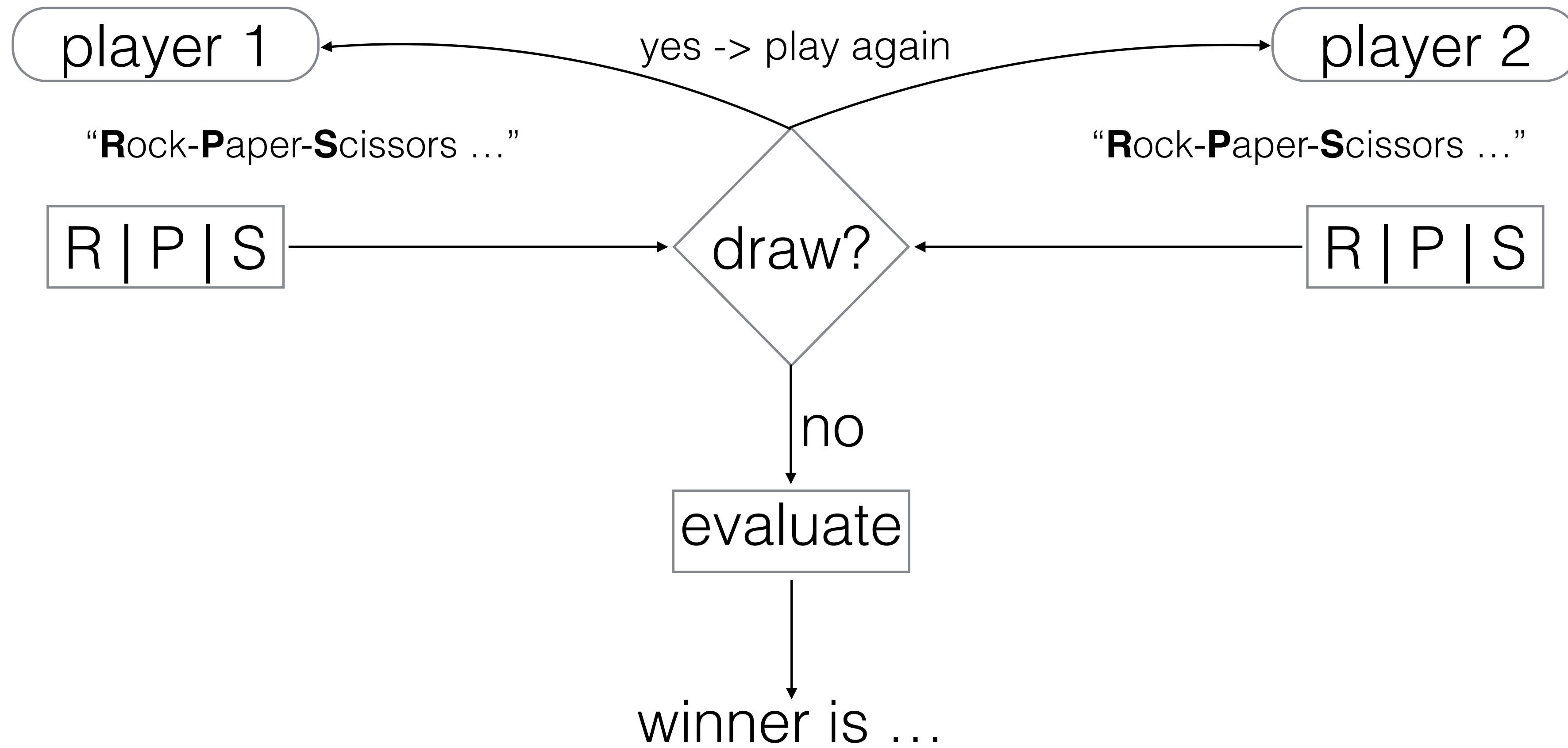
“**R**ock-**P**aper-**S**cissors ...”

R | P | S

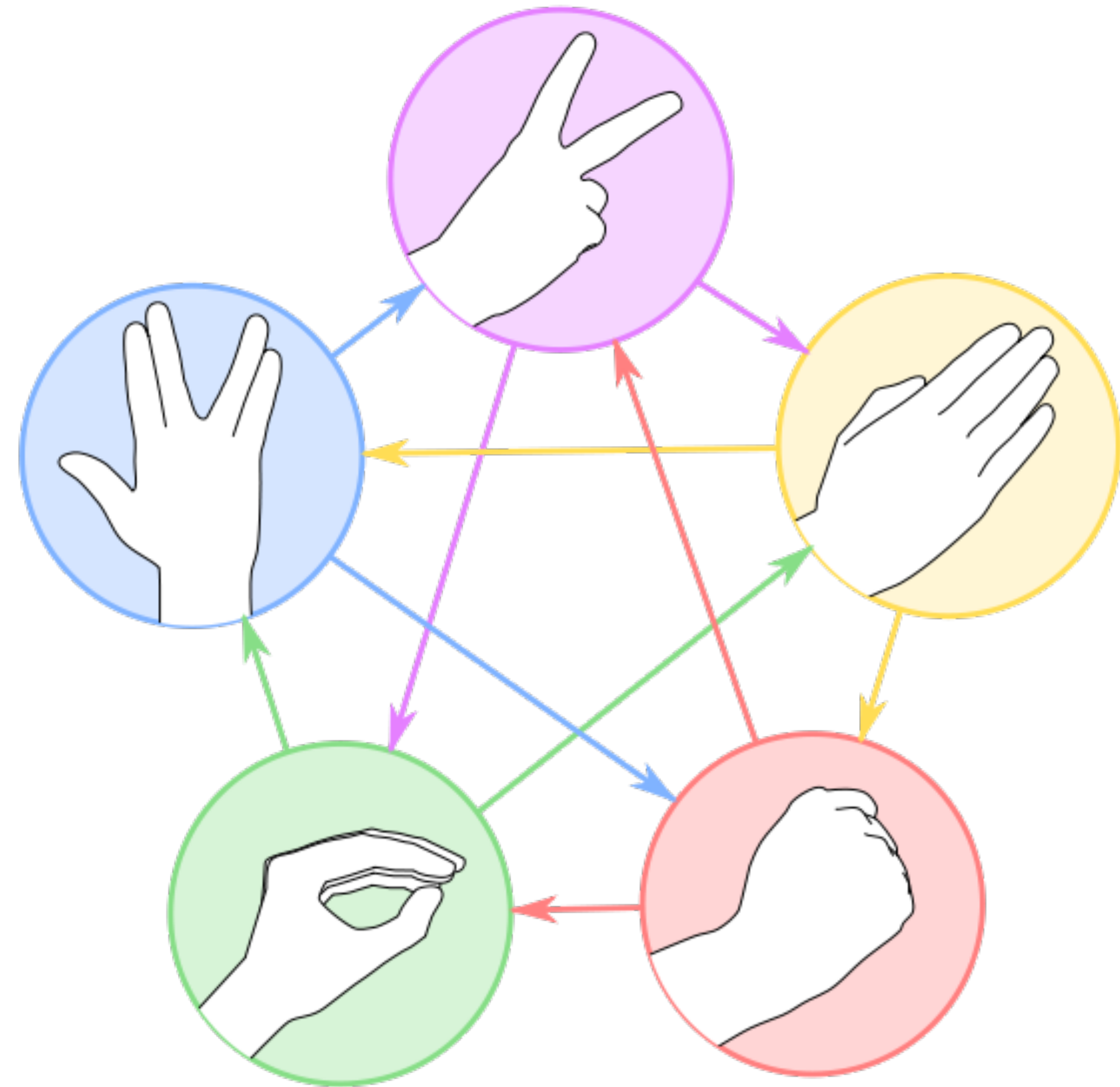








# Vyšší level



# Python vsuvka

- paměťová místa s daty jsou **odkazována** (viz PRP)
- typ proměnné, resp. paměťové místo přiděleno dynamicky, podle přiřazení (podle toho co je napravo)
- při ztrátě ukazatele (smazání, mimo platnost, ...) se paměť čistí (Garbage collector)
- téměř všechno v Pythonu je objekt nějaké třídy
- v Pythonu můžeme ukazovat/odkazovat na **všechno**

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

**A:** skončí chybou

**B:** c bude typu int

**C:** c bude typu float

Nepřeskakuj, dokud  
neodpovíš!

>>> 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]

Nepřeskakuj, dokud neodpovíš!



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

**A:** d je 1

**B:** d je -1

**C:** d je True

**D:** skončí chybou

Nepřeskakuj, dokud neodpovíš!

# pár příkladů z Python konsole

```
[92] python3
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 00:54:21)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> help(str)

>>> help(int)

>>> a = 1
>>> help(a)

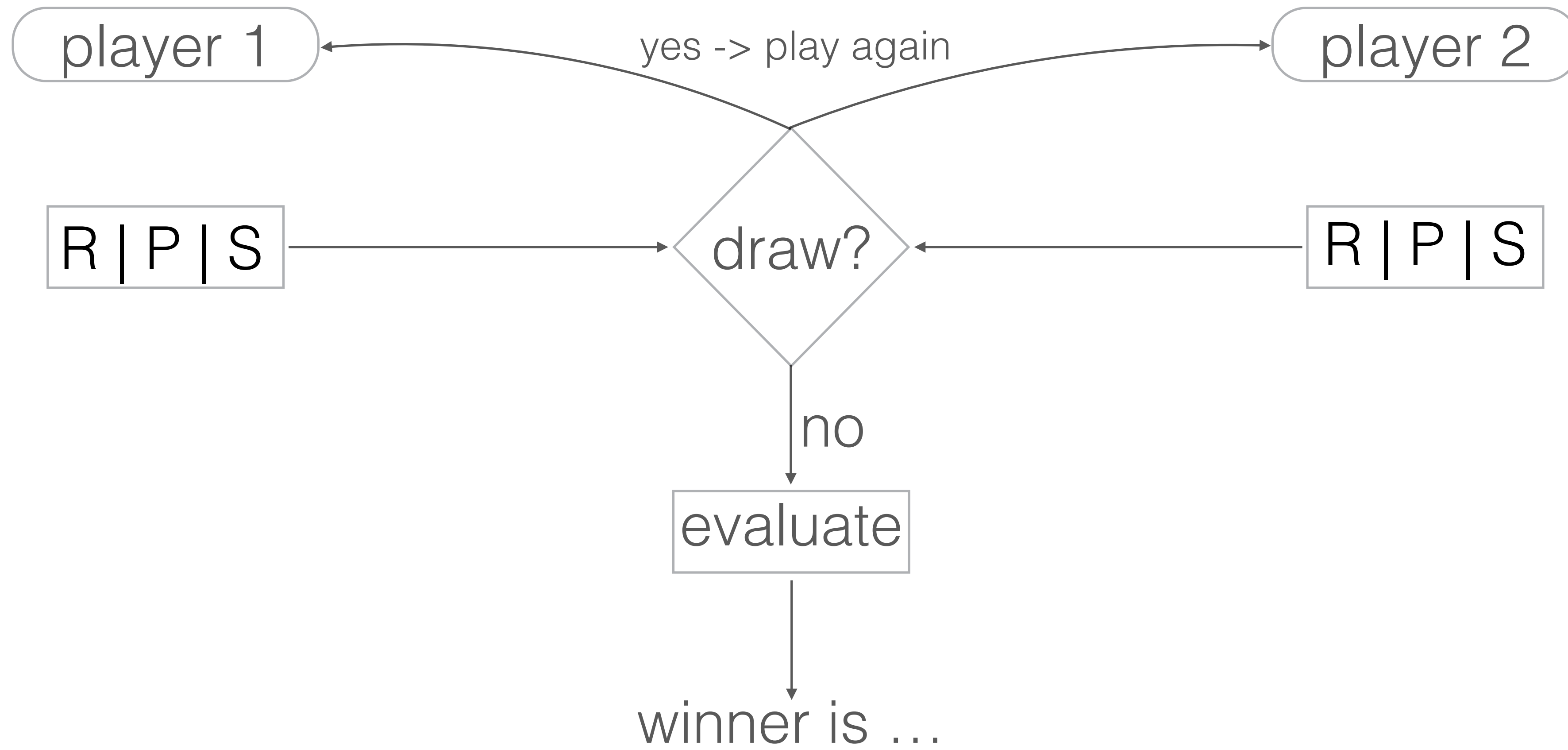
>>> b = True
>>> help(b)

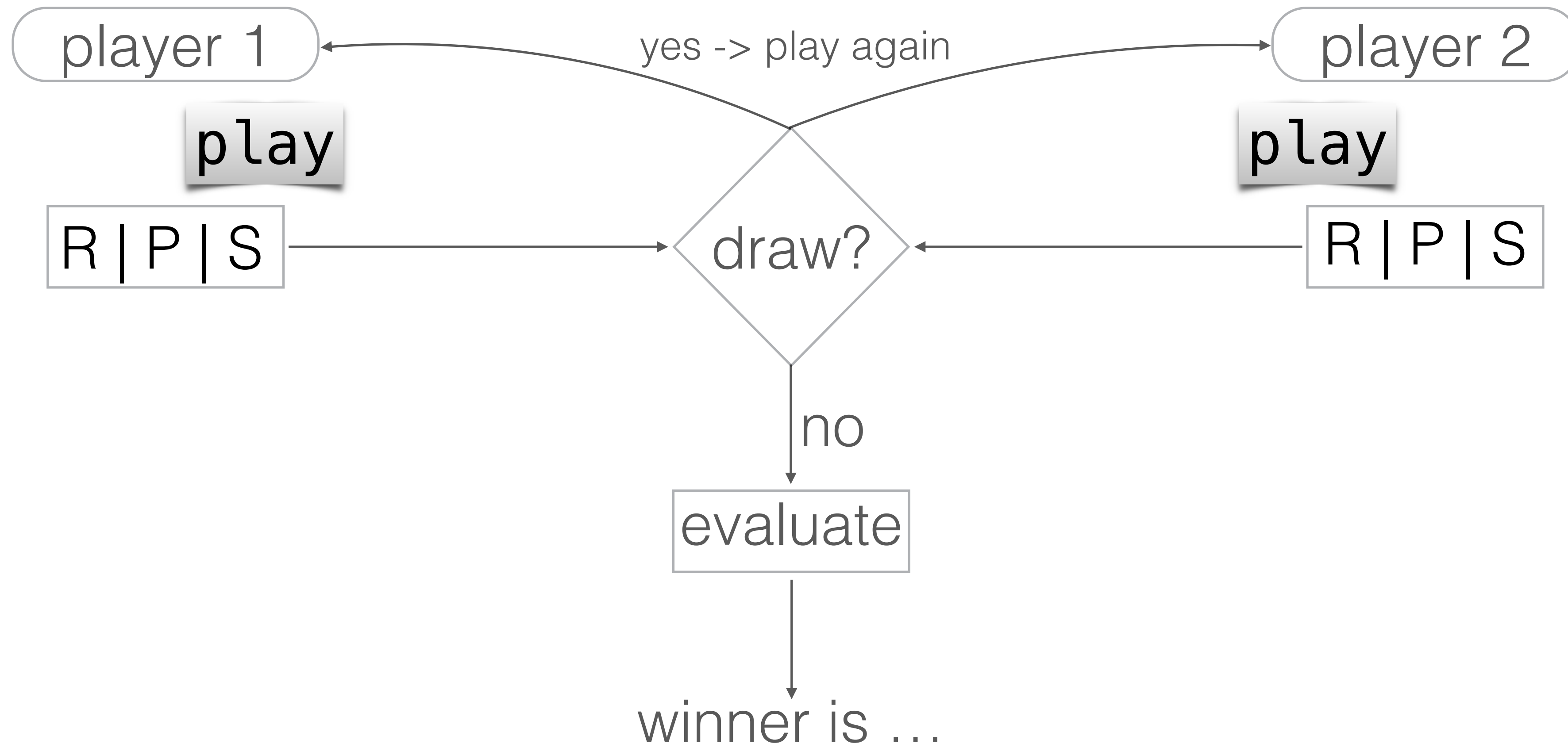
>>> help(abs)

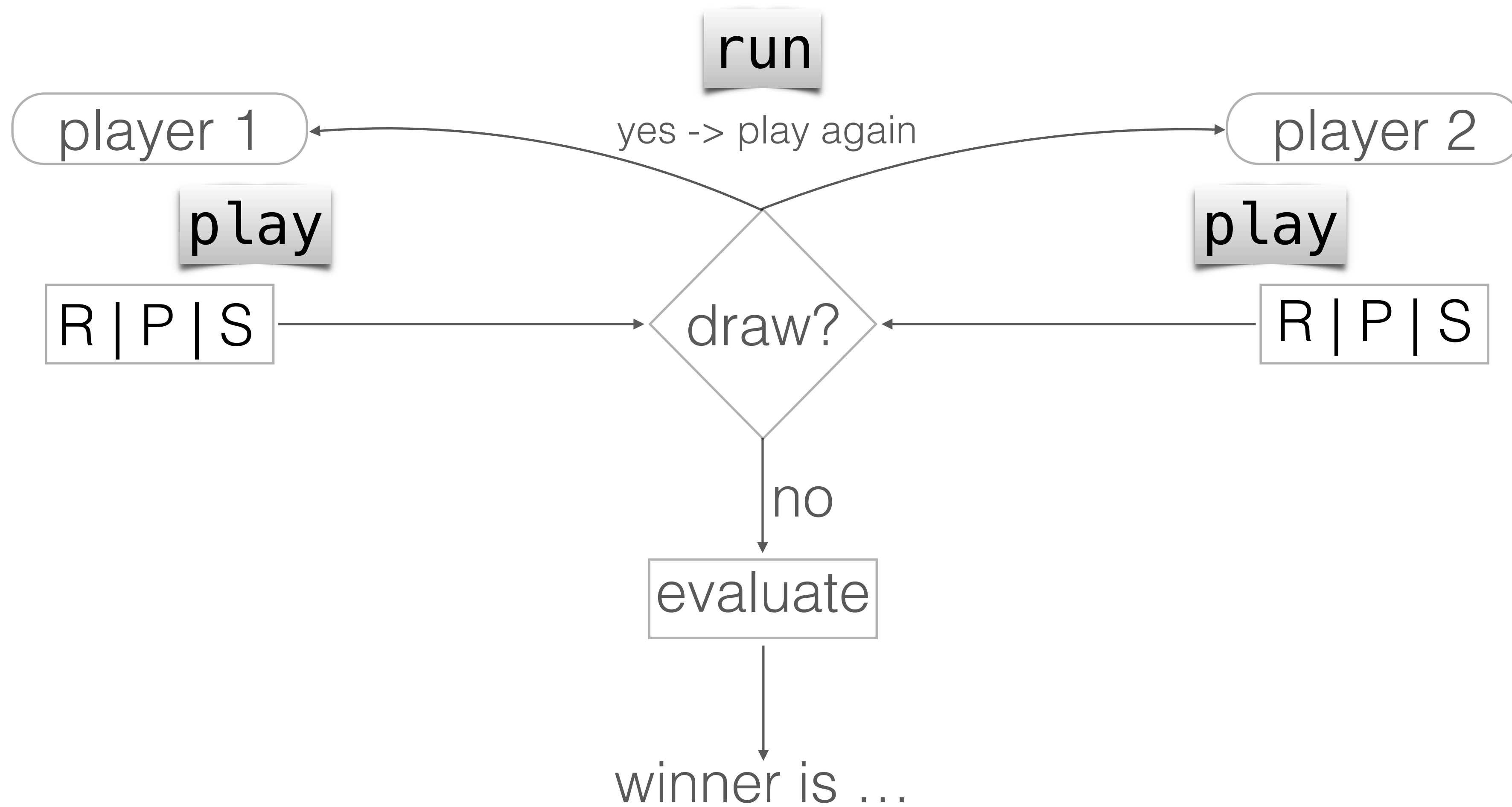
>>> c = abs
>>> c(-2)
2
```

# pokud si nejsme si jisti, python konzole to jistí

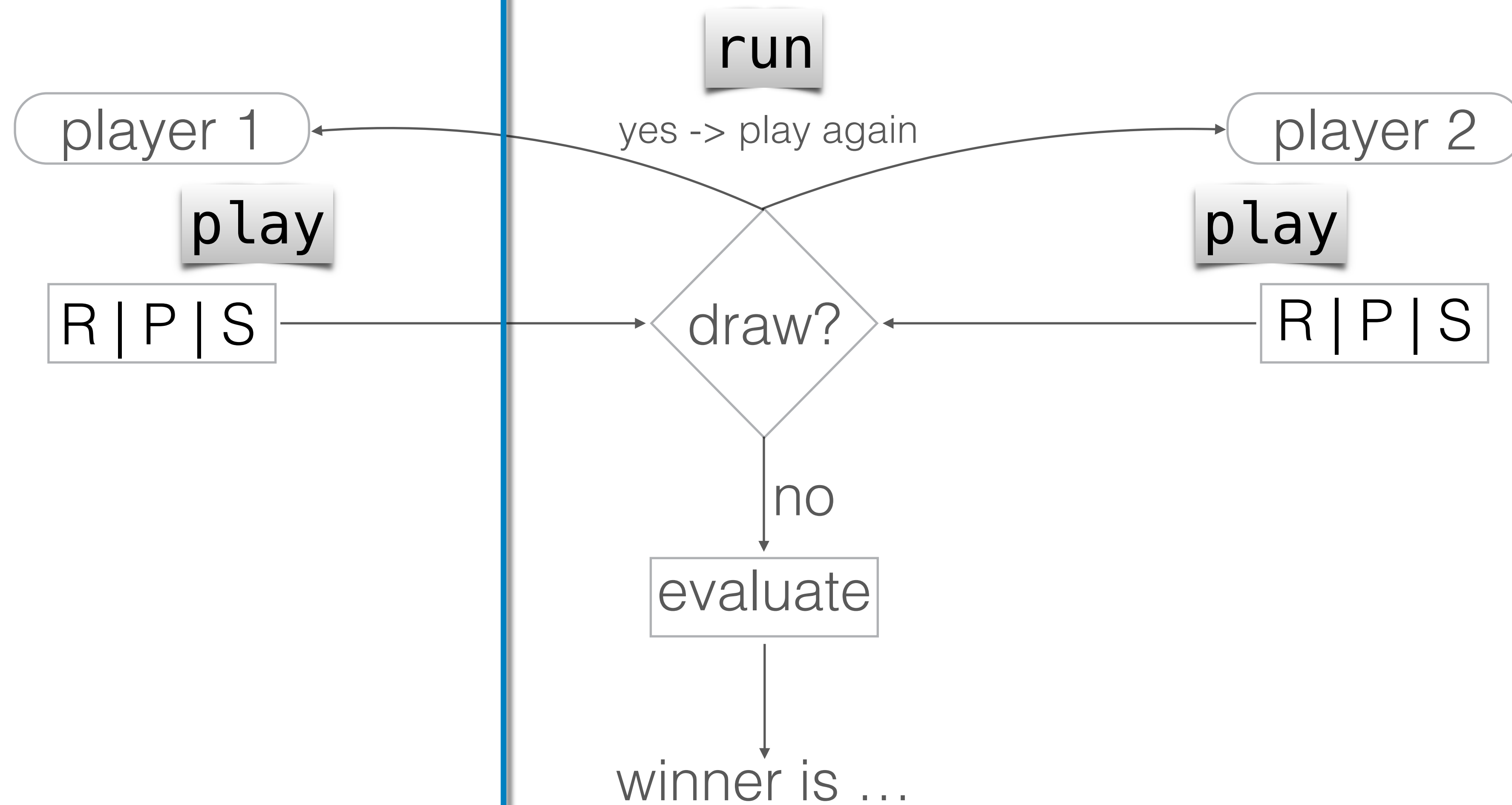
- `type(variable_name)`
- `isinstance(variable_name, data_type)`
- `help(variable_name)`
- klávesa Tab (autocomplete)
- ask Google ()
- <https://docs.python.org/3/tutorial/index.html>

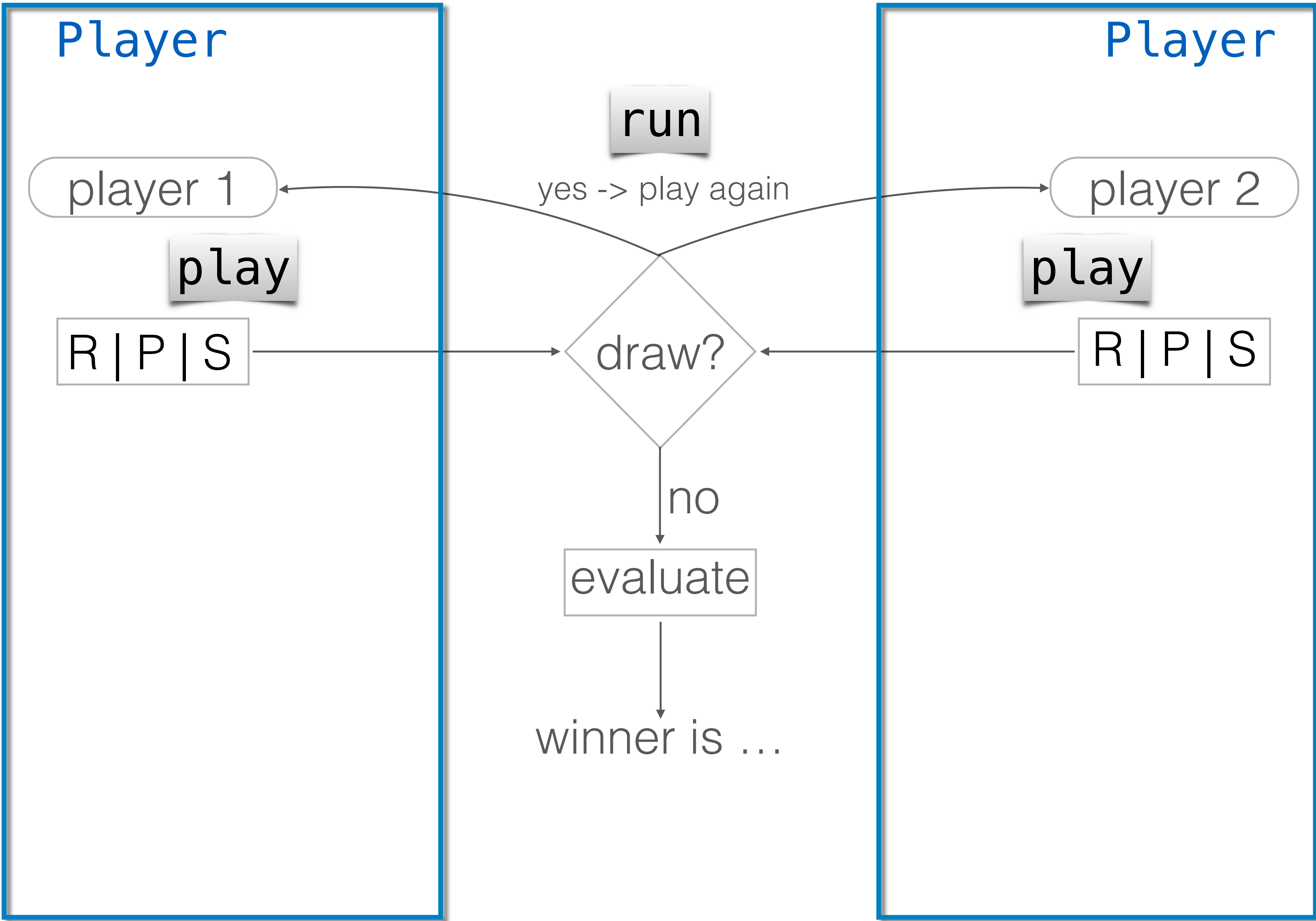




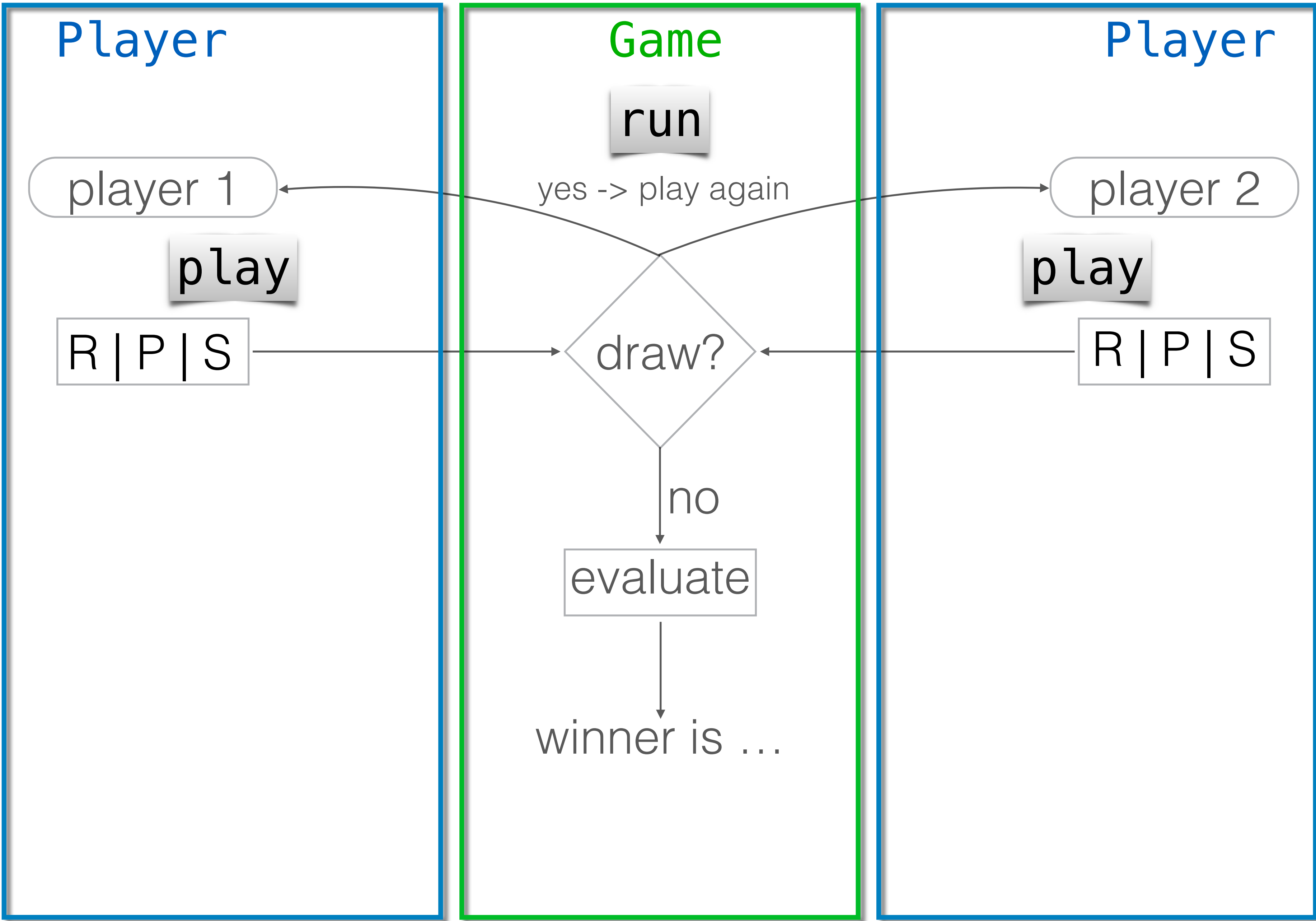


# Player









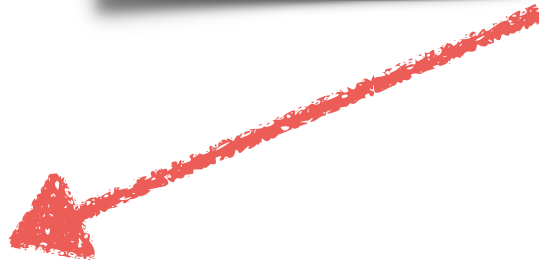
playerdummy.py – always play R

# playerdummy.py – always play R

```
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
```

# playerdummy.py – always play R

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
```

# playerdummy.py – always play R

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

# playerdummy.py – always play R

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'

# playerdummy.py – always play R

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

# playerdummy.py – always play R

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



# playerdummy.py – always play R

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

# playerdummy.py – always play R

```
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
```

obecná definice hráče

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?

playerdummyplus.py – play what is commanded

# playerdummyplus.py – play what is commanded

```
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
```

# playerdummyplus.py – play what is commanded

```
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

# playerdummyplus.py – play what is commanded

```
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

přičazení atributu objektu

# playerdummyplus.py – play what is commanded

```
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

# playerdummyplus.py – play what is commanded

```
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



# playerdummyplus.py – play what is commanded

```
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

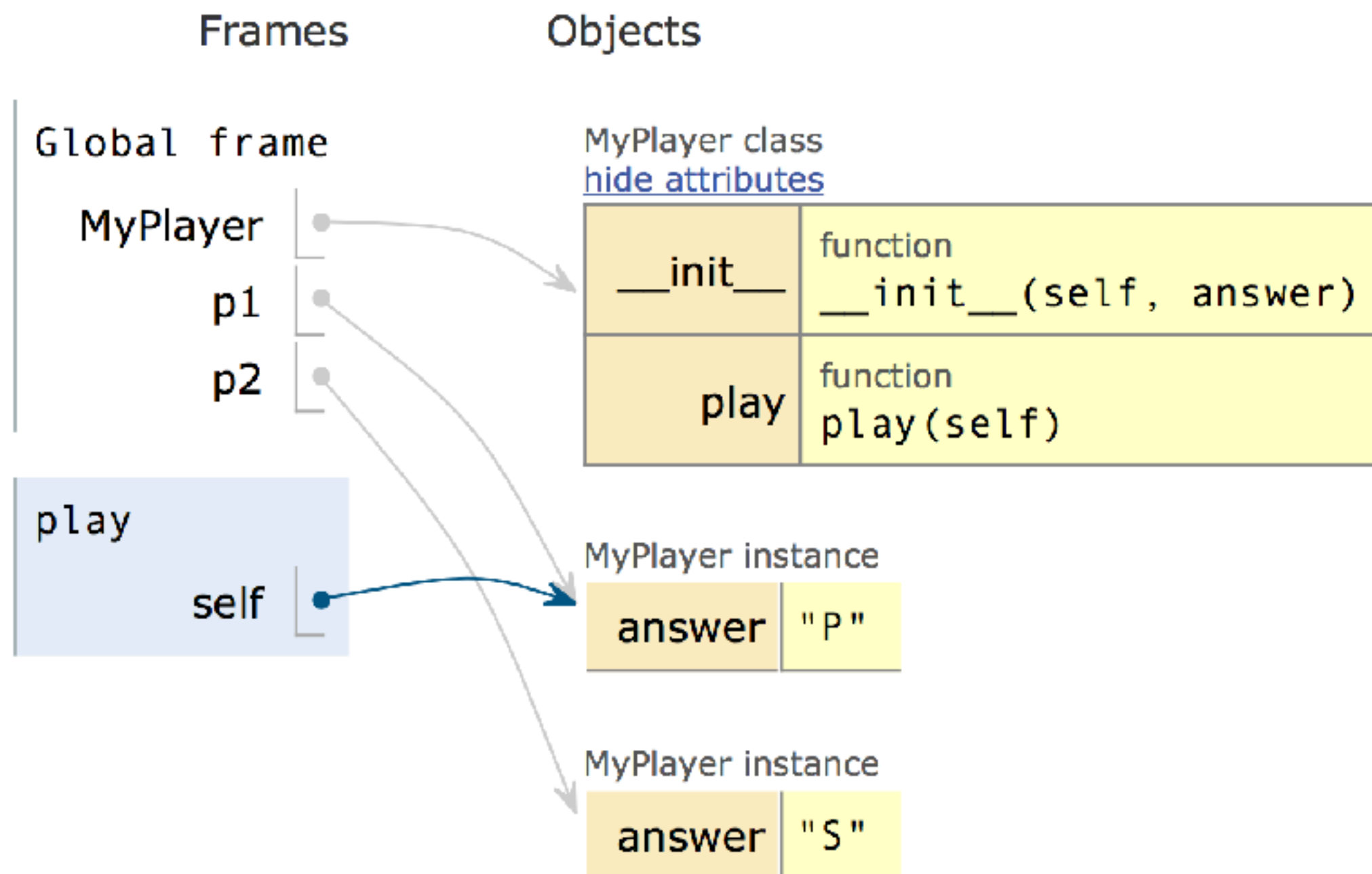
vlastnosti objektu mohu měnit

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()
```



<http://pythontutor.com/>

# playerdummyplusplus.py (dummy s pameti)

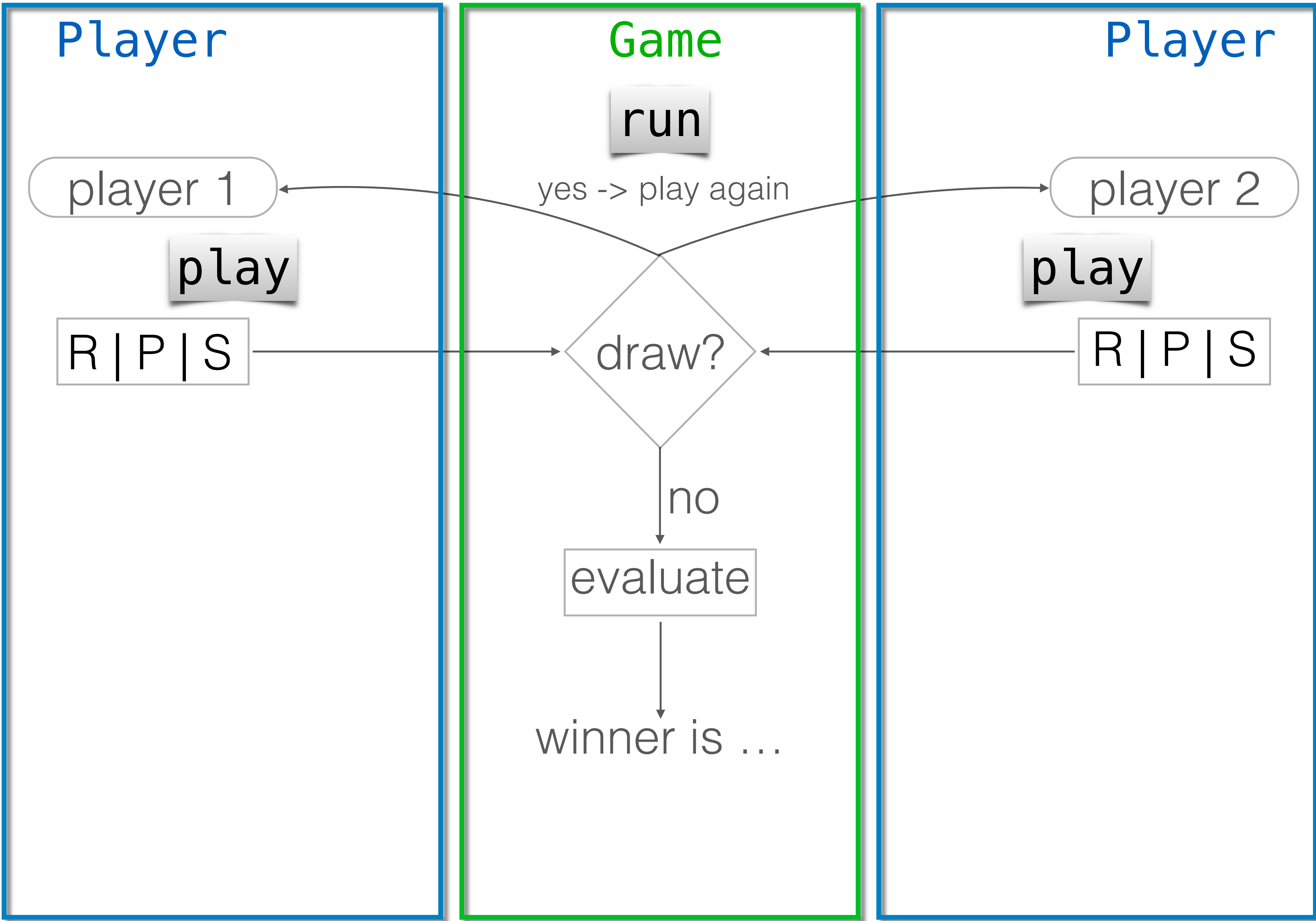
```
1 class MyPlayer:
2     '''A dummy player on steroids'''
3     def __init__(self, answer='R'):
4         self.answer = answer
5         self.history = []
6
7     def play(self):
8         return self.answer
9
10    def record(self, move):
11        self.history.append(move)
12
13 if __name__ == "__main__":
14     p1 = MyPlayer() # creating a default player
15     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
```

# playerdummyplusplus.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)
12
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
```

# playerdummyplusplus.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
```



Jak hrát hru?

# Jak hrát hru?

p1 = Player

p2 = Player



# Jak hrát hru?

p1 = Player

p2 = Player

draw = True

# Jak hrát hru?

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

# Jak hrát hru?

```
p1 = Player
```

```
p2 = Player
```

```
draw = True
```

```
while draw:
```

```
    move1 = p1.play
```

```
    move2 = p2.play
```

# Jak hrát hru?

```
p1 = Player
```

```
p2 = Player
```

```
draw = True
```

```
while draw:
```

```
    move1 = p1.play
```

```
    move2 = p2.play
```

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

# 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
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            score = evaluate_moves([move1,move2])
14            if score[0]>score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```

# 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
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            score = evaluate_moves ([move1, move2])
14            if score[0] > score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```

# 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
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            score = evaluate_moves([move1, move2])
14            if score[0] > score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```

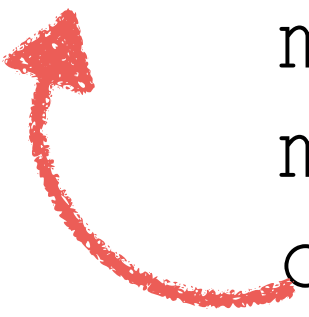


# Třída Game

```
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
6
7     def run (self):
8         draw = True     Dokud nehrají, tak je remíza
9         while draw:
10            move1 = self.p1.play()
11            move2 = self.p2.play()
12            draw = (move1 == move2)
13            score = evaluate_moves ([move1, move2])
14            if score[0] > score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```

# 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
6
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            score = evaluate_moves([move1, move2])
14            if score[0] > score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```



# 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
6
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            score = evaluate_moves([move1, move2])
14            if score[0] > score[1]:
15                self.winner = self.p1
16            else:
17                self.winner = self.p2
```

Porovnej tahy podle pravidel  
a přiřaď vítěze

# porovnej tahy

```
1 def evaluate_moves(move1, move2):
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 move1, move2 in (('P', 'R'), ('S', 'P'), ('R', 'S')):
9         return 1, 0
10    else:
11        return 0, 1
12
```

# porovnej tahy

```
1 def evaluate_moves(move1, move2):
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 move1, move2 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
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1,p2)
8     g.run()
9     print('Winner is:',g.winner)
10
```

# hlavní program game.py

```
1 import playertom
2 import playerdummy
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1,p2)
8     g.run()
9     print('Winner is:',g.winner)
10
```

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

# hlavní program game.py

```
1 import playertom
2 import playerdummy
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1,p2)
8     g.run()
9     print('Winner is:',g.winner)
10
```

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

vytvoř hráče



# hlavní program game.py

```
1 import playertom
2 import playerdummy
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1, p2)
8     g.run()
9     print('Winner is:', g.winner)
10
```

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

vytvoř hráče

inicializuj hru, předej hráče

# hlavní program game.py

```
1 import playertom
2 import playerdummy
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1, p2)
8     g.run()
9     print('Winner is:', g.winner)
10
```

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

vytvoř hráče

inicializuj hru, předej hráče

hraj

# hlavní program game.py

```
1 import playertom
2 import playerdummy
3     tady patří class Game: a spol. ...
4 if __name__ == "__main__":
5     p1 = playertom.MyPlayer()
6     p2 = playerdummy.MyPlayer()
7     g = Game(p1, p2)
8     g.run()
9     print('Winner is:', g.winner)
10
```

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

vytvoř hráče

inicializuj hru, předej hráče

hraj

oznam vítěze

# iterative game

```
9 class Game:
10     def __init__(self, p1, p2, minwins):
11         self.p1 = p1
12         self.p2 = p2
13         self.minwins = minwins
14
15     def duel(self):
16         draw = True
17         while draw:
18             p1_move = self.p1.play()
19             p2_move = self.p2.play()
20             draw = (p1_move == p2_move)
21         return evaluate(p1_move, p2_move)
22
23     def run(self):
24         draw = True
25         total_score = [0,0]
26         while max(total_score) < self.minwins:
27             score = self.duel()
28             total_score[0] += score[0]
29             total_score[1] += score[1]
30         if total_score[0] > total_score[1]:
31             return self.p1
32         else:
33             return self.p2
```