Rock-Paper-Scissors
Intro into objective Python

Tomáš Svoboda

Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University
2018-02

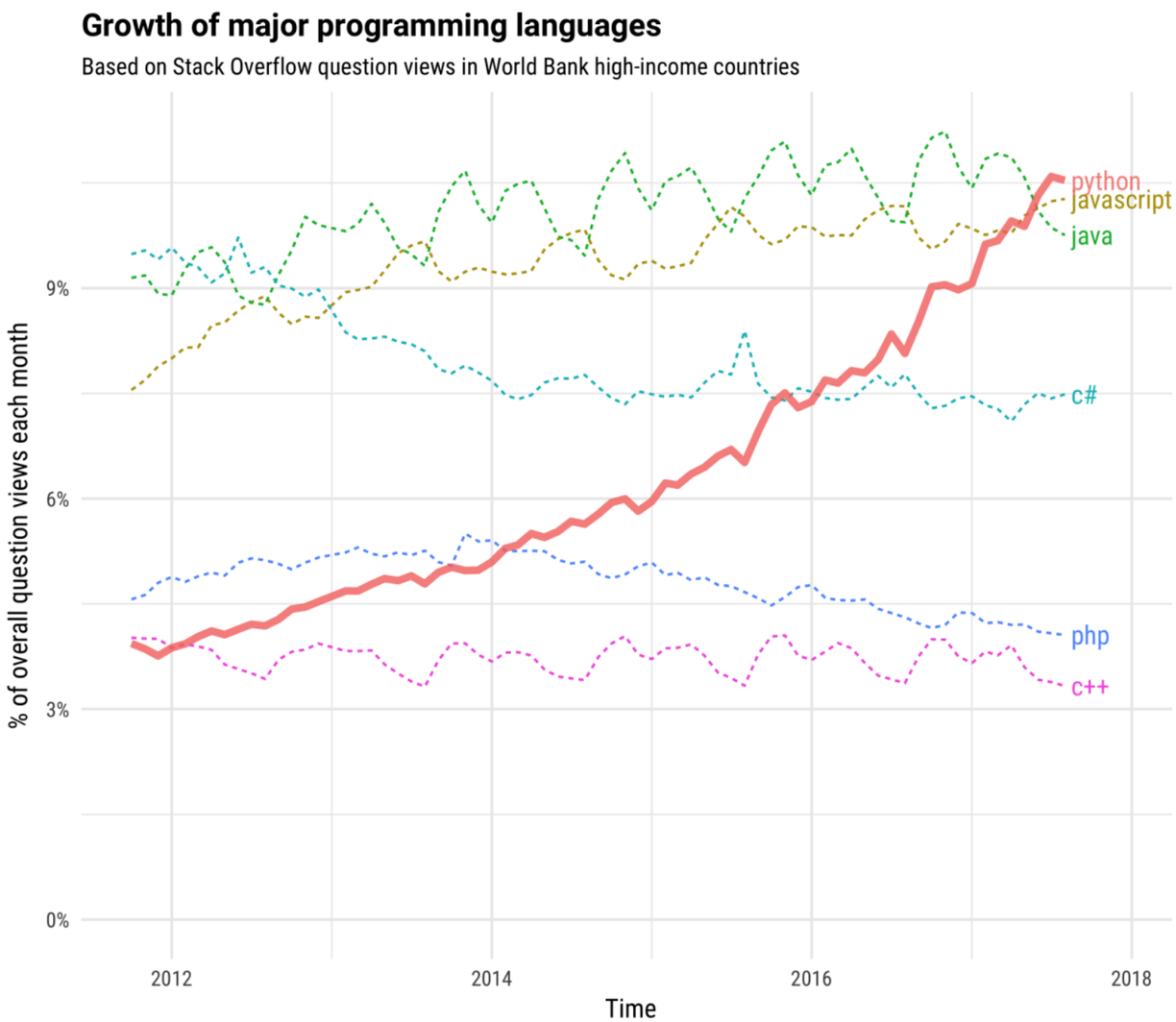
Reading, materials

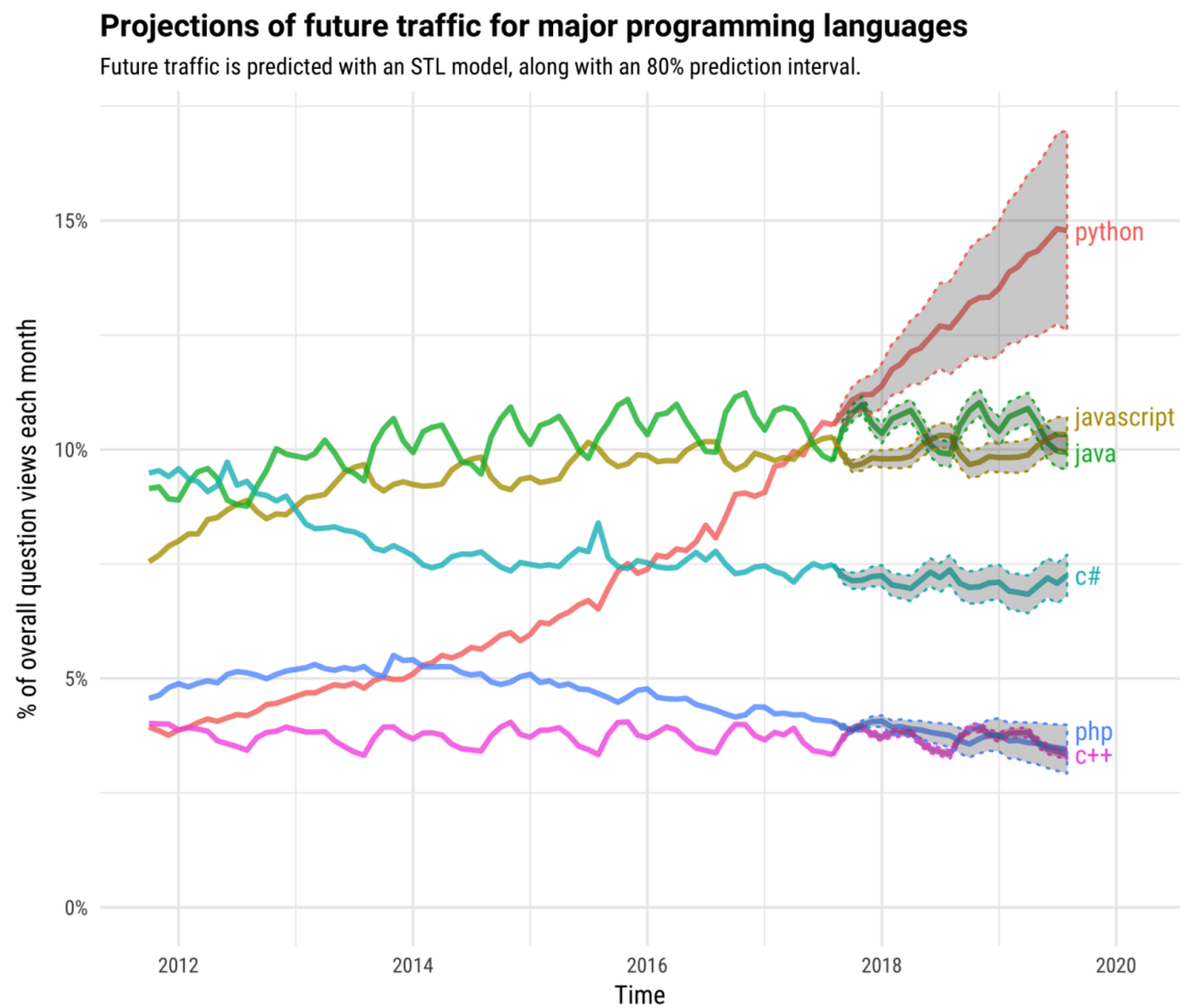
- https://cw.fel.cvut.cz/b172/courses/be5b33kui/literature
- https://cw.fel.cvut.cz/old/courses/be5b33prg/tutorials/start
- https://cw.fel.cvut.cz/old/courses/be5b33prg/lectures/start

Why Python?

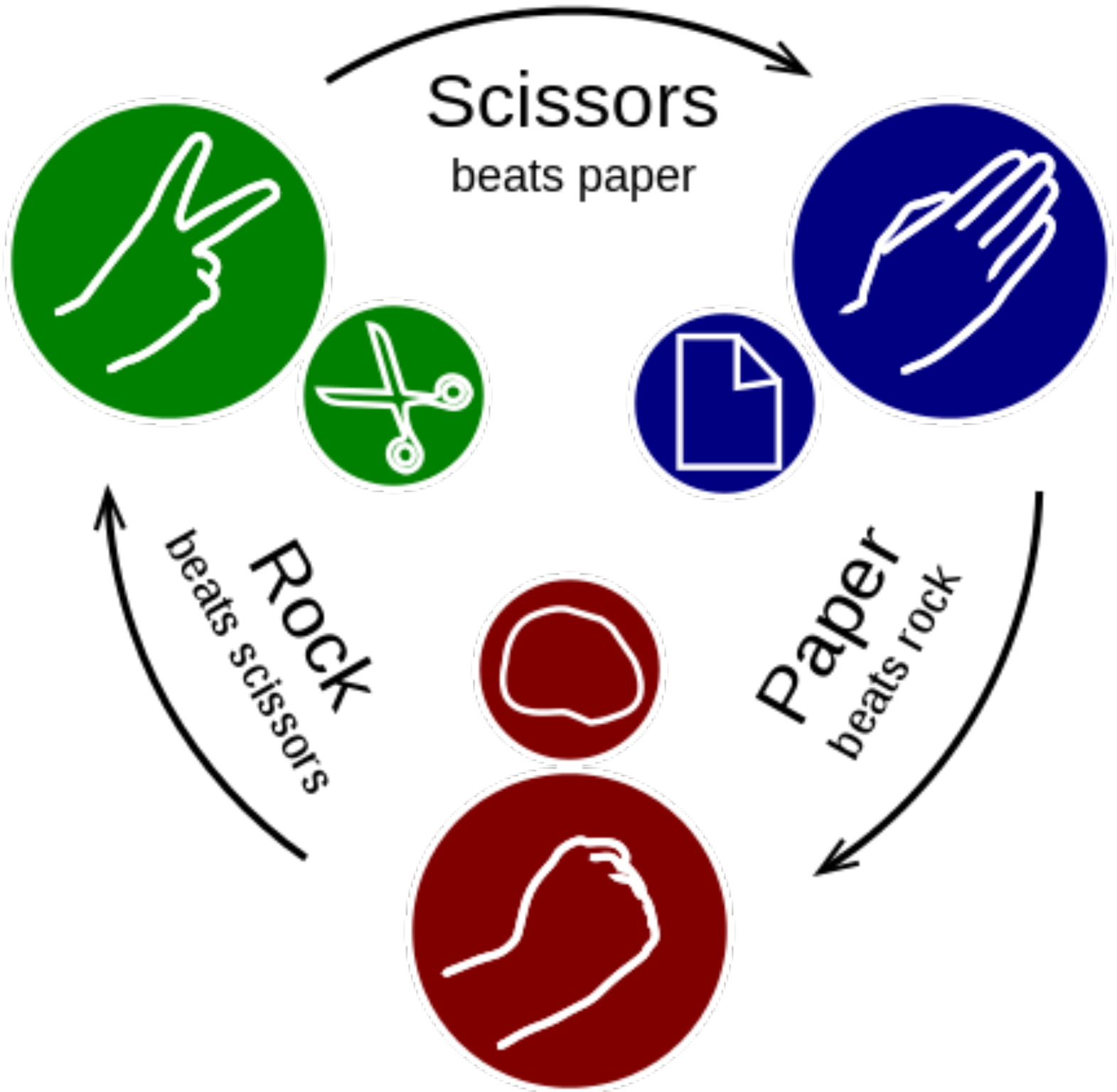
- Handy for engineers (rapid prototyping)
- Easy for beginners (steep learning curve)
- But strong for big apps: big data, AI …
  (https://www.tensorflow.org , https://www.scipy.org , http://scikit-learn.org/stable /, http://playground.arduino.cc/Interfacing/Python , …)
- Often used to command other programs (https://www.blender.org/manual/editors/python_console.html)
- Available for many platforms/operating systems (large community)

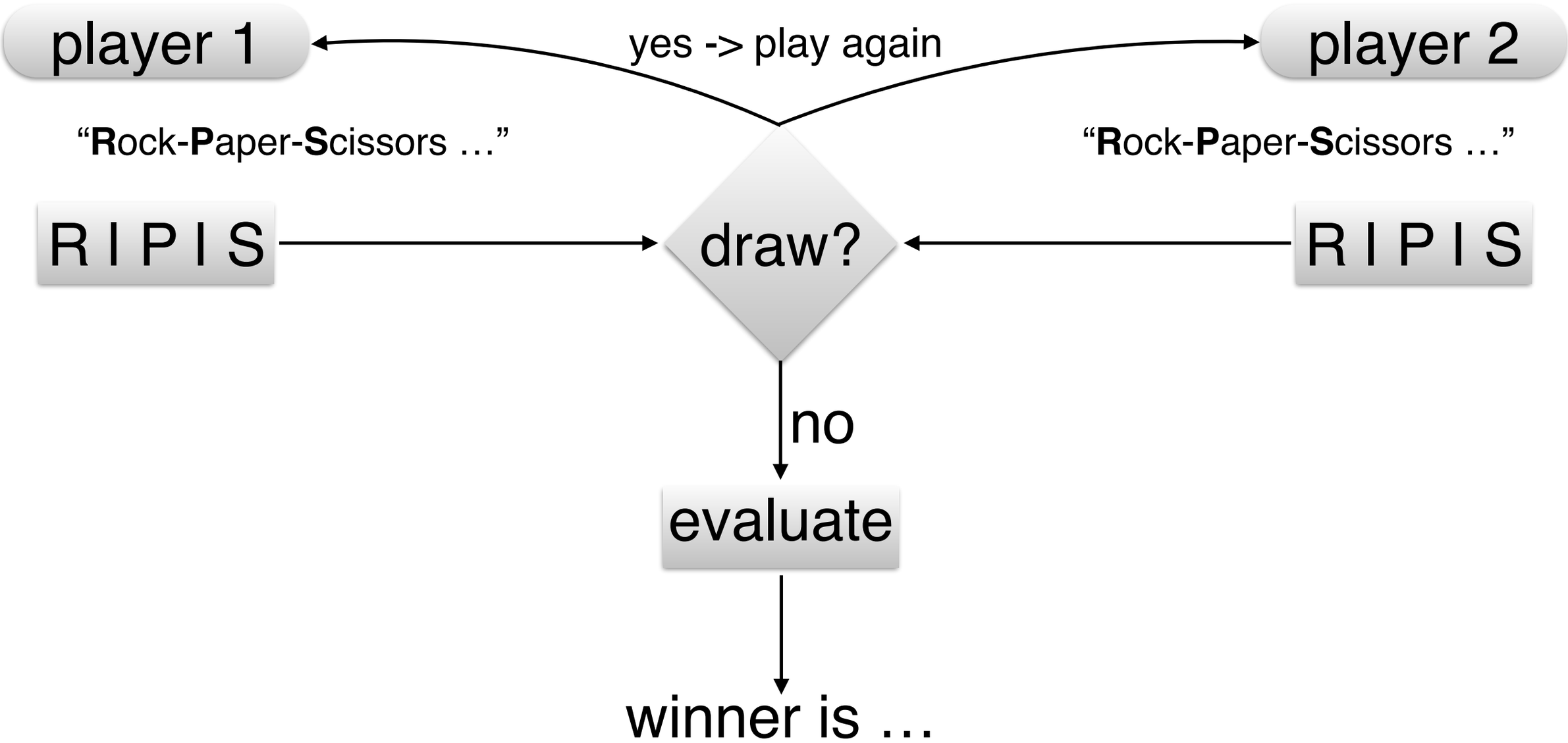
Why Python?

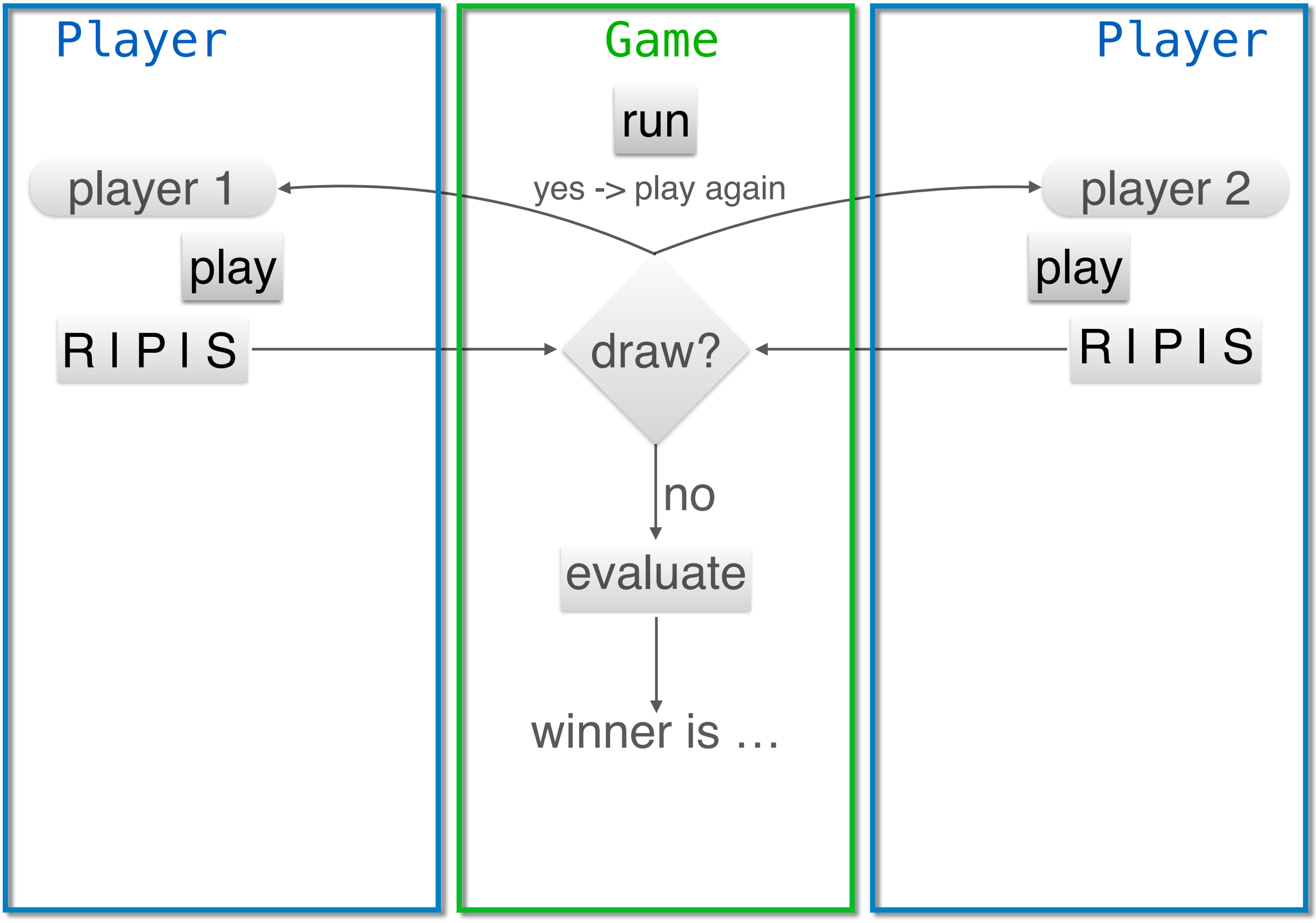

Growth of major programming languages
Based on Stack Overflow question views in World Bank high-income countries
% of overall question views each month
9%
6%
3%
0%
2012
2014
2016
2018
Time
python
javascript
java
c#
php
c++


Projections of future traffic for major programming languages
Future traffic is predicted with an STL model, along with an 80% prediction interval.
% of overall question views each month
15%
10%
5%
0%
2012
2014
2016
2018
2020
Time
python
javascript
java
c#
php
c++

Rock-Paper-Scissors


Scissors
beats paper
Rock
beats scissors
Paper
beats rock


player 1
yes -> play again
player 2
“Rock-Paper-Scissors …”
“Rock-Paper-Scissors …”
R I P I S
draw?
R I P I S
no
evaluate
winner is …


Player
Game
Player
run
player 1
yes -> play again
player 2
play
play
R I P I S
draw?
R I P I S
no
evaluate
winner is …

playerdummy.py

General definition: class of players

docstring - description

```
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
```

always return ‘R’

if run as the main program

create an instance (object) of the MyPlayer class

the player plays

What if we need to change the player during runtime?

playerdummyplus.py (dummy with attribute)

```
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
```

constructor of an instance

setting object attribute

a player reads the answer from its attribute

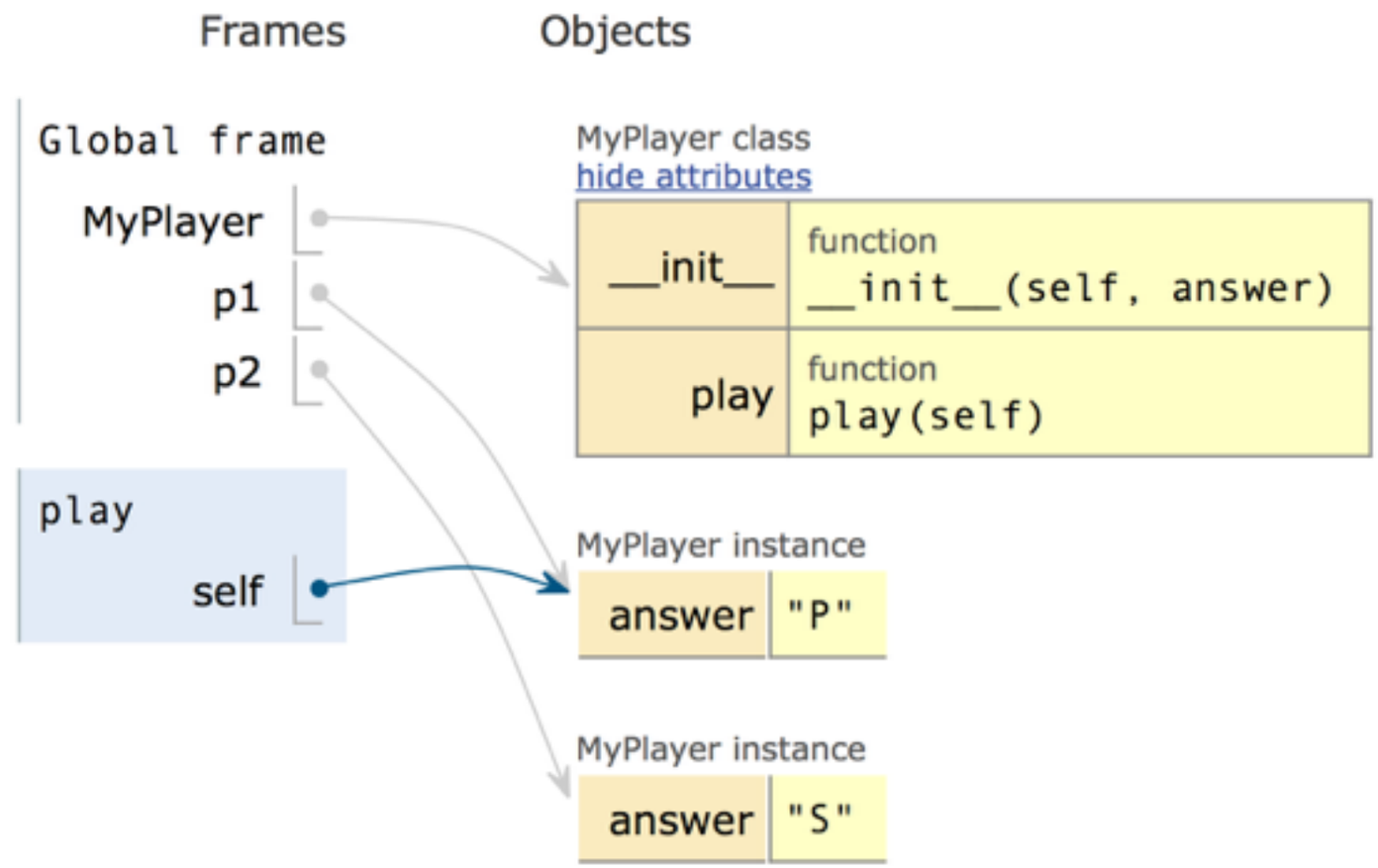
change the memory at runtime

now, it plays differently

visualisation

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()
```



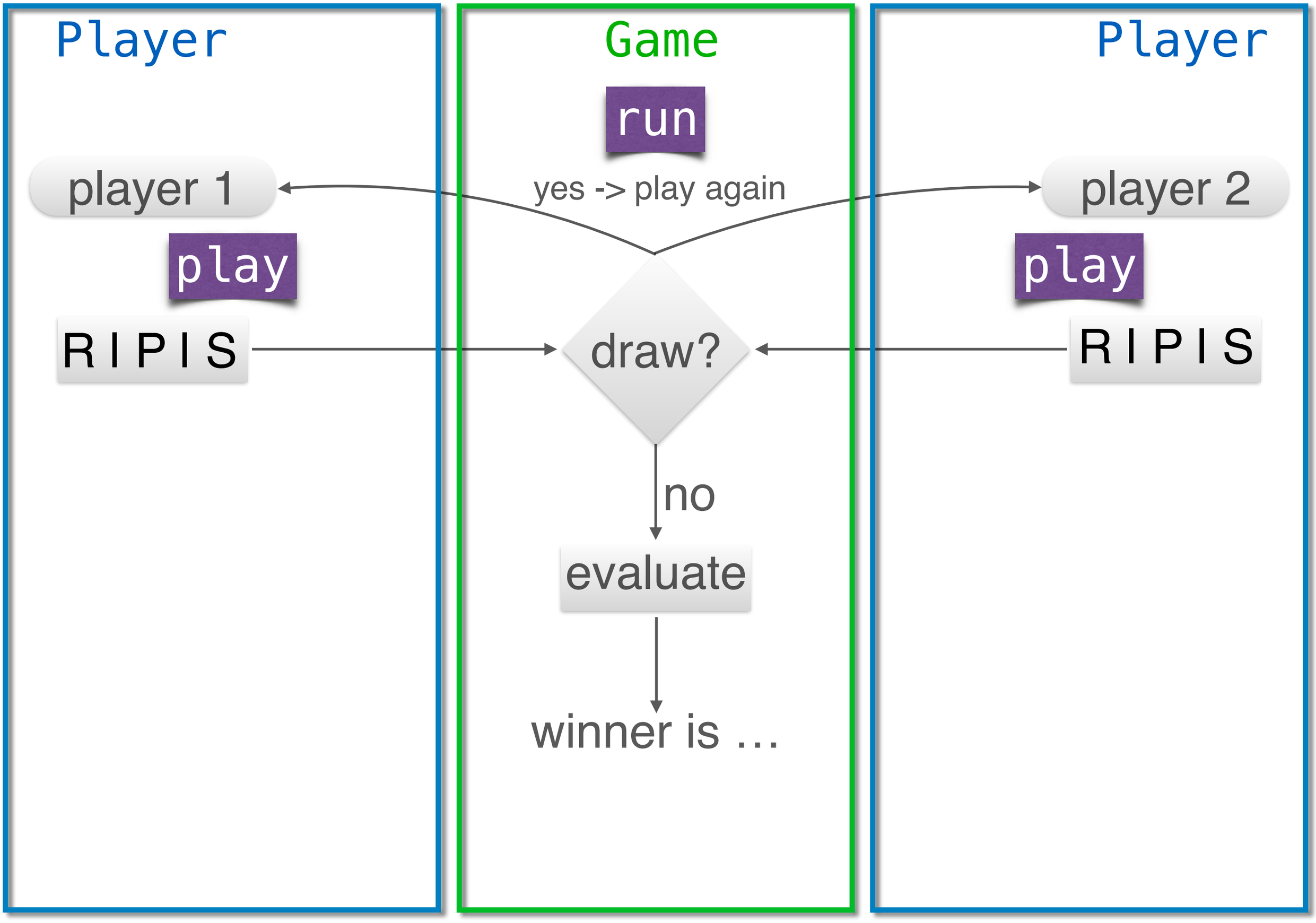
http://pythontutor.com/

playerdummyplusplus.py (dummy with memory)

```
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
```

initialize the memory - empty list

add the move to end of the list
append is a list *method*


Player
Game
Player
run
player 1
yes -> play again
player 2
play
play
R I P I S
draw?
R I P I S
no
evaluate
winner is …

How to play a game

```
p1 = Player
p2 = Player

draw = True

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

result = evaluate(move1,move2)
```

class Game

```
class Game:
    def __init__(self,p1,p2):
        self.p1 = p1
        self.p2 = p2
        self.winner = None

    def run(self):
        draw = True
        while draw:
            move1 = self.p1.play()
            move2 = self.p2.play()
            draw = (move1 == move2)
        result = evaluate_moves([move1,move2])
        if result[0]>result[1]:
            self.winner = self.p1
        else:
            self.winner = self.p2
```

players of the game

obviously, no winner at start

before they start, draw

while draw, keep playing

set the winner

evaluate moves

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

Paper > Rock, Scissors > Paper,
Rock > Scissors

main/control program

```
import playertom
import playerdummy

if __name__ == "__main__":
    p1 = playertom.MyPlayer()
    p2 = playerdummy.MyPlayer()
    g = Game(p1,p2)
    g.run()
    print('Winner is:',g.winner.__doc__)
```

import modules

some lines skipped here

create the player

instantiate the game, put players in

run the game

tell us who is the winner

For subsequent analysis, return index of the winner

```
class Game:
    def __init__(self,p1,p2):
        self.p1 = p1
        self.p2 = p2
        self.winner = None

    def run(self):
        draw = True
        while draw:
            move1 = self.p1.play()
            move2 = self.p2.play()
            draw = (move1 == move2)
        result = evaluate_moves([move1,move2])
        if result[0]>result[1]:
            self.winner = p1
            return 0
        else:
            self.winner = p2
            return 1
```

return index of the winner

repeated game

```
def compute_stats(winners):
    wins = [0,0]
    for winner in winners:
        wins[winner] = wins[winner]+1
    return wins

if __name__ == "__main__":
    p1 = playertom.MyPlayer()
    p2 = playerdummy.MyPlayer()
    winners = []
    for i in range(10):
        g = Game(p1,p2)
        winners.append(g.run())
        print('Winner is:',g.winner.__doc__)
    wins = compute_stats(winners)
    print(p1.__doc__, 'won %d times'%wins[0])
    print(p2.__doc__, 'won %d times'%wins[1])
```

number of victories

for each of the element in the list

init empy list

play 10x

add the result to the list

analyze the list, compute stats

iterative game

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