Ruby

Mateusz Lenik

VS

Python

Bartosz Woronicz

Ruby

Ruby

- Ruby != Ruby on Rails

Ruby

- Ruby != Ruby on Rails
- obiektowy

Ruby

- Ruby != Ruby on Rails
- obiektowy
- dynamicznie typowany

Ruby

- Ruby != Ruby on Rails
- obiektowy
- dynamicznie typowany
- funkcyjny

KISS

Keep It Simple, Stupid!

wszystko jest obiektem

wszystko jest obiektem

```
"".is_a?    Object # => true
```

wszystko jest obiektem

```
"".is_a?    Object # => true
1.is_a?     Object # => true
```

wszystko jest obiektem

```
"".is_a?    Object # => true
1.is_a?     Object # => true
nil.is_a?   Object # => true
```

wszystko jest obiektem

```
"".is_a?    Object # => true
1.is_a?     Object # => true
nil.is_a?   Object # => true
true.is_a? Object # => true
```

wszystko jest obiektem

wszystko jest obiektem

```
Object.is_a? Object # => true
```

wszystko jest obiektem

```
Object.is_a? Object # => true
Module.is_a? Object # => true
```

wszystko jest obiektem

```
Object.is_a? Object # => true
Module.is_a? Object # => true
Class.is_a?  Object # => true
```

wszystko jest obiektem

```
Object.is_a? Object # => true
Module.is_a? Object # => true
Class.is_a?  Object # => true
Object.is_a? Class  # => true
```

wszystko zwraca wartość

wszystko zwraca wartość

```
class Example; end # => nil
```

wszystko zwraca wartość

```
class Example; end # => nil

if true
```

wszystko zwraca wartość

```
class Example; end # => nil

if true
  "Chunky bacon!"
```

wszystko zwraca wartość

```
class Example; end # => nil

if true
  "Chunky bacon!"
end # => "Chunky bacon!"
```

wszystko zwraca wartość

wszystko zwraca wartość

```
color = :red unless is_cold?
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
x = case color
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
x = case color
    when :red then "awesome"
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
x = case color
    when :red then "awesome"
    when :black then "aweful"
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
x = case color
    when :red then "awesome"
    when :black then "aweful"
    else "undefined"
```

wszystko zwraca wartość

```
color = :red unless is_cold?
# ...
x = case color
    when :red then "awesome"
    when :black then "aweful"
    else "undefined"
    end # => "awesome"
```

mixins

mixins

```
class Example
```

mixins

```
class Example
  include Enumerable
```

mixins

```
class Example
  include Enumerable

  def each
```

mixins

```
class Example
  include Enumerable

  def each
    # ...
```

mixins

```
class Example
  include Enumerable

  def each
    # ...
    yield val
```

mixins

```
class Example
  include Enumerable

  def each
    # ...
    yield val
  end
```

mixins

```
class Example
  include Enumerable

  def each
    # ...
    yield val
  end
end
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"

[1, :chunky, "bacon"].join ", "
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"


[1, :chunky, "bacon"].join ", "
  # => "1, chunky, bacon"
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"


[1, :chunky, "bacon"].join ", "
  # => "1, chunky, bacon"
```

```ruby
[1,:chunky,"bacon"].map do lol
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"


[1, :chunky, "bacon"].join ", "
  # => "1, chunky, bacon"


[1,:chunky,"bacon"].map do |o|
  o.to_s.length
```

```ruby
"Ruby" * 3 # => "RubyRubyRuby"


[1, :chunky, "bacon"].join ", "
  # => "1, chunky, bacon"


[1,:chunky,"bacon"].map do |o|
  o.to_s.length
end # => [1, 6, 5]
```

Python

Python

- obiektowy

Python

- obiektowy
- indentation == readability

Python

- obiektowy
- indentation == readability
- silnie dynamicznie typowany

Python

- obiektowy
- indentation == readability
- silnie dynamicznie typowany
- dodatki j.funkcyjnych

Python

Python

```
class Duck(): pass
```

Python

```
class Duck(): pass
duck=Duck()
```

Python

```
class Duck(): pass
duck=Duck()

isinstance(Duck,  object) # => True
```

Python

```
class Duck(): pass
duck=Duck()

isinstance(Duck,   object) # => True
isinstance(duck,   object) # => True
```

Python

```
class Duck(): pass
duck=Duck()

isinstance(Duck,  object) # => True
isinstance(duck,  object) # => True
isinstance(False, object) # => True
```

Python

```
class Duck(): pass
duck=Duck()

isinstance(Duck,  object) # => True
isinstance(duck,  object) # => True
isinstance(False, object) # => True
isinstance(None,  object) # => True
```

```ruby
3 * 'kwa' # => 'kwakwakwa'
```

```ruby
3 * 'kwa' # => 'kwakwakwa'

'kwa' * 3 # => 'kwakwakwa'
```

```
3 * 'kwa' # => 'kwakwakwa'
'kwa' * 3 # => 'kwakwakwa'
map(lambda x:len(str(x)),
```

```
3 * 'kwa' # => 'kwakwakwa'
'kwa' * 3 # => 'kwakwakwa'
map(lambda x:len(str(x)),
[1, 'chunky', 'bacon'] )
```

```
3 * 'kwa' # => 'kwakwakwa'
'kwa' * 3 # => 'kwakwakwa'
map(lambda x:len(str(x)),
[1, 'chunky', 'bacon'] )
# => [1, 6, 5]
```

```
3 * 'kwa' # => 'kwakwakwa'
'kwa' * 3 # => 'kwakwakwa'
map(lambda x:len(str(x)),
[1, 'chunky', 'bacon'] )
# => [1, 6, 5]
', '.join(map(str,[1, 'chunky',
'bacon'])) # => '1, chunky, bacon'
```

Namespaces are great honkin' idea!

Namespaces are great honkin' idea!

– It's a weapon of great power, O'Neill
– Big and honkin'!

```
import re
```

```
import re

from re import *
```

```
import re

from re import *

from re import match
```

```
import re

from re import *

from re import match

import Integer
```

```
import re

from re import *

from re import match

import Integer
import Floating
```

```
import re

from re import *

from re import match

import Integer
import Floating

Integer.add()
```

```
import re

from re import *

from re import match

import Integer
import Floating

Integer.add()
Floating.add()
```

```
import re

from re import *

from re import match

import Integer
import Floating

Integer.add()
Floating.add()

if __name__=='__main__':
```

```
import re

from re import *

from re import match

import Integer
import Floating

Integer.add()
Floating.add()

if __name__=='__main__':
   pass
```

Słowniki

Słowniki

```
type( {} ) # => dict / builtin.dict
```

Słowniki

```
type( {} ) # => dict / builtin.dict

{
```

Słowniki

```
type( {} ) # => dict / builtin.dict

{
    'string1':'string2', 3:4,
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
    'lets_go_deeper': {
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
    'lets_go_deeper': {
      9:10
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
'lets_go_deeper': {
  9:10
  }
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
    'lets_go_deeper': {
      9:10
      }
}
```

Słowniki

```
type( {} ) # => dict / builtin.dict


{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
    'lets_go_deeper': {
      9:10
      }
    }
'string'.__hash__()
```

Słowniki

```
type( {} ) # => dict / builtin.dict

{
    'string1':'string2', 3:4,
    ('to','jest tupla'): 7.0,
    'lets_go_deeper': {
      9:10
      }
    }
'string'.__hash__()
hash('string')
```

Wcięcia

Wcięcia

```
class Jestem():
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
ja = Jestem()
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
ja = Jestem()
ja.TrocheWciety('piwo','paluszki')
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
ja = Jestem()
ja.TrocheWciety('piwo','paluszki')
# => Oto moje argumenty: (<__main__.Jestem
instance at 0x1fde368>, 'piwo', 'paluszki')
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
ja = Jestem()
ja.TrocheWciety('piwo','paluszki')
# => Oto moje argumenty: (<__main__.Jestem
instance at 0x1fde368>, 'piwo', 'paluszki')
ja.Samolubny()
```

Wcięcia

```
class Jestem():
  def TrocheWciety(*args):
      print 'Oto moje argumenty: {0}'.format(args)
   def Samolubny(self):
      print 'Oto ja: {0}'.format(self)
ja = Jestem()
ja.TrocheWciety('piwo','paluszki')
# => Oto moje argumenty: (<__main__.Jestem
instance at 0x1fde368>, 'piwo', 'paluszki')
ja.Samolubny()
# => Oto ja: <__main__.Jestem instance at
0x1fde368>
```

Dekoratory

Dekoratory

```
from django.contrib.auth.decorators import
login_required
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
      profile = get_user_profile(request.user)
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
      profile = get_user_profile(request.user)
      if profile.is_statsviewer != True:
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
      profile = get_user_profile(request.user)
      if profile.is_statsviewer != True:
        return
HttpResponseForbidden("FORBIDDEN")
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
      profile = get_user_profile(request.user)
      if profile.is_statsviewer != True:
        return
HttpResponseForbidden("FORBIDDEN")
      return function(request, *args, **kwargs)
```

Dekoratory

```
from django.contrib.auth.decorators import
login_required
def is_statsviewer(function):
@login_required
  def _inner(request, *args, **kwargs):
    if not request.user.is_superuser:
      profile = get_user_profile(request.user)
      if profile.is_statsviewer != True:
        return
HttpResponseForbidden("FORBIDDEN")
      return function(request, *args, **kwargs)
    return _inner
```

Generatory

Generatory

```
def firstn(n):
```

Generatory

```
def firstn(n):
    num = 0
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1

it = firstn(5)s
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1

it = firstn(5)s
# => <generator object firstn at 0x...>
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1


it = firstn(5)s
# => <generator object firstn at 0x...>
it.next() # => 0
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1


it = firstn(5)s
# => <generator object firstn at 0x...>
it.next() # => 0
it.next() # => 1
```

Generatory

```
def firstn(n):
    num = 0
    while num < n:
        yield num
        num += 1


it = firstn(5)s
# => <generator object firstn at 0x...>
it.next() # => 0
it.next() # => 1
sum(firstn(1000)) # => 499500
```

Wydajność

benchmark

benchmark

- mini-webaplikacja

benchmark

- mini-webaplikacja
- użycie Redis

benchmark

- mini-webaplikacja
- użycie Redis
- prezentacja w JSON

benchmark

- mini-webaplikacja
- użycie Redis
- prezentacja w JSON
- CherryPy vs Tornado vs Sinatra

```
%w(sinatra redis thin
json).each &method(:require)
```

```
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
```

```ruby
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
get '/' do
```

```
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
get '/' do
  content_type :json
```

```
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
get '/' do
  content_type :json

  key = r.randomkey
```

```
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
get '/' do
  content_type :json

  key = r.randomkey
  { key => r.get(key) }.to_json
```

```ruby
%w(sinatra redis thin
json).each &method(:require)

r = Redis.new
get '/' do
  content_type :json

  key = r.randomkey
  { key => r.get(key) }.to_json
end
```

```
import tornado.ioloop
```

```
import tornado.ioloop
import tornado.web
```

```
import tornado.ioloop
import tornado.web
from random import *
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
    (r"/", MainHandler),
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
    (r"/", MainHandler),
])
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
    application.listen(8888)
```

```
import tornado.ioloop
import tornado.web
from random import *
from json import *
from redis import *

r = Redis()
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.set_header("Content-Type", "application/json")
        k = r.randomkey()
        self.write(dumps({ k : r.get(k) }))

application = tornado.web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

wyniki

wyniki

- Ruby jest nie jest wiele gorszy

wyniki

- Ruby jest nie jest wiele gorszy
- Operacje na listach są wolne

wyniki

- Ruby jest nie jest wiele gorszy
- Operacje na listach są wolne
- CherryPy < Sinatra <= Tornado

Rozszerzenia

rake nokogiri rspec

rails *treetop* **sinatra**

tilt sass spork haml

thor **compass** thin

factory_girl foreman

mustache ffi prawn

pry rdiscount devise

capistrano resque

warden capybara

bundler **padrino**

nanoc oniguruma

IPython **PyPy**

PIL SciPy CTypes **Psyco**

PyGTK+ TwistedMatrix

BeautifulSoup

Django CherryPy **Flask**

Tornado

Społeczność

RAILSCONF2012

AUSTIN, TEXAS

4TH ANNUAL JRUBY CONFERENCE

RUBYCONF URUGUAY | 2011

A FRESH RUBY CONFERENCE

I'M ATTENDING

WROC_LOVE.RB

10-11 MARCH 2012 WROCLAW POLAND

MountainWest RubyConf 2012

EURUKO2012

AMSTERDAM - 1st & 2nd JUNE

RUBYCONF 2011

SEPTEMBER 29 - OCTOBER 1, NEW ORLEANS, LA

Barcelona RUBY CONFERENCE

RAILSCONF2012

AUSTIN, TEXAS

4TH ANNUAL

JRUBY

CONFERENCE

RUBYCONF

URUGUAY | 2011

A FRESH RUBY CONFERENCE

I'M ATTENDING

WROC_LOVE.RB

10-11 MARCH 2012 WROCLAW POLAND

MountainWest

RubyConf 2012

EURUKO2012

AMSTERDAM - 1st & 2nd JUNE

RUBYCONF 2011

SEPTEMBER 29 - OCTOBER 1, NEW ORLEANS, LA

Barcelona

RUBY CONFERENCE

Pythonowcy wszystkich krajów ŁĄCZCIE SIĘ!

@PyCon

@EuroPython

Problemy

brak metod prywatnych

brak metod prywatnych

```
class Example
```

brak metod prywatnych

```
class Example
  private
```

brak metod prywatnych

```
class Example
  private
  def example_method
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
  end
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
  end
end
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
  end
end

e = Example.new
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
  end
end

e = Example.new
e.example_method # => NoMethodError
```

brak metod prywatnych

```
class Example
  private
  def example_method
    "hello"
  end
end

e = Example.new
e.example_method # => NoMethodError
e.send :example_method # => "hello"
```

dziedziczone zmienne klas

dziedziczone zmienne klas

```
class Parent
```

dziedziczone zmienne klas

```
class Parent
  def test
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
  end
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
  end
end
```

dziedziczone zmienne klas

```
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
  end
end

Parent.new.test # => "hello"
```

dziedziczone zmienne klas

```ruby
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
  end
end

Parent.new.test # => "hello"
Child.new.test  # => "world"
```

dziedziczone zmienne klas

```ruby
class Parent
  def test
    @@test ||= "hello"
  end
end
class Child < Parent
  def initialize
    @@test = "world"
  end
end

Parent.new.test # => "hello"
Child.new.test  # => "world"
Parent.new.test # => "world"
```

brak standardu

brak standardu

- nowości tylko w MRI

brak standardu

- nowości tylko w MRI
- RubySpec na podstawie MRI

brak standardu

- nowości tylko w MRI
- RubySpec na podstawie MRI
- inne implementacje na podstawie RubySpec

brak API dla rozszerzeń

brak API dla rozszerzeń

- wywoływanie metod interpretera

brak API dla rozszerzeń

- wywoływanie metod interpretera
- problem z innymi implementacjami

brak API dla rozszerzeń

- wywoływanie metod interpretera
- problem z innymi implementacjami
- powstawanie gemów *-java

brak API dla rozszerzeń

- wywoływanie metod interpretera
- problem z innymi implementacjami
- powstawanie gemów *-java
- nie wszystkie gemy działają

Jawny `self`

Jawny `self`

```
class MifareClassicSector(object):
```

Jawny `self`

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
      [ 0 for x in range(self.blocksize *
(self.blocks-1)) ]
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
      [ 0 for x in range(self.blocksize *
(self.blocks-1)) ]
    else:
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
      [ 0 for x in range(self.blocksize *
(self.blocks-1)) ]
    else:
      self.__dict__['data'] = data + \
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
      [ 0 for x in range(self.blocksize *
(self.blocks-1)) ]
    else:
      self.__dict__['data'] = data + \
      [ 0 for x in range(self.blocksize * self.blocks -
len(data)) ]
```

Jawny self

```
class MifareClassicSector(object):
  def
__init__(self,blocksize=0x10,blocks=0x04,data=None):
  # values here can be defined by constructor later
    self.__dict__['blocksize'] = blocksize
    self.__dict__['blocks'] = blocks
    if not data:
      self.__dict__['data'] = \
      [ 0 for x in range(self.blocksize *
(self.blocks-1)) ]
    else:
      self.__dict__['data'] = data + \
      [ 0 for x in range(self.blocksize * self.blocks -
len(data)) ]
      self.initsectorlayout()
```

super

super

```
def __setattr__(self, name, value):
```

super

```
def __setattr__(self, name, value):
   if name == 'data':
```

super

```
def __setattr__(self, name, value):
    if name == 'data':
        super().__setattr__(self, name, value)
```

super

```
def __setattr__(self, name, value):
    if name == 'data':
        super().__setattr__(self, name, value)
    elif name == 'trailer':
```

super

```
def __setattr__(self, name, value):
    if name == 'data':
        super().__setattr__(self, name, value)
    elif name == 'trailer':
    # ...
```

import

import

```
import re
```

import

```
import re
import sys
```

import

```
import re
import sys
import os
```

import

```
import re
import sys
import os
import re, sys, os
```

import

```
import re
import sys
import os
import re, sys, os

from re import *
```

import

```
import re
import sys
import os
import re, sys, os

from re import *
from sys import *
```

import

```
import re
import sys
import os
import re, sys, os

from re import *
from sys import *
from os import *
```

import

```
import re
import sys
import os
import re, sys, os

from re import *
from sys import *
from os import *

from re, sys, os import *
```

brak przypisania w pętli

brak przypisania w pętli

```
while x+=1 < max:
```

brak przypisania w pętli

```
while x+=1 < max:
   pass
```

brak przypisania w pętli

```
while x+=1 < max:
    pass
```

```
while x < max:
```

brak przypisania w pętli

```
while x+=1 < max:
    pass
```

```
while x < max:
    x+=1
```

brak przypisania w pętli

```
while x+=1 < max:
    pass
```

```
while x < max:
    x+=1
    pass
```

lambda

lambda

```
>>> def make_incrementor (n):
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>>
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>>
>>> print( f(42), g(42) )
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>>
>>> print( f(42), g(42) )
44 48
```

lambda

```
>>> def make_incrementor (n):
        return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>>
>>> print( f(42), g(42) )
44 48
>>> print( make_incrementor(22)
```

Przyszłość

Ruby 2.0

Ruby 2.0

- Nowy GC

Ruby 2.0

- Nowy GC
- Enumerator::Lazy

Ruby 2.0

- Nowy GC
- Enumerator::Lazy
- Prekompilacja skryptów

Ruby 2.0

- Nowy GC
- `Enumerator::Lazy`
- Prekompilacja skryptów
- `refine` i `using`

Python

Python

- `from __future__ import *`

Python

- `from __future__ import *`

- Python3

Python

- `from __future__ import *`
- Python3
- PEP

Dziękujemy za uwagę

Pytania?