Programowanie w Ruby

Wykład 1

Marcin Młotkowski

7 października 2016

Plan wykładu

Informacje organizacyjne

Strona wykładu

`http://www.ii.uni.wroc.pl/~marcinm/dyd/ruby/`

Plan wykładu

Cele wykładu

- Poznanie języka Ruby, jego składni i semantyki
- poznanie środowiska Ruby on Rails

Plan wykładu

1. Język Ruby: typy podstawowe, składania, semantyka
2. Typy wbudowane
3. refleksje (introspekcje)
4. wątki
5. środowisko graficzne
6. Ruby on Rails

Źródła internetowe

Oficjalna strona

http://www.ruby-lang.org/pl/

Źródła internetowe

Oficjalna strona

http://www.ruby-lang.org/pl/

Dokumentacja

http://www.ruby-doc.org/

Inne materiały internetowe

Patrz: strona wykładu

Książki do Ruby

D. Thomas, C. Fowler, A. Hunt. Programowanie w języku Ruby, Helion 2006

Książki do Ruby

D. Thomas, C. Fowler, A. Hunt. Programowanie w języku Ruby, Helion 2006

- tzw. *The PickAxe* (od obrazka na okładce);
- dostępna w internecie;
- podobno najlepiej sprzedająca się książka z informatyki w Amazonie w 2005 i 2006 roku.

Książki do Ruby

 L. Carlson, L. Richardson. Ruby. Receptury, Helion 2007

 H. Fulton. Ruby. Tao programowania w 400 przykładach, Helion 2008

 D. Flanagan, Y. Matsumoto. Ruby. Programowanie, Helion 2009

 M.Fitzgerald. Ruby. Wprowadzenie, Helion 2007

Książki do Ruby on Rails

S. Holzner. Ruby on Rails, Helion 2008

B. Tate, L. Carlson, C. Hibbs. Ruby on Rails. Wprowadzenie, Helion 2009

Zasady prowadzenia pracowni

Pierwsza część semestru (ok. 10 tyg)

Po wykładzie będą ogłaszane listy krótkich zadań do zrobienia na najbliższą pracownię.

Zasady prowadzenia pracowni

Pierwsza część semestru (ok. 10 tyg)

Po wykładzie będą ogłaszane listy krótkich zadań do zrobienia na najbliższą pracownię.

Druga część semestru

Do wykonania będzie większy projekt, preferowane środowisko: Ruby on Rails

Zasady prowadzenia pracowni

Pierwsza część semestru (ok. 10 tyg)

Po wykładzie będą ogłaszane listy krótkich zadań do zrobienia na najbliższą pracownię.

Druga część semestru

Do wykonania będzie większy projekt, preferowane środowisko: Ruby on Rails

Zaliczenie semestru

Zdobycie przynajmniej połowy możliwych do zdobycia punktów.

Zasady prowadzenia pracowni

Pierwsza część semestru (ok. 10 tyg)

Po wykładzie będą ogłaszane listy krótkich zadań do zrobienia na najbliższą pracownię.

Druga część semestru

Do wykonania będzie większy projekt, preferowane środowisko: Ruby on Rails

Zaliczenie semestru

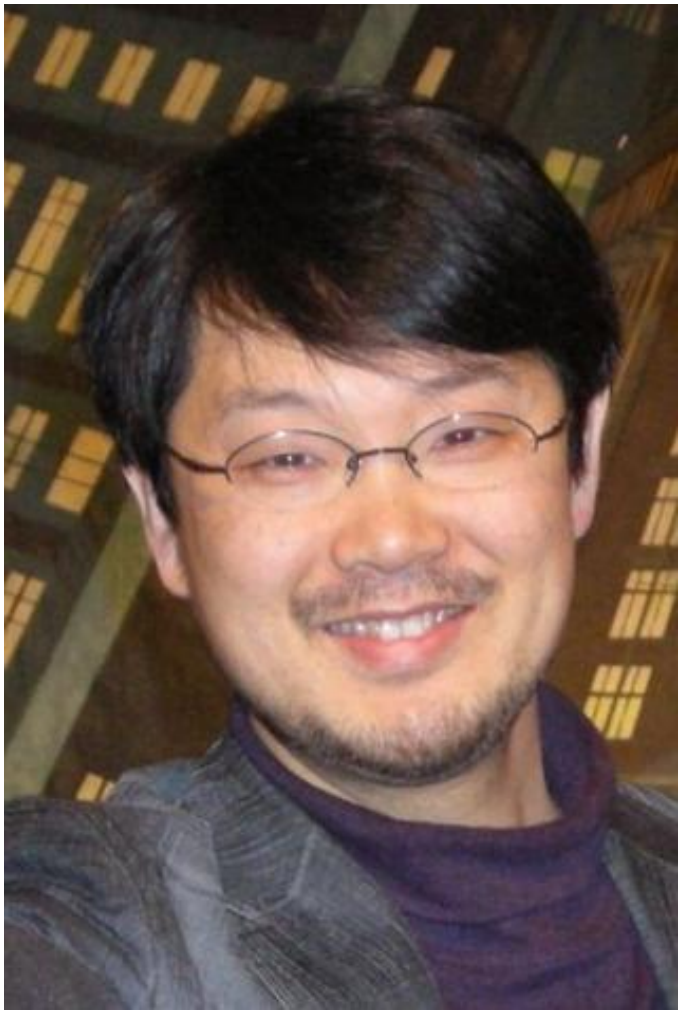
Zdobycie przynajmniej połowy możliwych do zdobycia punktów.

Egzamin

To jest kurs, więc egzaminu nie ma ;-)

Plan wykładu

Yukihiro Matsumoto a.k.a Matz



Nazwa języka

Stan obecny języka

Aktualna wersja

2.2.0

(stan na 6 października 2016)

Stan obecny języka

Aktualna wersja

2.2.0

(stan na 6 października 2016)

Rozwój języka

Społeczność

Pochodzenie języka

- trochę z Perla ($, $_);
- trochę ze Smalltalka;
- trochę z Pythona;
- trochę z CLOS'a;
- trochę z Lispu.

Co jest fajnego w Rubym

- programowanie strukturalne
- programowanie obiektowe
- programowanie funkcjonalne

Ogólnie o języku

- język interpretowany;
- typowanie dynamiczne (duck typing);
- brak deklaracji typów;
- sporo typów wbudowanych.

Co jest fajnego w Ruby?

Zoptymalizowany dla przyjemności programisty.

Co jest fajnego w Ruby?

Zoptymalizowany dla przyjemności programisty.

Skąd taka popularność?

Ruby on Rails!!!

Zastosowania

Gdzie się używa Ruby'ego

- Open Domain Server
- telefonia 3G (Lucent);
- robotyka (Siemens);
- systemy webowe (Basecamp, Blue Sequence dla Toyota Motors);
- i wiele innych.

Praca interaktywna

```
$ irb
irb(main):001:0> puts("A kuku")
A kuku
=> nil
irb(main):002:0> 2+2
=> 4
irb(main):003:0>
```

Wyrażenia w Ruby'm

Arytmetyka

- literały: 1024, 3.1415
- wyrażenia: (2 + 2) * 4

Wyrażenia w Ruby'm

Arytmetyka

- literały: 1024, 3.1415
- wyrażenia: (2 + 2) * 4

Napisy

'napis', "napis"

Składnia

Separacja instrukcji

- instrukcje są w kolejnych wierszach
- instrukcje można oddzielać średnikiem

Zmienne

Nazwy zmiennych

Na razie piszemy małą literą. Nazwy mogą się też zaczynać podkreśleniem.

Instrukcja warunkowa if

```
slowo = "dlugie"
if "krotkie".length < slowo.length
    puts 'Krótkie'
else; puts 'Długie' end
```

Instrukcja warunkowa case

```
zwierze = ''hau
case zwierze
    when 'hau', 'woof'
        puts 'pies'
    when 'miau'
        puts 'kot'
    else puts '???'
end
```

Jeszcze inna instrukcja warunkowa

```
puts "Zimno" if temperatura <= -20
puts "Pogoda" unless temp < 15 && temp > 20
```

Wyrażenia logiczne

- Fałsz: nil oraz false
- prawda: wszystko inne co nie jest fałszem;
- Operatory: and, &&, or, ||, not, !
- ==, defined?

Wyrażenia logiczne

- Fałsz: nil oraz false
- prawda: wszystko inne co nie jest fałszem;
- Operatory: and, &&, or, ||, not, !
- ==, defined?

Wartość wyrażenia logicznego

Wartością wyrażenia logicznego, gdy jest prawdziwe, jest ostatni operand.

Wyrażenia logiczne

- Fałsz: nil oraz false
- prawda: wszystko inne co nie jest fałszem;
- Operatory: and, &&, or, ||, not, !
- ==, defined?

Wartość wyrażenia logicznego

Wartością wyrażenia logicznego, gdy jest prawdziwe, jest ostatni operand.

Przykłady

'nie' and "tak"	'tak'
false and 99	false

Operatory logiczne

Kolejność obliczania wyrażeń logicznych (1)

x1 or x2 and x3:
(x1 or x2) and x3

Kolejność obliczania wyrażeń logicznych (2)

x1 || x2 && x3:
x1 || (x2 && x3)

Pętla while

```
a, b = 0, 1
while b < 10
    puts b
    a, b = b, a + b
end
```

Pętla for

```
for x in 1..10
    puts x
end

for el in [1,2,3,4,5,6,7,8,9,10]
    sum = sum + el
end
```

I jeszcze inne pętle

```
5.times do
    puts "Hurra!\n"
end
```

```
i += 1 while  i < 100
```

Pętle, uzupełnienie

Instrukcja break

kończy pętlę

Pętle, uzupełnienie

Instrukcja break

kończy pętlę

Instrukcja next

przechodzi na koniec pętli

Pętle, uzupełnienie

Instrukcja break

kończy pętlę

Instrukcja next

przechodzi na koniec pętli

Instrukcja redo

powtarza pętlę od początku bez przeliczania warunku lub pobierana kolejnego elementu (w iteratorze)

Tablice

```
[ 1, "dwa", 3.0 ]
```

Przetwarzanie bloków

```
tablica = [ [1,2,3], [4,5]]
suma = 0
i = 0
while i < tablica.length
    subtablica = tablica[i]
    j = 0
    while j < subtablica.length
        suma += subtablica[j]
        j += 1
    end
    i += 1
end
```

Sumowanie liczb: przykład

```
def suma(n)
    s=i=0
    for i in 1..n
        s += i
    end
    return s
end
```

Sumowanie liczb: przykład

```
def suma(n)
    s=i=0
    for i in 1..n
        s += i
    end
    return s
end
```

```
def suma(n)
    s=i=0
    for i in 1..n
        s += i
    end
    s
end
```

Zasięg zmiennych

Zmienne mają zasięg tylko lokalny, np.

```
zmienna = "A kuku"
def wypisz
    puts zmienna
end
```

in ‘wypisz’: undefined local variable or method ‘zmienna’ for main:Object (NameError)

Dygresja I

Kompilator wsadowy

```
$ ruby plik.rb
```

Dygresja I

Kompilator wsadowy

```
$ ruby plik.rb
```

Plik "wykonywalny"

```
#!/usr/bin/ruby
def silnia(n)
    if n == 0; 1
    else
        n * silnia(n-1)
    end
end
```

Dygresja II

Środowisko/edytor

vim, gedit, geany, notepad, ...

Dygresja II

Środowisko/edytor

vim, gedit, geany, notepad, ...

RubyMine firmy JetBrains

Produkt bezpłatny do celów edukacyjnych.

Dygresja II

Środowisko/edytor

vim, gedit, geany, notepad, ...

RubyMine firmy JetBrains

Produkt bezpłatny do celów edukacyjnych.

TryRuby.org

Można uruchomić kod w przeglądarce

Dygresja III

Różnice między puts a print

`puts` zawsze dodaje na końcu '\n'

`print` nie dodaje na końcu '\n'.

Dygresja III

Różnice między puts a print

`puts` zawsze dodaje na końcu '\n'

`print` nie dodaje na końcu '\n'.

```
tablica = [1,2,3]
puts tablica
print tablica
```

```
1
2
3
123
```

Dygresja IV: wczytywanie danych z konsoli

gets

```
puts "Skąd jesteś"
miasto = gets
```

Ale gets wczytuje ze znakiem '\n', lepiej więc

```
puts "Skąd jesteś"
miasto = gets.chomp
```