



Szkolenie Rails

Kurs programowania Ruby on Rails

Ruby podstawy

Ruby



Ruby to język bardzo prosty.

Nie musimy deklarować typów zmiennych.

Nie musimy includować bibliotek, aby rozpocząć pisanie programów.

Ruby to język „bardzo” obiektowy.

IRB



Konsola **IRB** - jest to interaktywny Ruby.

Za pomocą **IRB** możemy testować nasze programy napisane w Ruby.

IRB uruchamiamy z konsoli. Polecenie quit zamyka irb.

```
Last login: Fri Mar 14 17:35:12 on ttys000
MacBook-Pro-Micha:~ michalos$ irb
irb(main):001:0> puts "Witaj świecie".
Witaj świecie
=> nil
irb(main):002:0> 
```

Inny sposób pisania programów



Tworzymy plik z rozszerzeniem .rb

Następnie poleceniem `ruby nazwa_pliku.rb` uruchamiamy nasz plik.

```
MacBook-Pro-Micha:~ michalos$ ruby hello.rb
"Witaj świecie"
Witaj świecie
```

Polecenie p , bądź puts



Polecenie p bądź puts wypisuje na ekran dane.

```
p "Witaj świecie"  
puts "Witaj świecie"
```

Zmienna



Zmienna jest synonimem obszaru pamięci, służącego do przechowywania danych.

W Ruby nie musimy deklarować typu zmiennej

```
a = 5
```

deklarujemy zmienną , która #przechowuje liczbę całkowitą

```
a ="Witam"
```

deklarujemy zmienną , która #przechowuje znaki

Operatory porównania



`==` operator porównania

`!=` operator różny

`>` operator większości

`<` operator mniejszości

`>=` operator większy, równy

`<=` operator mniejszy, równy

`<=>` (`a<=>b`) zwraca 0 , gdy liczby równe, 1 jeśli a większe od b, -1 , gdy b większe od a

`===` operator porównywania zakresów

Operatory przypisania



`=` zwyczajne przypisanie

`+=` jest równoważne z `a = a+5`

`*=` jest równoważne z `a = a*5`

`-=` jest równoważne z `a = a-5`

`/=` jest równoważne z `a = a/5`

`%=` jest równoważne z `a = a%5`

Operatory przypisania 2



`a,b = 5,8` #przypisuje do zmiennej `a` wartość `5` , `a` do zmiennej `b` wartość `8`

`a,b = b,a` # zamienia 2 wartości

Logiczne operatory



AND logiczne I

OR logiczne LUB

&& logiczne I

|| logiczne LUB

NOT logiczne zaprzeczenie

! logiczne zaprzeczenie

Podstawowe operacje na liczbach



`a = 5**2 # 25`

`b = 64**0.5 #8.0`

`C = 64**0 # 1`

`D = 64**-1 # 0.015625`

Zaokrąglanie liczb



-wynikiem dzielenia jednej liczby całkowitej przez inną liczbę całkowitą jest liczba całkowita

-przykładowo:

$3/3$ # 1

$3/5$ # 0

$3/4$ # 0

$3.0/4$ # 0.75

$3/4.0$ # 0.75

Przykład liczby

```
irb(main):001:0> a = 1
=> 1
irb(main):002:0> a.class
=> Fixnum
irb(main):003:0> b = 5.0
=> 5.0
irb(main):004:0> b.class
=> Float
irb(main):005:0> b *=5
=> 25.0
irb(main):006:0> b/=3
=> 8.3333333333333333334
irb(main):007:0> a/=3
=> 0
irb(main):008:0> a+=5
=> 5
irb(main):009:0> a.to_f
=> 5.0
irb(main):010:0> a
=> 5
irb(main):011:0> a.class
=> Fixnum
irb(main):012:0> a = a.to_f
=> 5.0
irb(main):013:0> a.class
=> Float
```

Instrukcja if

If jeśli warunek jest spełniony wykonywana jest instrukcja.

```
irb(main):001:0> a = 5
=> 5
irb(main):002:0> b = 8
=> 8
irb(main):003:0> if b <a
irb(main):004:1> puts "Witam"
irb(main):005:1> end
=> nil
irb(main):006:0> if b>a
irb(main):007:1> puts "Witam"
irb(main):008:1> end
Witam
=> nil
```

Instrukcja if -else



If jeśli warunek jest spełniony wykonywana jest instrukcja wewnątrz if w przeciwnym wypadku instrukcja wewnątrz else.

```
MacBook-Pro-Micha:~ michalos$ irb
irb(main):001:0> a = 5
=> 5
irb(main):002:0> b = 10
=> 10
irb(main):003:0> if a>b
irb(main):004:1> puts "A większe od B"
irb(main):005:1> else
irb(main):006:1* puts "B większe od A"
irb(main):007:1> end
B większe od A
=> nil
```

Pętla for



- w praktyce dość rzadko używana w Ruby
- w innych językach dużo częściej
- przydatna w zadaniach algorytmicznych

```
irb(main):008:0> suma = 0
=> 0
irb(main):009:0> for i in 1..10
irb(main):010:1> suma+=i
irb(main):011:1> end
=> 1..10
irb(main):012:0> puts suma
55
```


Pętla while



Wykonywana podczas, gdy warunek jest spełniony.

```
irb(main):013:0> a = 30
=> 30
irb(main):014:0> while (a>1) do
irb(main):015:1* a = a-1
irb(main):016:1> end
=> nil
```

Metoda



- Metodę w Ruby definiujemy za pomocą słowa kluczowego def.
- Metody służą do tego , aby raz napisany kod móc wywoływać wielokrotnie.

Przykład definicji metody

```
Def hello(n)
```

```
n.times do
```

```
  Puts "Hello"
```

```
end
```

```
end
```

Metoda



- W Ruby możemy wywoływać metodę bez nawiasów czyli np. `Hello 2,3`.
- W Ruby metoda może zwracać kilka argumentów np. `Return a,b,c`.
- W Ruby nie musimy pisać słowa `return` , wtedy metoda zwraca ostatnią linię kodu.
- W Ruby metoda może przyjmować argument domyślny np. `hello(a=5)`.

Metoda – wywołanie przykład



```
irb(main):001:0> def hello(n)
irb(main):002:1> n.times do
irb(main):003:2* puts "Hello"
irb(main):004:2> end
irb(main):005:1> end
=> nil

irb(main):006:0> hello(5)
Hello
Hello
Hello
Hello
Hello
=> 5
```

Metoda – zwracanie wartości



- Metoda może zwracać wynik za pomocą słowa return bądź jako ostatnia linia w definicji metody

```
irb(main):008:0> def suma(a,b)
irb(main):009:1> return a+b
irb(main):010:1> end
=> nil

irb(main):011:0> def suma(a,b)
irb(main):012:1> a+b
irb(main):013:1> end
=> nil
```

Metoda – wywołanie bez nawiasów



- składnia Ruby jak najprostsza:)

```
irb(main):007:0> hello 5
Hello
Hello
Hello.1 at 2013-09-24 11:17:17 +0200
Hello
Hello
Hello
=> 5
2013-09-24 11:17:17 +0200
```

Metoda – wywołanie z zwracaniem



```
irb(main):014:0> suma(5,8)
=> 13
770.1 at 2013-09-24 11:17:17
irb(main):015:0> puts suma(5,8)
13
=> nil
irb(main):016:0> wynik = suma 5,8
=> 13
```

Metoda wartość domyślna



```
MacBook-Pro-Micha:~michalos$ irb
irb(main):001:0> def metoda(a=5)
irb(main):002:1> puts a
irb(main):003:1> end
=> nil
irb(main):004:0> metoda
5
=> nil
```


Hash -wprowadzenie



- jest to struktura analogiczna do HashMapy w Javie.
- służy do przechowywania kluczy i wartości.
- w Ruby jest stosowana na każdym kroku.

Hash -wprowadzenie



```
h = {"name" => "Fred", "surname" => "Flintstone"}
```

```
h["name"] #=> "Fred"
```

```
h["name"] = "Wilma"
```

```
#=> {"name" => "Wilma", "surname" => "Flintstone"}
```

```
h[:name] = "Wilma"
```

```
#=> {"name" => "Wilma", "surname" => "Flintstone", :name => "Wilma"}
```

```
h1 = Hash.new # to samo co {}
```

Hash 2



```
h = {:foo => "bar", :bar => "baz"}
```

```
h.empty? #=> false
```

```
h.size #=> 2
```

```
h.length #=> 2
```

```
h.include?(:foo) #=> true
```

```
h.has_key?(:foo) #=> true #synonim include?
```

```
h.has_value?("bar") #=> true
```

```
h.index("bar") #=> :foo
```

Hash – usuwanie

```
h = { "a" => 100, "b" => 200 }
```

```
h.delete("a")
```

```
#=> 100
```

```
h.delete("z")
```

```
#=> nil
```

```
h.delete("z") { |el| "#{el} not found"
```

```
} #=> "z not found"
```

Usuwanie warunkowe



```
h = { "a" => 100, "b" => 200, "c" =>
300 }
```

```
h.delete_if { |key, value| key >= "b" }
#=> {"a"=>100}
```

Hash merge

```
h1 = { "a" => 100, "b" => 200 }
```

```
h2 = { "b" => 254, "c" => 300 }
```

```
h1.merge(h2)  #=> {"a"=>100, "b"=>254,  
"c"=>300}
```

```
h1.merge(h2){|key, oldval, newval| newval - oldval}  
  
#=> {"a"=>100, "b"=>54,  
"c"=>300}
```

```
h1  #=> {"a"=>100, "b"=>200}
```

Konwersja do tablicy -hash



```
h = { "c" => 300, "a" => 100, "d" => 400, "c" => 300 }
```

```
h.to_a #=> [{"c", 300}, {"a", 100}, {"d", 400}]
```

Symbol



- czasami nie potrzebujemy wartości zmiennych w takich wypadkach używamy symboli
- symbol rozpoczynamy za pomocą :

```
s1 = :michal
```

```
s2 = :michal
```

```
str1 = "Alicja"
```

```
str2 = "Alicja"
```

```
s1.object_id == s2.object_id
```

```
#=> true
```

```
str1.object_id == str2.object_id
```

```
#=> false
```


Symbole



```
key = "Jabłko"
```

```
histogram[key] = 10
```

```
key.sub!("o", "a")
```

```
#=> "Jabłka"
```

```
histogram["Jabłka"]
```

```
#=> nil !!!
```

```
histogram["Jabłko"]
```

```
#=> 10
```

Rubykoans



- rubykaons to ćwiczenia, które są pomocne w nauce języka Ruby.
- <http://rubykoans.com/>
- ściągamy spakowane zadania i rozpakowujemy.

Sposób rozwiązywania RubyKoans



- jeśli znamy odpowiedź to ją wypełniamy
- jeśli nie to testujemy w konsoli jak zachowuje się nasz program

Rubykoans odpalenie konkretnego ćwiczenia



```
MacBook-Pro-Micha:koans michalos$ ruby about_arrays.rb
```

Usun teraz cały spam (wiadomości, które znajdowały się w folderze „Spam” dłużej niż 30 dni, zostaną usunięte automatycznie)

Idea Bank. Bank nr 1 dla.	5 tys. zł darmowego debetu dla Ciebie. Bez formalności. - Jeśli nie widzisz po	11:04
Zlotewyprzedaże.pl	-70% HUGO BOSS - Koszulki w 30 odsłonach - 30% Blazy, Longsleeve'y i PC	09:02
Vanquis Bank	Pieniądze także dla Ciebie! Sprawdź - Jeśli nie widzisz poprawnie tej wiadomości	07:55
Groupon	Dania Kuchni Zydowskiej / Kurs 1 z 3 języków Obcych - Ciężko Ci? - Idź i	07:27
AfterMarket.pl	Domeny First Minute (2013-10-07) - Domeny First Minute Nowe, świeże domeni	05:27
Nissan	Gwarancja wyjątkowych doznań - Jeśli nie widzisz poprawnie tej wiadomości	04:32
Getin Bank	Odbierz gotówkę nawet w 1 dzień! - Jeśli nie widzisz poprawnie tej wiadomości	02:22
Ania z Finansowy Supermar.	Pogotowie gotówkowe czy oszczędności na procent? - Jeśli nie widzisz popr	6 paź
Groupon	Porcja Pierogów i Zupa / Karnet Open na Siłownię i Fitness - Groupon w Twoi	6 paź
Sklep Orange.pl	Zgarnij Vivabox o wartości 165 zł kupując Huawei Ascend P6 za 1 zł - Jeśli n	6 paź
AfterMarket.pl	Domeny First Minute (2013-10-06) - Domeny First Minute Nowe, świeże domeni	6 paź
Getin Bank	Sprawdź najniższą ratę leasingową! Decyzja w 20 min! - Jeśli nie widzisz pop	6 paź
Sklep Orange.pl	Zgarnij Vivabox o wartości 165 zł kupując Huawei Ascend P6 za 1 zł! - Jeśli n	5 paź

Mountains are again merely mountains

Luki Ruby Koans



```
class AboutClasses < Neo::Koan
  class Dog
  end

  def test_instances_of_classes_can_be_created_with_new
    fido = Dog.new
    assert_equal __, fido.class
  end
end
```

Nil

- nil reprezentuje wartość pustą.

```
MacBook-Pro-Micha:~ michalos$ irb
irb(main):001:0> a = nil
=> nil
irb(main):002:0> a.class
=> NilClass
irb(main):003:0> a.nil?
=> true
irb(main):004:0> puts "To nie jest nil" if a
=> nil
```

Gem



- gem to biblioteka , którą możemy używać w Ruby.
- ktoś wykonał za nas pracę a my z niej korzystamy:)
- poleceniem `gem install nazwa_gemu` instalujemy nasz gem

-

Sprawdzanie dostępnych gemów



- <https://www.ruby-toolbox.com/>
- czym gem bardziej popularny tym większa szansa ,że będzie dla nas dobry
- czym częstsze update tym większa pewność tego ,że gem nie posiada większej ilości błędów
- co zrobić jeśli nie możemy znaleźć gemu do naszego rozwiązania???

Przykładowo....



random_data 1.6.0

Random data generator


INSTALL > `gem install random_data`

[Download](#) [Documentation](#) [Badge](#) [Subscribe](#) [RSS](#)

Authors
Mike Subelsky, Tom Harris

Owners
 

Links [Homepage](#)

**Gemfile**

```
gem "random_data", "~> 1.6.0"
```

RandomData

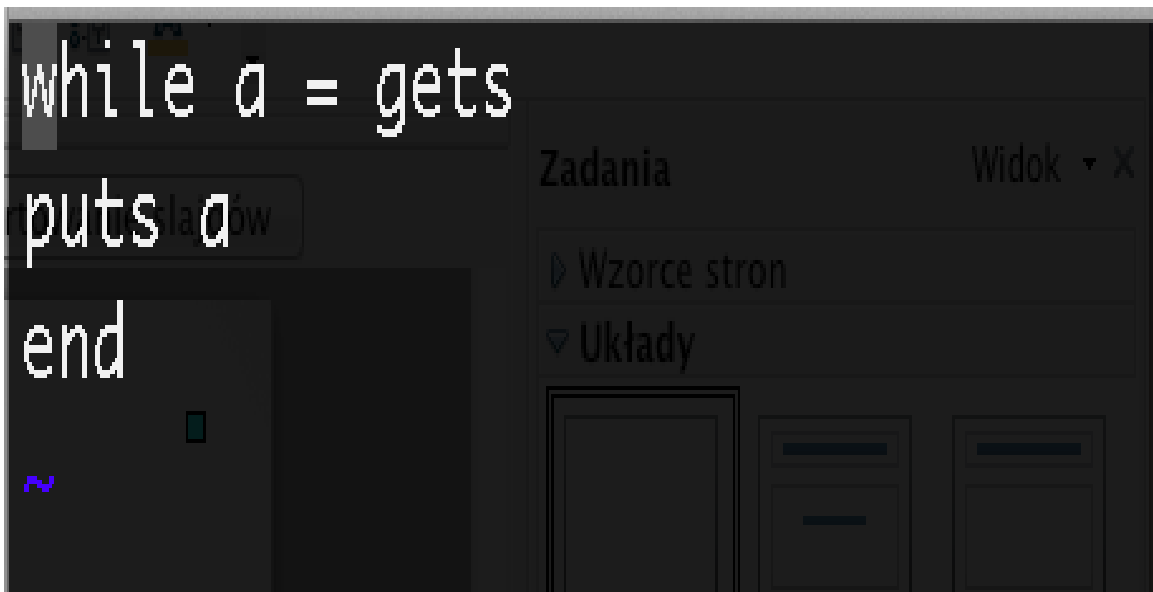
```
irb(main):002:0> require 'random_data'  
=> true  
irb(main):003:0> puts Random.firstname  
Charlotte  
=> nil  
irb(main):004:0>
```

Metoda class



- metoda class mówi nam do , której klasy należy dany obiekt
- dzięki temu możemy korzystać z dokumentacji odnośnie danej klasy
- ruby nie posiada jawnej deklaracji typu
- `is_a?(String)` sprawdza czy klasa jest danego typu.

Odczyt wielu danych



Łańcuch (String)



- String to sekwencja znaków.
- przechowujemy w nim wszelkie teksty
- size i length zwraca długość stringa

String przykłady



```
MacBook-Pro-Micha:~ michalos$ irb
irb(main):001:0> a = 'Witam'
=> "Witam"
irb(main):002:0> b='Dzieło Shakespera\'a'
=> "Dzieło Shakespera'a"
irb(main):003:0> c = "To jest znak tabulacji: (\t)"
=> "To jest znak tabulacji: (\t)"
irb(main):004:0> d = "Wynik #{5*3}"
=> "Wynik 15"
irb(main):005:0> e = 'Katalog C:\\TEMP'
=> "Katalog C:\\TEMP"
```

String - elementy

```
MacBook-Pro-Micha:~ michalos$ irb
irb(main):001:0> witam = "Witam"
=> "Witam"
irb(main):002:0> puts witam[0]
W
=> nil
irb(main):003:0> puts witam.length
5
=> nil
irb(main):004:0> puts witam.size
5
=> nil
irb(main):005:0> 
```

Tworzenie z łańcuchów tablice



```
irb(main):005:0> s1 = "Dawno dawno temu za górami za lasami"
=> "Dawno dawno temu za górami za lasami"
irb(main):006:0> s1.split
=> ["Dawno", "dawno", "temu", "za", "górami", "za", "lasami"]
irb(main):007:0> s2 = "Piotr, Adam, Grzesiek, Wojtek"
=> "Piotr, Adam, Grzesiek, Wojtek"
irb(main):008:0> s2.split(", ")
=> ["Piotr", "Adam", "Grzesiek", "Wojtek"]
```


Podłańcuchy

```
irb(main):009:0> a = "Adam Kowalski"  
=> "Adam Kowalski"  
irb(main):010:0> a[3,5]  
=> "m Kow"  
irb(main):011:0> a[-5,5]  
=> "alski"  
irb(main):012:0> a[4,-4]  
=> nil
```

String - zakresy

```
irb(main):001:0> a = "Witaj świecie , co słyszeć?"  
=> "Witaj świecie , co słyszeć?"  
irb(main):002:0> a[4..7]  
=> "j św"  
irb(main):003:0> a[4...7]  
=> "j ś"  
irb(main):004:0> a[4..-1]  
=> "j świecie , co słyszeć?"  
irb(main):005:0> a[3..8]  
=> "aj świ"
```

Zamiana na String



Do zamiany na String służy metoda `to_s`.

Może być ona wywoływana min. na tablicach, zmiennych całkowitych i zmiennoprzecinkowych.

```
irb(main):013:0> a = [1,2,3]
=> [1, 2, 3]
irb(main):014:0> a.to_s
=> "[1, 2, 3]"
irb(main):015:0> a.to_s.class
=> String
irb(main):016:0> a = 5
=> 5
irb(main):017:0> a.to_s
=> "5"
```

Zamiana na String



Do zamiany na String służy metoda `to_s`.

Może być ona wywoływana min. na tablicach, zmiennych całkowitych i zmiennoprzecinkowych.

```
irb(main):013:0> a = [1,2,3]
=> [1, 2, 3]
irb(main):014:0> a.to_s
=> "[1, 2, 3]"
irb(main):015:0> a.to_s.class
=> String
irb(main):016:0> a = 5
=> 5
irb(main):017:0> a.to_s
=> "5"
```

Metoda sub

- metoda sub zastępuje pierwsze wystąpienie określonego wzorca określonym podłańcuchem bądź blokiem

```
irb(main):006:0> a = "Napis"  
=> "Napis"  
  
irb(main):007:0> a = "Witaj witaj witaj Adam Kot Pies"  
=> "Witaj witaj witaj Adam Kot Pies"  
  
irb(main):008:0> a.sub("witaj", "cześć")  
=> "Witaj cześć witaj Adam Kot Pies"
```

Metoda reverse

- metoda reverse odwraca łańcuch znaków

```
irb(main):009:0> a = "Witam"  
=> "Witam"  
irb(main):010:0> a.reverse  
=> "matiW"  
irb(main):011:0> a.reverse!  
=> "matiW"  
irb(main):012:0> a  
=> "matiW"
```

Przeszukiwanie łańcuchów



- metoda `index` służy do przeszukiwania indeksu , na którym znajduje się znak w Stringu

```
irb(main):001:0> dane = "Michał Makaruk"  
=> "Michał Makaruk"  
irb(main):002:0> dane.index("M")  
=> 0  
irb(main):003:0> dane.index("Maka")  
=> 7  
irb(main):004:0> dane.index("Z")  
=> nil
```

Zamiana stringa na liczby

- metoda `to_f` służy do zamiany Stringa na floata
- metoda `to_i` służy do zamiany Stringa na inta

```
irb(main):009:0> liczba = 444
=> 444
irb(main):010:0> liczba.to_i
=> 444
irb(main):011:0> liczba.to_f
=> 444.0
irb(main):012:0> liczba = "444"
=> "444"
irb(main):013:0> liczba.to_i
=> 444
irb(main):014:0> liczba.to_f
=> 444.0
irb(main):015:0> liczba = "as3"
=> "as3"
irb(main):016:0> liczba.to_i
=> 0
```


Metoda split

- metoda split służy do rozdzielania łańcuchów

```
irb(main):005:0> a = "AAA,"  
=> "AAA,"  
irb(main):006:0> a.split(',')  
=> ["AAA"]  
irb(main):007:0> a = "AAA,adsads"  
=> "AAA,adsads"  
irb(main):008:0> a = "AAA,adsads,ads"  
=> "AAA,adsads,ads"  
irb(main):009:0> a.split(',')  
=> ["AAA", "adsads", "ads"]
```

" " via ' '



- " " interpretuje specjalne znaki takie jak `{kod_ruby}` , `\n` (znak nowej linii) , `\t` (tabulacja)
- " nie interpretuje specjalnych znaków (zazwyczaj zdarzają się wyjątki).

== via object_id == object2_id



- znak == sprawdza się czy wartości stringów są takie same
- == object_id sprawdza czy zmienne wskazują na ten sam obiekt w pamięci

Porównanie object_id

```
irb(main):001:0> ?a
=> "a"
irb(main):002:0> ?a
=> "a"
irb(main):003:0> a = "AAA"
=> "AAA"
irb(main):004:0> b = a
=> "AAA"
irb(main):005:0> a.object_id == b.object_id
=> true
```

Inne sposoby deklaracji String



```
irb(main):001:0> a = %(flexible quotes can handle both ' and " characters)
=> "flexible quotes can handle both ' and \" characters"
irb(main):002:0> long_string = <<EOS
irb(main):003:0" It was the best of times,
irb(main):004:0" It was the worst of times.
irb(main):005:0" EOS
=> "It was the best of times,\nIt was the worst of times.\n"
irb(main):006:0> long_string = %{
irb(main):007:0" It was the best of times,
irb(main):008:0" It was the worst of times.
irb(main):009:0" }
=> "\nIt was the best of times,\nIt was the worst of times.\n"
irb(main):010:0> █
```