Языки Интернет-программирования

Лекция 3. Ruby. Основы.

- Основные возможности
- Синтаксис
- Выражения и операции
- Некоторые базовые классы

МГТУ им. Н.Э. Баумана, доц. каф. ИУ-6,
к.т.н. Самарев Роман Станиславович

samarev@acm.org

17.09.2019

Язык Ruby

17.09.2019

- Автор языка:
  Юкихиро Мацумото (Matz)
- Первая официальная
  версия: 1995 г.
- Текущая версия Ruby 2.6.2 (Mar 2019)
- https://www.ruby-lang.org/
- http://tryruby.org/

```
# Ruby knows what you
# mean, even if you
# want to do math on
# an entire Array
cities  = %w[ Moscow
              London
              Oslo
              Paris
              Amsterdam
              Berlin ]
visited = %w[Berlin Oslo]

puts 'I still need ' +
     'to visit the ' +
     'following cities:',
     cities - visited
```

Основные свойства

17.09.2019

- Язык сверхвысокого уровня
- Объектный с динамической типизацией
- Универсальный (служебные скрипты, веб-приложения, графические программы с Qt-интерфейсом)

Основные идеи Ruby
Вольная интерпретация….

17.09.2019

- Текст программы - текст на естественном языке, понятный человеку
- Переменные и константы ссылаются на объект
- Всё есть объект, включая код методов
- Объект имеет методы и свойства (переменные и константы). Метод порождает объект
- Имя метода означает действие. Программируя, выбираем слово, которое лучше всего отражает смысл действия.
- Don’t Repeat Yourself

Базовый синтаксис

17.09.2019

- 7-ми битная ASCII-кодировка < Ruby 1.9
  Unicode >= Ruby 1.9

- Комментарии и строки могут быть в любой кодировке

- Программа исполняется сверху вниз. Отдельной главной функции не существует

Зарезервированные слова

17.09.2019

alias	and	BEGIN	begin	break
case	class	def	defined?	do
else	elsif	END	end	ensure
false	for	if	in	module
next	nil	not	or	rescue
redo	retry	return	self	super
then	true	undef	unless	until
when	while	yield		

Выражения

17.09.2019

- a = b + 5
- b = c * a
  c = a * 10
- b = c * a; c = a * 10
- Составное выражение (…)
- “Операторный” блок: {…} или do…end

Комментарии

17.09.2019

- Однострочные

```
# Программа на Ruby
if b == 2
 true                # специальный случай
else
 prime?(b)          # выполнится, если b != 2
end
```

- Многострочный комментарий

```
=begin
  Произвольный текст
          print "Ruby Program".
=end
```

Правила именования переменных

17.09.2019

Начинается	Назначение	Пример
Со строчной буквы	имена локальных переменных	local_variable
Со знака $ (доллар)	имена глобальных переменных	$global_variable
Со знака @	переменные экземпляра класса	@instance_variable
Со знака @@	переменные класса	@@class_variable
С прописной буквы	имена констант	TRUE
Со знака _	заменитель строчной буквы	__FILE__

Методы

17.09.2019

- Метод func1 возвращает результат вычисления **x*x**

```
def func1(x)
  x * x
end
```

- Методы func2 возвращает результат вычисления **y**

```
def func2(x)
  y = x
  y = x + 5 if x < 10
  y = x * 5 if x > 15
  return y
end
```

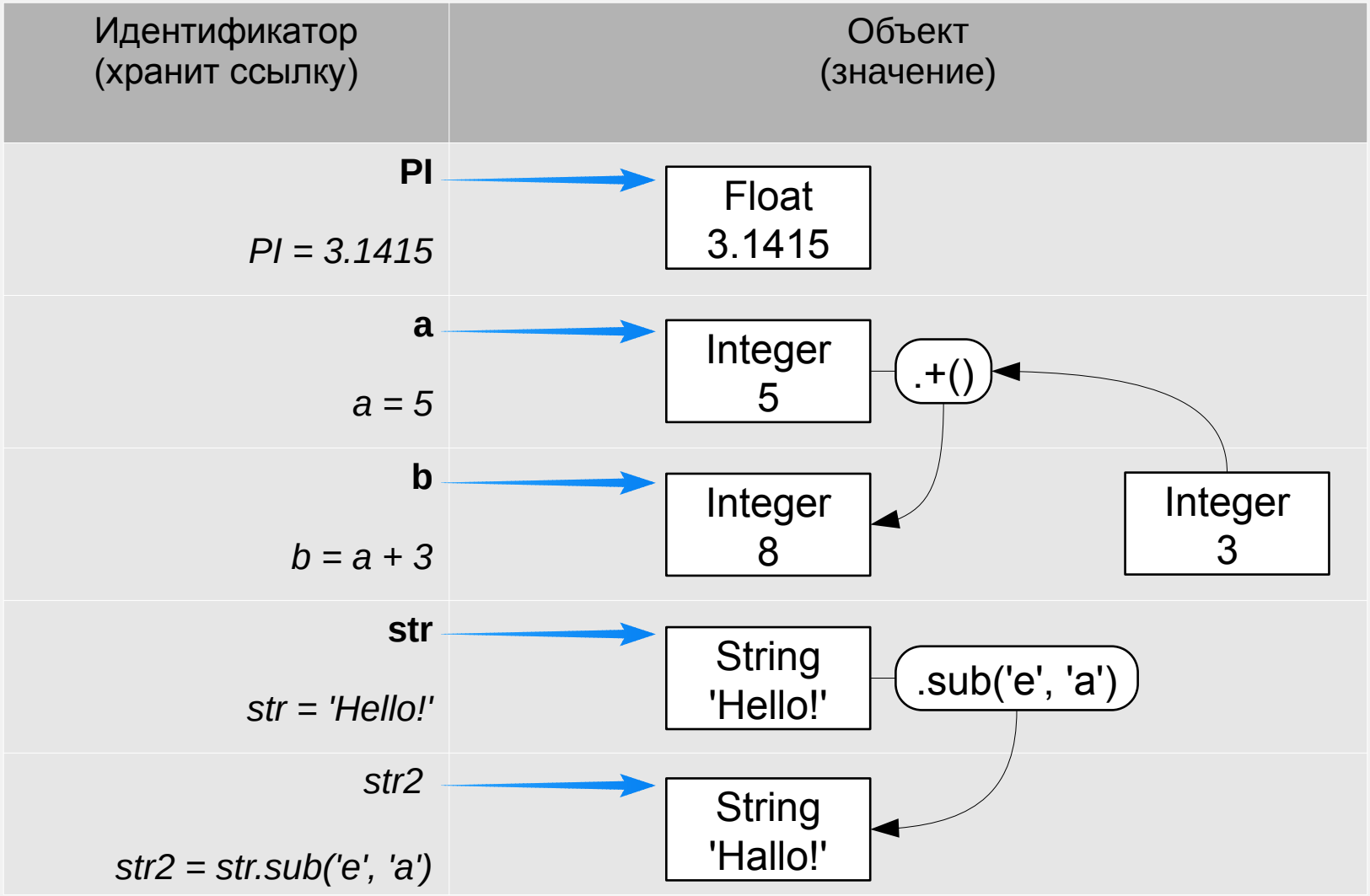
```
# Более короткая форма:
def func3(x)
   x<10 ? x+5 : (x>15 ? x*5 : x)
end
```

```
a = func1(10)   # вызов метода
```

- Метод возвращает результат **последней операции** или **return**
- Методы без класса присоединяются к Object !

Переменные, константы, объекты….

17.09.2019



Скобки для методов

17.09.2019

- Скобки можно не использовать
  - foobar
  - foobar ()
  - foobar a, b, c
  - foobar (a, b, c)

Особенность до версии 2.0

- Эквивалентные выражения:
  - x = y + z
  - x = y+z
  - x = y+ z
- Но не эквивалентно!
  - x = y **+z** # => x = y(+z)
- Для Ruby > 2.0 операции над переменными приоритетны

Правила именования методов

17.09.2019

- Имена методов начинаются со строчной буквы.
- Суффиксы:
  - ? – метод является предикатом (результат истина или ложь)
  - ! – метод производит изменение данных внутри объекта.

- Примеры:

```
obj.empty?          # объект пуст?
Numeric.nonzero?    # число не нуль?
obj.delete!         # удалить что-то из объекта!!!
obj.delete          # создать новый объект после удаления
obj.truncate!       # обрезать длину самого объекта
obj.remove_name!    # удалить имя из объекта
```

Приоритет операций

17.09.2019

- Отношение групп операций:
  [!, &&, ||] **>** [=, %=, ~=, /=, …] **>** [not, and, or]

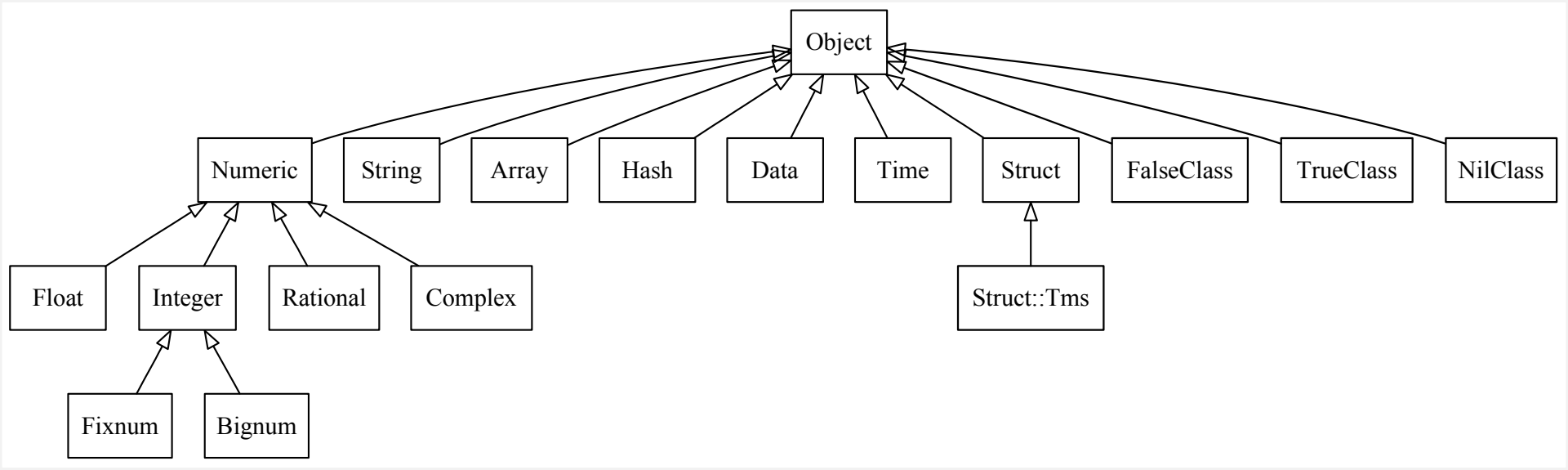
```
a = 'test'
b = nil
both = a && b       # both == nil
both = a and b      # both == 'test'
both = (a and b)    # both == nil
```

- https://ruby-doc.org/core-2.5.0/doc/syntax/precedence_rdoc.html
- http://www.tutorialspoint.com/ruby/ruby_operators.htm

Система типов

17.09.2019

```
anc_desc = {}
ObjectSpace.each_object(Class).select {|x| x < Object}.each {|c| anc_desc[c.name]=c.superclass.name}
File.open 'result.dot', 'w' do |file|
      file.puts %Q(digraph "Ruby #{RUBY_VERSION}" {\n)
      file.puts %Q(node [shape=box];\n edge [arrowtail="empty", dir=back];\n)
      anc_desc.each.sort_by{|desc, anc| anc+desc}.each{|desc, anc| file.puts %Q("#{anc}" -> "#{desc}";\n)}
      #               anc_desc.each {|desc, anc|  file.puts %Q("#{anc}" -> "#{desc}";\n)}
      file.puts '}';
end
system 'dot -Tsvg result.dot -o ruby.svg'
```



Числовые литералы

17.09.2019

```
5           # целое число
-12         # отрицательное целое число
4.5         # число с плавающей запятой
076         # восьмеричное число
0b010       # двоичное число
0x89        # шестнадцатиричное число

105327912   # число в десятичной записи
105_327_912 # то же в бухгалтерском формате
```

Некоторые специфические операторы

17.09.2019

- Множественное присваивание:

```
a, b = c, d;                a, b = b, a
a, b = [1, 2];              a, b, c = 10, 20, 30
```

- Операторы диапазона

```
1..10    # диапазон 1,2,..10
1...10   # диапазон 1,2,..9
```

- defined?

```
foo = 42
defined? foo    # => "local-variable"
defined? $_     # => "global-variable"
defined? bar    # => nil (undefined)
```

- http://www.tutorialspoint.com/ruby/ruby_operators.htm

“Операторный” блок

17.09.2019

- Однострочный

```
a = (1..3)
puts a.map { |n|  Math.sin(n) * 2 }.to_s
```

- Многострочный

```
a.map do |n|
        res = Math.sin(n)
        res * 2
end.to_s.tap { |obj| puts obj }
```

* Блок Ruby — объект, хранящий код

Ветвление

17.09.2019

- Выражения:

if conditional [then]
    code...
[elsif conditional [then]
    code...]...
[else
    code...]
end

unless conditional [then]
  code
[else
  code]
end

case expression
[when expression [, expression ...] [then]
  code]...
[else
  code]
end

- Модификаторы:

однострочные выражения
code **if condition**
code **unless conditional**

Примеры ветвлений

из книги «Фултон Х. Программирование на языке Ruby»

17.09.2019

Форма с if	Форма с unless

if x < 5 then
  statement
end
	unless x >= 5 then
	  statement
	end

if x > 2
  puts "x is greater than 2"
elsif x <= 2 and x!=0
  puts "x is 1"
else
  puts "I can't guess the number"
end
	unless x > 2 then
	  puts "x is less than 2"
	 else
	  puts "x is greater than 2"
	end

print "Value is set\n" **if $var**	print "Value is not set\n" **unless $var**
x = if a>0 then b else c end	x = unless a<=0 then c else b end

Особенность приведения к логическому типу

17.09.2019

Тестовая программа:

```
[nil, 0, 1, true, false, '', '123'].each do |i|
    puts i.inspect + "\t is true" if i
end
# puts [nil, 0, 1, true, false, '', '123'].select { |i| i }.map { |i| i.inspect + "\t is true" }
```

Результат:

```
0         is true
1         is true
true      is true
""        is true
"123"     is true
```

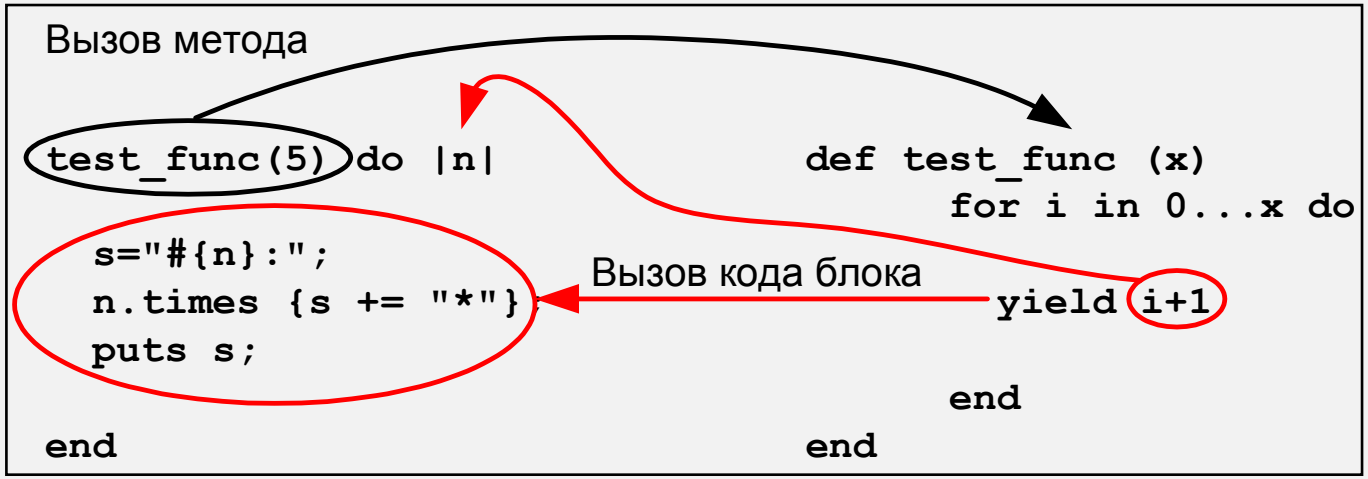
Только **false** и **nil** есть ложь!

Блоки

17.09.2019

- **Метод с блоком:**

```
# декларация метода
def test_func (x)
    for i in 0...x do
      yield i + 1
    end
end
# вызов метода
test_func(5) do |n|
    s="#{n}:"
    n.times {s += '*'}
    puts s
end
```



- *Результат:*

```
1:*
2:**
…
```

- **Эквивалентное преобразование кода:**

```
def test_func (x)
    for i in 0...x do
      n = i + 1; s = "#{n}:";  n.times { s += '*' }; puts s;
    end
end
test_func(5)
```

Циклы (редко используемые)

17.09.2019

```
# Цикл for для массива
for x in list do
  print "#{x} "
end

# Цикл for для диапазона
n = list.size - 1
for i in 0..n do
  print "#{list[i]} "
end

# Цикл 'loop'
i = 0;       n = list.size - 1
loop do
  print "#{list[i]} "
  i += 1
  break if i > n
  # break unless i <= n
end
```

```
# Цикл while
i = 0
while i < list.size do
  print "#{list[i]} "
  i += 1
end

# Цикл until
i = 0
until i == list.size do
  print "#{list[i]} "
  i += 1
end
```

Циклы (итераторы - методы, со смыслом)

17.09.2019

```ruby
# Цикл итератор 'upto'
n = list.size - 1
0.upto(n) { |i| print i, ' ' }
#=> 0 1 2 3 4  ...

# Цикл итератор 'downto'
5.downto(1) { |n| print n, '.. ' }
#=> "5.. 4.. 3.. 2.. 1..

# Цикл 'times'
n = list.size
5.times {|i| print i, ' ' }
#=> 0 1 2 3 4
```

```ruby
# Цикл итератор 'each'
list.each do |x|
  print "#{x} "
end

# Цикл итератор индекса 'each_index'
list.each_index do |i|
  print "#{list[i]} "
end

# Посчитать количество 'count'
puts (0..n).count { |i| i.odd? }

# Найти по условию 'detect'
puts [0, 1, 2, 3].detect { |i| i % 5 == 0 }
puts [0, 1, 2, 3].find { |i| i % 5 == 0 }
....
```

- https://ruby-doc.org/core-2.5.0/Enumerable.html

Исключения

17.09.2019

```
begin
  expression()    # контролируем выполнение выражения
[rescue [error_type [=> var],..]
  expr..].. # поймали исключение типа error_type
[else
  expr..]   # исключение, но не error_type
[ensure
  expr..]   # выполняется всегда
end
```

Основы классов

17.09.2019

- всё есть объект
- имя класса - это константа, которая ссылается на объект типа Class

```
p 1.class          # -> Fixnum
p 1.class.class    # -> Class
p Fixnum.class     # -> Class
p 'str'.class      # -> String
```

- Определение класса:

```
class Identifier [< superclass ]
  expr..
end
```

- **Синглтон** - одиночный глобальный экземпляр класса:

```
obj_name = []
class << obj_name
  expr …
end
```

```
log = []
class << log
  def out() puts 'Hi!' end
end
log.out
```

Пример класса

17.09.2019

```
class MyTest
  @@title = 'Тестирование производительности'      # переменная  класса

  def initialize(name, result)
    @name, @result = name, result                  # переменные экземпляра класса
  end

  def print_result
    puts "#{@name}: #{@result}"
  end

  def self.print_title
    puts @@title
  end
end

t1 = MyTest.new('module 1', 10);
t2 = MyTest.new('module 2', 50);

MyTest.print_title      # Тестирование производительности
t1.print_result         # module 1: 10
t2.print_result         # module 2: 50
```

Подключение файлов

17.09.2019

- подключение библиотечных файлов

метод: **require** 'filename'

- подключение файлов по относительному пути

Методы:
require_relative 'filename'
require './filename'

Строки
Класс String

17.09.2019

- Литералы (Ruby >=1.9 поддерживает Unicode)

```
str = 'Строка';                          path = '/home/user/'
str2 = 'Переменная \'str\' содержит строку'

str = "Знак табуляции: \t";              str = "Перенос \n строки"
str = "Еще один знак табуляции \011"

str = %q[Строка с символом переноса \n, но отображаемая как написано]
str = %Q[Строка с переносом \n строки]

str = <<EOF
Некоторый многострочный
    текст с отступами, которые так и перейдут в строку.
EOF
```

- http://www.ruby-doc.org/core-2.5.0/String.html

Вставка в код программы строк, содержащих UNICODE-символы

17.09.2019

- Обязательная декларация кодировки в первой строке!

```
# coding: utf-8

['Строка',
 'Переменная \'str\' содержит строку',

 "Знак табуляции: \t", "Перенос \n строки",
 "Еще один знак табуляции \011 !",

 %q[Строка с символом переноса \n, но отображаемая как написано],
 %Q[Строка с переносом \n строки],

 <<'EOF'
     Некоторый многострочный
     текст с отступами, которые так и перейдут в строку.
EOF
 ].each { |str| puts 'string: ' + str }
```

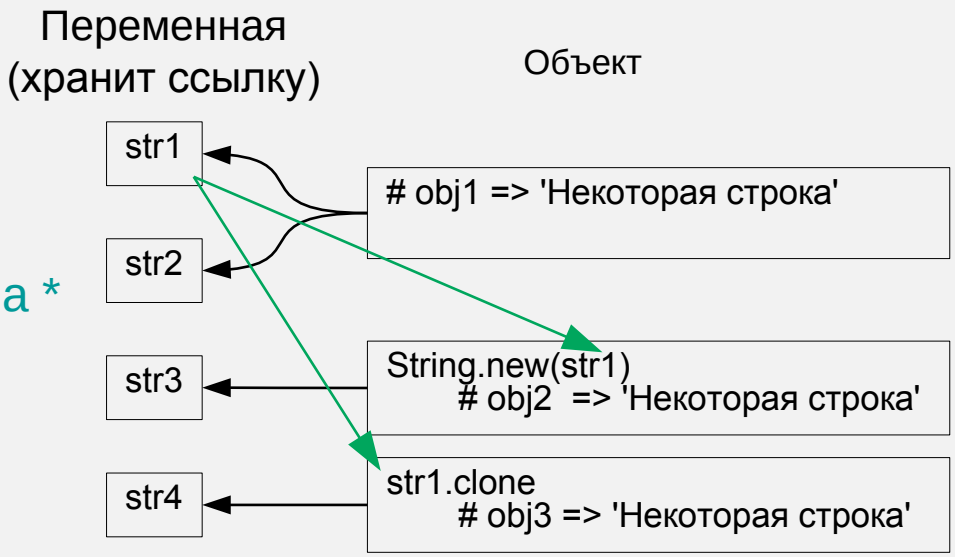
Примеры работы со строками

17.09.2019

```
# coding: utf-8

str1 = 'Некоторая строка'
str2 = str1
str3 = String.new(str1)
str4 = str1.clone
#заменим гласные в первой строке на *
str1.gsub!(/[еоая]/, '*')
puts str1, str2, str3, str4

#Н*к*т*р** стр*к*
#Н*к*т*р** стр*к*
#Некоторая строка
#Некоторая строка
```



- str1 и str2 содержали ссылку на один и тот же объект!
- str3 = String.new(str1) и str4 = str1.clone – явное создание копии объекта

Строки
Полезные операции

17.09.2019

a = 1; b = 4; puts "The number **#{a}** is less than **#{b}** "

"Some string".**include?** 'string'
"Ruby is a beautiful language".**start_with?** "Ruby"
"I can't work with any other language but Ruby".**end_with?** 'Ruby'
"I am a Rubyist".**index** 'R'
'Fear is the path to the dark side'.**split**
'Ruby' **+** 'Monk'
"Ruby".**concat**("Monk")
"I should look into your problem when I get time".**sub**('I','We')
"I should look into your problem when I get time".**gsub**('I','We')

- ASCII for Ruby <= 2.3
  - 'i am in lowercase'.**upcase** #=> 'I AM IN LOWERCASE'
  - 'This is Mixed CASE'.**downcase**
  - "ThiS iS A vErY ComPlEx SenTeNcE".**swapcase**
- UNICODE:
  - str.**mb_chars**.upcase и str.**mb_chars**.downcase – для UNICODE
  - > Ruby 2.4 - String/Symbol#upcase/downcase/swapcase/capitalize(!)

- https://ruby-doc.org/core-2.5.0/String.html

Регулярные выражения
Строки

17.09.2019

```
'Жыло-было шыбко шыпящее жывотное'.gsub(/(Ж|Ш|ж|ш)ы/) { $1 + 'и' }
    #=> "Жило-было шибко шипящее животное"

'Жыло-было шыбко шыпящее жывотное'.gsub(/([ЖШжш])ы/){ $1 + 'и' }
    #=> "Жило-было шибко шипящее животное"
```

Массив всех русских слов в тексте:

```
'Раз, два, три!'.scan(/[А-Яа-я]+/)    #=> ["Раз", "два", "три"]
```

Все знаки препинания:

```
'Раз, два, три!'.scan(/[, \.;:!]+/)    #=> [", ", ", ", "!"]
```

- http://ru.wikibooks.org/wiki/Ruby/Подробнее_о_строках
- http://rubular.com/ - online regular expressions editor

Регулярные выражения
Класс RegExp

17.09.2019

- Создание
  - Литералы /.../ или %r{...}
  - Конструктор Regexp::new

```
/hay/ =~ 'haystack'         #=> Позиция 0
/y/.match('haystack')       #=> #<MatchData "y">

s = 'a' * 25 + 'd' 'a' * 4 + 'c'
   #=> "aaaaaaaaaaaaaaaaaaaaaaaaadadadadac"
/(z|a+)*/ =~ s      #=> 0
/(z|a+)*c/ =~ s     #=> 32 - Пример не оптимального выражения!
```

- http://www.ruby-doc.org/core-2.4.0/Regexp.html

Регулярные выражения
Наборы символов

17.09.2019

- /./ - Любой символ кроме перевода строки.
- /./m – Любой символ (m разрешает перевод строки)
- /\w/ - Слово ([a-zA-Z0-9_])
- /\W/ - Не слово ([^a-zA-Z0-9_])
- /\d/ - Цифра ([0-9])
- /\D/ - Не цифра ([^0-9])
- /\h/ - Шестнадцатеричная цифра ([0-9a-fA-F])
- /\H/ - Не шестнадцатеричная цифра ([^0-9a-fA-F])
- /\s/ - символы пропуска: /[\t\r\n\f]/
- /\S/ - Не символы пропуска: /[^ \t\r\n\f]/

Регулярные выражения non-ASCII

17.09.2019

- /[[:alnum:]]/ - буква или цифра
- /[[:alpha:]]/ - буква
- /[[:blank:]]/ - пробел или табуляция
- /[[:cntrl:]]/ - управляющий символ
- /[[:digit:]]/ - цифра
- /[[:graph:]]/ - не пустой (исключаются пробелы, управляющие и пр.)
- /[[:lower:]]/ - буква в нижнем регистре
- /[[:print:]]/ - подобен [:graph:], но включает пробел
- /[[:punct:]]/ - знак препинания
- /[[:space:]]/ - пропуск ([:blank:], перевод строки и пр.)
- /[[:upper:]]/ - буква в верхнем регистре
- /[[:xdigit:]]/ - шестнадцатеричная цифра (i.e., 0-9a-fA-F)
- /[[:word:]]/ - символ в Unicode-категории: Letter, Mark, Number, Connector_Punctuation
- /[[:ascii:]]/ - символ в ASCII-кодировке

Регулярные выражения
Квантификаторы

17.09.2019

- * - ноль или более раз
- + - один или более раз
- ? – ноль или один раз
- {n} –точно n раз
- {n,} - n или более раз
- {,m} - m или менее раз
- {n,m} – не менее n и не более m раз

```
/<.+>/.match('<a><b>')  #=> #<MatchData "<a><b>">
```

Регулярные выражения
Якори

17.09.2019

- ^ - Начало строки
- $ - Конец строки
- \b – Граница слова
- \B – Не граница слова

- (?=pat) – Позитивный просмотр вперед. Найденная последовательность **соответствует, но не включает pat**.
- (?!pat) – Негативный просмотр вперед. Найденная последовательность **не соответствует и не включает pat**.
- (?<=pat) – Позитивный просмотр назад.

```
/(?<=<b>)\w+(?=<\/b>)/.match("Fortune favours the <b>bold</b>")
    #=> #<MatchData "bold">
```

- http://rubular.com/

Числа

17.09.2019

- **Fixnum** (целые числа, меньшие 2^{30});
- **Bignum** (целые числа, большие 2^{30});

} **Integer** in Ruby > 2.4

- **Float** (числа с плавающей запятой);
- Дробные числа **BigDecimal**;
- Рациональные числа **Rational**;
- Работа с матрицами **Matrix**;
- Комплексные числа **Complex**;
- Класс для порождения простых числе **Prime**.

Некоторые операции над числами

17.09.2019

- Возведение в степень

```
a = 64 ** 2       # 4096
d = 64 ** -1      # 0.015625
```

- Деление

```
3 / 4             # 0
3 / 4.0           # 0.75
```

- Приведение к плавающей точке

```
x = x.to_f / y
```

- Округление

```
pi = 3.14159
puts pi.round         # 3
temp = -47.6
puts temp.round       # -48
```

Форматирование вывода

17.09.2019

```
"%05d" % 123                    #=> "00123"
"%-5s: %08x" % [ "ID", self.object_id ]
                                #=> "ID   : 200e14d6 “
format "%05d", 123              #=> "00123"

x=123
x.to_s(2)      # 1111011

print "With #{x} values\n";
printf ("%8.2f", x/456.26)
```

Очень большие числа
Класс Bignum (до Ruby 2.4)

17.09.2019

```
x = 1000000
4.times do
  puts x.to_s() + "\t" + x.class.to_s()
  x *= x
end
```

Результат:

```
1000000 Fixnum
1000000000000           Bignum
1000000000000000000000  Bignum
1000000000000000000000000000000000000000000000  Bignum
```

Преобразование строки в число

17.09.2019

- Преобразование в число

```
num = '123'.to_i
num_f = '123.0123'.to_f
```

- Разбиение строки по словам

```
'123 abc 456 def 789 ghi'.split # ["123", "abc", "456", "def", "789", "ghi"]
```

- Разбиение строки по словам и преобразование в массив чисел

```
arr = gets.split.map(&:to_i)
```

- Выделение элементов по условию

```
'123 abc 456 def 789 ghi'.scan(/\d+/).map(&:to_i) # [123, 456, 789]
```

- Получение массива значений по строке формата

```
str = '123 abc 456 def 789 ghi'
str.scanf("%d%s") { |num,str| [ num * 2, str.upcase ] }
# => [[246, "ABC"], [912, "DEF"], [1578, "GHI"]]
```

Символы
Класс Symbol

17.09.2019

- Символ – аналог константной строки.

```
array = [          'string', 'string', 'string',
                   :string, :string, :string    ]
```

=> **3** объекта – String, **единственный** объект – Symbol

```
sym = :"This is a symbol"
```

- Преобразование в строки и обратно

```
a='somestr'
b=:somestr
a == b.to_str     # true
b == a.to_sym  # true
```

Диапазоны
Класс Range

17.09.2019

```
r1 = 1..3  # закрытый диапазон
r2 = 1...3 #открытый диапазон (не включая 3)
```

- Обход по диапазону

```
r1.each { |x| puts x }
(4..7).each { |x| puts x }
puts r1.first, r1.last
```

- Диапазоны со строками

```
a = 'a'.. 'z'
puts a.include? 'b'              # true
puts a.include? 'bb'             # false для ruby ≥1.9 и true для ≤1.8
puts ('a'.. 'zz').include?('bb')  # true
puts ('2'.. '5').include?('28')   # false для ≥1.9 и true для ≤1.8
```

Консольный вывод
Класс IO, объект STDOUT

17.09.2019

- **puts foo** - выводит foo как строку
  Эквивалентно **puts foo.to_s**
- **print** - выводит строку без **\n** в конце
- **printf** - аналогичен **C printf**
- **p foo** – вывод значения,
  эквивалентно puts foo.**inspect**

Консольный ввод
Класс IO, объект STDIN

17.09.2019

- **gets** - помещает результат ввода строки данных в переменную $_ и возвращает строку
- **getc** – читает один символ
- * метод String#encode позволяет перекодировать строку

Файлы
Класс File

17.09.2019

- Запись в файл (C-стиль)

```
f = File.new('out', 'w')
f.write('1234567890')   # пишем 10 символов
f.close
File.truncate('out', 5)    # обрезаем в 5 символов
File.size('out')
```

- Открытие в блоке (Ruby-стиль)

```
File.open('1.txt', 'w') do |f|
  f.puts 'что-то записывается в файл'
end
```

- Прочитать строки ruby-файла, который содержит этот код

```
File.open(__FILE__, 'r') do |f|
  while line = f.gets
    puts line
  end
end
```

Файлы
Класс File

17.09.2019

- Прочитать весь файл

```
str = File.read 'filename.txt'
```

- Прочитать все строки и сохранить в виде массива

```
array = File.readlines 'filename.txt'
```

- Проверить наличие файла

```
File.exist? 'filename.txt' # => true or false
```

- Проверить является ли директорией

```
File.directory?(file_name) # => true or false
```

- http://www.ruby-doc.org/core-2.5.0/File.html

Массивы
Класс Array

17.09.2019

- Создание при помощи литерала:

```
array = ['a', 'b', 'c', 'd', 'e']
array [array.size - 2]        #=> "d"
```

- Многомерные массивы:

```
[[1], [2, 3], [4]]   # разная длина элементов-массивов
[[1, 2], [3, 4]]     # одинаковая длина
```

- Создание при помощи метода класса new:

```
Array.new(size=0, obj=nil)
Array.new(array)
Array.new(size) { |index| block }
```

Массивы
Индексы элементов

17.09.2019

```
a = ['a', 'b', 'c', 'd', 'e' ]
a[0]          #=> "a"
a[6]          #=> nil
a[-2]         #=> "d" – 2-й с конца

a[1, 2]       #=> [ "b", "c" ]
a[1..3]       #=> [ "b", "c", "d" ]

a[3..-1]      #=> [ "d", "e" ] с 4-го с начала и до конца
a[-4, 2]      #=> [ "b", "c" ] с 4-го с конца, 2 элемента

str = '1234567890'
str[4..-1]    #=> "567890"
```

Массивы
Некоторые операции

- Проверка не пустого массива:

```
array = [1, 2, 4]
array.size > 0        #=> true
array.length > 0      #=> true

array.empty?          #=> false
array.any?            #=> true
```

- Поиск совпадения:

```
array = [1, 2, 3, 4, 5, 6, 7]
array.include?(5)   # true
```

Массивы
Определение max/min

17.09.2019

- Определение максимального/минимального элемента

['у', 'попа', 'была', 'собака'].max #=> "у" max по значению

['у', 'попа', 'была', 'собака'].max_by { |elem| elem.size }
#=> "собака" максимальный по размеру строки

['у', 'попа', 'была', 'собака'].min #=> "была" min по значению

['у', 'попа', 'была', 'собака'].min_by { |elem| elem.size }
#=> "у" минимальный по размеру строки

- http://ru.wikibooks.org/wiki/Ruby/Подробнее_о_массивах

Массивы
Сортировка

17.09.2019

- ['у', 'попа', 'была', 'собака'].sort
  #=> ["была", "попа", "собака", "у"] сортировка по значению

['у', 'попа', 'была', 'собака'].sort_by { |elem| elem.size }
  #=> ["у", "попа", "была", "собака"] сортировка по размеру строки

Для двумерных массивов:
[[1,0], [16,6], [2,1], [4,5],[4,0],[5,6]].sort_by { |elem| elem[1] }
#=> [[1, 0], [4, 0], [2, 1], [4, 5], [16, 6], [5, 6]] сортировка "внешних" элементов по значению "внутренних"
[[1,0], [16,6], [2,1], [4,5],[4,0],[5,6]].sort_by { |elem| elem[0] }
#=> [[1, 0], [2, 1], [4, 0], [4, 5], [5, 6], [16, 6]]

- http://ru.wikibooks.org/wiki/Ruby/Подробнее_о_массивах
- http://ru.wikibooks.org/wiki/Ruby/Справочник/Array
- https://ruby-doc.org/core-2.5.0/Array.html

Массивы
Слияние, вычитание…

17.09.2019

- Слияние/вычитание массивов

```
[1, 2, 3, 4] + [5, 6, 7] + [8, 9]          #=> [1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 1, 2, 2, 3, 3, 3, 4, 5] - [1, 2, 4]    #=> [3, 3, 3, 5]
```

- Удаление дубликатов:

```
[1, 2, 3, 4, 5, 5, 6, 0, 1, 2, 3, 4, 5, 7].uniq  #=> [1, 2, 3, 4, 5, 6, 0, 7]
```

- Размножение:

```
["1", "2", "3", "4"] * 2     #=> ["1", "2", "3", "4", "1", "2", "3", "4"]
[1, 2, 3, 4] * 2             #=> [1, 2, 3, 4, 1, 2, 3, 4]
[1, 2, 3, 4] + [1, 2, 3, 4]  #=> [1, 2, 3, 4, 1, 2, 3, 4]
```

- https://ruby-doc.org/core-2.5.0/Array.html

Ассоциативные массивы
Класс Hash

17.09.2019

- Создание при помощи литерала:

```
hash = {5 => 3, 1 => 6, 3 => 2}
hash[5]                 #=> 3
hash[2]                 #=> nil  - объект отсутствует
hash[3]                 #=> 2
```

- http://www.ruby-doc.org/core-2.5.0/Hash.html
- http://ru.wikibooks.org/wiki/Ruby/Справочник/Hash
- http://ru.wikibooks.org/wiki/Ruby/Подробнее_об_ассоциативных_массивах

Ассоциативные массивы
Создание конструктором

17.09.2019

```
h = Hash.new('Go Fish') # создается объект по-умолчанию! Иначе - nil
h["a"] = 100
h["b"] = 200
h["a"]              #-> 100
h["c"]              #-> "Go Fish"
# Изменяется единственный объект по-умолчанию
h["c"].upcase!      #-> "GO FISH"
h["d"]              #-> "GO FISH"
h.keys              #-> ["a", "b"]

# Создается новый объект по умолчанию каждый раз
h = Hash.new { |hash, key| hash[key] = "Go Fish: #{key}" }
h["c"]              #-> "Go Fish: c"
h["c"].upcase!      #-> "GO FISH: C"
h["d"]              #-> "Go Fish: d"
h.keys              #-> ["c", "d"]
```

Ассоциативные массивы

17.09.2019

- Создание из массивов

```
array = [1, 4, 5, 3, 2, 2]
Hash[*array]     #=> {1=>4, 5=>3, 2=>2}
#*array – оператор «splat»

array = [[1, 4], [5, 3], [2, 2]]
Hash[*array.flatten]      #=> {1=>4, 5=>3, 2=>2}
```

- Получение ключей и значение

```
{1=>4, 5=>3, 2=>2}.keys          #=> [1, 2, 5]
{1=>4, 5=>3, 2=>2}.values        #=> [4, 3, 2]
{a1: 4, b2: 3, :c3 => 2}.keys    #=> [:a1, :b2, :c3] — ключи типа Symbol
```

- http://www.ruby-doc.org/core-2.5.0/Hash.html

Множества

17.09.2019

- Частный случай массива

```
[1, 2, 3, 4, 5, 5, 6] | [0, 1, 2, 3, 4, 5, 7]
        #=> [1, 2, 3, 4, 5, 6, 0, 7]
```

- Класс Set

```
require 'set'
s1 = Set.new [1, 2]          # -> #<Set: {1, 2}>
s2 = [1, 2].to_set           # -> #<Set: {1, 2}>
s1 == s2                     # -> true
```

Множества. Класс Set
Некоторые операции

17.09.2019

```
s1 = Set.new [1, 2]
s2 = [1, 2].to_set
p s1 == s2
p s1.add("foo")          # -> #<Set: {1, 2, "foo"}>
p s1.add?("foo")         # -> nil (элемент уже добавлен)
p s1.merge([2, 6])       # -> #<Set: {6, 1, 2, "foo"}>
p s1.include?(1)         #-> true
p s2.subset?(s1)         #-> true
p s1.delete?(1)          # -> #<Set: {6, 2, "foo"}>
p s1.delete?(5)          # -> nil
p s1                     # -> #<Set: {6, 2, "foo"}>
```

- http://www.ruby-doc.org/stdlib-2.5.0/libdoc/set/rdoc/Set.html

Основные электронные ресурсы

17.09.2019

- http://www.ruby-lang.org/
- http://www.ruby-doc.org/
- http://tryruby.org
- http://rubymonk.com
- http://www.rubygems.org/

- http://ru.wikibooks.org/wiki/Ruby/Справочник

Литература

17.09.2019

- Основы языка программирования Ruby : учебное пособие / Р. С. Самарев. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2015. — 98, [2] с. : ил.
- Фултон Х. Программирование на языке Ruby.–М.:ДМК Пресс, 2007.-688 с.:ил.
- Д. Флэнаган, Ю. Мацумото. Язык программирования Ruby.–СПб.; Питер, 2011
- D. Thomas, C.Fowler, A. Hunt. Programming Ruby 1.9 & 2.0. The Pragmatic Programmers' Guide. (The Facets of Ruby) 4th Edition - Texas.Dallas: The Pragmatic Programmers, 2013 .- 888 p.

- http://ru.wikibooks.org/wiki/Ruby
- http://en.wikibooks.org/wiki/Ruby_Programming