CS 683 Emerging Technologies
Fall Semester, 2005
Doc 19 Ruby Regexp, Expression, Exceptions, Modules
Nov 8, 2005

References

Programming Ruby, The Pragmatic Programmers' Guide, Dave Thomas, ISBN 0-9745140-5-5

Some examples in this lecture are from the above text

Change in Presentation

Ubiquitous Presenter

In classroom ink on slides
In classroom student feedback

http://activecampus2.ucsd.edu/~mw4/CS683/

Ranges

a..b from a to b including b
a...b from a to b, excluding b

(2..4).to_a	[2, 3, 4]
(2...4).to_a	[2,3]
('a'..'d').to_a	['a', 'b', 'c', 'd']
('car'..'cat').to_a	['car', 'cas', 'cat']
(1..5) === 3	true
(1..5).include?(3)	true
(1..5).min	1
(1..5).reject {|k| k < 3}	[3, 4, 5]

Regular Expressions

Creation

```
a = Regexp.new('^\s*[a-z]')
b = /^\s*[a-z]/
c = %r{^\s*[a-z]}
```

Match

```
name = 'Roger Whitney'
name =~ /g/
/g/ =~ name
```

Pattern Matching Variables

```
$& what was matched by pattern
$` part of string preceding match
$' part of string after match
$~ MatchData object
```

```
$&  'g'
$`   'Ro'
$'    'er Whitney'
```

show_regexp for Later Slides

```
def show_regexp(string, pattern)
  if string =~ pattern
    "#{$`}<<#{$&}>>#{$'}"
  else
    'no match'
  end
end
```

```
show_regexp('cat', /a/)
show_regexp('yes | no', /\|/)
```

```
c<<a>>t
yes <<|>> no
```

Anchors

^	beginning of line
$	end of line
\A	beginning of string
\Z, \z	end of string
\b	word boundaries
\B	nonword boundaries

show_regexp('cat rat\nrat cat',/^rat/)	no match
show_regexp("cat rat\nrat cat",/^rat/)	cat rat <<rat>> cat
show_regexp("cat rat\nrat cat",/cat/)	<<cat>> rat rat cat
show_regexp("cat rat\nrat cat",/cat\Z/)	cat rat rat <<cat>>
show_regexp("cat rat\nrat cat",/\bat/)	no match
show_regexp("cat rat\nrat cat",/at/)	c<<at>> rat rat cat

Character Classes

[abc]	matches characters a, b or c
[^abc]	matches all characters except a, b, or c
.	matches any character except newline
[a-z]	matches all characters between a & z inclusive

\d	[0-9]
\D	[^0-9]
\s	[\s\t\r\n\f]
\S	[^\s\t\r\n\f]
\w	[A-Za-z0-9]
\W	[^ A-Za-z0-9]

show_regexp("bat cat rat", /[aeiou]/)	b<<a>>t cat rat
show_regexp("bat cat rat", /[\s]/)	bat<< >>cat rat
show_regexp("bat cat rat", /\s/)	bat<< >>cat rat
show_regexp("bat cat rat", /[\s][aeiou]/)	no match
show_regexp("bat cat rat", /[\s].[aeiou]/)	bat<< ca>>t rat
show_regexp("bat cat rat", /[a-z]/)	<<b>>at cat rat
show_regexp("bat cat rat", /[a-z]\s[a-z]/)	ba<<t c>>at rat
show_regexp("bat cat rat", /[a-z].[a-z]/)	<<bat>> cat rat

Repetition

r*	matches zero or more occurrences of r
r+	matches one or more occurrences of r
r?	matches zero or one occurrences of r
r{m.n}	matches at least m and at most n occurrences of r
r{m,}	matches at least m occurrences of r
r{m}	matches exactly m occurrences of r

show_regexp("bat cat rat sat", /[a-z]*/)	<<bat>> cat rat sat
show_regexp("bat cat rat sat", /\s.*\s*/)	bat<< cat rat sat>>
show_regexp("bat cat rat sat", /\s.*?\s/)	bat<< cat >>rat sat
show_regexp("bat cat rat sat", /(a|t){2,3}/)	b<<at>> cat rat sat

Substitution

a ="bat cat rat sat"	
a.sub(/[^aeiou]/, 'x')	xat cat rat sat
a.gsub(/[^aeiou]/, 'x')	xaxxxaxxxaxxxax
a.sub(/^./) {|x| x.upcase}	Bat cat rat sat
a.gsub(/[a]/) {|x| x.upcase}	bAt cAt rAt sAt
a.gsub(/\b\w/) {|x| x.upcase}	Bat Cat Rat Sat
a.gsub(/(\w+)\s(\w+)/, '\2 \1')	cat bat sat rat

Expressions

Operator Expressions

```
class Fixnum
  alias old_plus +

  def +(other)
    old_plus(other).succ
  end
end
```

```
1 + 2   -> 4
```

Boolean Expressions

Value	**Boolean Value**
nil, false	false
all other values	true

```
x = if 0
      5
    else
      10
    end
```

What is x?

Operator	**Explanation**
or, ||	or, shortcircuit evaluation
and, &&	and, shortcircuit evaluation
not, !	negation

Equality

Operator	Explanation
==	Equal value
===	and, shortcircuit evaluation
not, !	negation
<=>	Returns -1, 0, 1
=~	Regular expression match
eql?	True if receiver & argument have same type and equal values
equal?	compares object IDs

Case Expression

```
leap = case
      when year % 400 == 0: true
      when year % 100 == 0; false
      else year % 4 == 0
      end
puts leap
```

```
grade = case score
      when 0..60: 'F'
      when 61..70: 'D'
      when 71..80: 'C'
      when 81..90: 'B'
      when 90..100: 'A'
      else        'Illegal score'
      end
```

```
case input_line
when "test"
  run_test_cases
  print_test_results
when /p\s*(\w+)/
  print_source_code
when "q", "quit"
  exit
end
```

Break, Redo, Next & Retry

break	terminates immediate enclosing loop
redo	repeats loop from start without updating condition or fetching next element
next	start next iteration of loop
retry	restarts iterator loop

```
i = 0
loop do
  i +=1
  next if i < 3
  print i
  break if i > 4
end
```

Output

345

```
for k in 1..5
  puts "Now at #{k}. Restart?"
  retry if gets =~ /^y/
end
```

Input/Output

Now at 1. Restart? n
Now at 2. Restart? n
Now at 3. Restart? y
Now at 1. Restart? n
Now at 2. Restart? n
Now at 3. Restart? n
Now at 4. Restart? n
Now at 5. Restart? n

Variable Scope & Blocks, Oh no!

```
[1, 2, 3].each {|x| y = x + 1}
puts y , x
```

Result

Error, x & y not defined for puts

```
x = nil
y = nil
[1, 2, 3].each {|x| y = x + 1}
puts y , x
```

Result

4

3

```
if false
  x = nil
  y = nil
end
[1,2, 3].each {|x| y = x + 1}
puts y , x
```

Result

4

3

Exceptions

```
begin
  file = File.new('foo', 'w')
rescue SyntaxError => typo
  puts 'there was a typo' + typo
  raise typo
rescue NameError, IOError
  if file.path = 'foo'
    file = File.new('bar', w)
    retry
  else
    raise
  end
rescue                    #defaults StandardError
  puts 'error' + $!
  raise
rescue Exception => all_errors    #catches all errors
  puts 'This is the top'
else
  puts 'OK'
ensure
  file.close if nil != file
end
```

Exception Information

```
begin
  raise 'Extra info'
rescue RuntimeError => error
  puts "Message: " + error
end
```

Output

```
Message: Extra info
```

```
begin
  raise IndexError, 'Extra info'
rescue IndexError => error
  puts "Message: " + error
  puts "Message: " + error.message
end
```

Output

```
Message: Extra info
Message: Extra info
```

```
def test
  raise IndexError
end

begin
  test
rescue IndexError => error
  puts error.backtrace
end
```

Output

```
RaisingException.rb:19:in `test'
RaisingException.rb:22
Support/tmruby.rb:126:in `load'
Support/tmruby.rb:126
Support/tmruby.rb:100:in `fork'
Support/tmruby.rb:100
```

Catch-Throw

```
def sample
  x = 1
  throw :foo if x == 1
end

catch :foo do
  puts 'start'
  sample
  puts 'end'
end
```

Output

```
start
```

Modules

Namespace

File: example.rb

```
module Example
  PI = 3.1415
  @@x = 1

  def Example.foo
     @@x += 1
  end

  def instance_method
     @@x += 1
  end

  class Bar
    def hello
      'Hello'
    end
  end
end
```

Module

class methods
instance methods
constants
class variables

```
require 'example'

puts Example.foo
puts Example::PI
a = Example::Bar.new
puts a.hello

begin
  puts Example.instance_method
rescue ArgumentError
  puts 'Error'
end
```

Output

```
2
3.1415
Hello
Error
```

Mixins

file: sampleMixin.rb

```
module SampleMixin

 def foo(x)
   x + 1
 end

 def bar
   @y = 0 if @y == nil
   @y += 1
 end

 def name_clash
   'Mixin'
 end

 def call_other
   other + 1
 end

 def other
 end
end
```

file: useMixin.rb

```
require 'sampleMixin'

class UseMixin
 include SampleMixin

 def name_clash
   'Class'
 end

 def y
   @y
 end

 def other
   5
 end
end
```

```
require 'test/unit'
require 'useMixin'
class TestMixin < Test::Unit::TestCase
 def test_mixin
   a = UseMixin.new
   assert_equal(a.foo(2), 3)
   assert_equal(a.name_clash, 'Class')
   assert_equal(a.bar, 1)
   assert_equal(a.bar, 2)
   assert_equal(a.y, 2)
   assert_equal(a.call_other, 6)

 end
end
```