MI-RUB Ruby In Pieces

Pavel Strnad
pavel.strnad@fel.cvut.cz

Dept. of Computer Science, FEE CTU Prague,
Karlovo nám. 13, 121 35
Praha, Czech Republic

Contents

Ruby

Ruby is

- interpreted,
- dynamic typed,
- pure object-oriented language.

Hello World!

```
puts "Hello world!"

5.times { print "ahoj!\n" }
```

Fibonacci

```
def fib(n)
        a, b = 0, 1
        while b < n
                print b, " "
                a, b = b, a+b
        end
end

fib(100)
```

Objects

```
class Divka
        def initialize(jmeno, vek)
                @jmeno = jmeno
                @vek = vek
        end
        def to_s
                "Jmeno:\t#{@jmeno}\nVek:\t#{@vek}"
        end
        attr_reader :jmeno, :vek
end

moje_devce = Divka.new('Tereza', 18)
puts moje_devce.to_s
puts moje_devce.jmeno
puts moje_devce.vek
```

Ruby syntax `http://en.wikibooks.org/wiki/Ruby_Programming/Syntax`

Ruby provides many basic classes in its standard library:

- Array
- Hash
- Symbol
- String

Array

```
a = [ 1, 'cat', 3.14 ] # array with three elements
puts "The first element is #{a[0]}"
# set the third element
a[2] = nil
puts "The array is now #{a.inspect}"
```

```
a = [ 'ant', 'bee', 'cat', 'dog', 'elk' ]
a[0] # => "ant"
a[3] # => "dog"
# this is the same:
a = %w{ ant bee cat dog elk }
a[0] # => "ant"
a[3] # => "dog"
```

Hash

```
inst_section = {
  'cello' => 'string',
  'clarinet' => 'woodwind',
  'drum' => 'percussion',
  'oboe' => 'woodwind',
  'trumpet' => 'brass',
  'violin' => 'string'
}
```

```
p inst_section['oboe']
p inst_section['cello']
p inst_section['bassoon']
```

Ruby 1.9 way!

```
inst_section = {
  cello: 'string',
  clarinet: 'woodwind',
  drum: 'percussion',
  oboe: 'woodwind',
  trumpet: 'brass',
  violin: 'string'
}
puts "An oboe is a #{inst_section[:oboe]}"
```

Symbol

Symbols are constant names that you don’t have to predeclare and that are guaranteed to be unique!

Example

```
NORTH = 1
EAST = 2
SOUTH = 3
WEST = 4
Then, in the rest of your code, you could
use the constants instead of the numbers:
walk(NORTH)
look(EAST)
```

Ruby way!

Ruby provides much clearer way:

```
walk(:north)
look(:east)
```

Strings

Strings can be constructed two different ways.

```
'escape using "\\"' # => escape using "\"
'That\'s right' # => That's right
#OR interpreted string
par = 'boy'
"That's right #{par}." # => That's right boy.
```

Strings are objects!

```
"stressed".reverse # => "desserts"
"kRyPTON".upcase # => "KRYPTON"
```

If-Else contruct

```
if count > 10
  puts "Try again"
elsif tries == 3
  puts "You lose"
else
  puts "Enter a number"
end
```

While construct

```
while weight < 100 and num_pallets <= 30
  pallet = next_pallet()
  weight += pallet.weight
  num_pallets += 1
end
square = 2
square = square*square while square < 1000
```