Ruby : A Dynamic Object Oriented Scripting Language

Kuldeep Gharat
kuldeep@cse.iitb.ac.in

November 29, 2004

Standard Types

Overview

Ruby, like most other languages provides standard data types like :

- Numbers
- Strings
- Regular Expressions
- Anchors

Numbers

Number Classes

All numbers are objects of one of the three classes :

- Fixnum : Integers within a range – say from -2^{62} to 2^{62}-1
- Bignum : Integers outside the range of Fixnum
- Float : Floating point numbers

Conversions

Conversion of an integer from Fixnum class to Bignum class and vice-versa is transparent and done automatically by Ruby.

Numbers

Object Orientedness

Numbers are objects of their respective classes. Thus

We can find the absolute value of a number by using aNumber.abs and not abs(aNumber).

Floating points are written as 10.0e7 and not 10.e7. 10.e7 tries to call the method e7 of the corresponding class.

Iterators and Methods to manipulate Integers

Iterators

There are various iterators available for integers.

- times : 3.times {print "X "} will print X X X
- upto : 1.upto(5) { print i, " "} will print 1 2 3 4 5
- downto : 99.downto(95) { print i, " "} will print 99 98 97 96 95
- step : 50.step(70,5) { print i, " "} will print 50 55 60 65 70

methods

There are methods to convert numbers to integers and vice-versa:

- to_i : str.to_i will give integer representation of string str
- to_s : val.to_s will give string representation of number val

Strings

String Class

Strings in Ruby are sequences of 8-bit bytes. They are objects of the String class. Strings in Ruby are mostly similar to those in other languages like perl.

Substitution in Strings

One of the ways of string construction is by sustitution. This is done using the #{expr} sequence. Thus the value of any Ruby expression is substituted into the string.

Strings

String Construction

There are 3 ways to construct string literals in Ruby. They are :

- %q/general single-quoted string/
- %Q!general single-quoted string/
- Here document

Here Document Usage

Here Document Usage

A 'Here document' consists of lines in the source up to, but not including, the terminating string that must be specified after the << characters.

```
aString = <<END_OF_STRING
        The body of the string is the input
        lines upto one ending with the same
        text that followed the '<<'
END_OF_STRING
```

String Manipulation

Methods to Manipulate Strings

Here a few of the numerous(somewhere around 75) string manipulation methods.

'split' method

Splits each line into fields delimited by some character.
split can be used to extract Date, Month, Year from 10/12/2004

'scan' method

Extracts fields that match a regular expression.

String Manipulation

'chomp' method

Strips off newline character from the end of a line.
It is useful when input is read from console. The input lines are stripped of the newline characters at the end.

'squeeze' method

Trims runs of repeated characters.

Ranges

Ranges

Ranges occur naturally in the real world. Months, integers, etc..
Ruby lets us model these. Ruby models sequences, intervals and conditions using ranges. It uses the range operators "..", "..."

- (1..10).to_a >> results in [1 2 3 4 5 6 7 8 9]
- ('bar'..'bat').to_a >> results in ['bar', 'bas', 'bat']

Ranges

Methods for ranges

Here are a few methods that allow you to iterate over the ranges and test their contents.

digits = (1..9)

- include? : digits.include?(5)
- min : digits.min
- max : digits.max
- reject : digits.reject {i<5}
- each : digits.each do
  dial(digit)
  end

Regular Expressions

Regexp class

All regular expressions are objects of the class Regexp. They are used to match patters against strings.

Creating Regular Expressions

Regular Expressions can be created using of these :

- Regexp Constructor : Regexp.new(<regular expression>)
- Using /pattern/
- Using %r{pattern}

Matching Regular Expressions

Matching Objects to regular expressions

We can match an object to a regular expression by using the following

- 'match' method of Regexp class
- Match operators (positive match) and (negative match)

Anchors

Anchors

Regular expressions match patterns at the beginning of the string only. Anchors match a regular expression at the beginning or at the end.

^ matches the pattern at the beginning of a line only.

$ matches a pattern at the end of the line only.

`\A` matches a pattern at the beginning of a string.

`\Z` matches a pattern at the end of a string.

Anchors Examples

Examples

`str = "Hello World !\n This is IIT Bombay"`

- `/^Hello/` : matches Hello at the beginning of str
- `/Bombay$/` : matches Bombay at the end of the str
- `/^ This/` : matches the word ' This' in the middle of str.
- `/\AThis/` : does not match 'This' as it is in the middle of str.
- `/\AHello/`: matches 'Hello' because it is in the beginning of the string.

References

- http://www.ruby-doc.org/
- http://kylecordes.com/files/IntroToRuby.ppt
- http://pragmaticprogrammer.com/
- http://www.pragmaticprogrammer.com/talks/perlmongers

Thank you