UW Ruby Programming 110

Winter 2015

Michael Cohen

Lecture 1

Jan 8, 2015

Lecture 1

1. Intro
2. Whirlwind Tour of Ruby
3. Ruby Conventions
4. Working with Strings
5. Assignments
6. Resources

Section 1

Intro

What is Ruby?

— Object-oriented
— Dynamically typed
— Functional
— Open Source
— Elegant syntax

Brief History of Ruby

Created by Yukihiro "Matz" Matsumoto
First public release in 1995
Ruby 1.0 - Dec 25, 1996
Ruby 1.2 - Dec, 1998
Ruby 1.4 - Aug, 1999
Ruby 1.6 - Sep, 2000

Brief History of Ruby (continued)

Ruby 1.8 - Aug, 2003
Ruby 1.9 - Dec, 1997
Ruby 2.0 - Feb 24, 2013
Ruby 2.1 - Dec 25, 2013
Ruby 2.2 - Dec 25, 2014

Key Ruby Implementations

MRI **- "Matz" Ruby Interpreter**

JRuby **- Ruby on JVM**

Rubinius **- Ruby in Ruby**

RubyMotion **- Ruby for iOS, MacOSX, Android**

Philosophy

“I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language.”
— Matz

MINASWAN

“Matz is nice so we are nice”

First ruby program

create file `hello_world.rb`**:**

```
puts "Hello World!"
```

run the program:

```
ruby hello_world.rb
```

expected output:

```
Hello World!
```

Tools

- `ruby`
- `irb`
- `git` **and** `github.com`
- **text editor: SublimeText, Atom, vim, emacs, and many more...**

Section 2

Tour

Whirlwind Tour of Ruby

- Basic Types & Variables
- Basic Containers
- Conditionals & Loops
- Blocks & Iterators
- Methods & Classes

Section 2

Tour

Basic Types & Variables

Basic Types

Booleans

```
true
false
```

Basic Types

Nil

```
nil
```

Basic Types

Numbers

```
1          # Fixnum (i.e. integer)
3.14       # Float
0x1f77     # Hex
```

Basic Types

Numbers - basic operations

```
1 + 2     # addition
7 - 4     # subtraction
3 * 6     # multiplication
9 / 3     # division
7 % 3     # modulo
3 ** 9    # exponentiation
```

Basic Types

Numbers - comparisons

```
7 > 4     # greater-than
1 < 2     # less-than
1 <= 2    # less-than-equal
7 >= 4    # greater-than-equal
```

Basic Types

Ranges

```
1..5    # inclusive: 1, 2, 3, 4, 5
1...5   # exclusive: 1, 2, 3, 4
```

Basic Types

Strings

```
'hello world'   # single quoted
"hey there"     # double quoted
```

Basic Types

Symbols

```
:this_is_a_symbol
:another_symbol
```

Variables

```
x = "hey there"   # variable 'x' assignment

y = 1.0           # variable 'y' assignment

x = 2             # variable re-assignment
```

Section 2

Tour

Basic Containers

Basic Containers

Arrays

```
a = [1, 2, 3]                       # array of Fixnum
a[0]                                #=> 1

["hello", "world"]                  # array of String
[1, "two", 3.0]                     # mixed array
[[1, "one"], [2, "two"]] # nested array
```

Basic Containers

Hashes

```
# key-value mapping:
person = { :first => "Michael", :last => "Cohen"}

address = {
   :street => "123 Main St",
   :city => "Seattle",
   :zip => "98110"
}
```

Basic Containers

Hashes

```
# classic syntax:
{ :first => "Michael", :last => "Cohen"}

# new syntax:
{ first: "Michael", last: "Cohen"}
```

Basic Containers

Hashes

```
# retrieve a value for a key:
h[:city]        #=> "Bellevue"

# set a value for a key:
h[:city] = "Seattle"    # assignment
```

Section 2

Tour

Conditionals

Conditionals

if-elsif-else

```
happy = true
sad = true
if happy
  puts "clap your hands"
elsif sad
  puts ":-("
else
  puts "confused"
end
```

Conditionals

unless

```
sad = false
unless sad
   puts "clapping my hands"
end
```

Conditionals

modifiers

```
puts "clap your hands" if happy
puts "clap your hands" unless sad
```

Conditionals

case

```
case mood
  when "happy"
    puts ":-)"
  when "sad"
    puts ":-("
  when "surprised"
    puts ":-O"
  else
    puts ":-|"
end
```

Section 2

Tour

Loops

Loops

while

```
while happy
   puts "clap your hands"
end

puts ":-)" while happy
```

Loops

until

```
until happy
   puts ":-("
end

puts ":-)" until sad
```

Section 2

Tour

Blocks & Iterators

Basic example

```
3.times do
   print "Ho! "
end
```

Range iteration

```
(1..10).each do |i|
  puts i
end

(1..10).each {|i| puts i}
```

Array iteration

```
beatles = ["Paul", "John", "George", "Ringo"]
beatles.each do |beatle|
  puts beatle
end
```

Section 2

Tour

Methods

Methods

Basic example

```
def do_something
   puts "doing something"
end

do_something
```

Method with arguments

```
def squared(n)
   n*n
end

squared(2)     #=> 4
squared 3      #=> 9
```

Method calling another method

```
def cubed(n)
   n*squared(n)
end

cubed 3          #=> 27
```

Section 2

Tour

Classes

Classes

Basic example

```
class Shape
  def initialize(name)
    @name = name
  end

  def describe
    puts @name
  end
end

my_shape = Shape.new "fred"  # create a new instance of Shape
my_shape.describe            # invoke describe method on my_shape object
```

Classes

Inheritance

```
class Circle < Shape
  def initialize(name, radius)
    super name
    @radius = radius
  end

  def area
    3.14 * squared(@radius)
  end
end

my_circle = Circle.new "bob", 2  # create a new instance of Circle
my_circle.area                   # invoke area method on my_circle object
my_circle.describe               # invoke describe method on my_circle object
```

Section 3

Conventions

Conventions

- 2 spaces per tab
- UPPERCASE for constants
- CamelCase for class names
- snake_case for variables and methods
- method_name? for predicates
- method_name! for destructive methods

Section 4

Strings

Strings

Basics

```
'hello world'   # single quoted
"hey there"     # double quoted
```

Strings

Escaped characters

```
"hey there\n"      # new line

"actor\\writer"    #=> actor\writer
```

Strings

String interpolation

```
size = 20

# double-quoted:
"my size is #{size}"    #=> my size is 20

# single-quoted:
'my size is #{size}'    #=> my size is #{size}

# double-quoted with expression:
"my size is #{size*3}"  #=> my size is 60
```

Strings

Heredocs

```
long_string = <<ENDMARKER
  this is a very long string
  that covers multiple lines
  such as this
ENDMARKER
```

Strings

Comparisons

```
a = "aardvark"
b = "bee"
c = "cat"

a < c   #=> true
b > c   #=> false
a >= b  #=> false
b <= c  #=> true
```

Strings

spaceship operator: <=>

```
"abcdef" <=> "abcde"      #=> 1
"abcdef" <=> "abcdef"     #=> 0
"abcdef" <=> "abcdefg"    #=> -1
"abcdef" <=> "ABCDEF"     #=> 1
```

Strings

Concatenation

```
a = "aardvark"
b = "bee"

a + b       #=> "aardvarkbee"
a << b      #=> "aardvarkbee"
```

Strings

%w

```
strings1 = ["alpha", "beta", "charlie", "delta"]
strings2 = %w(alpha beta charlie delta)

strings1 == strings2    #=> true
```

Strings

length

```
# length:
a = "aardvark"
a.length              #=> 8

b = "bee"
b.length              #=> 3
```

Strings

slice, []

```
a = "hello there"
a.length          #-> 11

# single character:
a.slice 1         #=> "e"
a[1]              #=> "e"
a[4]              #=> "o"
a[11]             #=> nil
```

Strings

slice (cont)

```
a = "hello there"

# start, length:
a.slice 2, 3    #=> "llo"
a[2, 3]         #=> "llo"

# range from/to:
a[2..3]         #=> "ll"

# from end:
a[-3, 2]        #=> "er"
```

Strings

index

```
a = "hello there"

a.index 'e'     #=> 1
a.index 'lo'    #=> 3
a.index 'z'     #=> nil
```

Strings

conversions

```
pi = "3.14"
pi.to_i         #=> 3
pi.to_f         #=> 3.14

s = "hello"
s.to_sym        #=> :hello
s.to_i          #=> 0
s.to_f          #=> 0
```

Strings

empty?

```
s = "flipper is a dolphin"

s.empty?                    #=> false
"".empty?                   #=> true
"   ".empty?                #=> false
```

Strings

include?

```
s = "flipper is a dolphin"

s.include? "lip"     #=> true
s.include? "zip"     #=> false
```

Strings

start_with?

```
s = "flipper is a dolphin"

s.start_with? "fl"   #=> true
s.start_with? "li"   #=> false
```

Strings

end_with?

```
s = "flipper is a dolphin"

s.end_with? "phin"   #=> true
s.end_with? "z"      #=> false
```

Strings

case modification

```
s = "fliPPer NAME"

s.downcase     #=> "flipper name"

s.upcase       #=> "FLIPPER NAME"

s.swapcase     #=> "FLIppER name"

s.capitalize   #=> "Flipper name"
```

Strings

upcase vs. upcase!

```
s = "fliPPer NAME"

s.upcase          #=> "FLIPPER NAME"
s                 #=> "fliPPer NAME"

s.upcase!         #=> "FLIPPER NAME"
s                 #=> "FLIPPER NAME"
```

Strings

reverse / reverse!

```
s = "mary had a little lamb"

s.reverse    #=> "bmal elttil a dah yram"
```

Strings

chop / chop!

```
# chop removes the last character:
"string".chop          #=> "strin"
"".chop                #=> ""

"string".chop.chop     #=> "stri"
"x".chop.chop          #=> ""

"string\n".chop        #=> "string"
```

Strings

chomp / chomp!

```
# chomp removes the last record separator:
"hello".chomp                    #=> "hello"
"hello\n".chomp                  #=> "hello"
"hello \n there".chomp           #=> "hello \n there"

"hello".chomp("llo")             #=> "he"
```

Strings

delete / delete!

```
# removes characters from string:
"hello".delete "l"          #=> "heo"
"hello".delete "lo"         #=> "he"
"hello".delete "l","lo"     #=> "heo"

s = "Let's eat, Grandma!"
s.delete ","                #=> "Let's eat Grandma!"
```

Strings

split

```
s1 = "flipper is a dolphin"
s1.split          #=> ["flipper", "is", "a", "dolphin"]

s2 = "abracadabra"
s2.split "r"      #=> ["ab", "acadab", "a"]

"abc".split ""    #=> ["a", "b", "c"]
```

Strings

Iterators: each_char/line

```
s = "flipper"

s.each_char {|c| print c, " "}    #=> f l i p p e r

lines = "one\ntwo\nthree\n"
lines.each_line {|l| print l.chomp, " "}  #=> one two three
```

Section 5

Assignments

Assignments

Logistics

Assignment #1 due on 1/13/2015 @ 6pm

Pull request on:

```
https://github.com
        /planetcohen
        /uw-ruby-101
        /assignments/assignment01.rb
```

Assignments

Sample Problem - number_to_string

```
# implement a method `number_to_string`

# it accepts a number
# and returns the a string version of that number.

n1 = 123
number_to_string n1  #=> "one two three"

n2 = 246
number_to_string n2  #=> "two four six"
```

Assignments

Sample Problem - number_to_string continued

```
# extend `number_to_string` to accept a language specifier

n1 = 123

number_to_string n1, :en    #=> "one two three"

number_to_string n1, :de    #=> "eins zwei drei"

number_to_string n1, :fr    #=> "un deux trois"
```

Assignments

Problem 1 - titleize

```
# implement a method `titleize`

# it accepts a string
# and returns the same string with each word capitalized.

# Your method should generate the following results:

titleize "hEllo WORLD"            #=> "Hello World"

titleize "gooDbye CRUel wORLD"    #=> "Goodbye Cruel World"
```

Assignments

Problem 2 - my_reverse

```
# Write your own implementation of `reverse` called `my_reverse`
# You may *not* use the built-in `reverse` method

# Your method should generate the following results:

my_reverse "Hello World"           #=> "dlroW olleH"

my_reverse "Goodbye Cruel World"   #=> "dlroW leurC eybdooG"
```

Assignments

Problem 3 - palindrome?

```
# Write a method `palindrome?`
# that determines whether a string is a palindrome

# Your method should generate the following results:

palindrome? "abba"                                  #=> true
palindrome? "aBbA"                                  #=> true
palindrome? "abb"                                   #=> false

palindrome? "Able was I ere I saw elba"             #=> true
palindrome? "A man, a plan, a canal, Panama"        #=> true
```

Resources

Ruby Language
http://ruby-lang.org/

Ruby Docs
http://ruby-doc.org/

“PickAxe” book
http://ruby-doc.org/docs/ProgrammingRuby/