

Automation: Time to learn Ruby

Vyas Sekar

10/4/06

15-441 Ruby Recitation

1

Why do we want a scripting language?

- Why not Assembly, C, C++, Java ..
- Much easier to program in
 - ♦ Shorten the edit-develop-compile cycle
- Re-use existing components
 - ♦ E.g. TCP server, retrieve web pages
- Easy short-cuts for common operations
 - ♦ Text-processing, Strings, Regexp
- Fewer low-level hassles
 - ♦ Types, memory management etc

10/4/06

15-441 Ruby Recitation

2

Some examples

- Shell-script
- Sed/Awk
- Perl
- Python
- Tcl/Tk
- Smalltalk
-

10/4/06

15-441 Ruby Recitation

3

Some downsides ..

- Most commonly cited: Performance
 - ♦ Not good for ..
 - Compute-intensive operations
 - Creating data structures, algorithms
 - ♦ Less true as hardware makes up ..
- Common problem: unpredictable ..
 - ♦ Interpreted, not compiled
 - ♦ Don't require types/initialization
- Another common problem: mysterious..
 - ♦ *From the manpage: Perl actually stands for Pathologically Eclectic Rubbish Lister, but don't tell anyone I said that.*

10/4/06

15-441 Ruby Recitation

4

Ruby .. Some background

- Often called "multi-paradigm"
 - ♦ Procedural + OOP + Functional features
 - ♦ But a high-level scripting language!
- Philosophy: Principle of Least Surprise
 - ♦ What you expect is most likely what you get
- Features
 - ♦ Truly object-oriented
 - ♦ Support for Perl-like regular expressions
 - ♦ Syntax a lot like Python/Perl
- Trivia: The language was created by [Yukihiro "Matz" Matsumoto](#), 1995

10/4/06

15-441 Ruby Recitation

5

Okay ... Lets get started

File: Helloworld.rb

```
#!/usr/bin/ruby #<--
# please have useful comments
# unlike the one here!
def sayHelloworld(name) #<--
  puts "Hello world #{name}" #<--
end #<--
sayHelloworld("vyas") #<--
```

10/4/06

15-441 Ruby Recitation

6

Basic syntax rules

- Comments start with a # character, go to EOL
- Each expression is delimited by ; or newlines
 - Using newlines is the Ruby way
- Local variables, method parameters, and method names should all start with a lowercase letter or _
- Global variables are prefixed with a \$ sign
- Class names, module names, and constants should start with an uppercase letter
- Class instance variables begin with an @ sign

10/4/06

15-441 Ruby Recitation

7

Control structures

- The usual suspects
 - if, while, for, until
 - Iterators: each
- if example

```
if (score > 10)
  puts "You have cleared the checkpoint"
elsif (score > 5) # this is cool!
  puts " You have passed"
else
  puts "You have failed :-( "
end
```

10/4/06

15-441 Ruby Recitation

8

Control structures ..

- while example

```
team = 1
while (team <= maxteam)
  result = grade(team)
  team = team + 1
end
```
- Shortcuts:

```
puts "Passed!" if (score >= 10)
score = score+1 while (score <= 100)
```

10/4/06

15-441 Ruby Recitation

9

Arrays

- ```
array1 = Array.new
array1[0] = 1
array1[1] = 2
index = 0
#traditional way
while (index < array1.size)
 puts array1[index].to_s
 index = index + 1
end
```
- ```
array2 = [3, 4, 5, 6]
array2.each {|x| puts x} #Ruby way
```
- Useful functions: reverse, sort

10/4/06

15-441 Ruby Recitation

10

Hashes

- Most amazing feature of scripting languages
 - Along with regular expressions
- ```
hash1 = Hash.new
hash1["champions"] = "steelers"
hash1["runnersup"] = "seahawks"
hash1.each do |key,value|
 puts "#{key} are #{value}"
end
hash1.delete("runnersup")
```
- e.g. where you might use this

```
nick2ip["nick"] = ipaddr_connect
```

10/4/06

15-441 Ruby Recitation

11

## Strings and regular expressions

- ```
s = 'This is a new string '
earl = "Earl"
s = "My name is #{earl}"
answer = 42
s = 'The answer name is ' + answer.to_s
```
- Many useful functions:
to_i, upcase, downcase, reverse
 - Note: need explicit to_i (unlike perl)
- Regular expression matching

```
if string =~ / Hello\sWorld/
  puts "the string is Hello World"
end
```
- Commonly used regular expressions: \s, \w, \d, ., *, +

10/4/06

15-441 Ruby Recitation

12

Strings and regular expressions..

- Substitution:

```
language.sub(/Perl/, 'Ruby') #first  
language.gsub(/Perl/, 'Ruby') #all
```
- Interested in not only matching but also values?

```
s="12:50am"  
if s=~(/(\d+):(\d+)(\w+)/)  
  puts "Hour:#$1, Min:#$2 #3"  
end
```
- split example

```
helloworld='Hello World'  
(hello,world) = helloworld.split  
numbers='1,2,3,4,5'  
splitarray = numbers.split(',')
```

10/4/06

15-441 Ruby Recitation

13

Code blocks and yield

- Code blocks defined by {} or do-end
- yield example:

```
def method_yields  
  yield  
end  
method_yields { puts "Come here!" }
```
- Fancier example:

```
def fibUpTo(max)  
  i1, i2 = 1, 1 # parallel assignment  
  while i1 <= max  
    yield i1  
    i1, i2 = i2, i1+i2  
  end  
end  
fibUpTo(1000) { |f| print f, " " }
```
- each and iterators defined this way

10/4/06

15-441 Ruby Recitation

14

Basic file I/O

- Common printing tasks: printf, print, puts
- Getting user input: gets
- Reading a file

```
1. aFile = File.open("testfile", "r")  
  # process the file line by line  
  aFile.each_line do |line|  
    line_new = line.chomp  
    puts line_new  
  end  
2. IO.foreach("testfile") { |f| puts f }
```
- Getting rid of pesky EOL:

```
chomp, chomp!, chop, chop!
```
- Alternative reading whole file into array

```
arr = IO.readlines("testfile")
```

10/4/06

15-441 Ruby Recitation

15

Basic file I/O

- Writing to a file

```
wFile = File.open("debuglog", 'w')  
wFile.puts "debug message\n"  
wFile.close
```

10/4/06

15-441 Ruby Recitation

16

Classes

```
class IRC #class name starts in capital  
  attr_reader :server, :port #shortcut for access outside  
  attr_writer :nick #shortcut for writing @nick outside  
  def initialize(server, port, nick, channel) #constructor  
    @server = server #instance variables start with @  
    @port = port  
    @nick = nick  
    @channel = channel  
  end  
  def connect #another method  
    #instance variables don't need declaration  
    @server_connection = TCPSocket.open(@server, @port)  
  end  
  def send(s)  
    @server_connection.send(s)  
  end  
end  
ircc = IRC.new($SERVER, $PORT, '', '') #create an object of type IRC  
ircc.nick = 'vyas'
```

10/4/06

15-441 Ruby Recitation

17

Time for a little reflection ..

- A quick useful example
- Questions so far ...

10/4/06

15-441 Ruby Recitation

18

Topics Useful for Project Testing

- Controlling processes
- Network connections
- Handling exceptions
- Ruby unit testing

10/4/06

15-441 Ruby Recitation

19

Controlling processes

- System calls:

```
system("tar xzf test.tgz")
result = `date`
IO.popen("ls -la") { |f|
  f.each_line {|filename| puts filename}
}
```

- Forking a child

```
if ((pid=fork) == nil)
  exec("sort testfile > output.txt")
else
  puts "the child pid is #{pid}"
  cpid = Process.wait
  puts "child #{cpid} terminated"
end
```

10/4/06

15-441 Ruby Recitation

20

Controlling processes: Timeout

- Want to kill a child with a timeout
 - ♦ Frequently needed with network tests

```
if ((cpid = fork) == nil)
  while (1) # this is the child
    puts "first experiment"
  end
else
  before = Time.now
  flag = 1
  while (flag == 1)
    if ((Time.now - before) > timeout)
      Process.kill("HUP", cpid)
      flag = 0
    end
    sleep(sleepinterval)
  end
end
```

10/4/06

15-441 Ruby Recitation

21

Controlling processes: Timeout

- Want to kill a child with a timeout
 - ♦ Frequently needed with network tests

```
require 'timeout'
if ((cpid = fork) == nil)
  # this is the child
  begin
    status = Timeout.timeout(5) do
      while (1)
        puts "child experiment"
      end
    rescue Timeout::Error
      puts "Child terminated with timeout"
    end
  end
else
  puts Time.now
  Process.wait
  puts Time.now
end
```

10/4/06

15-441 Ruby Recitation

22

Network Connections

- Simple tcp client

```
require 'socket' #use this module
t=TCPSocket.new('localhost','ftp')
t.gets
t.close
```

- Simple tcp-echo server

```
require 'socket'
port = 34657
server = TCPServer.new('localhost',port)
while (session = server.accept)
  input = "#{session.gets}"
  session.print "#{input}\r\n"
  session.close
end
```

- Slightly advanced server: extend GServer

10/4/06

15-441 Ruby Recitation

23

Handling Exceptions a.k.a Having to code in the real world

```
f = File.open("testfile")
begin
  # .. process
rescue => err # local var to get exception
  # .. handle error
  puts "Error #{err}"
ensure
  # executed always, put cleanup here
  f.close unless f.nil?
end
```

- ensure is optional, goes after rescue
- retry Use with caution: Infinite loop!

10/4/06

15-441 Ruby Recitation

24

Ruby unit testing

```
require socket
require 'test/unit'
Class TestIRC < Test::Unit::TestCase
  def setup # runs setup before every test
    @irc = IRC.new('$SERVER','$PORT','','')
    @irc.connect()
  end
  def teardown # runs teardown after every test
    @irc.disconnect()
  end
  def test_USER_NICK # can define many such functions
    @irc.send_user('please give me :The MOTD')
    assert(@irc.test_silence(1))
    @irc.send_nick('gnychis') ##
    lookup types of asserts under docs
    assert(@irc.get_motd())
  end
end
```

10/4/06

15-441 Ruby Recitation

25

Useful resources

- <http://www.ruby-doc.org/docs/ProgrammingRuby/>
- <http://www.ruby-lang.org/en/documentation/quickstart/>
- <http://www.zenspider.com/Languages/Ruby/QuickRef.html>
- <http://sitekreator.com/satishtalim/introduction.html>
- ri Class, ri Class.method
- irb: do quick test of code snippets
- And of course google

10/4/06

15-441 Ruby Recitation

26

Parting thoughts ..

- Why scripting: programmer efficiency
- Treat them as programming languages
 - ♦ It may be quicker to write a longer but cleaner script than an arcane one-liner
- Avoid getting stuck with religious battles
 - ♦ Perl vs Ruby vs Python
- Bottom-line
 - ♦ Be efficient
 - ♦ Write good code

10/4/06

15-441 Ruby Recitation

27

Some personal views ..

- If you are a Perl/Python/Smalltalk expert ..
 - ♦ May want to stick to it for now..
 - ♦ But legend has it that
 - Ruby is much better ..
 - Transition is not that bad ..
- Use a reasonably *sane* language
 - ♦ Writing undecipherable code is good for obfuscated code contests not real life ..

10/4/06

15-441 Ruby Recitation

28

Announcements ..

- Collect HW1
- UDP socket example has been posted for reference

10/4/06

15-441 Ruby Recitation

29