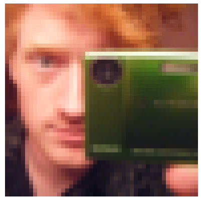
Ruby (on Rails)

CSE 190M, Spring 2009

Week 1

The Players

- Kelly "Everyday I'm Hustlin' " Dunn 
- Kim "Mouse" Todd
- Ryan "Papa T" Tucker

About the Section

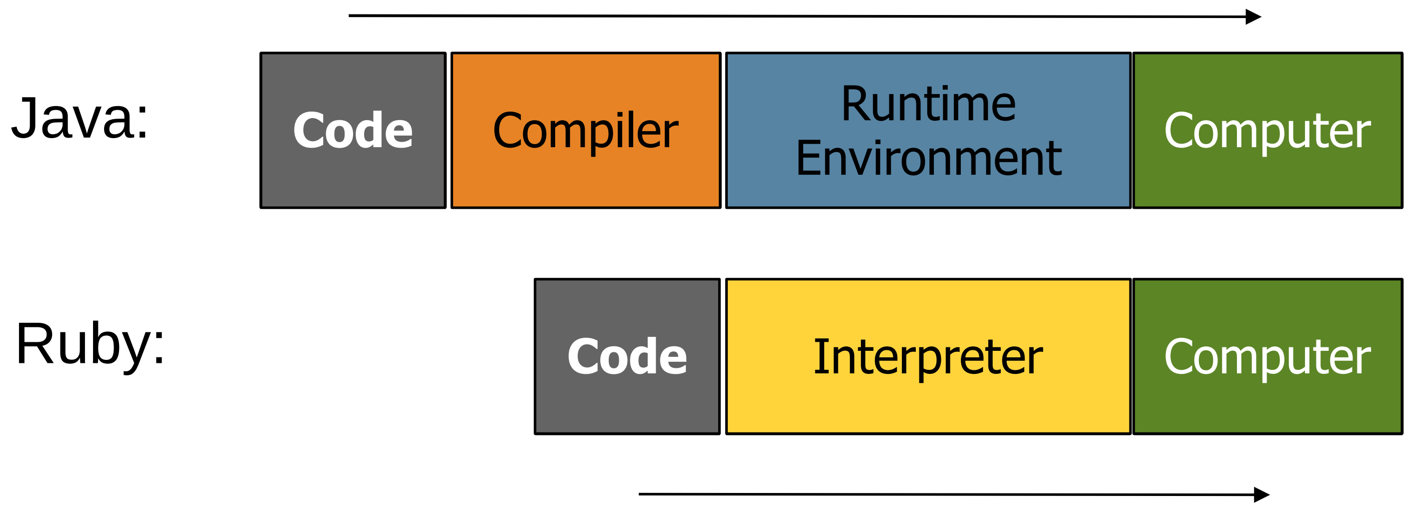
- Introduce the Ruby programming language
- Use Ruby to template web pages
- Learn about Ruby on Rails and its benefits

What is Ruby?

- Programming Language
- Object-oriented
- Interpreted

Interpreted Languages

- Not compiled like Java
- Code is written and then directly executed by an **interpreter**
- Type commands into interpreter and see immediate results



What is Ruby on Rails (RoR)

- Development framework for web applications written in Ruby
- Used by some of your favorite sites!

Advantages of a framework

- Standard features/functionality are built-in
- Predictable application organization
  - Easier to maintain
  - Easier to get things going

Installation

- Windows
  - Navigate to: http://www.ruby-lang.org/en/downloads/
  - Scroll down to "Ruby on Windows"
  - Download the "One-click Installer"
  - Follow the install instructions
    - Include RubyGems if possible (this will be necessary for Rails installation later)
- Mac/Linux
  - Probably already on your computer
  - OS X 10.4 ships with broken Ruby! Go here…
    - http://hivelogic.com/articles/view/ruby-rails-mongrel-mysql-osx

hello_world.rb

```ruby
puts "hello world!"
```

puts vs. print

- "puts" adds a new line after it is done
  – analogous System.out.println()

- "print" does not add a new line
  – analogous to System.out.print()

Running Ruby Programs

- Use the Ruby interpreter

```
ruby hello_world.rb
```

  - "ruby" tells the computer to use the Ruby interpreter

- Interactive Ruby (irb) console

```
irb
```

  - Get immediate feedback
  - Test Ruby features

Comments

```
# this is a single line comment

=begin
   this is a multiline comment
   nothing in here will be part of the code
=end
```

Variables

- Declaration – No need to declare a "type"
- Assignment – same as in Java
- Example:

```
x = "hello world"          # String
y = 3                      # Fixnum
z = 4.5                    # Float
r = 1..10                  # Range
```

Objects

- Everything is an object.
  - Common Types (Classes): Numbers, Strings, Ranges
  - nil, Ruby's equivalent of null is also an object
- Uses "dot-notation" like Java objects
- You can find the class of any variable

```
x = "hello"
x.class          →     String
```

- You can find the methods of any variable or class

```
x = "hello"
x.methods
String.methods
```

Objects (cont.)

- There are many methods that all Objects have
- Include the "?" in the method names, it is a Ruby naming convention for boolean methods
  - nil?
  - eql?/equal?
  - ==, !=, ===
  - instance_of?
  - is_a?
  - to_s

Numbers

- Numbers are objects
- Different Classes of Numbers
  - FixNum, Float

3.eql?2	→	false
-42.abs	→	42
3.4.round	→	3
3.6.rount	→	4
3.2.ceil	→	4
3.8.floor	→	3
3.zero?	→	false

String Methods

"hello world".length	→	11
"hello world".nil?	→	false
"".nil?	→	false
"ryan" > "kelly"	→	true
"hello_world!".instance_of?String	→	true
"hello" * 3	→	"hellohellohello"
"hello" + " world"	→	"hello world"
"hello world".index("w")	→	6

Operators and Logic

- Same as Java
  - Multiplication, division, addition, subtraction, etc.
- Also same as Java AND Python (WHA?!)
  - "and" and "or" as well as "&&" and "||"
- Strange things happen with Strings
  - String concatenation (+)
  - String multiplication (*)
- Case and Point: There are many ways to solve a problem in Ruby

if/elsif/else/end

- Must use "elsif" instead of "else if"
- Notice use of "end". It replaces closing curly braces in Java
- Example:

```
if (age < 35)
   puts "young whipper-snapper"
elsif  (age < 105)
   puts "80 is the new 30!"
else
   puts "wow… gratz..."
end
```

Inline "if" statements

- Original if-statement

```
if age < 105
   puts "don't worry, you are still young"
end
```

- Inline if-statement

```
puts "don't worry, you are still young" if age < 105
```

for-loops

- for-loops can use ranges
- Example 1:

```
for i in 1..10
      puts i
end
```

- Can also use blocks (covered next week)

```
3.times do
  puts "Ryan! "
end
```

for-loops and ranges

- You may need a more advanced range for your for-loop
- Bounds of a range can be expressions
- Example:

```
for i in 1..(2*5)
    puts i
end
```

while-loops

- Can also use blocks (next week)
- Cannot use "i++"
- Example:

```
i = 0
while i < 5
    puts i
    i = i + 1
end
```

unless

- "unless" is the logical opposite of "if"

- Example:

```
unless (age >= 105)              # if (age < 105)
  puts "young."
else
  puts "old."
end
```

until

- Similarly, "until" is the logical opposite of "while"
- Can specify a condition to have the loop stop (instead of continuing)
- Example

```
i = 0
until (i >= 5)      # while (i < 5), parenthesis not required
    puts I
    i = i + 1
end
```

Methods

- Structure

  def ***method_name***(***parameter1***, ***parameter2***, **…**)

          statements

  end

- Simple Example:

```
def print_ryan
  puts "Ryan"
end
```

Parameters

- No class/type required, just name them!
- Example:

```
def cumulative_sum(num1, num2)
   sum = 0
   for i in num1..num2
            sum = sum + i
   end
   return sum
end

# call the method and print the result
puts(cumulative_sum(1,5))
```

Return

- Ruby methods return the value of the last statement in the method, so…

```
def add(num1, num2)
  sum = num1 + num2
  return sum
end
```

can become

```
def add(num1, num2)
  num1 + num2
end
```

User Input

- "gets" method obtains input from a user
- Example

```
name = gets
puts "hello " + name + "!"
```

- Use chomp to get rid of the extra line

```
puts "hello" + name.chomp + "!"
```

- chomp removes trailing new lines

Changing types

- You may want to treat a String a number or a number as a String
  - to_i – converts to an integer (FixNum)
  - to_f – converts a String to a Float
  - to_s – converts a number to a String
- Examples

"3.5".to_i	→	3
"3.5".to_f	→	3.5
3.to_s	→	"3"

Constants

- In Ruby, constants begin with an Uppercase
- They should be assigned a value at most once
- This is why local variables begin with a lowercase
- Example:

```
Width = 5
def square
  puts  ("*" * Width + "\n") * Width
end
```

Week 1 Assignment

- Do the Space Needle homework from 142 in Ruby
    - http://www.cs.washington.edu/education/courses/cse142/08au/homework/2/spec.pdf
    - DOES need to scale using a constant
- Use syntax that is unique to Ruby whenever possible
- Expected output can be found under the Homework 2 Section
    - http://www.cs.washington.edu/education/courses/cse142/08au/homework.shtml

References

- Web Sites
  - http://www.ruby-lang.org/en/
  - http://rubyonrails.org/
- Books
  - Programming Ruby: The Pragmatic Programmers' Guide (http://www.rubycentral.com/book/)
  - Agile Web Development with Rails
  - Rails Recipes
  - Advanced Rails Recipes