In this lecture, we will discuss…

- Brief history of Ruby
- Another programming language to learn? Why?
- Basic Ruby principles and conventions

Ruby History

- Invented by Yukihiro “Matz” Matsumoto
- Version 1.0 released in 1996 (Japan)
- Popularized by Ruby on Rails beginning in 2005

Ruby: High Level Overview

- Dynamic

- Object-oriented
  - Object-possessed, almost everything is an object

- **Elegant**, **expressive and declarative**
  - Terse at times, but extremely readable

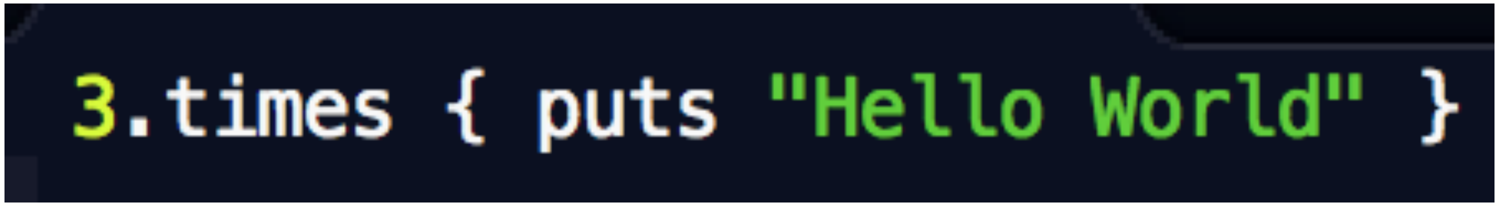
- Influenced by Perl, Smalltalk, Eiffel and Lisp

“Designed to make programmers happy”

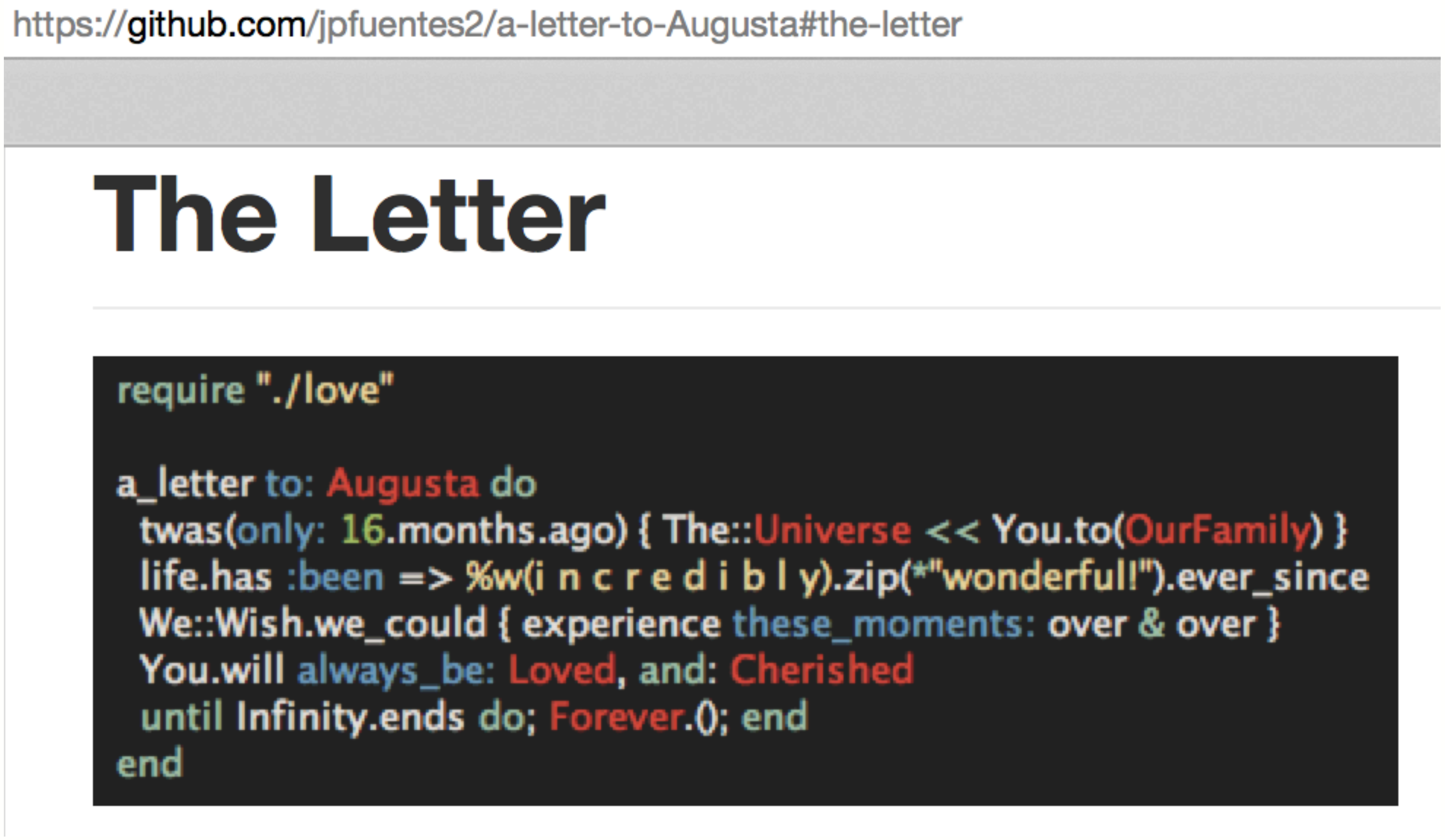
...Java...

```java
public class Print3Times {
  public static void main(String[] args){
    for(int i = 0; i < 3; i++) {
      System.out.println("Hello World!");
    }
  }
}
```

...Ruby...


```ruby
3.times { puts "Hello World" }
```

Carried away…



https://github.com/jpfuentes2/a-letter-to-Augusta#the-letter

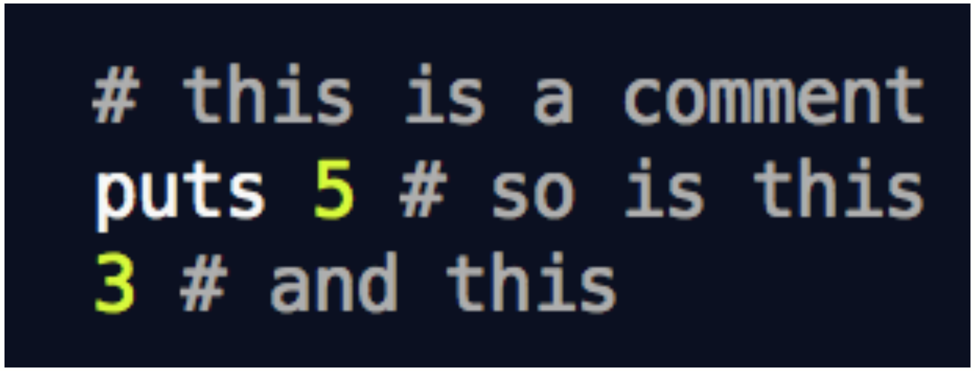
The Letter

```
require "./love"

a_letter to: Augusta do
  twas(only: 16.months.ago) { The::Universe << You.to(OurFamily) }
  life.has :been => %w(i n c r e d i b l y).zip(*"wonderful!").ever_since
  We::Wish.we_could { experience these_moments: over & over }
  You.will always_be: Loved, and: Cherished
  until Infinity.ends do; Forever.(); end
end
```

Ruby Basics

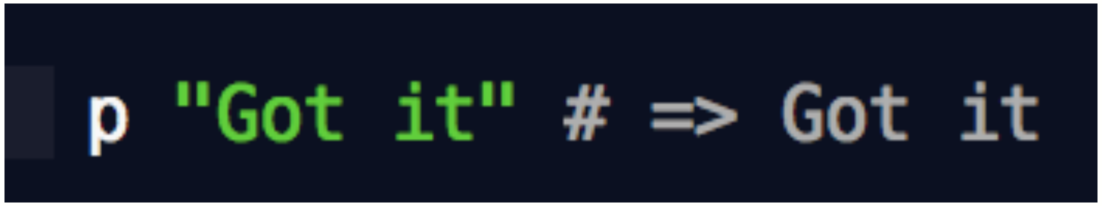
- 2 space indentation for each nested level is encouraged
  - Not required (unlike Python)
- # is used for comments
  - Use comments in moderation – the code itself should tell the story
- Everything is evaluated!



```
# this is a comment
puts 5 # so is this
3 # and this
```

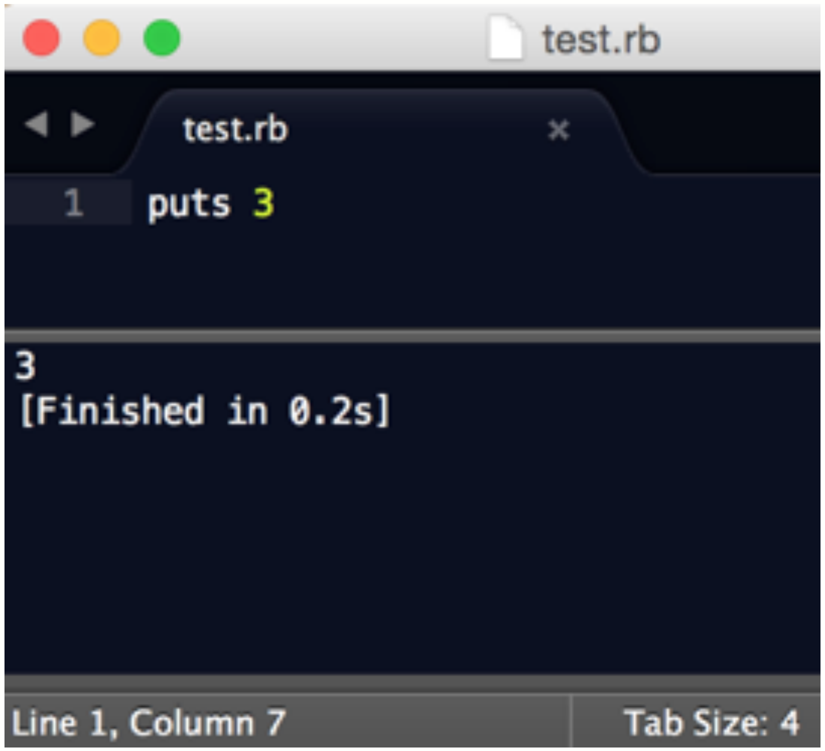
Printing to Console

- *puts* - Standard Ruby method to print strings to console (as in **put s**tring)
  - Adds a new line after the printed string
  - Similar to *System.out.println()* in Java
  - Used for most of the examples
- *p* - Prints out internal representation of an object
  - Debugger-style output



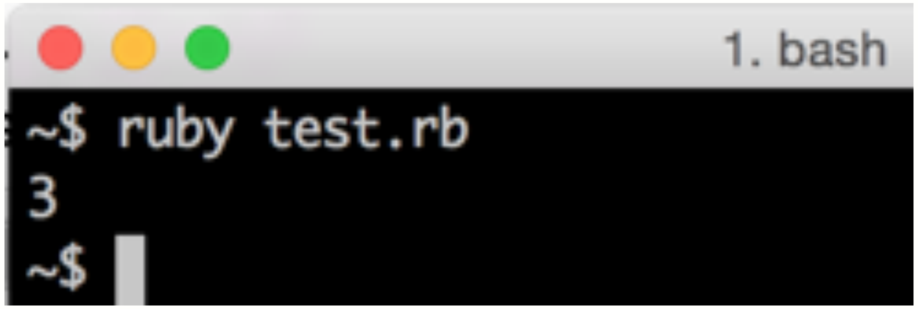
```
p "Got it" # => Got it
```

Executing Ruby



```
puts 3
```

```
3
[Finished in 0.2s]
```



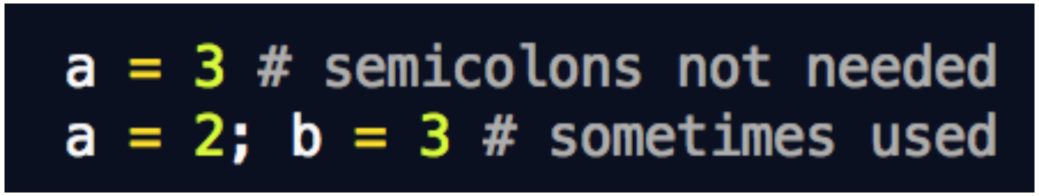
```
~$ ruby test.rb
3
~$
```

Naming Conventions

- Variables
  - Lowercase or `snake_case` if multiple words
- Constants
  - Either `ALL_CAPS` or `FirstCap`
- Classes (and Modules)
  - `CamelCase`

Drop the Semicolons

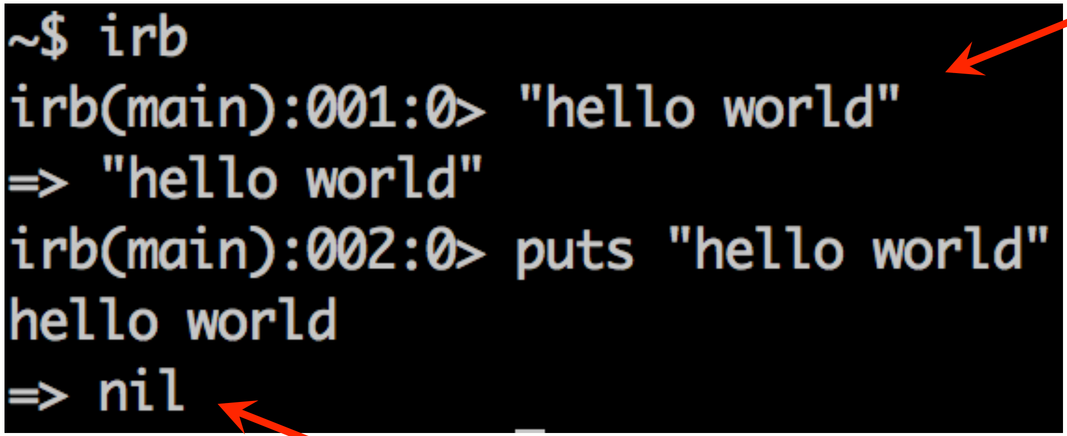
- Leave semicolons off at the end of the line
- Can cram several statements in with a semicolon in between
  - Usually highly discouraged



```
a = 3 # semicolons not needed
a = 2; b = 3 # sometimes used
```

IRB – Interactive Ruby

- Console-based interactive Ruby interpreter
  - REPL (Read Evaluate Print Loop)
- Comes with a Ruby installation
- Lets you experiment (quickly!)



```
~$ irb
irb(main):001:0> "hello world"
=> "hello world"
irb(main):002:0> puts "hello world"
hello world
=> nil
```

Anything evaluates to something – no need to assign to a variable

puts returns nil

Summary

- Ruby is extremely expressive
- Everything is evaluated

What's next?

- Flow of control in Ruby