Lesson 2

Data!

What is irb?

How do you write and run a Ruby program?

Why Should you comment your code?

Objectives for today

Making mathemagicians

Volatile variables

Putting the value into strings

Sexy symbols

An array (of flowers)

Smoking hashes

Looooooooping

But first, more Git stuff

Mathemagic

+	Addition	5 + 3 => 8
–	Subtraction	5 – 3 => 2
*	Multiplication	5 * 3 => 15
/	Division	5 / 3 => 1
%	Modulus	5 % 3 => 2
**	Power of	5 ** 3 => 125

```
a = 4
b = 2
c = 5


puts a + b * c
=> 14


puts (a + b) * c
=> 40
```

```
puts 2 / 4
=> 0


puts 2.0 / 4
=> 0.5
```

```
puts 2 / 3
=> 0


puts 2.0 / 3
=> 0.6666666666666666
```

Variables

Can combine letters, numbers, and underscore

By convention, all lowercase, words separated by underscore

Cannot begin with a number

```
name
first_name
_value
person1
long_cat_is_long
```

Can be preceded by one of the following funny symbols, which means assigns it a scope:

$, @@ or @

Variables with no funny symbol are local

WTF is a scope? Scope is where can a variable be read after it is assigned

```
age, name = 13, "Oscar the Cat"

puts age
=> 13


puts name
=> “Oscar the Cat”
```

.to_s – converts variable value to string

.to_f – converts variable value to floating point

.to_i – converts variable value to integer

```
a, b = 4, 3
```

```
puts a / b
=> 1
```

```
puts a / b.to_f
=> 1.3333333333333333
```

Strings and symbols

Strings are a list of characters

Mutable, they can be changed

Used to hold and manipulate text for display

```
puts "ruby" + " rocks"
=> ruby rocks
```

Symbols are a list of characters, preceded by a colon

Immutable, they *cannot* be changed

Commonly used as key names in hashes

```
puts :ruby + " rocks"
=> NoMethodError: undefined method "+"
for :ruby:Symbol
```

```
first = "Brian"
last = "May"
age = 65
```

```
puts "#{first} #{last} is #{age}"
=> Brian May is 65
```

$stdin.gets – wait for user input, and return the value

$stdin.gets.chomp – same, removes the trailing line return

```
puts "Type your name:"
input = $stdin.gets.chomp
puts "You typed #{input} on the command line"
=> You typed Rik on the command line
```

Strings are objects of the String class

String class supports methods to modify strings, such as

```
name = "Roger"
puts name.upcase
=> ROGER
```

```
name = "John"
puts name.downcase
=> john
```

Loooads more at
http://www.ruby-doc.org/core-1.9.3/String.html

Hip hip array

```
[]
```

```
[“John”]
```

```
[“John”, “Paul”, “George”, “The Drummer”]
```

```
band = [“John”, “Paul”, “George”, “The Drummer”]

band[0]
=> “John”

band[3]
=> “The Drummer”
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

take_that.unshift(“Robbie”)
=> [“Robbie”, “Gary”, “Mark”, “Howard”, “Jason”]

he_left = take_that.shift

puts take_that
=> [“Gary”, “Mark”, “Howard”, “Jason”]

puts he_left
=> “Robbie”
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

take_that.push(“Robbie”)
=> [“Gary”, “Mark”, “Howard”, “Jason”, “Robbie”]

take_that.pop
=> [“Gary”, “Mark”, “Howard”, “Jason”]
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

take_that << “Robbie”
=> [“Gary”, “Mark”, “Howard”, “Jason”, “Robbie”]
```

http://www.ruby-doc.org/core-1.9.3/Array.html

Hashes

```
{}
{ “name” => “Robbie” }

{ “name” => “Robbie”, “expression” => “Smug” }
```

```
{}

{ :name => “Robbie” }

{ :name => “Robbie”, :expression => “Smug” }
```

```
{}

{ name: “Robbie” }

{ name: “Robbie”, expression: “Smug” }
```

```
rob = { name: “Robbie”, expression: “Smug” }

puts rob[:name]
=> “Robbie”
```

```
rob[:age] = 38

puts rob[:age]
=> 38
```

```
band = Hash.new
band[:singer] = "Freddie"

band.has_key?(:singer)
=> true
band.has_value?("Brian")
=> false
band.empty?
=> false
band.size
=> 1
```

http://www.ruby-doc.org/core-1.9.3/Hash.html

Looooooooooooops

```
count = 1
loop do
    puts count
    count = count + 1
    break if count > 5
end
```

```
=>  1
=>  2
=>  3
=>  4
=>  5
```

```
count = 1
while count < 5 do
    puts count
    count = count + 1
end
```

```
=> 1
=> 2
=> 3
=> 4
=> 5
```

```
count = 1
until count == 5 do
    puts count
    count = count + 1
end
```

```
=> 1
=> 2
=> 3
=> 4
=> 5
```

```
3.times do
    puts “Odelay!”
end
```

```
=> “Odelay!”
=> “Odelay!”
=> “Odelay!”
```

```
3.times do |num|
    puts “Odelay number #{num}!”
end
```

```
=> “Odelay number 0!”
=> “Odelay number 1!”
=> “Odelay number 2!”
```

```
3.times do |i|
    2.times do |j|
        puts “#{i} x #{j}”
    end
end
```

```
=> “0 x 0”
=> “0 x 1”
=> “1 x 0”
=> “1 x 1”
=> “2 x 0”
=> “2 x 1”
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

for member in take_that
    puts member
end
```

```
=> “Gary”
=> “Mark”
=> “Howard”
=> “Jason”
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

take_that.each do |member|
    puts member
end
```

```
=> “Gary”
=> “Mark”
=> “Howard”
=> “Jason”
```

```
rob = { name: “Robbie”, expression: “Smug” }

rob.each do |key, value|
    puts “#{key.to_s} is #{value}”
end

=> “name is Robbie”
=> “expression is Smug”
```

```
(2..5).each do |count|
    puts count
end
```

```
=> 2
=> 3
=> 4
=> 5
```

```
(2...5).each do |count|
    puts count
end
```

```
=> 2
=> 3
=> 4
```

```
take_that = [“Gary”, “Mark”, “Howard”, “Jason”]

take_that[0..2]
=> [“Gary”, “Mark”, “Howard”]

take_that[1..2]
=> [“Mark”, “Howard”]

take_that[0...2]
=> [“Gary”, “Mark”]
```