
non_eggroll = 0
kitty_toys.each do |toy|
next if toy[:shape] == 'eggroll'
non_eggroll = non_eggroll + 1
end
kitty_toys = [
{:shape => 'sock', :fabric => 'cashmere'},
{:shape => 'mouse', :fabric => 'calico'},
{:shape => 'eggroll', :fabric => 'chenille'}
#Save the jibberish to a new file
print "File encoded. Please enter a name for this idea: "
idea_name = gets.strip
File::open('idea-' + idea_name + '.txt', 'w') do |f|
f << idea
end
DO... OR DO NOT THERE IS NO TRY!
todays
todays
print(if at_hotel.nil?
"No clue if he's in the hotel."
elsif at_hotel == true
"Definitely in."
elsif at_hotel == false
"He's out."
else
"The system is on the freeeeitz."
MIKAMAI
RUBY
ACADEMY
20
13

Elia Schito

Mikamai (@elia on twitter/github)

The **Ruby** Academy, 2013

1st part: **The Ruby**

I. Intro + Ruby Basics

II. Ruby

III. More Ruby!

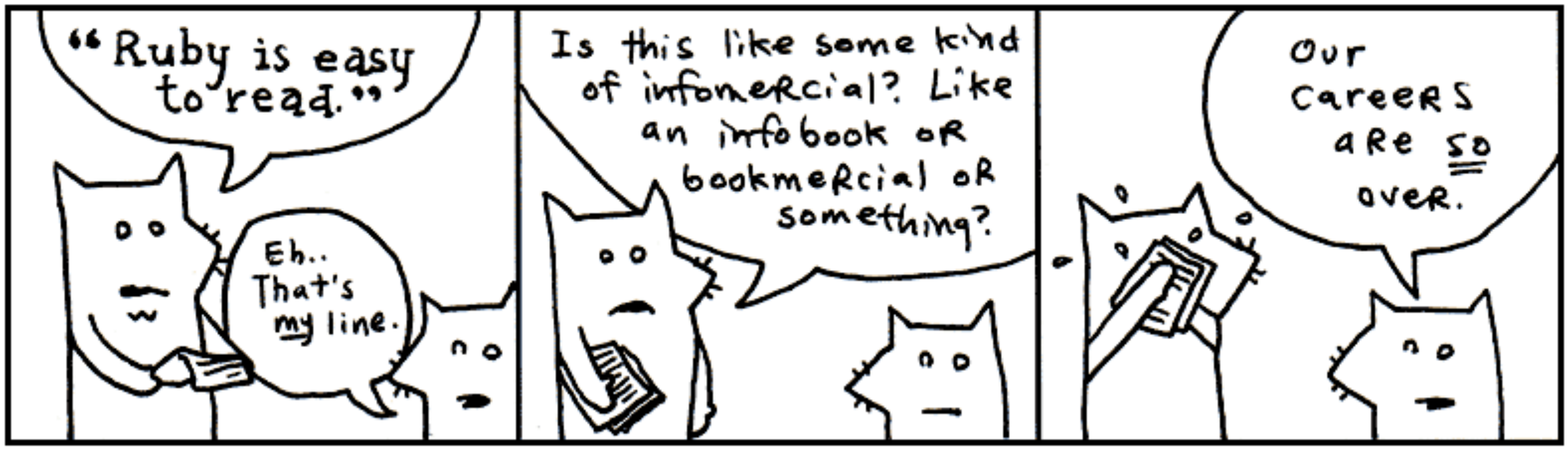
2nd part: **Tests & Rails**

IV. **Testing**: why & how

V. Rails

VI. More Rails

The **Ruby** ~~programming~~ Language


"Ruby is easy to read."
Eh.. That's my line.
Is this like some kind of infomercial? Like an infobook or bookmercial or something?
Our careers are so over.

Ruby is **Easy to Read.**

We can no longer truthfully call it a computer language.

It is coderspeak.

It is the language of our thoughts.

Examples

Read Aloud!

```
5.times { print "Odelay!" }


exit unless "restaurant".include? "aura"
```

```
['toast', 'cheese', 'wine'].each { |food|
  print food.capitalize
}
```

Read Aloud!

```
5.times { print "Odelay!" }
# Five times print “Odelay!”.

exit unless "restaurant".include? "aura"
# Exit unless the word restaurant includes the word aura.

['toast', 'cheese', 'wine'].each { |food|
  print food.capitalize
}
# With the words ‘toast’, ‘cheese’, and ‘wine’:
# take each food and print it capitalized.
```

Read Aloud!

```
5.times { print "Odelay!" }
# => Odelay!Odelay!Odelay!Odelay!Odelay!

exit unless "restaurant".include? "aura"
# => <program exits>

['toast', 'cheese', 'wine'].each { |food|
  print food.capitalize
}
# => ToastCheeseWine
```

Ruby is basically built from sentences.

Not exactly English, but short collections of words
and punctuation which encompass a single thought.

These sentences can form books, pages, entire novels.
Novels that can be read by humans & computers.

The parts of the **Speech**

Variables

Any plain, lowercase word is a Variable

```
x
y
banana2
phone_a_quail
```

```
not
234_starts_with_number
```

Use variables to nickname stuff

```
teddy_bear_fee = 121.08

total = orphan_fee + teddy_bear_fee + gratuity
```


Man, his examples aRe wieRd.
Sssh. $121.08 ain't a bad teddy bear fee.

Numbers

Integers: digits with options + or -

```
1
23
-10000

# also, to make big numbers readable:

population = 12_000_000_000
```

Decimals (Floats), Scientific, Hex, Bin

```
3.14

-808.08

12.043e-04

0xC1A0

0b0110010010010l
```

Strings

Double or Single “quotes”

```
"sealab"
'2021'
"These cartoons are hilarious!"

avril_quote = "I'm a lot wiser.  Now I know
what the business is like -- what you have
to do and how to work it."

print avril_quote
```

Decimals (Floats), Scientific, Hex, Bin

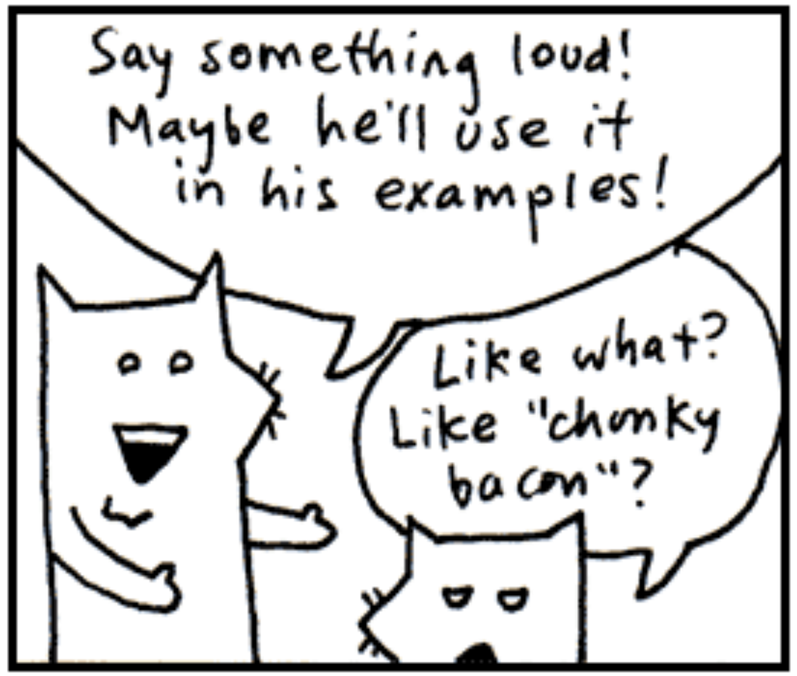
```
3.14

-808.08

12.043e-04

0xC1A0

0b0110010010010l
```


Say something loud! Maybe he'll use it in his examples!
Like what? Like "chunky bacon"?

Symbols

Symbols are lightweight strings

```
# use when need string that you won't print

:a
:b
:ponce_de_leon

# they're easier to digest for the computer,
# the colon indicates the bubbles
# trickling up from your computer's stomach
# as it digests the symbol
```


Chunky bacon!!
Chunky bacon!!

Constants

Constants: like variables, but capitalized

```
Time
Array
Bunny_Lake_is_Missing

# can't be changed after they're set

EmpireStateBuilding = "350 5th Avenue, NYC, NY"
```


Come on, seriously. Chunky bacon.
Chunky.
Bacon.

Methods

if variables and constants are the names, **methods are the verbs**

Methods: attached.with.a.dot

```
front_door.open
front_door.open.close

front_door.is_open?
front_door.shut!

# ".is_open?" could have been written like this:
Door.test_to_see_if_its_open
```

Method arguments

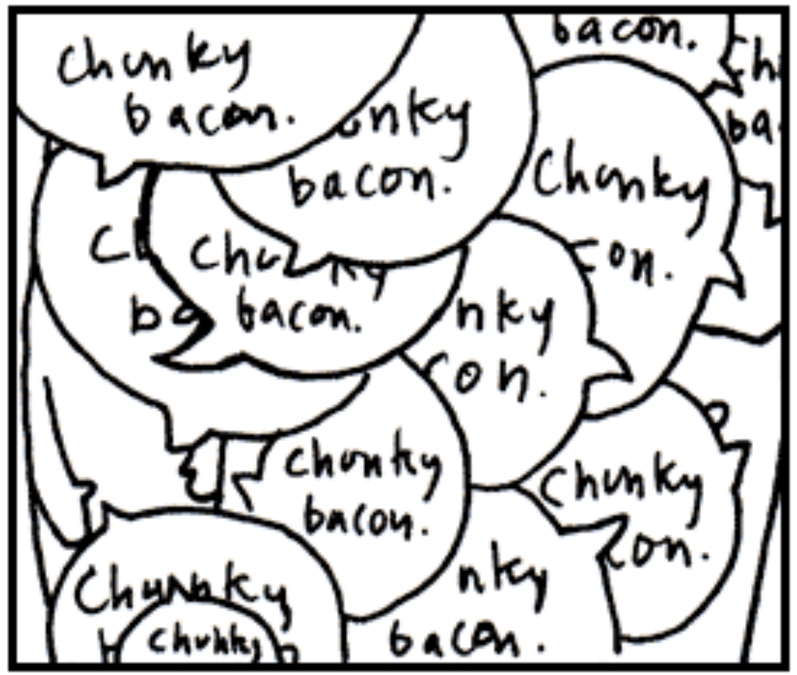
Method arguments are attached to the end of a method.

The arguments are **usually** surrounded by parentheses and separated by commas.

Methods: attached.with.a.dot

```
front_door.paint(3, :red)
front_door.paint(3, :red).dry(30).close

print "See, no dot."
```


chunky bacon.
bacon.
chunky bacon.
chunky bacon.
chunky bacon.
chunky bacon.
chunky bacon.
chunky bacon.
chunky bacon.

Global variables

Globals: the value of $$

```
$x
$1
$chunky
$CHunKY_bACOn
```

Instance variables

Instance vars: “@” stands for at-tribute

```
@x
@y
@only_the_chunkiest_cut_of_bacon_I_have_ever_seen
```

Blocks

Blocks: curly or do/end

```
2.times {
  print "Yes, I've used chunky bacon in my examples,
but never again!"
}

loop do
  print "Much better."
  print "Ah.  More space!"
  print "My back was killin' me in those crab pincers."
end
```

Block arguments: down the pipe!

```
|x|
|x,y|
|up, down, all_around|

{ |x,y| x + y }
```

Ceci n'est pas une pipe.

Ranges

Ranges: surrounded by () and separated by an ...

```
# is a range, representing the numbers 1 through 3.
(1..3)
```

```
# is a range, representing a lowercase alphabet.
('a'..'z')

# represents the numbers 0 through 4.
(0...5)
# the extra dot kicks off the last value in the range
```

Arrays

Ranges: surrounded by () and separated by an ...

```
[1, 2, 3]
# is an array of numbers.

['coat', 'mittens', 'snowboard']
# is an array of strings.
```

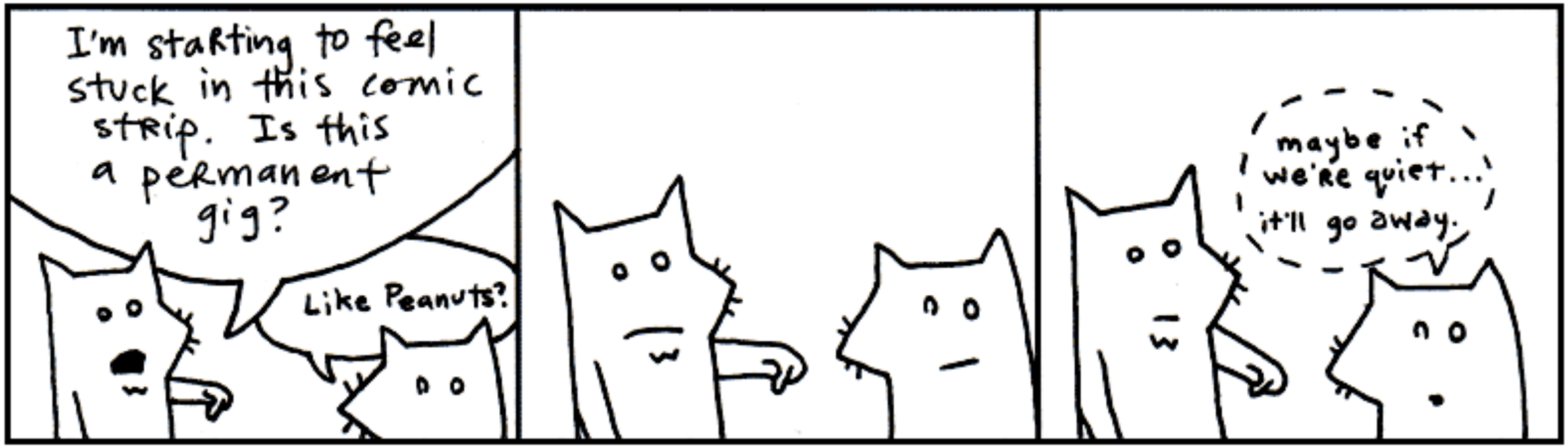
The caterpillar

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Hashes

Hash: curly braces outside, arrows inside

```
{'a' => 'aardvark', 'b' => 'badger'}
{:a => 'aardvark', :b => 'badger'}
{a: 'aardvark', b: 'badger'}
```


I'm starting to feel stuck in this comic strip. Is this a permanent gig?
Like Peanuts?
maybe if we're quiet... it'll go away.

Regular Expressions

Regexp: surrounded by /slashes/

```
/ruby/                      # "hey ruby boy!"  => "ruby"
/[0-9]+/                    # "during 1910"    => "1910"
/^\d{3}-\d{3}-\d{4}/  # "123-456-7890 it's the number"
=> "123-456-7890"
```

Operators

You know, addition, and subtraction, and so on

```
**  !   ~   *   /   %   +   -   &
<<  >>  |   ^   >   >=  <   <=  <=>
||  !=  =~  !~  &&  +=  -=  ==  ===
..  ...  not  and  or
```


Woohoo! I don't know what chunky bacon is, but we did it!
We're in the book!

Keywords

You know, addition, and subtraction, and so on

```
alias   and     BEGIN   begin   break   case    class   def     defined
do      else    elsif   END     end     ensure  false   for     if
in      module  next    nil     not     or      redo    rescue  retry
return  self    super   then    true    undef   unless  until   when
while   yield
```

So, yes. **You've kept up nicely.**
But now I need to start seeing good marks from you.

Read Aloud!

```
5.times { print "Odelay!" }

exit unless "restaurant".include? "aura"

['toast', 'cheese', 'wine'].each { |food|
  print food.capitalize
}
```

An Example to Help You Grow Up and know the real world ☺

Read Aloud!

```
require 'net/http'
Net::HTTP.start('www.ruby-lang.org', 80) do |http|
  print( http.get('/en/LICENSE.txt').body )
end
```

thanks!