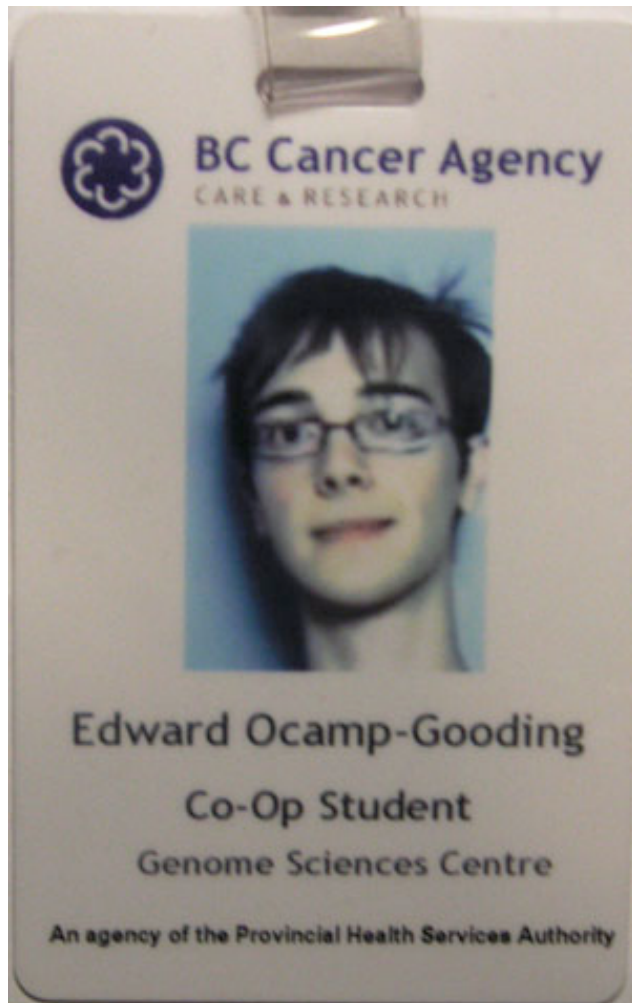


5.0.0.1.1

Hello Ruby A scrumptiously delicious intro

- What you shouldn't have to do
- Things you should want
- Why Ruby?
- What's it look like?
- Why should I care?
- When not to use it





Hi, I'm eduardo@bcgsc.ca

I am not a Perl programmer.

Things I don't like to do when programming, but find myself sometimes doing anyway:

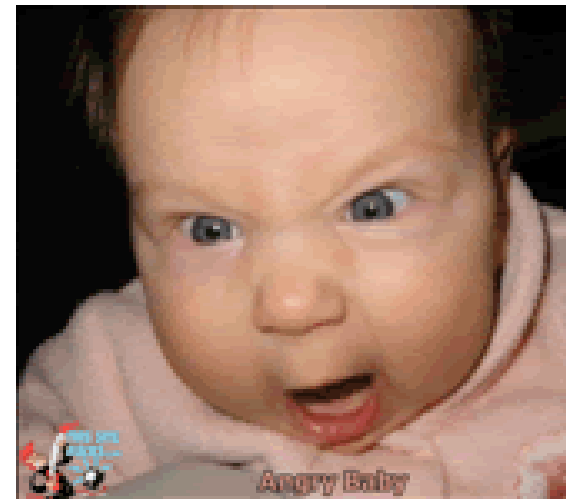
Spoon-feeding compilers

Sifting through stack trace vomit

Sifting through other people's less-than-elegant code, searching for what should be obvious

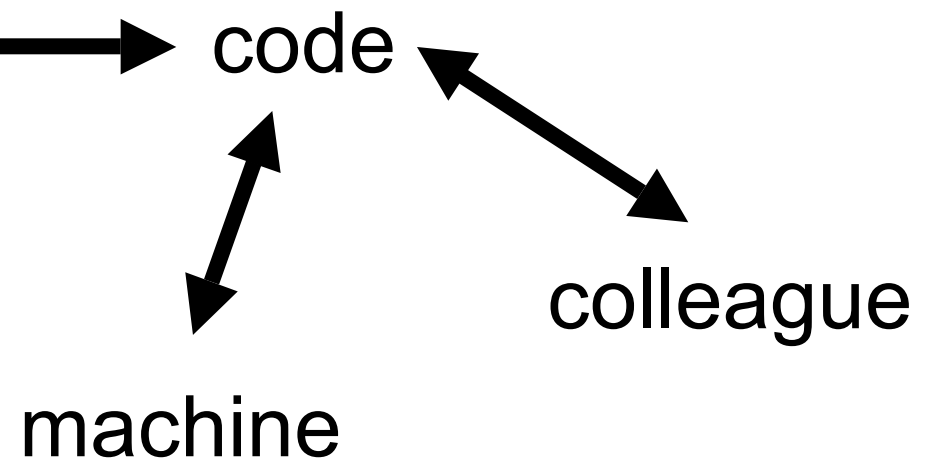
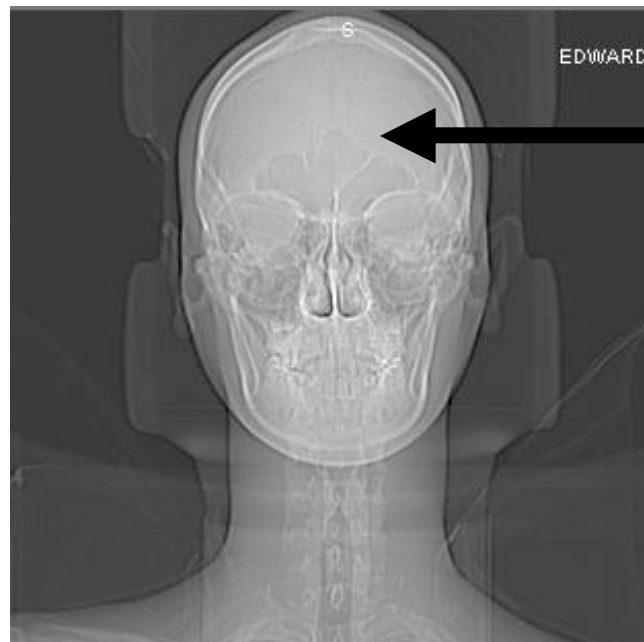
Typing like crazy to do something trivial

I don't want to be surprised anymore



Things I want out of a programming experience:

- Minimize time it takes to get **into** my head
- Minimize time it takes to get **out** of my head
- Minimize frustration -> minimize effort -> conventional things are easy



Most important thing I want when programming:

- Happiness



Why Ruby?

Easy to read

Easy to learn

Great for string processing and system call...calling

Full toolbox (CPAN-like package manager, built-in debugger, built-in interactive session, batteries included)

Designed with programmer joy in mind



What's it look like?

```

1
2 # This is a delicious sorting program
3
4 food_list = ['gelato', 'chicken', 'garlic',
5             'beans', 'lobster']
6
7 puts "I'm thinking of #{food_list.size} foods."
8
9 if food_list.include?("chicken")
10   feels_like_chicken_tonight = true
11 elsif food_list.type == "Array"
12   food_list << 'broccoli'
13 else
14   puts "There's no chicken."
15 end
16
17 # Sortissimo!
18 puts food_list.sort

```

Perl equivalent

```

@food_list = (qw(gelato chicken garlic beans lobster));

print qq(I'm thinking of $#food_list foods.);

if (grep($_ eq "chicken", @food_list)) {
  $feels_like_chicken_tonight = 1;
} elsif (@food_list) {
  # we know food_list is an array
  push @food_list, "broccoli";
} else {
  print "There's no chicken.";
}

print sort @food_list;

```

Let's write a simple function

```

20 def sort_by_length(list)
21   for i in (1..list.size).to_a.reverse do
22     for j in (0...(i - 1)) do
23       if list[j].length > list[j + 1].length
24         # Swap elements
25         temp_element = list[j]
26         list[j] = list[j + 1]
27         list[j + 1] = temp_element
28       end
29     end
30   end
31   return list
32 end
33
34 puts "ok! sort it\n"
35 puts sort_by_length(food_list)
36

```

Perl equivalent

```

sub sort_by_length {
  @list = @_;
  for $i reverse(0 .. @list-1) {
    for $j (0 .. $i-1) {
      if(length($list[$j]) > length($list[$j+1])) {
        @list[$j,$j+1] = @list[$j+1,$j];
      }
    }
  }
  return @list;
}

```

A Ruby approach

```

36
37 # An easier way of writing it
38 def sort_by_length(list)
39   list.sort do |first_element, second_element|
40     first_element.length <=> second_element.length
41   end
42 end
43
44 puts "Sorting again!"
45 puts sort_by_length(food_list)
46

```

Perl equivalent

```

sub sort_by_length {
  sort {length($a) < length($b)} @_;
}

```

Ruby's iterator

```

list do |x|
  CODE

```

is like Perl's map

```

map { CODE } @list

```

except Ruby defines the local iterator explicitly (`|x|`, `!var|`, etc) whereas Perl uses `$_`

What can it do?

```

48
49 # Putting sorted results in a file
50 output_file = File.new("sorted_food_list.txt", "w")
51 sorted_food_list = food_list.sort
52
53 for food in sorted_food_list
54   output_file.puts food
55 end
56
57 output_file.close
58
59 require 'open-uri'
60 bcgsc = open("http://www.bcgsc.ca")
61 page = bcgsc.read
62 page.include?("Gene") # => true
63 page.index("Gene")   # => 3606
64

```

Perl equivalent

```

$fh = open(">sorted_food_list.txt");
for $food (sort @food_list) {
  print $fh $food;
}
close($fh);

```

```

use LWP::Simple;
my $html = get("http://www.bcgsc.ca");
if($html =~ /Gene/) {
  print index($html,"Gene");
}

```

Why should I care?

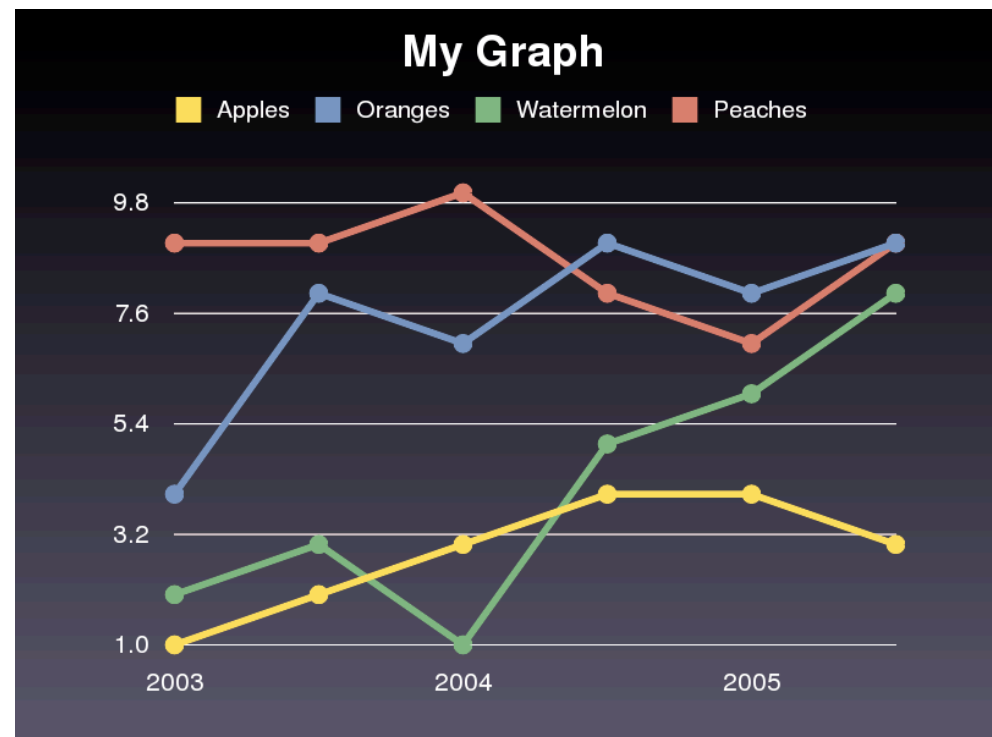
People are irrational and the world outside changes: dealing with requirement changes and getting help if you need it is easy.

Pretty graphs are nice (and easy)

```

1  #!/usr/bin/ruby
2
3  require 'rubygems'
4  require 'gruff'
5
6  g = Gruff::Line.new
7  g.title = "My Graph"
8
9  g.data("Apples", [1, 2, 3, 4, 4, 3])
10 g.data("Oranges", [4, 8, 7, 9, 8, 9])
11 g.data("Watermelon", [2, 3, 1, 5, 6, 8])
12 g.data("Peaches", [9, 9, 10, 8, 7, 9])
13
14 g.labels = {0 => '2004', 2 => '2005', 4 => '2006'}
15
16 g.write('my_fruity_graph.png')

```



Hitting the fan (debugging)

\$ ruby -rdebug broken_program.rb

```
edward-ocampo-goodings-computer:~/Tutorials/Ruby edward$ ruby -rdebug food_sort_2.rb
Debug.rb
Emacs support available.
```

```
food_sort_2.rb:4:food_list = ['gelato', 'chicken', 'garlic',
(rdb:1) list 0-20
[0, 20] in food_sort_2.rb
```

```
1
2 # This is a delicious sorting program
3
=> 4 food_list = ['gelato', 'chicken', 'garlic',
5             'beans', 'lobster']
6
7 puts "I'm thinking of #{food_list.size} foods."
8
9 if food_list.include?("chicken")
10   feels_like_chicken_tonight = true
11 elsif food_list.type == "Array"
12   food_list << 'broccoli'
13 else
14   puts "There's no chicken."
15 end
16
17 # Sort them by their length, using the worst sort ever.
18
19 def sort_by_length(list)
20   for i in (1..list.size).to_a.reverse do
(rdb:1) b 19
Set breakpoint 1 at food_sort_2.rb:19
```

```
(rdb:1) v l
bcgsc => nil
feels_like_chicken_tonight => nil
food => nil
food_list => nil
output_file => nil
page => nil
sorted_food_list => nil
(rdb:1) []
```

When not to use Ruby

Machine speed (write in C and call it from Ruby)

Sadomasichsm

Same programming time, same programming room

Code jams and an open problem