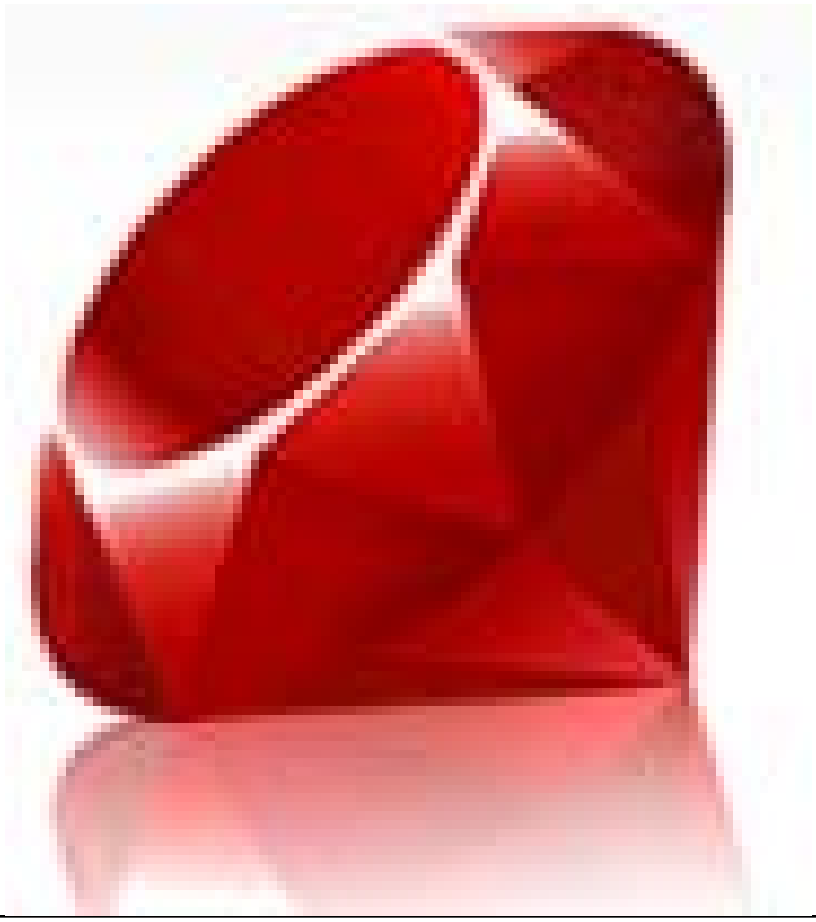
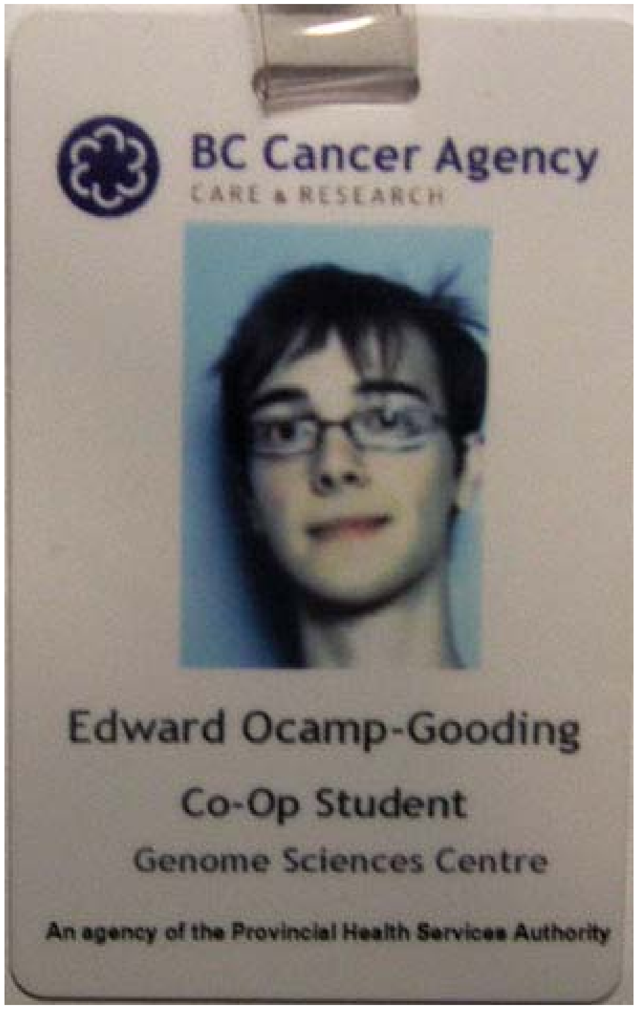
5.0.0.1.1

Hello Ruby
A scrumptiously delicious intro



- Why Ruby?
- What's it look like?
- So how is it different than Perl?
- Make it go!
- dealing with variables



Hi, I'm edwardo@bcgsc.ca

I am not a Perl programmer.

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

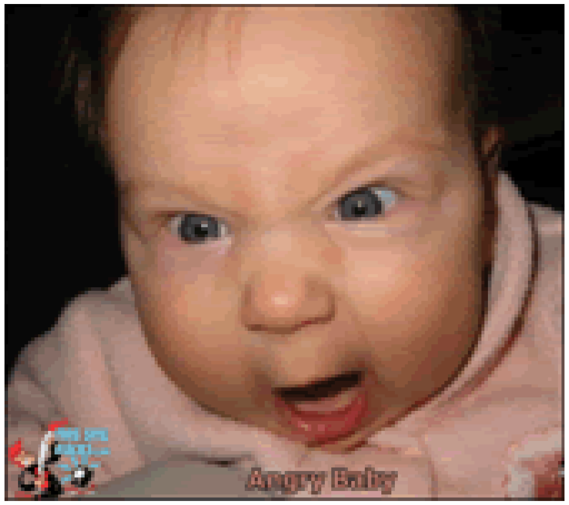
Spoon-feeding compilers

Sifting through stack trace vomit

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

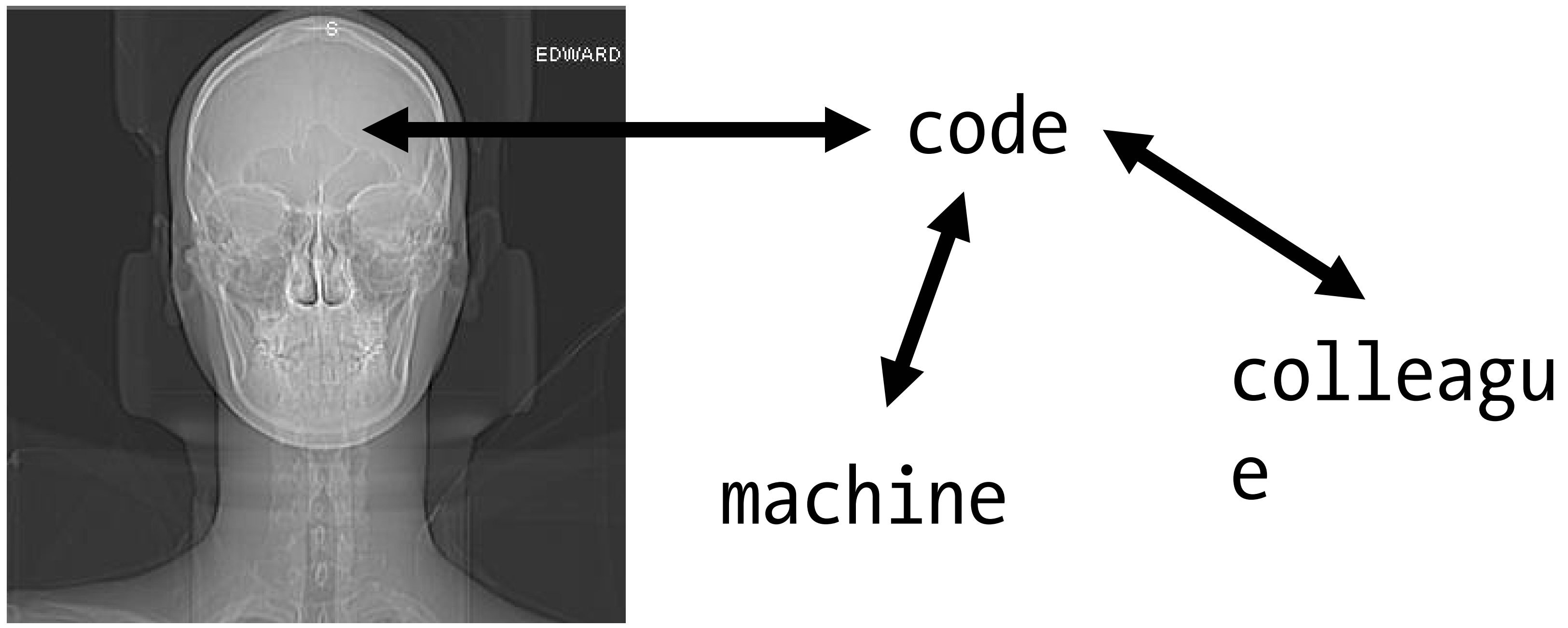
Typing like crazy to do something trivial

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

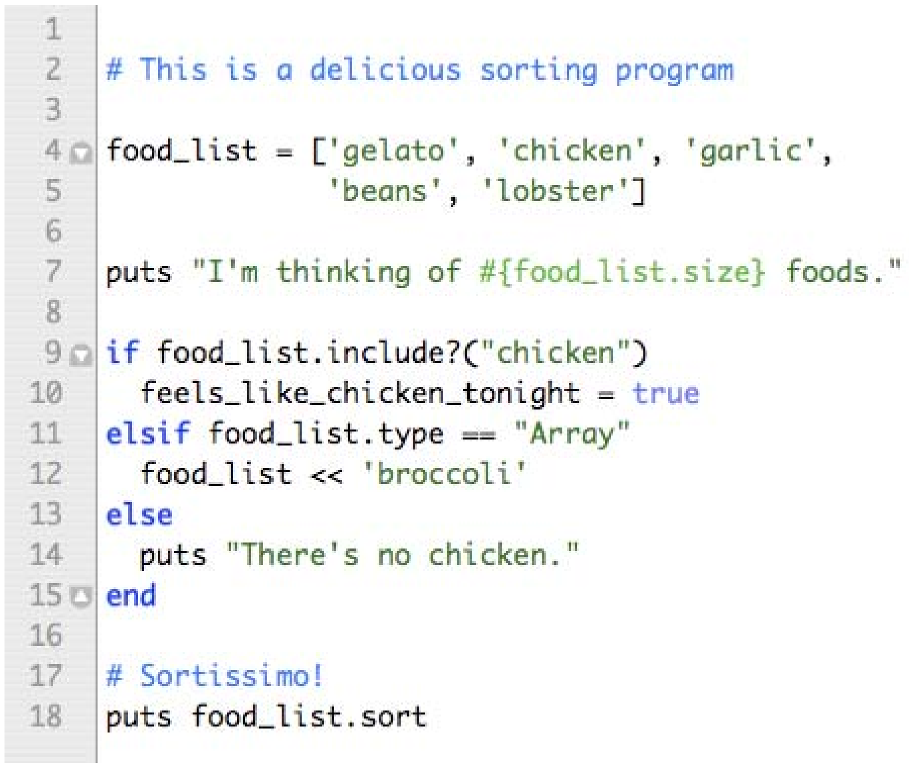
Easy to learn

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

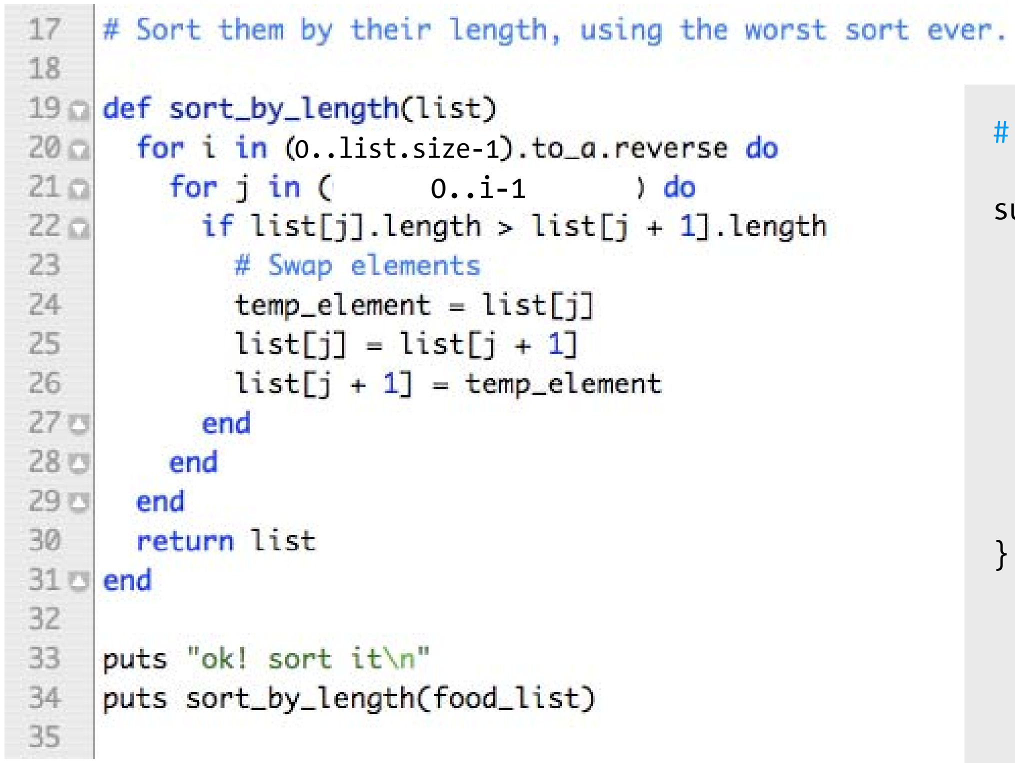
What’s it look like?



```

# This is a delicious sorting program

food_list = ['gelato', 'chicken', 'garlic',
             'beans', 'lobster']

puts "I'm thinking of #{food_list.size} foods."

if food_list.include?("chicken")
  feels_like_chicken_tonight = true
elsif food_list.type == "Array"
  food_list << 'broccoli'
else
  puts "There's no chicken."
end

# Sortissimo!
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)) {
  $feels_like_chicken_tonight = 1;
} elsif (@food_list) {
  # we know food_list is an array
  # can test references using ref($x) eq "ARRAY"
  push @food_list, “broccoli”;
} else {
  print “There’s no chicken.”;
}

print sort @food_list;
```

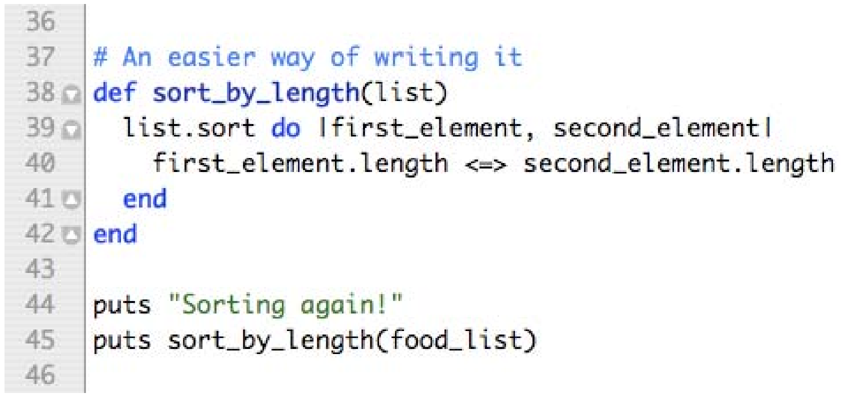
Let’s write a simple function



```ruby
# Sort them by their length, using the worst sort ever.

def sort_by_length(list)
  for i in (0..list.size-1).to_a.reverse do
    for j in (      0..i-1       ) do
      if list[j].length > list[j + 1].length
        # Swap elements
        temp_element = list[j]
        list[j] = list[j + 1]
        list[j + 1] = temp_element
      end
    end
  end
  return list
end

puts "ok! sort it\n"
puts sort_by_length(food_list)
```

```perl
# 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;
}
```

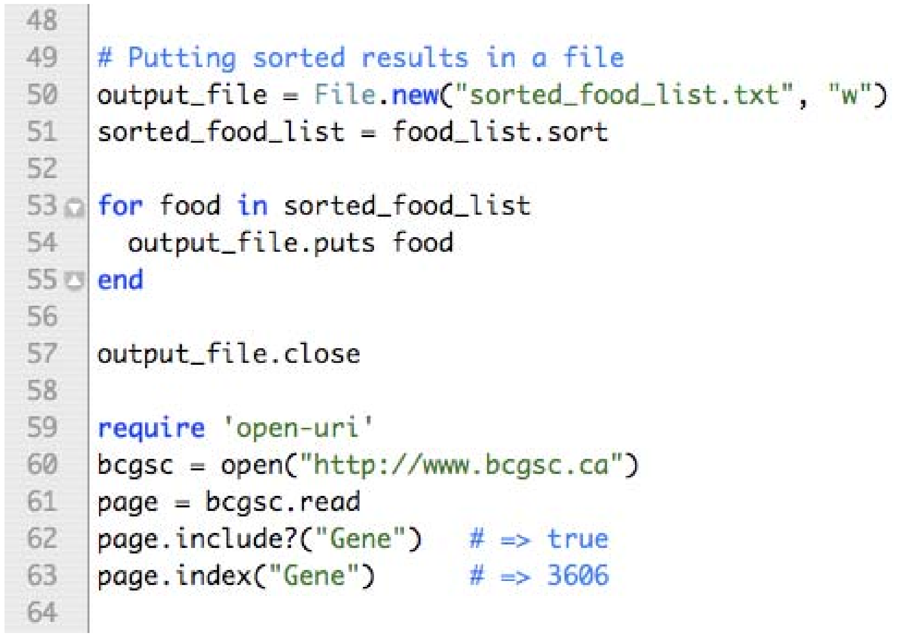
A Ruby approach



```
# An easier way of writing it
def sort_by_length(list)
  list.sort do |first_element, second_element|
    first_element.length <=> second_element.length
  end
end

puts "Sorting again!"
puts sort_by_length(food_list)
```

```
# 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?



```ruby

# Putting sorted results in a file
output_file = File.new("sorted_food_list.txt", "w")
sorted_food_list = food_list.sort

for food in sorted_food_list
  output_file.puts food
end

output_file.close

require 'open-uri'
bcgsc = open("http://www.bcgsc.ca")
page = bcgsc.read
page.include?("Gene")    # => true
page.index("Gene")       # => 3606

```

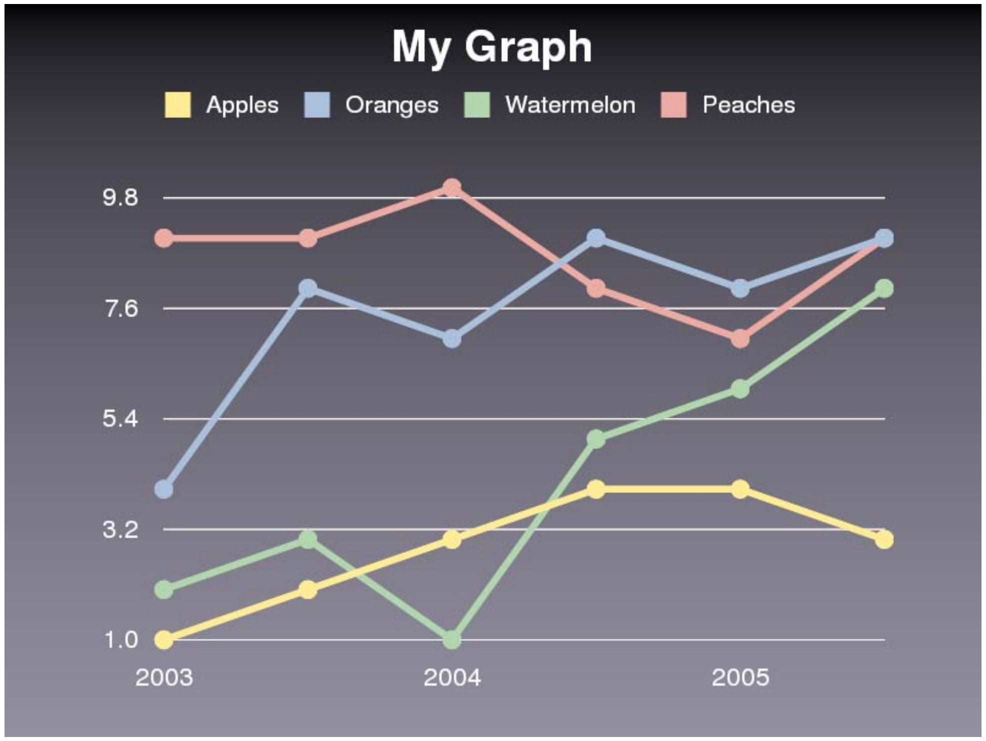
```perl
# Perl equivalent

$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.

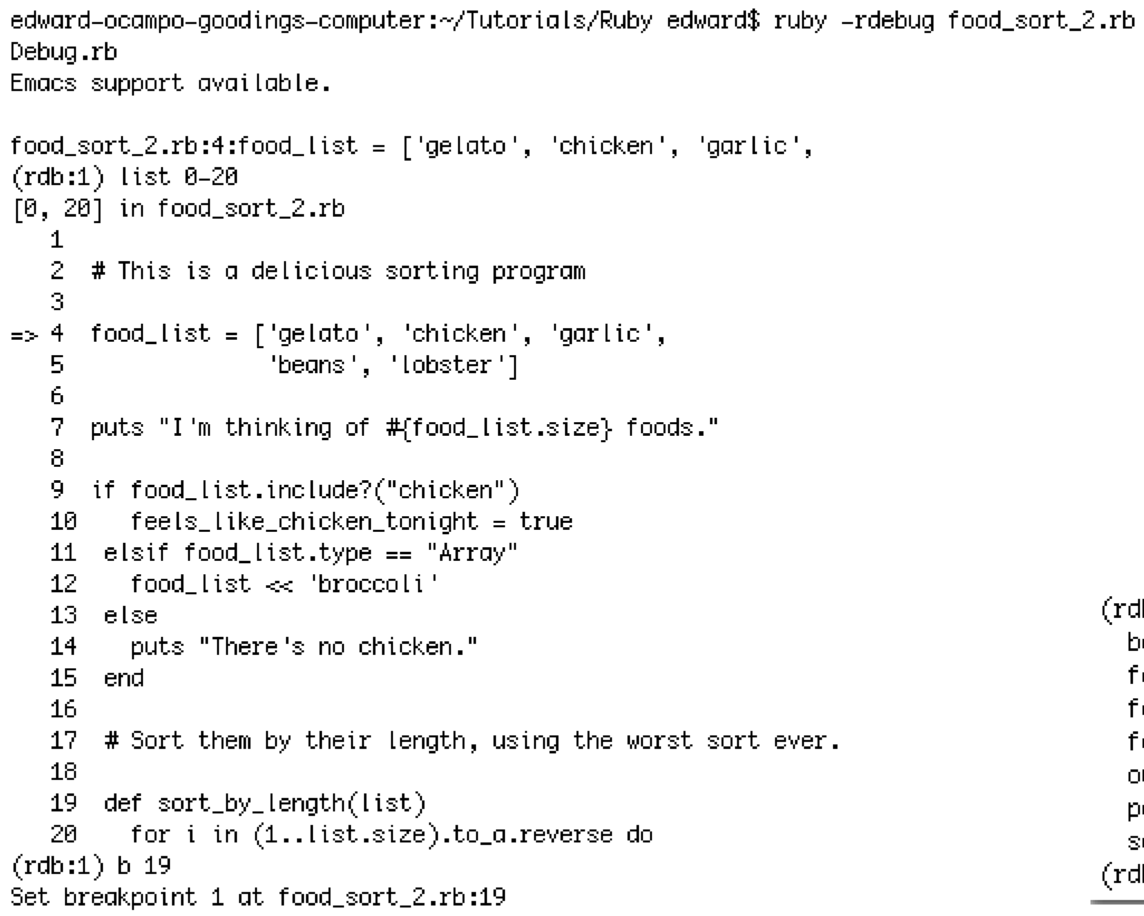
Pretty graphs are nice (and easy)

```ruby
#!/usr/bin/ruby

require 'rubygems'
require 'gruff'

g = Gruff::Line.new
g.title = "My Graph"

g.data("Apples", [1, 2, 3, 4, 4, 3])
g.data("Oranges", [4, 8, 7, 9, 8, 9])
g.data("Watermelon", [2, 3, 1, 5, 6, 8])
g.data("Peaches", [9, 9, 10, 8, 7, 9])

g.labels = {0 => '2004', 2 => '2005', 4 => '2006'}

g.write('my_fruity_graph.png')
```

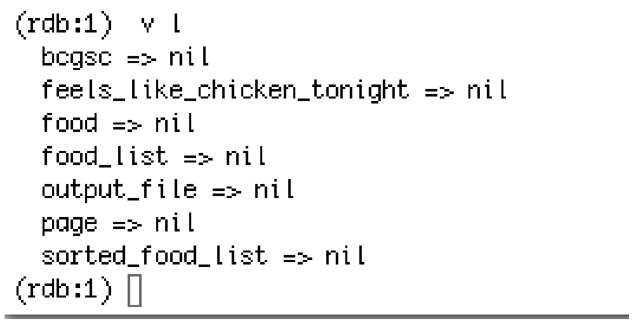


Hitting the fan (debugging)

$ 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)

Sadomasochism

Same programming time, same programming room

Code jams and an open problem