Creator

Yukihiro Matsumoto “Matz”

Influences

Lisp, Smalltalk, Perl

Trade-offs

Simplicity for Safety
Productivity for Performance

irb

Ruby's Interactive Console

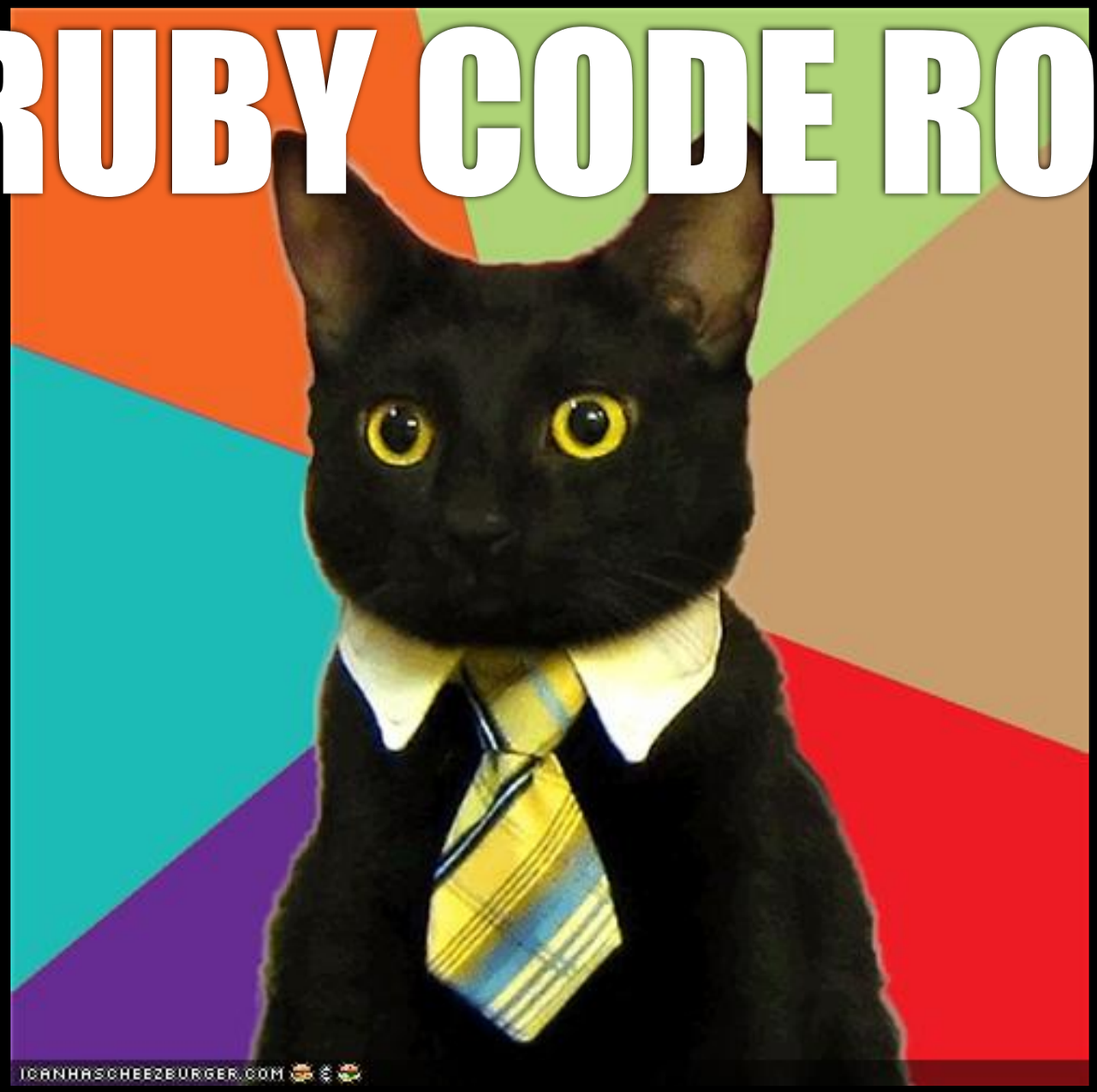
Syntax

WHEN U DECLARE VARS

U DECLARE WEAKNESS

Courage Wolf

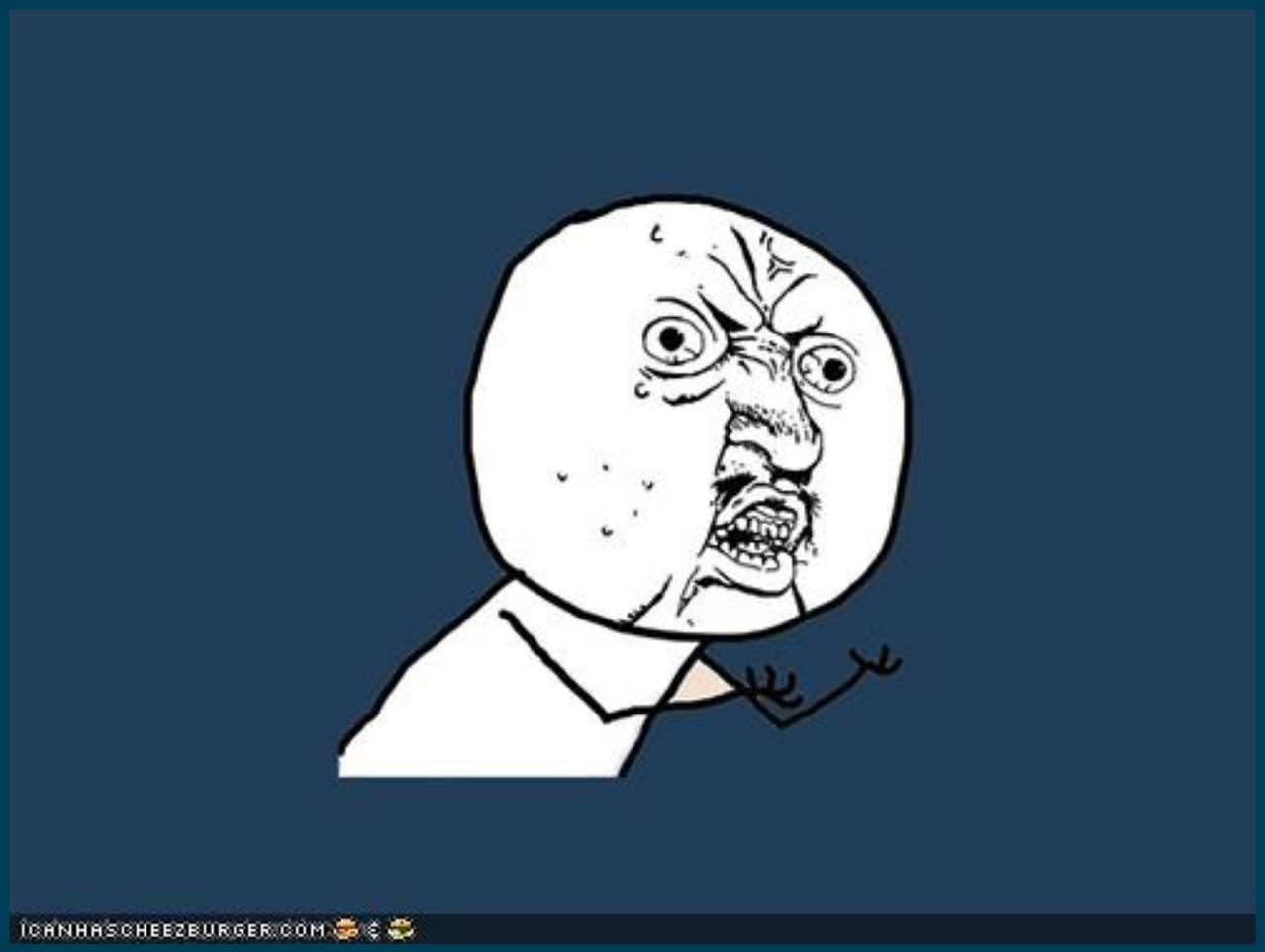
RUBY CODE ROI



ALWAYS RETURN SOMETHING

Business Cat

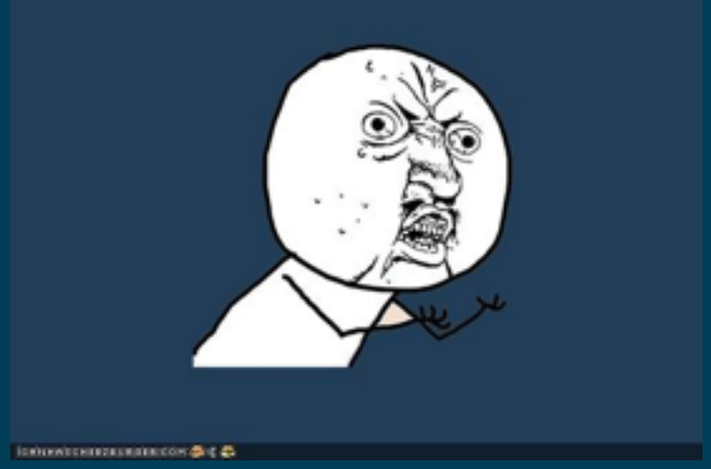
Y U NO EVALUTE STRING?



SINGLE QUOTES!!

```
puts “hello, #{language}”
```

Y U NO?



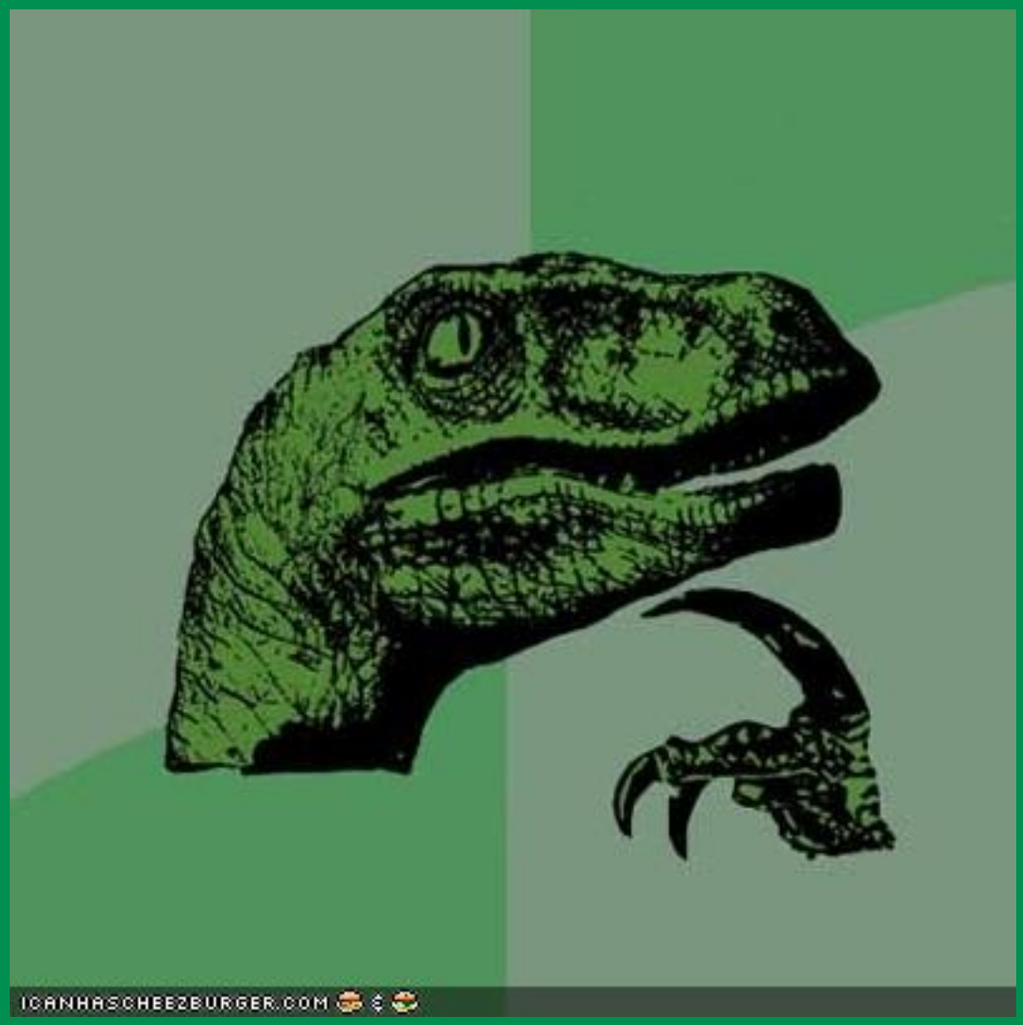
```
>> puts ‘literal string’
literal string

>> subject = ‘world’
=> “world”

>> puts “hello, #{subject}”
hello, world
```

Y U NO?

HUMANS ARE CARBON



RUBY IS OBJECTS

Philosoraptor

```
>> 4.class
=> FixNum

>> 4.methods
=> [“inspect”, “%”, “<<“, ...

>> false.class
=> FalseClass
```

Philosoraptor

MY CONDITIONALS

ALWAYS SUCCESSFUL

Success Kid

```
>> x = 4
=> 4

>> puts ‘True!!’ if x == 4
True!!
=> nil

>> puts ‘True!!’ unless x == 4
=> nil

>> puts ‘True!!’ if not true
=> nil

>> puts ‘True!!’ if !true
=> nil
```

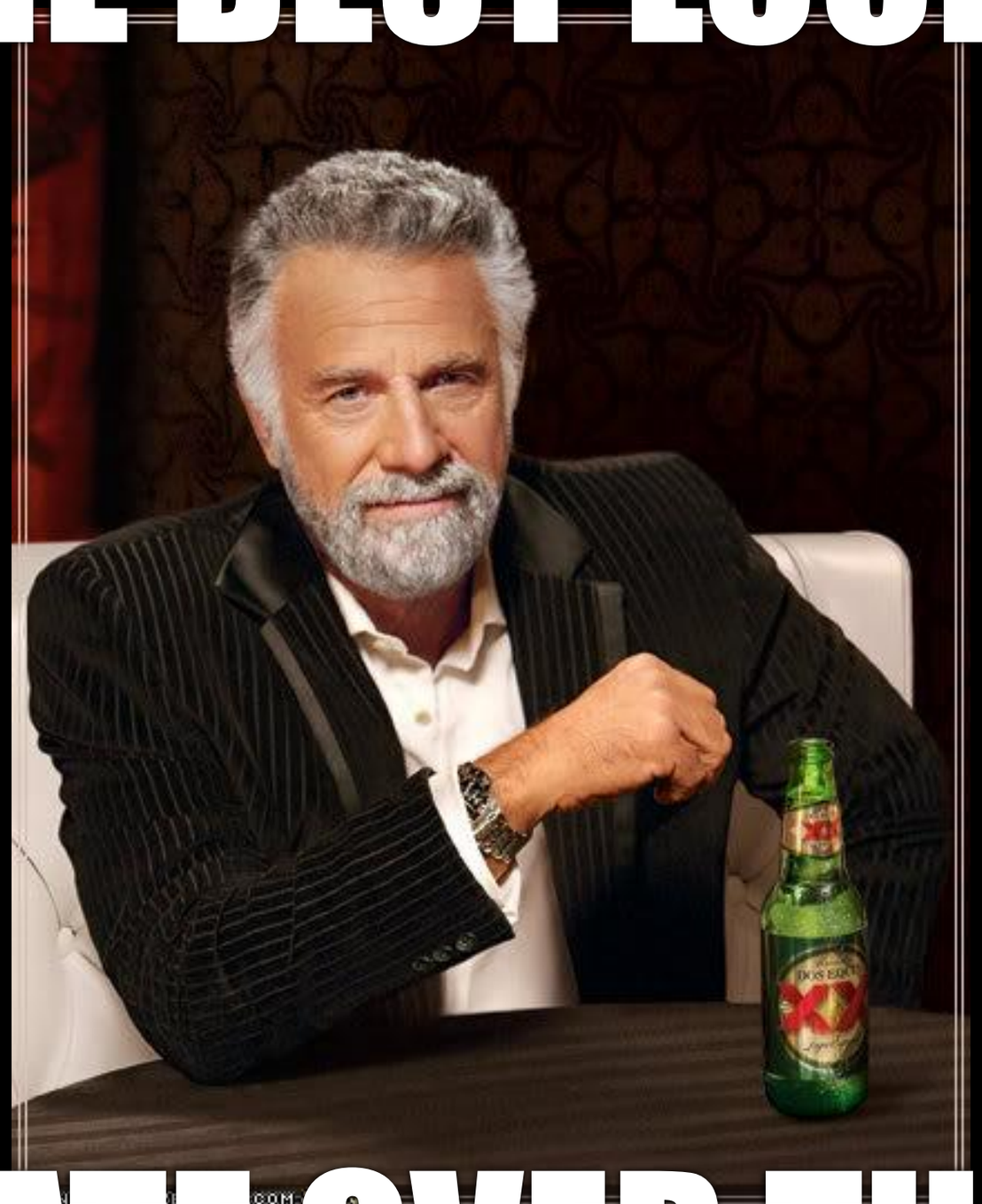
Success Kid

```
# Everything but nil and false
# evaluate to true. 0 is true!
>> puts “This is true” if 0
This is true
=> nil

# and, &&
# or, ||
# &, | are the non-short circuit
# equivalents
```

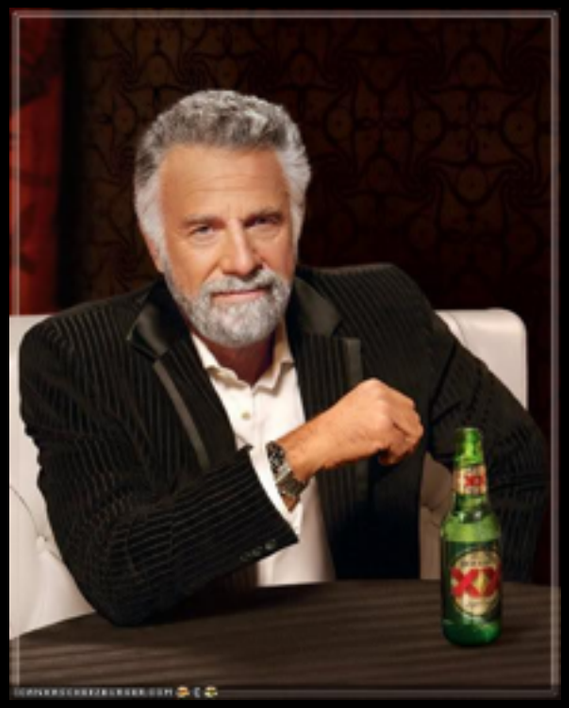
Success Kid

THE BEST LOOPS



ITERATE OVER THIRST

The Most Interesting Man in the World

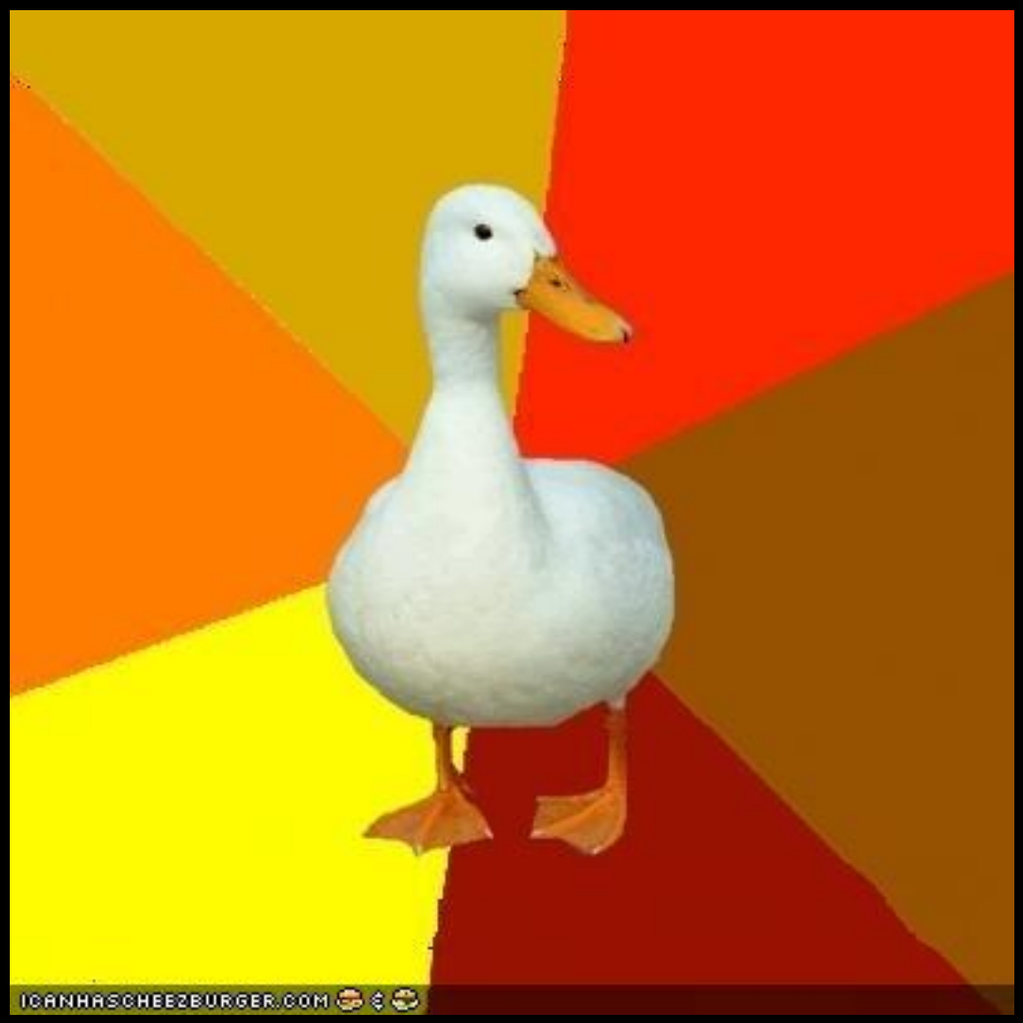


```
>> x = x + 1 while x < 10
=> nil
>> x
=> 10

>> x = x - 1 until x == 1
=> nil
>> x
=> 1
```

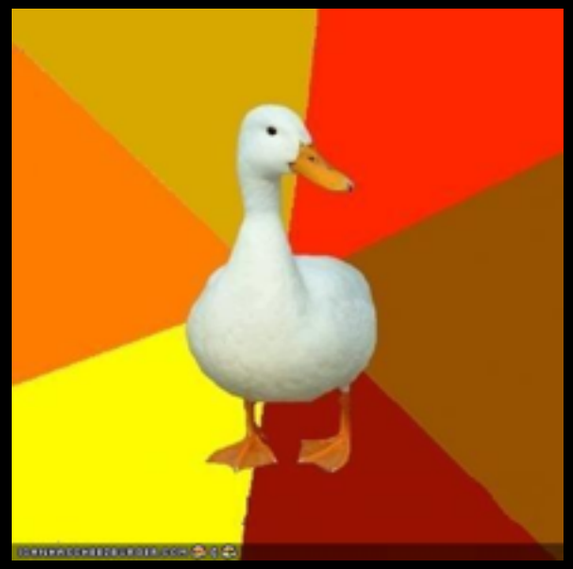
The Most Interesting Man in the World

KEYBOARD CAT HAS NOTHING ON



DUCK TYPING

Technologically Impaired Duck



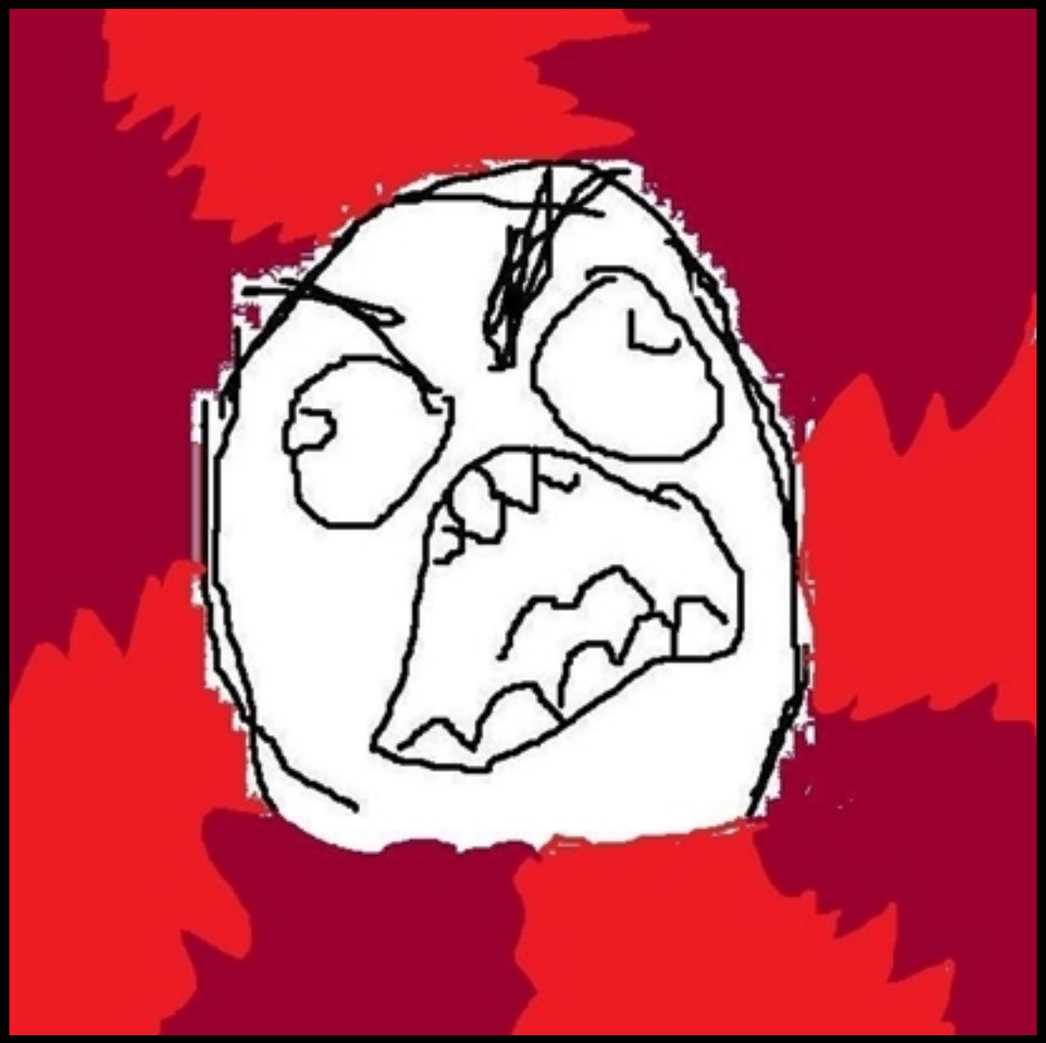
```
>> 4 + ‘four’
TypeError: String can’t be coerced
into Fixnum

# Strongly typed
# Dynamic: Checked at run time

>> a = [‘100’, 100.0]
=> [‘100’, 100.0]
>> while i < 2
>>    puts a[i].to_i
>>    i += 1
>> end
100
100
```

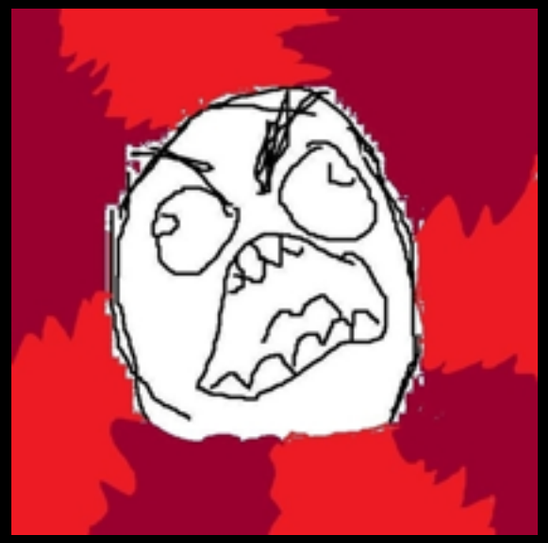
Technologically Impaired Duck

FFFFFUUUUUUUUU



UUUUUUUNCTIONS

FFFUUUUUU



```
>> def tell_the_truth
>>     true
>> end

# Last expression is return value
```

Technologically Impaired Duck

ARRRRRRRAYS

AND HASHES


```
>> animals = [‘lions’, ‘tigers’]
=> [‘lions’, ‘tigers’]

>> numbers = {1 => ‘one’, 2 => ‘two’}
=> {1=>”one”, 2=>”two”}

# Symbols
>> ‘string’.object_id
=> 3092010
>> ‘string’.object_id
=> 3089690
>> :string.object_id
=> 69618
>> :string.object_id
=> 69618
```


```
>> def winning(options = {})
>>   if(options[:profession] == :gambler)
>>     true
>>   else
>>     false
>>   end
>> end
=> nil

>> winning
=> false

# {} optional for last parameter
>> winning(:profession => :lawyer)
=> true
```

YO DAWG I HEARD YOU LIKED CODE BLOCKS

SO YOU COULD RUN CODE IN YOUR CODE

```
>> 3.times { puts ‘hi’ }
hi
hi
hi

>> animals = [‘lions’, ‘tigers’]
>> animals.each {|a| puts a}
lions
tigers

# Blocks can be passed as
parameters
>> def pass_block(&block)
>> end
>> pass_block { puts ‘hi’ }
```

Classes

```
class MyClass
  def initialize(name)
    @name = name # instance var
    @@other = ‘’ # class var
  end

  def name
    return @name
  end

  # methods that check end in ?
end

my_class = MyClass.new(‘Name’)
my_class.name # returns ‘Name’
```

Modules

```
module MyModule
  def name
    return @name
  end
end

class MyClass
  include MyModule

  def initialize(name)
    @name = name
  end
end

my_class = MyClass.new(‘Name’)
my_class.name # returns ‘Name’
```

Enumerable

```
# Implements each method
```

Comparable

```
# Implements <=> (spaceship) method
```

Open Classes

```
# First invocation defines
# Second invocation modifies

class NilClass
  def blank?
    true
  end
end

class String
  def blank?
    self.size == 0
  end
end

['', 'person', nil].each {|a| puts a unless a.blank?}
# outputs person
```

method_missing