CF:G 2

Welcome back!

#omg wt lang nxt

- C
- C++
- Objective-C
- Java
- Swift
- Python
- Ruby
- Haskell
- C#
- Erlang
- Go
- F#
- D
- lolcat
- etc…

VS.

“Whether you are more of a Ruby or Python programmer depends on your personality.

If your favourite toy at a young age was Lego, choose Python. If it was clay, use Ruby.”

—Mattias Petter Johansson

But—

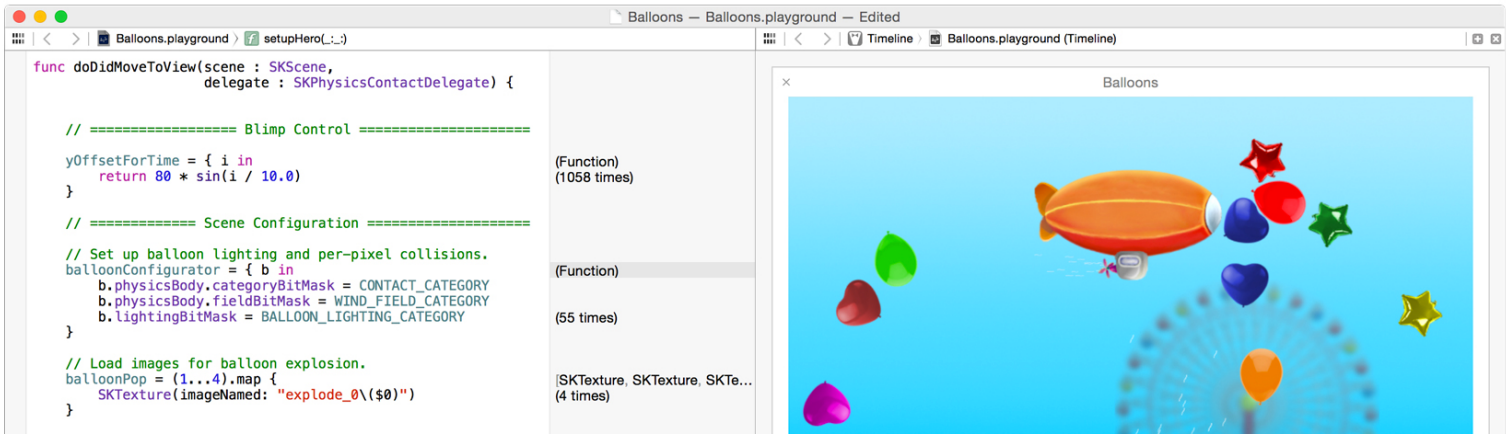
- It doesn't really matter, because…
- Once you know programming, picking up a new language is fairly easy.
- For instance…

Swift

A new programming language for iOS and OS X.

Introducing Swift

Swift is an innovative new programming language for Cocoa and Cocoa Touch. Writing code is interactive and fun, the syntax is concise yet expressive, and apps run lightning-fast. Swift is ready for your next iOS and OS X project — or for addition into your current app — because Swift code works side-by-side with Objective-C.



- Ruby can be run interactively via the ‘irb’ command
- Mac: open Terminal, type **irb**, hit enter
- Windows: (http://rubyinstaller.org/) then open **Interactive Ruby** from the Ruby section of your Start Menu.

irb 101 ftw amirite

1. Type “Hello World” and hit enter

Ruby obeys you!

```
2. puts “Hello World”
```

`puts` is the basic command to print something out in Ruby. But then what’s the `=> nil` bit? That’s the result of the expression. `puts` always returns nil, which is Ruby’s absolutely-positively-nothing value.

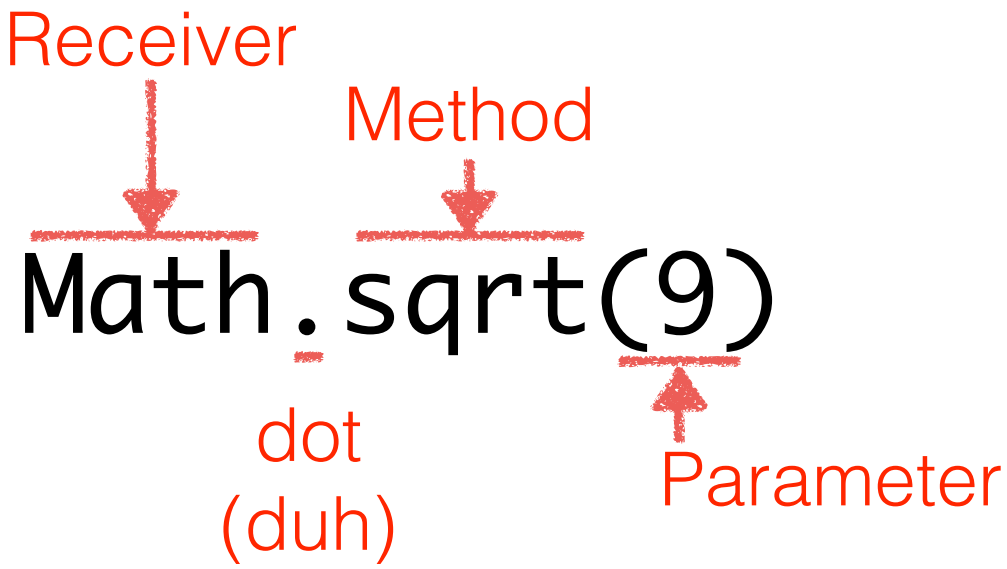
Your free calculator is here

```
3. 3+2
4. 3*2
5. 3**2
6. Math.sqrt(9)
```

Whoa.

Modules group code by topic.
Math also has things like **sin()** and **tan()**

Message in a dot call



Write this down, you will see it a lot.

Variables are to programming as fish are to water, or something.

```
irb> a = 3 ** 2
=> 9
irb> b = 4 ** 2
=> 16
irb> Math.sqrt(a+b)
=> 5.0
```

Um, ok… nice calculator?

Put a function into a variable

```
def h
puts “Hello world!”
end

=> :h

h
=> “Hello world!”
h()
=> “Hello world!”
```

More responsive!

```
def h(name)
puts "Hello #{name}!"
end
=> :h
h "Emma"
Hello Emma!
=> nil
```

Even more responsive!

```
def h(name = "World")
puts "Hello #{name.capitalize}!"
end
=> :h
h "alex"
Hello Alex!
=> nil
h
Hello World!
=> nil
```

Greet Scott!

```ruby
class Greeter
    def initialize(name = "World")
        @name = name
    end
    def say_hi
        puts "Hi #{@name}!"
    end
    def say_bye
        puts "Bye #{@name}, come back soon."
    end
end

```

Greeter or lesson

```
irb> g = Greeter.new("Poppy")
=> #<Greeter:0x16cac @name="Poppy">
irb> g.say_hi
Hi Poppy!
=> nil
irb> g.say_bye
Bye Poppy, come back soon.
=> nil
```

No use for @name

```
irb> g.@name
SyntaxError: compile error
(irb):52: syntax error
        from (irb):52
```

What's inside?

```
irb> Greeter.instance_methods
```

```
=> ["method", "send", "object_id", "singleton_methods",
    "__send__", "equal?", "taint", "frozen?",
    "instance_variable_get", "kind_of?", "to_a",
    "instance_eval", "type", "protected_methods", "extend",
    "eql?", "display", "instance_variable_set", "hash",
    "is_a?", "to_s", "class", "tainted?", "private_methods",
    "untaint", "say_hi", "id", "inspect", "==", "===",
    "clone", "public_methods", "respond_to?", "freeze",
    "say_bye", "__id__", "=~", "methods", "nil?", "dup",
    "instance_variables", "instance_of?"]
```

Eek.

```
irb> Greeter.instance_methods(false)
```

```
=> ["say_bye", "say_hi"]
```

Aah.

```
irb> g.respond_to?("name")
=> false
irb> g.respond_to?("say_hi")
=> true
irb> g.respond_to?("to_s")
=> true
```

Never too late

```
irb> class Greeter
irb>   attr_accessor :name
irb> end
=> nil
```

```
irb(main):047:0> g = Greeter.new("Andy")
=> #<Greeter:0x3c9b0 @name="Andy">
irb(main):048:0> g.respond_to?("name")
=> true
irb(main):049:0> g.respond_to?("name=")
=> true
irb(main):050:0> g.say_hi
Hi Andy!
=> nil
irb(main):051:0> g.name="Betty"
=> "Betty"
irb(main):052:0> g
=> #<Greeter:0x3c9b0 @name="Betty">
irb(main):053:0> g.name
=> "Betty"
irb(main):054:0> g.say_hi
Hi Betty!
=> nil

```

MegaGreeter 2000™

- Switch to a file.
  - Advantage: you can save it
  - Disadvantage: you won't see the results live
- You can use something like **LightTable** for seeing results as you type, and using a file
- Make a new text file with the extension **.rb**

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

class MegaGreeter
  attr_accessor :names

  # Create the object
  def initialize(names = "World")
    @names = names
  end

  # Say hi to everybody
  def say_hi
    if @names.nil?
      puts "..."
    elsif @names.respond_to?("each")
      # @names is a list of some kind, iterate!
      @names.each do |name|
        puts "Hello #{name}!"
      end
    else
      puts "Hello #{@names}!"
    end
  end

  # Say bye to everybody
  def say_bye
    if @names.nil?
      puts "..."
    elsif @names.respond_to?("join")
      # Join the list elements with commas
      puts "Goodbye #{@names.join(", ")}.  Come back soon!"
    else
      puts "Goodbye #{@names}.  Come back soon!"
    end
  end

end

if __FILE__ == $0
  mg = MegaGreeter.new
  mg.say_hi
  mg.say_bye

  # Change name to be "Zeke"
  mg.names = "Zeke"
  mg.say_hi
  mg.say_bye

  # Change the name to an array
  mg.names = ["Albert", "Brenda"
    "Dave", "Engelbert"]
  mg.say_hi
  mg.say_bye

  # Change to nil
  mg.names = nil
  mg.say_hi
  mg.say_bye
end
```

- Save the file
- Open up Terminal / Command Prompt again
- Type ruby (and a space after)
- Drag the file onto your Terminal window
- Press enter

```
1  #!/usr/bin/env ruby
2
```

means comment

```ruby
# Say hi to everybody
def say_hi
  if @names.nil?
    puts "..."
  elsif @names.respond_to?("each")
    # @names is a list of some kind, iterate!
    @names.each do |name|
      puts "Hello #{name}!"
    end
  else
    puts "Hello #{@names}!"
  end
end
```

first, check whether @names has a value.

if we can iterate,

we iterate over each name.

otherwise, we just use name as-is.

```ruby
  # Say bye to everybody
  def say_bye
    if @names.nil?
      puts "..."
    elsif @names.respond_to?("join")
      # Join the list elements with commas
      puts "Goodbye #{@names.join(", ")}.  Come back soon!"
    else
      puts "Goodbye #{@names}.  Come back soon!"
    end
  end

end
```

first, check whether @names has a value.

if we can **join**,

we join the names into one string.

otherwise, we just use name as-is.

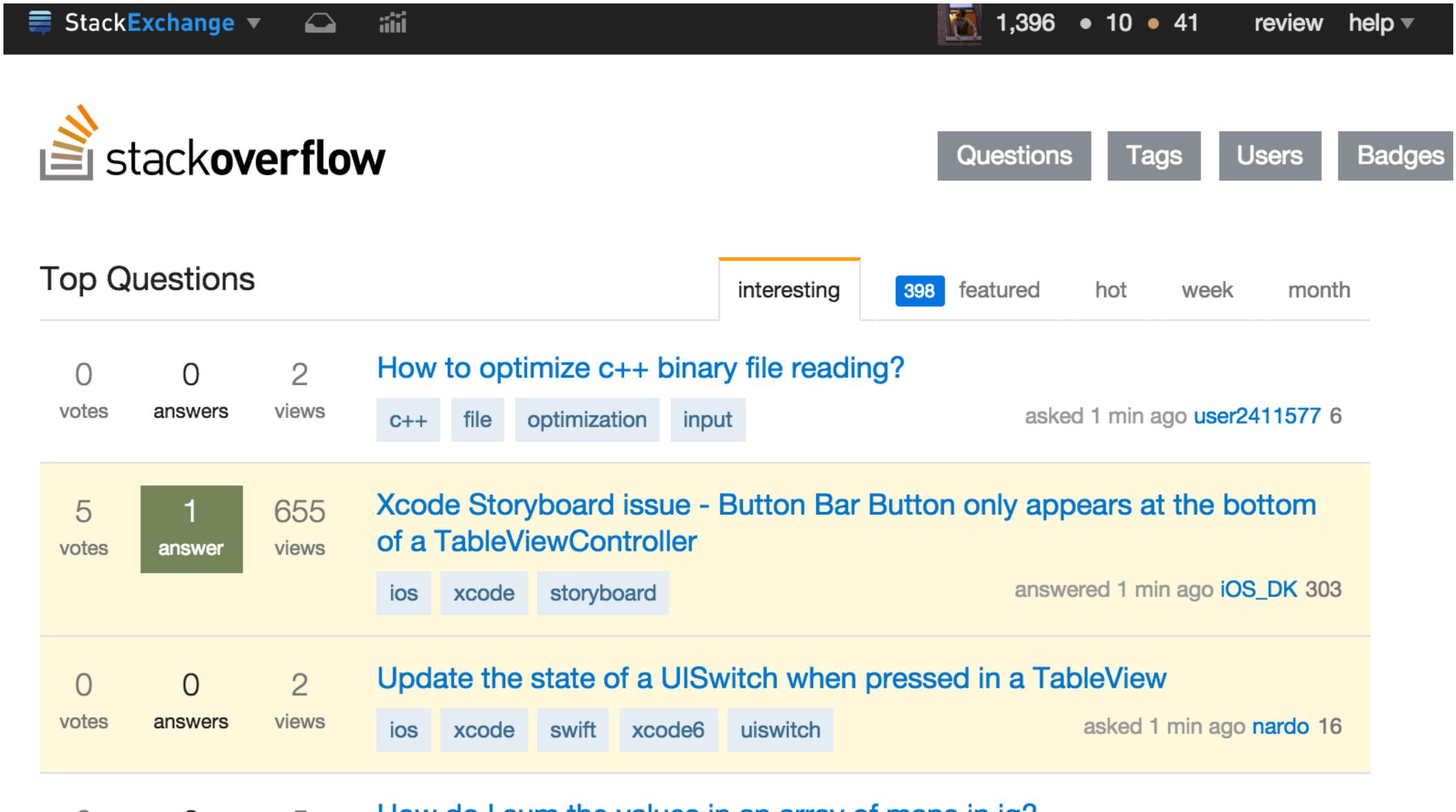
```
if __FILE__ == $0
```

`__FILE__` is the magic variable that contains the name of the current file. `$0` is the name of the file used to start the program. This check says “If this is the main file being used...” This allows a file to be used as a library, and not to execute code in that context, but if the file is being used as an executable, then execute that code.

```ruby
mg = MegaGreeter.new
mg.say_hi
mg.say_bye

# Change name to be "Zeke"
mg.names = "Zeke"
mg.say_hi
mg.say_bye

# Change the name to an array of names
mg.names = ["Albert", "Brenda", "Charles",
  "Dave", "Engelbert"]
mg.say_hi
mg.say_bye

# Change to nil
mg.names = nil
mg.say_hi
mg.say_bye
```

OK?

Pro dev resource of the week:



App Ideas

- An app that texts your phone every time someone retweets or favourites your tweet

  (Twitter API - read tweets, Twilio)

- An app that tweets a random picture of a puppy to any of your followers who tweet something negative

  (Twitter API - post tweets, text analysis, Google image search)

- An app that sends random pictures from the internet to your friends via Snapchat

  (Snapchat unofficial API, Google image search)

- An app that posts pictures to an Instagram account only if they contain mostly the colour red (or whatever)

  (Instagram API, Google image search, image pixel analysis)

Etc.

- Work in pairs or on your own
- Server will be via **Heroku**, which we'll talk about next week
- Present your initial ideas at the start of class next week
- But of course they can change/evolve.