Hi there.

I'm @floordrees

I always loved languages

Little did I know that the languages I'd learn in my (mid) twenties would be the most important ones.

I started August last year

I'm not a developer

Dax Huiberts

@daxhuiberts

Lead Developer at Publitas.com

or "Mad Scientist"

JC

@jancoenhulleman

Team lead at SCSC, previously ASR, ABN, ING

works on Codecademy translations

All programmers have been coding ever since they were potty trained

untrue story

the mathematics you'll have to deal with are as basic as sums and subtractions

Why Ruby is the better language

actually, I kinda like Python too.

we've got Sinatra

we've got unicorns

we've got Rails Girls

we've got #fridayhug's

"Ruby is a dynamic, reflective, general-purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features."

Ruby was first designed and developed in 1993 by Yukihiro "Matz" Matsumoto.

As of 2010, there are a number of complete or upcoming alternative implementations of Ruby, including YARV, JRuby, Rubinius, IronRuby, MacRuby (and its iOS counterpart, RubyMotion), mruby, HotRuby, Topaz and Opal.

Matz:

“I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That’s why I decided to design my own language.”

Matz:

“I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language.”

Matz:

“They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines.”

object-oriented

Every value is an object, including classes and instances. Variables always hold references to objects. Every function is a method and methods are always called on an object. Methods defined at the top level scope become members of the Object class. Since this class is an ancestor of every other class, such methods can be called on any object.

object-oriented

Ruby supports inheritance with dynamic dispatch, mixins and singleton methods. Though Ruby does not support multiple inheritance, classes can import modules as mixins.

Non-OOP programs may be one ‘long’ list of commands.

Syntax

similar to that of Perl and Python

class and method definitions signaled by keywords

keywords are used to define logical code blocks, without braces

line breaks are significant and taken as the end of a statement

unlike Python, indentation is not significant

Syntax

Ruby keeps all of its instance variables completely private to the class and only exposes them through accessor methods.

attr_writer, attr_reader, etc.

Unlike the ‘getter’ and ‘setter’ methods of other languages like C++ or Java, accessor methods in Ruby are created with a single line of code via metaprogramming.

Let's say you have a class Person:

```
class Person
end

person = Person.new
person.name # => no method error
```

Obviously we never defined method name. Let's do that:

```
class Person
  def name
    @name # simply returning an instance variable @name
  end
end

person = Person.new
person.name # => nil
person.name = "Dennis" # => no method error
```

We can read the name, but that doesn't mean we can assign the name. Those are two different methods. Former called reader and latter called writer. We didn't create the writer yet so let's do that.

```
class Person
  def name
    @name
  end

  def name=(str)
    @name = str
  end
end

person = Person.new
person.name = 'Dennis'
person.name # => "Dennis"
```

Awesome. Now we can write and read instance variable `@name` using reader and writer methods. But why waste time writing these methods every time?

```ruby
class Person
  attr_reader :name
  attr_writer :name
end
```

Even this can get repetitive. When you want both reader and writer, just use accessor:

```ruby
class Person
  attr_accessor :name
end

person = Person.new
person.name = "Dennis"
person.name # => "Dennis"
```

```ruby
class Person
  attr_accessor :name

  def greeting
    "Hello #{@name}"
  end
end

person = Person.new
person.name = "Dennis"
person.greeting # => "Hello Dennis"
```

Around 2005, interest in the Ruby language surged in tandem with Ruby on Rails, a popular web application framework written in Ruby.

Ruby on Rails is developed by David Heinemeier Hanson (DHH)

The Rails philosophy

Rails is designed to make programming web applications easier by making assumptions about what every developer needs to get started. It allows you to write less code while accomplishing more.


dry

Convention Over Configuration

Sinatra is a free and open source software web application library and domain-specific language written in Ruby.

Sinatra is maintained by
Travis CI's Konstantin Haase

Sinatra is small and flexible. It does not follow the typical model-view-controller pattern used in other frameworks, such as Ruby on Rails.

Instead, Sinatra focuses on 'quickly creating web-applications in Ruby with minimal effort'.

Apple | BBC | the British Government | LinkedIn | Engine Yard | Heroku | GitHub | Songbird

github.com/floord/uva_ruby

http://tryruby.org