Full Stack Notes

Introduction to Ruby / Ruby Basics

Ruby Basics

Everything in Ruby is an object. A few crucial tid-bits of Ruby knowledge to start.

Table of Contents

Ruby is OO

Ruby is an Object-Oriented language. Everything you manipulate in Ruby is an object. This means that every bit of information and code can be given their own properties and actions. Object-oriented programming calls properties by the name *instance variables* and actions are known as *methods*.

We've already seen that numbers are objects in Ruby. In the Chunky Bacon program we used the `times` method of a FixNum object. To determine the type of any object you can use the *class* method.

```
puts 2.class
```

Output:

```
FixNum
```

The parent class of all objects in Ruby is the Object class.

Comments

In Ruby comments begin with the pound symbol (sometimes called a hash).

```
# This is a Ruby comment
puts 'A comment follows this line.' # This is also a comment
```

Unlike Java, there are no multi-line comments in Ruby*.

```
/* This is a multi-line
comment in Java or C++.
Pretty slick, eh? */
```

In Ruby you would have to do this:

```
# This is a multi-line
# comment in Ruby.
# Ya, sorta lame.
```

Multi-line Comments in Ruby

Ruby actually does have multi-line comments, but I find them awkward to use. These comment block start with a `=begin` and end with a `=end`.

```
=begin
This is a multi-
line comment in Ruby.
=end
```

The `=begin` and `=end` cannot be indented and must be the only text on that line.

Expression Seperators

Using semicolons to terminate your expressions is optional in Ruby. Most Ruby code uses line ends to indicate the end of an expression or statement.

When a line ends in the middle of an expression, Ruby realizes that it will continue on the next line. For example, I can split x = 2 + 2 over two lines.

```
x = 2 +

2
```

Ruby sees a single expression as it expects something after the +. If I wanted to put multiple statements on a line then I could use semicolons.

```
x = 2 + 2; y = x + 1
```

In this course we’ll try our best to avoid using semicolons.

Nil is Null

In Ruby `nil` represents emptiness. This means that any variable that is `nil` contains no value. The variable exists, it isn’t undefined, but it hasn’t been assigned a value.

```ruby
if (clown_car == nil)
  puts 'Where are all the clowns?'
end
```

This code will error as `clown_car` is not defined.