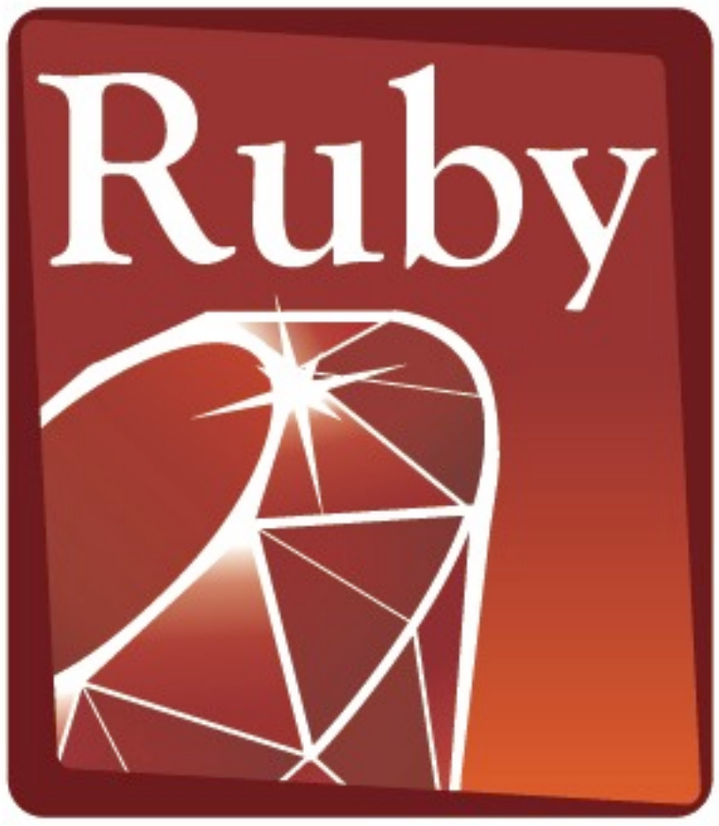

Ruby: The Core Language

Ruby Programming Certificate
UW PCE CP110
Winter 2013
Week 4

Ruby => :chainsaw

```
"Hello".foo
# method lookup ...
# method lookup fails
NoMethodError: undefined method
'foo' for "Hello":String
```

Ruby => :chainsaw

```
class Object
  def method_missing name
    # raise NoMethodError
    "It's Fine."
  end
end


"Hello".foo
=> "It's Fine."
```

Stuff

- Survey
  - 12 people did not respond
  - Pace
    - More in-class review of homework
- Lightning talk now optional
- Group work
- Available to discuss issues, contact me

Agenda

- Final Project Example: ITunesParser
- Guest Speaker: Ivan Storck on Pry
- This Week's Homework
- Week 1 & 2 Homework
- Conditionals
- Classes
- Week 3 Homework / Lab

ITunesParser

- Library to parse an iTunes library file
- Uses Nokogiri
- Exposes API to examine file
- Includes script that uses the API
- Includes Test::Unit tests
- My Project for this course 3 years ago!

Guest Speaker:
Ivan Storck

Pry

Wk4 Homework

- A light week
- Define a few classes
- Basic specs will be provided
- Write a few more specs

Break!

Conditionals

- if else end
- unless else end
- case when else end

if then else end

```
if array.empty? then
  # code to handle empty array

else
  # do something
end
```

```
if array.empty? then array.push(x) end
```

```
if array.empty?; array.push(x); end
```

if elsif else end

```
if array.empty?
  # code here to handle empty array

elsif array.size < 10
  # deal with fewer than 10 items

elsif array.size < 100
  # deal with up to 99 items

else
  # we've got 100 or more items
end
```

unless else end

```
# unless is the opposite of if

unless array.empty?
  # code to handle non-empty array
else
  # array is empty
end


# there is no unlessif !!
```

statement modifiers

```
array.push(x) if array.empty?
```

```
array.push(x) unless array.size > 0
```

case when then else end

```
case food
when 'Soup' then get_spoon
when 'Salad'; get_fork
when 'Pizza' # eat with hands!
else get_all_silverware
end
```

case when else end

```
case food
when 'Soup'
  get_spoon
when 'Salad'
  get_fork
when 'Pizza'
  # eat with hands!
else
  get_all_silverware
end
```

when is amazing

```
case food
when 'Soup', 'Salad' then get_silverware
when String then get_all_silverware
when Array, Hash then eat_first_course
else
  # handle unknown food!
end
```

Classes

- Classes are templates for objects
- Classes define
  - namespace
  - methods
  - CamelCased names
  - Classes are objects, but instances of the class Class

```
String.class
=> Class
"Hello".class
=> String
```

the Object class

- Classes have a parent, or `#superclass`
- Effectively, Object is the root of of the Ruby class hierarchy
  - Technically it's the empty class BasicObject
- The methods in Object are available to all other objects
- Object is the default superclass of any class you define

Defining classes

```
class MyEmptyClass; end


# an autoshop!
class AutoShop

  # return all employees
  def employees
    # code for this instance method
  end

end
```

Initializing objects

```
class AutoShop
  def initialize name
    # initialize method is called from ::new
    # code to initialize instance of class.
    # parameters passed to #new are available
  end
end

AutoShop.new 'My Shop'
 => #<AutoShop:0x007faf2b140c08>
```

Inheritance

```
class AutoShop
   ...
end

class AutoShop < Object
   ...
end

class AutoShop < Business
   ...
end
```

Class Hierarchy

```
class AutoShop < Business
  ...
end

my_shop = AutoShop.new

my_shop.class
=> AutoShop

my_shop.class.superclass
=> Business

my_shop.class.superclass.superclass
=> Object
```

super

```
class AutoShop < Business

  def initialize name

    # initialization for my class
    @makes = [:toyota, :ford]

    # invoke superclass initialization.
    # passes params along by default,
    #   unless you override
    super

  end
end
```

Instance Variables

```
class MyClass
  def initialize name
    @name = name
  end
  def greet
    puts "Hello #{@name}"
  end

end

MyClass.new('Brandon').name
=> NoMethodError
```

Setters and Getters

```
class MyClass
  def initialize name
    @name = name
  end
  def name
    @name
  end
  def name= str
    @name = str
  end
end

my_class = MyClass.new('Joe')
my_class.name = 'Bob'
my_class.name # => 'Bob'
```

Attribute Accessors

```
class MyClass

  attr_accessor :name

end

my_class = MyClass.new()
my_class.name = 'Bob'
my_class.name # => 'Bob'
```

Attribute Readers

```
class MyClass
  attr_reader :created, :state
  def initialize
    @created = Time.now
  end

end

MyClass.new().created # => 2013-01-31 16:58:56
```

Instance Methods

```
class MyClass

  def greet
    puts "Hello #{@name}"
  end

end

MyClass.new('Brandon').greet
=> "Hello Brandon"

MyClass.greet
=> NoMethodError
```

Lab

Homework
(posted soon)

- Reading assignment
  - Conditionals
    - Class basics
- Homework Assignment
- Wk4 Quiz