Learning New Languages & Introduction to Ruby

Slides adapted from Prof. Heimann CMU-P

Philosophy of Ruby

"For me, the purpose of life is, at least partly, to have joy. Programmers often feel joy when they can concentrate on the creative side of programming, so Ruby is designed to make programmers happy."

— Yukihiro Matsumoto

Three Principles

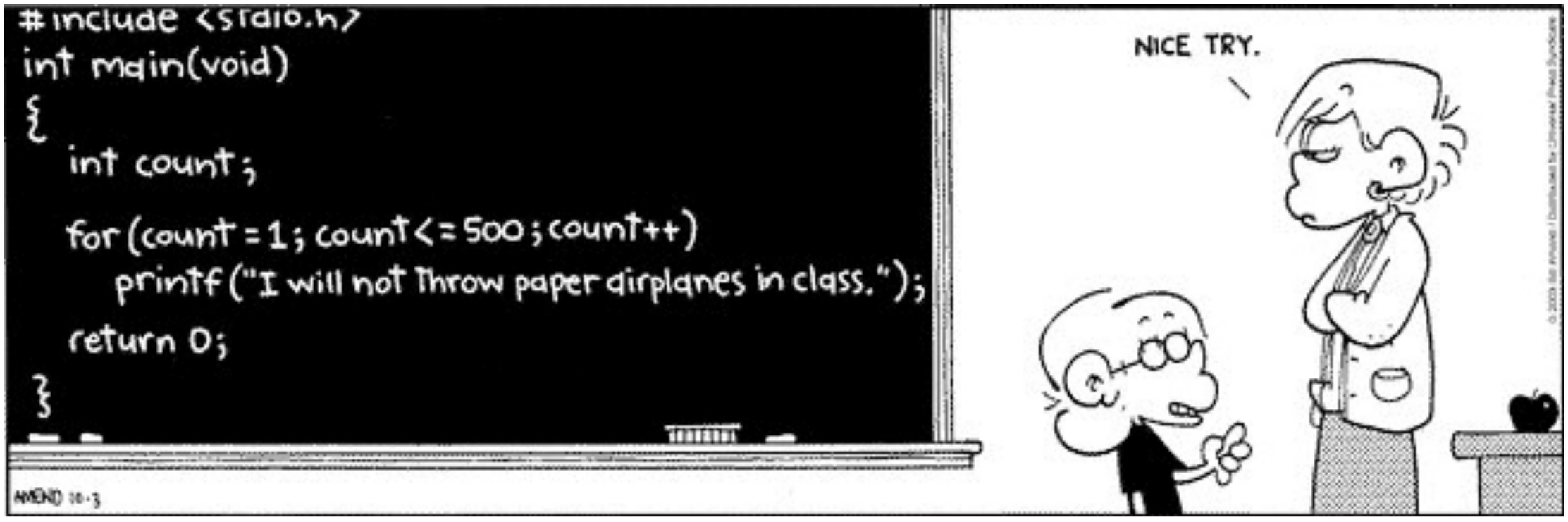
1. **Conciseness**—Writing code in Ruby should involve the minimum amount of commands necessary. Code should be terse but also understandable.

2. **Consistency**—Ruby coding should follow common conventions that make coding intuitive and unambiguous.

3. **Flexibility**—There is no one right way. You should be able to pick the best approach for your needs and be able to even modify the base classes if necessary.

   These three together lead to an important concept in Ruby —
   the principle of least surprise.

Comic of the Day...



The Ruby Way

The Ruby Way

```ruby
500.times { puts "I will not throw paper airplanes" }
```

The Ruby Way

```ruby
500.times { puts "I will not throw paper airplanes" }
```

```ruby
(1..500).each { |i| puts "I will not throw paper airplanes" }
```

The Ruby Way

```ruby
500.times { puts "I will not throw paper airplanes" }
```

```ruby
(1..500).each { |i| puts "I will not throw paper airplanes" }
```

```ruby
for i in (1..500) do
  puts "#{i}. I will not throw paper airplanes"
end
```

Data Types and Structures

Learn about basic data types & structures

- Examples
  - Strings
  - Numbers
  - Arrays
  - Hashes

Everything is an object

Looking at Strings, we see:

```
phrase = "i AM arthur, king of the britons"

puts phrase.class             # >> String
puts phrase.length            # >> 32
puts phrase.capitalize        # >> I am arthur, king of the britons
puts phrase.upcase            # >> I AM ARTHUR, KING OF THE BRITONS
puts phrase.downcase          # >> i am arthur, king of the britons
puts phrase.reverse           # >> snotirb eht fo gnik ,ruhtra MA i
puts phrase.upcase.reverse    # >> SNOTIRB EHT FO GNIK ,RUHTRA MA I
puts phrase.split             # >> i
                              # >> AM
                              # >> arthur,
                              # >> king
                              # >> of
                              # >> the
                              # >> britons
puts phrase.split('a')        # >> i AM
                              # >> rthur, king of the britons
puts phrase.index('a')        # >> 5
puts phrase[5..10]            # >> arthur
puts phrase.capwords          # =>
# ~> -:14: undefined method `capwords' for "i AM arthur, king of the britons":String (NoMethodError)
```

Revising the String class

```ruby
class String
  def capwords
    @words = self.split
    @revised = %w[]
    @words.each do |word|
      @revised << word.capitalize
    end
    @final = @revised.join(" ")
  end
end

phrase = "i AM arthur, king of the britons"
phrase.capwords  # => "I Am Arthur, King Of The Britons"
```

Determining what methods are available

```
str = "fred"
str.class           # => String
String.superclass   # => Object
Object.superclass   # => nil

str.public_methods - Object.public_methods  # !> useless use of - in void context
#     ["%", "select", "[]=", "<<", "each_byte", "gsub", "casecmp", "to_str", "partition",
#      "tr_s", "empty?", "tr!", "rstrip", "*", "match", "grep", "chomp!", "+", "next!",
#      "swapcase", "ljust", "to_i", "swapcase!", "upto", "between?", "reject", "sum", "hex",
#      "insert", "reverse!", "chop", "delete", "dump", "tr_s!", "concat", "member?", "succ",
#      "find", "each_with_index", "strip!", "rjust", "to_f", "index", "collect", "oct", "all?",
#      "slice", "length", "entries", "chomp", "upcase", "sub!", "squeeze", "upcase!", "crypt",
#      "delete!", "detect", "unpack", "zip", "lstrip!", "center", "map", "rindex", "any?",
#      "split", "strip", "size", "sort", "gsub!", "count", "succ!", "downcase", "min", "squeeze!",
#      "downcase!", "intern", "next", "find_all", "each_line", "each", "rstrip!", "slice!", "sub",
#      "replace", "inject", "tr", "reverse", "sort_by", "lstrip", "to_sym", "capitalize", "max",
#      "chop!", "capitalize!", "scan", "[]"]
```

Destructive and Predicate methods

```
str = "fred"
str.capitalize  # => "Fred"
puts str        # >> fred
str.capitalize! # => "Fred"
puts str        # >> Fred
str.reverse     # => "derF"
puts str        # >> Fred
str.reverse!    # => "derF"
puts str        # >> derF

str.include?('ed') # => true
```

Object public methods

```
Object.public_methods # =>

["inspect", "private_class_method", "const_missing", "clone", "method",
  "public_methods", "public_instance_methods", "instance_variable_defined?",
  "method_defined?", "superclass", "equal?", "freeze", "included_modules",
  "const_get", "methods", "respond_to?", "module_eval", "class_variables",
  "dup", "protected_instance_methods", "instance_variables",
"public_method_defined?", "__id__", "object_id", "eql?", "const_set",
"id", "singleton_methods", "send", "class_eval", "taint", "frozen?",
"instance_variable_get", "include?", "private_instance_methods",
"__send__", "instance_of?", "private_method_defined?", "to_a", "name",
"autoload", "type", "new", "<", "protected_methods", "instance_eval",
"<=>", "==", ">", "display", "===", "instance_method", "instance_variable_set",
"kind_of?", "extend", "protected_method_defined?", "const_defined?", ">=",
"ancestors", "to_s", "<=", "public_class_method", "allocate", "hash", "class",
"instance_methods", "tainted?", "=~", "private_methods", "class_variable_defined?",
"nil?", "untaint", "constants", "autoload?", "is_a?"]
```

Looking at numbers

```
var = 42
var.class              # => Fixnum
Fixnum.superclass      # => Integer
Integer.superclass     # => Numeric
Numeric.superclass     # => Object
```

```
var = 3.14
var.class              # => Float
Float.superclass       # => Numeric
Numeric.superclass     # => Object
```

```
var = 420000000000000
var.class              # => Bignum
Bignum.superclass      # => Integer
Integer.superclass     # => Numeric
Numeric.superclass     # => Object
```

Numeric public methods

```ruby
var = 42
var.public_methods - Object.public_methods
#     ["%", "<<", "singleton_method_added", "&", ">>", "round", "divmod",
#     "integer?", "chr", "*", "+", "to_i", "-", "upto", "between?", "prec",
#     "truncate", "/", "modulo", "succ", "|", "zero?", "~", "to_f", "prec_i",
#     "step", "to_int", "^", "remainder", "+@", "nonzero?", "-@", "**",
#     "floor", "prec_f", "quo", "downto", "id2name", "size", "abs", "next",
#     "coerce", "ceil", "div", "times", "to_sym", "[]"]
```

```ruby
var = 3.14159
var.public_methods - Object.public_methods # =>
#     ["%", "singleton_method_added", "round", "divmod", "nan?", "integer?",
#      "*", "+", "to_i", "-", "between?", "prec", "truncate", "/", "infinite?",
#      "modulo", "zero?", "to_f", "prec_i", "step", "to_int", "remainder",
#      "finite?", "+@", "nonzero?", "-@", "**", "floor", "prec_f", "quo",
#      "abs", "coerce", "ceil", "div"]
```

Numeric public methods

```ruby
var = 42
var.public_methods - Object.public_methods
#     ["%", "<<", "singleton_method_added", "&", ">>", "round", "divmod",
#     "integer?", "chr", "*", "+", "to_i", "-", "upto", "between?", "prec",
#     "truncate", "/", "modulo", "succ", "|", "zero?", "~", "to_f", "prec_i",
#     "step", "to_int", "^", "remainder", "+@", "nonzero?", "-@", "**",
#     "floor", "prec_f", "quo", "downto", "id2name", "size", "abs", "next",
#     "coerce", "ceil", "div", "times", "to_sym", "[]"]
```

```ruby
var = 3.14159
var.public_methods - Object.public_methods # =>
#     ["%", "singleton_method_added", "round", "divmod", "nan?", "integer?",
#      "*", "+", "to_i", "-", "between?", "prec", "truncate", "/", "infinite?",
#      "modulo", "zero?", "to_f", "prec_i", "step", "to_int", "remainder",
#      "finite?", "+@", "nonzero?", "-@", "**", "floor", "prec_f", "quo",
#      "abs", "coerce", "ceil", "div"]
```

```ruby
var1 = 12
var2 = 4 * 3
diff = var1 - var2
diff.zero? # => true
```

```ruby
var1 = "12".to_i
var2 = 4 * 3
puts var1 - var2 # >> 0
```

```ruby
var1 = 12
var2 = 4.5 * 3.5 # => 15.75
var3 = (var1 + var2)/3 # => 9.25
var1.between?(var2,var3) # => false
var1.between?(var3,var2) # => true
```

Tax example

```ruby
taxrate = 0.07
print "Enter price of item: "
s = gets
subtotal = s.to_f
tax = subtotal * taxrate
puts
puts "Item cost:\t$#{subtotal}"
puts "Sales tax:\t$#{tax}"
puts "Total due:\t$#{subtotal+tax}"

# >> Enter price of item: 100
# >> Item cost:    $100.0
# >> Sales tax:    $7.0
# >> Total due:    $107.0
```

Add some formatting

```
taxrate = 0.07
puts "Enter price of item: "
s = gets
subtotal = s.to_f
tax = subtotal * taxrate
grand_total = subtotal + tax

# do some formatting
f_subtotal = sprintf("%8.2f", subtotal)
f_tax = sprintf("%8.2f", tax)
f_grand = sprintf("%8.2f", grand_total)

puts "Item cost:\t$#{f_subtotal}"
puts "Sales tax:\t$#{f_tax}"
puts "Total due:\t$#{f_grand}"

# >> Enter price of item: 100
# >> Item cost:    $ 100.00
# >> Sales tax:    $   7.00
# >> Total due:    $ 107.00
```

Arrays

```ruby
food = Array.new
food[0] = "gagh"
food[1] = "blood wine"
food[2] = "raktajino"

human_food = ['turkey', 'caviar', 'potatoes']
human_drink = %w(beer wine water prune\ juice)

puts food
# >> gagh
# >> blood wine
# >> raktajino

puts human_food.join(', ')
# >> turkey, caviar, potatoes

puts human_drink.join(' :: ')
# >> beer :: wine :: water :: prune juice
```

Arrays

```ruby
human_food.each do |new_food|
  food << new_food
end

food.each do |this_food|
  puts this_food.capitalize
end

# >> Gagh
# >> Blood wine
# >> Raktajino
# >> Turkey
# >> Caviar
# >> Potatoes
```

Arrays

```
scores = [95, 90, 80, 90, 100, 85, 80, 80, 85, 70, 55]¬
¬
scores.length # => 11¬
scores.sort # => [55, 70, 80, 80, 80, 85, 85, 90, 90, 95, 100]¬
scores # => [95, 90, 80, 90, 100, 85, 80, 80, 85, 70, 55]¬
scores.sort! # => [55, 70, 80, 80, 80, 85, 85, 90, 90, 95, 100]¬
scores  # => [55, 70, 80, 80, 80, 85, 85, 90, 90, 95, 100]¬
scores.reverse # => [100, 95, 90, 90, 85, 85, 80, 80, 80, 70, 55]¬
scores  # => [55, 70, 80, 80, 80, 85, 85, 90, 90, 95, 100]¬
```

Hashes

```
grades = { "Fred" => 95, "Bill" => 90, "Ryan"  => 80,
           "Bob" => 90, "Sandy"  => 100, "Seth" => 85,
           "Caitlin" => 80, "Butch" => 80, "Beth" => 85,
           "Eustance" => 70, "Clyde" => 55 }

grades.each {|elem| puts "#{elem[0]} scored #{elem[1]}" }
# >> Eustance scored 70
# >> Beth scored 85
# >> Caitlin scored 80
# >> Sandy scored 100
# >> Fred scored 95
# >> Butch scored 80
# >> Seth scored 85
# >> Bill scored 90
# >> Ryan scored 80
# >> Clyde scored 55
# >> Bob scored 90
```

Control Structures

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
while i < options.size do
  puts options[i]
  i += 1
end
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
while i < options.size do
  puts options[i]
  i += 1
end

# >> spam
# >> spam
# >> bacon
# >> eggs
# >> spam
# >> sausage
# >> spam
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
begin
  puts options[i]
  i += 1
end while i < options.size


# >> spam
# >> spam
# >> bacon
# >> eggs
# >> spam
# >> sausage
# >> spam
```

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

```
# >> spam
# >> spam
# >> bacon
# >> eggs
# >> spam
# >> sausage
# >> spam
```

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
until i == options.size do
  puts options[i]
  i += 1
end
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
until i == options.size do
  puts options[i]
  i += 1
end

# >> spam
# >> spam
# >> bacon
# >> eggs
# >> spam
# >> sausage
# >> spam
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
begin
  puts options[i]
  i += 1
end until i == options.size


# >> spam
# >> spam
# >> bacon
# >> eggs
# >> spam
# >> sausage
# >> spam
```

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

i = 0
n = options.size - 1
loop do
  puts options[i]
  i += 1
  break if i > n
end
```

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

n = options.size
n.times do |i|
  puts options[i]
end
```

Loops

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Loops

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

n = options.size - 1
0.upto(n) do |i|
  puts options[i]
end
```

Blocks

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Blocks

```
options = %w[spam spam bacon eggs spam sausage spam]

for food in options do
  puts food
end
```

Blocks

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Blocks

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

options.each {|food| puts food.capitalize}
```

Blocks

```ruby
options = %w[spam spam bacon eggs spam sausage spam]

options.each {|food| puts food.capitalize}

# >> Spam
# >> Spam
# >> Bacon
# >> Eggs
# >> Spam
# >> Sausage
# >> Spam
```

Blocks

```
options = %w[spam spam bacon eggs spam sausage spam]
```

Blocks

```
options = %w[spam spam bacon eggs spam sausage spam]


options.each do |food|
  if food != "spam"
    puts food.capitalize
  end
end
```

Blocks

```ruby
options = %w[spam spam bacon eggs spam sausage spam]


options.each do |food|
  if food != "spam"
    puts food.capitalize
  end
end


# >> Bacon
# >> Eggs
# >> Sausage
```

Blocks

Blocks

```
options.each do |food|
  if food != "spam"
    puts food.capitalize
  end
end
```

Blocks

```ruby
options.each do |food|
  if food != "spam"
    puts food.capitalize
  end
end
```

- or -

```ruby
options.each {|food| puts food.capitalize if food != "spam" }
```

Blocks

```
options.each do |food|
  if food != "spam"
    puts food.capitalize
  else
    puts "Quiet, you vikings!"
  end
end
```

Blocks

```ruby
options.each do |food|
  if food != "spam"
    puts food.capitalize
  else
    puts "Quiet, you vikings!"
  end
end

# >> Quiet, you vikings!
# >> Quiet, you vikings!
# >> Bacon
# >> Eggs
# >> Quiet, you vikings!
# >> Sausage
# >> Quiet, you vikings!
```

Blocks

```ruby
spam_count = 0
options.each do |food|
  if food != "spam"
    puts food.capitalize
  else
    spam_count += 1
  end
end
puts "#{spam_count} times spam is listed"
```

Blocks

```ruby
spam_count = 0
options.each do |food|
  if food != "spam"
    puts food.capitalize
  else
    spam_count += 1
  end
end
puts "#{spam_count} times spam is listed"


# >> Bacon
# >> Eggs
# >> Sausage
# >> 4 times spam is listed
```

Blocks

```
options.each {|food| puts food.capitalize if food != "spam" }
```

Blocks

```ruby
options.each {|food| puts food.capitalize if food != "spam" }
```

- or -

```ruby
options.each {|food| puts food.capitalize unless food == "spam" }
```

Comic of the Day...