Basic Ruby Syntax

```ruby
sum = 0
i = 1
while i <= 10 do
  sum += i*i
  i = i + 1
end
puts "Sum of squares is #{@sum}\n"
```

- No variable declarations
- Newline is statement separator
- `do ... end` instead of `{ ... }
- Optional parentheses in method invocation
- Substitution in string value
Variable Names and Scopes

foo         Local variable
$foo        Global variable
@foo        Instance variable in object
@@foo       Class variable
MAX_USERS   “Constant” (by convention)
Ruby String Syntax

- Single quotes (only \' and \"")
  
  'Bill\'s "personal" book'

- Double quotes (many escape sequences)

  "Found #{count} errors\nAborting job\n"

- %q (similar to single quotes)

  %q<Nesting works:  <b>Hello</b>>

- %Q (similar to double quotes)

  %Q|She said "#{greeting}"|

- “Here documents”

  <<END
  First line
  Second line
  END
x = Array.new
x << 10
x[0] = 99
y = ["Alice", 23, 7.3]
x[1] = y[1] + y[-1]

person = Hash.new
person["last_name"] = "Rodriguez"
person[:first_name] = "Alice"
order = {:item => "Corn Flakes", :weight => 18}
order = {item: "Corn Flakes", weight: 18}
Ruby Statements

```ruby
if x < 10 then
  ...
elsif x < 20
  ...
else
  ...
end

while x < 10 do
  ...
end

array = [14, 22, 34, 46, 92]
for value in array do
  ...
end
```
def fac(x)
    if x <= 1 then
        return 1
    end
    return x * fac(x - 1)
end
Arguments: Defaults, Variable #

```ruby
def inc(value, amount=1)
    value+amount
end

def max(first, *rest)
    result = first
    for x in rest do
        if (x > result) then
            result = x
        end
    end
    return result
end
```
Keyword Arguments

```ruby
def create_widget(size, properties)
  ...
end

create_widget(6, {:id => "table22", :class => "Cart"})
create_widget(6, :id => "table22", :class => "Cart")
create_widget(6, id: "table22", class: "Cart")
```
Blocks, Iterators, Yield

```
odd_numbers(3) do |i|
  print(i, "\n")
end

def odd_numbers(count)
  number = 1
  while count > 0 do
    yield(number)
    number += 2
    count -= 1
  end
end
```

Block: code passed to method

Iterator

Invoke method’s block
def sum_odd(count)
    sum = 0
    odd_numbers(count) do |i|
        sum += i
    end
    return sum
end

def odd_numbers(count)
    number = 1
    while count > 0 do
        yield(number)
        number += 2
        count -= 1
    end
end
Equivalent Code

```ruby
array = [14, 22, 34, 46, 92]
for value in array do
  print(value, "\n")
end

array = [14, 22, 34, 46, 92];
array.each do |value|
  print(value, "\n")
end
```
class Point
  def initialize(x, y)
    @x = x
    @y = y
  end

  def x
    @x
  end

  def x=(value)
    @x = value
  end
end

p = Point.new(3, 4)
puts "p.x is #{p.x}"
p.x = 44
Module Example

class MyClass
    include Enumerable
    ...
    def each
        ...
    end
end

New methods available in MyClass:
    min, max, sort, map, select, ...