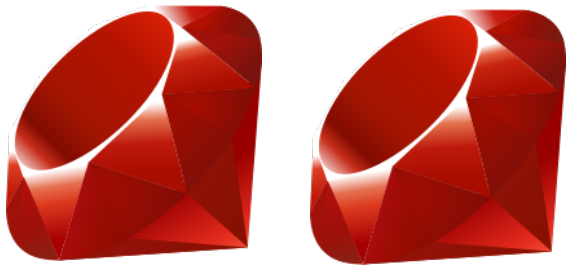


Scope

Scope – local variables

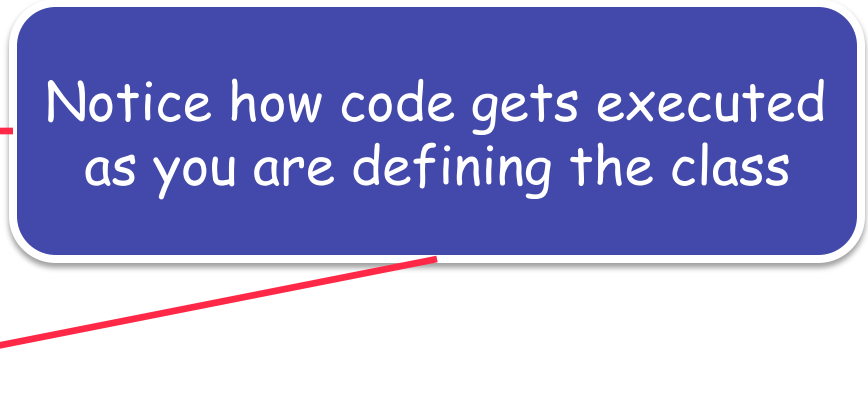
- Methods and classes begin new scope for variables
- Outer scope variables do not get carried over to the inner scope
- Can use `local_variables` method to see which variables are in (and which are not in) the current scope

Scope – local variables – example

```
v1 = 1

class MyClass
  v2 = 2
  p local_variables # => [:v2]
  def my_method
    v3 = 3
    p local_variables
  end
  p local_variables # => [:v2]
end

obj = MyClass.new
obj.my_method # => [:v3]
p local_variables # => [:v1, :obj]
```



Scope - Constants

- Constant is any reference that begins with uppercase, including classes and modules
- Constants' scope rules are different than variable scope rules
- Inner scope CAN see constants defined in outer scope and can also override outer constants

Scope – Constants – example

```ruby
module Test
  PI = 3.14
  class Test2
    def what_is_pi
      puts PI
    end
  end
end
Test::Test2.new.what_is_pi # => 3.14

module MyModule
  MyConstant = 'Outer Constant'
  puts MyConstant # => Outer Constant
  class MyClass
    puts MyConstant # => Outer Constant
    MyConstant = 'Inner Constant'
    puts MyConstant # => Inner Constant
  end
end
```

Scope – Block – Closure

- Blocks inherit outer scope (*default*)
- Block is a closure – remembers the context in which it was defined and uses that context whenever it is called
- Context includes value of `self`, constants, class variables and local variables
- Can use `lambda` to execute a block later

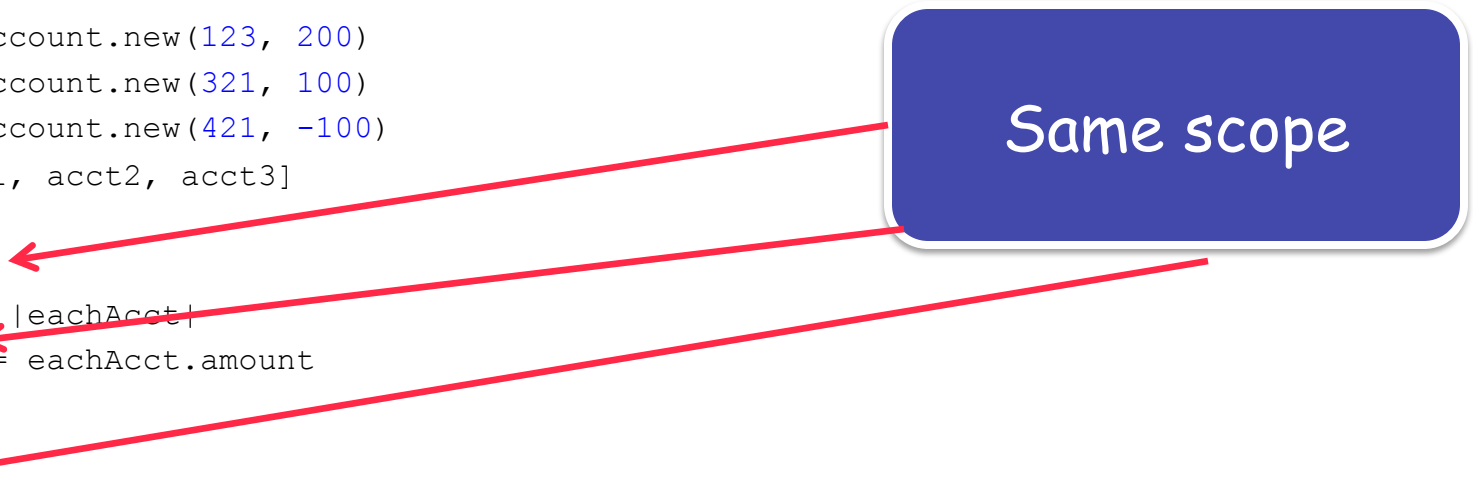
Scope – Block – Example

```
class BankAccount
  attr_accessor :id, :amount
  def initialize(id, amount)
    @id = id
    @amount = amount
  end
end

acct1 = BankAccount.new(123, 200)
acct2 = BankAccount.new(321, 100)
acct3 = BankAccount.new(421, -100)
accts = [acct1, acct2, acct3]

total_sum = 0
accts.each do |eachAcct|
  total_sum += eachAcct.amount
end

puts total_sum # => 200
```



Local Scope – Block (Continued)

- Even though blocks share the outer scope – a variable CREATED inside the block is only available to the block
- Parameters to the block are ALWAYS local to the block – even if they have the same name as variables in the outer scope
- Can explicitly declare block-local variables
  - After a semicolon in the block parameter list

Local Scope – Block – Example

```ruby
arr = [5, 4, 1]
cur_number = 10
arr.each do |cur_number|
  some_var = 10 # not available outside the block
  print cur_number.to_s + " " # => 5 4 1
end
puts # print a blank line
puts cur_number # => 10

adjustment = 5
arr.each do |num;adjustment|
  adjustment = 10
  print "#{num + adjustment} " # => 15 14 11
end
puts
puts adjustment # => 5 (Not affected by the block)
```

Local to the block

Block-local