Ruby Explorations III

Mark Keane...CSI...UCD

De Basics

Part III: More nuts and bolts....

A: Ruby, what are you ?

B: defining classes over several files

C: .learning more about Ruby types

D: ..more about Ruby variables

E: ...about Ruby I/O

Part A:

step back...Ruby what are you?

Ruby is a...

- scripting language; not a static, compiled language like C or Java
- so, Ruby programs are lists of statements to be executed (or scripts)
- statements are executed sequentially; except where flow of control is diverted (e.g., by **if**...)
- note, class/module definitions and methods calls are all statements in Ruby
- Ruby steps through the statements evaluating each

So...

```
x = “mark”

def hail_the_king(name)
    puts “hail king ” + name
end

hail_the_king(x)
```

king2.rb

How Ruby interprets...

- goes through each statement, evaluating it (i.e., what value does it return...)
- IF the item is lower case; it can be an object, a variable or start of a method call (**x** or **hail_the_king** or **def**)
  - THEN, return the value of the object/variable or run the method and see what value it returns (nb, side effects are actions like creating a method definition)
- IF the item is uppercase; it can be a Constant or a Class method or a kernel method (e.g., PI, Person.new, Float)
  - THEN do it as above

How Ruby parses...

- IF the object/variable is followed by a dot then get the following method name THEN evaluate the object/variable, get the method definition and see if you can invoke it with the value you have found (e.g., “marko”.length)

A remaining mystery in **def**

- most of the time we define instance methods in a class that are invoked using object instances: “str”.size
- sometimes we define class methods to do some task Person.new (or those weird hidden Kernel functions)
- but we have also defined methods at the top-level (as we have done in files) that just take arguments and are not within a class (so, they are not invoked via objects) e.g., hail(mark)

Top-level Methods

- ...are instance methods of **Object** (but self is not **Object**, self is **main**)
- ...are always private (don't ask...)
- why ?

1. since they are methods of Object they can (in theory) be used with any object
2. since they are private they must be invoked like functions with no explicit receiver

Part B:

looking at defining classes...over several files

Ruby Classes:Recall

- key idea in OOP is that every thing is an object
- an object has associated methods; actions that are specifically carried out on that object
- some methods come free when you create the object
- others you will define within the object; they may or may not be visible outside the object
- the resulting compartmentalisation is really great...

Ruby Classes

- we saw how methods can be in many files
- ...and saw some issues around accessing them
- similarly, we have seen how sometimes you need to say it is *this* method in *this* class, I want (need to use class object File, Person...)
  - e.g., File.open and Person.new(“mark”)
- so, let’s consider many classes over many files...

2 Classes in 2 files

- consider...
- ***a class for printing names and changing them***
- a class for recording visits

```
class Testo
  attr_accessor :name, :surname

  def initialize(name, surn)
     @name = name
     @surname = surn
  end

  def man_name
     "mr " + @name + @surname
  end

  def man_name=(name_array)
     @name = name_array[1]
     @surname = name_array[2]
  end
end

mark = Testo.new("mark", "keane")
p mark.man_name
mark.man_name = [“mr”, “mark”, “bean”]
p mark.man_name
```

classo.rb

nb use of **method=**

definitions of **man_name** are distinguished by their syntax

2 Classes in 2 files

- consider...
- *a class for printing names and changing them*
- ***a class for recording visits***

```
class Visit
 attr_accessor :place, :person

 def initialize(pl, per)
     @place = pl
     @person = per
 end

 def print_visit
     puts "#{@person} visited #{@place}"
 end
end
```

class2b.rb

Class use I

- to use **Visit** class we need to do **require_relative**
- after that it will work ok

```
class Visit
  attr_accessor :place, :person

  def initialize(pl, per)
    @place = pl
    @person = per
  end

  def print_visit
    puts "#{@person} visited #{@place}"
  end
end
```

class2b.rb

```
require_relative 'class2b'

class Testo
  attr_accessor :name, :surname

  def initialize(name, surn)
    @name = name
    @surname = surn
  end

  def man_name
    "mr " + @name + @surname
  end

  def man_name=(name_array)
    @name = name_array[1]
    @surname = name_array[2]
  end
end

mark = Testo.new("mark", "keane")
p mark.man_name; p mark
trip = Visit.new("venice", mark)
trip.print_visit
```

class2a.rb

```
$ ruby class2a.rb
"mr markkeane"
#<Testo:0x007fd9cb9030d8 @name="mark",@surname="keane
#<Testo:0x007fd9cb9030d8> visited venice
```

Let's fix the printing first...

- consider...
- *a class for printing names and changing them*
- ***a class for recording visits***

```
class Visit
 attr_accessor :place, :person

 def initialize(pl, per)
     @place = pl
     @person = per
 end

 def print_visit
     puts "#{@person.man_name} visited #{@place}"
 end
end
```

class2b.rb

Better to run 2 from a 3rd

- it is best to be as OOP as possible
- we could make ***top.rb*** a class too, so a particular run is an instance
- **obj.method** form finds the method for us

```
require_relative 'class2a'
require_relative 'class2b'

mark = Testo.new("mark", "keane")
mark.man_name

trip = Visit.new("venice", mark)
trip.print_visit
```

top.rb

class2a.rb

class2b.rb

Part C:

back to learning more about Ruby types...

Strings 101

- strings and arrays will get you through
- REM: counts off from 0
- many of its methods work with arrays too

```
>> 43.to_s
=> “43”
>>“string”.length
=> 6
>>“hi” + “guys”
=> “higuys”
>> “hi” + 3
????
>> String.new
??
>> String.new(“me”)
```

if we drop the quotes?

http://www.ruby-doc.org/core/classes/String.html

String Hacking I

- combining strings
- **concat** and **<<** do assignment too
- **+** does not

```
>> str = "foo"
=> "foo"
>> str.concat("bar")
=> "foobar"
>> str
=> "foobar"
>> str + "tt"
=> "foobartt"
>> str
=> "foobar"
>> str << "ff"
=> "foobarff"
>> str
=> "foobarff"
```

http://www.ruby-doc.org/core/classes/String.html

String Hacking II

- cutting strings
- none of these change variables, they just cut

```
>> str << "\n"
=> "foobarff\n"
>> str.chomp
=> "foobarff"
>> str.chop
=> "foobarff"
>> str.chop.chop
=> "foobarf"
>> str
=> "foobarff\n"
>> str[0,4]
=> "foob"
>> str[5,10]
=> "rff\n"
>> str[5,9]
```

String Equality

- nb , **=** and **==**


```
>> str_a = "foobar"
=> "foobar"
>> str_b = "FooBar"
=> "FooBar"
>> str_a == str_b
=> false
>> str_a == str_a
=> true
>> str_a.eql?(str_b)
=> false
>> str_a != str_b
=> true
```

```
>> “HeLLo”.downcase
“hello”
```

Strings 2 Arrays

what went wrong !

- turn a string into an array (nb, single item)
- not what you wanted?

```
>> “ggddaa”.split(/-/)

=> [“ggddaa”]

>>str = “gg dd aa”

=> “gg dd aa”

>>str.split

=> [“gg”, “dd”, “aa”]

>> “fffss”.split
 [“fffss”]

>> “fffss”.split(//)

=>["f","f","f","s","s"]
```

http://www.ruby-doc.org/core/classes/String.html

Symbols 101

- symbols are immutable, strings are mutable (so most string methods don't work)
- can save memory over string usage and runtime speed
- if you want to name things (e.g., hash keys)

what does a symbol eval to?

```
>> :marko
=> :marko
>> p :marko
:marko
=> :marko
>> :marko.class
=> Symbol
>> :marko.to_s
=> "marko"
>> :marko.to_i
NoMethodError
>> :marko = "keane"
SyntaxError:
unexpected '='
```

http://www.ruby-doc.org/core/classes/Symbol.html

Arrays 101

- strings and arrays will get you through
- REM: counts off from 0
- alot of methods that work with strings, work here too (sort of)

```
>> Array.new(3)
=> [nil, nil, nil]
>> foop = [43]
=> [43]
>>foop.length
=> 1
>> foop + [“doop”]
=> [43, “doop”]
>> foop + “doop”
TypeError: no implicit
conversion of String into Array
>> foop + “doop”.split
=> [43, “doop”]
```

var is eval-ed

http://www.ruby-doc.org/core/classes/Array.html

Array Hacking I

- combining arrays
- variable assignment etc works as with strings

```
>> arr1 = ["foo"]
=> ["foo"]
>> arr1.concat(["bar"])
=> ["foo", “bar”]
>> arr1
=> ["foo", “bar”]
>> arr1 + "tt"
=> TypeError ...
>> arr1 + ["tt"]
=> ["foo", “bar”, “tt”]
>> arr1
=> ["foo", “bar”]
>> arr1 << ["ff"]
=> ["foo",“bar”,[“ff”]]
>> arr1
=> ["foo",“bar”,[“ff”]]
```

how would we create an array of variables

http://www.ruby-doc.org/core/classes/Array.html

Array Hacking II

- getting parts of arrays
- cutting arrays

```
>> arr2 = [“one”,”two”,”tree”]
=> [“one”,”two”,”tree”]
>> arr2[0]
=> "one"
>> arr2[2]
=> "tree"
>> arr2[1,2]
=> [”two”,”tree”]
>> arr2[5]
=> nil
>> arr2[2] + arr2[0]
=> ”treeone”
>> arr2.first
=> "one"
>> arr2.last
=> "tree"
```

http://www.ruby-doc.org/core/classes/Array.html

Array Equality

```
>> arr1 = ["foo”,”bar"]
=> ["foo”,”bar"]
>> arr2 = ["foo”,”bim"]
=> ["foo”,”bim"]
>> arr3 = ["foo”,”bar"]
=> ["foo”,”bar"]
>> arr4 = [“bar”, “foo”]
>> arr1 == arr2
=> false
>> arr1 != arr2
=> true
>> arr1 == arr3
=> true
>> arr1 == arr4
=> false
>> arr1.eql?(arr3)
=> true
```

http://www.ruby-doc.org/core/classes/Array.html

Arrays 2 Strings

- turn an array into a string
- not what you wanted?

```
>> arr = [“gg”, “dd”, “aa”]
=> [“gg”, “dd”, “aa”]
>> arr.to_s
=> “[\”gg\”, \”dd\” , \”aa\”]”
>> arr.join
=> “ggddaa”
>> arr.join(“ “)
=> “gg dd aa”
>> arr.join(“_“)
=> “gg_dd_aa”
>> arr << “ff”
=> [“gg”, “dd”, aa”, “ff”]
>> arr.join
=> “ggddaaff”
```

for multiply embedded

Arrays, Blocks & Iterators Ia

- recall blocks, **do..end** and **{|foo| puts foo}**
- **each**, **select**, **collect**, **map**
- note what is returned
- nb, relationship to **Enumerable** module and **Enumerator** class

```
>> foo = [“qi”,”qa”, “quo”]
=> [“qi”,”qa”, “quo”]
>> foo.each {|ele| puts ele + “e”}
qie
qae
quoe
=> [“qi”,”qa”, “quo”]
```

ele bound to each element of **foo** and **puts** applied

whole form returns the value of **foo**

```
>> [1,2,3].each.class
#<Enumerator: [1, 2, 3]:each>
```

http://www.ruby-doc.org/core/classes/Array.html

Arrays, Blocks & Iterators lb

- recall blocks, **do..end** and **{|foo| puts foo}**
- **select**
- nb, what is returned

```
>> foo = [“qi”,”qa”, “quo”]
=> [“qi”,”qa”, “quo”]

>> foo.select {|elo| elo.include?(“a”)}
=> [”qa”]
```

elo bound to each element of **foo** and **include?** test applied

whole form returns array of items that meet the test

http://www.ruby-doc.org/core/classes/Array.html

Iterators 2a

- **each** and **do...end**
- spot the deliberate

- how would you convert it to obj calls

```
class Sausage
  attr_accessor :name, :made_of, :taste

  def initialize(name, made, tasty)
    @name,@made_of,@taste = name, made,tasty
  end

  def self.judge_all(sausies)
    sausies.each do |saussy|
      if saussy.made_of = “pork”
        then saussy.taste = “yummy”
      elsif saussy.made_of = “offal”
        then saussy.taste = “yuck”
      else saussy.taste = “dont know” end
    end
  end
end

saus1 = Sausage.new(“selbys”,”pork”,nil)
saus2 = Sausage.new(“grandby”,”offal”,nil)
saus3 = Sausage.new(“blalong”,”beef”,nil)
p saus1
p saus2
Sausage.judge_all([saus1, saus2, saus3])
p saus1
p saus2
```

many vars

end of **do**

end of **if**

array of objs

food_wrong.rb

http://www.ruby-doc.org/core/classes/Array.html

Iterators 2b

- **each** and **do...end**
- this uses **==**
- say we wanted it to take any number of objects as args and we can't use an array

```
class Sausage
  attr_accessor :name, :made_of, :taste

  def initialize(name, made, tasty)
    @name,@made_of,@taste = name, made,tasty
  end

  def self.judge_all(sausies)
    sausies.each do |saussy|
      if saussy.made_of == "pork"
        then saussy.taste = "yummy"
      elsif saussy.made_of == "offal"
        then saussy.taste = "yuck"
      else saussy.taste = "dont know"
      end
    end
  end
end

saus1 = Sausage.new("selbys","pork",nil)
saus2 = Sausage.new("grandby","offal",nil)
saus3 = Sausage.new("blalong","beef",nil)
p saus1
p saus2
Sausage.judge_all([saus1, saus2, saus3])
p saus1
p saus2
```

equality

equality

food_wrong.rb

http://www.ruby-doc.org/core/classes/Array.html

Iterators 2c

- **each** and **do...end**
- this uses * or **splat**
- **splat** gathers up args into an array
- so, method can take any number of args

```
class Sausage
  attr_accessor :name, :made_of, :taste

  def initialize(name, made, tasty)
    @name,@made_of,@taste = name, made,tasty
  end

  def self.judge_all(*sausies)
    sausies.each do |saussy|
      if saussy.made_of == “pork”
        then saussy.taste = “yummy”
      elsif saussy.made_of == “offal”
        then saussy.taste = “yuck”
      else saussy.taste = “dont know”
      end
    end
  end
end

saus1 = Sausage.new(“selbys”,”pork”,nil)
saus2 = Sausage.new(“grandby”,”offal”,nil)
saus3 = Sausage.new(“blalong”,”beef”,nil)
p saus1
p saus2
Sausage.judge_all(saus1, saus2, saus3)
p saus1
p saus2
```

splat

any no of args

food.rb

http://www.ruby-doc.org/core/classes/Array.html

Iterators 3

- **each** and **do...end**
- a personal favourite
- though there are primitives for it

default arg

```
def combos(array1, array2, n = 0)
  array1.each do |ele1|
    array2.each do |ele2|
      n += 1
      puts "#{ele1}, #{ele2}, on #{n}"
    end
  end
end

combos(["a","b","c"], [1,2,3])
```

combo.rb

method can
be called with 2 or 3 args

http://www.ruby-doc.org/core/classes/Array.html

Iterators 4a

- **each** and **do...end**
- beware of llamas in lynx clothing and the values returned by calls

default arg

```
def weird(array1, out = [])
  array1.each {|ele1|
    out = puts ele1 + “out” }
  out
end

p weird([“a”,“c”,“x”])
```

weird.rb

http://www.ruby-doc.org/core/classes/Array.html

Iterators 4b

- **each** and **do...end**
- beware of llamas in lynx clothing and the values returned by calls

default arg

```
def weird(array1, out = [])
  array1.each {|ele1|
    out = puts ele1 + “out” }
  out
end

p weird([“a”,“c”,“x”])
```

weird.rb

```
$ ruby weird.rb
aout
cout
xout
nil
$
```

how would you make out work...

http://www.ruby-doc.org/core/classes/Array.html

Part D:

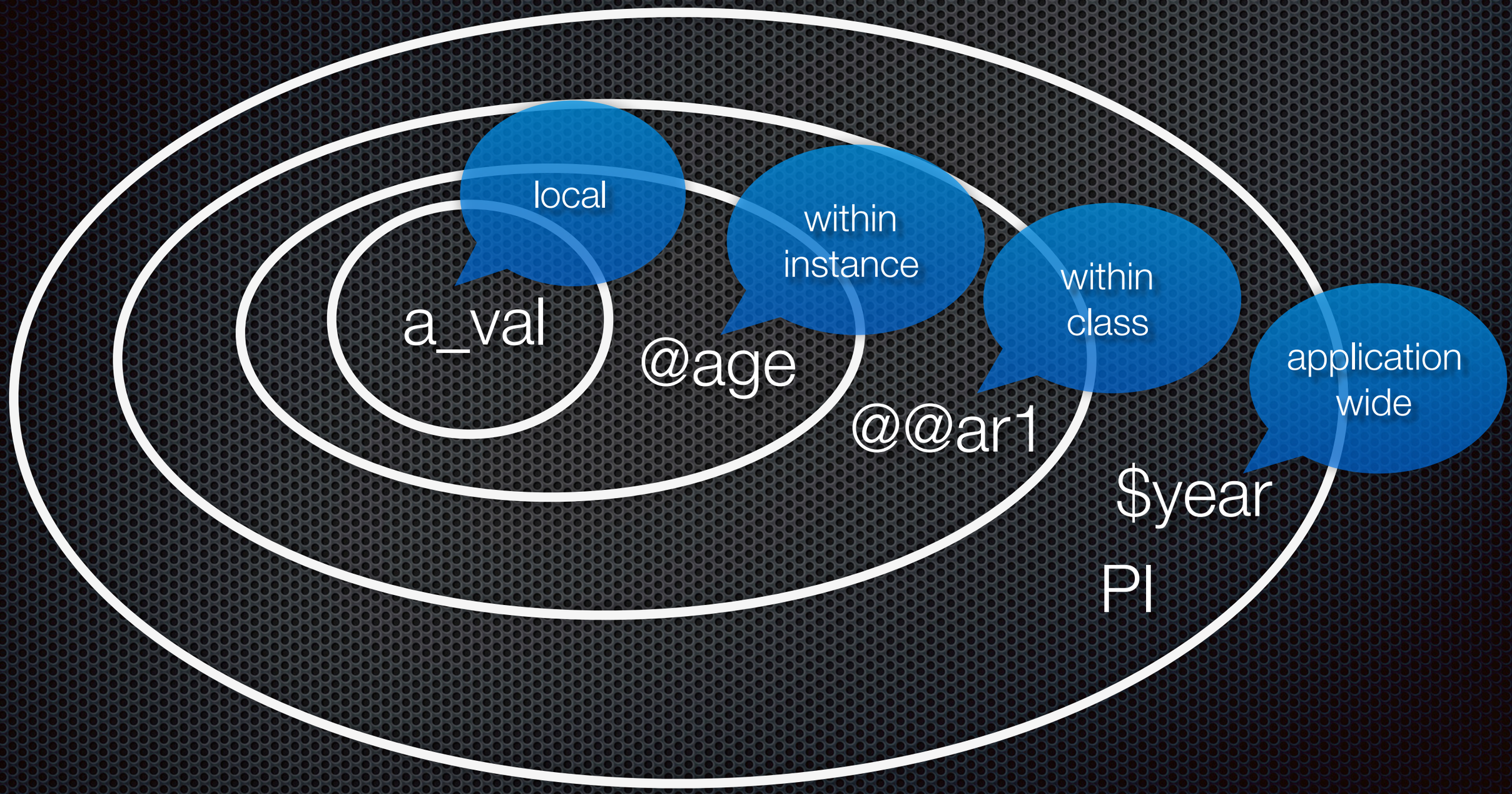
back to learning more about Ruby variables...

Ruby Variables...

← Variables → Constants

Local	Global	Instance	Class	& Class names
name	$debug	@name	@@total	PI
fish_n_chip	$BUYER	@point_1	@@syntab	FeetPerM
x_axis	$_	@X	@@N	String
thn1134	$plan9	@_	@@x_pos	MyClass
_26	$Global	@plan9	@@ONE	JazzS

Ruby Variables...



@inst_v v @@class v

- shows how we access from instance or class
- changes from anywhere are forever

```
class Visit
  @@all = []
  Places = [“berne”,”venice”]
  attr_accessor :place, :person
  def initialize(pl, per)
 @place,@person = pl, per
  end

  def to_s
 puts "#{@person} visited #{@place}"
  end

  def all
    @@all
  end

  def self.all
    @@all
  end

  def add_visit_via_instance
 @@all << self
 end

  def self.add_visit_via_class=(inst)
    @@all << inst
  end
end
```

class var initialised

constant

inst access

class access

inst assign

class assign

vars.rb

http://www.ruby-doc.org/core/classes/Array.html

Variables

```
foo = Visit.new("venice", "ruth_b")
bar = Visit.new("berne", "mark_k")
p foo
p bar
puts "Value of @@all, found via foo, is now:"
p foo.all
foo.add_visit_via_instance
puts "value of @@all, found via bar, is now"
p bar.all
Visit.add_visit_via_class = bar
puts "value of @@all, found via the class, is now:"
p Visit.all
p Visit::Places
```

vars.rb

```
$ruby vars.rb
#<Visit:0x007fb79296c520 @place="venice", @person="ruth_b">
#<Visit:0x007fb79296c4a8 @place="berne", @person="mark_k">
Value of @@all, found via foo, is now:
[]
value of @@all, found via bar, is now
[#<Visit:0x271a0 @person="ruth_b", @place="venice">]
value of @@all, found via the class, is now:
[#<Visit:0x271a0 @person="ruth_b", @place="venice">,
 #<Visit:0x27164 @person="mark_k", @place="berne">]
["berne", "venice"]
```

http://www.ruby-doc.org/core/classes/Array.html

Part E:

more Ruby I/O, reading from csv files...

csv files

- csv.file to read
- without comments
- and with comments
- we use the CSV class...library for parsing such files

row headers

```
"Fname","Surname","Age"
"Mark","Keane",49
"Mahatma","Gandi",67
"Grommit","Dog",10
"Haroun","Al-raschid",56
"Regina","Spector",31
```

people0.csv

OR

```
"Fname","Surname","Age"
# THIS FILE CONTAINS PEOPLE
# May 31 2010
##########################################
#          All PEOPLE
# (1) Can be very sensitive to errors
# (2) no space after a comma
# (3) entries read as strings even numbers
# (4) need code to ignore the commented lines
##########################################
"Mark","Keane",49
"Mahatma","Gandi",67
"Grommit","Dog",10
"Haroun","Al-raschid",56
"Regina","Spector",31
```

people.csv

http://ruby-doc.org/stdlib/libdoc/csv/rdoc/index.html

You may have to load a Gem

- try it using **require 'csv'**; if you get an error, do this:

  $ gem install fastercsv [with a **sudo** if needs be...]

- with multiple versions....gem2.0 install fastercsv
- if you get errors...lets talk (CSC -> faster_csv -> CSV)

http://ruby-doc.org/stdlib/libdoc/csv/rdoc/index.html

```ruby
require 'csv'

CSV.foreach("people0.csv", :headers => true) do |row|
    p row
  end

CSV.foreach("people0.csv") do |row|
    p row
  end

CSV.foreach("people0.csv") do |row|
    puts row[1]
  end
```

readO.rb

```
"Fname","Surname","Age"
"Mark","Keane",49
"Mahatma","Gandi",67
```

people0.csv

```
$ ruby reado.rb
#<CSV::Row "Fname":"Mark" "Surname":"Keane" "Age":"49">
#<CSV::Row "Fname":"Mahatma" "Surname":"Gandi" “Age":"67">

["Fname", "Surname", "Age"]
["Mark", "Keane", "49"]
["Mahatma", "Gandi", “67"]

Surname
Keane
Gandi
```

http://ruby-doc.org/stdlib/libdoc/csv/rdoc/index.html

```ruby
require 'csv'

CSV.foreach("people0.csv", :headers => true) do |row|
    p row
  end

CSV.foreach("people0.csv") do |row|
    p row
  end

CSV.foreach("people0.csv") do |row|
    puts row[1]
  end
```

uses headers

file name, hardwired

readO.rb

```
"Fname","Surname","Age"
"Mark","Keane",49
"Mahatma","Gandi",67
```

people0.csv

```
$ ruby reado.rb
#<CSV::Row "Fname":"Mark" "Surname":"Keane" "Age":"49">
#<CSV::Row "Fname":"Mahatma" "Surname":"Gandi" “Age":"67">

["Fname", "Surname", "Age"]
["Mark", "Keane", "49"]
["Mahatma", "Gandi", “67"]

Surname
Keane
Gandi
```

http://ruby-doc.org/stdlib/libdoc/csv/rdoc/index.html

```ruby
require 'csv'

class Person
  def initialize(fname, surname, age)
   @fname, @surname, @age = fname, surname, age
  end
end

class PimpyReader
  def initialize()
   @people = Array.new
  end

  def get_people
   @people
  end

  def read_in_people(file_name)
    CSV.foreach(file_name, :headers => true) do |row|
       fname, surname, age = row[0],row[1],row[2]
    if  !fname.include?("#")  #gets rid of comment lines
       then @people << Person.new(fname,surname,age.to_i)
    end
     end
    @people
  end
end
```

reader.rb

```
"Fname","Surname","Age"
# THIS FILE CONTAINS PEOPLE
# May 31 2014
#########################################
#          All PEOPLE
# (1) Can be very sensitive to errors
# (2) no space after a comma
# (3) entries read as strings even numbers
# (4) need code to ignore the commented lines
#########################################
"Mark","Keane",49
"Mahatma","Gandi",67
"Grommit","Dog",10
"Haroun","Al-raschid",56
"Regina","Spector",31
```

people.csv

```
require 'csv'

class Person
  def initialize(fname, surname, age)
   @fname, @surname, @age = fname, surname, age
  end
end

class PimpyReader
  def initialize()
   @people = Array.new
  end

  def get_people
   @people
  end

  def read_in_people(file_name)
    CSV.foreach(file_name, :headers => true) do |row|
       fname, surname, age = row[0],row[1],row[2]
    if  !fname.include?("#")
       then @people << Person.new(fname,surname,age.to_i)
    end
     end
    @people
  end
end
```

person class

reader class

special each for reading file lines

multi-assign

make it a no

reader.rb

```ruby
reader = PimpyReader.new
csv_file_name = ARGV[0]  #assumes you are using command-line interface
p csv_file_name
reader.read_in_people(csv_file_name)
p reader.get_people
```

reader.rb

```
$ ruby reader.rb people.csv
"people.csv"
[#<Person:0x5104b4 @fname="Mark", @surname="Keane", @age=49>, #<Person:
0x510004 @fname="Mahatma", @surname="Gandi", @age=67>, #<Person:
0x517b4c @fname="Grommit", @surname="Dog", @age=10>, #<Person:
0x51769c @fname="Haroun", @surname="Al-raschid", @age=56>, #<Person:
0x5171ec @fname="Regina", @surname="Spector", @age=31>]
$
```

```
 ruby reader.rb
nil
/opt/local/lib/ruby2.3/2.3.0/csv.rb:1265:in `initialize':
no implicit conversion of nil into String (TypeError)
  from /opt/local/lib/ruby2.3/2.3.0/csv.rb:1265:in `open'
  from /opt/local/lib/ruby2.3/2.3.0/csv.rb:1265:in `open'
  from /opt/local/lib/ruby2.3/2.3.0/csv.rb:1130:in `foreach'
  from reader.rb:23:in `read_in_people'
  from reader.rb:37:in `<main>'
```

What's the Problem....

ARGV is like an I/O splat !

ARGV[0]

```
$ ruby reader.rb people.csv
```

ARGV = [“people.csv”]

ARGV[0]

ARGV[2]

```
$ ruby reader.rb a.csv b.csv c.scv
```

ARGV[1]

ARGV = [“a.csv”, “b.csv”, “c.csv”]

```
$ruby reader.rb
/opt/local/lib/ruby1.9/1.9.1/csv.rb:1330:in
`initialize': can't convert nil into String (TypeError)
```

ARGV[0] is [ ]

ARGV = [ ]

If you are in RubyMine...

```
/usr/bin/ruby -e $stdout.sync=true;
$stderr.sync=true;load($0=ARGV.shift) /Users/user/Desktop/
X_Teaching/Ruby:2011-14/Lects&Pracs.2014/RubyWeek4 (Oct 3).13/
RubyLect4.progs/reader.rb

nil

/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/
ruby/2.0.0/csv.rb:1254:in `initialize': no implicit conversion of
nil into String (TypeError)

 from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/
lib/ruby/2.0.0/csv.rb:1254:in `open'

 from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/
lib/ruby/2.0.0/csv.rb:1254:in `open'

 from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/
lib/ruby/2.0.0/csv.rb:1119:in `foreach'

 from /Users/user/Desktop/X_Teaching/Ruby:2011-14/Lects&Pracs.
```