Learning Ruby - 4

Ruby Code Blocks

Ruby Code Blocks

- Pay Attention - **This is important!**
- Code blocks in Ruby are not like “blocks of code” in other programming languages
- Blocks can **implement** "callbacks" and "iterators"
- Blocks can be **passed around** from method to method
- Blocks are typically **associated** with method calls

Block Examples Seen So Far

```
3.times { print "Ho! " }

1.upto( 5 ) { |i| puts i }

my_a.each { |element| puts element if element.length > 4 }

classic_rock.collect do | song, artist |
   puts "#{artist} performs '#{song}'."
end # of do.

puts songs.collect { | song, artist | "#{artist} performs '#{song}'." }.sort
```

How Code Blocks Work

- The code within **{ .. }** or **do .. end** is passed to the method (as if it were another parameter)
- The method then arranges to call the block of code as required
- It is possible to write custom methods that take advantage of this Ruby feature
- The "yield" method will call an associated code block from within a method
- Ruby code blocks are no more evident than when used as **iterators**

More Block Examples

```
ENV.collect do | key, value |
   print key + ' -> ' + value + "\n"
end # of do.

[ 'a', 'e', 'i', 'o', 'u' ].each { |ch| puts ch }

( 'f'..'t' ).each { |ch| print ch }

[2, 3, 5, 7, 11].find { |prime| prime*prime > 30 }

kids = %w( joseph aaron aideen )
kids.each { |child| puts "#{child}\n" }
```

More ... Ruby So Far

- Code blocks are cool!
- Code blocks are used to implement iterators
- Take the time to understand how they work
- Before using **while** (or any other looping construct), ask yourself if an iterating code block will do instead