**Common Enumerables**

Ruby's robust enumerable methods are what makes it a uniquely readable and expressive programming language. Classic enumerables like each, map, and select are staples but there are more enumerables that you will want to familiarize yourself with to write even cleaner code! This is meant to be an overview of a few methods you'll find useful, so you'll want to reference the [Ruby Docs](https://ruby-doc.org/) for the complete documentation of every method available in ruby!

**all?**

Return true when all elements result in true when passed into the block.

p [2, 4, 6].all? { |el| el.even? } # => true

p [2, 3, 6].all? { |el| el.even? } # => false

**any?**

Return true when all at least one element results in true when passed into the block.

p [3, 4, 7].any? { |el| el.even? } # => true

p [3, 5, 7].any? { |el| el.even? } # => false

**none?**

Return true when no elements result in true when passed into the block.

p [1, 3, 5].none? { |el| el.even? } # => true

p [1, 4, 5].none? { |el| el.even? } # => false

**one?**

Return true when exactly one element results in true when passed into the block.

p [1, 4, 5].one? { |el| el.even? } # => true

p [1, 4, 6].one? { |el| el.even? } # => false

p [1, 3, 5].one? { |el| el.even? } # => false

**count**

Return a number representing the count of elements that result in true when passed into the block.

p [1, 2, 3, 4, 5, 6].count { |el| el.even? } # => 3

p [1, 3, 5].count { |el| el.even? } # => 0

**sum**

Return the total sum of all elements

p [1, -3, 5].sum # => 3

**max and min**

Return the maximum or minimum element

p [1, -3, 5].min # => -3

p [1, -3, 5].max # => 5

p [].max # => nil

**flatten**

Return the 1 dimensional version of any multidimensional array

multi\_d = [

[["a", "b"], "c"],

[["d"], ["e"]],

"f"

]

p multi\_d.flatten # => ["a", "b", "c", "d", "e", "f"]