**Pre-Exercise Note**

We will continue using the command line and pry to test our code, navigate our computer, and perform many other amazing tasks. Getting comfortable with these tools early is very important in becoming an efficient developer. Once you learn them, these tools will make just about any operation you can think of faster than using a GUI and mouse.

**Iteration Exercises**

We're going to implement some Array methods. There are descriptions for each method, but it may help to also consult the official Ruby docs for [Enumerable](http://ruby-doc.org/core-2.1.2/Enumerable.html) and [Array](http://ruby-doc.org/core-2.1.2/Array.html). These are good resources in general, and useful reading during this first week.

**Note:** Unlike in the prepwork, there are no specs to compare against. You'll have to test your code in pry.

**Learning Goals**

* Be able to create directories and files from the command line
* Know how to extend classes
* Know how to use pry to test methods
* Be able to write methods that take a block as an argument
* Get comfortable reasoning about how enumerable methods work with arrays

Start by opening the terminal and changing your directory to the Desktop using the cd command. Create a new directory for your work using the mkdir command and then open a new file in VS Code called enumerables.rb using the code command.

[Enumerable](http://ruby-doc.org/core-2.1.2/Enumerable.html)

**My Each**

Extend the Array class to include a method named my\_each that takes a block, calls the block on every element of the array, and returns the original array. Do not use Enumerable's each method. I want to be able to write:

# calls my\_each twice on the array, printing all the numbers twice.

return\_value = [1, 2, 3].my\_each do |num|

puts num

end.my\_each do |num|

puts num

end

# => 1

2

3

1

2

3

p return\_value # => [1, 2, 3]

**My Select**

Now extend the Array class to include my\_select that takes a block and returns a new array containing only elements that satisfy the block. Use your my\_each method!

Example:

a = [1, 2, 3]

a.my\_select { |num| num > 1 } # => [2, 3]

a.my\_select { |num| num == 4 } # => []

**My Reject**

Write my\_reject to take a block and return a new array excluding elements that satisfy the block.

Example:

a = [1, 2, 3]

a.my\_reject { |num| num > 1 } # => [1]

a.my\_reject { |num| num == 4 } # => [1, 2, 3]

**My Any**

Write my\_any? to return true if any elements of the array satisfy the block and my\_all? to return true only if all elements satisfy the block.

Example:

a = [1, 2, 3]

a.my\_any? { |num| num > 1 } # => true

a.my\_any? { |num| num == 4 } # => false

a.my\_all? { |num| num > 1 } # => false

a.my\_all? { |num| num < 4 } # => true

[Array](http://ruby-doc.org/core-2.1.2/Array.html)

**My Flatten**

my\_flatten should return all elements of the array into a new, one-dimensional array. Hint: use recursion!

Example:

[1, 2, 3, [4, [5, 6]], [[[7]], 8]].my\_flatten # => [1, 2, 3, 4, 5, 6, 7, 8]

**My Zip**

Write my\_zip to take any number of arguments. It should return a new array containing self.length elements. Each element of the new array should be an array with a length of the input arguments + 1 and contain the merged elements at that index. If the size of any argument is less than self, nil is returned for that location.

Example:

a = [ 4, 5, 6 ]

b = [ 7, 8, 9 ]

[1, 2, 3].my\_zip(a, b) # => [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

a.my\_zip([1,2], [8]) # => [[4, 1, 8], [5, 2, nil], [6, nil, nil]]

[1, 2].my\_zip(a, b) # => [[1, 4, 7], [2, 5, 8]]

c = [10, 11, 12]

d = [13, 14, 15]

[1, 2].my\_zip(a, b, c, d) # => [[1, 4, 7, 10, 13], [2, 5, 8, 11, 14]]

**My Rotate**

Write a method my\_rotate that returns a new array containing all the elements of the original array in a rotated order. By default, the array should rotate by one element. If a negative value is given, the array is rotated in the opposite direction.

Example:

a = [ "a", "b", "c", "d" ]

a.my\_rotate #=> ["b", "c", "d", "a"]

a.my\_rotate(2) #=> ["c", "d", "a", "b"]

a.my\_rotate(-3) #=> ["b", "c", "d", "a"]

a.my\_rotate(15) #=> ["d", "a", "b", "c"]

**My Join**

my\_join returns a single string containing all the elements of the array, separated by the given string separator. If no separator is given, an empty string is used.

Example:

a = [ "a", "b", "c", "d" ]

a.my\_join # => "abcd"

a.my\_join("$") # => "a$b$c$d"

**My Reverse**

Write a method that returns a new array containing all the elements of the original array in reverse order.

Example:

[ "a", "b", "c" ].my\_reverse #=> ["c", "b", "a"]

[ 1 ].my\_reverse #=> [1]

**Review**

Now that we're all warmed up, let's review the [iteration exercises](https://assets.aaonline.io/fullstack/ruby/assets/prep_iteration_exercises.rb) from the prepwork. Implement the following methods:

* #factors(num)
* #bubble\_sort!(&prc)
* #bubble\_sort(&prc)
* #substrings(string)
* #subwords(word, dictionary)

Although these exercises are from the prepwork and come with specs, use this opportunity to practice your own testing skills. Write out each method, think of a few different example cases, and test out your code in pry.

If you are having a tough time thinking of example cases, check out the [specs](https://assets.aaonline.io/fullstack/ruby/assets/w1d1_spec.zip). You can use those examples to test your solutions. Don't forget to move your enumerables\_array.rb into a lib directory!