Enumerable module, a set of methods are available to do traversing, sorting, searching etc across the collection(Array, Hashes, Set, HashMap).

1. **For Loop:**

```
CountriesName = ["India", "Canada", "America", "Iraq"]
for country in CountriesName
    puts country
end
```

2. **Each Iterator:**

Same set of work can be done with each loop which we did with for loop.

```
CountriesName = ["India", "Canada", "America", "Iraq"]
CountriesName.each do |country|
    puts country
end
```

Each iterator, iterate over every single element of the array.
each ---------- iterator
do ---------- start of the block
|country| ----- argument passed to the block
puts country----block

3. **each_with_index Iterator:**

each_with_index iterator provides the element for the current iteration and index of the element in that specific collection.

```
CountriesName = ["India", "Canada", "America", "Iraq"]
CountriesName.each_with_index do |country, index|
    puts country + "  " + index.to_s
end
```

4. **each_index Iterator:**

Just to know the index at which the element is placed in the collection.

```
CountriesName = ["India", "Canada", "America", "Iraq"]
CountriesName.each_index do |index|
    puts index
end
```

5. **map:**
"map" acts as an iterator and also used to fetch the transformed copy of the array. To fetch the new set of the array rather than introducing the change in the same specific array.
Let's deal with for loop first:
You have an array arr = [1,2,3,4,5]
You need to produce new set of array.

```
arr = [1,2,3,4,5]
newArr = []
for x in 0..arr.length-1
   newArr[x] = -arr[x]
end
```

The above mentioned array can be iterated and can produce new set of the array using map method.

```
arr = [1,2,3,4,5]
newArr = arr.map do |x|
           -x
         end

puts arr
[1,2,3,4,5]

puts newArr
[-1, -2, -3, -4, -5]
```

map is returning the modified copy of the current value of the collection. arr has unaltered value.

Difference between each and map:
1. map returned the modified value of the collection.

Let's see the example:

```
arr = [1,2,3,4,5]
```

```ruby
newArr = arr.map do |x|
           puts x
           -x
         end

puts newArr
[-1, -2, -3, -4, -5]
```

map method is the iterator and also return the copy of transformed collection.

```ruby
arr = [1,2,3,4,5]
newArr = arr.each do |x|
           puts x
           -x
         end

puts newArr
[1,2,3,4,5]
```

each block will throw the array because this is just the iterator.
Each iteration, doesn't actually alter each element in the iteration.

6. **map!**
map with bang changes the original collection and returned the modified collection not the copy of the modified collection.

```ruby
arr = [1,2,3,4,5]
arr.map! do |x|
      puts x
    -x
end
puts  arr
[-1, -2, -3, -4, -5]
```

7. **Combining map and each_with_index**
Here each_with_index will iterator over the collection and map will return the modified copy of the collection.

```ruby
CountriesName = ["India", "Canada", "America", "Iraq"]
```

```ruby
newArray =
CountriesName.each_with_index.map do |value, index|
    puts "Value is #{value} and the index is #{index}"
    "Value is #{value} and the index is #{index}"
end

newArray =
CountriesName.each_with_index.map do |value, index|
    if ((index%2).eql?0)
        puts "Value is #{value} and the index is #{index}"
        "Value is #{value} and the index is #{index}"
    end
end

puts newArray
 ["Value is India and the index is 0", nil, "Value is America and the
index is 2", nil]
```

8. **select**

```ruby
MixedArray = [1, "India", 2, "Canada", "America", 4]
MixedArray.select do |value|
    (value.class).eql?Integer
end
```

select method fetches the result based on satisfying certain condition.

9. **inject methods**

inject method reduces the collection to a certain final value.
Let's say you want to find out the sum of the collection.
With for loop how would it work

```ruby
arr = [1,2,3,4,5]
sum = 0
for x in 0..arr.length-1
   sum = sum + arr[0]
end
puts sum
15
```

So above mentioned sum can be reduce by single method

```
arr = [1,2,3,4,5]
arr.inject(0) do |sum, x|
    puts x
    sum = sum + x
end
```

inject(0) - passing initial value sum = 0
If used inject with no argument sum = arr[0]
sum - After each iteration, total is equal to the return value at
the end of the block.
x - refers to the current iteration element

inject method is also an iterator. Summary: Best way to transform the collection is to make use of Enumerable module to compact the clunky code.