Collections in Ruby

Creation

```
bros = []
bros = [ 'groucho', 'harpo', 'chico' ]
bros = Array.new
bros = Array.new( 3 )
bros = Array.new( 3, true )
bros = Array.new( 4 ) { Hash.new }
bros = Array.new( 2 ) { Array.new( 2 ) }
%w{ Hello World }
%w[ Hello World ]
%w/ Hello World /
```

Accessing Elements

```
arr = [1, 2, 3, 4, 5, 6]
arr[2] # => 3
arr[100]  # => nil
arr[-1] # => 6
arr[-3] # => 4
arr[2, 3] # => [3, 4, 5]
arr[1..4] # => [2, 3, 4, 5]
```

Accessing Elements

```
arr.at(0) # => 1
arr.first # => 1
arr.last # => 6
arr.take(3)
# => [1, 2, 3] - Grabs the first three elements
arr.drop(3)
# => [4, 5, 6] - Grabs the last three elements
arr.fetch(100) # => IndexError: index 100 outside of
arr.fetch(100, "ERROR") # => "ERROR"
```

Adding Elements

```
arr = [1, 2, 3, 4]
arr.push(5)
arr << 6
\# = [1, 2, 3, 4, 5, 6] Uses push behind the scenes
arr.unshift(9)
\# = [0, 1, 2, 3, 4, 5, 6] Adds an element to the start
arr.insert( 3, 'Serge' )
# => [ 0, 1, 2, 'Serge', 3, 4, 5, 6 ]
arr.insert( 4, 'didnt marry', 'Jane')
# => [0, 1, 2, 'Serge', 'didnt marry', 'Jane', 3, 4, 5, 6]
```

Removing Elements

```
# Pop removes the last element and returns it
# (it is destructive)
arr = [1, 2, 3, 4, 5, 6]
arr.pop # => 6
arr \# \Rightarrow [1, 2, 3, 4, 5]
# To retrieve and at the same time remove the first item
arr.shift # => 1
# Delete at a particular index
arr.delete at( 2 )
```

Removing Elements

```
# To delete a particular element anywhere
arr = [1, 2, 2, 3]
arr.delete(2) # => [1, 3]
# Compact will remove nil values
arr = ['foo', 0, nil, 'bar', 7, 'baz', nil]
arr.compact #=> ['foo', 0, 'bar', 7, 'baz']
# Remove duplicates
arr = [2, 5, 6, 556, 6, 6, 8, 9, 0, 123, 556]
arr.uniq # => [2, 5, 6, 556, 8, 9, 0, 123]
```

```
arr = [1, 2, 3, 4, 5]
arr.each do |el|
   puts el
end
arr.each { |el| puts el }
arr.reverse each do |el|
    puts el
end
arr.reverse_each { |el| puts el }
```

```
# DON'T DO IT THESE WAYS!
arr = [1,2,3,4,5,6]
for x in 0..(arr.length-1)
    puts arr[x]
end
# or, with while:
x = 0
while x < arr.length</pre>
    puts arr[x]
    x += 1
end
for el in arr
    puts el
```

```
arr = [1, 2, 3]
arr.map do |a|
     a * 2
end
arr.map { |a| a * 2 } arr.map! { |a| a * 2 }
```

Predicates and Destruction

Predicate Method

Methods that return a boolean value - they always end with a question mark in Ruby (2.even?)

Destructive Method

Changes the data - often they end in an exclamation mark

Blocks

```
# Very similar to anonymous functions in JavaScript
arr = [2, 4, 6, 8, 10]
arr.map { | num | num * 3 }
arr.map! { | num | num * 3 }
arr.map do | num |
    num * 3
end
arr.map! do | num |
    num * 3
end
```

Selection and Rejection

```
arr = [1, 2, 3, 4, 5, 6]
arr.select do |a|
    a > 3
end
arr.select \{ |a| a > 3 \}
arr.select! { |a| a > 3 }
arr.reject { |a| a < 4 }
arr.reject! \{ |a| a < 4 \}
arr.delete_if \{ |a| | a < 4 \}
arr.keep_if \{ |a| a < 4 \}
```

Array Comparison

```
array1 = ["x", "y", "z"]
array2 = ["w", "x", "y"]
array1 | array2
# Combine Arrays & Remove Duplicates (Union)
# => ["x", "y", "z", "w"]
array1 & array2
# Get Common Elements between Two Arrays (Intersection)
array1 - array2
# Remove Any Elements from Array 1 that are
# contained in Array 2. (Difference)
```

Have a crack at these exercises

Object IDs, Strings vs. Symbols and False

(Almost) everything is an object in Ruby! Every time a new anything is created, it gets assigned a new object_id and also a new place in memory.

```
"Hello".object_id
:random_symbol.object_id

false.object_id
{}.object_id
```

The only things that are considered falsey in Ruby is the boolean false and nil

Creation of a hash

```
random hash = {}
person = {
    name: "Elke",
    location: "Berkeley"
person = {
    :name => "Elke",
    :location => "Berkeley"
person = {
    "name": "Elke",
    "location": "Berkeley"
```

Creation of a hash

```
hash = Hash.new
# Normally a hash will return nil if the property is undefined
# We can pass in default values to this quite easily though
hash = Hash.new( false )
hash["Something"] #=> Will return false
hash = {}
hash.default = false
hash["Elke"] # => false
```

Accessing Values

```
person = {
    :name => "Elke",
    :location => "Berkeley",
    "Skill in Mr. Squiggle" => 5
person[:name]
person["Skill in Mr. Squiggle"]
```

Adding and Removing Keys

```
serge = {
    "name" => "Serge",
    "nationality" => "French"
serge[:counterpart] = "Jane (temporarily)"
serge[:counterpart]
serge.delete(:counterpart)
```

```
serge = {
    name: "Serge",
    nationality: "French"
serge.each do |key, value|
    puts "Key: #{key} and Value: #{value}"
end
serge.keys.each do |key|
    puts key
end
serge.values.each do |value|
    puts value
end
```

Have a crack at these exercises

Here is your homework