Ruby Enumerables: 'any', 'all', 'none', and 'find'

Ruby offers various enumerables on collections that check for validity of the objects within it.

Consider the following example:

```
> arr = [1, 2, 3, 4, 5, 6]
=> [1, 2, 3, 4, 5, 6]
> h = {"a" => 1, "b" => 2, "c" => 3}
=> {"a" => 1, "b" => 2, "c" => 3}
```

The any? method returns `true` if the block *ever* returns a value other than `false` or `nil` for any element passed to it:

```
> arr.any? {|a| a % 2 == 0} # checks if any number in the array is even
=> True
> h.any? {|key, value| value.is_a? String} # checks if any value of the Hash object is of the type String
=> False
```

The all? method returns `true` if the block *never* returns `false` or `nil` for any element passed to it:

```
> arr.all? {|a| a.is_a? Integer} # checks if all elements of the array are of the type Integer
=> True
> h.all? {|key, value| key.is_a? String} # checks if all keys of the Hash object are of the type String
=> True
```

The none? method returns `true` if the block *never* returns `true` for any element passed to it:

```
> arr.none? {|a| a.nil?} # Checks if none of the elements in the array are of nil type
=> True
> h.none? {|key, value| value < 3} # checks if all values of the Hash object are less than 3
=> False
```

The find method returns the first element for which block is *not* `false`:

```
> arr.find {|a| a > 5} # returns the first element greater than 5 and `nil` if none satisfies the condition
=> 6
> h.find {|key, value| key == "b"} # returns an Array of the first match [key, value] that satisfies the
condition and nil otherwise
=> ["b", 2]
```

Task

Based on what you've learned above, complete the functions declared in your editor below.