Enumerable Methods

Real-Life Example

Jeff takes pictures of all of the 1410 students.

```
students.each do |student|
    jeff.take_picture(student)
end
```

Other Use Cases

- You have a collection of orders and you want to select only the unfulfilled orders.
- You have a collection of events, and you want to group them by their type (music event, political event, etc.)

What are enumerable methods?

(arrays and hashes)

From Ruby Docs:

"The Enumerable mixin provides collection classes with several traversal and searching methods, and with the ability to sort."

In other words:

Enumerable methods are methods that can be used on arrays and hashes to go through each element or search for elements/an element.

Two ways to write enumerable methods

```
array.method do |item|
    item.do_something
end
```

```
array.method { |item| item.do_something }
```

Outline

- **Enumerable methods that iterate over a collection**
  each, map/collect, sort_by, reduce/inject
- **Enumerable methods that filter a collection**
  select/find_all, detect/find, reject
- **Enumerable methods that return true or false**
  all?, any?, none?, one?
- **Enumerable methods that do other cool things**
  zip, group_by, max_by, min_by

Enumerable methods to iterate over a collection

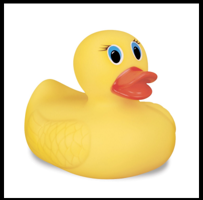
#each

#map/#collect

#sort_by

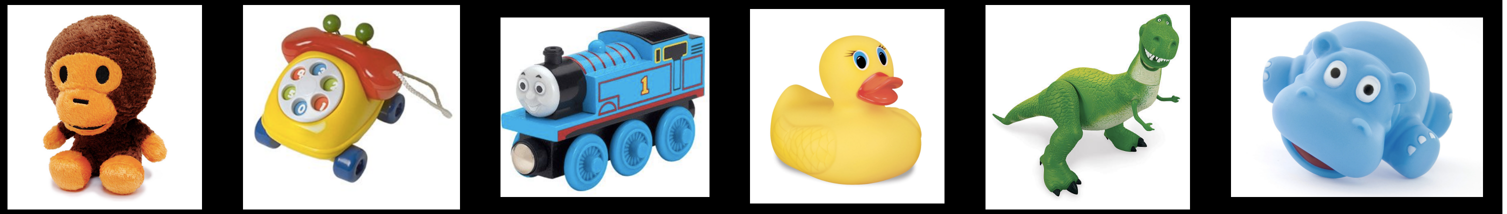
#reduce/#inject

toys = [



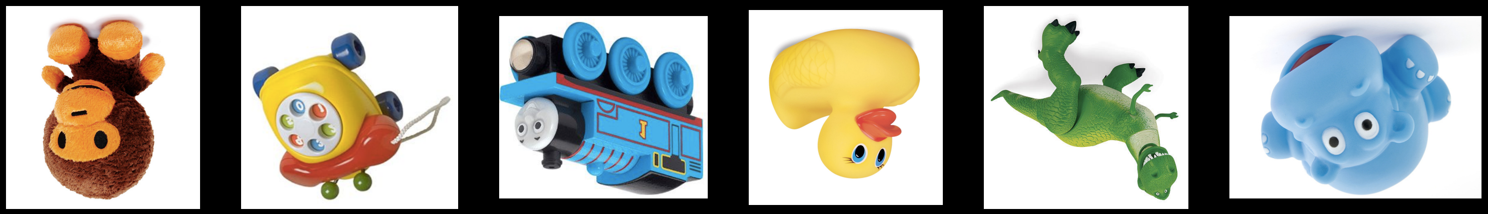
]

#each

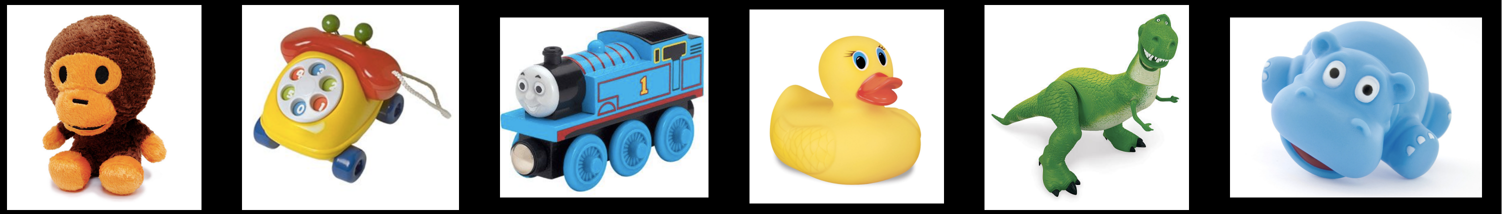
toys = []

```
toys.each do |toy|
     print toy.vertical_flip
end
```

What gets printed?



What gets returned?

[]

#each

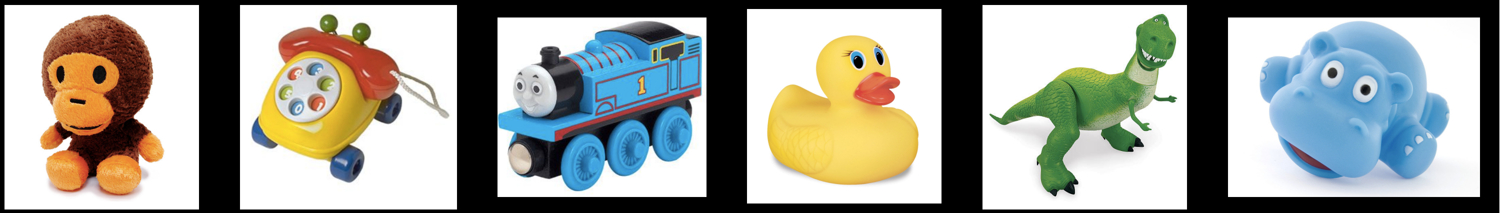
- iterates over each element in the array, does whatever you tell it to do within the block
- returns original array (unless elements have been permanently modified, like calling .capitalize! on an array of strings)

Let’s try it in pry.

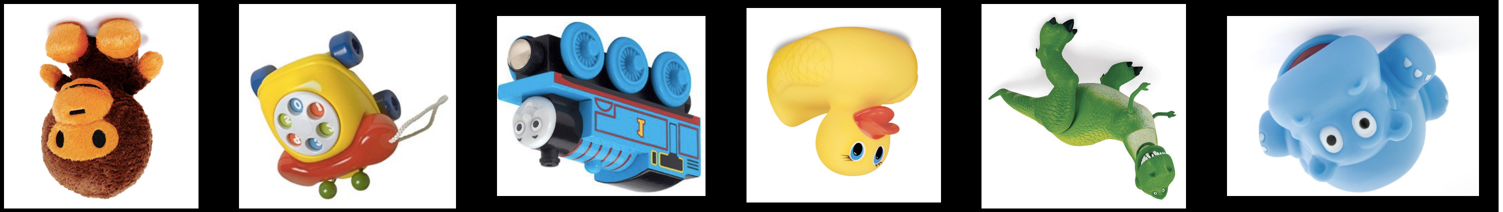
Any enumerable method can be written with #each, but finding the right enumerable method can save time and effort.

#map
#collect

(they’re the same thing!)

toys = []

```
toys.map do |toy|
     toy.vertical_flip
end
```

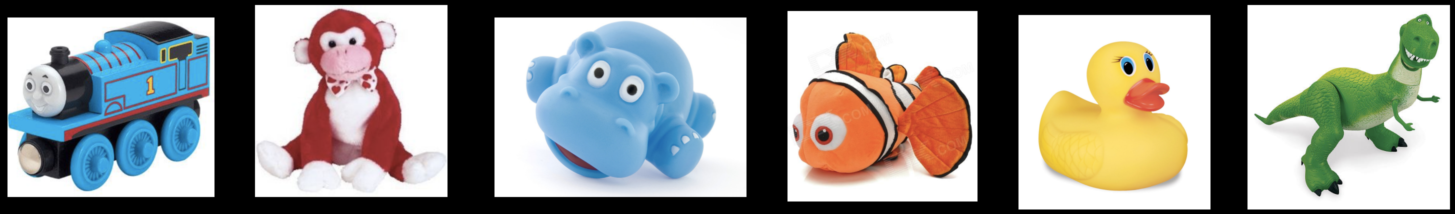
What gets returned? []

#map or #collect

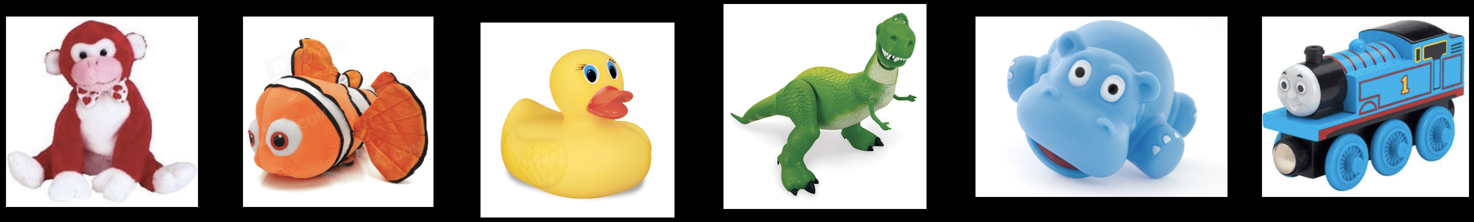
- iterates over each element in the array, does whatever you tell it to do within the block
- returns a mutated array and original array is still intact
- can mutate the original array with .map!

Let’s try it in pry.

#sort_by

toys = []

```
toys.sort_by do |toy|
      toy.color
end
```

What gets returned? []

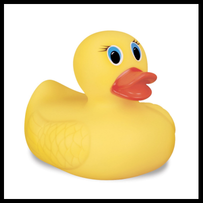
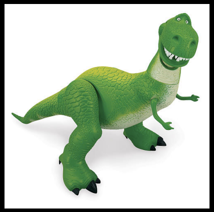
#sort_by

- sorts the items in the collection according to the value of the block

Let’s try it in pry.

#inject
#reduce

(they’re the same thing!)

toys = [ ]

```
toys.reduce do |sum, toy|
     sum + toy
end
```

What gets returned?

#reduce or #inject

- in general, combines all elements of collection using instructions in a block, but can take on many forms:
    - reduce(initial, sym)
    - reduce(sym)
    - reduce(initial) { |memo, obj| block }
    - reduce { |memo, obj| block }

Let's try it in pry.

break?

Enumerable methods to filter a collection

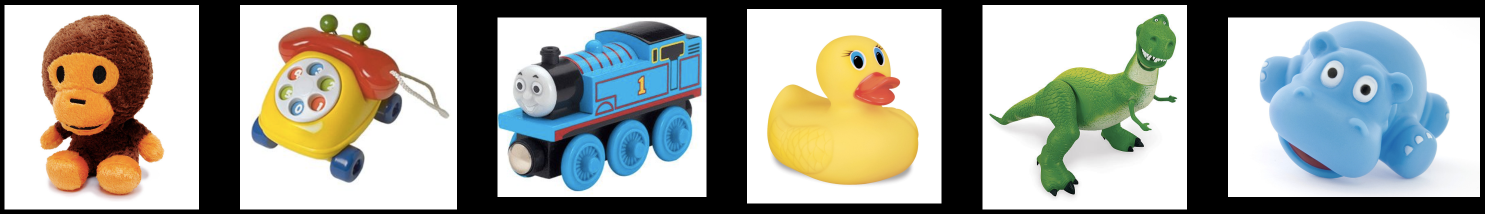
#select/#find_all

#detect/#find

#reject

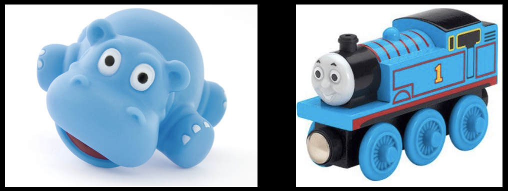
#select
#find_all

(they’re the same thing!)

toys = []

```
toys.select do |toy|
    toy.color == "blue"
end
```

What gets returned?

[]

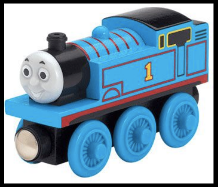
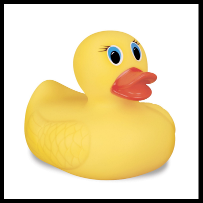
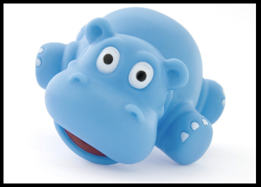
#select or #find_all

- iterates over collection and returns an array of all elements for which the block returns true

Let’s try it in pry.

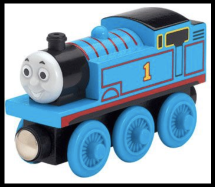
#detect
#find

(fun fact: they’re the same thing!)

toys = [  ]

```
toys.find do |toy|
     toy.color == "blue"
end
```

What gets returned?

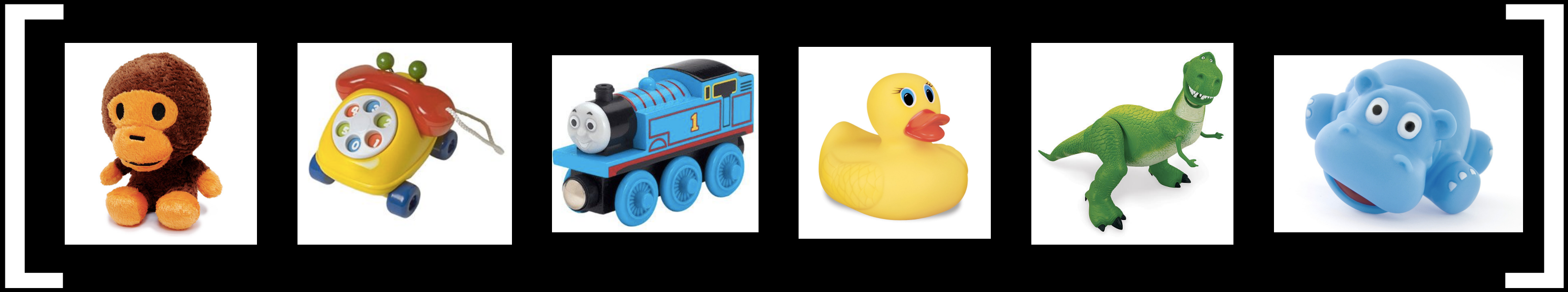


#detect or #find

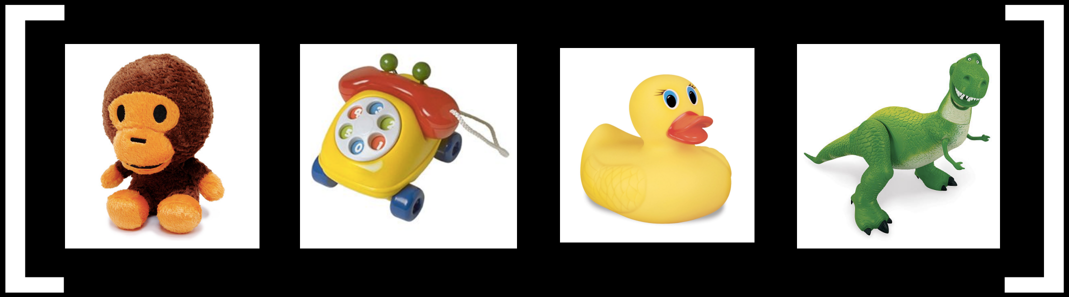
- iterates over collection and returns THE FIRST ELEMENT for which the block returns true

Let’s try it in pry.

#reject

toys = 

```
toys.reject do |toy|
      toy.color == “blue”
end
```

What gets returned? 

#reject

- iterates over collection and returns an array of elements for which the block condition is false
- in other words, it rejects any elements that are true for the block condition

Let’s try it in pry.

break?

Enumerable methods that check a collection and return true or false

#all?	toys.all? { |toy| toy.is_squishy? }	=> false
#any?	toys.any? { |toy| toy.is_squishy? }	=> true
#none?	toys.none? { |toy| toy.is_a?(Integer) }	=> true
#one?	toys.one? { |toy| toy.is_blue? }	=> false

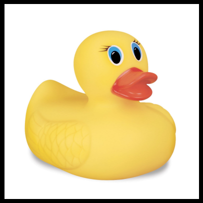
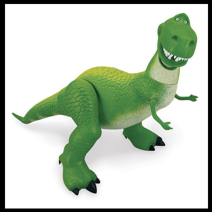
Enumerable methods that do other cool things

#zip

#count

#group_by

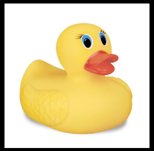
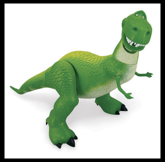
#max_by

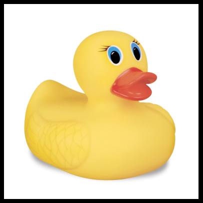
toys = [ ]

children = [ ]

toys.**zip**(children)

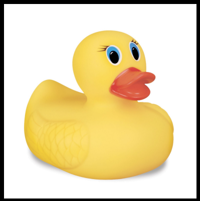
What gets returned?

[[ ,], [ ,], [,]]

toys = []

```
toys.count do |toy|
    toy.name.include?('e')
end
```

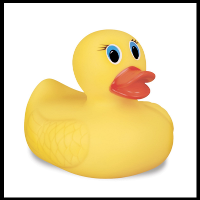
What gets returned? **3**

toys = []

```
toys.group_by do |toy|
      toy.color
end
```

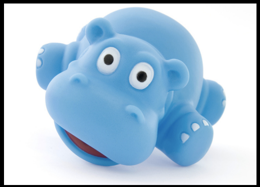
What gets returned?

```
{
  :red => [monkey],
  :orange => [fish],
  :yellow => [duck],
  :green => [dino],
  :blue => [train, hippo]
}
```

toys = []

```
toys.max_by do |toy|
      toy.name.length
end
```

What gets returned?



break,
then start on enumerable methods practice

What is the enumerable methods practice?