

Suggested response format (based on feedback from other students & Srdjan's blog post):

- **What does the code output? What are the return values?**
- **Answer the why behind the output/return:**
  - Focus only on the lines of code that deliver the output and return values.
- **Summarize what the problem demonstrates and why:** This problem demonstrates the concept of local variable scope/etc...
  - This can be at the beginning or end of your answer - personal preference.

**Using Markdown:** use ``backticks`` (Markdown formatting) to highlight variable names, methods, and lines you are referring to: On ``line 1`` we initialize the local variable...

**Always aim to answer: What does the following code output and return? Why? What concept does it demonstrate?**

Additional Practice Problems:

1. [Collection Methods from Lesson 4](#)
2. [Ruby Basics: Variable Scope](#)
3. [Ruby Basics: Return](#)

## Local Variable Scope

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = "Hello"
b = a
a = "Goodbye"
puts a
puts b
```

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = 4
```

```
loop do
  a = 5
  b = 3

  break
end

puts a
puts b
```

### Example 3

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = 4
b = 2

loop do
  c = 3
  a = c
  break
end

puts a
puts b
```

### Example 4

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def example(str)
  i = 3
  loop do
    puts str
    i -= 1
    break if i == 0
  end
end
```

```
example('hello')
```

### Example 5

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def greetings(str)
  puts str
  puts "Goodbye"
end

word = "Hello"

greetings(word)
```

[Problem link](#)

### Example 6

What does the following code return? What does it output? Why? What concept does it demonstrate?

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

counter = 0
sum = 0

loop do
  sum += arr[counter]
  counter += 1
  break if counter == arr.size
end

puts "Your total is #{sum}"
```

### Example 7

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = 'Bob'

5.times do |x|
```

```
a = 'Bill'
end

p a
```

### Example 8

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
animal = "dog"

loop do |_|
  animal = "cat"
  var = "ball"
  break
end

puts animal
puts var
```

## Variable Shadowing

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = 4
b = 2

2.times do |a|
  a = 5
  puts a
end

puts a
puts b
```

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
n = 10

1.times do |n|
  n = 11
end

puts n
```

### Example 3

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
animal = "dog"

loop do |animal|
  animal = "cat"
  break
end

puts animal
```

## Object Passing/Variables As Pointers

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = "hi there"
b = a
a = "not here"
```

What are a and b?

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = "hi there"
b = a
a << ", Bob"
```

What are a and b?

### Example 3

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = [1, 2, 3, 3]
b = a
c = a.uniq
```

What are a, b, and c? What if the last line was `c = a.uniq!`?

### Example 4

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def test(b)
  b.map {|letter| "I like the letter: #{letter}"}
end

a = ['a', 'b', 'c']
test(a)
```

What is `a`? What if we called `map!` instead of `map`?

### Example 5

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = 5.2
b = 7.3

a = b

b += 1.1
```

What is `a` and `b`? Why?

### Example 6

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def test(str)
  str += '!'
  str.downcase!
end
```

```
test_str = 'Written Assessment'
test(test_str)

puts test_str
```

[Link](#) to explanation of examples below

#### Example 7

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def plus(x, y)
  x = x + y
end

a = 3
b = plus(a, 2)

puts a
puts b
```

#### Example 8

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def increment(x)
  x << 'b'
end

y = 'a'
increment(y)

puts y
```

#### Example 9

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def change_name(name)
  name = 'bob'      # does this reassignment change the object outside the
method?
end

name = 'jim'
change_name(name)
puts name
```

#### Example 10

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def cap(str)
  str.capitalize!   # does this affect the object outside the method?
end

name = "jim"
cap(name)
puts name
```

#### Example 11

What is `arr`? Why? What concept does it demonstrate?

```
a = [1, 3]
b = [2]
arr = [a, b]
arr

a[1] = 5
arr
```

#### Example 12

[Link to example](#)



```

arr1 = ["a", "b", "c"]
arr2 = arr1.dup
arr2.map! do |char|
  char.upcase
end

puts arr1
puts arr2

```

## Object Mutability/Mutating Methods

[Here's the article](#) with some of the below examples

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

```

def fix(value)
  value.upcase!
  value.concat('!')
  value
end

s = 'hello'
t = fix(s)

```

What values do `s` and `t` have? Why?

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```

def fix(value)
  value = value.upcase
  value.concat('!')
end

s = 'hello'
t = fix(s)

```

What values do `s` and `t` have? Why?

### Example 3

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def fix(value)
  value << 'xyz'
  value = value.upcase
  value.concat('!')
end

s = 'hello'
t = fix(s)
```

What values do `s` and `t` have? Why?

### Example 4

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def fix(value)
  value = value.upcase!
  value.concat('!')
end

s = 'hello'
t = fix(s)
```

What values do `s` and `t` have? Why?

### Example 5

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def fix(value)
  value[1] = 'x'
  value
end

s = 'abc'
t = fix(s)
```

What values do `s` and `t` have? Why?

### Example 6

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def a_method(string)
  string << ' world'
end

a = 'hello'
a_method(a)

p a
```

### Example 7

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
num = 3

num = 2 * num
```

### Example 8

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = %w(a b c)
a[1] = '-'
p a
```

### Example 9

[Link to page with #9 & #10](#)

```
def add_name(arr, name)
  arr = arr + [name]
end

names = ['bob', 'kim']
add_name(names, 'jim')
puts names
```

### Example 10

```
def add_name(arr, name)
```

```
    arr = arr << name
  end

  names = ['bob', 'kim']
  add_name(names, 'jim')
  puts names
```

## Each, Map, Select

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

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

array.select do |num|
  puts num if num.odd?
end
```

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

arr.select { |n| n.odd? }
```

### Example 3

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_array = arr.select do |n|
  n + 1
end
p new_array
```

#### Example 4

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_array = arr.select do |n|
  n + 1
  puts n
end
p new_array
```

#### Example 5

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
words = %w(jump trip laugh run talk)

new_array = words.map do |word|
  word.start_with?("t")
end

p new_array
```

#### Example 6

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

arr.each { |n| puts n }
```

#### Example 7

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

incremented = arr.map do |n|
  n + 1
end
p incremented
```

### Example 8

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_array = arr.map do |n|
  n > 1
end
p new_array
```

### Example 9

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_array = arr.map do |n|
  n > 1
  puts n
end
p new_array
```

### Example 10

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = "hello"

[1, 2, 3].map { |num| a }
```

### Example 11

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
[1, 2, 3].each do |num|
  puts num
end
```

## Other Collection Methods

[Link to all examples below](#)

### Example 1

```
[1, 2, 3].any? do |num|  
  num > 2  
end
```

### Example 2

```
{ a: "ant", b: "bear", c: "cat" }.any? do |key, value|  
  value.size > 4  
end
```

### Example 3

```
[1, 2, 3].all? do |num|  
  num > 2  
end
```

### Example 4

```
{ a: "ant", b: "bear", c: "cat" }.all? do |key, value|  
  value.length >= 3  
end
```

### Example 5

```
[1, 2, 3].each_with_index do |num, index|  
  puts "The index of #{num} is #{index}."  
end
```

### Example 6

```
{ a: "ant", b: "bear", c: "cat" }.each_with_object([]) do |pair, array|  
  array << pair.last  
end
```

### Example 7

```
{ a: "ant", b: "bear", c: "cat" }.each_with_object({}) do |(key, value), hash|
  hash[value] = key
end
```

### Example 8

```
odd, even = [1, 2, 3].partition do |num|
  num.odd?
end

p odd
p even
```

## Truthiness

### Example 1

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
a = "Hello"

if a
  puts "Hello is truthy"
else
  puts "Hello is falsey"
end
```

### Example 2

What does the following code return? What does it output? Why? What concept does it demonstrate?

```
def test
  puts "written assessment"
end

var = test
```



```
if var
  puts "written assessment"
else
  puts "interview"
end
```

