SWINBURNE UNIVERSITY OF TECHNOLOGY

Introduction to Programming

Week 5, Topic 2: Parameters, pointers, variables and the stack.

This topic covers:

- Parameter passing
- Memory management
  - Stack
  - Heap
- Garbage Collection
- Passing by Reference Example
- This week's tasks – an further intro to arrays.

Parameter Passing

- Parameters can be:
  - Pass by Reference; OR
  - Pass by Value

Pass by Value

- Pass by value takes a copy of the variable and passes it to the called function or procedure.

```
MacBook-Pro:Lecture5Code mmitchell$ ruby swap.rb
a is: 100
b is: 50
In swap a is: 50
In swap b is: 100
a is: 100
b is: 50
MacBook-Pro:Lecture5Code mmitchell$
```

```ruby
def swap(a, b)
  c = a
  a = b
  b = c

  puts "In swap a is: " + a.to_s
  puts "In swap b is: " + b.to_s

end

def main

  a = 100
  b = 50

  puts "a is: " + a.to_s
  puts "b is: " + b.to_s

  swap(a, b)

  puts "a is: " + a.to_s
  puts "b is: " + b.to_s

end
```

Pass by Value

- Also for strings in Ruby:

```
MacBook-Pro:Lecture5Code mmitchell$ ruby string_swap.rb
a is: This is string one
b is: This is string two
In swap a is: This is string two
In swap b is: This is string one
a is: This is string one
b is: This is string two
MacBook-Pro:Lecture5Code mmitchell$
```

```ruby
def swap(a, b)
  c = a
  a = b
  b = c

  puts "In swap a is: " + a.to_s
  puts "In swap b is: " + b.to_s

end

def main
  a = "This is string one"
  b = "This is string two"

  puts "a is: " + a.to_s
  puts "b is: " + b.to_s

  swap(a, b)

  puts "a is: " + a.to_s
  puts "b is: " + b.to_s
end

main
```

Pass by Reference

- For strings in Ruby:

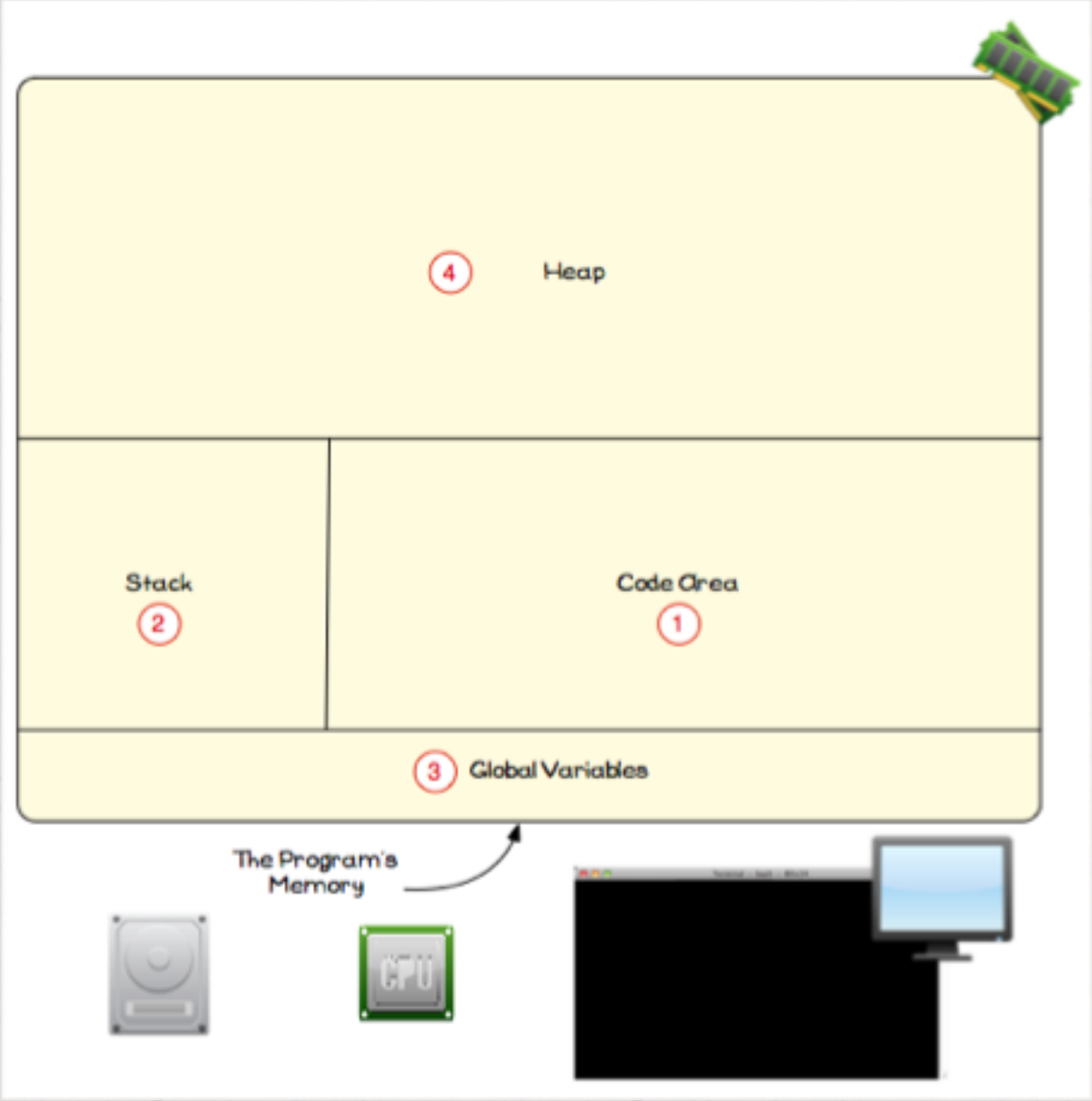
```
MacBook-Pro-6:Week5Code mmitchell$ ruby string_swap_reference.rb
a is: This is string one
b is: This is string two
In swap a is: This is string two
In swap b is: This is string one
a is: This is string two
b is: This is string one
MacBook-Pro-6:Week5Code mmitchell$
```

```ruby
class My_string
    attr_accessor :text
    def initialize(arg)
        @text = arg
    end
end

def swap(a, b)
  c = My_string.new("nothing")
  c.text = a.text
  a.text = b.text
  b.text = c.text

  puts "In swap a is: " + a.text
  puts "In swap b is: " + b.text
end

def main
  a = My_string.new("This is string one")
  b = My_string.new("This is string two")

  puts "a is: " + a.text
  puts "b is: " + b.text

  swap(a, b)

  puts "a is: " + a.text
  puts "b is: " + b.text
end

main
```

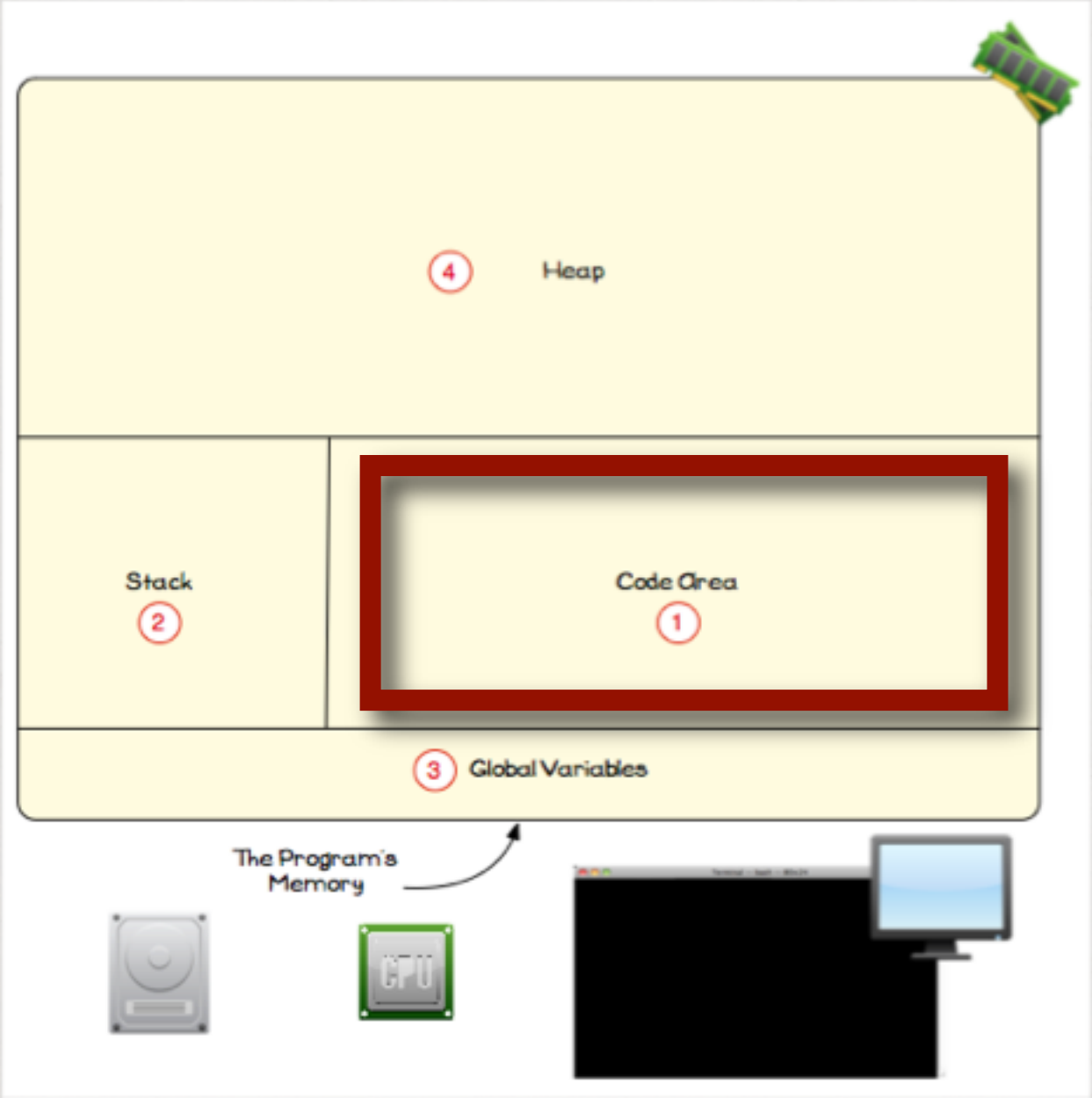
Pass by Reference

- But in general Ruby is Pass-by-Reference.
- This means that a reference is passed to the calling function and procedure.
- Before we look at an another example, lets look at how memory is managed.

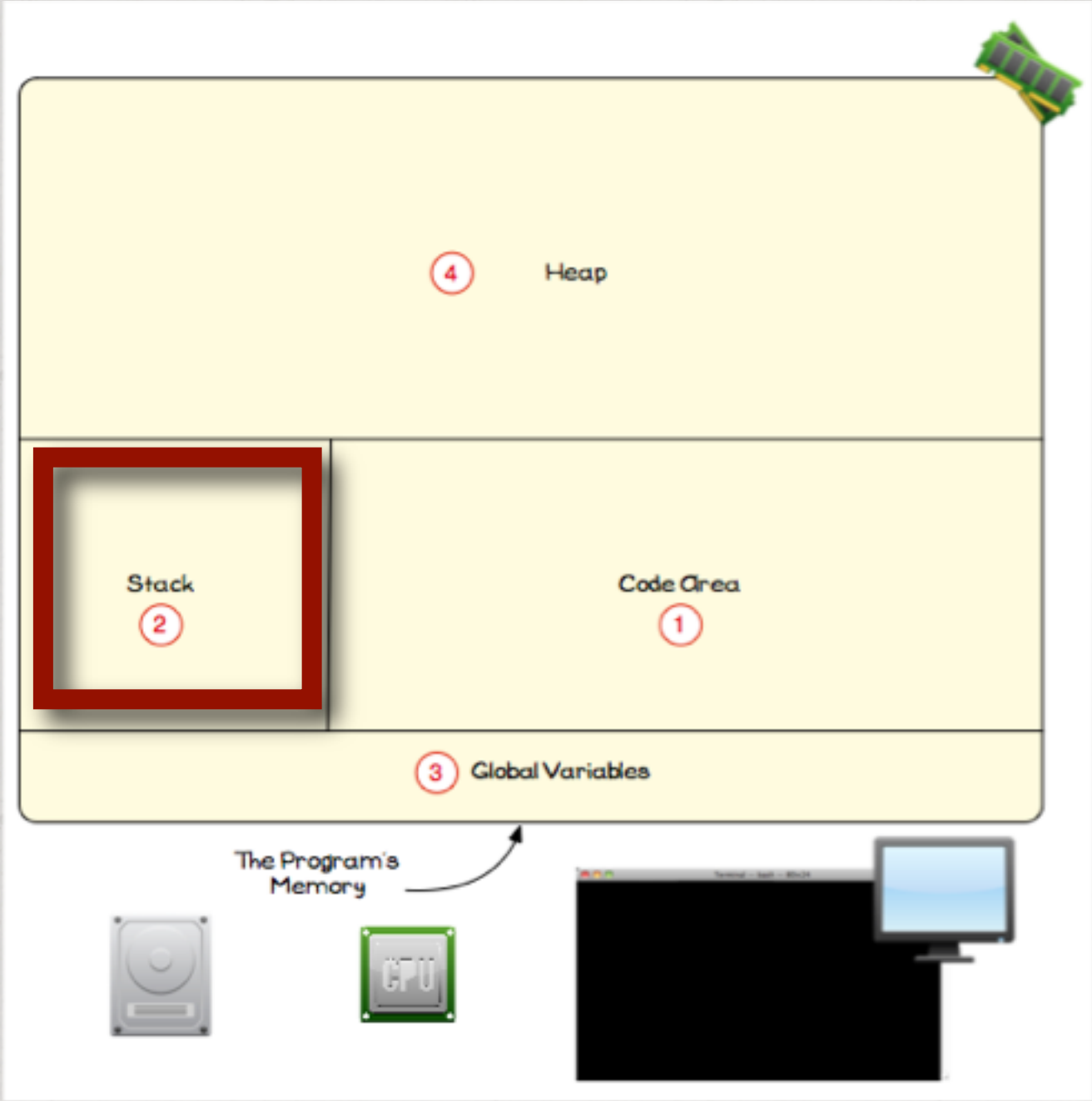
Memory organisation...



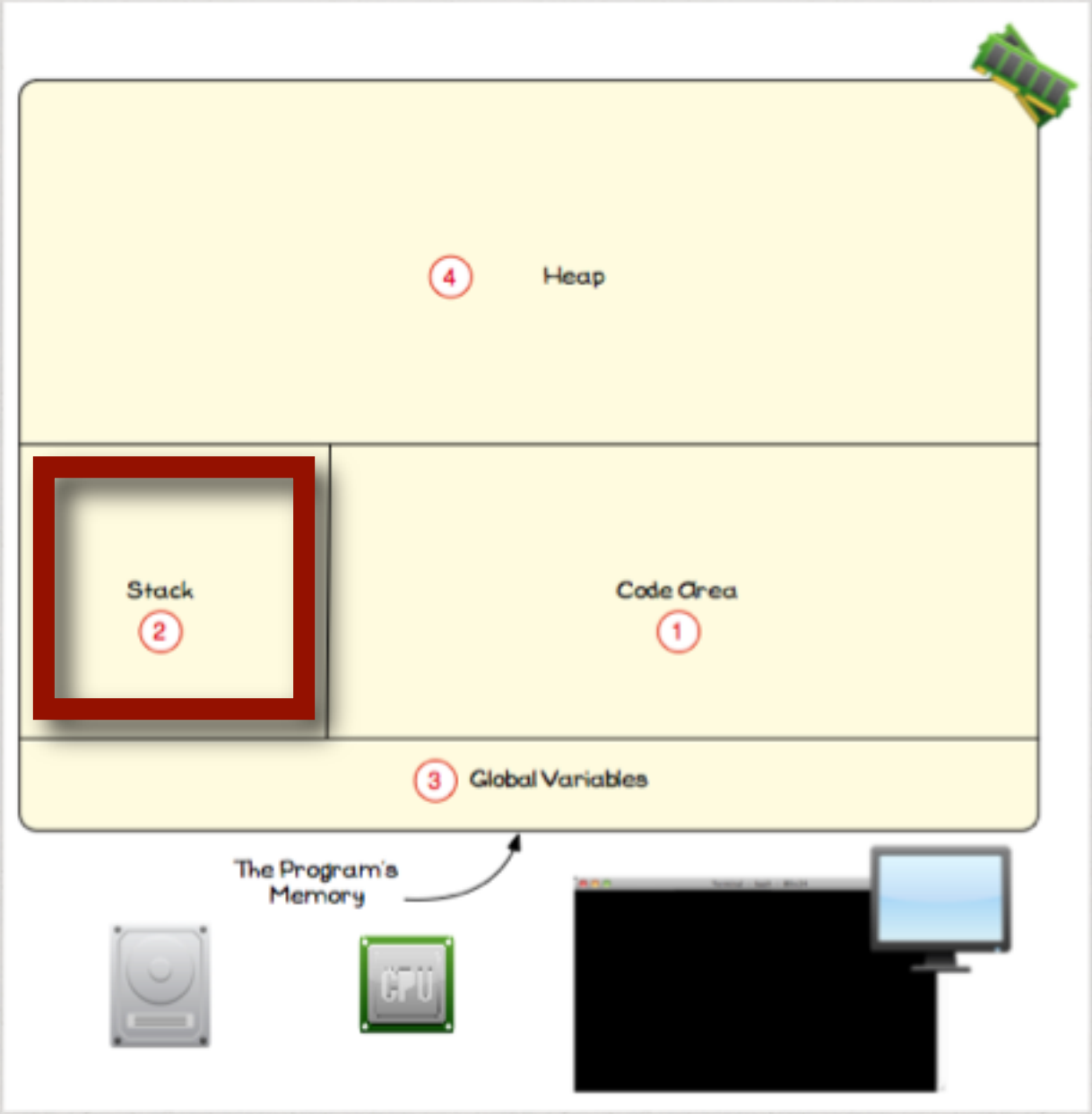
The code area stores the machine code from your program



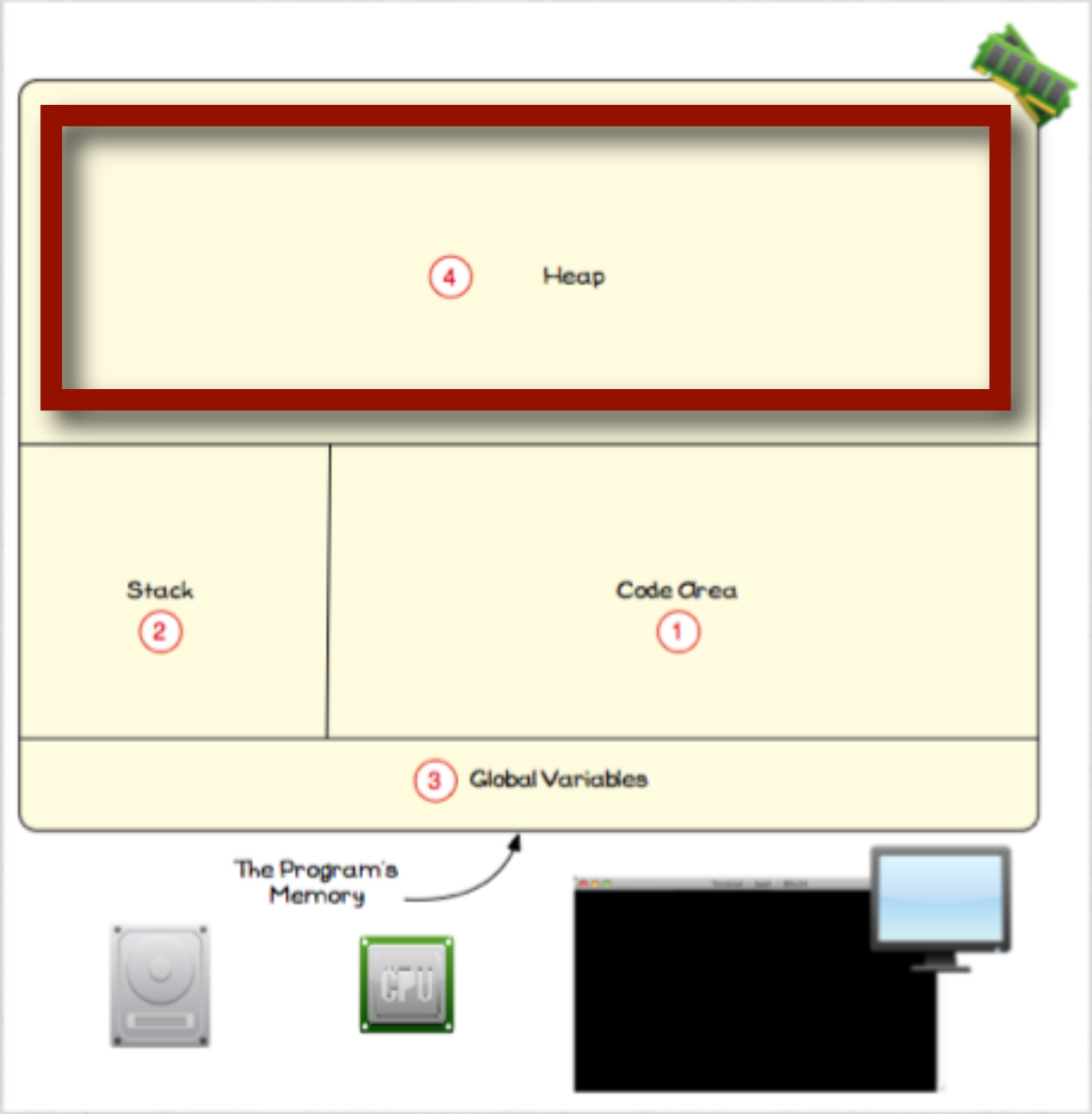
The stack stores frames that manage the function and procedure calls



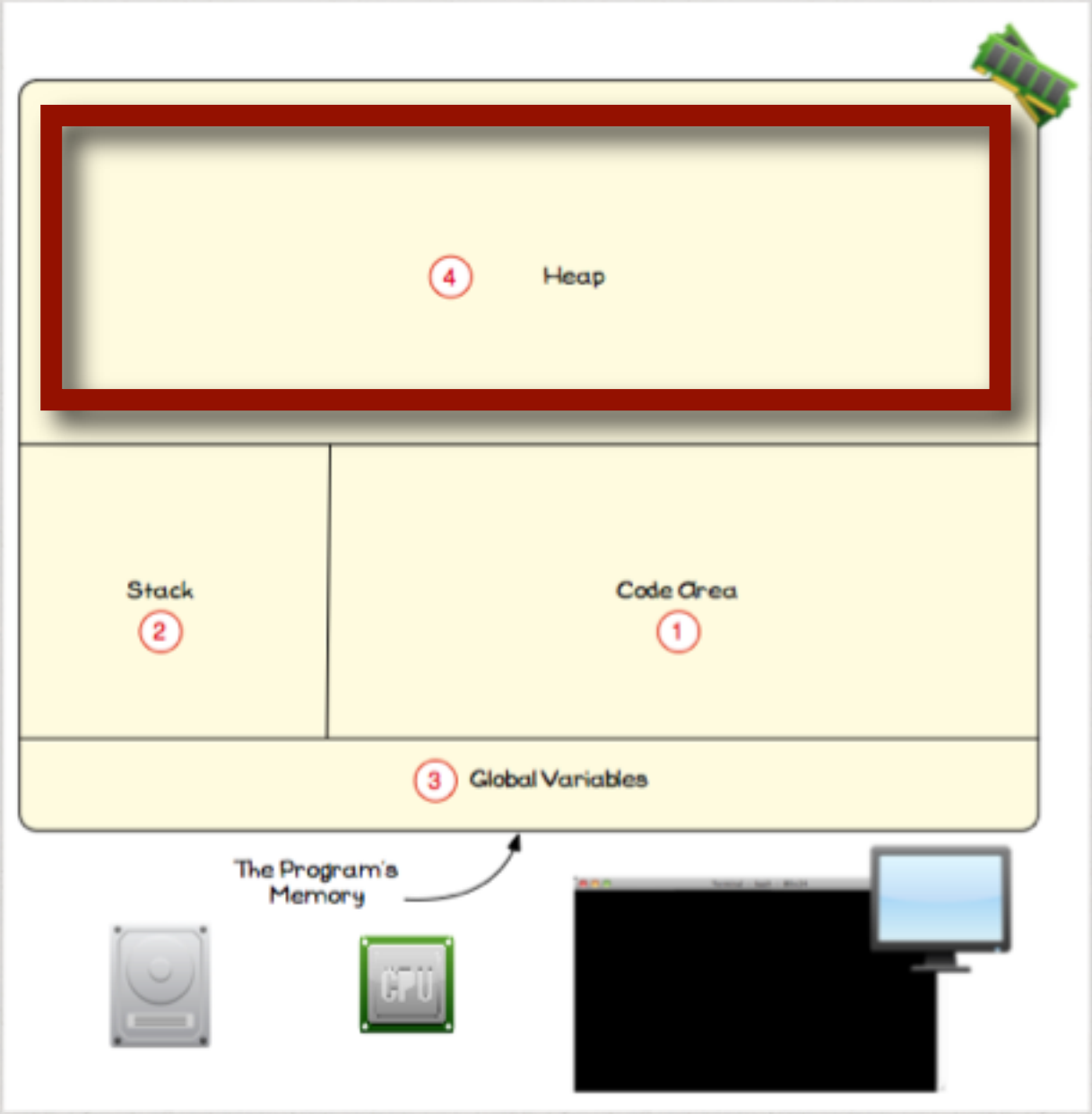
Everything on the stack has a fixed size, with the computer managing this memory allocation



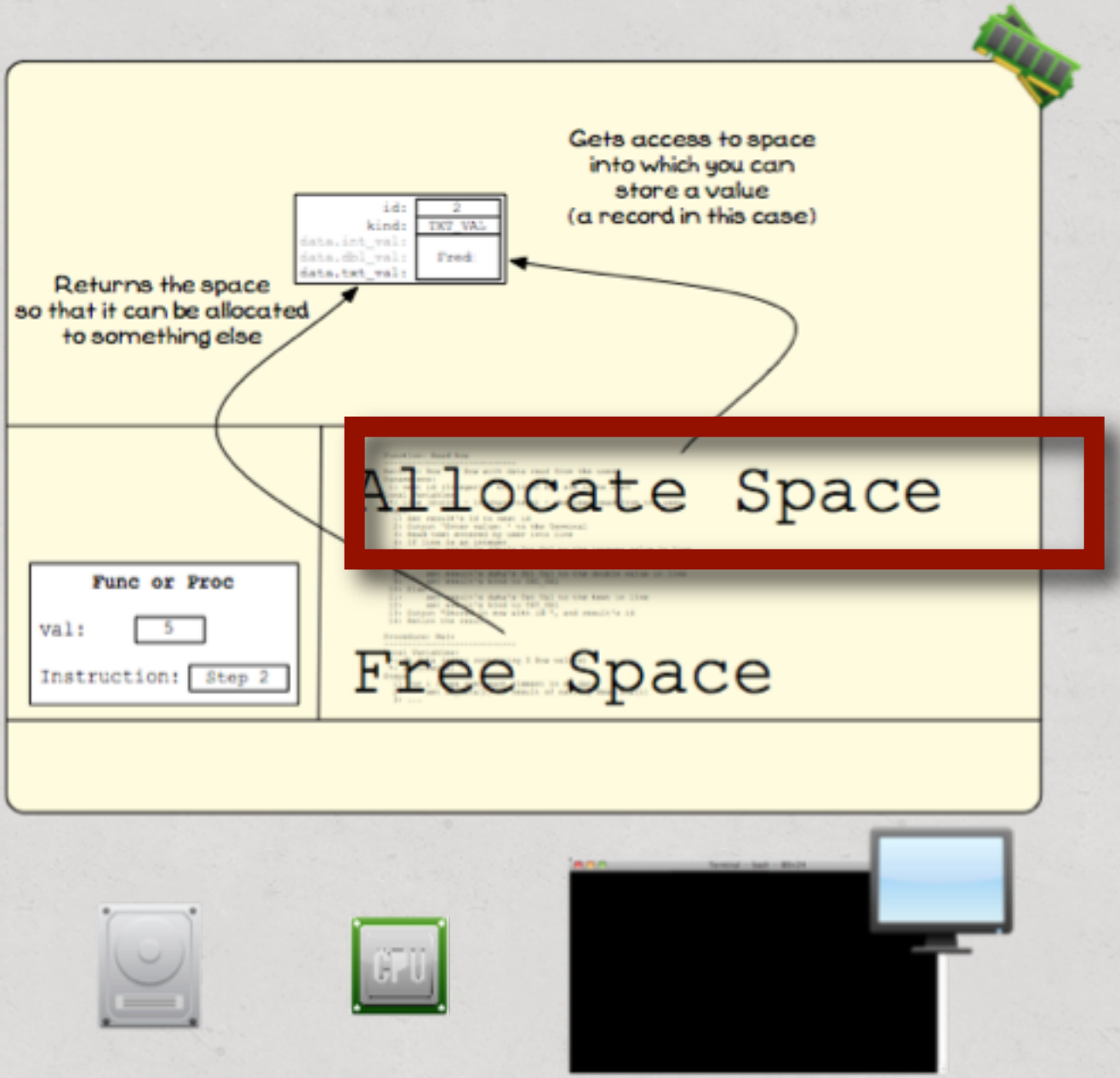
The top area is called the **heap**.



The heap stores dynamically allocated memory.

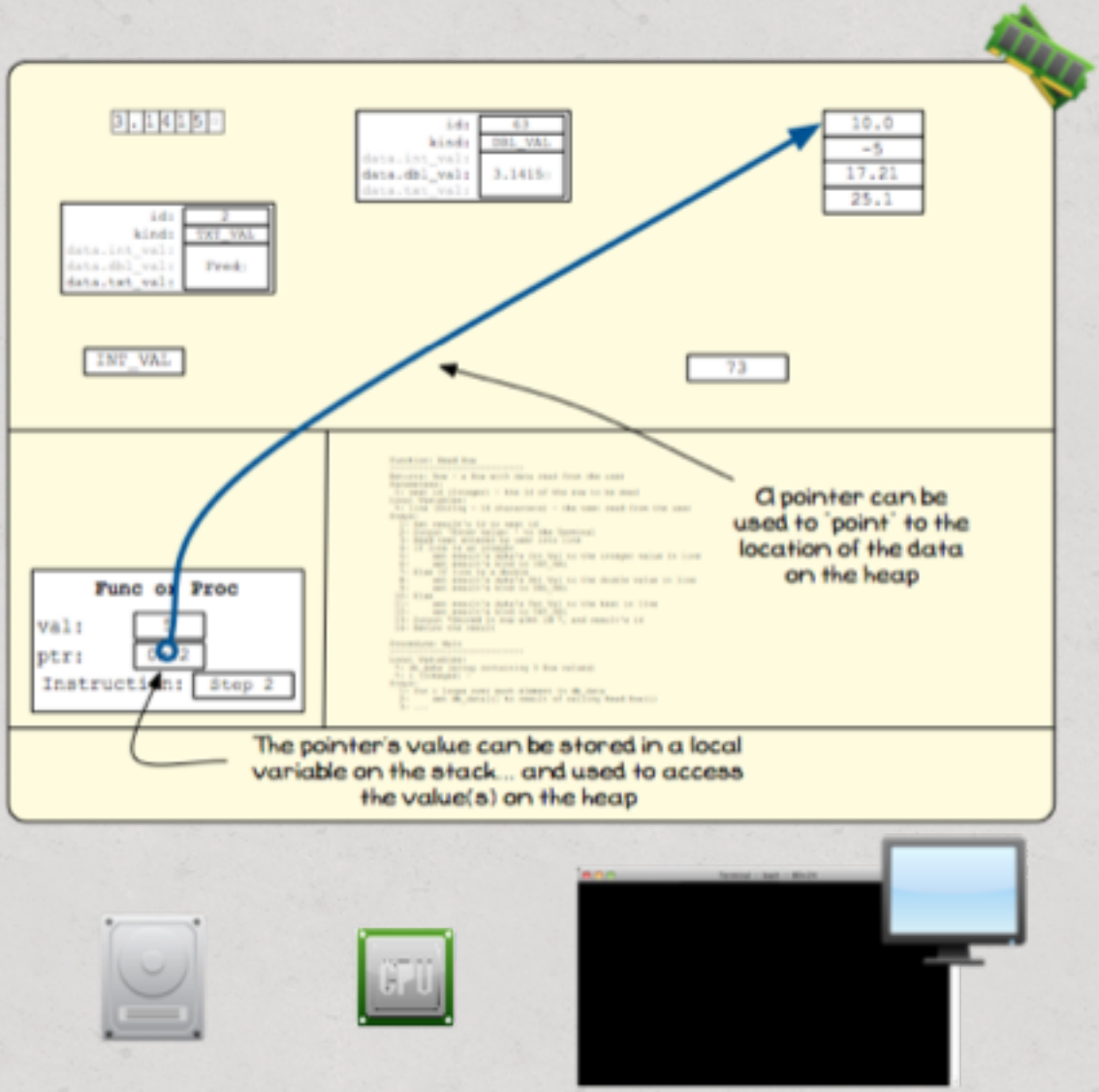


You can ask to be allocated space.

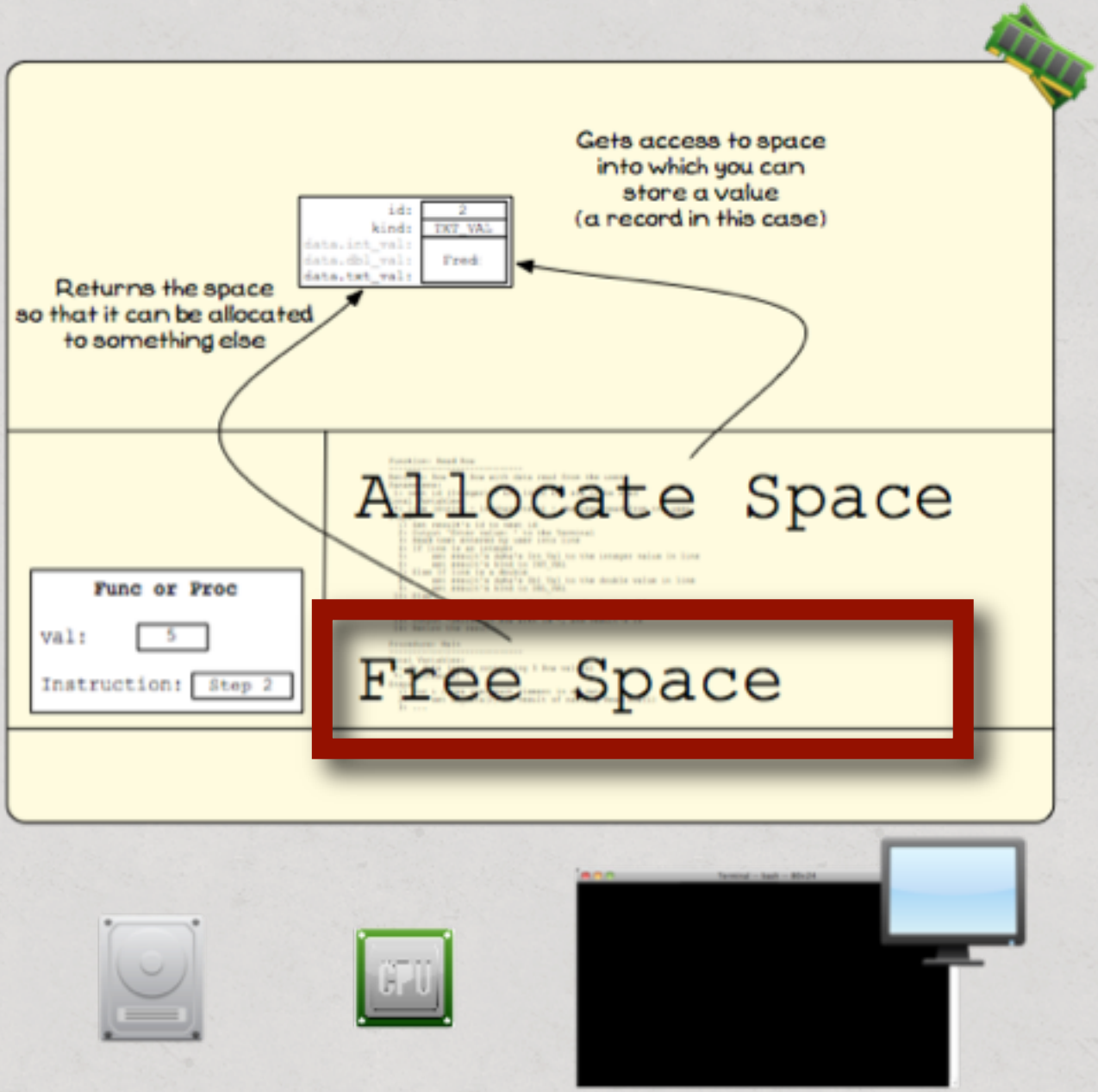


Pointers and references

For variables on the heap you need a **pointer** or **reference** to the variable (i.e to its memory location)



In the C language you also need to free the memory when you are finished with a variable.



Managing Memory

- In Ruby stack variables are created as soon as the variable name is encountered in the code.
- But you are responsible for allocating space on the heap.
- You typically do this in Ruby by using the keyword **new().**
- In Ruby freeing memory is managed by the Garbage Collector.

Garbage Collection

- http://timetobleed.com/garbage-collection-slides-from-la-ruby-conference/
- https://thorstenball.com/blog/2014/03/12/watching-understanding-ruby-2.1-garbage-collector/
- http://www.atdot.net/~ko1/activities/rubyconf2013-ko1_pub.pdf

Example:

- Let us look at the Lights example, where a switch is shared by two lights.
- Clicking the light passes in a different **reference** for the switch to work with.
- Before we look at the example, lets look at this week's tasks.

Tasks for this week:

- Music Records
- Track File Handling
- Button Test

Now the Example:

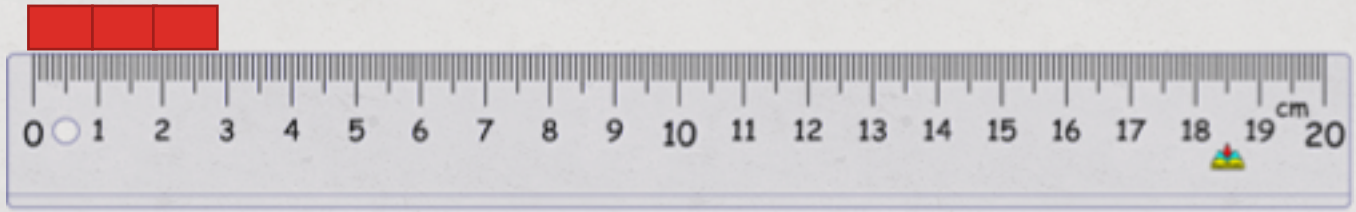
Reminder on Arrays

- We covered arrays in detail in Week 4.
- Remember how to put things in, and how to get things out.

What is an array?

- It holds multiple items (like a list)
- Eg: [‘Fred’, ‘Sam’, ‘Jenny’, ‘Jill’]

Consider our ‘bricks” example:



Put things in an array:

- It holds multiple items (like a list)
- Eg:

```
['Fred', 'Sam', 'Jenny', 'Jill']
```

- We can add to the array in Ruby as follows:

```
test_array = ['Fred', 'Sam', 'Jenny', 'Jill']

test_array << 'John'
```

This gives:

['Fred', 'Sam', 'Jenny', 'Jill', 'John']

How to get things out?

Using:

```
test_array = ['Fred', 'Sam', 'Jenny', 'Jill']
```

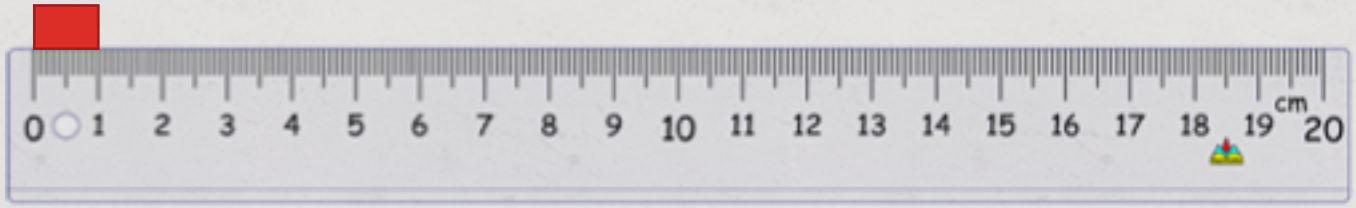
If we do:

```
puts test_array[0]
```

Will output:

```
MacBook-Pro-6:Week5Code mmitchell$ ruby test_array.rb
Fred
```

NB: Arrays start from zero:



End of Topic 2.

Terminology:

- Reference
- Pointer
- Pass by value
- Pass by reference