Ruby Case Expressions

&

Ruby Iterators (for, while loops)

Kaichi, Elmer, (Johnathan)

Loop

- Simple Loop
- Control Loop Execution (do end loop)
- While loop
- Do/While Loop
- Until Loop
- For Loop

Simple Loop

The simplest way to create a *loop* in Ruby is using the *loop* method.

```ruby
# loop_example.rb

loop do
  puts "This will keep printing until you hit Ctrl
end
```

```
This will keep printing until you hit Ctrl + c
This will keep printing until you hit Ctrl + c
This will keep printing until you hit Ctrl + c
This will keep printing until you hit Ctrl + c
This will keep printing until you hit Ctrl + cInter
from (pry):2:in `puts'
[2] pry(main)>
```

Control Loop Execution

The *break* keyword allows us to exit a loop at any point, so any code after a *break* will not be executed.

```
3  i = 0
4  loop do
5    i += 2
6    puts i
7    if i == 10
8      break       # this will cause execution to exi
9    end
10 end
```

```
1  $ ruby conditional_loop.rb
2  2
3  4
4  6
5  8
6  10
```

While Loops

- A while loop is a given that parameter confirms to a boolean. Once the boolean becomes false, the while loop is not executed again.

```ruby
# countdown.rb

x = gets.chomp.to_i

while x >= 0
  puts x
  x = x - 1
end

puts "Done!"
```

Do/While Loop

- A **do/while loop** works in a similar way to a while loop; one important difference is that the code within the loop gets executed one time, prior to the conditional check to see if the code should be executed. In a "do/while" loop, the conditional check is placed at the end of the loop as opposed to the beginning.

```ruby
# perform_again.rb

loop do
  puts "Do you want to do that again?"
  answer = gets.chomp
  if answer != 'Y'
    break
  end
end
```

Do/While Loop Example 2

```ruby
begin
  puts "Do you want to do that again?"
  answer = gets.chomp
end while answer == 'Y'
```

Until Loop

The *until loop* is simply the opposite of the *while loop*. You can substitute it in order to phrase the problem in a different way.

```
irb(main):001:0> x = gets.chomp.to_i
5
=> 5
irb(main):002:0> until x < 0
irb(main):003:1> puts x
irb(main):004:1> x -= 1
irb(main):005:1> end
5
4
3
2
1
0
=> nil
```

Ruby Iterators Example

While Loop

```ruby
arr = ["John", "George", "Paul", "Ringo"]
i = 0

while arr[i]
    puts arr[i]
    i += 1
end
```

Ruby Iterators Example

While as a modifier

```ruby
arr = ["John", "George", "Paul", "Ringo"]
i = -1

puts arr[i += 1] while arr[i]
```

Ruby Iterators Example

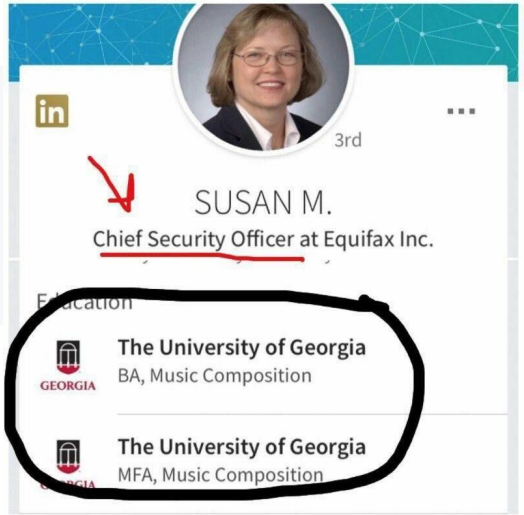
For loop

```ruby
arr = ["John", "George", "Paul", "Ringo"]

for item in arr
    puts item
end
```

Ruby Iterators Example

For loop with string interpolation

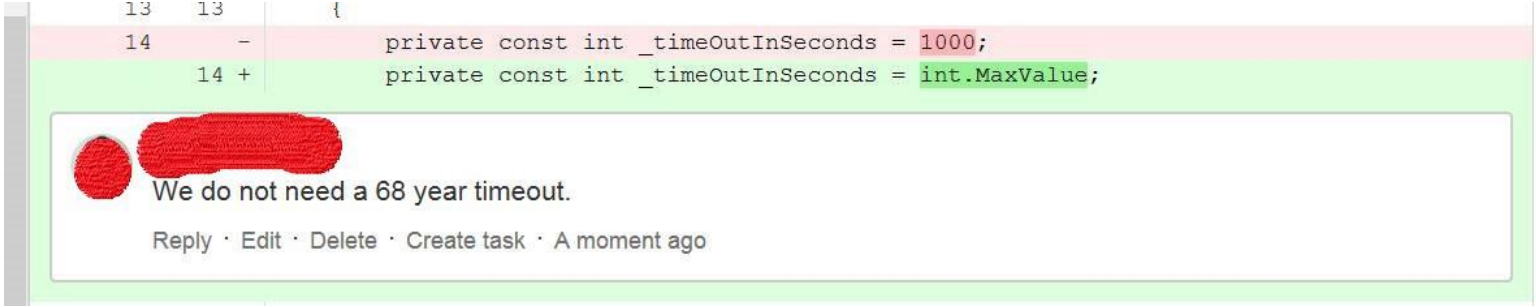
```ruby
joe = { :name => "Joe", :age => 30, :job => "plumber" }

for key, val in joe
    puts "#{key} is #{val}"
end
```



Ruby Case Expression

Normal

```ruby
hour = 15

case
when hour < 12
    puts "Good Morning"
when hour > 12 && hour < 17
    puts "Good Afternoon"
else
    puts "Good Evening"
end
```



Ruby Case Expression

v2

```ruby
hour = 15

message = case
    when hour < 12
        "Good Morning"
    when hour > 12 && hour < 17
        "Good Afternoon"
    else
        "Good Evening"
    end

puts message
```

MacOS

Linux

Windows