Cheatography

Ruby Cheat Sheet

by litoblee via cheatography.com/19693/cs/2671/

Three Basic Data Types

```
my_num = 25 #numbers
my_boolean = true #Booleans
my_string = "Ruby" #Strings
```

Loop and Next

```
i = 0
loop do
i += 1
next if i % 2 == 1
puts "#{i}"
break if i == 10
end
```

upto

```
95.upto(100) { |num| print num, " " }
=> 95 96 97 98 99 100
"L".upto("P") { |letter| print letter, " " }
=> L M N O P
```

Proc

```
multiples_of_3 = Proc.new do |n|
n % 3 == 0
end
(1..10).to_a.select(&multiples_of_3)
#==> [3, 6, 9]
```

users string. (converts "s" to "th")

```
if user_input.include? "s"
user_input.gsub!(/s/, "th")
```

users string. (converts "s" to "th") (cont)

```
else
puts "Nothing to do here!"
end
```

use a splat argument

```
def whats_up(greeting, *bros)
bros.each { |bro| puts "#{greetings}, #{bro}!" }
end
what_up("What up", "Justin", "Ben", "Kevin Sorbo")
```

use .respond_to

```
[1, 2, 3].respond_to?(:push)
```

would return true, since you can call .push on an array object.

```
[1, 2, 3].respond_to?(:to_sym)
```

would return false, since you can't turn an array into a symbol.

Array to string

```
numbers_array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
strings_array = numbers_array.map(&:to_s)
```

Use a counter and Until Loop to count to ten

```
counter = 1
until counter == 11
puts counter
counter = counter + 1
end
```

combined comparison operator

It's <=>. It returns 0 if the first operand (item to be compared) equals the second, 1 if first operand is greater than the second, and -1 if the first operand is less than the second.

Collect

```
fibs = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
doubled_fibs = fibs.collect { |n| n * 2 }
#==>[2, 2, 4, 6, 10, 16, 26, 42, 68, 110]
```

Lambda

```
lam = lambda { |x| puts x*2 }
[1,2,3].each(&lam)
lam = lambda { puts "Hello World" }
lam.call
```

inclusive and exclusive ranges

```
for num in 1..10 #includes 10
for num in 1...10 #excludes 10
```

Select

```
movie_ratings = {
primer: 3.5,
the_matrix: 5,
uhf: 1,
}
good_movies = movie_ratings.select { |k, v| v > 3 }
=> {:primer=>3.5, :the_matrix=>5}
```

Yield

```
def yield_name(name)
puts "In the method! Let's yield."
yield("Kim")
puts "In between the yields!"
yield(name)
puts "Block complete! Back in the method."
end
yield_name("Scott") { |n| puts "My name is #{n}" }
#==>
In the method! Let's yield.
My name is Kim
In between the yields!
My name is Scott
Block complete! Back in the method.
```