Blocks and Sorting

Ruby Combined Comparison Operator

In Ruby, the *combined comparison operator*, `<=>` , also known as the spaceship operator is used to compare two objects. It returns `0` if the first operand equals the second, `1` if the first operand is greater than the second, and `-1` if the first operand is less than the second.

```
puts "Keanu" <=> "Adrianna" # The first letters of each word are compared in ASCII order and since "K" comes after "A", 1 is printed.

puts 1 <=> 2 # -1

puts 3 <=> 3 # 0

#<=> can also be used inside of a block and to sort values in descending order:
my_array = [3, 0, 8, 7, 1, 6, 5, 9, 4]
my_array.sort! { |first_num, second_num| second_num <=> first_num }
print my_array
#Output => [9, 8, 7, 6, 5, 4, 3, 1, 0]
```

Ruby Method Splat

In a Ruby method, a splat (`*`) operator is used to indicate that a parameter can have an unknown number of arguments.

```
#The * preceding the parameter "clubs" allows for multiple arguments to be passed into the method when you actually call it.
def extra_curriculars(*clubs)
  clubs.each { |club| puts "After school, I'm involved with #{club}" }
end

extra_curriculars("chess club", "gymnastics", "anime club", "library services")

#Output
#After school, I'm involved with chess club
#After school, I'm involved with gymnastics
#After school, I'm involved with anime club
#After school, I'm involved with library services
```

Ruby Block Parameter

In Ruby, a method can take a *block* as a parameter. Passing a *block* to a method is a great way of abstracting certain tasks from the method and defining those tasks when we call the method.

```
# The block, {|i| puts i}, is passed the current array item each time it is evaluated. This block prints the item.
[1, 2, 3, 4, 5].each { |i| puts i }
```

Ruby Return

In Ruby, the `return` keyword is used to pass back a value from a method.

```
def generous_tip(bill)
  return bill * (0.25)
end

generous_tip(100) # 25

#In this example, the generous_tip method is returning the product of bill and 0.25. In order to see that value, a "puts" or "print" can be added before the method call.
```

Ruby Sort Method

In Ruby, the `.sort` array method is used to sort items in an array in ascending order (least to greatest).

```ruby
my_array = [3, 4, 8, 7, 1, 6, 5, 9, 2]
my_array.sort!
#Attaching an ! to the end of .sort or any other Ruby method modifies the original array.
print my_array
# => [1, 2, 3, 4, 5, 6, 7, 8, 9]
#If you didn't use !, print my_array returns the original array.
```

Ruby Method Parameters & Arguments

In Ruby, *parameters* are placeholders for real values or *arguments* passed into a method when it is called. When calling a method that requires parameters, arguments (ie. real values) must be passed in for those parameters.

```ruby
def square(num) # num is the parameter
  puts num ** 2
end

square(5) #5 is the argument
#Output => 25
```

Ruby method

A Ruby *method* is a reusable section of code written to execute a certain task. It is defined with the `def` keyword, followed by a method name, a method body, and ends with the `end` keyword:

```ruby
def greeting
  puts "Hello world!"
end

#In this example, the first line or header contains the keyword "def" and the method name. puts "Hello world!" is within the body of the method, which describes the certain task that the method carries out. It is also indented two spaces by convention. Following the body, the method ends with the end keyword.
```

Ruby Block

In Ruby, a *block* is a section of code defined within the keywords `do` and `end` or with curly braces `{}`. This is usually preceded by an integer followed by `.times` to indicate how many times the code is to be executed.

```ruby
2.times do
  puts "I'm a code block!"
end

#Output
#I'm a code block!
#I'm a code block!

3.times { puts "So am I!" }

#Output
#"So am I!"
#"So am I!"
#"So am I!"
```