Ruby Three

Object Inheritance

- In Ruby, one object can *inherit* from another object, taking all of its methods and attributes with it
- Inheritance is indicated at object definition, using the < operator

```
class Dog < Animal
```

- We would say the above out loud like this: "class Dog extends Animal"

Object Inheritance Example

```
class Animal
  attr_accessor :legs
end

class Dog < Animal
  def woof
    puts "WOOF"
  end
end

my_dog = Dog.new
my_dog.legs = 4
my_dog.woof
>"WOOF"
```

Exercise - Object Inheritance

- Create two objects. Have the second object inherit from the first
- Instantiate instances of each object and prove that the first object's methods can be accessed from an instance of the second object
- Use comments in your script to explain what you are doing

`super`

- In a child object, we can call all of the existing code from a parent method using the `super` keyword
- See the next slide for an example

super

```ruby
class User
  def create
    add_record(@name, @email)
    activate_account
  end
end

class Admin < User
  def create
    super
    add_permissions("Admin")
  end
end
```

Exercise

- Create an "object zoo". You should create a class that models a Zoo which has at least one attribute called exhibits. Inside of this attribute is an array of object instances extended from an Animal object that have their own unique traits depending on which animal they are.
- Create unique methods on all of your objects depending on what kind of animal they are supposed to emulate. You could use super and even have a method on Animal that is customized per specific animal objects.

The bang (!)

A bang reverses the boolean value

```
!true == false
```

.nil?

Checks to see if the referenced item is equal to nil, or nothing

unless

The opposite of if. Can be used as a one-liner, just like if can:

```
widget_counter = 3 unless a.nil?

widget_counter = 5 if widgets.length == 5
```

Recursive Methods

A recursive method is a method that calls itself.

```ruby
def can_i_have_your_number
  puts "Can I have your number?"
  number = gets
  can_i_have_your_number unless number.length > 1
end
can_i_have_your_number
```

Exercise

Write a recursive method to subtract 1 from a variable equal to 100 until the variable is equal to 0.

Exercise

Create a "choose your own adventure" style game using Ruby. Store characters as classes with unique attributes and methods. Use methods and recursive methods to simplify the logical flow of the game and the `gets` method to get user input.

Use this exercise to be sure you're familiar with advanced Ruby topics.