Object-Oriented Programming, Part I

Ruby Class Variables

In Ruby, *class variables* are attached to the class in which they are declared. A class variable should be declared with two `@` symbols preceding it.

```
class Child
  @@children = 0
  def initialize(name, birth_year)
    @name = name
    @birth_year = birth_year
    @@children +=1
  end

  def self.children_added
    return @@children
  end

end

naomi = Child.new("Naomi", 2006)
bertha = Child.new("Bertha", 2008)

puts Child.children_added # => 2
```

Ruby .new Method

In Ruby, a new class instance can be created by calling the `.new` method of the class. Arguments to the class' `initialize` method can be passed in to the `.new` call.

```
class Fighter
  def initialize(name, style, division, age)
    @name = name
    @style = style
    @division = division
    @age = age
  end
end

conor = Fighter.new("Conor", "mixed martial arts", "Welterweight", 31)
```

Ruby Instance Variable

In Ruby, the `@` symbol is used to signify an *instance variable*. Instance variables hold a value specific to each instance of that class, not to all members of the class itself.

```
class Student
  def initialize(name, grade)
    @name = name
    @grade = grade
  end
end

# In this example, name and grade are the instance variables.
```

Ruby initialize Method

In a Ruby *class*, an `initialize` method is used to generate new instances of the class. It is usually the first method of a class.

```
class Person
  def initialize
    # this code runs when a new instance is created
  end
end

#Every time Person.new is called, the initialize method of the Person class is called.
```

Ruby Class

A Ruby *class* is used to organize and model objects with similar attributes and methods.

```
class NewClass
  # code for this class
end


# A basic class definition consists of the class keyword, the name of the class in CamelCase (with the first letter capitalized) format, and an end keyword.
```

Ruby super Keyword

Ruby's built-in `super` keyword is used to directly access the attributes or methods of a superclass. This means a class with `super` will inherit the attributes or methods of a superclass.

```
class Trip
  def initialize(duration, price)
    @duration = duration
    @price = price
  end
end


class Cruise < Trip
  def initialize(duration, price)
    super
  end
end

spain_backpacking = Trip.new(14, 800.00)
carnival = Cruise.new(7, 2400.00)

#In this example, the Cruise class
inherits from the Trip class and all of
its attributes, including duration and
price, are carried over with the super
keyword.
```

codecademy

Ruby attr_reader attr_writer Methods

In Ruby, `attr_reader` and `attr_writer` are methods used to read and write variables, respectively.

```
class Student
  attr_reader :name
  attr_writer :name
  def initialize(name)
    @name = name
  end
end
#In this example, Ruby is able to both read and write the @name instance variable since it was passed to attr_reader and attr_writer as a symbol.

top_student = Student.new("Jyothi")
puts top_student.name # => Jyothi
#In classes with attr_reader, instance variables can be accessed using . notation

puts top_student.name # => Jyothi
top_student.name = "Anika"
puts top_student.name # => Anika
#In classes with attr_writer, instance variables can be reassigned using . notation
```

Print   Share