RUBY FOR PROGRAMMERS

This cheat-sheet accompanies the Bitwise Courses course on *Ruby For Programmers* by Huw Collingbourne.

Classes

Ruby objects are 'instances' of classes. Before an object can be created a Class must be defined. This is an example of the simplest possible class:

```
class MyClass
end
```

When a class descends from an ancestor, the ancestor class name is placed to the right of the `<` symbol. Here `MyClass` descends from `MyBaseClass`:

```
class MyClass < MyBaseClass
end
```

Methods

Class names begin with a capital letter. Methods (functions) should start with a lowercase letter after the `def` keyword. Both class and method definitions terminate with the keyword `end`. This class has single method:

```
class MyClass < MyBaseClass
  def mymethod
  end
end
```

Class and Instance variables

A class variable begins with `@@` (e.g. `@@xxx`) and is shared by all objects created from a class. An instance variable begins with `@` (e.g. `@yyy`) and can be accessed only by a specific object.

Creating objects

An object is created by calling the `new` method following the class name and a dot. The newly created object can then be assigned to a variable.

EXAMPLE:

```
ob = MyClass.new
```

Constructors

When a class contains a method named `initialize` this will be automatically called when an object is created using the `new` method. It is a good idea to use an `initialize` method to set the values of an object's instance variables.

EXAMPLE:

```
class Treasure
  def initialize( aName, aDescription )
    @name         = aName
      @description  = aDescription
  end
end
```

If the `initialize` method takes arguments, these must be passed from `new` like this:

```
t1 = Treasure.new("Sword", "an Elvish weapon forged of gold")
```

Parentheses are optional, and this is also acceptable:

```
t1 = Treasure.new "Sword", "an Elvish weapon forged of gold"
```