Cheatography

Ruby Cheat Sheet

by Ruby Gray via cheatography.com/68488/cs/17257/

General Remarks about the Language

IRB is an interactive Ruby Shell good for starting out. ($ irb)

Ruby does not need to be compiled (most of the time) since it is interpreted.

Everything in Ruby is an object.

Ruby variables don't have types - only objects do.

Ruby wants you to omit ()'s and ;'s

Ruby will make you happy so enjoy.

Control Structure

```
if [conditional] ...
else ... end
----------------------
---------
[conditional] ? true block
... : false block ...
----------------------
---------
if [conditional] ...
elsif [conditional] ...
else ... end
----------------------
---------
while [conditional] ...
end
----------------------
---------
until [conditional] ...
end
----------------------
---------
case someVar
when [condition] ...
when [condition] ...
(as many whens as needed)
else ... end
----------------------
---------
```

Control Structure (cont)

```
someVar.each do |x| ...
end
someVar.each {|x| ... }
----------------------
---------
for each x in someVar do
... end
```

Variable Classifications

variable_name	Local Variable
VARIABLE	Constant Variable
@variable_name	Instance Variable
@@variable_name	Class Variable
=	Assignment

Everything is an object so variables do not have explicit data types

Strings

.length	Length of the String
.count(param)	How many times the param appears in the String
.insert(pos, param)	Inserts the param in the position of the String
.upcase	Converts all characters to uppercase
.downcase	Converts all characters to lowercase
.swapcase	Converts all uppercase characters lowercase & lowercase to uppercase
.reverse	Reverses the order of the characters

Strings (cont)

.split	Breakes up a String on whitespace and stores all those strings in an array
.chop	Removes the last character
.strip	Removes all whitespace, tabs, new lines & carriage returns
.chomp	Removes the last character if it's a new line or carriage return
[start, end]	Returns a substring
.to_i	Converts to integer
+	Concatenates strings
.index(position)	Returns the character in the specified position
.clear	Removes all content

Function Structure

```
def methodName ... end
def methodName (param1,
param2) ... end
```

No Return Statements.
No need for ()'s with no parameters.

Operators

Arithmetic	Operators
+	Addition
-	Subtraction
*	Multiplication
/	Division

Operators (cont)

%	Modulus
**	Exponent
Compare	Operators
==	Values Equal?
!=	Values Not Equal?
>	Left op greater than Right op?
<	Left op Less than Right op?
>=	Left op greater than or equal to Right op?
<=	Left op less than or equal to Right op?
<=>	Spaceship: returns 0 if ops are equal, 1 if Left op is greater than Right op and -1 if Left op is less than Right op
===	A case comparative for when control structure
.eql?	Values Equal for both type and value?
.equal?	Values are the same object?
Logical	Operators
and, &&	AND
or, ||	OR
not, !	NOT

Arrays

array = Array.new(lengthParam)	Creates array where each element is NIL
array = [element1, element2, ...]	Creates array with the specified elements
array[index]	Returns the element value at index
array.length	Returns the size of the array
array.push param <<	Adds the params as separate elements to the end of the array
array.pop	Removes the element from the end
array.unshift param	Adds the params as separate elements to the front
array.shift	Removes the element from the front
array.reverse	Reverses the order of elements
array.shuffle	Randomly shuffles up order of elements
array.sort	Sorts the array of elements
array.include? param	Returns true if the param exists in the array
array.uniq	Returns an array of only the unique elements

Arrays (cont)

array.fill param	Sets all the array elements to the param
array.each {|x| ...}	Iterates over each element in the array
array.each_index{|i| ...}	iterates over each index in the array

A single array can hold elements of different object types.

Ranges

range = Range.new (start, end)	Creates a new Range from the starting point to the end point
range = start..end	Creates a range from start to end inclusive
range = start...end	Creates a range from start to end exclusive
range.to_a	Converts a range to an array
range.each	Iterators through each element
range.include? (param)	Returns true if the param exists in the Range
range.last param	Returns the last element. Param can be added to provide more than just the last.

Above each function can be used with (start..end) or (start...end) as well

Hashes

map = Hash.new	Creates an empty hash map
map = Hash.new(default)	Creates an empty hash map where if key or value cannot be found, default value is returned.
map = Hash["key1" => value1, "key2" => value2, ...]	Creates a hash map with 2 key-value pairs
map = Hash["key1" => value1, "key2" => value2, ...]	Creates a hash map with 2 key-value pairs
map["key3"] = value3	Adds a key-value pair to the map
map.has_key? key	Returns true if the key exists as a key in the hash map
maps.has_value? value	Returns true if the value exists as a value in the hash map
map.fetch key	Returns the value that corresponds to the key
map.delete param	Deletes the key-value pair with the key param

Hashes (cont)

map.length	Returns the number of key-value pairs in the hash map
map.keys	Returns an array of all the keys in the hash map
map.values	Returns an array of all the values in the hash map
map.sort	Sorts the keys of the hash map in alphabetical order
map.inspect	Returns the current state of the hash map
map.each {|k, v| ... }	Iterates over each key-value pair in the hash map
map.each_key {|k| ... }	Iterates over each key in the hash map
map.each_value {|v| ... }	Iterates over each value in the hash map
map.each_value {|v| ... }	Iterates over each value in the hash map

Class Structure

```
class className1
@instVariable
@@clssVariable
...
attr_accessor
:instVariable
..
```

Class Structure (cont)

```
        def initialize
        ...
        end

        def classMethod1 param
        ...
        end
end
class className2
...
end
```

Class Details

Initialize	A method that is called internally when .new is called to create the object.
attr_reader	Instance variable getter
attr_writer	Instance variable setter
attr_accesor	Instance variable getter and setter
className.new	Method called to create the specified object

No overloading methods in Ruby.
Multiple classes can be written in the same file.