Ruby Cheat Sheet

Love2Code

General Syntax Rules

~Comments start with a pound or sharp (#) character and go to the end of a line (EOL).
~For multi-line comments use “=begin” and “=end” and anything included between them will be skipped by the interpreter.
~Every expression is finished (delimited) by a semicolon followed by a new line.
~Including a backslash (\) at the end of a line will not terminate the expression.

Types

NUMBERS
123 1_234 123.45 1.2e-3
0xffff (hex) 0b01011 (binary)
0377 (octal)
?a = ASCII character
?\C-a = Control-a
?\M-a = Meta-a
?\M-\C-a = Meta-Control-a

STRINGS
no interpolation
(interpolation) and backslashes \n
%q (no interpolation)
%Q (interpolation and backslashes)
% (interpolation and backslashes)
echo command interpretation with
interpolation and backslashes
%x (echo command interpretation with
interpolation and backslashes)

Basic types include numbers, strings, ranges, symbols, arrays, and hashes. In Ruby, files are also included because they are used often.

Reserved Words

alias, and, BEGIN, begin, break, case, class, def, defined do, else, elseif, END, end, ensure, false, for, if, in, module, next, nil, not, or, redo, rescue, retry, return, self, super, then, true, undef, unless, until, when, while, yield

Global Constants

TRUE = true value. FALSE = false value. NIL = nil value
STDIN = Standard input and default value for $stdin
STDOUT = Standard output and default value for $stdout
STDERR = Standard error output and default value for $stderr
ENV = Hash which contains current environment variables
ARGF = The alias to $<, ARGV Meta-IO across all files.
ARGV = Array of all arguments given on run
DATA = The file object of the script
RUBY_VERSION = Ruby version string
RUBY_Engine = Ruby implementation you’re running
RUBY_RELEASE_DATE = Release date string for cur version
RUBY_PLATFORM = Platform identifier

Arrays

[1, 2, 3]

%w(add val now #{1+1}) == [”add”, “val”, “now”, “\#[1+1]”]

%W(add val now #{1=1}) == [”add”, “val”, “now”, “2”]

Keep in mind, indexes may be negative but they index backwards if so.

Reg Expression

‘.’ Any character except newline
[set] Any single character of a set
[^set] Any single character not part of a set
‘*’ 0 or more previous regular expressions
*? 0 or more previous regular expressions (nongreedy)
+ 1 or more previous regular expressions
+? 1 or more previous regular expressions (nongreedy)
? 0 or 1 previous regular expression
| Alternation
( ) Grouping of regular expressions
^ Beginning of a line or string
$ End of a line or string
#{m,n} At least M
but most n previous regular expressions
#{m,n}? At least M
but most n previous regular expressions (nongreedy)
\A Beginning of a string
\b Backspace (0x08, inside [ ] only)
\B Non-word Boundary
\b Word boundary (outside [ ] only)
\d Digit, same as [0-9]
\D Non-digit
\S Non-whitespace character
\s Whitespace character [\t \n \r \f]
\W Non-word character
\w Word character [0-9, A-Za-z_]
\z End of a string
\Z End of a string, or before newline at the end
(?#) Comment
(?:) Grouping without back references
(?=) Zero-width positive look-ahead assertion
(?ix-ix) Turns on/off i/x options,
localized in the group if any
(?ix-ix:) Turns on/off i/x options,
localized in non-capturing group

Mode Strings

“r” R/O, start of file (default)
“r+” R/W, start of file
“w” W/O, truncates or creates
“w+” R/W, truncates or creates
“a” W/O, end of file or creates
“a+” R/W, end of file or creates
“b” Binary file mode (DOS/Windows only).

Files

File.join (p1, p2, ... pN) => “p1/p2/.../pN
Platform independent paths
File.new (path, mode_string = ”r”) => file
File.new (path, mode_num [, perm_num]) => file
File.open (filename, mode_string = “r”) {|file| block} => nil
File.open (filenmae [, mode_num [, perm_num]])
{|file| block} => nil
IO.foreach (path, sepstring = $/) {|line| block}
IO.readlines (path) => array

Variables

$global_variable
@@class_variable
@instance_variable
CONSTANT
::TOP_LEVEL_CONSTANT
OtherClass::CONSTANT
local_variable

Special Character Classes

[:alnum:] = Alpha-numeric characters
[:alpha:] = Alphabetic characters
[:blank:] = Whitespace
[:cntrl:] = Control characters
[:digit:] = Decimal digits
[:graph:] = Graph characters
[:lower:] = Lower-case characters
[:print:] = Printable characters
[:punct:] = Punctuation characters
[:space:] = Whitespace including tabs, carriage returns, and more
[:upper:] = Upper-case characters
[;xdigit:] = Hexadecimal digits

Pre-Defined Variables

DEBUG The boolean status of the -d switch
FILENAME The current input file from ARGF
LOAD_PATH Load path for scripts and binary modules
stderr Current standard error output
stdin Current standard input
stdout Current standard output
VERBOSE Verbose flag, as set by the -v switch
$! Exception object passed to #raise
$@ Stack backtrace generated by last exception raised
$& String matched by last successful match
$’ String to the left of last successful match
$` String to the right of last successful match
$+ Highest group matched by last successful match
$1 The Nth group of last successful match
$~ MatchData instance of last match
$= Flag for case insensitive (defaults to NIL)
$/ Input record separator
$ Output record separator
$, Output field separator for print and array
$; Default separator for string
$. Current line number for last file from input
$> Default output for print, and printf
$0 Name of script being executed
$$ Process number of Ruby running the script
$? Status of last executed child process

Pseudo Variables

self Receiver of current method
nil Sole instance of Class NilClass
true Sole instance of Class TrueClass
false Sole instance of Class FalseClass
__FILE__ Current source file name
__LINE__ Current line number in source file

Ranges

1..10
1...10
“a”..”z”
“a”...”z”
(1..10) === 5 #True
(1..10) === 10 #False
(1...10) === 10 #False
(1..10) === 15 #False