

RUBY REFERENCE SHEET

Mathematical Operators + - * / % **

- Order of Precedence **, then {*, /, %}, then {+,-}
- Left associativity except for **

Relational Operators == != < <= > >=

Logical Operators && (and) || (or) ! (not)

Variables

- All variable names must start with a lowercase letter.
- The remainder of the variable name (if any) can consist of any combination of uppercase letters, lowercase letters, digits and underscores (_).
- Variables are case sensitive.

Assignment Statements

- The lefthand side must contain a single variable.
- The righthand side can be any valid Ruby expression.

Defining Methods (Functions)

```
def methodname(parameterlist)
  instructions
end
```

- The name of a method follows the same rules as names for variables. (Ruby convention: methods that cause a side effect have names that end in ! and method that return true or false have names that end in ?)
- The parameter list can contain 1 or more variables that represent data to be used in the method's computation. A method can have 0 parameters.
- You can use the `return` instruction to return the value of a variable or expression or use `return` by itself to return immediately without returning a result.

Loops

```
for loop_variable in start_value .. end_value do
  loop body
end
```

```
while condition do
  loop body
end
```

Conditional Statements

```
if condition then
  statement_list
end
```

```
if condition then
  statement_list1
else
  statement_list2
end
```

Output & other functions

<code>print</code>	prints the value supplied
<code>puts</code>	prints the value supplied with a newline
<code>to_s</code>	converts the data value to a string (example: <code>15.to_s</code>)
<code>to_i</code>	converts the data value to an integer (example: <code>"25".to_i</code>)

Declaring new arrays:

<code>array1 = Array.new(20)</code>	# an uninitialized array of size 20
<code>array2 = []</code>	# an empty array
<code>array3 = Array(1..10)</code>	# an array with the values 1 through 10
<code>array4 = [3,5,7,9,11]</code>	# a 5 element array with initial values
<code>array5 = [[1,2,3], [4,5,6]]</code>	# an array of arrays (a 2D array)

Array Operations

<code>[i]</code>	returns the element at index <code>i</code> in the array (e.g. <code>array3[6]</code>)
<code>[i..j]</code>	returns a new array with the elements from the current array from index <code>i</code> to index <code>j</code> Example <code>array6 = array4[1..3]</code>
<code><< x</code>	appends <code>x</code> to the end of the array (e.g. <code>array2 << 16</code>)
<code>first</code>	returns the first element of the array (e.g. <code>array4.first</code>)
<code>last</code>	returns the last element of the array
<code>length</code>	returns the number of elements in the array
<code>each { }</code>	processes each element of the array based on the given code Example: <code>array4.each { item print item }</code>
<code>delete_if { }</code>	deletes each element of the array that matches the given condition Example: <code>array4.delete_if(item item > 6)</code>
<code>index(element)</code>	returns the index of the first occurrence of the given element
<code>include?(item)</code>	returns true if the array includes the given item, false otherwise
<code>clone</code>	returns a copy of the array
<code>slice!(i)</code>	removes and returns the item at position <code>i</code> in the array
<code>[row].length</code>	returns the number of columns in the given row of a 2D array example: <code>array5[1].length</code> returns the number of columns in row 1 of <code>array5</code>

Strings

Strings can be treated as an array of characters. The value of each position of a string is its ASCII value.

<code>s = "hello"</code>	Output:	104
<code>for i in 0..s.length-1 do</code>		101
<code>print s[i], "\n"</code>		108
<code>end</code>		108
		111

Running Ruby functions in irb

<code>load filename</code>	Loads a Ruby file	Example: <code>load "f1.rb"</code>
<code>quit</code>	Exits out of irb	