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(parameters)
  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**

- **print**  prints the value supplied
- **puts**  prints the value supplied with a newline
- **to_s**  converts the data value to a string (example: `15.to_s`)
- **to_i**  converts the data value to an integer (example: `“25”.to_i`)

**Declaring new arrays:**

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

**Array Operations**

- `[i]`  returns the element at index `i` in the array (e.g. `array3[6]`)
- `[i..j]`  returns a new array with the elements from the current array from index `i` to index `j`
  
  Example  `array6 = array4[1..3]`

- `<< x`  appends `x` to the end of the array (e.g. `array2 << 16`)

- `first`  returns the first element of the array (e.g. `array4.first`)

- `last`  returns the last element of the array

- `length`  returns the number of elements in the array

- `each { }`  processes each element of the array based on the given code
  
  Example:  `array4.each { |item| print item }`

- `delete_if { }`  deletes each element of the array that matches the given condition
  
  Example:  `array4.delete_if( |item| item > 6 )`

- `index(element)`  returns the index of the first occurrence of the given element

- `include?(item)`  returns true if the array includes the given item, false otherwise

- `clone`  returns a copy of the array

- `slice!(i)`  removes and returns the item at position `i` in the array

- `[]`.length  returns the number of columns in the given row of a 2D array
  
  Example:  `array5[1].length`  returns the number of columns in row 1 of array5

**Strings**

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

```
s = "hello"
for i in 0..s.length-1 do
    print s[i], "\n"
end
```

Output:  
104
101
108
108
111

**Running Ruby functions in irb**

- `load filename`  Loads a Ruby file
  
  Example:  `load “f1.rb”`

- `quit`  Exits out of irb