Ruby For Programmers

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

Strings

Ruby strings may either be delimited by single quotation marks: `'Hello world'` or by double quotation marks: `"Hello world"`. A single-quoted string is displayed literally. A double-quoted string evaluates anything between `#{ }` as Ruby code.

Example:

Assume you write this code to prompt a user to enter a name (let's assume it is "Mary") which is assigned to the variable, `name`:

```
print( 'Enter your name: ' )
name = gets()
```

Now if you enter this (a single-quoted string)…

```
puts('Hello #{name}' )
```

This is displayed:

```
Hello #{name}
```

But if you enter this (a double-quoted string)…

```
puts("Hello #{name}" )
```

This is displayed:

```
Hello Mary
```

You can embed any valid Ruby code in a double-quoted string. You can even call Ruby methods ore evaluate mathematical expressions. You can also use special characters such as `\n` (newline) and `\t` (tab), like this:

```
puts( "\n\t#{(1 + 2) * 3}\nGoodbye" )
```

OUTPUT:

```
        9
Goodbye
```

Format Strings

Ruby provides the `printf` method to print 'format strings' containing specifiers starting with a percent sign, `%`. The format string may be followed by one or more data items separated by commas; the list of data items should match the number and type of the format specifiers. The actual data items replace the matching specifiers in the string and they are formatted accordingly. These are some common formatting specifiers:

```
%d - decimal number
%f - floating point number
%o - octal number
%p - inspect object
%s - string
%x - hexadecimal number
```

You can control floating point precision by putting a point-number before the floating point formatting specifier, `%f`. For example, this would display the floating point value to two digits:

```
printf("%0.02f", 10.12945 )       # displays 10.13
```