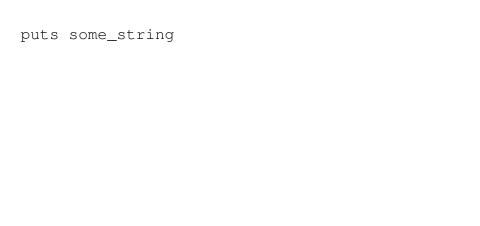
Single quoted strings are interpreted literally.

Double quoted strings allows further processing on the string, like the substitution of variables into \${VARIABLE}.



obj.class obj.methods

Block form: if condition

#statements

One-line form:

statement if condition

They are more idiomatic Ruby for "if not" and "while not" constructions.

Every type can be interpreted as a boolean. nil and and the literal false evaluate to false. Everything else evaluates to true.

&&/|| short-circuit, as in Java. Also, &&/|| have the alternatives and/or.

Type checking isn't done until run-time, when an operation is

actually executed.

Ruby is duck-typed, meaning one type can be used as another so long as it has the needed fields and methods to make an operation work. This is to be contrasted with languages like Java where one type can be used as another only if one is a subclass of the other.