Chapter 2 - A Closer Look

* 2.1 Syntax
  + Terms- either a constant, variable, or a structure
    - Characters are divided into 4 categories
      * A B C … Y Z
      * a b c … y z
      * 0 1 2 … 9
      * + - \* / \ ~ ^ < > : . ? @ # $ &
    - Constants name specific objects or specific relationships
      * Examples: likes mary john book wine owns jewels can\_steal
    - Variables
      * Look like atoms, except they have names beginning with a capital letter or an underline sign “\_”
    - Structures is a single object consisting of a collection of other objects, called components
      * Components are grouped together into a single structure for convenience in handling them
* 2.2 Characters
  + Names of constants and variables are built up from strings of characters
    - A B C … Z
    - a b c … z
    - 0 1 2 … 9
    - ! “ # $ % ‘ ( ) = - ~ ^ | \ { } [ ] \_ ` @ + ; \* : < > , . ? /
* 2.3 Operators
  + Form of syntax that makes some structures easier to read
* 2.4 Equality and Unification
  + “=” is an infix operator which tries to prove equality
  + ?- X = Y. Prolog attempts to unify X and Y, and the goal succeeds if they unify
* 2.5 Arithmetic
  + Arithmetic operations are useful for comparing numbers and for calculating results
* 2.6 Summary of Satisfying Goals
  + Prolog performs a task in response to a question from the programmer
  + Question provides a conjunction of goals to be satisfied
  + Prolog uses known clauses to satisfy goals
  + A rule can only reduce the task to that of satisfying a conjunction of subgoals
  + Clause can only be used if it unifies the goal under consideration
  + If a goal cannot be satisfied, backtracking will be initiated
  + Backtracking consists of reviewing what has been done, attempting to re-satisfy the goals by finding an alternative way to satisfy them
  + Successful satisfaction of a conjunction of goals
    - Prolog attempts to satisfy the goals in a conjunction, whether they appear in a rule body or in a question, in the order they are written (left to right)
  + Consideration of Goals in backtracking
    - When a failure is generated, the “flow of satisfaction” passes back along the way it has come
  + Unification
    - A not instantiated variable will unify with any object
    - Otherwise, an integer or atom will unify with only itself
    - Otherwise, a structure will unify with another structure with the same functor and number of arguments, and all the corresponding arguments must unify