Variables (General Clauses)

In a simple query, you can use variables to ask Prolog to find who likes tennis. For example:

```
  likes(X, tennis).
```

This query uses the letter *X* as a variable to indicate an unknown person. Variable names in Turbo Prolog must begin with a capital letter (or an underscore), after which any number of letters (upper-case or lower-case), digits, or underline characters (_) can be used. For example, the following are valid variable names:

```
  My_first_correct_variable_name
  Sales_10_11_86
```

Careful choice of variable names makes programs more readable. For example,

```
  likes(Person, tennis).
```

is better than

```
  likes(X, tennis).
```

How Variables Get Their Values

You may have noticed that Prolog has no assignment statement; this is a significant distinction between Prolog and other programming languages. ***Variables in Prolog get their values by being matched to constants in facts or rules.***

Until it gets a value, a variable is said to be *free*; when it gets a value, it becomes *bound*. But it only stays bound for the time needed to obtain one solution to the query; then Prolog unbinds it, backs up, and looks for alternative solutions.

This is a very important point: ***You can't store information by giving a value to a variable.*** Variables are used as part of the pattern-matching, process, not as a kind of information storage.

Anonymous Variables

Anonymous variables enable you to unclutter your programs. If you only need certain information from a query, you can use anonymous variables to ignore the values you don't need. In Prolog, the anonymous variable is represented by a lone underscore ("_").

The anonymous variable can be used in place of any other variable. The difference is that the anonymous variable will never get set to a value.

The following *parents* example demonstrates how the anonymous variable is used.

```
PREDICATES
  male(symbol)
  female(symbol)
  nondeterm parent(symbol, symbol)

CLAUSES
  male(ayman).
  male(blal).
  female(samr).
  female(fatma).
  parent(ayman,blal).
  parent(samr,blal).
   parent(blal,fatma).
```

For example, in the following query, you need to know which people are parents, but you don't need to know who their children are. Prolog realizes that each time you use the underscore symbol in the query, you don't need information about what value is represented in that variable's place.

```
GOAL
  parent(Parent, _).
```

Given this query, Prolog replies

```
  Parent= ayman
  Parent= samr
  Parent= blal
  3 Solutions
```

In this case, because of the anonymous variable, Prolog finds and reports three parents, but it does not report the values associated with the second argument in the ***parent*** clause.

The anonymous variable matches anything.

Anonymous variables can also be used in facts. The following Prolog facts

```
  owns(_, shoes).
  eats(_).
```

could be used to express the natural language statements

```
  Everyone owns shoes.
  Everyone eats.
```