What is Prolog?

• Prolog (programming in logic) is a logic-based programming
language: programs correspond to sets of logical formulas and
the Prolog interpreter uses logical methods to resolve queries.
• Prolog is a declarative language: you specify what problem you
want to solve rather than how to solve it.
• Prolog is very useful in some problem areas, such as artificial
intelligence, natural language processing, databases, . . . , but
pretty useless in others, such as for instance graphics or numerical algorithms.

Facts

Facts are predicates followed by a dot. Facts are used to define something as being unconditionally true.

A little Prolog program consisting of four facts:
bigger(elephant, horse).
bigger(horse, donkey).bigger(donkey, dog).
bigger(donkey, monkey).

Queries

- **Queries** are predicates (or sequences of predicates) followed by a dot. They are typed in at the Prolog prompt and cause the system to reply.
- **Programs**: Facts and rules are called clauses. A Prolog program is a list of clauses

After compilation we can query the Prolog system:
?- bigger(donkey, dog).
Yes
?- bigger(monkey, elephant).
No

Rules:

Rules consist of a head and a body separated by :-. The head of a rule is true if all predicates in the body can be proved to be true.

```
grandfather(X, Y) :-
father(X, Z),
parent(Z, Y).
```

Terms

Prolog terms are either numbers, atoms, variables, or compound
terms.

Atoms start with a lowercase letter or are enclosed in single quotes.
Ex:
elephant, xYZ, a_123, 'Another pint please'

Variables start with a capital letter or the underscore.
Ex:
X, Elephant, _G177, MyVariable.

Compound terms have a functor (an atom) and a number of arguments (terms):
is_bigger(horse, X)
f(g(Alpha, _), 7)
'My Functor'(dog)

*Atoms and numbers are called atomic terms.
*Atoms and compound terms are called predicates.
*Terms without variables are called ground terms.

Example 1:-

There are some facts , rules ,according to them find the output of these queries.

likes(john,jane).

likes(jane,john).

likes(jack,jane).

friends(X,Y) :- likes(X,Y), likes(Y,X).

?- likes(john,jane).

?- likes(jane,jack).

?- likes(john,X).

?- friends (john,X).

?- friends (X,Y).

Example 2:-

There are some facts , according to them find the output of these queries.

Food(burger).

Food(sandwich).

Food(pizza)..

Lunch(sandwich).

Dinner(pizza).

Meal(X):- food(X).

?-food(pizza).

?-dinner(burger).

?-meal(X).

?-meal(X),lunch(X).

?-dinner(sandwich).

?-meal(X),dinner(X).

Matching

Two terms match if they are either identical or if they can be made identical by substituting their variables with suitable ground terms.

We can explicitly ask Prolog whether two given terms match by using the equality-predicate = (written as an infix operator).

Unification : the process of finding the values of variables that make two terms equal.

?- born(mary, yorkshire) = born(mary, X).
X = yorkshire
Yes

Examples:

?-john=john.

?-john='john'.

?-name(ss)=\name(dd).

?-teacher(noor,X)=teacher(Z,ahmed).

?-likes(noor,X)=likes(X,ahmed).

?-X=Y,X is 5+2,Y=7

?-X=Y,X=5+2,Y=7

?-person(X,Y,Z)=person(john,smith,23).

?-person(john,Y,23)=person(X, smith,24).

?-Places(london,dog,X)=places(Z,Z,23).
?-Places(london,london,X)=places(Z,Z,23).

Structure :-

structures are a fundamental data type used to represent complex entities by grouping related pieces of information together.

Example:

likes(ahmed,add(portsaid,age(twenty))).
?- likes(ahmed,X).
X = add(portsaid, age(twenty)).
?- likes(ahmed,add(X,_)).
X = portsaid.
?- likes(ahmed,add(_,X)).
X = age(twenty).
?- likes(ahmed,add(portsaid,age(X))).
X = twenty.

The Anonymous Variable

The variable _ (underscore) is called the anonymous variable.
Every occurrence of _ represents a different variable (which is why instantiations are not being reported).

```
?- p(_, 2, 2) = p(1, Y, _).
Y = 2
Yes
```

```
Male(ahmed,Mohamed).
?- male(_).
?-male(_,_).
```

Arithmetic Expressions in Prolog

Prolog comes with a range of predefined arithmetic functions and
operators. Expressions such as 3 + 5, for example, are valid Prolog
terms.
So, what's happening here?

```
?- 3 + 5 = 8.
No
```

Examples:

```
?- 3\=4
true.
?- 4='4'.
False.
?- X is 10, Z is X+1.
X = 10,
Z = 11.
?- X is sqrt(36).
X = 6.0.
?- X is 30,Y is 5,Z is X+Y*Y+sin(X).
X = 30,
Y = 5,
Z = 54.01196837590714.
?- 10 is 7+13-11+9.
false.
```

Matching vs. Arithmetic Evaluation

The terms 3 + 5 and 8 do not match. In fact, when we are interested in the sum of the numbers 3 and 5, we can't get it through matching, but we need to use arithmetic evaluation.
We have to use the is-operator:
?- X is 3 + 5.
X = 8
Yes

The is-Operator

The is-operator causes the term to its right to be evaluated as an arithmetic expressions and matches the result of that evaluation
with the term on the operator's left. (The term on the left should usually be a variable.)

Example:
?- Value is 3 * 4 + 5 * 6, OtherValue is Value / 11.
Value = 42
OtherValue = 3.81818
Yes