



Prolog – Lists & Pattern Matching

- The unification operator: $=/2$
 - The expression $A=B$ is true if A and B are terms and unify (look identical)

arity

```
?- a = a.  
true  
?- a = b.  
false  
?- a = X.  
X = a  
?- X = Y.  
true
```

Read Sections 1&2
of Prolog Tutorial
online

NOTE: This is where Prolog really shines as an AI programming language:

- ☞ Knowledge representation - List
- ☞ Knowledge processing - pattern matching



Prolog – Lists & Pattern Matching

- Lists – a convenient way to represent abstract concepts
 - Prolog has a special notation for lists.

[a]
[a,b,c]
[]

↙ Empty
List

[bmw, vw, mercedes]
[chicken, turkey, goose]



Prolog – Lists & Pattern Matching

- Pattern Matching in Lists

?- [a, b] = [a, X].
X = b

?- [a, b] = X.
X = [a, b]

But:

?- [a, b] = [X].
no

- The Head-Tail Operator: [H|T]

?- [a, b, c] = [X | Y];
X = a
Y = [b, c]

?- [a] = [Q | P];
Q = a
P = []



Prolog – Lists

The predicate first/2: accept a list in the first argument and return the first element of the list in second argument.

```
first(List,E) :- List = [H|_], E = H;
```



Prolog – Lists

The predicate last/2: accept a list in the first argument and return the last element of the list in second argument.

Recursion: there are always two parts to a recursive definition; the base and the recursive step.

```
last([A],A).  
last([A|L],E) :- last(L,E).
```