

EX.NO: 7

INTRODUCTION TO PROLOG

AIM

To learn PROLOG terminologies and write basic programs.

TERMINOLOGIES

1. Atomic Terms: -

Atomic terms are usually strings made up of lower- and uppercase letters, digits, and the underscore, starting with a lowercase letter.

Ex:

dog
ab_c_321

2. Variables: -

Variables are strings of letters, digits, and the underscore, starting with a capital letter or an underscore.

Ex:

Dog
Apple_420

3. Compound Terms: -

Compound terms are made up of a PROLOG atom and a number of arguments (PROLOG terms, i.e., atoms, numbers, variables, or other compound terms) enclosed in parentheses and separated by commas.

Ex:

is_bigger(elephant,X)
f(g(X,_),7)

4. Facts: -

A fact is a predicate followed by a dot.

Ex:

bigger_animal(whale).
life_is_beautiful.

5. Rules: -

A rule consists of a head (a predicate) and a body (a sequence of predicates separated by commas).

Ex:

is_smaller(X,Y):-is_bigger(Y,X).
aunt(Aunt,Child):-sister(Aunt,Parent),parent(Parent,Child).

SOURCE CODE:

KB1:

```
woman(mia).  
woman(jody).  
woman(yolanda).  
playsAirGuitar(jody).  
party.  
Query 1: ?-woman(mia).  
Query 2: ?-playsAirGuitar(mia).  
Query 3: ?-party.  
Query 4: ?-concert.
```

OUTPUT: -

```
?- woman(mia).  
true.  
  
?- playsAirGuitar(mia).  
false.  
  
?- party.  
true.  
  
?- concert.  
ERROR: Unknown procedure: concert/0 (DWIM could not correct goal)  
?- ■
```

KB2:

```
happy(yolanda).  
listens2music(mia).  
Listens2music(yolanda):-happy(yolanda).  
playsAirGuitar(mia):-listens2music(mia).  
playsAirGuitar(Yolanda):-listens2music(yolanda).
```

OUTPUT: -

```
?- playsAirGuitar(mia).  
true.  
  
?- playsAirGuitar(yolanda).  
true.  
  
?- ■
```

KB3:

```
likes(dan,sally).  
likes(sally,dan).  
likes(john,brittney).  
married(X,Y) :- likes(X,Y) , likes(Y,X).  
friends(X,Y) :- likes(X,Y) ; likes(Y,X).
```

OUTPUT: -

```
?- likes(dan,X).  
X = sally.  
  
?- married(dan,sally).  
true.  
  
?- married(john,brittney).  
false.
```

KB4:

```
food(burger).  
food(sandwich).  
food(pizza).  
lunch(sandwich).  
dinner(pizza).  
meal(X):-food(X).
```

OUTPUT:

```
?-  
|   food(pizza).  
true.  
  
?- meal(X),lunch(X).  
X = sandwich ,  
  
?- dinner(sandwich).  
false.  
?-
```

KB5:

```
owns(jack,car(bmw)).  
owns(john,car(chevy)).  
owns(olivia,car(civic)).  
owns(jane,car(chevy)).  
sedan(car(bmw)).  
sedan(car(civic)).  
truck(car(chevy)).
```

OUTPUT:

```
?-  
|   owns(john,X).  
X = car(chevy).  
  
?- owns(john,_).  
true.  
  
?- owns(Who,car(chevy)).  
Who = john ,  
  
?- owns(jane,X),sedan(X).  
false.  
  
?- owns(jane,X),truck(X).  
X = car(chevy).
```

RESULT:

Thus the python code is implemented successfully and the output is verified.