Logic Programming

Nihal Shah Parekkattil

RA1911003010868

Python Logic programming is a programming paradigm that sees computation as automatic reasoning over a database of knowledge made of facts and rules. It is a way of programming and is based on formal logic.

A program in such a language is a set of sentences, in logical form, one that expresses facts and rules about a problem domain.

1- Implement using pyDatalog: Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2).

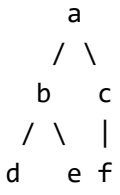
a. Define a predicate **brother(X,Y)** which holds if X and Y are brothers.

b. Define a predicate **cousin(X,Y)** which holds if X and Y are cousins.

c. Define a predicate **grandson(X,Y)** which holds if X is a grandson of Y.

d. Define a predicate **descendent(X,Y)** which holds if X is a descendent of Y.

e. Consider the following genealogical tree:



What are the answers generated by your definitions for the queries:

- brother(X,Y)
- cousin(X,Y)
- grandson(X,Y)
- descendent(X,Y)

In [9]:
```
from pyDatalog import pyDatalog
pyDatalog.create_terms('a,b,c,d,e,f,brother,cousin,grandson,descendent,X,Y')
+brother('b','c')
+brother('d','e')
+cousin('d','f')
+cousin('e','f')
+grandson('d','a')
+grandson('e','a')
+grandson('f','a')
+descendent('b','a')
+descendent('c','a')
+descendent('d','b')
+descendent('f','c')
+descendent('e','b')
```

In [10]:
```
print(pyDatalog.ask('brother(X,Y)'))
```

```
{('d', 'e'), ('b', 'c')}
```

In [11]:
```
print(pyDatalog.ask('cousin(X,Y)'))
```

```
{('d', 'f'), ('e', 'f')}
```

In [12]:
```
print(pyDatalog.ask('grandson(X,Y)'))
```

```
{('f', 'a'), ('e', 'a'), ('d', 'a')}
```

In [13]:
```
print(pyDatalog.ask('descendent(X,Y)'))
```

```
{('f', 'c'), ('e', 'b'), ('d', 'b'), ('c', 'a'), ('b', 'a')}
```

2- Encode the following facts and rules in pyDatalog:

- Bear is big
- Elephant is big
- Cat is small
- Bear is brown
- Cat is black
- Elephant is gray
- An animal is dark if it is black
- An animal is dark if it is brown

Write a query to find which animal is dark and big.

In [8]:
```
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,bear,elephant,cat,small,big,brown,black,gray,dark')
+big('elephant')
+big('bear')
+small('cat')
+black('cat')
+brown('bear')
+gray('elephant')
dark(X)<=black(X) or brown(X)
print(big(X),dark(X))
```

```
X
--------
bear
elephant X
---
cat
```

3-The following are the marks scored by 5 students.

Student-Name	Mark
Ram	90
Raju	45
Priya	85
Carol	70
Shyam	80

Enter the above data using pyDatalog.

Write queries for the following:

a. Print Student name and mark of all students.

b. Who has scored 80 marks?

c. What mark has been scored by Priya?

d. Write a rule ‘passm’ denoting that pass mark is greater than 50. Use the rule to print all students who failed.

e. Write rules for finding grade letters for a marks and use the rule to find the grade letter of a given mark.

In [14]:
```
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,student,marks,passm,grades')
+student('ram')
+student('raju')
+student('priya')
+student('carol')
+student('shyam')
+marks('90','ram')
+marks('45','raju')
+marks('85','priya')
+marks('70','carol')
+marks('80','shyam')
+grades('ram','O')
+grades('priya','A')
+grades('shyam','A')
+grades('carol','B')
+grades('raju','E')
```

In [19]:
```
print(marks(X,Y))
```

```
X  | Y
---|------
80 | shyam
70 | carol
85 | priya
45 | raju
90 | ram
```

In [18]:
```
print(marks('80',X))
```

```
X
-----
shyam
```

In [17]:
```
print(marks(X,'priya'))
```

```
X
--
85
```

In [16]:
```
passm(X)<=grades(X,'E')
print(passm(X))
```

```
X
----
raju
```

In [15]:
```
i = int(input("Enter Mark "))
if i>89 :
    print('O')
elif i<90 and i>=80:
    print('A')
elif i<80 and i>=70:
    print('B')
elif i<70 and i>=60:
    print('C')
elif i<60 and i>=50:
    print('D')
else:
    print('E')
```

```
Enter Mark 75
B
```

4- Solve the set of queries in the previous question using imperative programming paradigm in Python. Store the data in a dictionary.

In [21]:
```
marks={90:'ram',85:'priya',80:'shyam',70:'carol',45:'raju'}
print("Marks of students :")
for i in marks:
    print(i,marks[i])
print("Student who scored 80 : ",marks[80])
for i in marks:
    if marks[i]=='priya':
        print("Priya scored : ",i)
for i in marks:
    if i<50:
        print("Student who failed : ",marks[i])
print("Students and their grades :")
for i in marks:
    if i>=90:
        print(marks[i] ,'O')
    elif i<90 and i>=80:
        print(marks[i],'A')
    elif i<80 and i>=70:
        print(marks[i],'B')
    elif i<70 and i>=60:
        print(marks[i],'C')
    elif i<60 and i>=50:
        print(marks[i],'D')
    else:
        print(marks[i],'E')
```

```
Marks of students :
90 ram
85 priya
80 shyam
70 carol
45 raju
Student who scored 80 :  shyam
Priya scored :  85
Student who failed :  raju
Students and their grades :
ram O
priya A
shyam A
carol B
raju E
```

5- Write a recursive program to find factorial of a number using pyDatalog.

In [23]:
```
from pyDatalog import pyDatalog
pyDatalog.create_terms('factorial, N')
num=int(input('Enter any number:'))
factorial[N] = N*factorial[N-1]
factorial[1] = 1
print(factorial[num]==N)
```

```
Enter any number:6
N
---
720


------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------
```