**Supplementary Material for Session 1**

1. **Queries to KB and finding an answer using Backward Chaining in Prolog:**

Is Hasib a grandparent of Rebeka?

grandparent ('Hasib', 'Rebeka').

* parent ('Hasib', Y), parent(Y, 'Rebeka').

parent ('Hasib', 'Rakib'). [Y ← Rakib]

* parent ('Rakib', 'Rebeka').

Yes.

* Various types of queries are possible.

Who are parents of Rebeka? parent(X, 'Rebeka').

Who are parents? parent(X, \_).

Is Hasib a parent? parent('Hasib', \_).

Is Hasib a parent of Rebeka? parent('Hasib', 'Rebeka').

Who have parents? parent(\_, X).

Who are parents of whom? parent(X, Y).

Is there anybody who is a grandparent of somebody. grandparent(\_, \_).

Does Sohel have a grandparent? grandparent(\_,'Sohel').

Who is a parent and, also, has a parent? parent(X, \_), parent(\_, X).

Who is either a parent or has a parent? parent(X, \_); parent(\_, X).

* Various rules may also be formulated for father, mother, brother, sister, aunt, uncle, etc. There may be more than one rule to define, for example, a grandfather.

[brother(X,Y):-parent(Z,X), parent(Z,Y), male(X), not(X=Y).]

* Nesting of the following type should be avoided.

greatGrandParent (X, Z) :- parent(X, Y), grandparent(Y, Z).

greatGreatGrandParet(X, Z) :- parent(X, Y), greatGrandParent(Y,Z).

1. **Working with Structured Data and functions in Python:**

* Lists, strings and tuples are ordered sequences of objects.
* Lists and tuples can contain any type of objects. Lists and tuples are like arrays.
* Lists are mutable so they can be extended or reduced at will.
* Tuples, like strings, are immutable. Tuples are faster and consume less memory.
* Strings contain only characters.
* A dictionary is an unordered collection of key-value pairs, which can be modified.

#List  
l1=[0,2,1]  
l1[1]

l1[1]=3  
l1.append(5)  
l2=[3,4,5]  
l1.extend(l2)  
print(**"Length:"** ,len(l1))  
#Tuple  
L3=(2,4,1)

L3[1] # L3[1]= 5 not allowed

#String  
S=**"This is AUST"**#Dictionary  
d = {**"a"**:1, **"b"**:2}  
d[**"z"**]=4 # d[**"b"**] returns 2  
for key in d:  
 print(key)  
for value in d.values():  
 print(value)  
for key, value in d.items():  
 print(key , **":"**, value)

#Python is Easy

#Observe the code of user defined function

def fssum():

a=int(input("Start:"))

d=int(input("Interval:"))

n=int(input("n:"))

i,s=1,0

while(i<=n):

s=s+a+d\*(i-1)

i=i+1

print("Sum:",s)

input("Press Enter to continue")

# Main

t=int(input("How many times?"))

for i in range(t):

print("Iteration:",i+1)

fssum()

#Python is Easy

#Observe the dialog in shell

>>> x=[1,2,3]

>>> y=(9,8)

>>> x

[1, 2, 3]

>>> y

(9, 8)

>>> x,y=y,x

>>> x

(9, 8)

>>> y

[1, 2, 3]

>>> for i in range(5):

print(i)

>>> for i in range(1,10,2):

print(i, end=’ ’)