Supplementary Material for Session 2

I. Recursion in Prolog:

i) Ancestor
   a. A parent is an ancestor.
   b. A parent of an ancestor is an ancestor.
      [X is an ancestor of Y, if X is a parent of an ancestor of Y.]

```
-------------------------
ancestor(X, Y) :- parent(X, Y), !.
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

ii) Factorial of 0 is 1.
If n is greater than 0, then factorial of n is the product of n and the factorial of n-1.

[Factorial of N is F, if N is greater than 0 and F is the product of N and factorial of N-1.]

```
-----------------------
factorial(0, 1):-!.
factorial(N, F) :- N>0, N1 is N-1, factorial(N1, F1), F is N*F1.
```

iii) 100+105+110+ … +(100+(n-1)x5)

Sum of the 1st one element is 100.
Sum of the 1st n elements is the sum of 1st n-1 elements and the nth element, which is 100+(n-1)x5.

```
--------------------
sum1(1, 100):-!.
sum1(N, S):-N1 is N-1, sum1(N1, S1), S is S1+100+(n-1)x5.
```

iv) Representing a weighted graph and finding the length of a path

```
neighbor(i,a,35). neighbor(i,b,45). neighbor(a,c,20).
neighbor(a,d,30). neighbor(b,d,25). neighbor(b,e,35).
neighbor(b,f,27). neighbor(c,d,30). neighbor(c,g,47).
neighbor(d,g,30). neighbor(e,g,25).

pathLength(X,Y,L):- neighbor(X,Y,L),!.
pathLength(X,Y,L):- neighbor(X,Z,L1), pathLength(Z,Y,L2), L is L1+L2.
```

II. Working with Python:

```
# Use of global variables in Python
def f():
    global s
    print (s)
    s = "I love python!"
    print (s)

# Global Scope
s = "Python is great!"
f()
print (s)
```