# 1 - Lists in Prolog:

can be regarded as special Prolog structures that can be used to represent an ordered sequence of Prolog terms.

**Examples:**

| | |
|---|---|
| [ice_cream, coffee, chocolate] | a list with three elements (all constants) |
| [a, b, c, c, d, e] | a list with six elements (all constants) |
| [ ] | a list with no elements in it |
| [dog(fido), cat(rufus), goldfish(jimmy)] | a list with three elements |
| [happy(fred),[ice_cream,chocolate],[1,[2],3]] | a list with three elements! |
| the list [a,b,c] is not the same as [a,c,b] | |

# 2 - How to construct/deconstruct a list???

a simple way of splitting a list into two bits:

- the first element (if there is one!)
- the rest of the list.

# 3 - Some Possible Matches:

| Query | Result |
|---|---|
| 1. [b,a,d]=[d,a,b] | **fails ---as the order matters** |
| 2. [X]=[b,a,d] | **fails ---the two lists are of different lengths** |
| 3. [X\|Y]=[he,is,a,cat] | **succeeds with X=he, Y=[is,a,cat]** |
| 4. [X,Y\|Z]=[a,b,c,d] | **succeeds with X=a, Y=b, Z=[c,d]** |
| 5. [X\|Y]=[] | **fails ---the empty list can't be deconstructed** |
| 6. [X\|Y]=[[a,[b,c]],d] | **succeeds with X=[a,[b,c]], Y=[d]** |
| 7. [X\|Y]=[a] | **succeeds with X=a, Y=[]** |

## 4 – Example Queries:

1. [a,bX]=[A,B,c] succeeds with A=a, B=b and X=[c].
2. [a,b]=[b,a] fails.
3. [a|[b,c]]=[a,b,c] succeeds.
4. [a,[b,c]]=[a,b,c] fails where list with 2 elements <> list
5. [a,X]=[X,b] fails
6. [a[]]=[X] succeeds with X=a.
7. [a,b,X,c]=[A,B,Y] fails where list with 4 elements <> list with three elements
8. [HT]=[[a,b],[c,d]] succeeds with H=[a,b] and T=[[c,d]].
9. [[X],Y]=[a,b] fails where [X]=a. This fails.

## 5 - Try the following:

- X= [1, 2, 3, 4, 6,7 ], Y=X.
- [X|Y]= [1, 2, 3, 4,6, 7].
- [X, Y, Z]=[1, 2, 3].
- [X, Y, Z]=[1, 2, 3, 4].
- [First|Rest]=[1, 2, 3, 4, 5, 6, 7].
- [First, Second|Rest]=[Second, Third, First], First=1.
- [Fudge| Chocolate]=[1, 2, 3,4 ,5 , 6, 7].
- [sin(X), cos(X)]=[sin(a), cos(b)].

# 6 – Exercises:

## 1 - List Length:

Take 2 params, 1s one the LIST, 2nd one is the return value (list length)

→ Let's write the rule

list_length([], 0).

list_length([_|Xs], L):- list_length(Xs, L0), L is L0 + 1.

## 2 - List Sum:

Take 2 params, 1s one the LIST, 2nd one is the return value (sum of the list value)

→ Let's write the rule

list_sum([], 0).

list_sum([X0|Xs], Res):- list_sum(Xs, Res0), Res is Res0 + X0.

## 3 – Element Count into List:

count 1st param occurrence in the 2nd param list into the 3rd one.

**Example:**

count_occur (1, [1, 2, 3, 1, 2, 3, 1], Res) → Res = 3

→ Let's write the rule

count_occur(_, [], 0).

count_occur(X, [X|Xs], Res):- count_occur(X, Xs, Res_temp), Res is Res_temp+1.

count_occur(X, [Y|Ys], Res):- X\=Y, count_occur(X, Ys, Res).