```
%country(name, population (in thousands), ca
country(sweden, 8823, stockholm).
country(usa, 221000, washington).
country(france, 56000, paris).
country(denmark, 3400, copenhagen).
% city(name, in_country, population).
city(lund, sweden, 88).
city(new_york, usa, 5000).  % Paris, Texas.
city(paris, usa, 1).  % Paris, Texas.
city(copenhagen, denmark, 1200).
city(aarhus, denmark, 330).
city(odense, denmark, 120).
city(stockholm, sweden, 1300).
city(gothenburg, sweden, 350).
city(washington, usa, 3400).
city(paris, france, 2000).
city(marseilles, france, 1000).
```

```
%country(name, population (in thousands), ca
country(sweden, 8823, stockholm).
country(usa, 221000, washington).
country(france, 56000, paris).
country(denmark, 3400, copenhagen).
% city(name, in_country, population).
city(lund, sweden, 88).
city(new_york, usa, 5000).  % Paris, Texas.
city(paris, usa, 1).  % Paris, Texas.
city(copenhagen, denmark, 1200).
city(aarhus, denmark, 330).
city(odense, denmark, 120).
city(stockholm, sweden, 1300).
city(gothenburg, sweden, 350).
city(washington, usa, 3400).
city(paris, france, 2000).
city(marseilles, france, 1000).
```

September
1st Session 2013/2014 (A131)

Wan Hussain Wan Ishak

School of Computing
UUM College of Arts and Sciences
Universiti Utara Malaysia

(P) 04-9284786 (F) 04-9284753
(E) hussain@uum.edu.my
(U) http://wanhussain.com

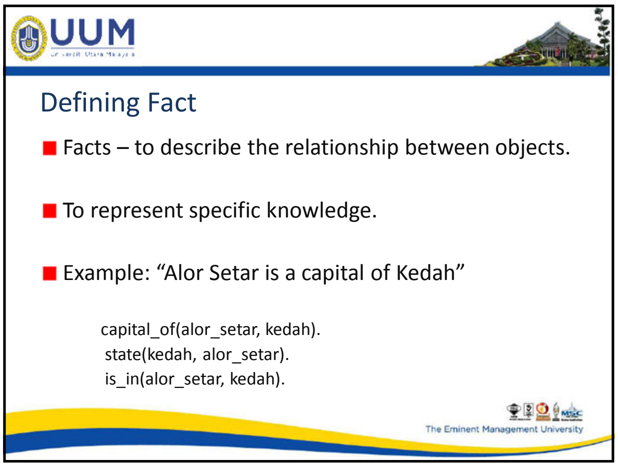
```
%country(name, population (in thousands), ca
country(sweden, 8823, stockholm).
country(usa, 221000, washington).
country(france, 56000, paris).
country(denmark, 3400, copenhagen).
% city(name, in_country, population).
city(lund, sweden, 88).
city(new_york, usa, 5000).  % Paris, Texas.
city(paris, usa, 1).  % Paris, Texas.
city(copenhagen, denmark, 1200).
city(aarhus, denmark, 330).
city(odense, denmark, 120).
city(stockholm, sweden, 1300).
city(gothenburg, sweden, 350).
city(washington, usa, 3400).
city(paris, france, 2000).
city(marseilles, france, 1000).
```

Lecture notes

Constructing Prolog Program

- Defining facts and rules
- Using connectors
- Constructing query
- Problem representation
- Input & output predicates
- Subroutines

Universiti Utara Malaysia
STIN2024
Pengaturcaraan
Programming
Logik
Logic
The Eminent Management University

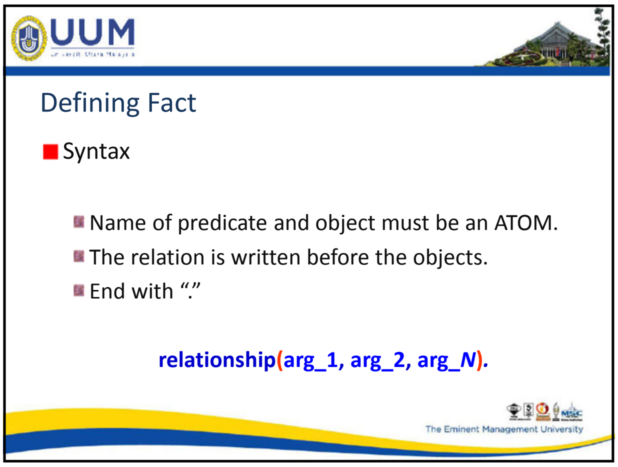


Defining Fact

- Facts – to describe the relationship between objects.
- To represent specific knowledge.
- Example: “Alor Setar is a capital of Kedah”

```
capital_of(alor_setar, kedah).
 state(kedah, alor_setar).
 is_in(alor_setar, kedah).
```

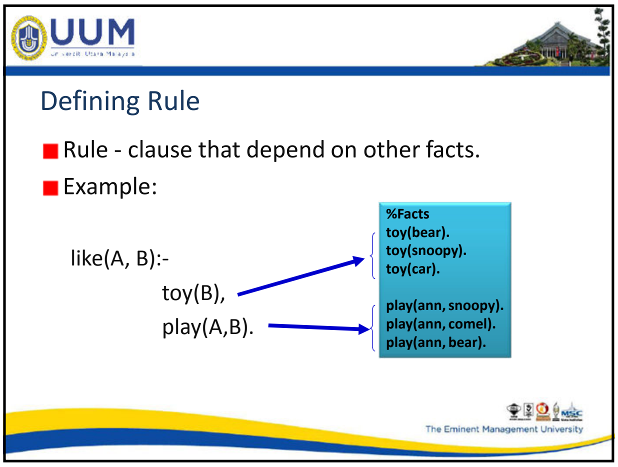
UUM
The Eminent Management University



Defining Fact

- Syntax
  - Name of predicate and object must be an ATOM.
  - The relation is written before the objects.
  - End with “.”

relationship(arg_1, arg_2, arg_*N*).

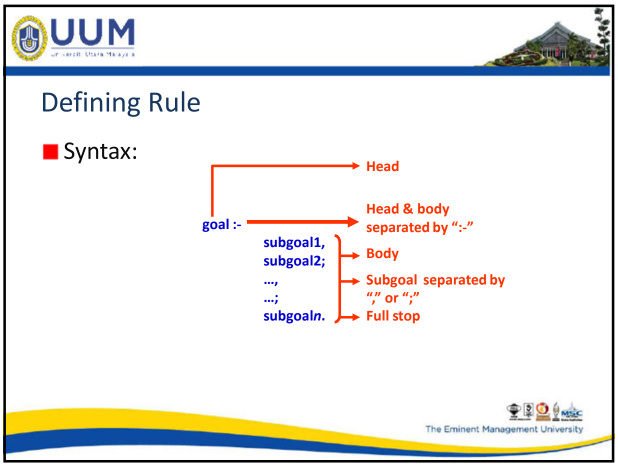


Defining Rule

- Rule - clause that depend on other facts.
- Example:

```
like(A, B):-
        toy(B),
        play(A,B).
```

```
%Facts
toy(bear).
toy(snoopy).
toy(car).

play(ann, snoopy).
play(ann, comel).
play(ann, bear).
```



Defining Rule

- Syntax:

```
goal :-
        subgoal1,
        subgoal2;
        …,
        …;
        subgoaln.
```

Head

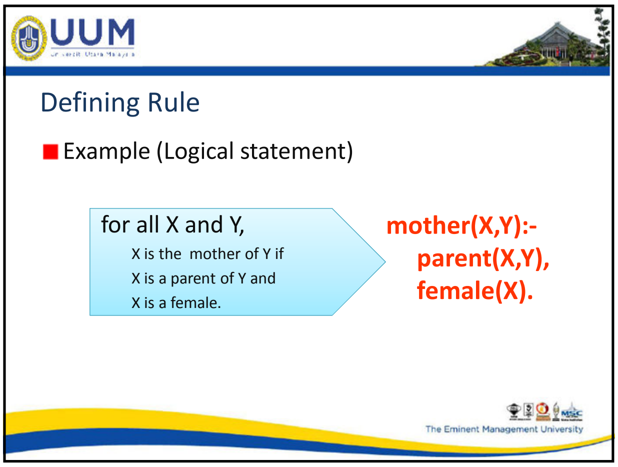
Head & body separated by “:-”

Body

Subgoal separated by “,” or “;”

Full stop

The Eminent Management University



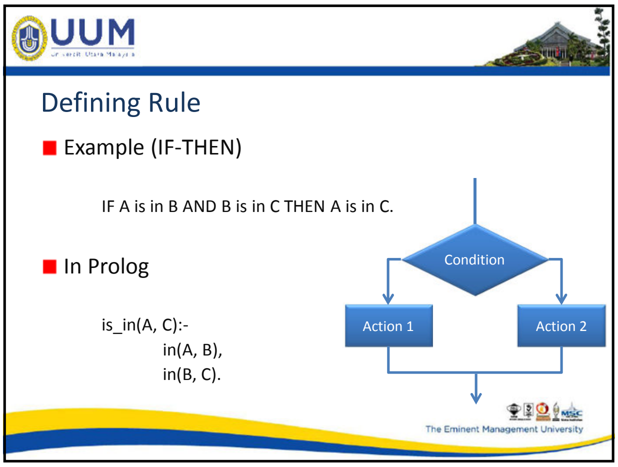
Defining Rule

- Example (Logical statement)

for all X and Y,
X is the mother of Y if
X is a parent of Y and
X is a female.

```
mother(X,Y):-
    parent(X,Y),
    female(X).
```

The Eminent Management University



Defining Rule

- Example (IF-THEN)

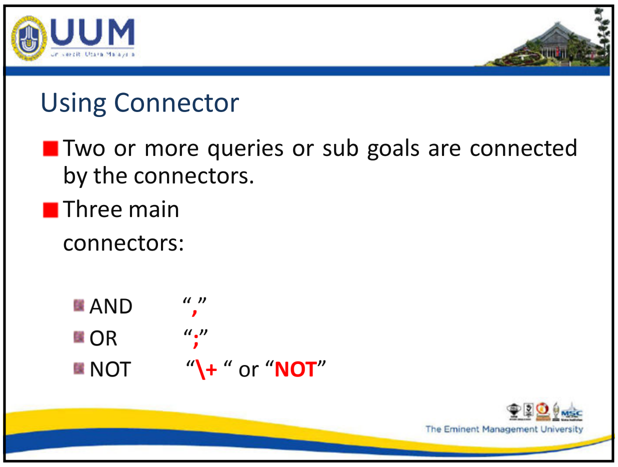
IF A is in B AND B is in C THEN A is in C.

- In Prolog

```
is_in(A, C):-
        in(A, B),
        in(B, C).
```

Condition
Action 1
Action 2

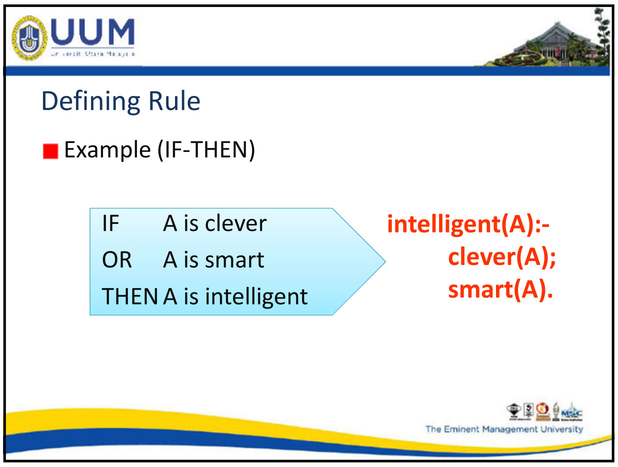
The Eminent Management University



Using Connector

- Two or more queries or sub goals are connected by the connectors.
- Three main connectors:
  - AND “,”
  - OR “;”
  - NOT “\+ “ or “NOT”

The Eminent Management University



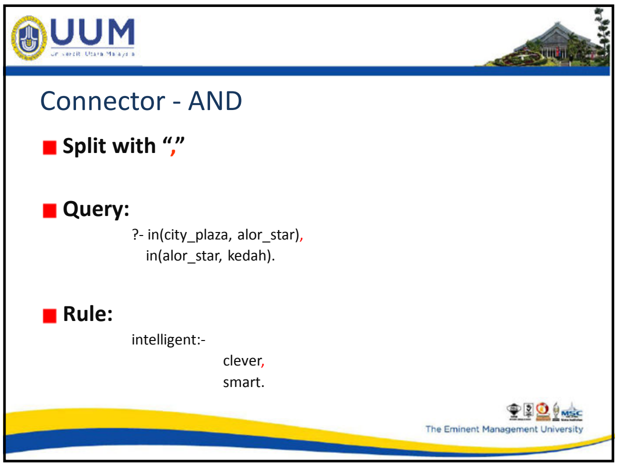
Defining Rule

- Example (IF-THEN)

IF A is clever
OR A is smart
THEN A is intelligent

```
intelligent(A):-
    clever(A);
    smart(A).
```

The Eminent Management University



Connector - AND

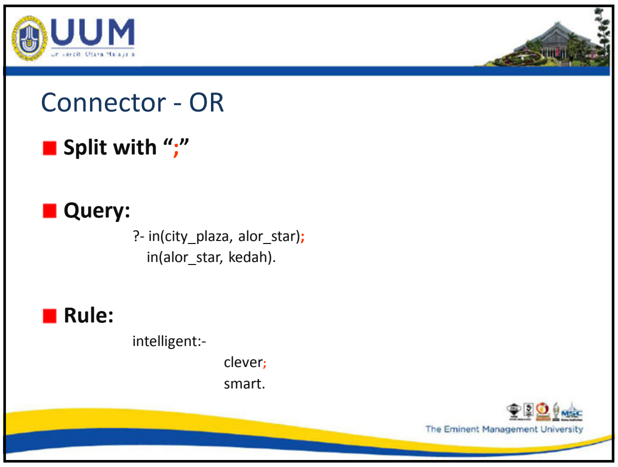
- **Split with “,”**
- **Query:**

```
?- in(city_plaza, alor_star),
   in(alor_star, kedah).
```

- **Rule:**

```
intelligent:-
            clever,
            smart.
```

The Eminent Management University



Connector - OR

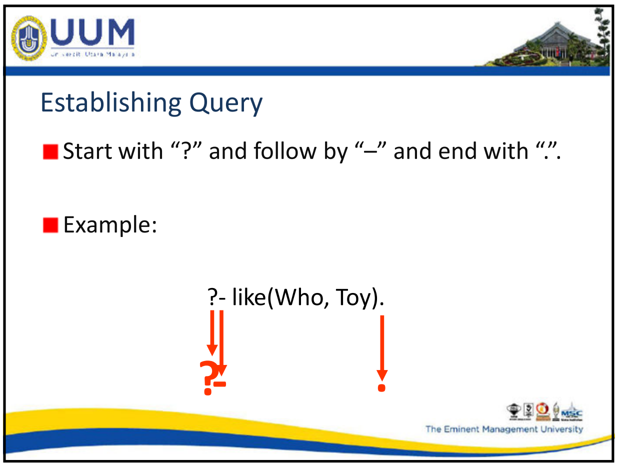
- **Split with ";"**
- **Query:**

```
?- in(city_plaza, alor_star);
   in(alor_star, kedah).
```

- **Rule:**

```
intelligent:-
        clever;
        smart.
```

The Eminent Management University



Establishing Query

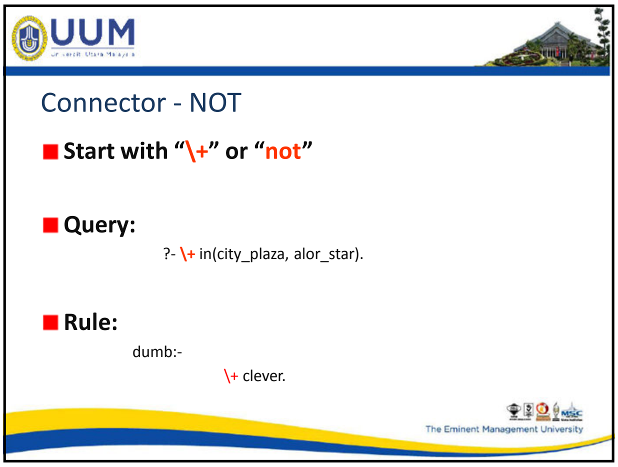
- Start with "?" and follow by "–" and end with ".".
- Example:

?- like(Who, Toy).

?-

.

The Eminent Management University



Connector - NOT

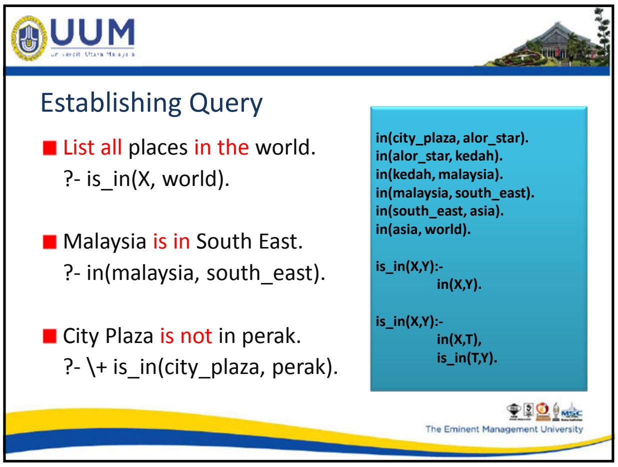
- **Start with "\+" or "not"**
- **Query:**

```
?- \+ in(city_plaza, alor_star).
```

- **Rule:**

```
dumb:-
        \+ clever.
```

The Eminent Management University

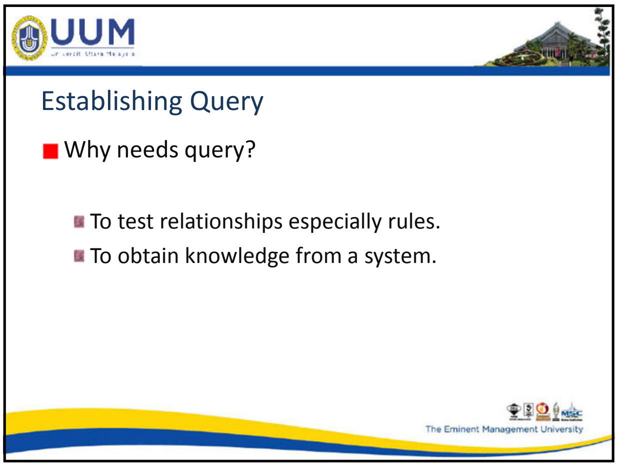


Establishing Query

- List all places in the world.
  ?- is_in(X, world).
- Malaysia is in South East.
  ?- in(malaysia, south_east).
- City Plaza is not in perak.
  ?- \+ is_in(city_plaza, perak).

```
in(city_plaza, alor_star).
in(alor_star, kedah).
in(kedah, malaysia).
in(malaysia, south_east).
in(south_east, asia).
in(asia, world).

is_in(X,Y):-
        in(X,Y).

is_in(X,Y):-
        in(X,T),
        is_in(T,Y).
```

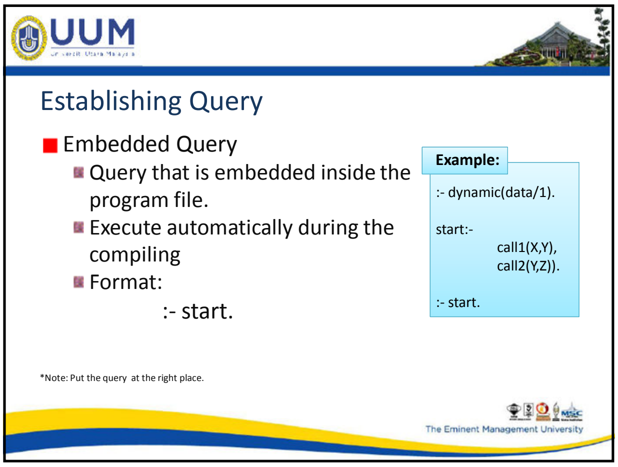
The Eminent Management University



Establishing Query

- Why needs query?
  - To test relationships especially rules.
  - To obtain knowledge from a system.

The Eminent Management University



Establishing Query

- Embedded Query
  - Query that is embedded inside the program file.
  - Execute automatically during the compiling
  - Format:

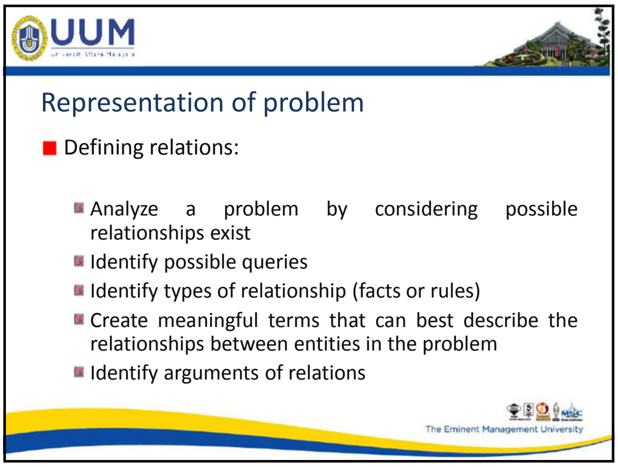
    :- start.

Example:

```
:- dynamic(data/1).

start:-
        call1(X,Y),
        call2(Y,Z)).

:- start.
```

*Note: Put the query at the right place.

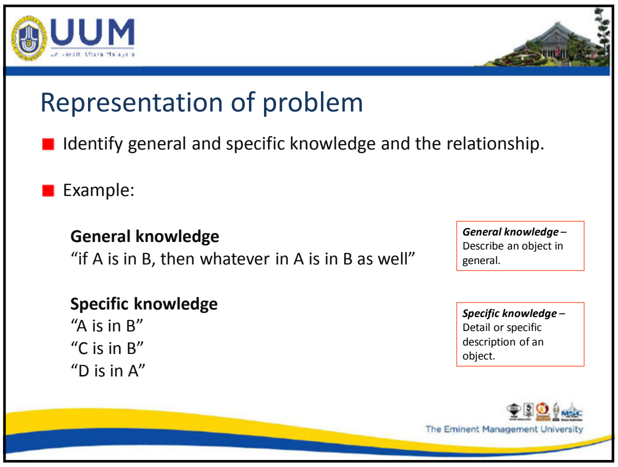
The Eminent Management University



Representation of problem

- Defining relations:
  - Analyze a problem by considering possible relationships exist
  - Identify possible queries
  - Identify types of relationship (facts or rules)
  - Create meaningful terms that can best describe the relationships between entities in the problem
  - Identify arguments of relations

The Eminent Management University



Representation of problem

- Identify general and specific knowledge and the relationship.
- Example:

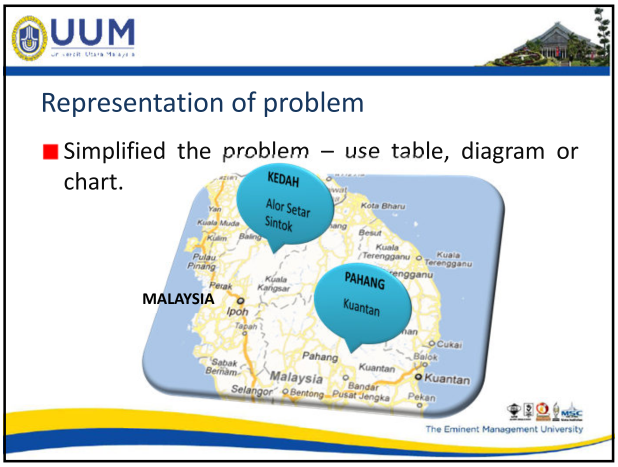
General knowledge
"if A is in B, then whatever in A is in B as well"

General knowledge – Describe an object in general.

Specific knowledge
"A is in B"
"C is in B"
"D is in A"

Specific knowledge – Detail or specific description of an object.

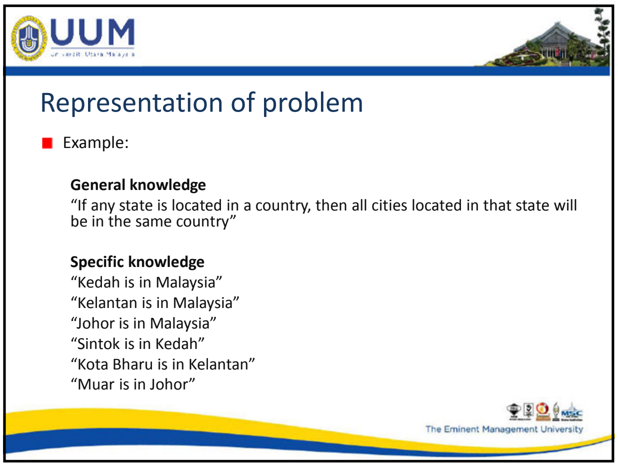
The Eminent Management University



Representation of problem

- Simplified the problem – use table, diagram or chart.

The Eminent Management University



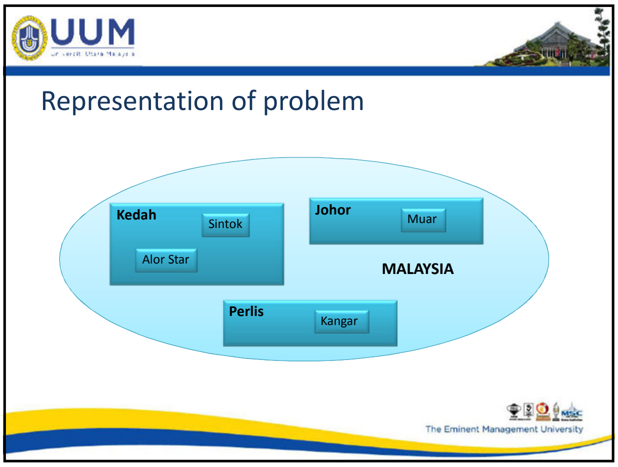
Representation of problem

- Example:

General knowledge
"If any state is located in a country, then all cities located in that state will be in the same country"

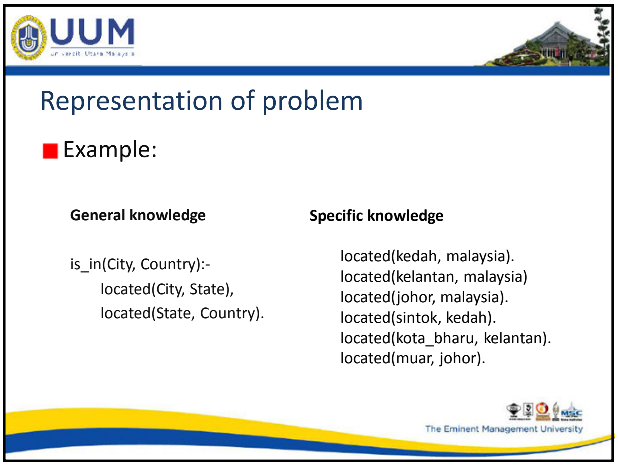
Specific knowledge
"Kedah is in Malaysia"
"Kelantan is in Malaysia"
"Johor is in Malaysia"
"Sintok is in Kedah"
"Kota Bharu is in Kelantan"
"Muar is in Johor"

The Eminent Management University



Representation of problem

The Eminent Management University



Representation of problem

- Example:

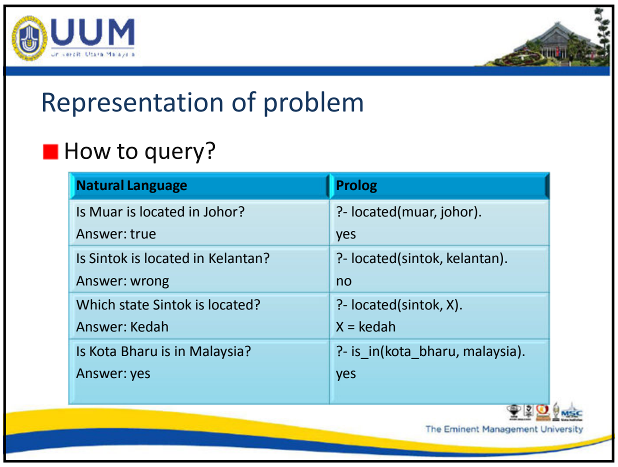
General knowledge

```
is_in(City, Country):-
      located(City, State),
      located(State, Country).
```

Specific knowledge

```
located(kedah, malaysia).
located(kelantan, malaysia)
located(johor, malaysia).
located(sintok, kedah).
located(kota_bharu, kelantan).
located(muar, johor).
```

The Eminent Management University



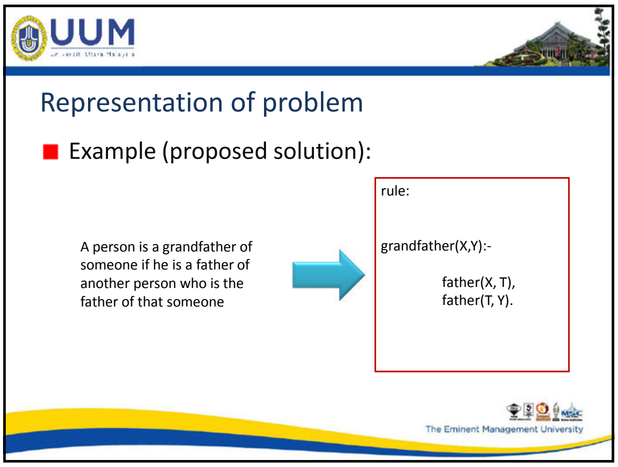
UUM

Representation of problem

- How to query?

Natural Language	Prolog
Is Muar is located in Johor? Answer: true	?- located(muar, johor). yes
Is Sintok is located in Kelantan? Answer: wrong	?- located(sintok, kelantan). no
Which state Sintok is located? Answer: Kedah	?- located(sintok, X). X = kedah
Is Kota Bharu is in Malaysia? Answer: yes	?- is_in(kota_bharu, malaysia). yes

The Eminent Management University



UUM

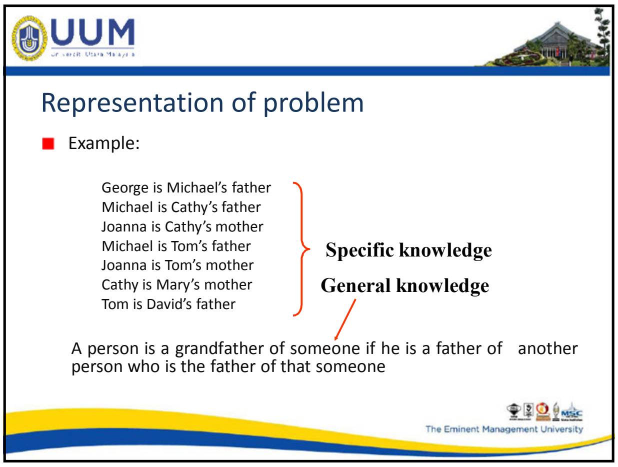
Representation of problem

- Example (proposed solution):

A person is a grandfather of someone if he is a father of another person who is the father of that someone

```
rule:

grandfather(X,Y):-
        father(X, T),
        father(T, Y).
```

The Eminent Management University



UUM

Representation of problem

- Example:

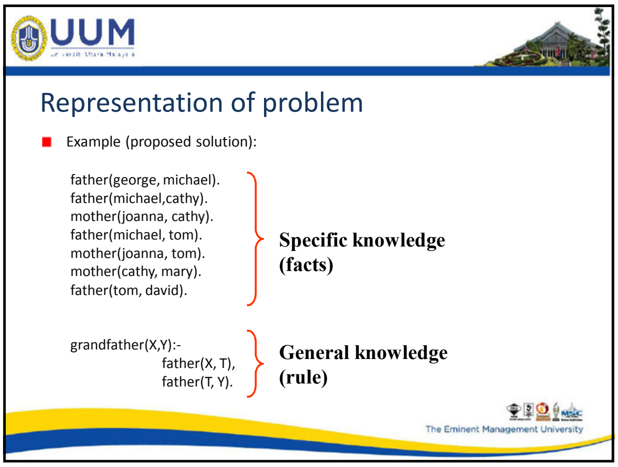
George is Michael's father
Michael is Cathy's father
Joanna is Cathy's mother
Michael is Tom's father
Joanna is Tom's mother
Cathy is Mary's mother
Tom is David's father

Specific knowledge

General knowledge

A person is a grandfather of someone if he is a father of another person who is the father of that someone

The Eminent Management University



UUM

Representation of problem

- Example (proposed solution):

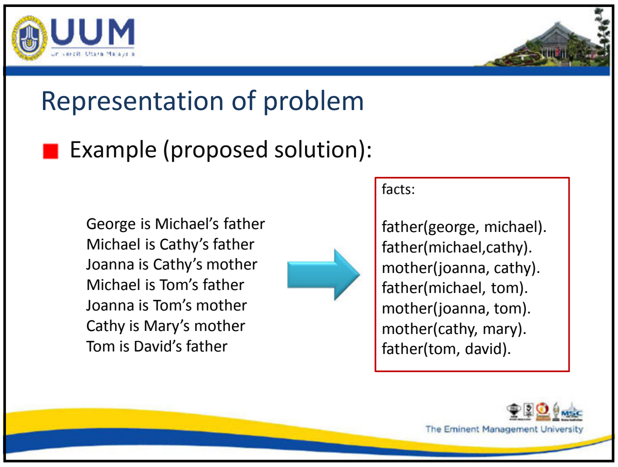
```
father(george, michael).
father(michael,cathy).
mother(joanna, cathy).
father(michael, tom).
mother(joanna, tom).
mother(cathy, mary).
father(tom, david).
```

Specific knowledge (facts)

```
grandfather(X,Y):-
            father(X, T),
            father(T, Y).
```

General knowledge (rule)

The Eminent Management University



UUM

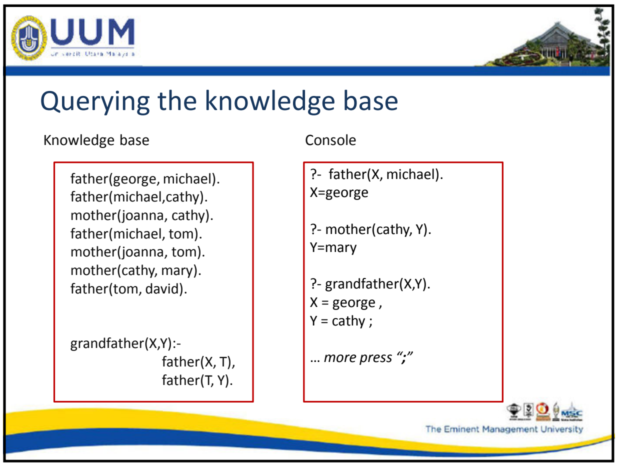
Representation of problem

- Example (proposed solution):

George is Michael's father
Michael is Cathy's father
Joanna is Cathy's mother
Michael is Tom's father
Joanna is Tom's mother
Cathy is Mary's mother
Tom is David's father

```
facts:

father(george, michael).
father(michael,cathy).
mother(joanna, cathy).
father(michael, tom).
mother(joanna, tom).
mother(cathy, mary).
father(tom, david).
```

The Eminent Management University



UUM

Querying the knowledge base

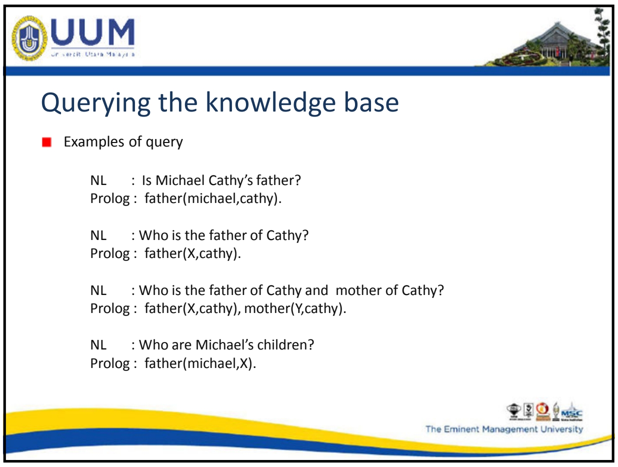
Knowledge base

```
father(george, michael).
father(michael,cathy).
mother(joanna, cathy).
father(michael, tom).
mother(joanna, tom).
mother(cathy, mary).
father(tom, david).


grandfather(X,Y):-
            father(X, T),
            father(T, Y).
```

Console

```
?-  father(X, michael).
X=george

?- mother(cathy, Y).
Y=mary

?- grandfather(X,Y).
X = george ,
Y = cathy ;

... more press ";"
```

The Eminent Management University



Querying the knowledge base

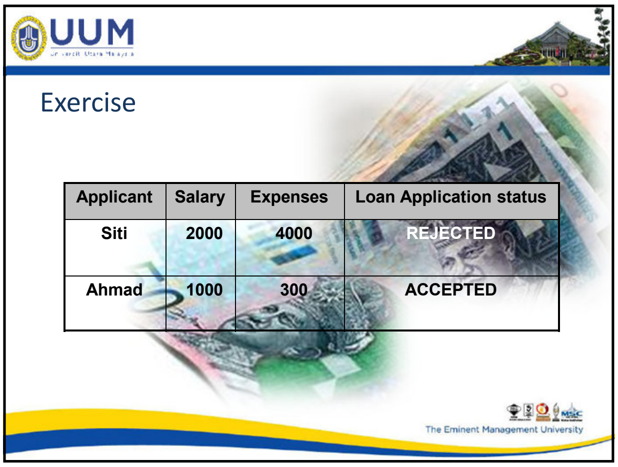
- Examples of query

NL : Is Michael Cathy's father?
Prolog : father(michael,cathy).

NL : Who is the father of Cathy?
Prolog : father(X,cathy).

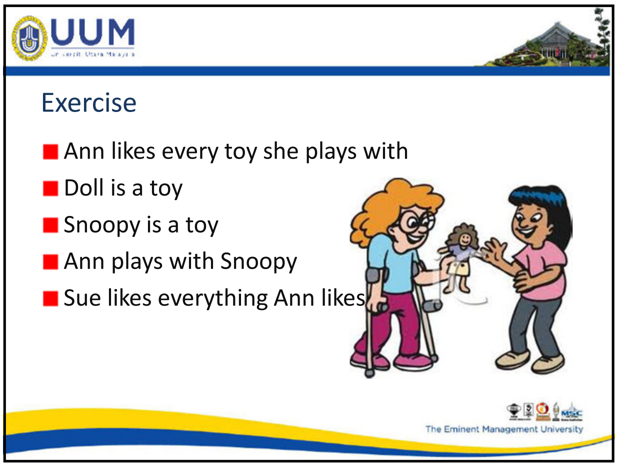
NL : Who is the father of Cathy and mother of Cathy?
Prolog : father(X,cathy), mother(Y,cathy).

NL : Who are Michael's children?
Prolog : father(michael,X).



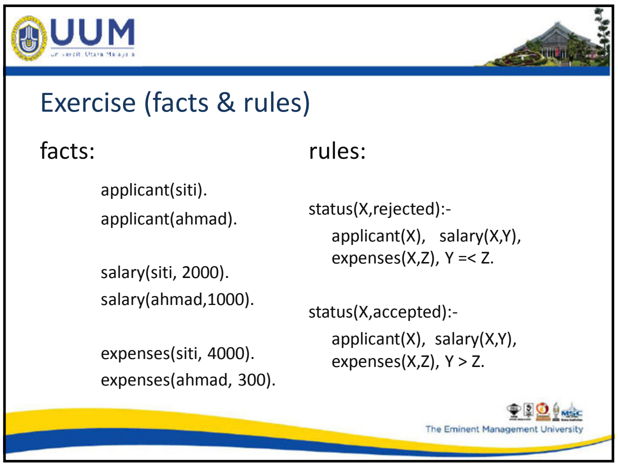
Exercise

Applicant	Salary	Expenses	Loan Application status
Siti	2000	4000	REJECTED
Ahmad	1000	300	ACCEPTED



Exercise

- Ann likes every toy she plays with
- Doll is a toy
- Snoopy is a toy
- Ann plays with Snoopy
- Sue likes everything Ann likes



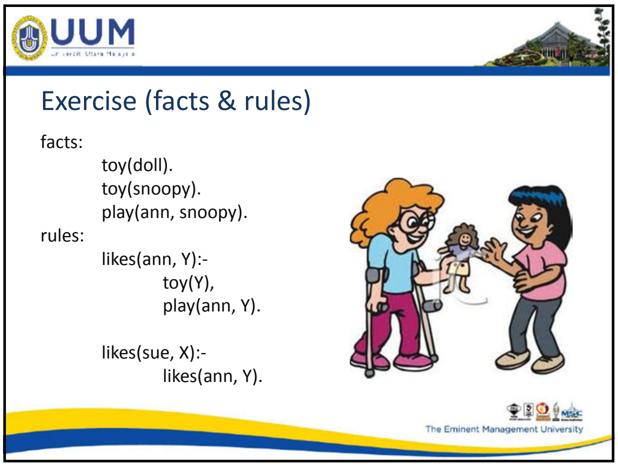
Exercise (facts & rules)

facts:

```
applicant(siti).
applicant(ahmad).

salary(siti, 2000).
salary(ahmad,1000).

expenses(siti, 4000).
expenses(ahmad, 300).
```

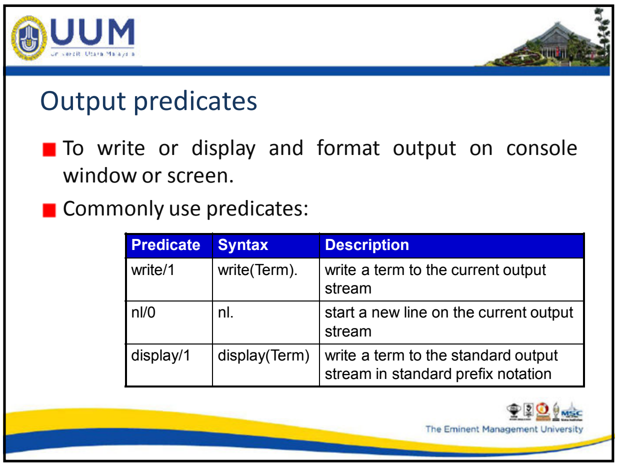
rules:

```
status(X,rejected):-
    applicant(X),  salary(X,Y),
    expenses(X,Z), Y =< Z.

status(X,accepted):-
    applicant(X), salary(X,Y),
    expenses(X,Z), Y > Z.
```



Exercise (facts & rules)

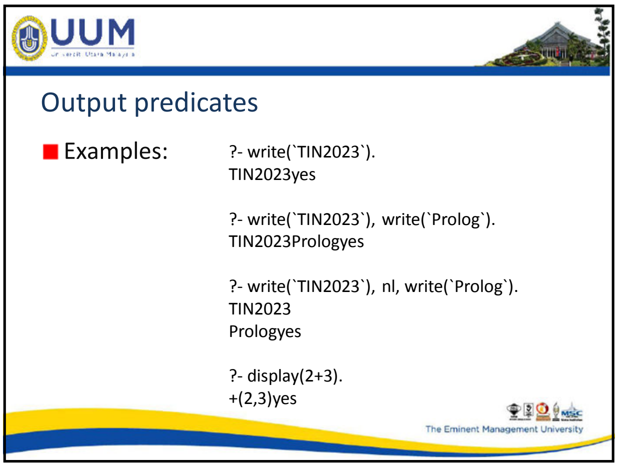
```
facts:
        toy(doll).
        toy(snoopy).
        play(ann, snoopy).
rules:
        likes(ann, Y):-
                toy(Y),
                play(ann, Y).

        likes(sue, X):-
                likes(ann, Y).
```



Output predicates

- To write or display and format output on console window or screen.
- Commonly use predicates:

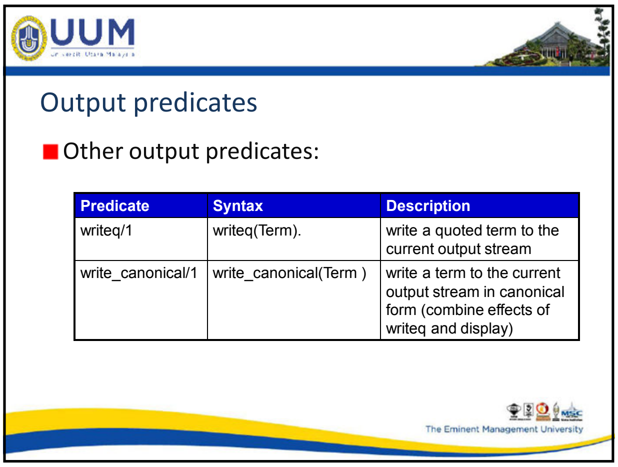
Predicate	Syntax	Description
write/1	write(Term).	write a term to the current output stream
nl/0	nl.	start a new line on the current output stream
display/1	display(Term)	write a term to the standard output stream in standard prefix notation


UUM
The Eminent Management University

Output predicates

- Examples:

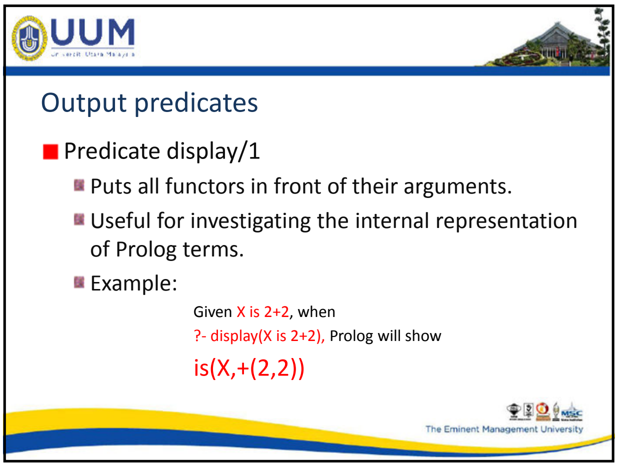
```
?- write(`TIN2023`).
TIN2023yes

?- write(`TIN2023`), write(`Prolog`).
TIN2023Prologyes

?- write(`TIN2023`), nl, write(`Prolog`).
TIN2023
Prologyes

?- display(2+3).
+(2,3)yes
```


UUM
The Eminent Management University

Output predicates

- Other output predicates:

Predicate	Syntax	Description
writeq/1	writeq(Term).	write a quoted term to the current output stream
write_canonical/1	write_canonical(Term)	write a term to the current output stream in canonical form (combine effects of writeq and display)


UUM
The Eminent Management University

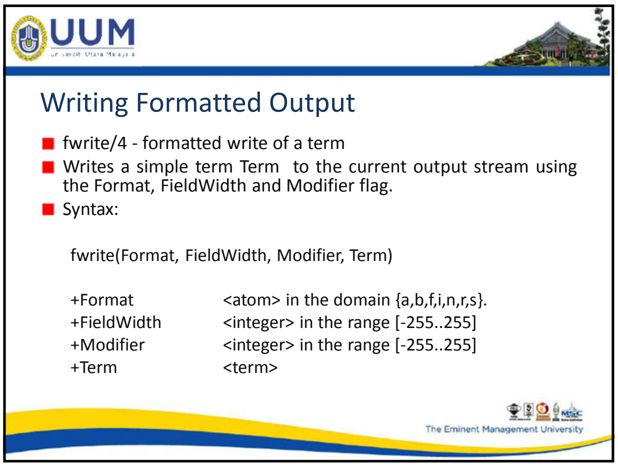
Output predicates

- Predicate display/1
  - Puts all functors in front of their arguments.
  - Useful for investigating the internal representation of Prolog terms.
  - Example:

Given X is 2+2, when

?- display(X is 2+2), Prolog will show

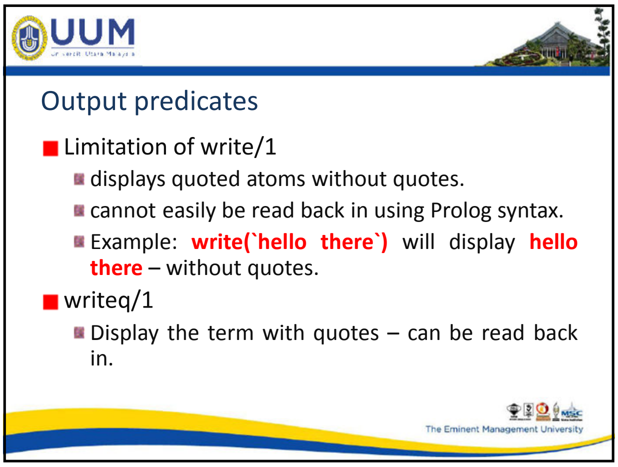
is(X,+(2,2))


UUM
The Eminent Management University

Writing Formatted Output

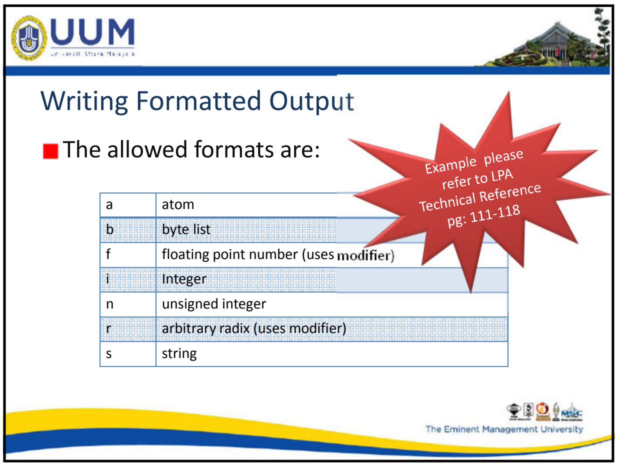
- fwrite/4 - formatted write of a term
- Writes a simple term Term to the current output stream using the Format, FieldWidth and Modifier flag.
- Syntax:

```
fwrite(Format, FieldWidth, Modifier, Term)

+Format        <atom> in the domain {a,b,f,i,n,r,s}.
+FieldWidth    <integer> in the range [-255..255]
+Modifier      <integer> in the range [-255..255]
+Term          <term>
```


UUM
The Eminent Management University

Output predicates

- Limitation of write/1
  - displays quoted atoms without quotes.
  - cannot easily be read back in using Prolog syntax.
  - Example: **write(`hello there`)** will display **hello there** – without quotes.
- writeq/1
  - Display the term with quotes – can be read back in.

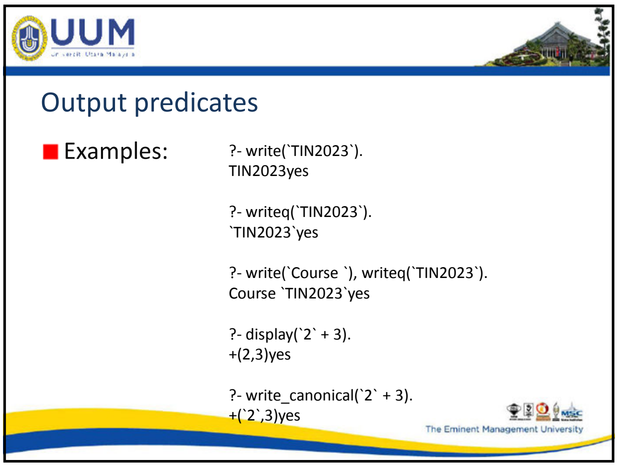

UUM
The Eminent Management University

Writing Formatted Output

- The allowed formats are:

Example please refer to LPA Technical Reference pg: 111-118

a	atom
b	byte list
f	floating point number (uses modifier)
i	Integer
n	unsigned integer
r	arbitrary radix (uses modifier)
s	string

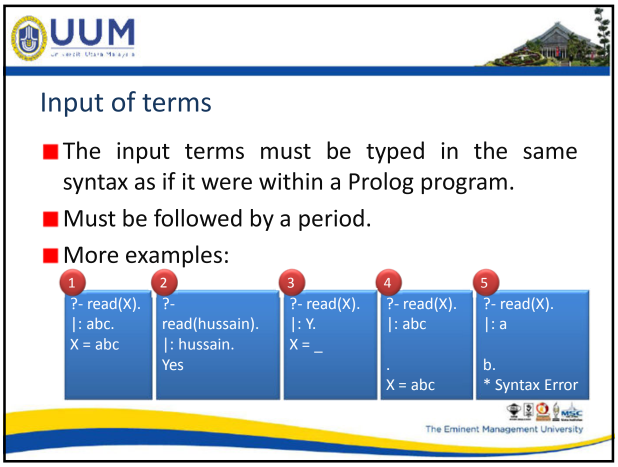


Output predicates

- Examples:

```
?- write(`TIN2023`).
TIN2023yes

?- writeq(`TIN2023`).
`TIN2023`yes

?- write(`Course `), writeq(`TIN2023`).
Course `TIN2023`yes

?- display(`2` + 3).
+(2,3)yes

?- write_canonical(`2` + 3).
+(`2`,3)yes
```

The Eminent Management University



Input of terms

- The input terms must be typed in the same syntax as if it were within a Prolog program.
- Must be followed by a period.
- More examples:

1
```
?- read(X).
|: abc.
X = abc
```

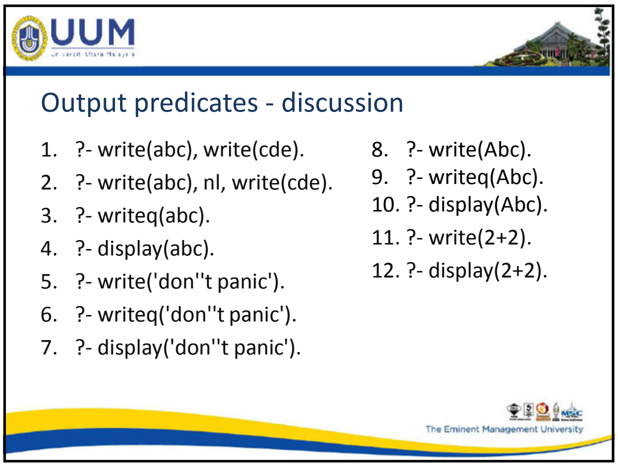
2
```
?-
read(hussain).
|: hussain.
Yes
```

3
```
?- read(X).
|: Y.
X = _
```

4
```
?- read(X).
|: abc

.
X = abc
```

5
```
?- read(X).
|: a

b.
* Syntax Error
```

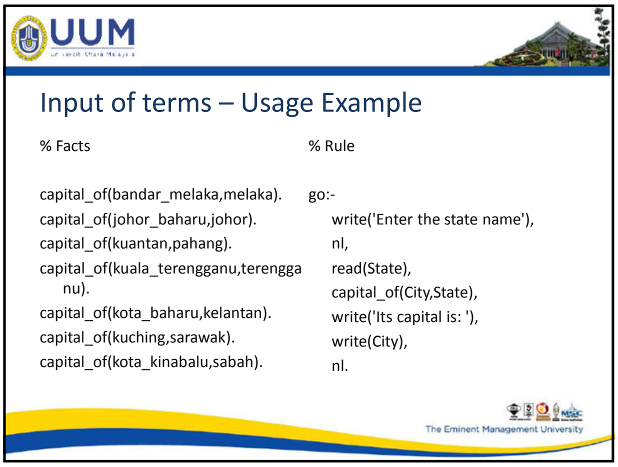
The Eminent Management University



Output predicates - discussion

1. ?- write(abc), write(cde).
2. ?- write(abc), nl, write(cde).
3. ?- writeq(abc).
4. ?- display(abc).
5. ?- write('don''t panic').
6. ?- writeq('don''t panic').
7. ?- display('don''t panic').
8. ?- write(Abc).
9. ?- writeq(Abc).
10. ?- display(Abc).
11. ?- write(2+2).
12. ?- display(2+2).

The Eminent Management University

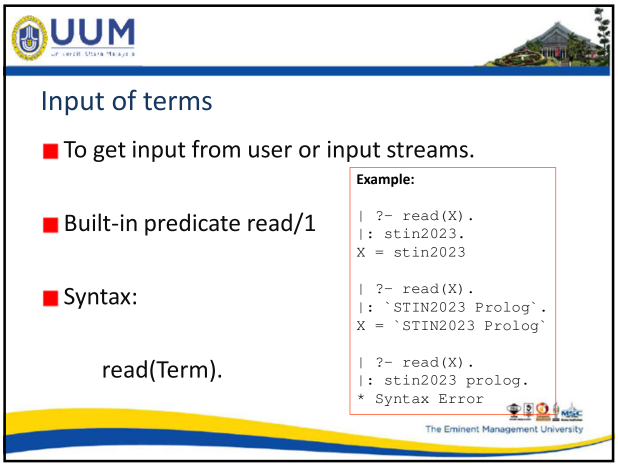


Input of terms – Usage Example

```
% Facts

capital_of(bandar_melaka,melaka).
capital_of(johor_baharu,johor).
capital_of(kuantan,pahang).
capital_of(kuala_terengganu,terengganu).
capital_of(kota_baharu,kelantan).
capital_of(kuching,sarawak).
capital_of(kota_kinabalu,sabah).
```

```
% Rule

go:-
    write('Enter the state name'),
    nl,
    read(State),
    capital_of(City,State),
    write('Its capital is: '),
    write(City),
    nl.
```

The Eminent Management University



Input of terms

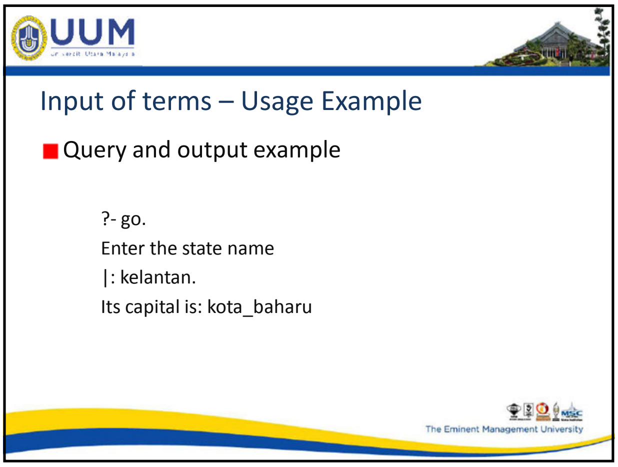
- To get input from user or input streams.
- Built-in predicate read/1
- Syntax:

read(Term).

Example:

```
| ?- read(X).
|: stin2023.
X = stin2023

| ?- read(X).
|: `STIN2023 Prolog`.
X = `STIN2023 Prolog`

| ?- read(X).
|: stin2023 prolog.
* Syntax Error
```

The Eminent Management University

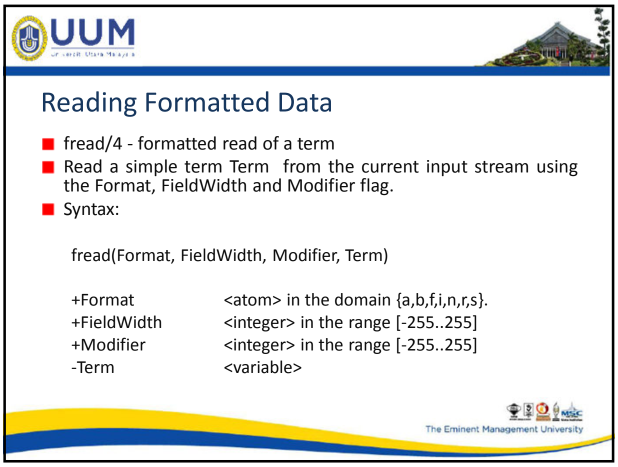


Input of terms – Usage Example

- Query and output example

```
?- go.
Enter the state name
|: kelantan.
Its capital is: kota_baharu
```

The Eminent Management University

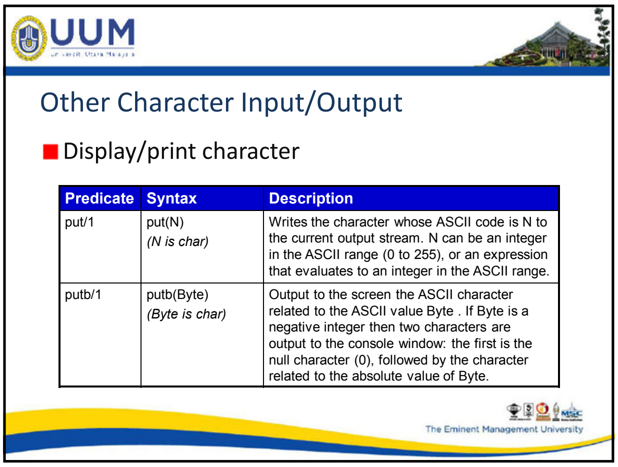


Reading Formatted Data

- fread/4 - formatted read of a term
- Read a simple term Term from the current input stream using the Format, FieldWidth and Modifier flag.
- Syntax:

```
fread(Format, FieldWidth, Modifier, Term)

+Format          <atom> in the domain {a,b,f,i,n,r,s}.
+FieldWidth      <integer> in the range [-255..255]
+Modifier        <integer> in the range [-255..255]
-Term            <variable>
```

The Eminent Management University

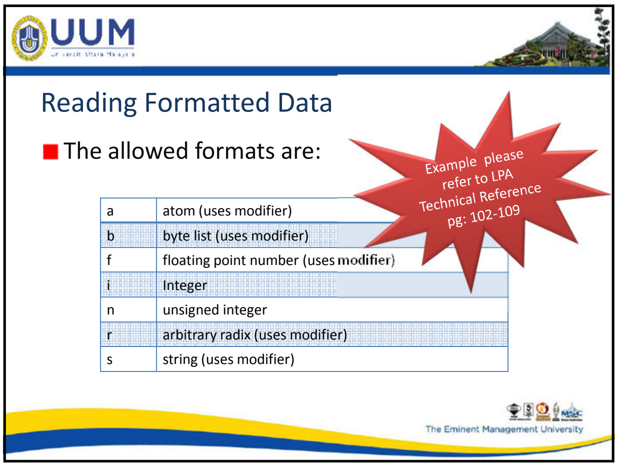


Other Character Input/Output

- Display/print character

Predicate	Syntax	Description
put/1	put(N) *(N is char)*	Writes the character whose ASCII code is N to the current output stream. N can be an integer in the ASCII range (0 to 255), or an expression that evaluates to an integer in the ASCII range.
putb/1	putb(Byte) *(Byte is char)*	Output to the screen the ASCII character related to the ASCII value Byte . If Byte is a negative integer then two characters are output to the console window: the first is the null character (0), followed by the character related to the absolute value of Byte.

The Eminent Management University



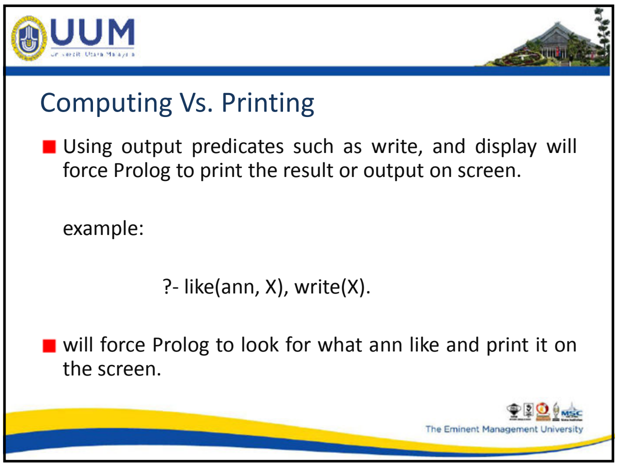
Reading Formatted Data

- The allowed formats are:

Example please refer to LPA Technical Reference pg: 102-109

a	atom (uses modifier)
b	byte list (uses modifier)
f	floating point number (uses modifier)
i	Integer
n	unsigned integer
r	arbitrary radix (uses modifier)
s	string (uses modifier)

The Eminent Management University



Computing Vs. Printing

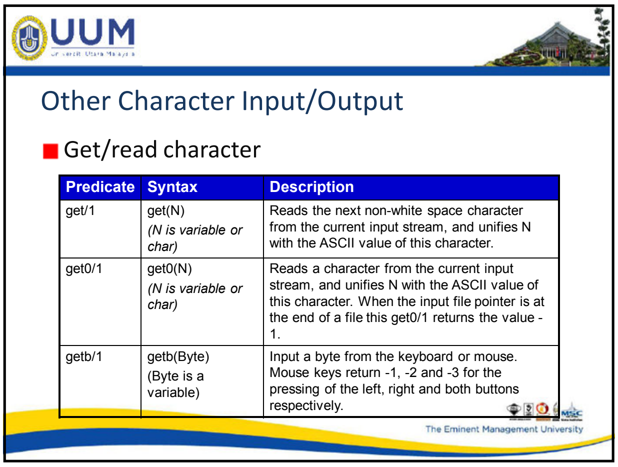
- Using output predicates such as write, and display will force Prolog to print the result or output on screen.

  example:

  ?- like(ann, X), write(X).

- will force Prolog to look for what ann like and print it on the screen.

The Eminent Management University

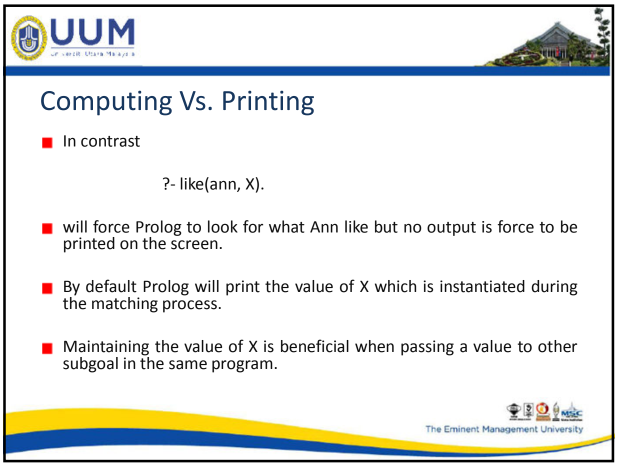


Other Character Input/Output

- Get/read character

Predicate	Syntax	Description
get/1	get(N) *(N is variable or char)*	Reads the next non-white space character from the current input stream, and unifies N with the ASCII value of this character.
get0/1	get0(N) *(N is variable or char)*	Reads a character from the current input stream, and unifies N with the ASCII value of this character. When the input file pointer is at the end of a file this get0/1 returns the value -1.
getb/1	getb(Byte) (Byte is a variable)	Input a byte from the keyboard or mouse. Mouse keys return -1, -2 and -3 for the pressing of the left, right and both buttons respectively.

The Eminent Management University



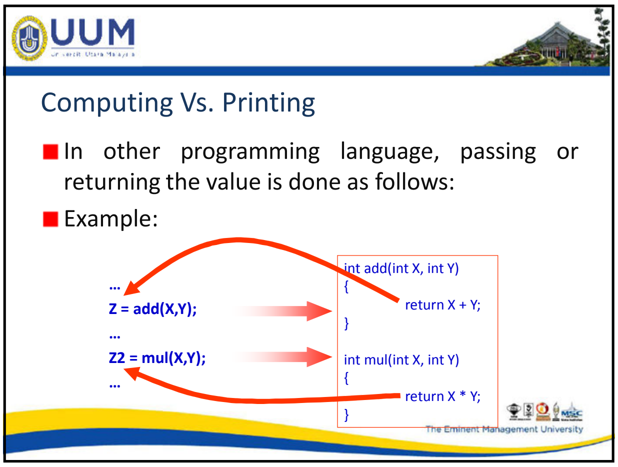
Computing Vs. Printing

- In contrast

  ?- like(ann, X).

- will force Prolog to look for what Ann like but no output is force to be printed on the screen.
- By default Prolog will print the value of X which is instantiated during the matching process.
- Maintaining the value of X is beneficial when passing a value to other subgoal in the same program.

The Eminent Management University

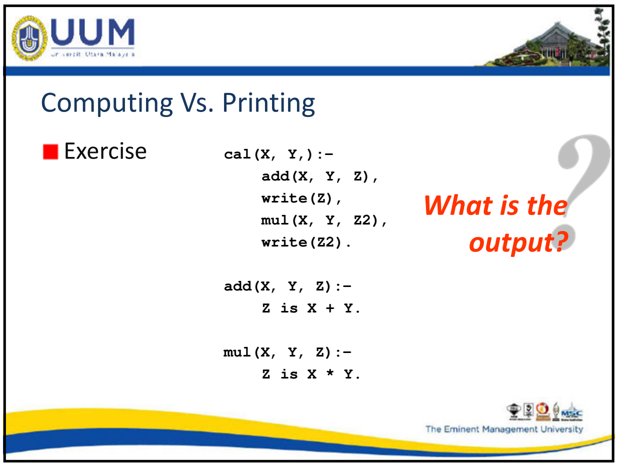


Computing Vs. Printing

- In other programming language, passing or returning the value is done as follows:
- Example:

```
...
Z = add(X,Y);
...
Z2 = mul(X,Y);
...
```

```
int add(int X, int Y)
{
        return X + Y;
}

int mul(int X, int Y)
{
        return X * Y;
}
```

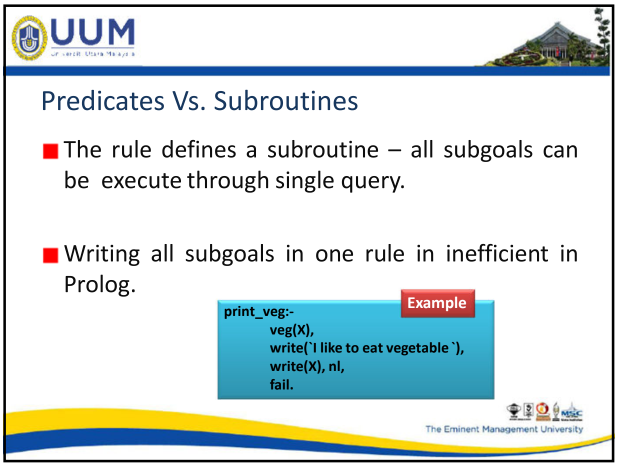


Computing Vs. Printing

- Exercise

```
cal(X, Y,):-
    add(X, Y, Z),
    write(Z),
    mul(X, Y, Z2),
    write(Z2).

add(X, Y, Z):-
    Z is X + Y.

mul(X, Y, Z):-
    Z is X * Y.
```

What is the output?

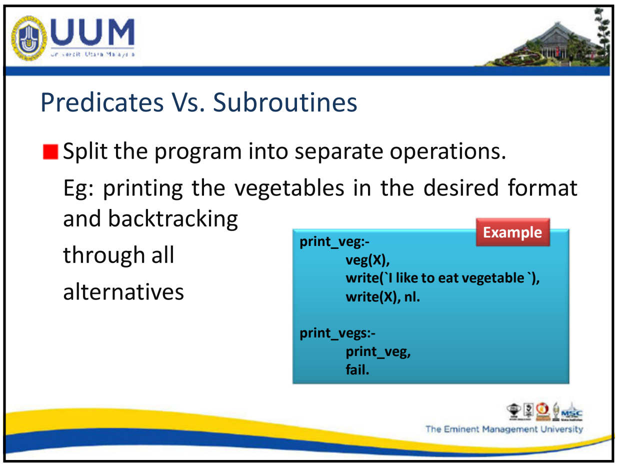


Predicates Vs. Subroutines

- The rule defines a subroutine – all subgoals can be execute through single query.
- Writing all subgoals in one rule in inefficient in Prolog.

Example

```
print_veg:-
        veg(X),
        write(`I like to eat vegetable `),
        write(X), nl,
        fail.
```



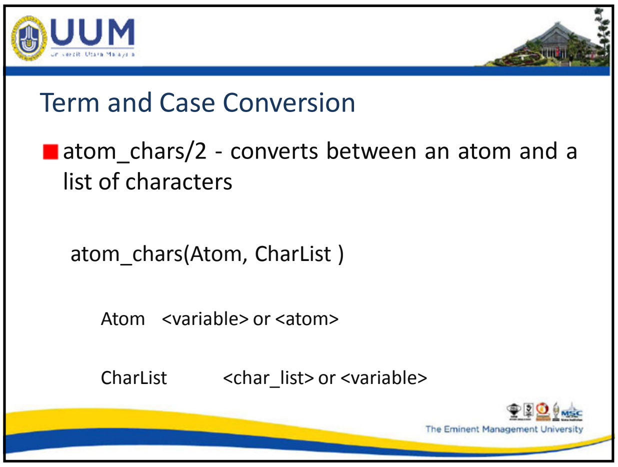
Predicates Vs. Subroutines

- Split the program into separate operations.

  Eg: printing the vegetables in the desired format and backtracking through all alternatives

Example

```
print_veg:-
        veg(X),
        write(`I like to eat vegetable `),
        write(X), nl.

print_vegs:-
        print_veg,
        fail.
```

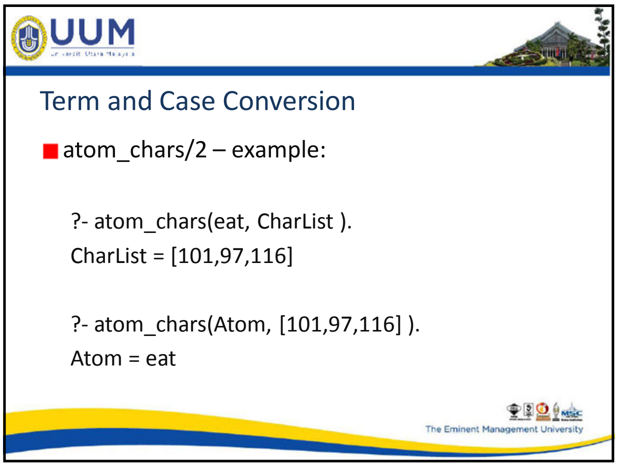


Term and Case Conversion

- atom_chars/2 - converts between an atom and a list of characters

atom_chars(Atom, CharList)

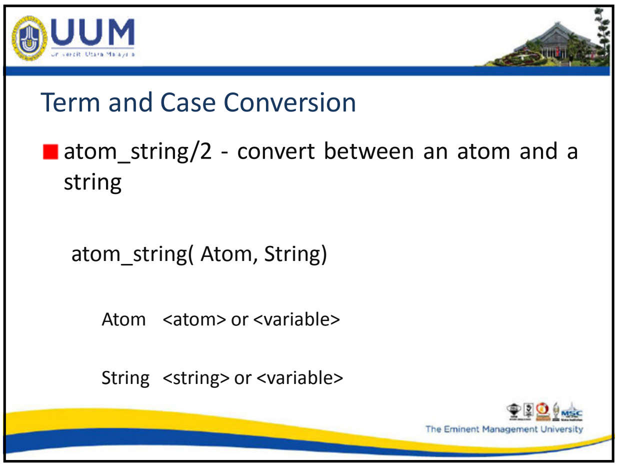
Atom	<variable> or <atom>
CharList	<char_list> or <variable>



Term and Case Conversion

- atom_chars/2 – example:

```
?- atom_chars(eat, CharList ).
CharList = [101,97,116]

?- atom_chars(Atom, [101,97,116] ).
Atom = eat
```



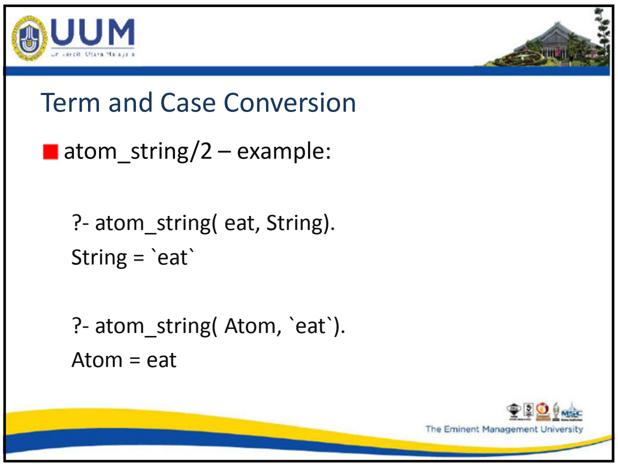
Term and Case Conversion

- atom_string/2 - convert between an atom and a string

atom_string(Atom, String)

Atom	<atom> or <variable>
String	<string> or <variable>

The Eminent Management University

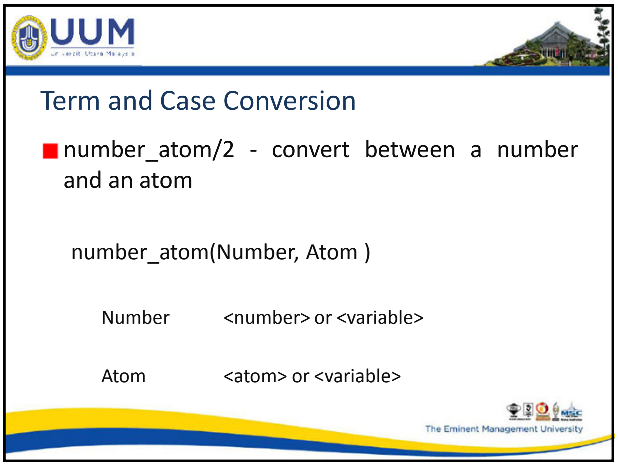


Term and Case Conversion

- atom_string/2 – example:

```
?- atom_string( eat, String).
String = `eat`

?- atom_string( Atom, `eat`).
Atom = eat
```

The Eminent Management University



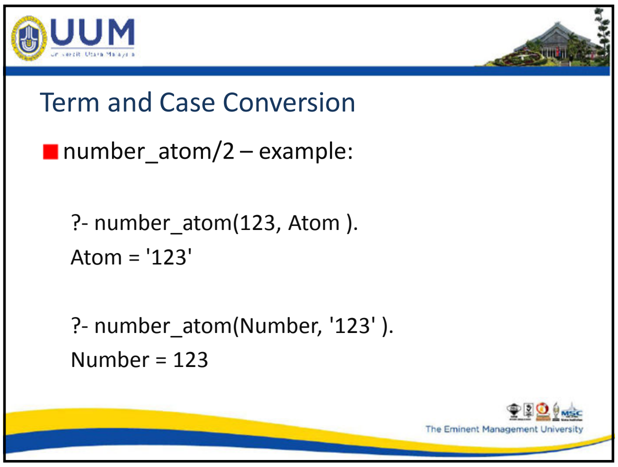
Term and Case Conversion

- number_atom/2 - convert between a number and an atom

number_atom(Number, Atom)

Number	<number> or <variable>
Atom	<atom> or <variable>

The Eminent Management University

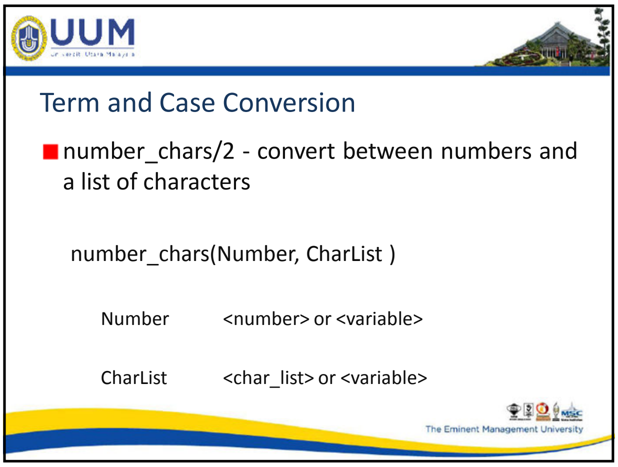


Term and Case Conversion

- number_atom/2 – example:

```
?- number_atom(123, Atom ).
Atom = '123'

?- number_atom(Number, '123' ).
Number = 123
```

The Eminent Management University



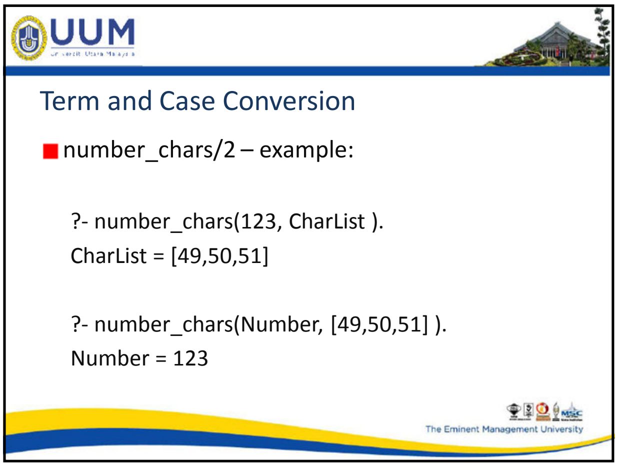
Term and Case Conversion

- number_chars/2 - convert between numbers and a list of characters

number_chars(Number, CharList)

Number	<number> or <variable>
CharList	<char_list> or <variable>

The Eminent Management University

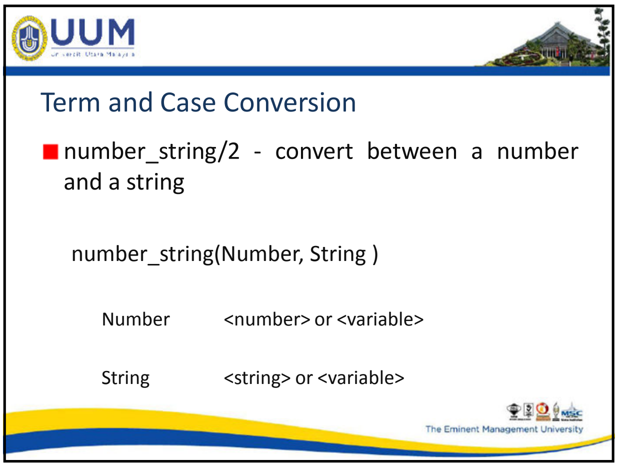


Term and Case Conversion

- number_chars/2 – example:

```
?- number_chars(123, CharList ).
CharList = [49,50,51]

?- number_chars(Number, [49,50,51] ).
Number = 123
```

The Eminent Management University

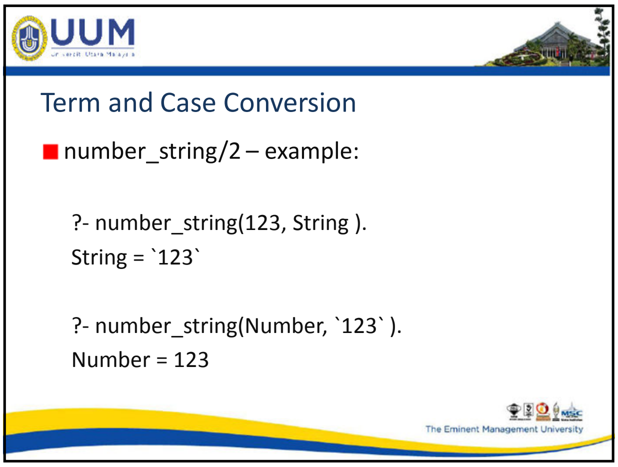


Term and Case Conversion

- number_string/2 - convert between a number and a string

```
number_string(Number, String )
```

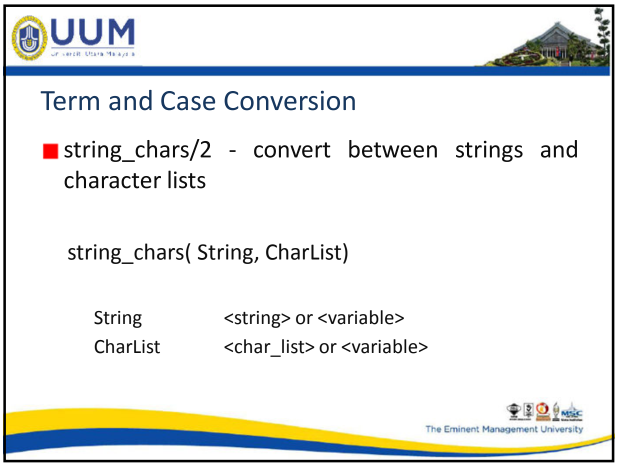
Number	<number> or <variable>
String	<string> or <variable>



Term and Case Conversion

- number_string/2 – example:

```
?- number_string(123, String ).
String = `123`

?- number_string(Number, `123` ).
Number = 123
```

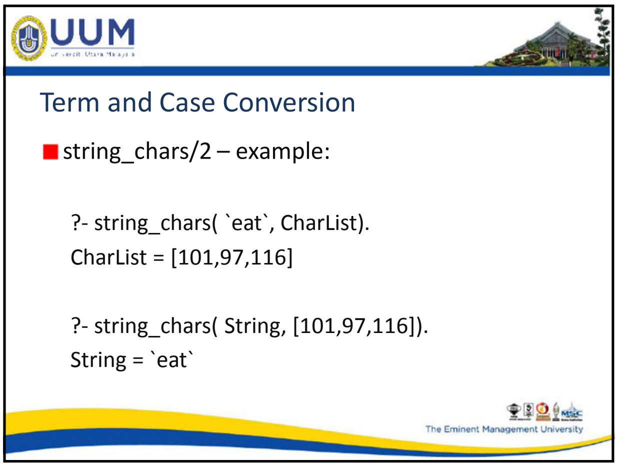


Term and Case Conversion

- string_chars/2 - convert between strings and character lists

```
string_chars( String, CharList)
```

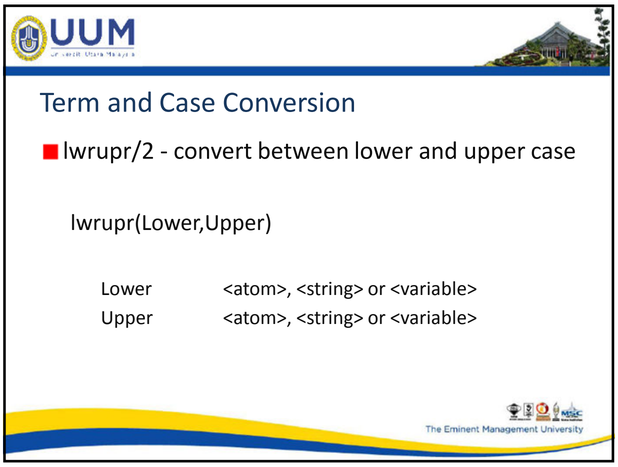
String	<string> or <variable>
CharList	<char_list> or <variable>



Term and Case Conversion

- string_chars/2 – example:

```
?- string_chars( `eat`, CharList).
CharList = [101,97,116]

?- string_chars( String, [101,97,116]).
String = `eat`
```

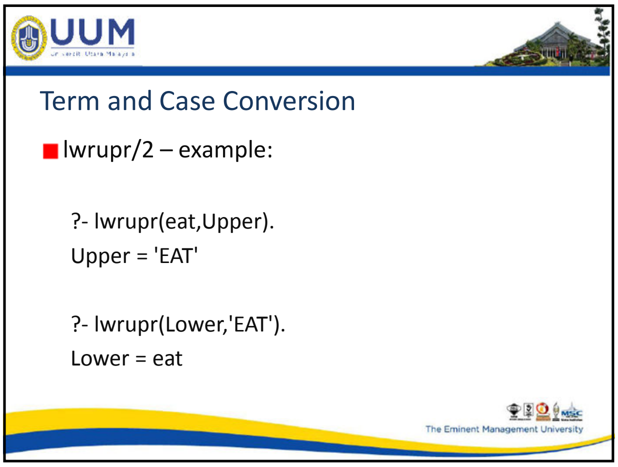


Term and Case Conversion

- lwrupr/2 - convert between lower and upper case

```
lwrupr(Lower,Upper)
```

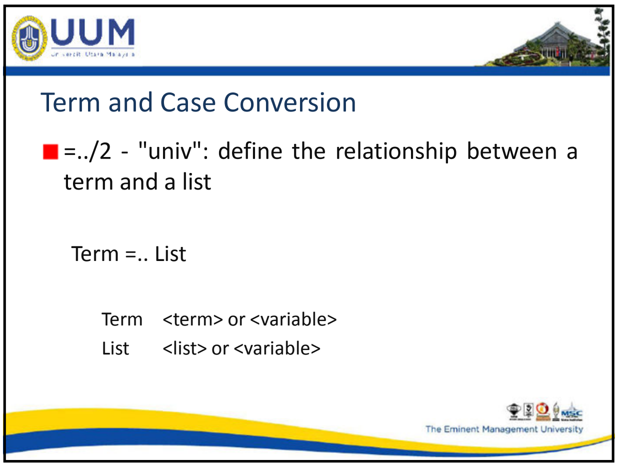
Lower	<atom>, <string> or <variable>
Upper	<atom>, <string> or <variable>



Term and Case Conversion

- lwrupr/2 – example:

```
?- lwrupr(eat,Upper).
Upper = 'EAT'

?- lwrupr(Lower,'EAT').
Lower = eat
```



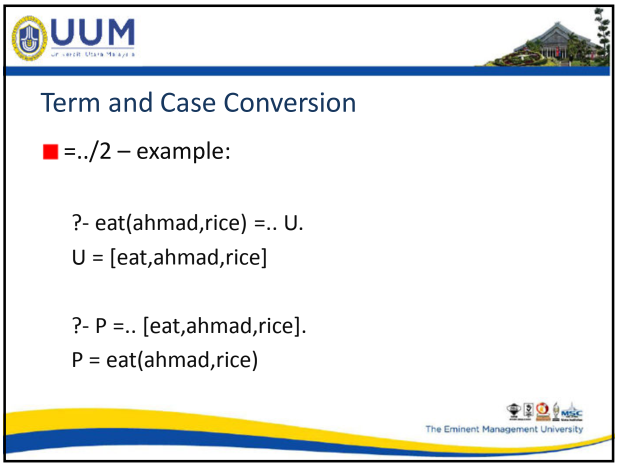
Term and Case Conversion

- =../2 - "univ": define the relationship between a term and a list

Term =.. List

Term	<term> or <variable>
List	<list> or <variable>

The Eminent Management University



Term and Case Conversion

- =../2 – example:

```
?- eat(ahmad,rice) =.. U.
U = [eat,ahmad,rice]

?- P =.. [eat,ahmad,rice].
P = eat(ahmad,rice)
```

The Eminent Management University