Variables

Declarative and Procedural Interpretations of Rules

- Rules have both a *declarative* and a *procedural* interpretation
- The declarative interpretation of the rule

  chases(X,Y):-dog(X),cat(Y),write(X),write(' chases '),write(Y),nl.

  is: '**chases(X,Y)** is true if **dog(X)** is true and **cat(Y)** is true and **write(X)** is true, etc.'
- The procedural interpretation is 'To satisfy **chases(X,Y)**, first satisfy **dog(X)**, then satisfy **cat(Y)**, then satisfy **write(X)**, etc.'

Declarative and Procedural Interpretations of Rules…

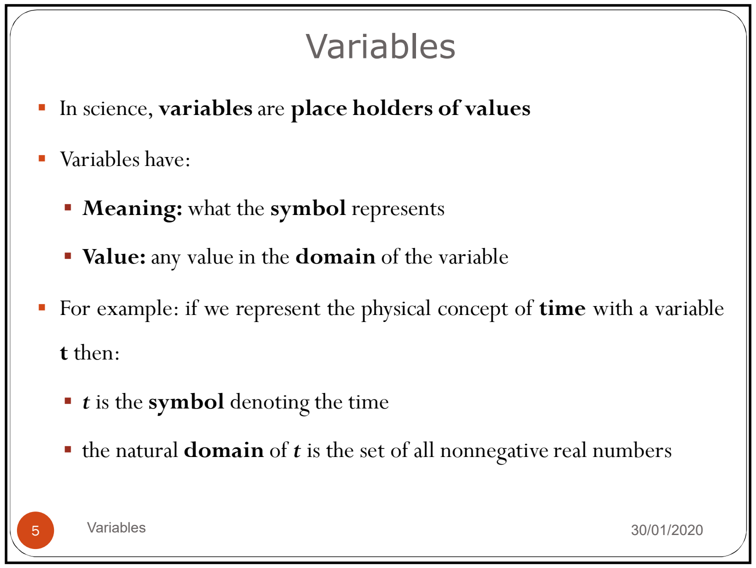
- Facts are generally interpreted *declaratively*.
- The fact

  dog(fido).

  is read as 'fido is a dog'.
- The order of the clauses defining a predicate and the order of the goals in the body of each rule are irrelevant to the declarative interpretation but of vital importance to the procedural interpretation and thus to determining whether or not the sequence of goals entered by the user at the system prompt is satisfied.

Predicates

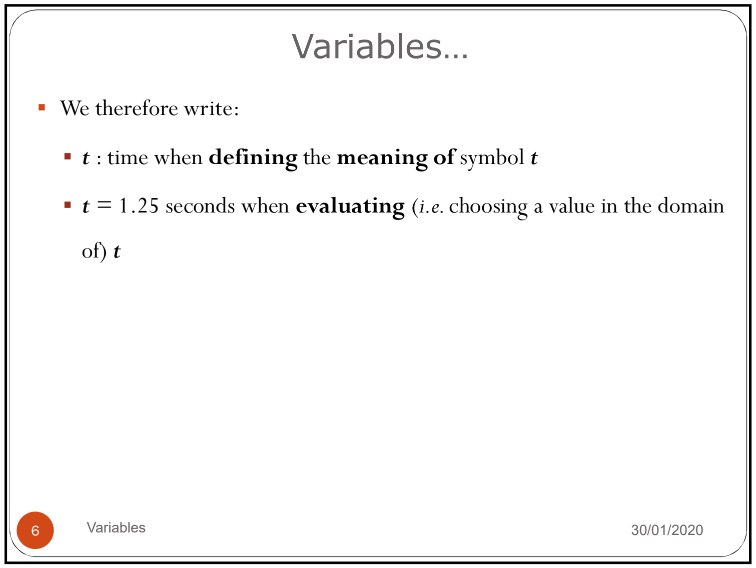
- The use of the term 'predicate' in Prolog is closely related to its use in mathematics.
- A **predicate** can be thought of as a relationship between a number of values (its arguments) such as *likes(henry,mary)* or *X=Y*, which can be either true or false.



Variables

- In science, **variables** are **place holders of values**
- Variables have:
  - **Meaning:** what the **symbol** represents
  - **Value:** any value in the **domain** of the variable
- For example: if we represent the physical concept of **time** with a variable **t** then:
  - *t* is the **symbol** denoting the time
  - the natural **domain** of *t* is the set of all nonnegative real numbers

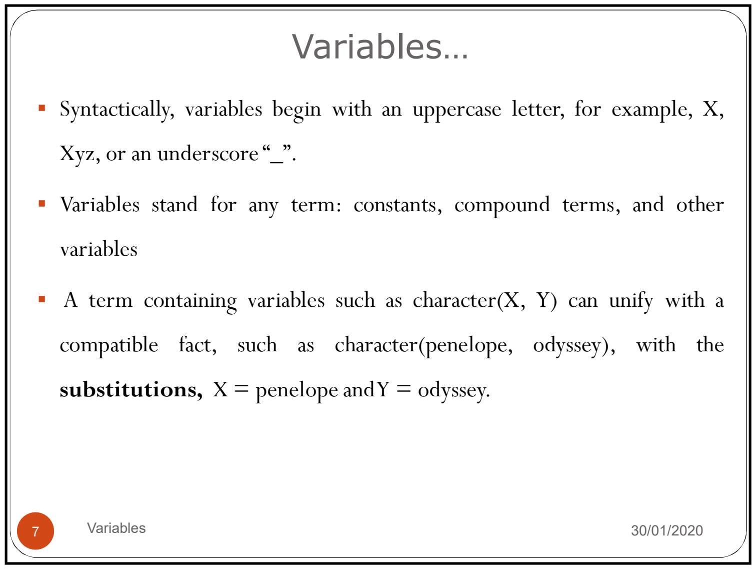
5 Variables 30/01/2020



Variables…

- We therefore write:
  - ***t*** : time when **defining** the **meaning of** symbol ***t***
  - ***t*** = 1.25 seconds when **evaluating** (*i.e.* choosing a value in the domain of) ***t***

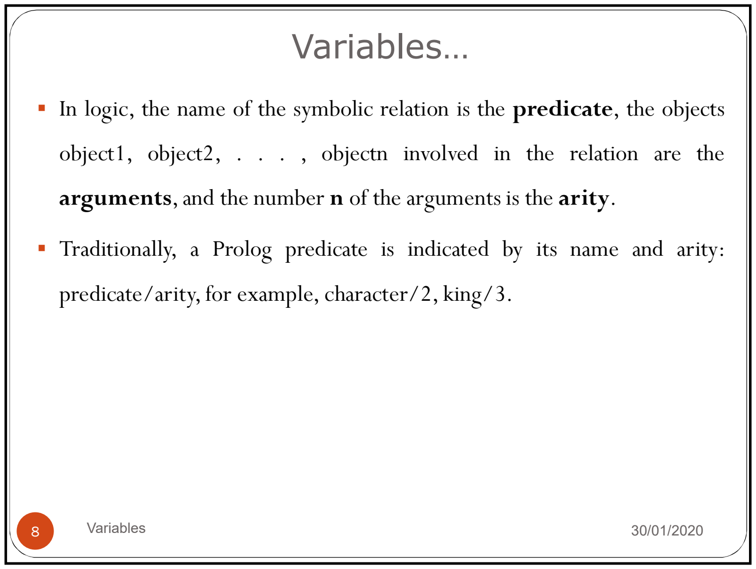
6 Variables 30/01/2020



Variables…

- Syntactically, variables begin with an uppercase letter, for example, X, Xyz, or an underscore "_".
- Variables stand for any term: constants, compound terms, and other variables
- A term containing variables such as character(X, Y) can unify with a compatible fact, such as character(penelope, odyssey), with the **substitutions,** X = penelope and Y = odyssey.

7 Variables 30/01/2020



Variables…

- In logic, the name of the symbolic relation is the **predicate**, the objects object1, object2, . . . , objectn involved in the relation are the **arguments**, and the number **n** of the arguments is the **arity**.
- Traditionally, a Prolog predicate is indicated by its name and arity: predicate/arity, for example, character/2, king/3.

8 Variables 30/01/2020

Variables…

- In Prolog, all forms of data are called **terms**. The constants, i.e., atoms or numbers, are terms
- The fact king(menelaus, sparta, achaean) is a **compound term** or a **structure**, that is, a term composed of other terms – **subterms**. The arguments of this compound term are constants
- A compound term consists of a **functor** – the name of the relation – and arguments. The leftmost functor of a term is the **principal functor**.

Variables…

- In a query a variable is a name used to stand for a term that is to be determined, e.g. variable *X* may stand for atom *dog*, the number 12.3, or a compound term or a list
- Variables can be used in the head or body of a clause and in goals entered at the system prompt. However, their interpretation depends on where they are used

Variables

- In science, **variables** are **place holders of values**
- Variables have:
  - **Meaning:** what the **symbol** represents
  - **Value:** any value in the **domain** of the variable
- For example: if we represent the physical concept of **time** with a variable **t** then:
  - *t* is the **symbol** denoting the time
  - the natural **domain** of *t* is the set of all nonnegative real numbers

Variables…

- A same principal functor with a different **arity** corresponds to different predicates: character/3 is thus different from character/2. A constant is a special case of a compound term with no arguments and an arity of 0. The constant abc can thus be referred to as abc/0.

Variables in goals

- Variables in goals can be interpreted as meaning *'find values of the variables that make the goal satisfied*'.
- For example, the goal

  ?-large_animal(A).

  can be read as 'find a value of A such that **large_animal(A)** is satisfied'.

Binding Variables

- Initially all variables used in a clause are said to be ***unbound***, meaning that they do not have values
- When the Prolog system evaluates a goal some variables may be given values such as *dog*, *-6.4* etc. This is known as ***binding*** the variables
- A variable that has been bound may become unbound again and possibly then bound to a different value by the process of ***backtracking.***

Lexical Scope of Variables

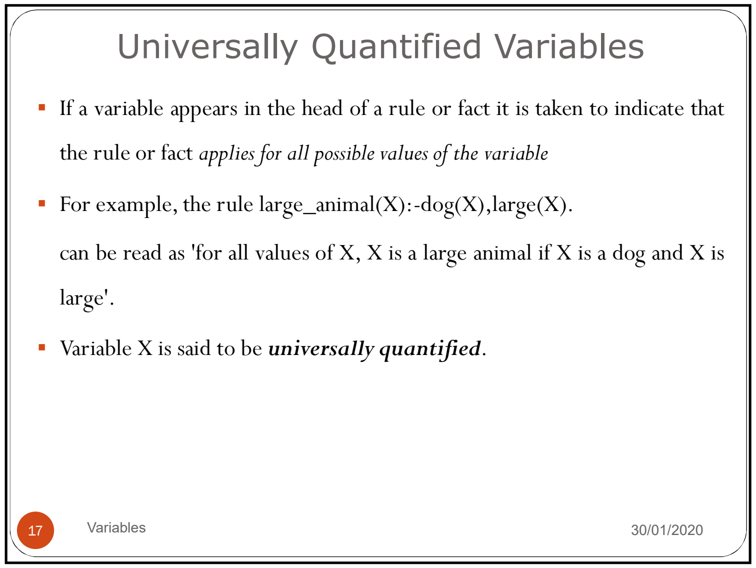
- In a clause such as parent(X,Y):-father(X,Y). The variables *X* and *Y* are entirely unrelated to any other variables with the same name used elsewhere
- All occurrences of variables *X* and *Y* in the clause can be replaced consistently by any other variables, e.g. by *First_person* and *Second_person* giving:

  parent(First_person,Second_person):- father(First_person,Second_person).
- This does not change the meaning of the clause (or the user's program) in any way
- This is often expressed by saying that the ***lexical scope*** of a variable is the clause in which it appears

Quantifiers

- Quantifiers are used to **declare ranges** of a predicate's variables in which the predicate assumes **either 0 or 1, but no both.**
- A predicate with no quantifiers is said to be a **free predicate**
- There are two **quantifiers:**

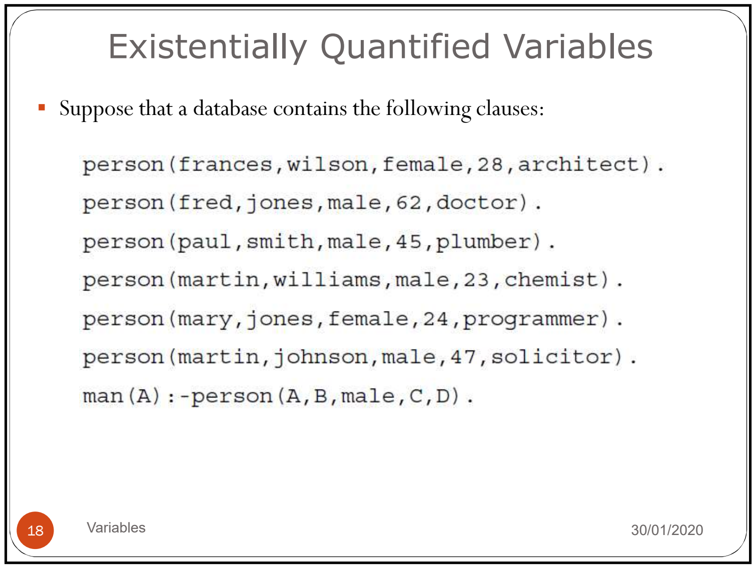
NAME	SEMANTIC	SYMBOL
Universal	FOR ALL	∀
Existential	THERE EXISTS	∃



Universally Quantified Variables

- If a variable appears in the head of a rule or fact it is taken to indicate that the rule or fact *applies for all possible values of the variable*
- For example, the rule large_animal(X):-dog(X),large(X).

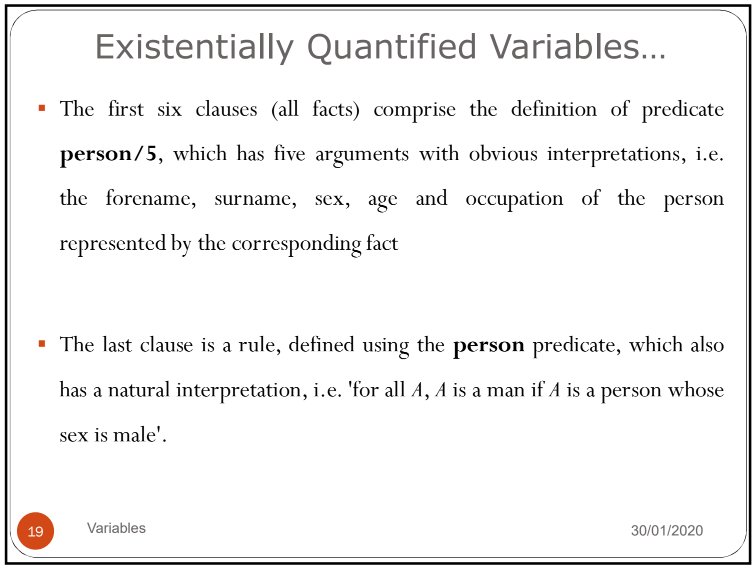
  can be read as 'for all values of X, X is a large animal if X is a dog and X is large'.
- Variable X is said to be ***universally quantified***.



Existentially Quantified Variables

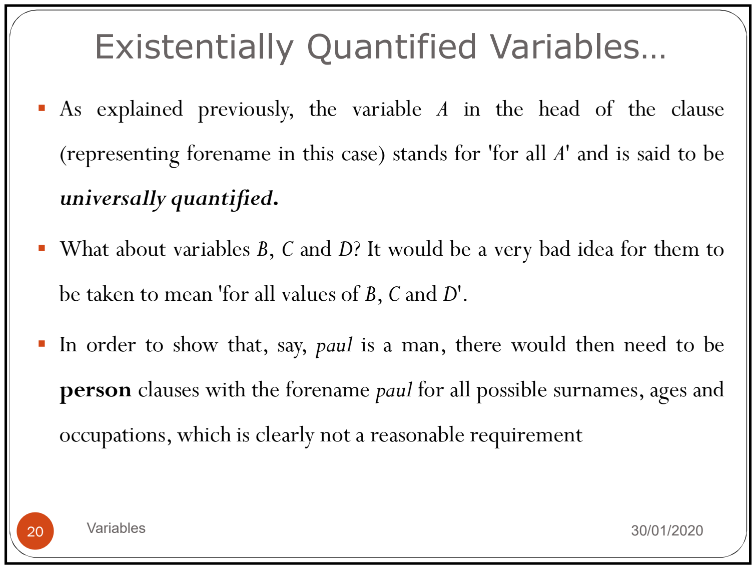
- Suppose that a database contains the following clauses:

```
person(frances,wilson,female,28,architect).
person(fred,jones,male,62,doctor).
person(paul,smith,male,45,plumber).
person(martin,williams,male,23,chemist).
person(mary,jones,female,24,programmer).
person(martin,johnson,male,47,solicitor).
man(A):-person(A,B,male,C,D).
```



Existentially Quantified Variables…

- The first six clauses (all facts) comprise the definition of predicate **person/5**, which has five arguments with obvious interpretations, i.e. the forename, surname, sex, age and occupation of the person represented by the corresponding fact
- The last clause is a rule, defined using the **person** predicate, which also has a natural interpretation, i.e. 'for all *A*, *A* is a man if *A* is a person whose sex is male'.



Existentially Quantified Variables…

- As explained previously, the variable *A* in the head of the clause (representing forename in this case) stands for 'for all *A*' and is said to be ***universally quantified.***
- What about variables *B*, *C* and *D*? It would be a very bad idea for them to be taken to mean 'for all values of *B*, *C* and *D*'.
- In order to show that, say, *paul* is a man, there would then need to be **person** clauses with the forename *paul* for all possible surnames, ages and occupations, which is clearly not a reasonable requirement

Existentially Quantified Variables…

- A far more helpful interpretation would be to take variable *B* to mean 'for at least one value of *B*' and similarly for variables *C* and *D*.
- This is the convention used by the Prolog system. Thus the final clause in the database means 'for all *A*, *A* is a man if there a person with forename *A*, surname *B*, sex male, age *C* and occupation *D*, for at least one value of *B*, *C* and *D*'

Existentially Quantified Variables…

- By virtue of the third **person** clause, *paul* qualifies as a man, with values *smith*, *45* and *plumber* for variables *B*, *C* and *D* respectively.

  ?- man(paul).

  yes
- The key distinction between variable *A* and variables *B*, *C* and *D* in the definition of predicate **man** is that *B*, *C* and *D* do not appear in the head of the clause

Existentially Quantified Variables…

- The convention used by Prolog is that if a variable, say *Y*, appears in the body of a clause but not in its head it is taken to mean 'there is (or there exists) at least one value of Y'. Such variables are said to be ***existentially quantified***
- Thus the rule

  dogowner(X):-dog(Y),owns(X,Y).

  can be interpreted as meaning 'for all values of X, X is a dog owner if there is some Y such that Y is a dog and X owns Y'.

The Anonymous Variable

- In order to find whether there is a clause corresponding to anyone called *paul* in the database, it is only necessary to enter a goal such as:

  ?- person(paul,Surname,Sex,Age,Occupation).

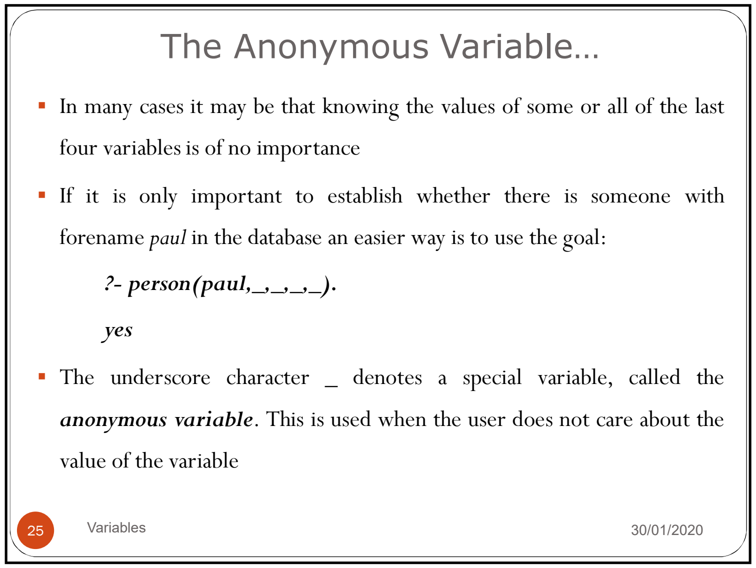
  at the prompt. Prolog replies as follows:

  Surname = smith ,

  Sex = male ,

  Age = 45 ,

  Occupation = plumber

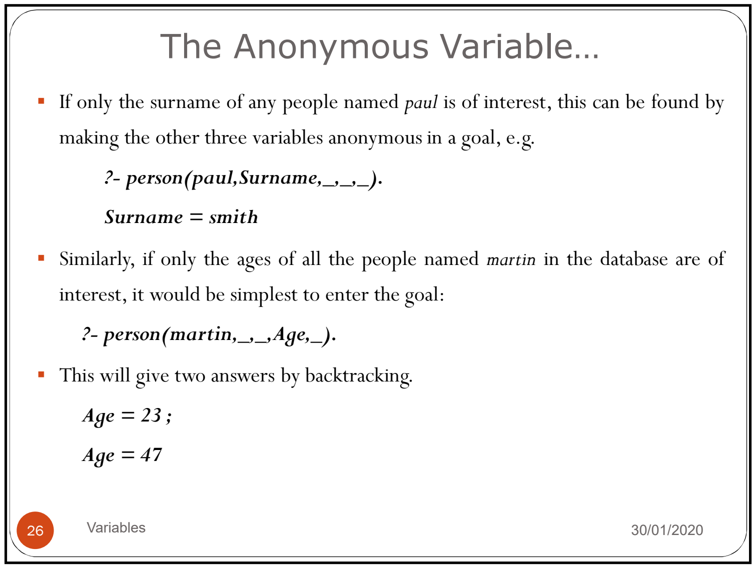


The Anonymous Variable…

- In many cases it may be that knowing the values of some or all of the last four variables is of no importance
- If it is only important to establish whether there is someone with forename *paul* in the database an easier way is to use the goal:

  ?- person(paul,_,_,_,_).

  yes
- The underscore character _ denotes a special variable, called the ***anonymous variable***. This is used when the user does not care about the value of the variable



The Anonymous Variable…

- If only the surname of any people named *paul* is of interest, this can be found by making the other three variables anonymous in a goal, e.g.

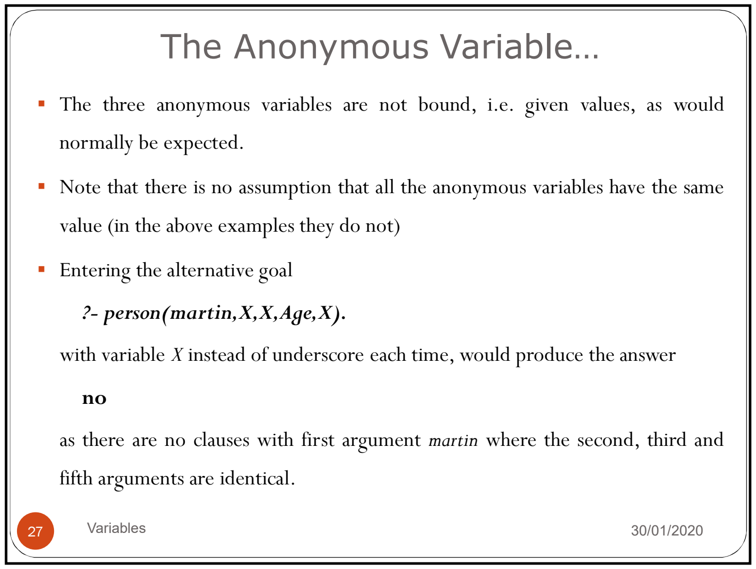
  ?- person(paul,Surname,_,_,_).

  Surname = smith
- Similarly, if only the ages of all the people named *martin* in the database are of interest, it would be simplest to enter the goal:

  ?- person(martin,_,_,Age,_).
- This will give two answers by backtracking.

  Age = 23 ;

  Age = 47



The Anonymous Variable…

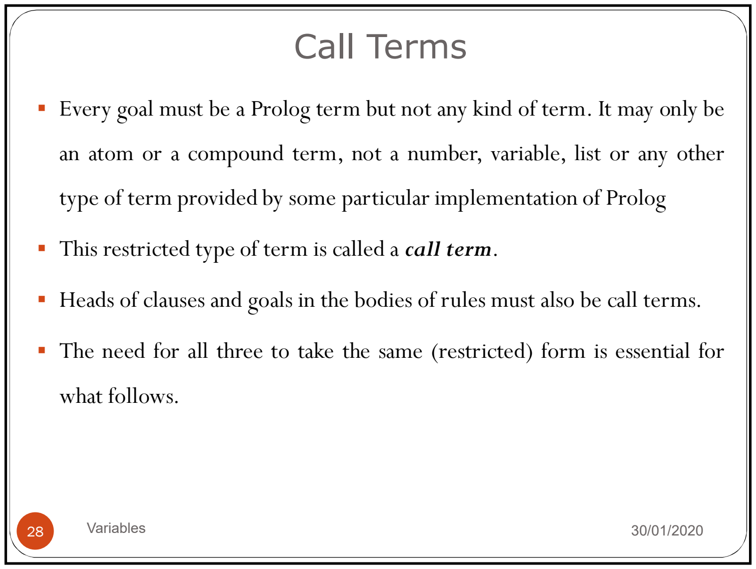
- The three anonymous variables are not bound, i.e. given values, as would normally be expected.
- Note that there is no assumption that all the anonymous variables have the same value (in the above examples they do not)
- Entering the alternative goal

  ?- person(martin,X,X,Age,X).

  with variable *X* instead of underscore each time, would produce the answer

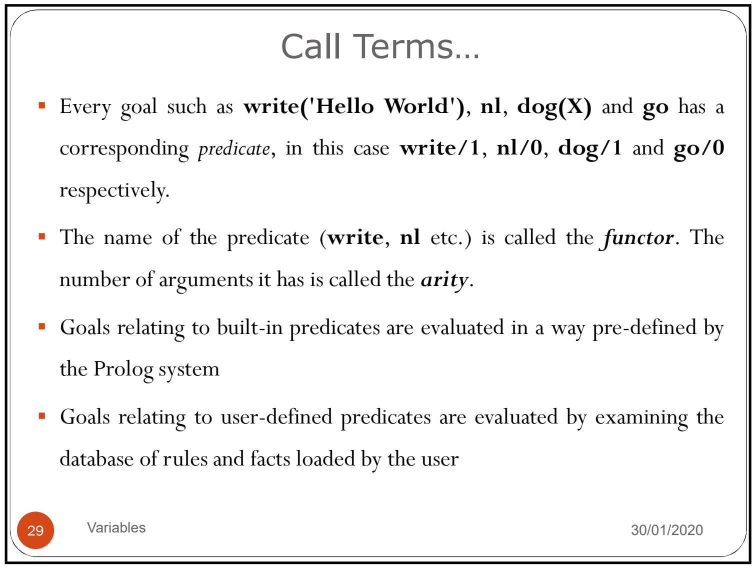
  no

  as there are no clauses with first argument *martin* where the second, third and fifth arguments are identical.



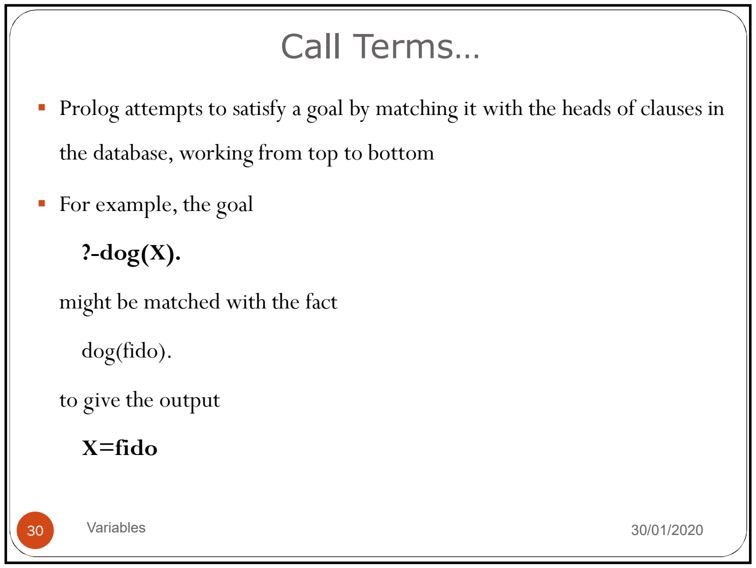
Call Terms

- Every goal must be a Prolog term but not any kind of term. It may only be an atom or a compound term, not a number, variable, list or any other type of term provided by some particular implementation of Prolog
- This restricted type of term is called a ***call term***.
- Heads of clauses and goals in the bodies of rules must also be call terms.
- The need for all three to take the same (restricted) form is essential for what follows.



Call Terms…

- Every goal such as **write('Hello World')**, **nl**, **dog(X)** and **go** has a corresponding *predicate*, in this case **write/1**, **nl/0**, **dog/1** and **go/0** respectively.
- The name of the predicate (**write**, **nl** etc.) is called the ***functor***. The number of arguments it has is called the ***arity***.
- Goals relating to built-in predicates are evaluated in a way pre-defined by the Prolog system
- Goals relating to user-defined predicates are evaluated by examining the database of rules and facts loaded by the user



Call Terms…

- Prolog attempts to satisfy a goal by matching it with the heads of clauses in the database, working from top to bottom
- For example, the goal

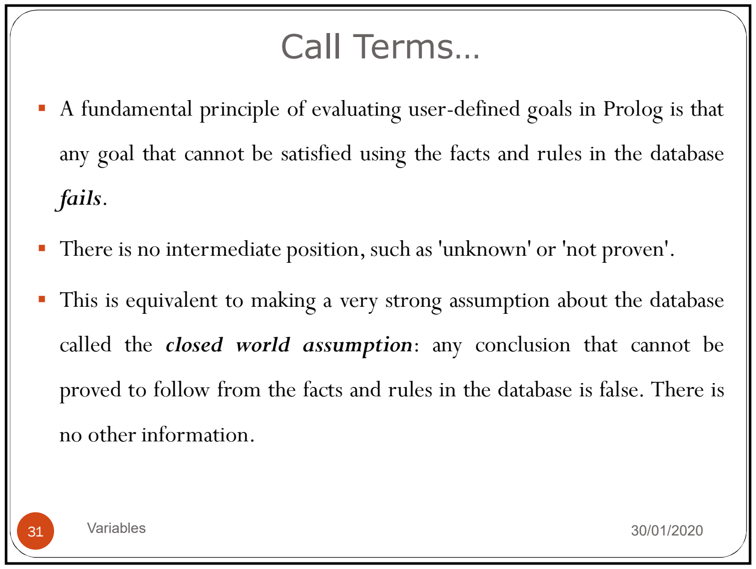
  ?-dog(X).

  might be matched with the fact

  dog(fido).

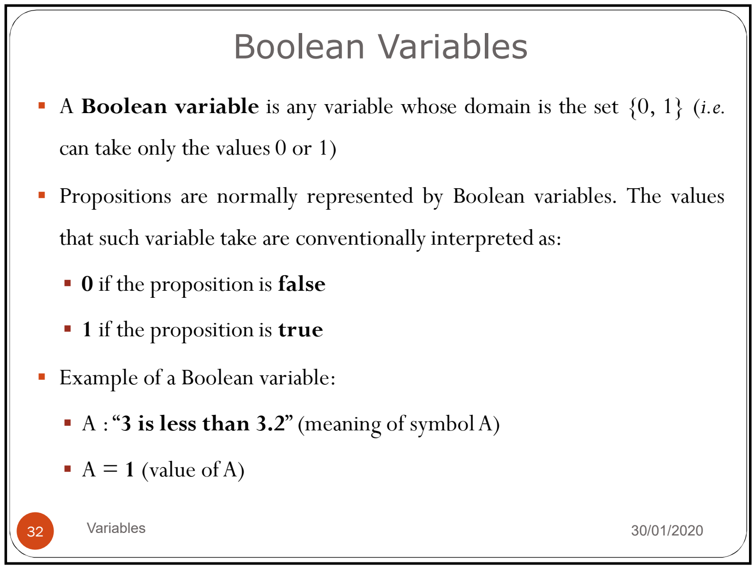
  to give the output

  X=fido



Call Terms…

- A fundamental principle of evaluating user-defined goals in Prolog is that any goal that cannot be satisfied using the facts and rules in the database *fails*.
- There is no intermediate position, such as 'unknown' or 'not proven'.
- This is equivalent to making a very strong assumption about the database called the ***closed world assumption***: any conclusion that cannot be proved to follow from the facts and rules in the database is false. There is no other information.

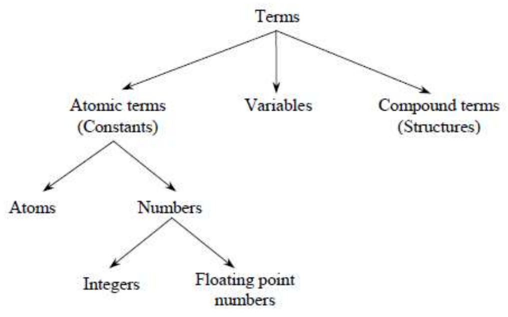


Boolean Variables

- A **Boolean variable** is any variable whose domain is the set {0, 1} (*i.e.* can take only the values 0 or 1)
- Propositions are normally represented by Boolean variables. The values that such variable take are conventionally interpreted as:
  - **0** if the proposition is **false**
  - **1** if the proposition is **true**
- Example of a Boolean variable:
  - A : "**3 is less than 3.2**" (meaning of symbol A)
  - A = **1** (value of A)

Data Types in Prolog

- Every data object in Prolog is a term. Terms divide into atomic terms, variables, and compound terms



Data Types in Prolog…

- **Atoms** are sequences of letters, numbers, and/or underscores beginning with a lowercase letter, as ulysses, iSLanD3, king_of_Ithaca.
  - Some single symbols, called solo characters, are atoms: ! ;
  - Sequences consisting entirely of some specific symbols or graphic characters are atoms: + - * / ^ < = > ˜ : . ? @ # $ & \ ‘
  - Any sequence of characters enclosed between single quotes is also an atom, as ’king of Ithaca’. A quote within a quoted atom must be double quoted: ’I”m’

Data Types in Prolog…

- Numbers are either decimal integers, as -19, 1960, octal integers when preceded by 0o, as 0o56, hexadecimal integers when preceded by 0x, as 0xF4, or binary integers when preceded by 0b, as 0b101
- Floating-point numbers are digits with a decimal point, as 3.14, -1.5. They may contain an exponent, as 23E-5 (23 10−5) or -2.3e5 (2.3 10−5)
- The ASCII numeric value of a character x is denoted 0’x, as 0’a (97), 0’b (98), etc

Data Types in Prolog…

- Variables are sequences of letters, numbers, and/or underscores beginning with an uppercase letter or the underscore character.
- Compound terms consist of a **functor**, which must be an atom, followed immediately by an opening parenthesis, a sequence of terms separated by commas, and a closing parenthesis.

Data Types in Prolog…

- Prolog uses two types of comments:
- Line comments go from the **"%"** symbol to the end of the line:
  - *% This is a comment*
- Multiline comments begin with a "/*" and end with a "*/":

```
/*
this
is
a comment */
```

Exercise 1

- Try to predict what Prolog will output in response to each of the following goals, and then try them.

i. **?-write(hello).**

ii. **?-write(Hello).**

iii. **?-write('Hello!').**

iv. **?-write('Hello!'),nl.**

v. **?-100=100.**

vi. **?-100=1000/10.**

Exercise 1

vii. **?-100 is 1000/10.**

viii. **?-1000 is 100*10.**

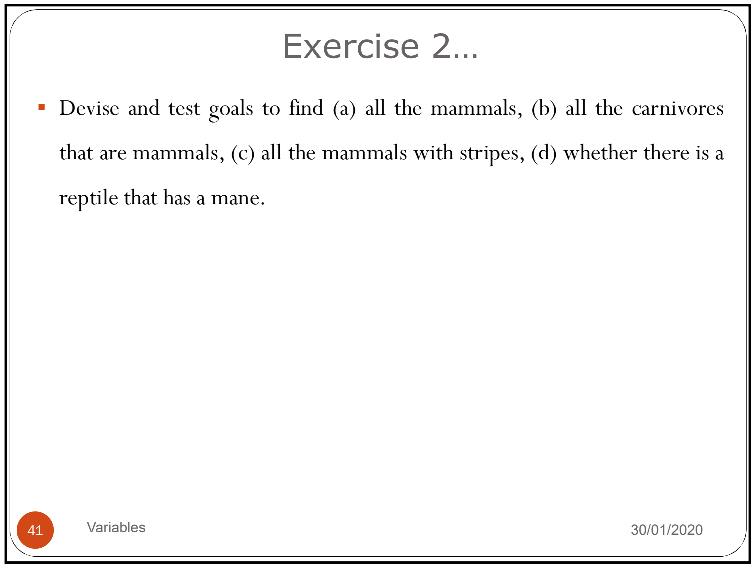
ix. **?-2 is (5+7)/6.**

x. **?-74 is (5+7)*6.**

Exercise 2

- Type the following program into a file and load it into Prolog.

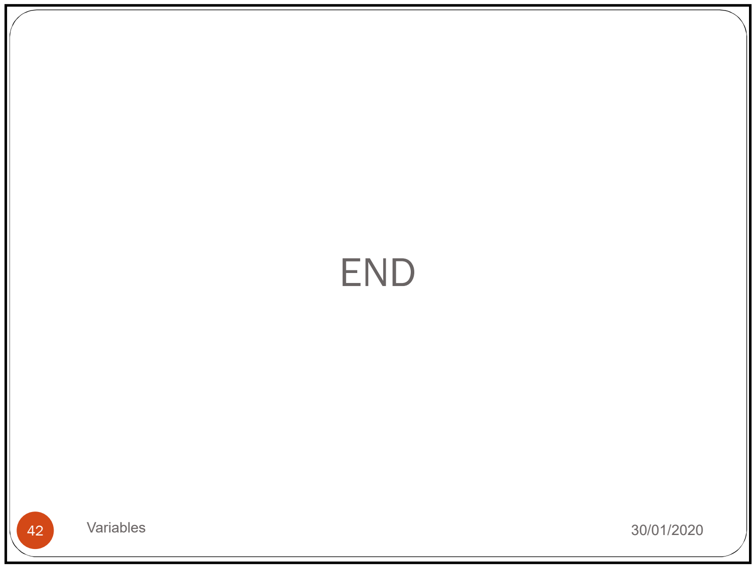
```
/* Animals Database */
animal(mammal,tiger,carnivore,stripes).
animal(mammal,hyena,carnivore,ugly).
animal(mammal,lion,carnivore,mane).
animal(mammal,zebra,herbivore,stripes).
animal(bird,eagle,carnivore,large).
animal(bird,sparrow,scavenger,small).
animal(reptile,snake,carnivore,long).
animal(reptile,lizard,scavenger,small).
```



Exercise 2…

- Devise and test goals to find (a) all the mammals, (b) all the carnivores that are mammals, (c) all the mammals with stripes, (d) whether there is a reptile that has a mane.

41 Variables 30/01/2020



END

42 Variables 30/01/2020