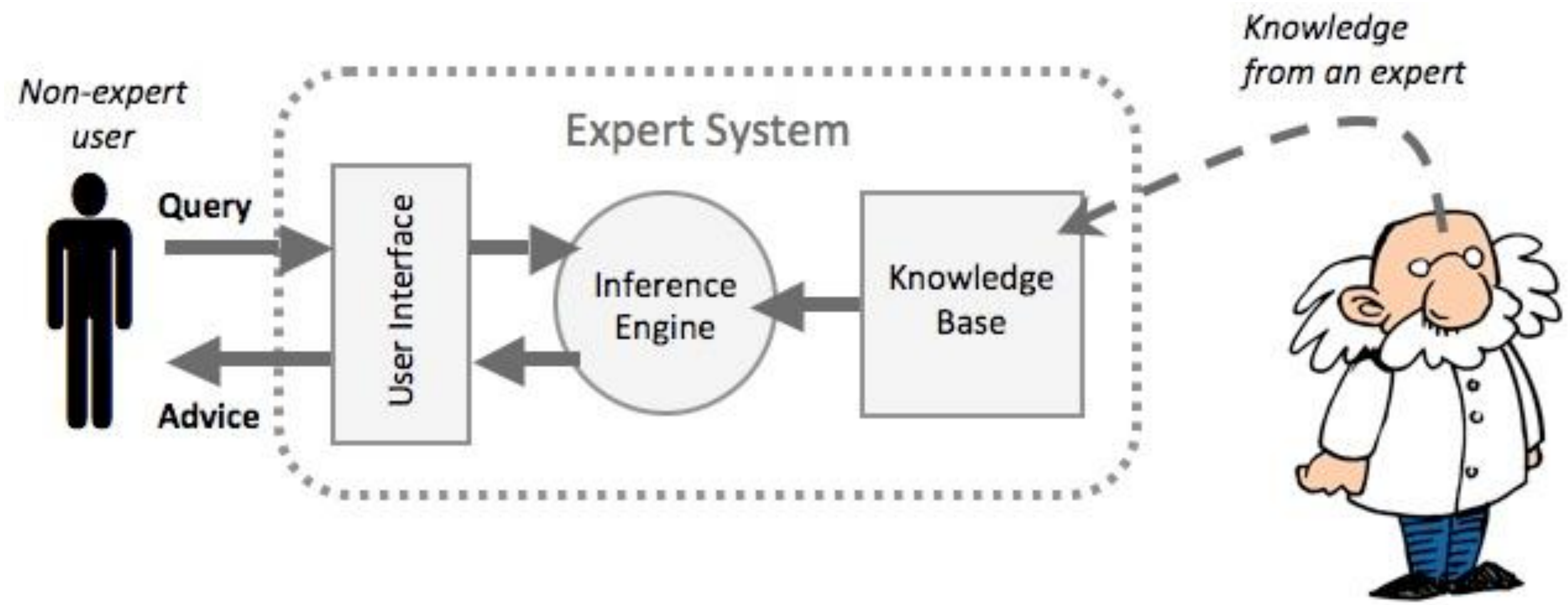


Expert systems

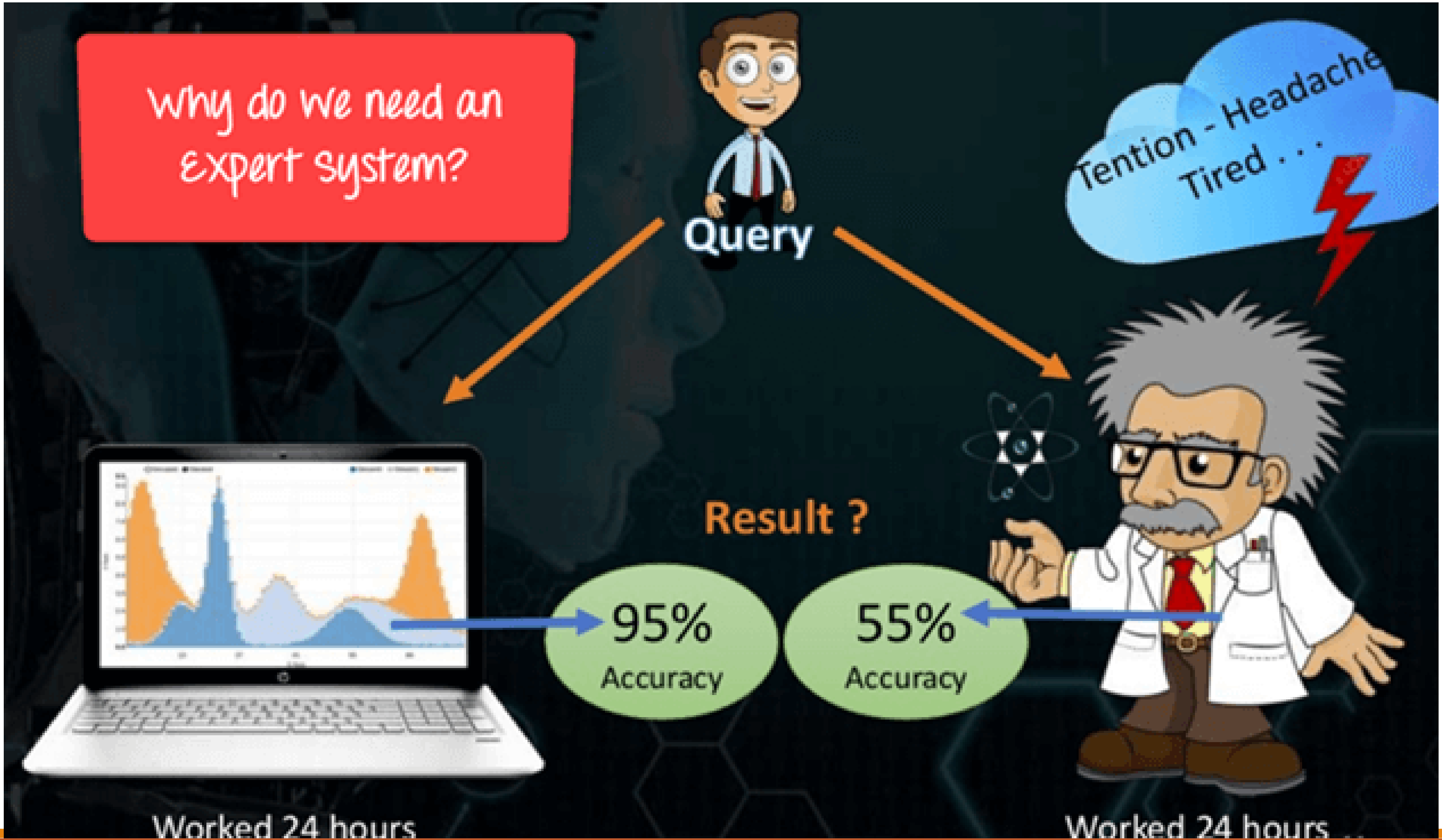
CREATE A SIMPLE EXPERT SYSTEM

LESSON OBJECTIVES

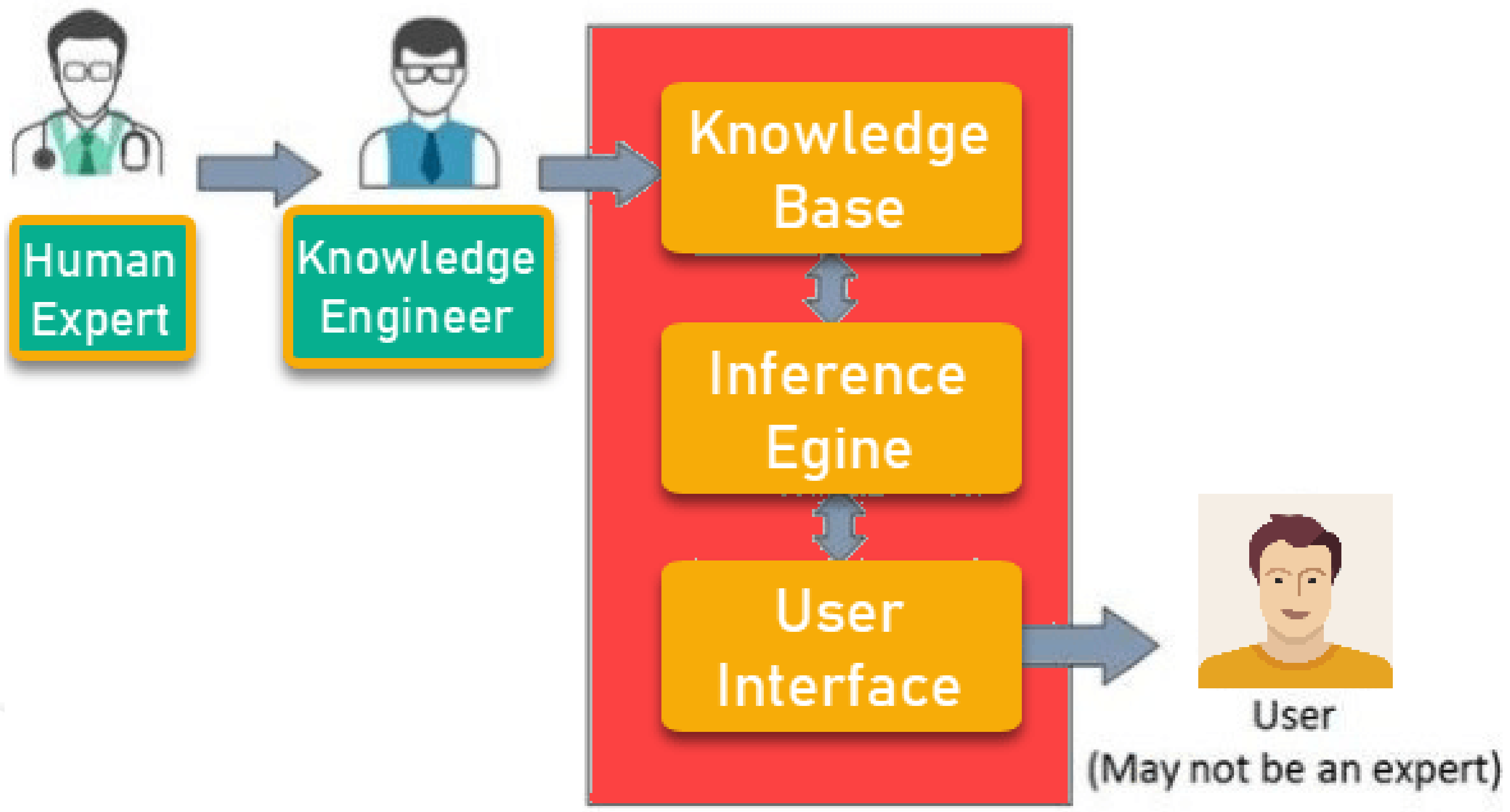
- Define expert system
- Identify key features and components of an expert system
- Advantages and limitations of expert systems
- Applications of expert systems

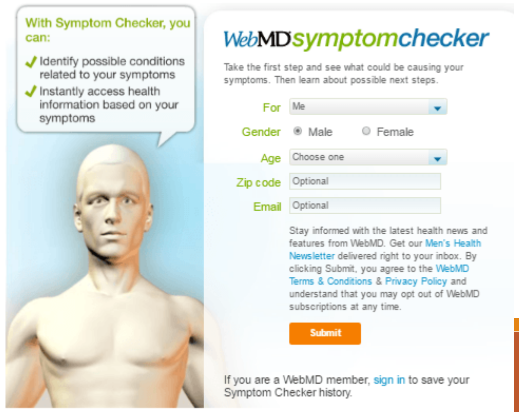
EXPERT SYSTEM

- Defined a computer system that mimics the **decision-making** ability of a human;
- Expert systems simulate the judgement and behavior of a human or organization that has expert knowledge and experience.


Why do we need an Expert system?
Query
Tention - Headache
Tired . . .
Result ?
95% Accuracy
55% Accuracy
Worked 24 hours
Worked 24 hours . .

Components of the expert system




With Symptom Checker, you can:

- ✓ Identify possible conditions related to your symptoms
- ✓ Instantly access health information based on your symptoms

WebMD symptomchecker

Take the first step and see what could be causing your symptoms. Then learn about possible next steps.

For	Me
Gender	[✓ Male] Female
Age	Choose one
Zip code	Optional
Email	Optional

Stay informed with the latest health news and features from WebMD. Get our Men's Health Newsletter delivered right to your inbox. By clicking Submit, you agree to the WebMD Terms & Conditions & Privacy Policy and understand that you may opt out of WebMD subscriptions at any time.

If you are a WebMD member, sign in to save your Symptom Checker history.

User Interface

The user interface is the most crucial part of the expert system. This component takes the user's query in a readable form and passes it to the inference engine. After that, it displays the results to the user. In other words, *it's an interface that helps the user communicate with the expert system.*

Inference Engine

The inference engine is the brain of the expert system. Inference engine contains *rules* to solve a specific problem. *It refers the knowledge from the Knowledge Base*.

It selects facts and rules to apply when trying to answer the user's query. It provides reasoning about the information in the knowledge base. It also helps in deducting the problem to find the solution. This component is also helpful for formulating conclusions.

Knowledge Base

The knowledge base is a repository of facts. It stores all the knowledge about the problem domain. It is like a large container of knowledge which is obtained from different experts of a specific field.

Thus *we can say that the success of the Expert System mainly depends on the highly accurate and precise knowledge.*

Object	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6
dog	mammal	can be a pet	lives on land	makes bark sounds	body is covered in fur	walks on 4 legs
whale	mammal	not a pet	lives in water	makes sonic sound	body covered in skin	swims; no legs
duck	bird	not a pet	lives in water	makes quack sounds	body covered in feathers	swims; has two legs

- so if we had a series of questions:

– is it a mammal?	YES
– can it be a pet?	NO
– does it live in water?	YES
– does it make sonic sounds?	YES
– is its body covered in skin?	YES
– does it have any legs?	NO

conclusion: it is a WHALE.

Example use of an expert system (medical diagnosis)

Input screen

- First of all an interactive screen is presented to the user
- The system asks a series of questions about the patient's illness
- The user answers the questions asked (either as multiple choice or YES/NO questions)
- A series of questions are asked based on the user's responses to previous questions

Expert system

- The inference engine compares the symptoms entered with those in the knowledge base looking for matches
- The rules base (inference rules) is used in the matching process
- Once a match is found, the system suggests the probability of the patient's illness being identified accurately
- The expert system also suggests possible solutions and remedies to cure the patient or recommendations on what to do next
- The explanation system will give reasons for its diagnosis so that the user can determine the validity of the diagnosis or suggested treatment

Output screen

- The diagnosis can be in the form of text or it may show images of the human anatomy to indicate where the problem may be
- The user can request further information from the expert system to narrow down even further the possible illness and its treatment

APPLICATIONS OF EXPERT SYSTEMS

» oil and mineral prospecting

» diagnosis of a patient’s illness

» fault diagnostics in mechanical and electronic equipment

» tax and financial calculations

» strategy games, such as chess

» logistics (efficient routing of parcel deliveries)

» identification of plants, animals and chemical/biological compounds.

GROUP RESEARCH AND CLASS DISCUSSION:
What are the benefits and limitations of expert systems

BENEFITS OF EXPERT SYSTEMS

» they offer a high level of **expertise**

» they offer high **accuracy**

» the results are **consistent**

» they can store **vast amounts** of ideas and facts

» they can make **traceable logical solutions and diagnostics**

» it is possible for an expert system to have **multiple expertise**

» they have very **fast response times** (much quicker than a human expert)

» they provide **unbiased** reporting and analysis of the facts

» they indicate the **probability** of any suggested solution being correct.

LIMITATIONS OF EXPERT SYSTEMS

» users of the expert system need **considerable training** in its use to ensure the

system is being used correctly

» the set up and maintenance **costs** are very high

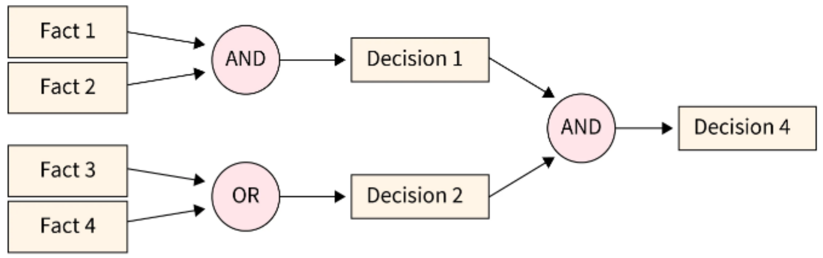
» they tend to give very ‘**cold**’ responses that may not be appropriate in certain medical situations

» they are only as good as the information/facts entered into the system - **GIGO**

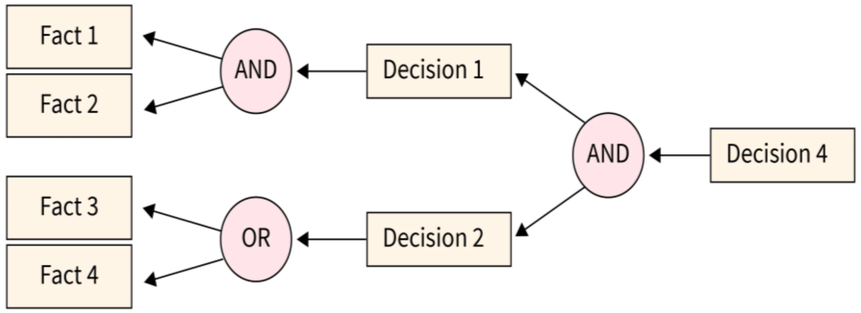
» users sometimes make the very dangerous assumption that they are **infallible**.

Forward chaining vs Backward chaining

- **Forward chaining** is a reasoning or inference strategy used in expert systems and artificial intelligence, where the system starts with available facts and uses rules to derive new conclusions. It begins with known data and incrementally applies rules to reach a goal or decision.



- **Backward chaining** starts with a goal or desired outcome and works backward through a set of rules and facts to determine if the goal can be satisfied.



EXAMPLES:

Forward Chaining Example (MYCIN) _In MYCIN, a medical expert system, forward chaining starts with **patient data** (e.g., symptoms like fever, blood test results) and **applies rules** to infer a diagnosis. For instance, if Rule 1 says "IF fever AND positive blood culture, THEN suspect bacterial infection," and Rule 2 says "IF bacterial infection AND gram-negative bacteria, THEN consider E. coli," the system chains forward from symptoms to conclude E. coli as a possible cause, recommending antibiotics.

Backward Chaining Example (PROSPECTOR): In PROSPECTOR, a geological expert system, backward chaining starts with a **goal**, like "Is there a copper deposit?" It **checks if conditions** for a copper deposit (e.g., specific rock types, magnetic anomalies) are met by working backward through rules. If Rule 1 says "IF volcanic rock AND high magnetic reading, THEN copper deposit likely," the system queries the user or database for these conditions, verifying or refuting the goal.

PROLOG PROGRAMMING

OBJECTIVES

- Define what does Prolog programming means
- Describe each of the basic contents of prolog: facts, rules and queries
- Write simple prolog programs using basic rules
- Define output from written Prolog program

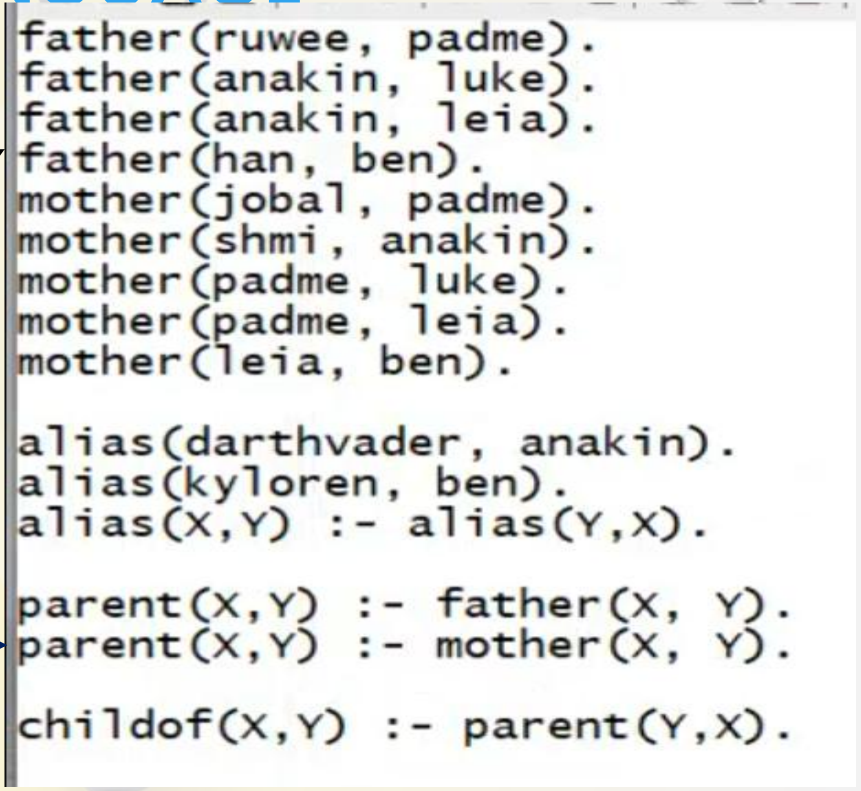
Facts about Prolog Programming (Logic)

- Prolog stands for **PROgramming** in **LOGic (Declarative language)**
- Prolog is a program consists of data based on the **facts and rules**
- Used in AI applications such as **natural language interfaces**, **automated reasoning systems** and **expert systems**.

ELEMENTS OF A PROLOG LANGUAGE

Prolog language basically has **three** different elements –

- **Facts** – Facts are the statements that state the objects or describe the relationship between objects.
- **Rules** – Rules are the conditional statements about objects and their relationships.
- **Queries** – Prolog queries are the questions asked by the user to prolog interpreter about facts and rules stored in it's database.

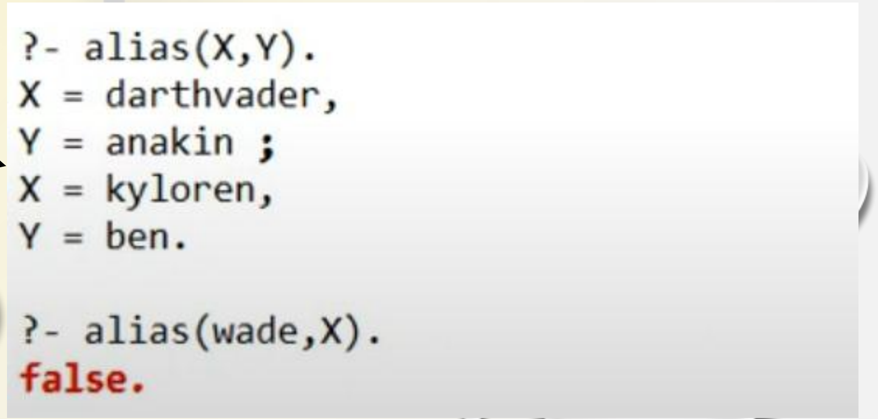


```
father(ruwee, padme).
father(anakin, luke).
father(anakin, leia).
father(han, ben).
mother(jobal, padme).
mother(shmi, anakin).
mother(padme, luke).
mother(padme, leia).
mother(leia, ben).

alias(darthvader, anakin).
alias(kyloren, ben).
alias(X,Y) :- alias(Y,X).

parent(X,Y) :- father(X, Y).
parent(X,Y) :- mother(X, Y).

childof(X,Y) :- parent(Y,X).
```



```
?- alias(X,Y).
X = darthvader,
Y = anakin ;
X = kyloren,
Y = ben.

?- alias(wade,X).
false.
```

ELEMENTS OF A PROLOG LANGUAGE: FACTS

Facts are property of an object.

We declare facts describing explicit relationship between objects and properties of the objects

Examples

• John's phone number 8705678976	phonenum(john, 8705678976).
• Paris is the capital of France	capital(paris, france).

Facts are also called **predicate** or **clause**

Facts should have

* Names of properties/relationships beginning with **lower case letters**.
* The **relationship** name appears as the first term.
* Objects are comma-separated arguments within parentheses.
* A period "." must end a fact.

EXAMPLES: FACTS

```
likes(john, susie).                        /* John likes Susie */
likes(X, susie).                            /* Everyone likes Susie */
likes(john, Y).                            /* John likes everybody */
likes(john, Y), likes(Y, john).            /* John likes everybody and everybody likes John */
likes(john, susie); likes(john,mary).      /* John likes Susie or John likes Mary */
not(likes(john,pizza)).                    /* John does not like pizza */
```

ELEMENTS OF A PROLOG LANGUAGE: RULES

Rules are relationship of objects.

It uses the following logical implication to describe a relationship among facts.

✓ **And**	**,**
✓ **If**	**:-**
✓ **Or**	**;**
✓ **Not**	**not**

- **Variables**: **X/Y** must begin with an upper-case letter.
- Predicate names, function names, and the names for objects must begin with a **lowercase letter**
- Constants: numbers are enclosed in single quotes

! prevents Prolog from checking other alternative rules.

\= means not equal

PROLOG LANGUAGE: RULES

Prologs Rules consists of two parts separated by "**:-**"

- The first part is similar to a fact (a predicate with arguments). (called the **left_hand_side**)
- The second part consists of other clauses (facts or rules which are separated by commas) which must all be true for the rule itself to be true. (called the **right_hand_side**)

A Prolog rule takes the form

left_hand_side :- right_hand_side .

This sentence is interpreted as: *left_hand_side* ***if*** *right_hand_side*.

The **left_hand_side** is restricted to a **single, positive, literal**, which means it must consist of a positive atomic expression. It cannot be negated, and it cannot contain logical connectives.

EXAMPLES RULES

Examples of valid rules:

```
friends(X,Y) :- likes(X,Y),likes(Y,X).                  /* X and Y are friends if they like each other */

hates(X,Y) :- not(likes(X,Y)).                          /* X hates Y if X does not like Y. */

enemies(X,Y) :- not(likes(X,Y)),not(likes(Y,X)).        /* X and Y are enemies if X does not like Y and if Y does
                                                                   not like X */
```

Examples of invalid rules:

```
left_of(X,Y) :- right_of(Y,X)              /* Missing a period */

likes(X,Y),likes(Y,X) :- friends(X,Y).     /* LHS is not a single literal */

not(likes(X,Y)) :- hates(X,Y).             /* LHS cannot be negated */
```

ELEMENTS OF A PROLOG LANGUAGE: QUERIES

Queries

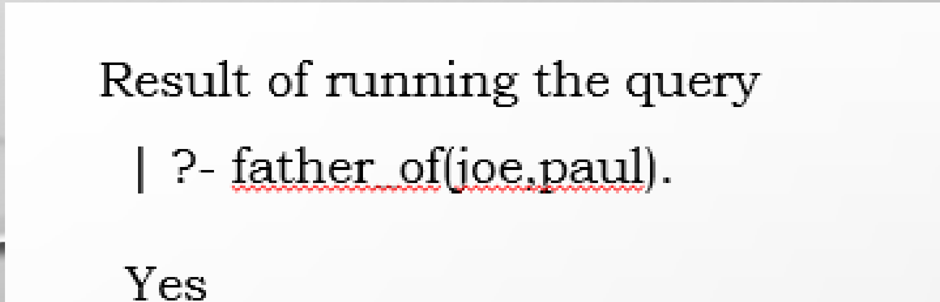
A query is a statement starting with a predicate and followed by its arguments, some of which are variables.

The **database** is assumed to represent **what is true** about a particular problem domain.

In making a query you are asking prolog whether it can prove that your query is true. If so, it answers "yes" and displays any **variable bindings** that it made in coming up with the answer.

If it fails to prove the query true, it answers "no".

Whenever you run the prolog interpreter, it will **prompt** you with **?-**.



Result of running the query

```
| ?- father_of(joe,paul).

Yes
```

QUERIES EXAMPLES

This is a database consists of the following facts about a fictitious family

```
father_of(joe,paul).
father_of(joe,mary).
mother_of(jane,paul).
mother_of(jane,mary).
male(paul).
male(joe).
female(mary).
female(jane).
```

Result of running the query

```
| ?- father_of(joe,paul).

Yes

| ?- father_of(paul,mary).

No

| ?- father_of(X,mary).

X = joe

yes
```

Activity

Suppose we are working with the following knowledge base:

```
wizard(ron).
hasWand(harry).
quidditchPlayer(harry).
wizard(X):-  hasBroom(X),  hasWand(X).
hasBroom(X):-  quidditchPlayer(X).
```

How does Prolog respond to the following queries?

1. wizard(ron).
2. witch(ron).
3. wizard(hermione).
4. witch(hermione).
5. wizard(harry).
6. wizard(Y).
7. witch(Y).

Activity

Suppose we are working with the following knowledge base:

```
wizard(ron).
hasWand(harry).
quidditchPlayer(harry).
wizard(X):-  hasBroom(X),  hasWand(X).
hasBroom(X):-  quidditchPlayer(X).
```

How does Prolog respond to the following queries?

1.wizard(ron). **Yes**

2.witch(ron). **No**

3.wizard(hermione). **No**

4.witch(hermione). **No**

5.wizard(harry). **Yes**

6.wizard(Y). **harry**

7.witch(Y). **No**

Open the following link of a prolog code in the swi-prolog
and complete the online exercise

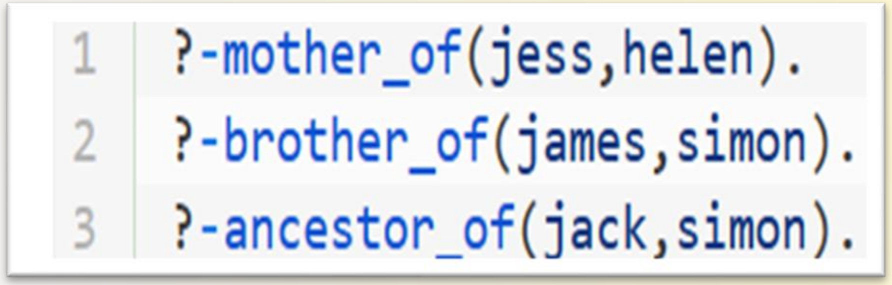
https://swish.swi-prolog.org/p/prolog-family-tree.pl

QUERIES

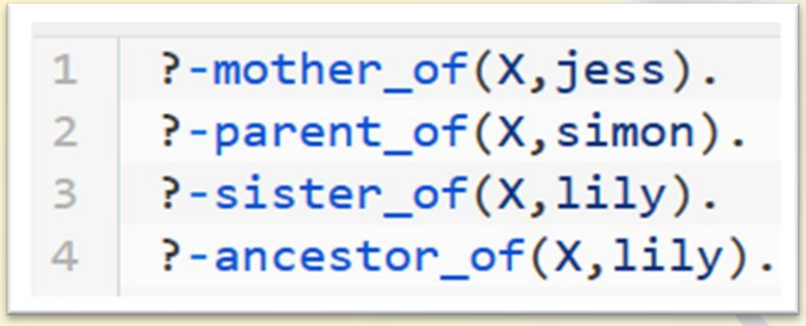
Using the online data, you accessed before complete the following queries

True or False Queries:

The following queries would return either True or False.



```
?-mother_of(jess,helen).
?-brother_of(james,simon).
?-ancestor_of(jack,simon).
```



```
?-mother_of(X,jess).
?-parent_of(X,simon).
?-sister_of(X,lily).
?-ancestor_of(X,lily).
```

Using the online data, you accessed before complete the following template to complete the following family tree