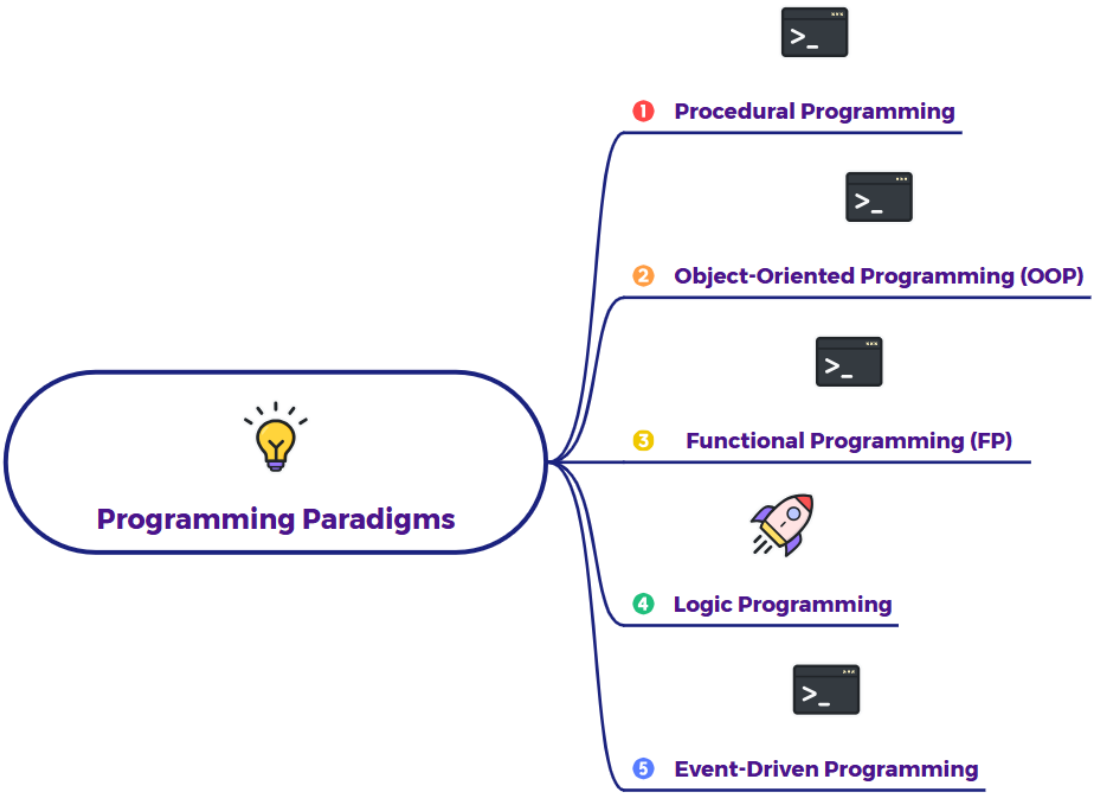
Explain Logical programming with an example



The "Logic Family Tree" Way:

Imagine you have a big family, and you want a system to figure out relationships. In Logic Programming, you don't tell the computer *how* to find a grandparent; you tell it **the basic truths (facts) and the rules that define relationships**, and then you just **ask questions**. The computer then uses its "logic engine" to deduce the answers.

1. Stating the Facts (What We Know is True):

First, you tell the computer some simple, undeniable truths about your family. These are like single pieces of information.

- parent(john, mary). (Meaning: John is a parent of Mary.)
- parent(mary, peter). (Meaning: Mary is a parent of Peter.)
- parent(susan, mary). (Meaning: Susan is a parent of Mary.)
- parent(david, lisa). (Meaning: David is a parent of Lisa.)

- parent(lisa, sarah). (Meaning: Lisa is a parent of Sarah.)
- male(john).
- female(mary).

2. Stating the Rules (How Relationships Are Defined):

Next, you give the computer some general rules about how relationships work. These rules allow the computer to figure out more complex relationships from the basic facts.

- **Rule for grandparent:**
    - "X is a grandparent of Z IF X is a parent of Y AND Y is a parent of Z."
    - (grandparent(X, Z) :- parent(X, Y), parent(Y, Z).)
- **Rule for father:**
    - "X is the father of Y IF X is a parent of Y AND X is male."
    - (father(X, Y) :- parent(X, Y), male(X).)

3. Asking Questions (Querying the System):

Now, with your facts and rules loaded, you can ask the computer questions:

- **Question 1:** "Is John a parent of Mary?"
    - (?- parent(john, mary).)
    - **Computer's Answer:** "Yes." (It directly finds this as a fact.)
- **Question 2:** "Is Peter a parent of John?"
    - (?- parent(peter, john).)
    - **Computer's Answer:** "No." (It searches all facts and finds no such truth.)
- **Question 3:** "Who is a grandparent of Peter?"
    - (?- grandparent(X, peter).)
    - **Computer's Thinking (simplified):**

        - "To be a grandparent of Peter, I need someone (Y) who is a parent of Peter, and someone (X) who is a parent of Y."
        - "Okay, looking at facts: Mary is a parent of Peter (so Y could be Mary)."
        - "Now, who is a parent of Mary?"
        - "Facts say: John is a parent of Mary (so X could be John)."
        - "Facts say: Susan is a parent of Mary (so X could be Susan)."
    - **Computer's Answer:** "X = John; X = Susan."
- **Question 4:** "Who is the father of Mary?"
    - (?- father(X, mary).)
    - **Computer's Thinking:**
        - "To be a father of Mary, X must be a parent of Mary AND X must be male."
        - "Facts: John is a parent of Mary. Is John male? Yes, male(john) is a fact."
        - "Facts: Susan is a parent of Mary. Is Susan male? No, male(susan) is not a fact (and female(mary) doesn't imply male(susan))."
    - **Computer's Answer:** "X = John."

Key Points of Logic Programming in this Example:

- **Declarative:** You **declare** what is true (facts) and what rules apply (rules). You *don't* specify a step-by-step algorithm on how to find a grandparent. You just define *what* a grandparent is.
- **Inference Engine:** The computer has a built-in "logic engine" that uses the facts and rules to deduce answers to your questions. It figures out the "how" for you.
- **Symbolic Reasoning:** It's very good at dealing with relationships and symbols (john, mary, parent) rather than just numbers or explicit sequences.

Why is this "Logic" way useful?

- **Knowledge Representation:** Excellent for building "expert systems" or "knowledge bases" where you define a lot of truths and rules (like medical diagnostic systems, legal reasoning systems).
- **Problem Solving:** Good for problems where the solution isn't a direct calculation but requires deducing facts from a set of constraints and rules (like puzzles, scheduling).
- **Natural Language:** Concepts are often closer to how humans think about relationships.

Logic Programming is like giving a computer a set of true statements (facts) and a set of rules (how truths relate to each other), and then asking it to figure out if certain other statements are true or what conclusions it can draw, all without telling it the exact steps to follow.