London Underground example
connected(bond_street, oxford_circus, central).
connected(oxford_circus, tottenham_court_road, central).
connected(bond_street, green_park, jubilee).
connected(green_park, charing_cross, jubilee).
connected(green_park, piccadilly_circus, piccadilly).
connected(piccadilly_circus, leicester_square, piccadilly).
connected(green_park, oxford_circus, victoria).
connected(oxford_circus, piccadilly_circus, bakerloo).
connected(piccadilly_circus, charing_cross, bakerloo).
connected(tottenham_court_road, leicester_square, northern).
connected(leicester_square, charing_cross, northern).
Two stations are nearby if they are on the same line with at most one other station in between:

\[
\begin{align*}
\text{nearby}(\text{bond\_street}, \text{oxford\_circus}). \\
\text{nearby}(\text{oxford\_circus}, \text{tottenham\_court\_road}). \\
\text{nearby}(\text{bond\_street}, \text{tottenham\_court\_road}). \\
\text{nearby}(\text{bond\_street}, \text{green\_park}). \\
\text{nearby}(\text{green\_park}, \text{charing\_cross}). \\
\text{nearby}(\text{bond\_street}, \text{charing\_cross}). \\
\text{nearby}(\text{green\_park}, \text{piccadilly\_circus}). \\
\end{align*}
\]

or better

\[
\begin{align*}
\text{nearby}(X, Y) :&= \text{connected}(X, Y, L). \\
\text{nearby}(X, Y) :&= \text{connected}(X, Z, L), \text{connected}(Z, Y, L). \\
\end{align*}
\]
Compare

\[
\text{nearby}(X, Y) :\neg\text{connected}(X, Y, L).
\text{nearby}(X, Y) :\neg\text{connected}(X, Z, L), \text{connected}(Z, Y, L).
\]

with

\[
\text{not\_too\_far}(X, Y) :\neg\text{connected}(X, Y, L).
\text{not\_too\_far}(X, Y) :\neg\text{connected}(X, Z, L1), \text{connected}(Z, Y, L2).
\]

This can be rewritten with don’t cares:

\[
\text{not\_too\_far}(X, Y) :\neg\text{connected}(X, Y, \_).
\text{not\_too\_far}(X, Y) :\neg\text{connected}(X, Z, \_), \text{connected}(Z, Y, \_).
\]

Exercise 1.1
?-nearby(tottenham_court_road, W)

nearby(X1, Y1) :- connected(X1, Y1, L1)

{ X1 -> tottenham_court_road, Y1 -> W }  

?-connected(tottenham_court_road, W, L1)   

connected(tottenham_court_road, leicester_square, northern)  

{ W -> leicester_square, L1 -> northern }  

Proof tree
Exercise 1.2
A station is reachable from another if they are on the same line, or with one, two, … changes:

\[
\text{reachable}(X, Y) : \neg \text{connected}(X, Y, L).
\]
\[
\text{reachable}(X, Y) : \neg \text{connected}(X, Z, L_1), \text{connected}(Z, Y, L_2).
\]
\[
\text{reachable}(X, Y) : \neg \text{connected}(X, Z_1, L_1), \text{connected}(Z_1, Z_2, L_2), \text{connected}(Z_2, Y, L_3).
\]

…

or better

\[
\text{reachable}(X, Y) : \neg \text{connected}(X, Y, L).
\]
\[
\text{reachable}(X, Y) : \neg \text{connected}(X, Z, L), \text{reachable}(Z, Y).
\]
Recursion (2)
reachable(X, Y, noroute):=connected(X, Y, L).
reachable(X, Y, route(Z, R)):=connected(X, Z, L),
                    reachable(Z, Y, R).

?-reachable(oxford_circus, charing_cross, R).
R = route(tottenham_court_road, route(leicester_square, noroute));
R = route(picadilly_circus, noroute);
R = route(picadilly_circus, route(leicester_square, noroute))

Structured terms
reachable(X,Y,[]) :- connected(X,Y,L).
reachable(X,Y,[Z|R]) :- connected(X,Z,L),
               reachable(Z,Y,R).

?-reachable(oxford_circus,charing_cross,R).
R = [tottenham_court_road,leicester_square];
R = [piccadilly_circus];
R = [picadilly_circus,leicester_square]
This list can be written in many ways:

\[
\text{(a, (b, (c, []))})
\]

\[
[a | [b | [c | []]]]\\
\]

\[
[a | [b | [c]]]
\]

\[
[a | [b, c]]
\]

\[
[a, b, c]
\]

\[
[a, b | [c]]
\]

...
Lists of arbitrary length:

\[
\begin{align*}
\text{list}([]) & \end{align*}
\]

\[
\begin{align*}
\text{list}([\text{First} | \text{Rest}]) & : \neg \text{list}(\text{Rest}). \\
\end{align*}
\]

Lists of even length:

\[
\begin{align*}
\text{evenlist}([]) & \end{align*}
\]

\[
\begin{align*}
\text{evenlist}([\text{First}, \text{Second} | \text{Rest}]) & : \neg \text{evenlist}(\text{Rest}). \\
\end{align*}
\]

Lists of odd length:

\[
\begin{align*}
\text{oddlist}([\text{One}]) & \end{align*}
\]

\[
\begin{align*}
\text{oddlist}([\text{First}, \text{Second} | \text{Rest}]) & : \neg \text{oddlist}(\text{Rest}). \\
\end{align*}
\]

or alternatively:

\[
\begin{align*}
\text{oddList}([\text{First} | \text{Rest}]) & : \neg \text{evenlist}(\text{Rest}). \\
\end{align*}
\]
Prolog has very simple syntax

- constants, variables, and structured terms refer to objects
  - variables start with uppercase character
  - functors are never evaluated, but are used for naming

- predicates express relations between objects

- clauses express true statements
  - each clause independent of other clauses

Queries are answered by matching with head of clause

- there may be more than one matching clause
  - query answering is search process

- query may have 0, 1, or several answers

- no pre-determined input/output pattern (usually)