# Functional and **Logic programming**

## Tutorial 1: Prolog as a database, simple recursion

# Why to learn logic programming?

- LP is great for prototyping algorithms and/or search strategies.
- The ideas from "declarative" languages appear more and more in traditional languages.
  - Semantic web
  - LINQ in C#
- Most importantly: **If you learn different ways to "think about problems", you become a better programmer.**
- You receive credits for it!

# Why not to learn logic programming?

- If you have ∞ of time and lust, use C for performance.
- Input/Output is cumbersome in Prolog...
  In fact it really sucks.

# First steps in Prolog 1/3

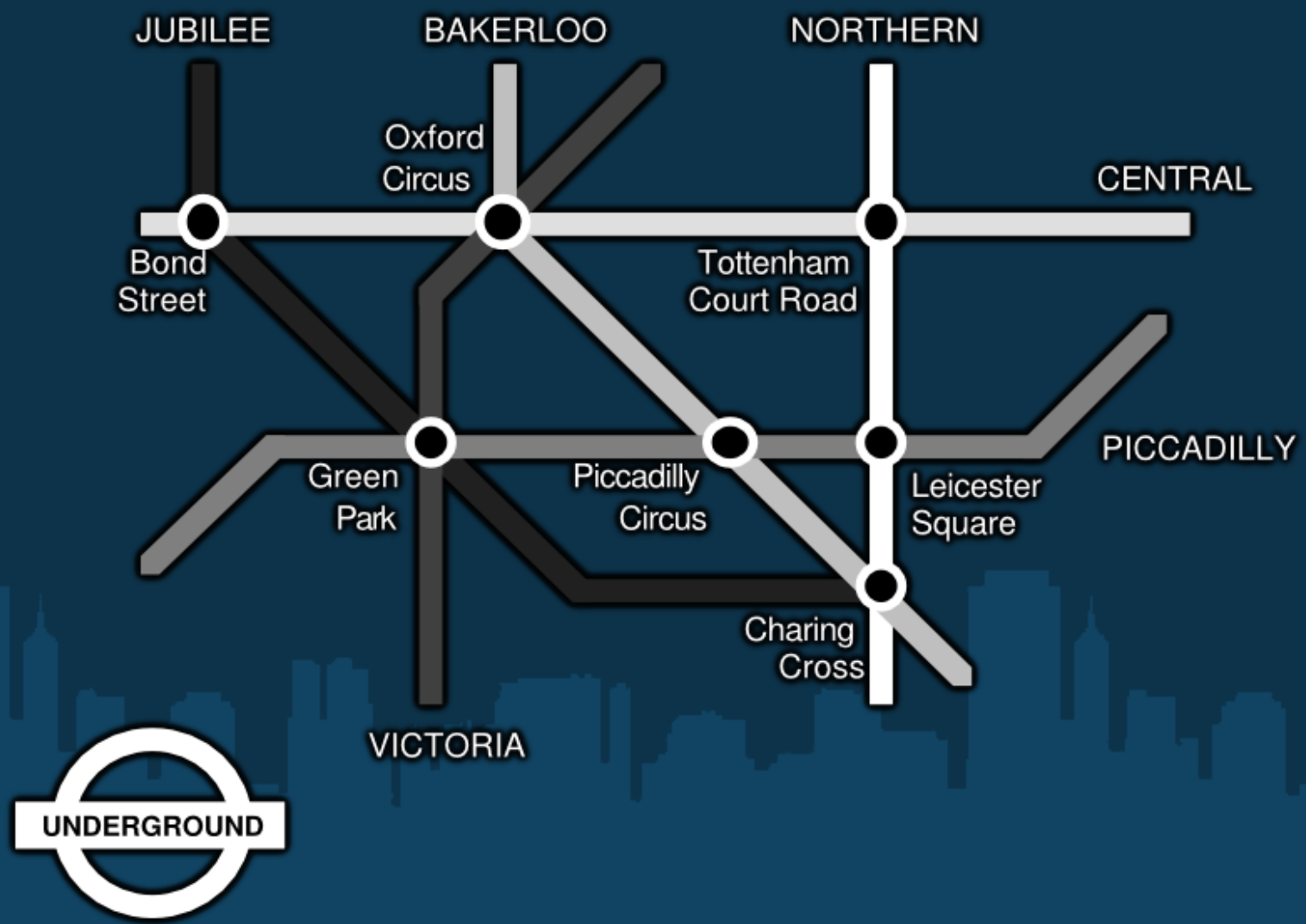
1. Install SWI-Prolog.
    1. If Ubuntu, run:
       **$ sudo apt-get install swi-prolog-x**
    2. If Windows, install Ubuntu and continue from step 1.1.
2. Do never ever use imperative code any more!
3. Download **tut1.pl** from **pastebin.com/u/radek**

# First steps in Prolog 2/3

```
$ swipl

?- consult(tut1).
% tut1 compiled 0.00 sec, 1,736 bytes
true.

?- connected(X,Y,Z).
X = bond_street,
Y = oxford_circus,
Z = central ; % Now keep pressing n or Enter.

?- connected(green_park,X,_).
X = piccadilly_circus ;
X = oxford_circus.
```

# First steps in Prolog 3/3

- Open **tut1.pl** and study the code!
- Find a missing connection and correct it!

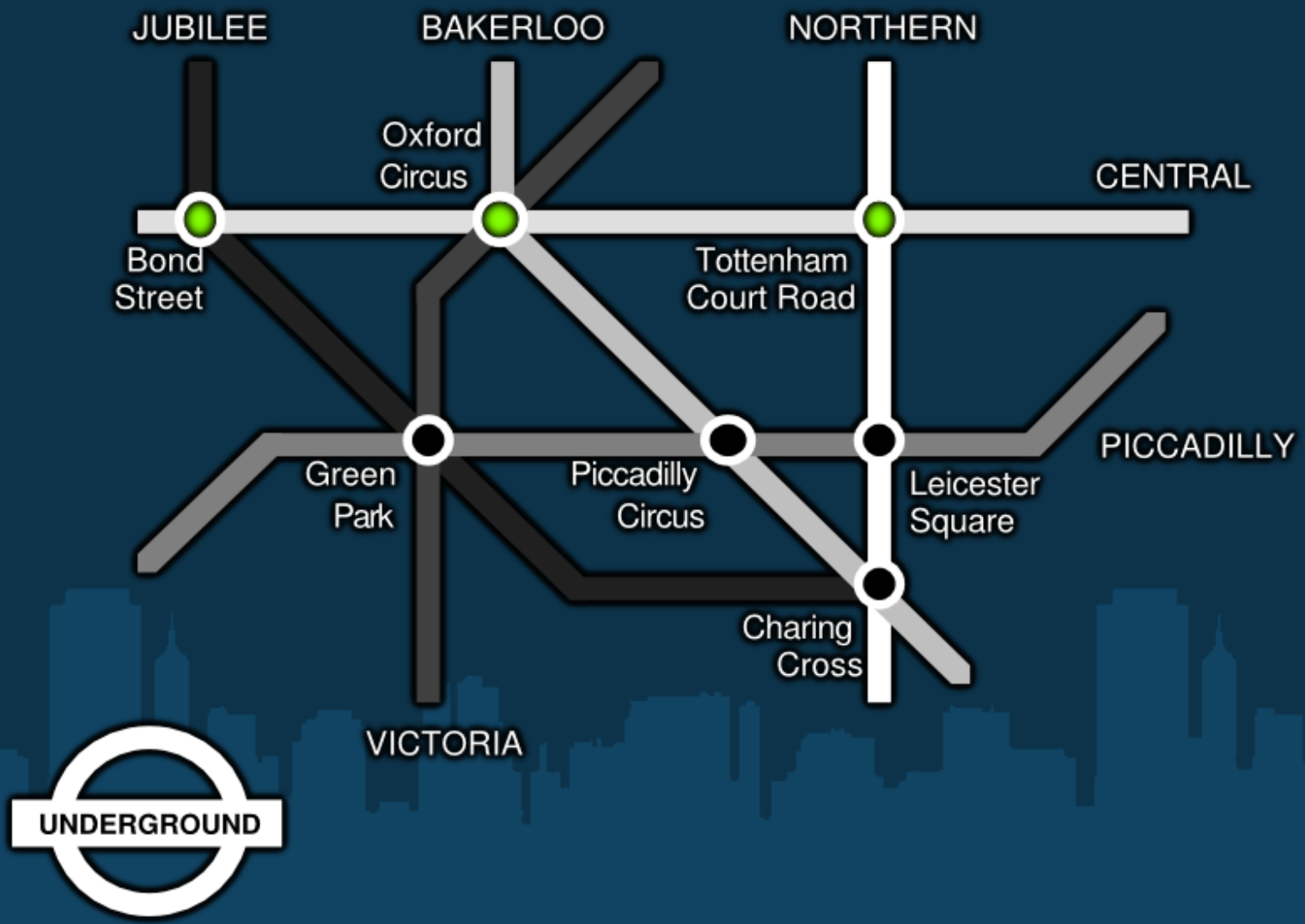


# What to do next?

- Does it remind you of SQL?
  - INSERT is made by compiling the **.pl** file.
  - SELECT si done by querying **connected(St1,St2,Line).**
- Prolog can do more!
- Implement a VIEW:
  Stations **X** and **Y** are **nearby(X,Y)** if they are share the same line and there is at most 1 station inbetween.

# The **nearby** predicate

Stations **X** and **Y** are **nearby(X,Y)** if they are share the same line and there is at most 1 station inbetween.



# The **nearby** predicate

**?- reachable(X,Y).**
**X = bond_street,**
**Y = oxford_circus ;**
**X = oxford_circus,**
**Y = tottenham_court_road ;**

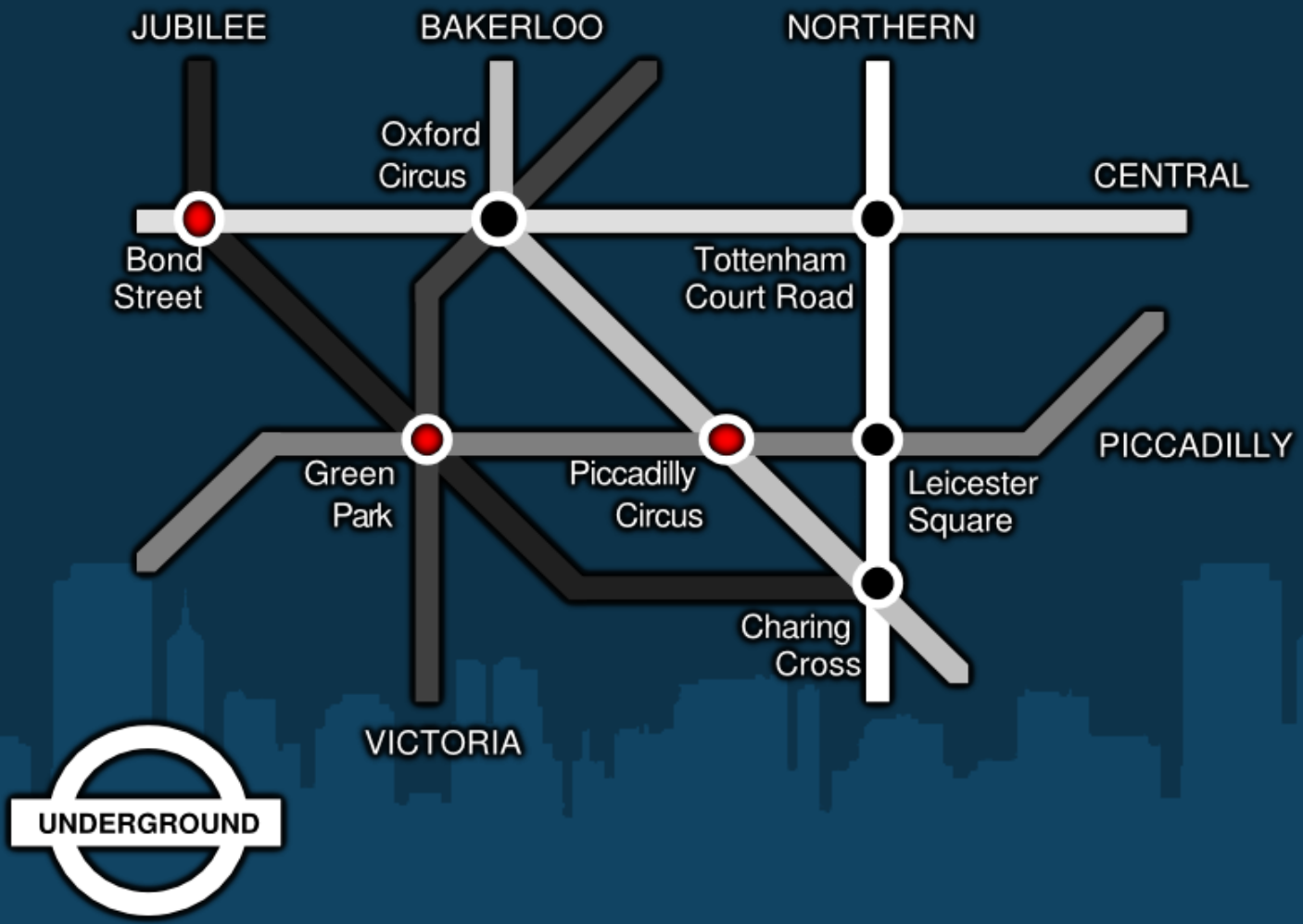
... same lines as **connected(X,Y)** produces, but then ...

**X = bond_street,**
**Y = tottenham_court_road ;**

This is new! You need to combine 2 predicates.

# Go beyond SQL

Implement a **reachable(X,Y)** predicate:
Stations **X** and **Y** are reachable if it is possible to travel from **X** to **Y** (and possibly changing the line).



# The **reachable** predicate

**?- reachable(X,Y).**

... same lines as **nearby(X,Y)**, but then out of the blue ...

**X = bond_street,**
**Y = piccadilly_circus;**

This is new! You need to take *Central Line* first and then change to *Bakerloo*.

# Debug trace of reachable (tottenham_court_road,X)

```
[trace]  ?- reachable(tottenham_court_road,X).
   Call: (6) reachable(tottenham_court_road, _G369) ? creep
   Call: (7) connected(tottenham_court_road, _G369, _G422) ? creep
   Exit: (7) connected(tottenham_court_road, leicester_square, northern) ? creep
   Exit: (6) reachable(tottenham_court_road, leicester_square) ? creep
X = leicester_square ;
   Redo: (6) reachable(tottenham_court_road, _G369) ? creep
   Call: (7) connected(tottenham_court_road, _G421, _G422) ? creep
   Exit: (7) connected(tottenham_court_road, leicester_square, northern) ? creep
   Call: (7) reachable(leicester_square, _G369) ? creep
   Call: (8) connected(leicester_square, _G369, _G422) ? creep
   Exit: (8) connected(leicester_square, charing_cross, northern) ? creep
   Exit: (7) reachable(leicester_square, charing_cross) ? creep
   Exit: (6) reachable(tottenham_court_road, charing_cross) ? creep
X = charing_cross ;
   Redo: (7) reachable(leicester_square, _G369) ? creep
   Call: (8) connected(leicester_square, _G421, _G422) ? creep
   Exit: (8) connected(leicester_square, charing_cross, northern) ? creep
   Call: (8) reachable(charing_cross, _G369) ? creep
   Call: (9) connected(charing_cross, _G369, _G422) ? creep
   Fail: (9) connected(charing_cross, _G369, _G422) ? creep
   Redo: (8) reachable(charing_cross, _G369) ? creep
   Call: (9) connected(charing_cross, _G421, _G422) ? creep
   Fail: (9) connected(charing_cross, _G421, _G422) ? creep
   Fail: (8) reachable(charing_cross, _G369) ? creep
   Fail: (7) reachable(leicester_square, _G369) ? creep
   Fail: (6) reachable(tottenham_court_road, _G369) ? creep
false.
```