

Artificial Intelligence: SWI Prolog Hints & tips

Oliver Ray

bristol.ac.uk



- Developed by J. Wielemaker at Uni of Amsterdam's Human-Computer Studies group (previously called Social Science Informatics Sociaal-Wetenschappelijke Informatica)
- Popular, free, open-source, cross-platform, well-documented toolchain with excellent libraries and community support under continuous development from 1987-2022
- Includes terminal-based program (**SWIPL**) a integrated graphical editor/debugger/profiler (**PceEmacs**) and an extensive hyperlinked, searchable online reference manual (**PIDoc**)
- Can also be used with a browser-based GUI (**SWISH**) that facilitates basic user interaction, Jupiter-style **notebooks** and which is also hosted as a free (sand-boxed) **web-service**
- In practice, Prolog programs are typically edited using generic file editors (e.g. **Notepad++**, **Atom**, **VS-Code**, etc.) with Prolog syntax bindings and run via SWIPL command-line tools (using the SWI **make** command to detect and reload any modified predicate definitions)

SWI Local Installation Tips

The latest SWI **setup instructions** and **binary installers** (and **source codes**) are available from <https://www.swi-prolog.org/download/stable>. Either follow the instructions to download, make and configure the source files or try one of the following platforms shortcuts:

- On **Linux** try running **sudo apt-add-repository ppa:swi-prolog/stable**.
- On **Mac** try running **brew install swi-prolog**. Alternatively download and run the **.dmg** file, drag swipl into the applications folder, add it to the path with something like **export PATH=\$PATH:/Applications/SWI-Prolog.app/Contents/MacOS** (and, as the publisher is “unknown”, to run it for the very the first time you’ll need to open the applications folder in finder, right click on swipl, and select open)
- On **Windows** download/run the **.exe** file (64 or 32 bit) making sure to tick the checkbox saying “**add swipl to the system path for ...**” (or you’ll need to do this later by adding something like “**C:\Program Files\swipl\bin**” to the system path using something like **System-> About-> Advanced System Settings->Environmental Variables->Path->Add**)

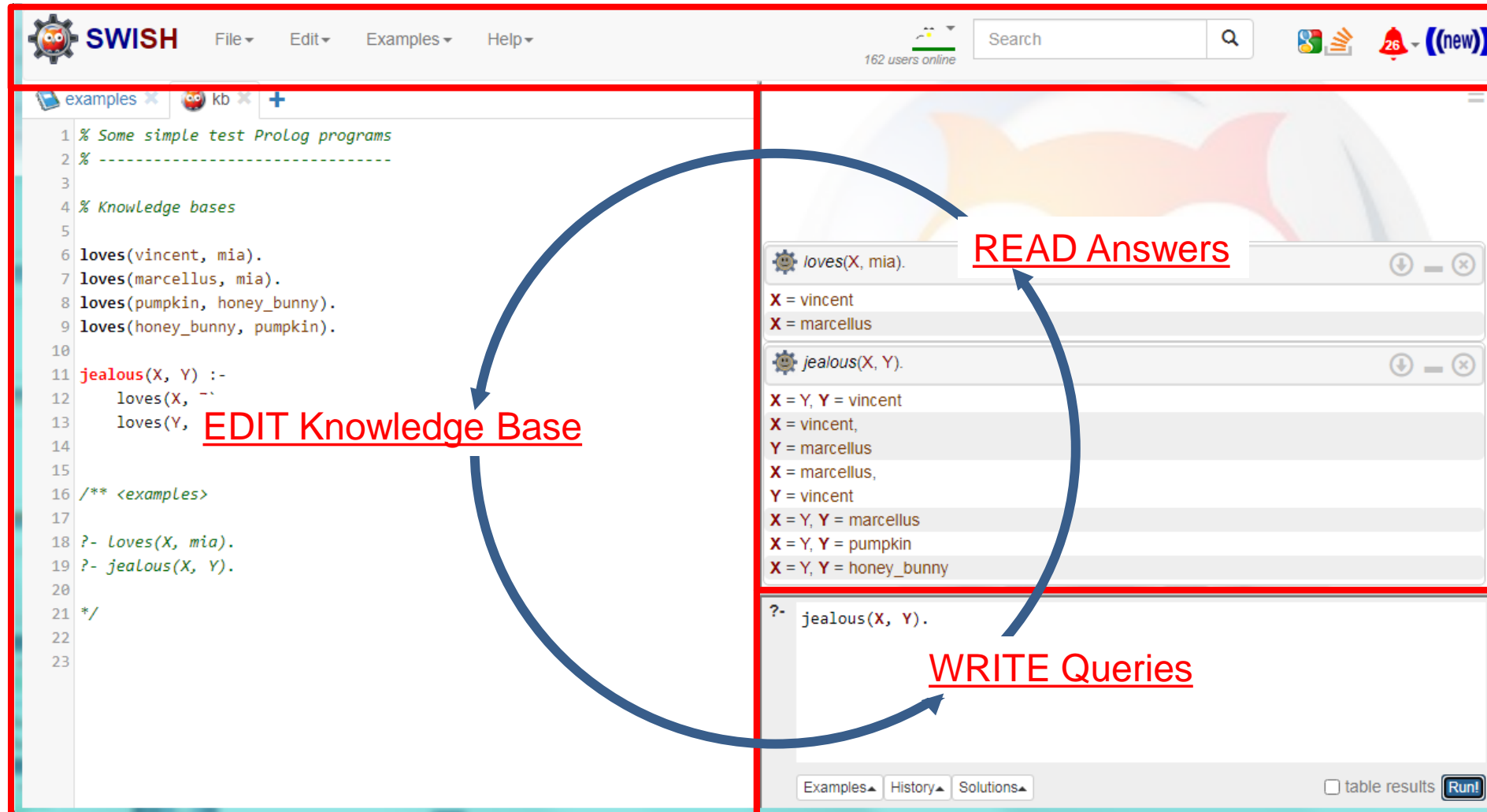
You may (optionally) obtain **SWISH** from [GitHub](#) or [Docker](#) to simulate the online GUI
see SWISH installation notes: [swish \(swi-prolog.org\)](https://www.swi-prolog.org/swish)

SWI Running Tips

- On **Linux** or **Mac** type **swipl** in a terminal; or double click the swipl **app**
- On **Windows** type **swipl** (command-line app) or **swipl-win** (window-based app) in a terminal (cmd or powershell); or double click the swipl **app** ; or double click a **.pl** file in an Explorer window (if necessary, after associating swipl or swipl-win with the .pl extension – not PERL!)
- On **Lab Machines** SWIPL and SWISH should both be pre-installed.
 - To run SWIPL
 - simply type **swipl** in a BASH terminal;
 - To run SWISH:
 - First copy it (one time only) to your home dir: `cp -r /opt/swish ~/`
 - Then go to your brand new swish copy: `cd ~/swish`
 - And start a SWISH server by typing : `swipl run.pl`
 - Now open any web browser and go to url: <http://localhost:3050/>
 - You can remotely access lab machines with this [Remote Desktop Guidance](#)
- Alternatively, you can use the **SWISH server** at <https://swish.swi-prolog.org/>

Useful SWI Commands

- `> swipl` % run SWI command interface
- `?- current_directory(D,D).` % show source directory (and keep it the same)
- `?- working_directory(D,D).` % in SWI use *working*, not *curent*
- `?- [file].` % load source code from file.pl
- `?- ['file.ext'].` % load source code from file.ext
- `?- [file1, file2].` % load source code from file1.pl and file2.pl
- `?- listing.` % show current predicatate definitions
- `?- listing(predicate/arity)` % show definitions for a specified predicate
- `?- make.` % detect and reload any modified definitions
- `;` % show more answers!
- `?- halt.` % exit from Prolog
- `<ctrl>-d` % abort / end of file
- `<ctrl>-c` % abort
- `> swipl file.ext` % run SWI command interface and load file
- `? associated_file(F).` % show the name of the loaded file
- `?- edit.` % open editor on the associated_file



The image shows the SWISH (Simple Web Interface for SWI-Prolog) interface, which is used for running Prolog programs. The interface is divided into two main sections: a code editor on the left and a query results panel on the right.

Left Panel (Code Editor): This panel contains the Prolog code. The code defines a knowledge base with facts and a rule. The facts are:

```
1 % Some simple test Prolog programs
2 % -----
3
4 % Knowledge bases
5
6 loves(vincent, mia).
7 loves(marcellus, mia).
8 loves(pumpkin, honey_bunny).
9 loves(honey_bunny, pumpkin).
10
11 jealous(X, Y) :-
12     loves(X, _),
13     loves(Y, _).
14
15 /** <examples>
16
17 ?- loves(X, mia).
18 ?- jealous(X, Y).
19
20 */
21
22
23
```

The code is labeled **EDIT Knowledge Base** with a red arrow pointing to it.

Right Panel (Query Results): This panel displays the results of queries. It shows two queries and their corresponding answers.

Query 1: `loves(X, mia).` The results are:

- X = vincent
- X = marcellus

Query 2: `jealous(X, Y).` The results are:

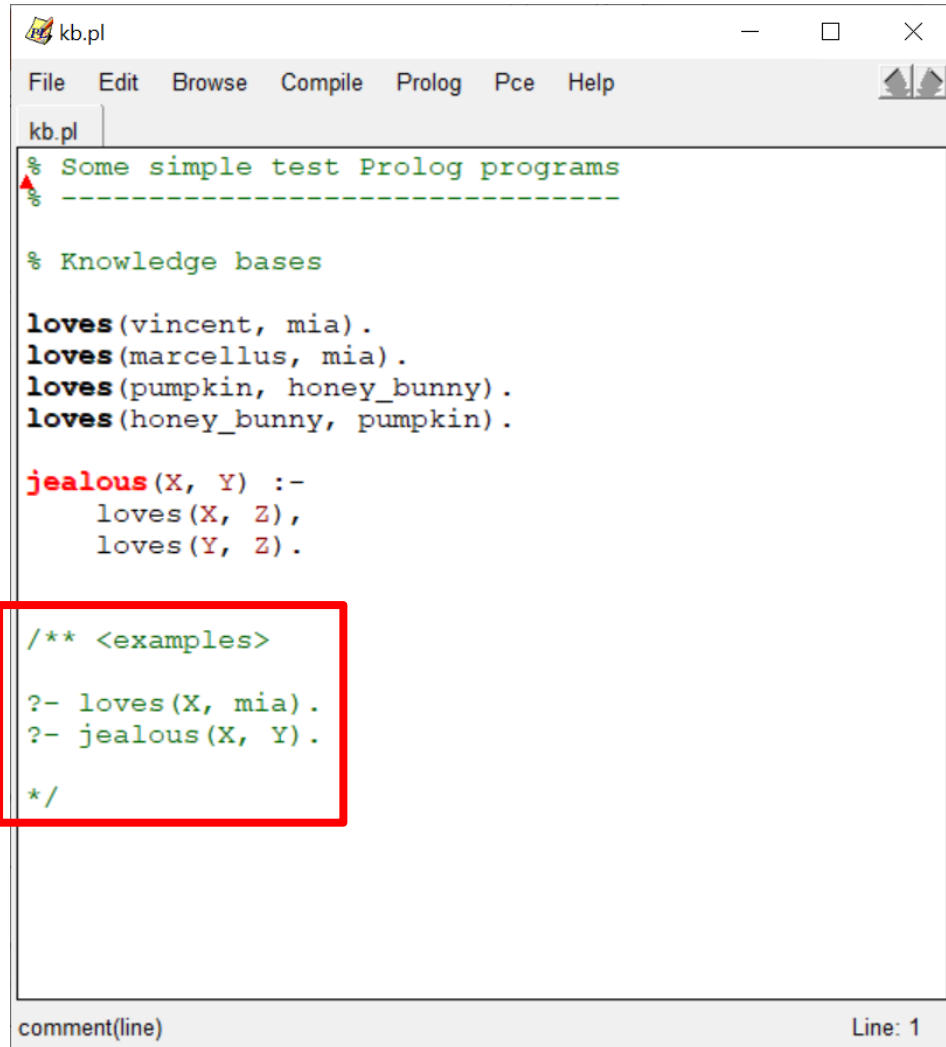
- X = Y, Y = vincent
- X = vincent, Y = marcellus
- X = marcellus, Y = vincent
- X = Y, Y = marcellus
- X = Y, Y = pumpkin
- X = Y, Y = honey_bunny

The results are labeled **READ Answers** with a red arrow pointing to them.

Bottom Panel (Query Input): This panel is used to enter queries. It shows the query `?- jealous(X, Y).` and is labeled **WRITE Queries** with a red arrow pointing to it.

The interface also includes a search bar, a user count (162 users online), and a "Run!" button at the bottom right.

Workflow with PceEmacs/swipl-win



```
kb.pl
File Edit Browse Compile Prolog Pce Help
kb.pl
% Some simple test Prolog programs
% -----

% Knowledge bases

loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

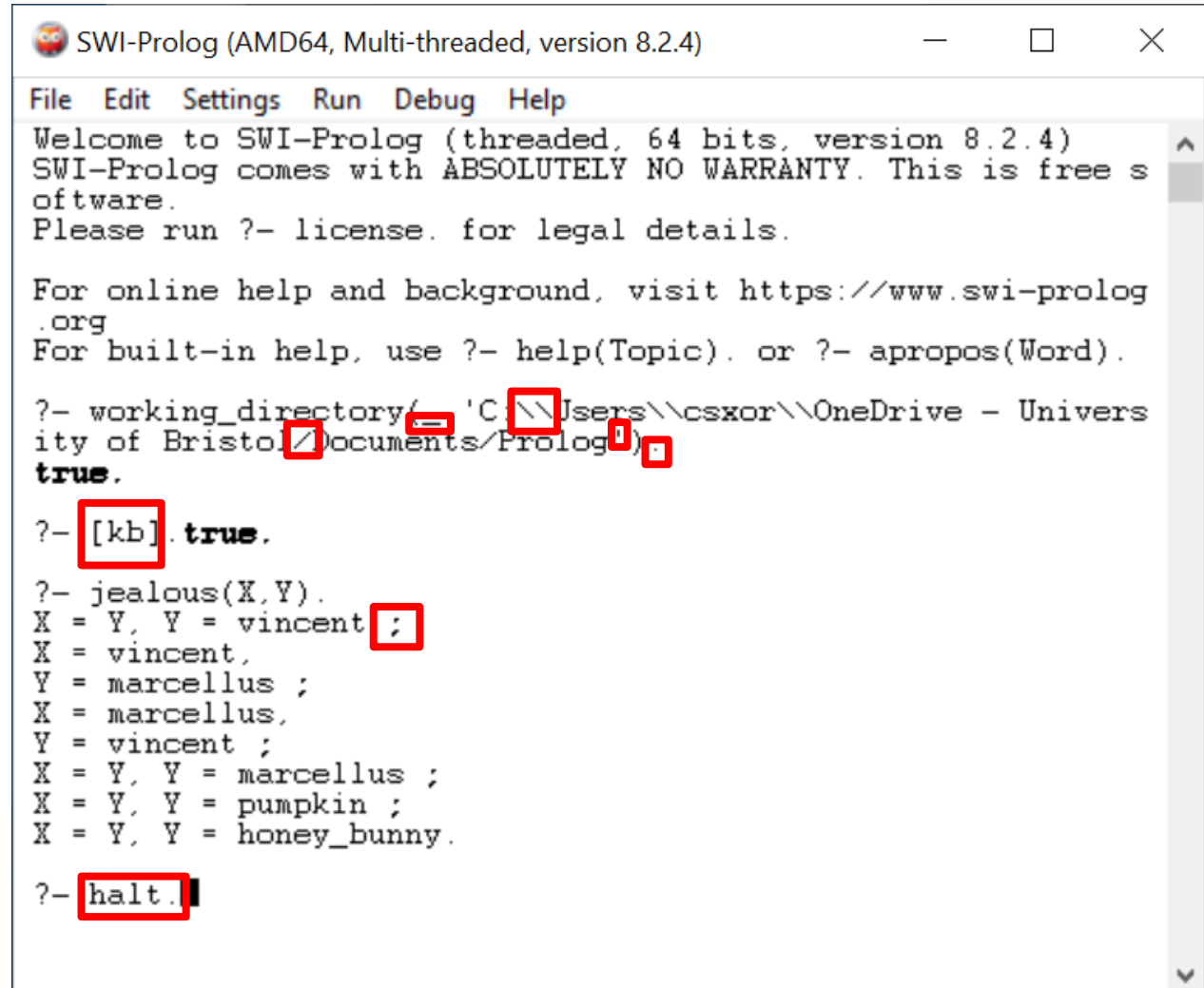
jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).

/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```

comment(line) Line: 1



```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free s
oftware.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog
.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

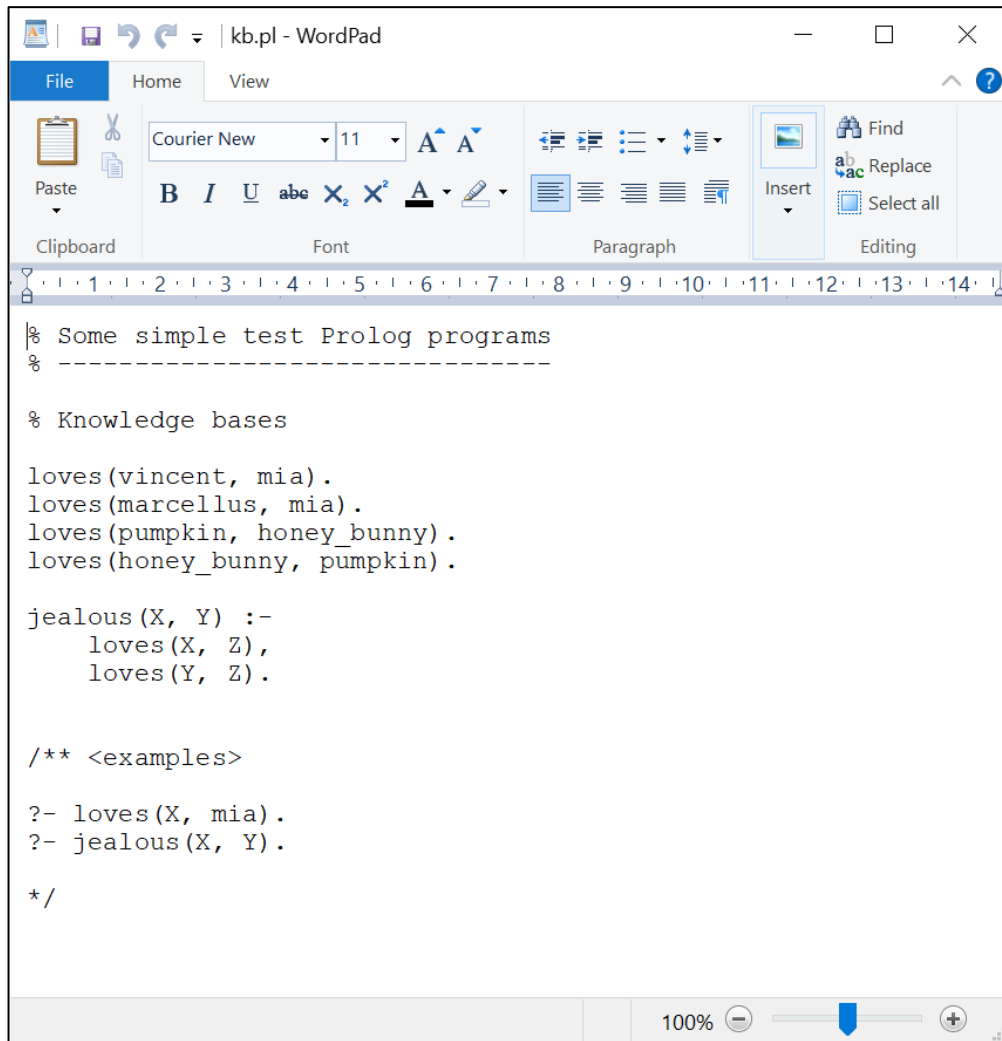
?- working_directory(_,'C:\\Users\\csxor\\OneDrive - Univers
ity of Bristol\\Documents\\Prolog')
true.

?- [kb].true.

?- jealous(X,Y).
X = Y, Y = vincent ;
X = vincent,
Y = marcellus ;
X = marcellus,
Y = vincent ;
X = Y, Y = marcellus ;
X = Y, Y = pumpkin ;
X = Y, Y = honey_bunny.

?- halt.
```

Workflow with text-editor/swipl



```
% Some simple test Prolog programs
% -----

% Knowledge bases

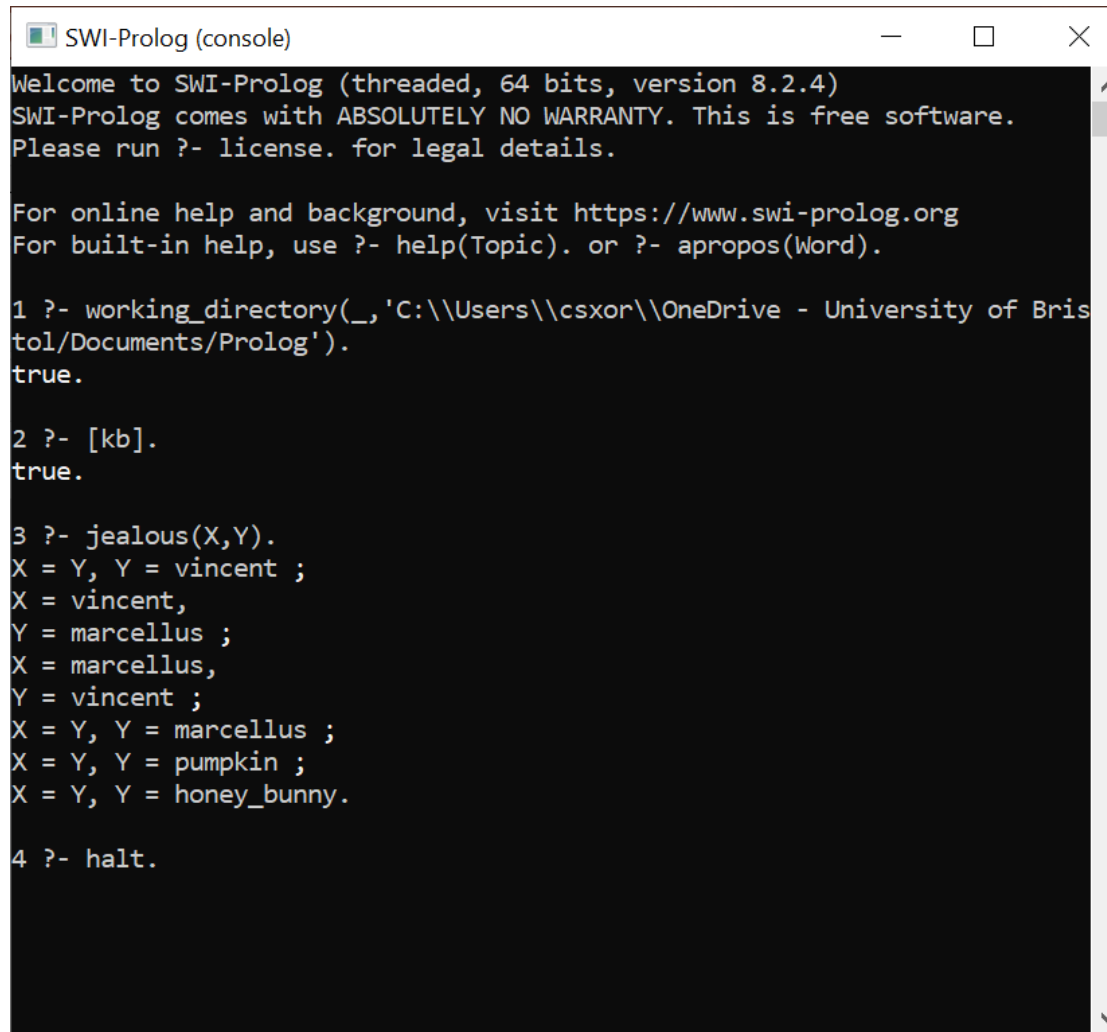
loves(vincent, mia).
loves(marcellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :-
    loves(X, Z),
    loves(Y, Z).

/** <examples>

?- loves(X, mia).
?- jealous(X, Y).

*/
```



```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- working_directory(_, 'C:\\Users\\csxor\\OneDrive - University of Bristol\\Documents\\Prolog').
true.

2 ?- [kb].
true.

3 ?- jealous(X,Y).
X = Y, Y = vincent ;
X = vincent,
Y = marcellus ;
X = marcellus,
Y = vincent ;
X = Y, Y = marcellus ;
X = Y, Y = pumpkin ;
X = Y, Y = honey_bunny.

4 ?- halt.
```


Miscellaneous Usage Notes

- There may be cosmetic differences between the output of the different SWI apps or when running those apps on different operating systems
 - e.g. one student noticed erroneous characters occasionally appear in the knowledge base if using the SWISH web service from a lab machine
 - e.g. last year some students noticed that some problems were caused the default web browser not being correctly configured on some lab machines
- Students are advised to spend a bit of time initially finding an installation setup that works for them on their own machine
- When using a 3rd party editor to modify Prolog programs, you should use the command **?- make.** to reload any updated files into SWIPL
- Remember to press semicolon ; to obtain more answers to a query!
- You can exit from Prolog using the command **?- halt.**

- Logical operators:

`:-` (if) `,` (and) `;` (or) `\+` (not)

- Comparison operators for (ground) numbers:

`<` , `>` , `=<` , `>=` , `==` , `\==`

- comparison operators for (arbitrary) terms:

`@<` , `@>` , `@=<` , `@>=` , `==` , `\==`

- Examples:

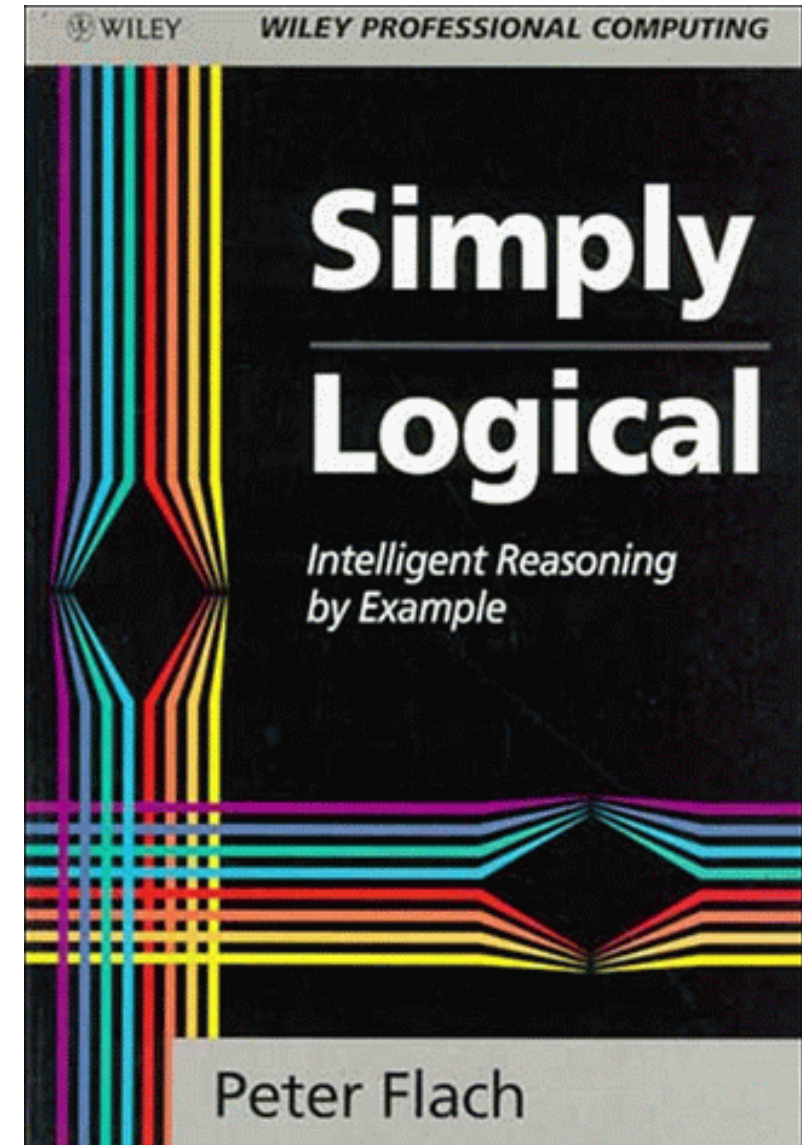
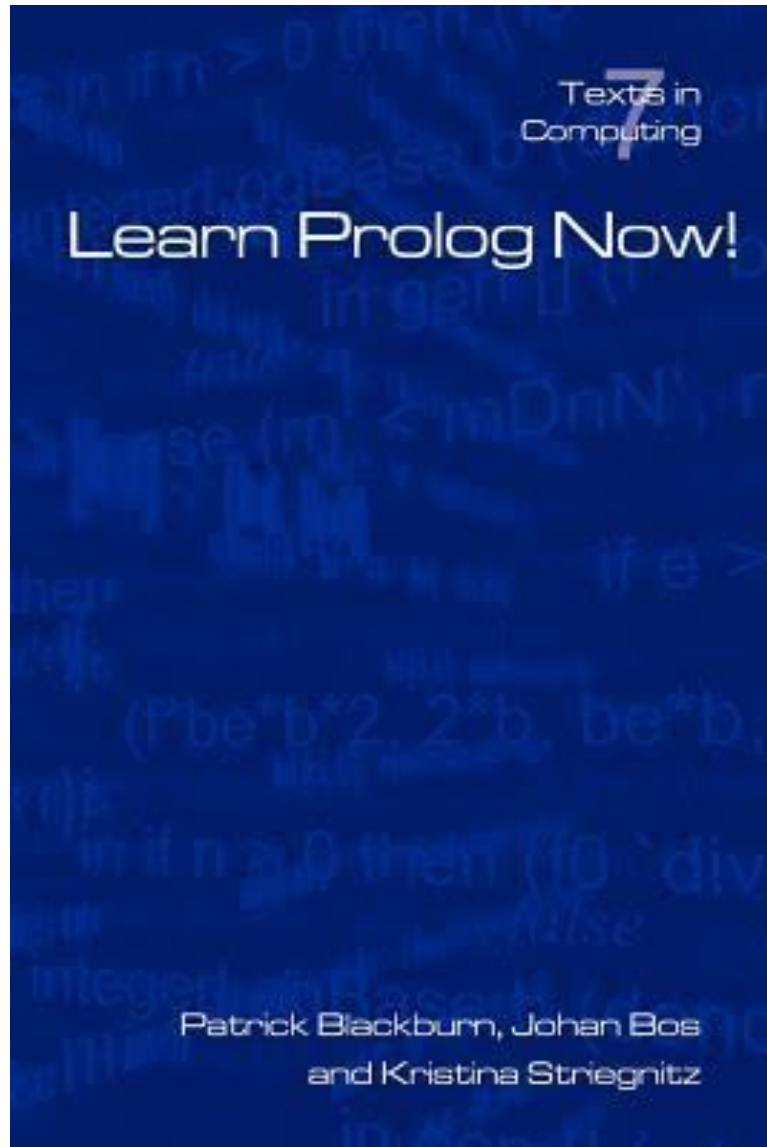
`teenager(X) :- (male(X) ; female(X)), age(X,Y), Y>12, Y<20.`

`brother(X,Y) :- male(X), parent(X,Z), parent(Y,Z), X\==Y.`

`only_child(X) :- \+ (parent(X,Z), parent(Y,Z), X\==Y).`

this space is important!

Free Online Prolog Resources



Thank you