University of BRISTOL

Artificial Intelligence: Logic Programming I

Oliver Ray

History of programming languages (simplified)

Functional: evaluate function definitions on given arguments

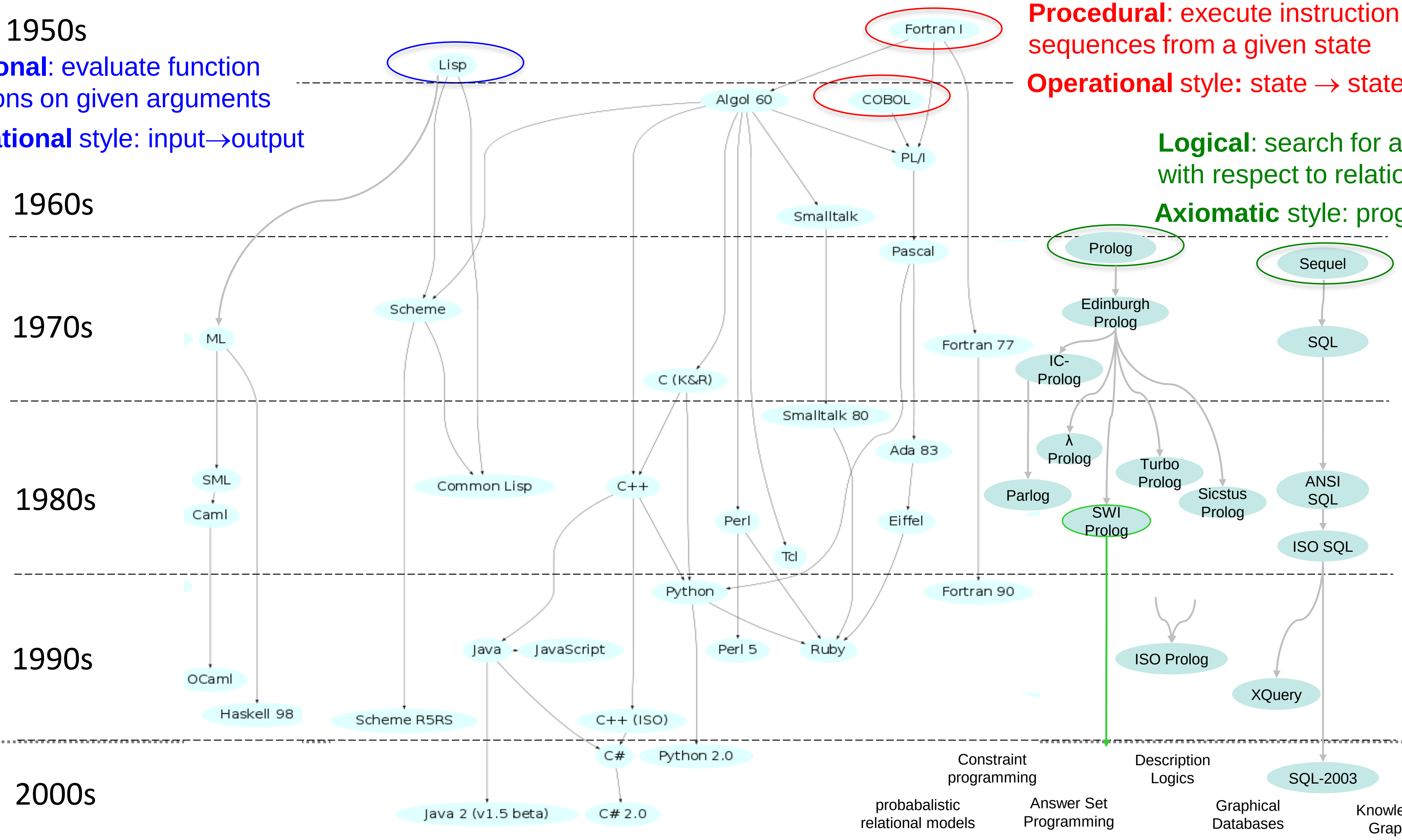
Denotational style: input→output

Procedural: execute instruction sequences from a given state

Operational style: state → state

Logical: search for answers to queries with respect to relational constraints

Axiomatic style: program |= consequences



PROgrammation en LOGique

Prolog is the most common logic programming language

- Simple syntax, no typing, homoiconic, queries may return any number of answers: none, one or many
- It focuses effort on problem definition, is great for proto-typing and has powerful language processing abilities
- It is also the basis of a many powerful extensions used in real-world tasks: e.g. Constraint Logic Prog. (CLP), Inductive Logic Prog. (ILP), Answer Set Prog. (ASP), …

University of BRISTOL

Datalog

We begin our exploration of Prolog starting from a very simple database perspective (**Datalog**$^{\neg}$) with an example

ACTOR		
Title	Name	Role
american_beauty	kevin_spacey	lester_burnham
…	…	…

ACTRESS		
Title	Name	Role
american_beauty	annette_bening	carolyn_burnham
…	…	…

MOVIE	
Title	Year
american_beauty	1999
anna	1987
…	…

DIRECTOR	
Title	Director
american_beauty	sam_mendes
anna	yurek_bogayevicz
…	…

Qu: Who directed a movie released after 2000 which they also acted in?

There are three key parts of a Prolog program:

- Prolog facts → relational database
  - one predicate per table; one fact per row

```
movie(american_beauty,1999).
```

- Prolog rules → relational views
  - intentional definition; materialised when needed

```
released_after(M,Y) :- movie(M,Z), Z>Y.
```

- Prolog queries → relational algebra (RA)
  - but Prolog is ***much*** easier than RA! (or SQL)

```
?- actor(_, A, _), director(_, A).
```

```
?- ( actor(M, A, _) ; actress(M, A, _) ) , director(M, A), released_after(M,1999) .
```

project union join select

SWI-Prolog

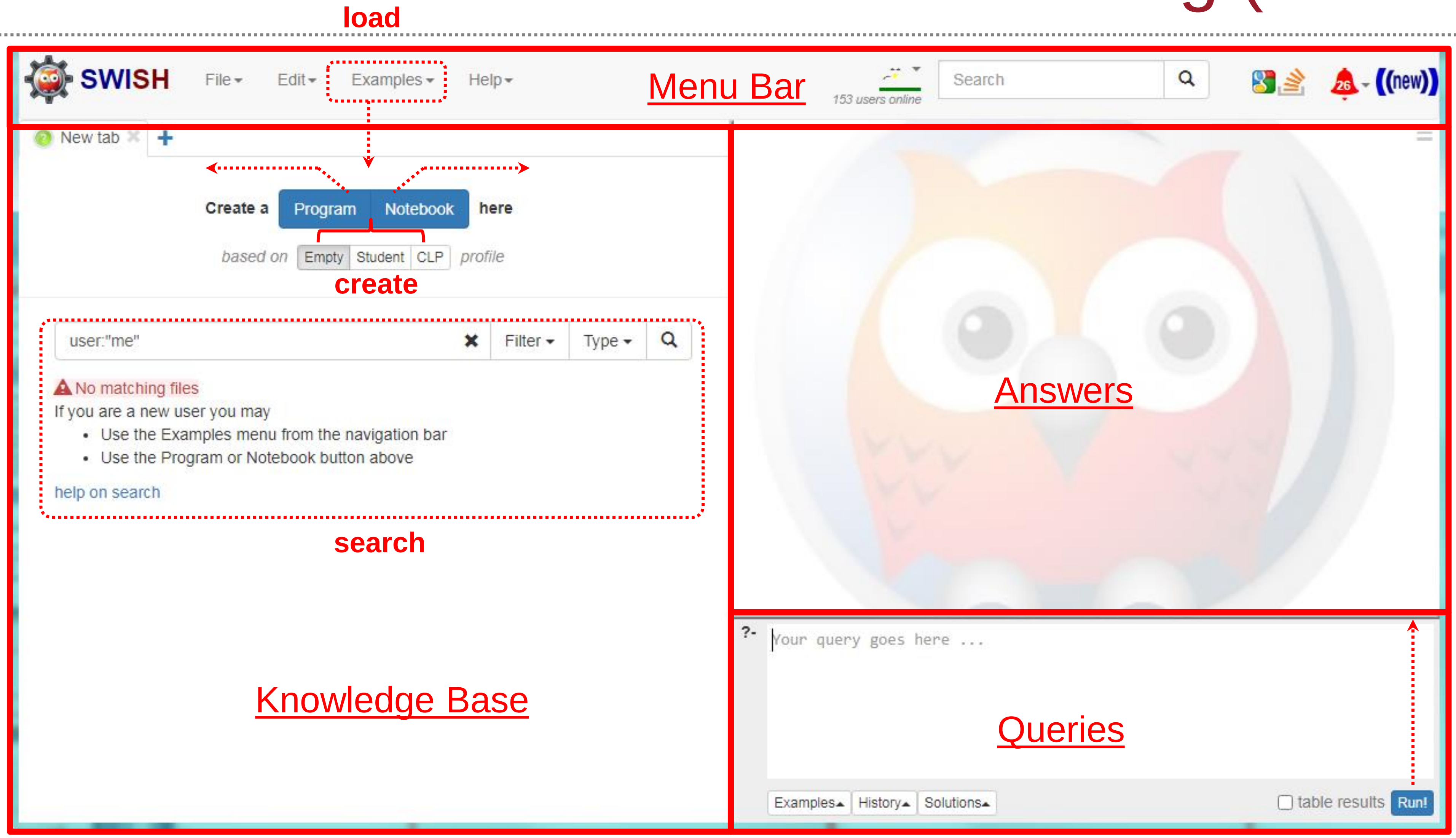
SWI-Prolog is a popular, well-supported, free Prolog system

- SWIPL Engine + SWISH IDE
- Easy to install on Linux, Mac and Windows
- Hosted as a sand-boxed web-service
- Pre-installed on CS lab machines
- User-friendly with lots of examples
- https://www.swi-prolog.org/

All these methods will be suitable for the week 1 lab, but weeks 2&3 will be best done with local installation

https://www.swi-prolog.org/pldoc/man?section=implhistory

SWI for Sharing (SWISH)



user:"me"

Filter Type

No matching files

If you are a new user you may

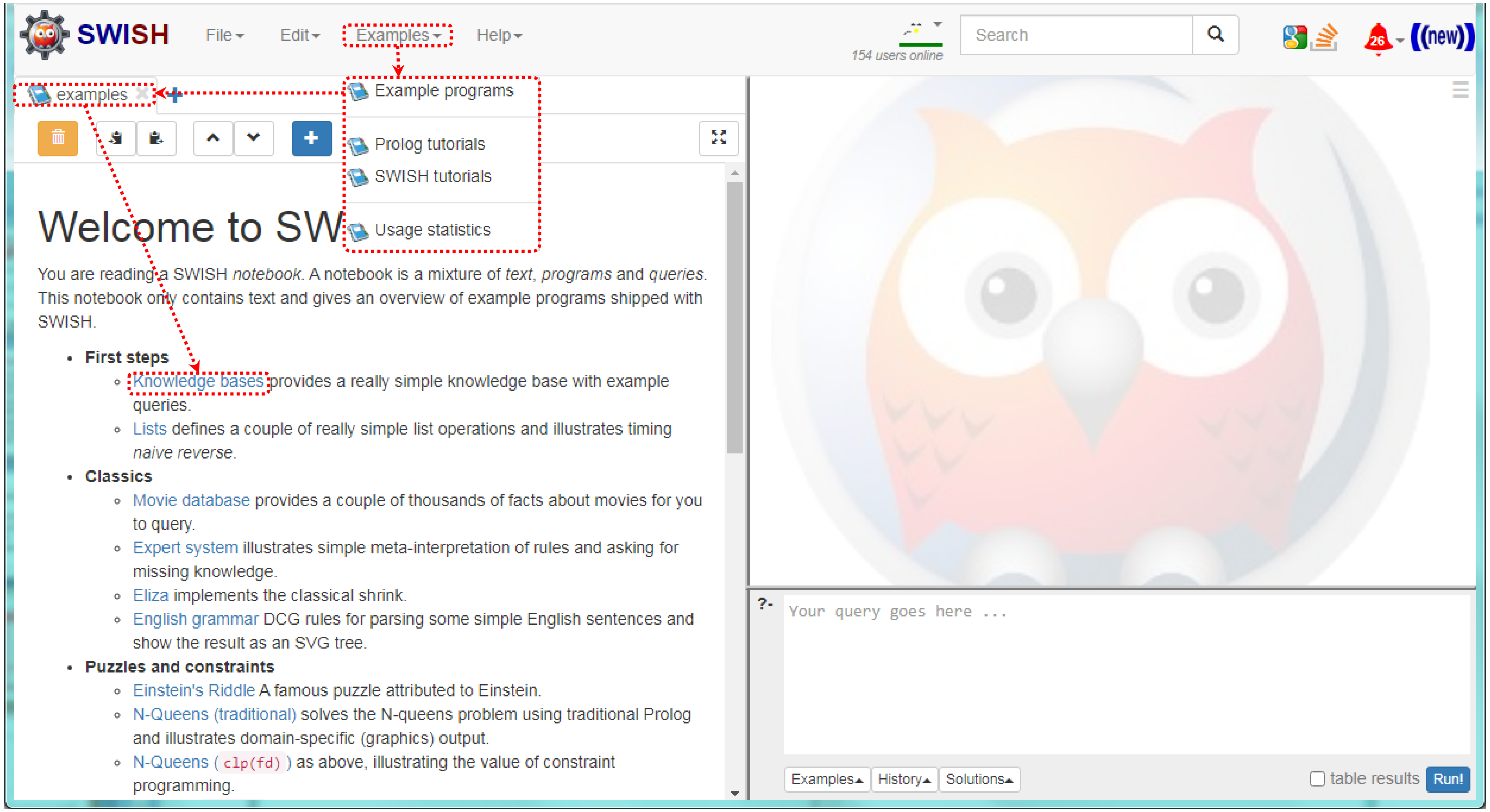
- Use the Examples menu from the navigation bar
- Use the Program or Notebook button above

help on search

Your query goes here ...

Can run locally or on hosted server at https://swish.swi-prolog.org/

Example Program

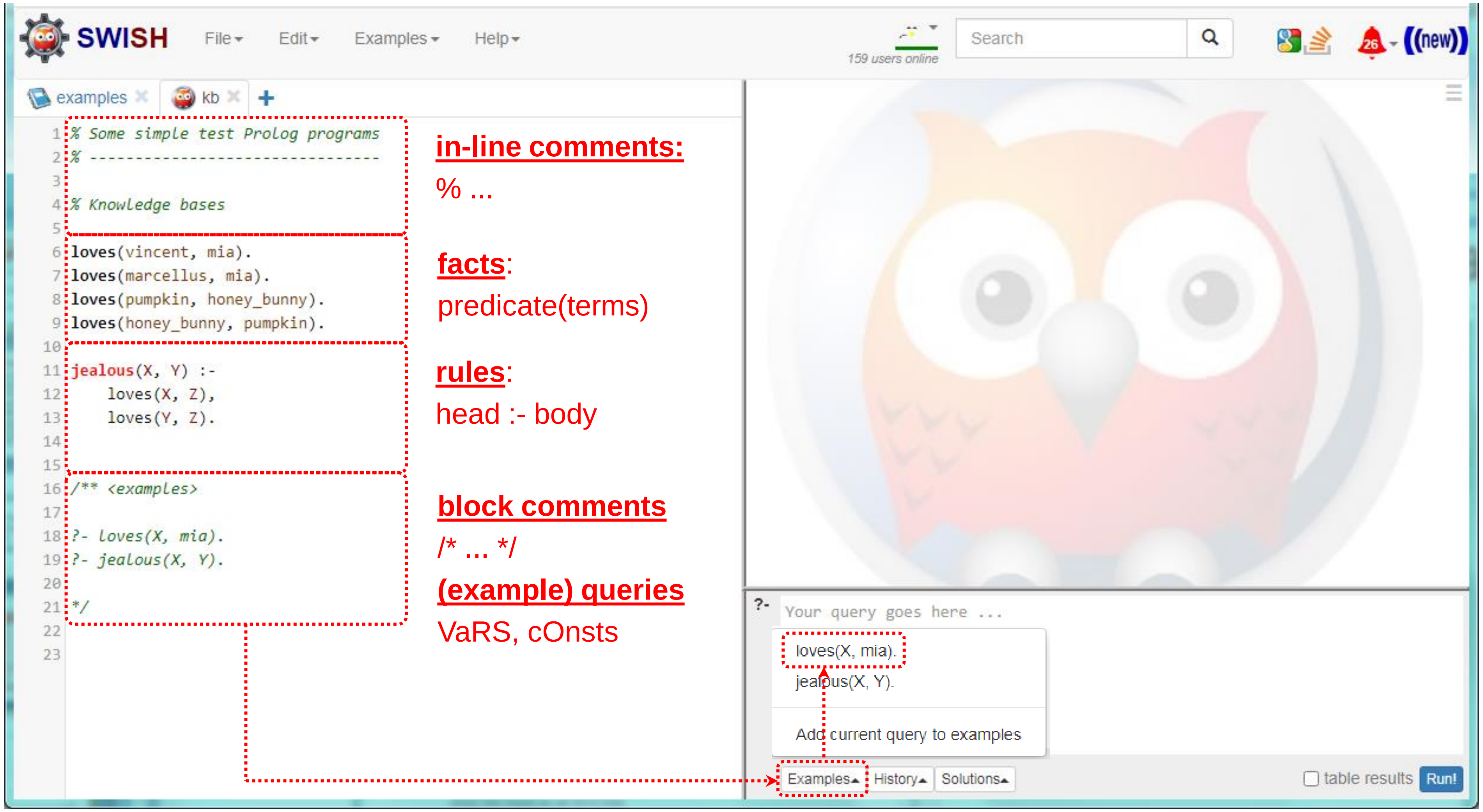


Welcome to SW

You are reading a SWISH *notebook*. A notebook is a mixture of *text*, *programs* and *queries*. This notebook only contains text and gives an overview of example programs shipped with SWISH.

- **First steps**
  - Knowledge bases provides a really simple knowledge base with example queries.
  - Lists defines a couple of really simple list operations and illustrates timing *naive reverse*.
- **Classics**
  - Movie database provides a couple of thousands of facts about movies for you to query.
  - Expert system illustrates simple meta-interpretation of rules and asking for missing knowledge.
  - Eliza implements the classical shrink.
  - English grammar DCG rules for parsing some simple English sentences and show the result as an SVG tree.
- **Puzzles and constraints**
  - Einstein's Riddle A famous puzzle attributed to Einstein.
  - N-Queens (traditional) solves the N-queens problem using traditional Prolog and illustrates domain-specific (graphics) output.
  - N-Queens (`clp(fd)`) as above, illustrating the value of constraint programming.

Example Program



```
% 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).

*/
```

in-line comments:
% ...

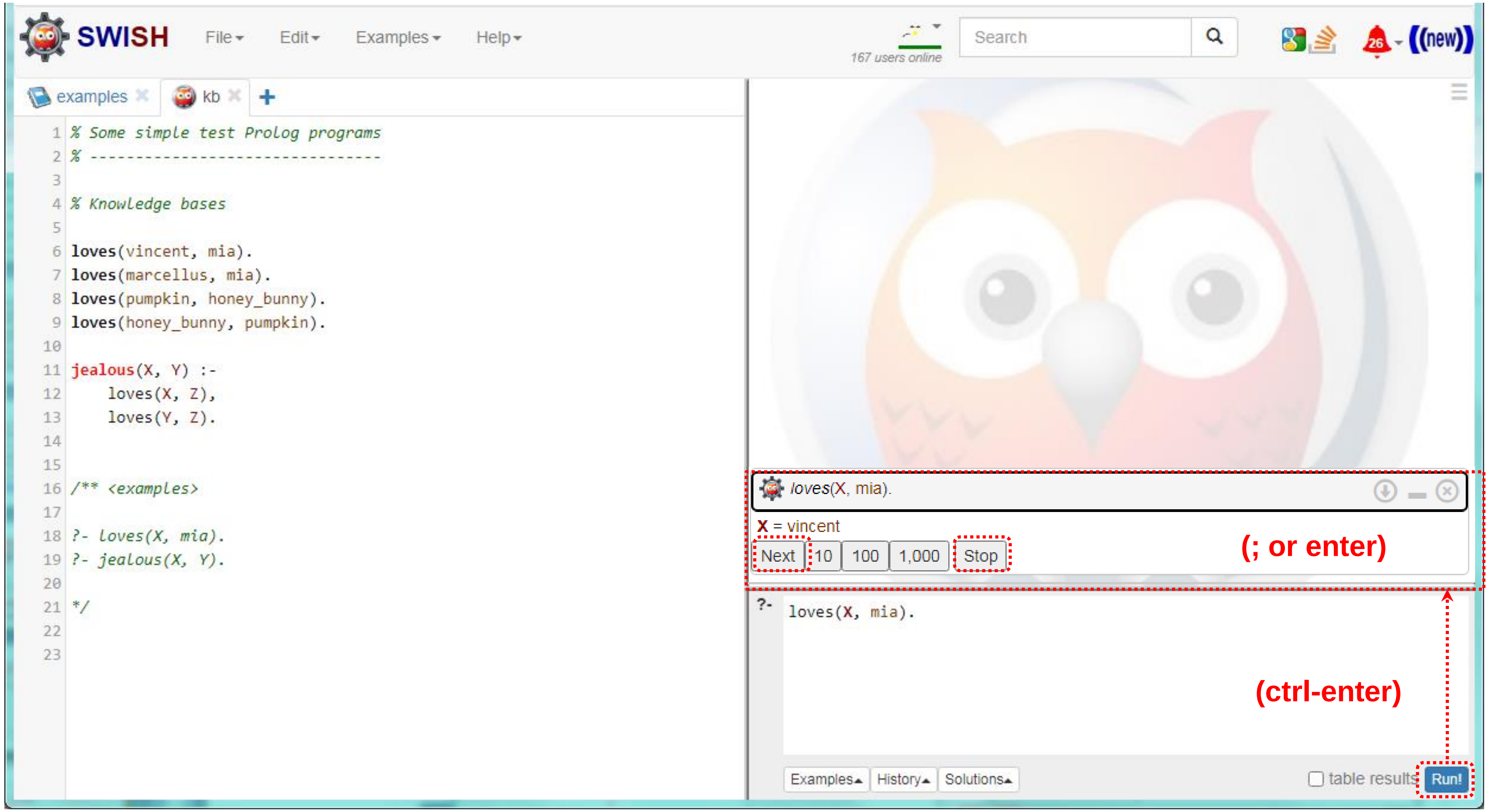
facts:
predicate(terms)

rules:
head :- body

block comments
/* ... */

(example) queries
VaRS, cOnsts

Example Program

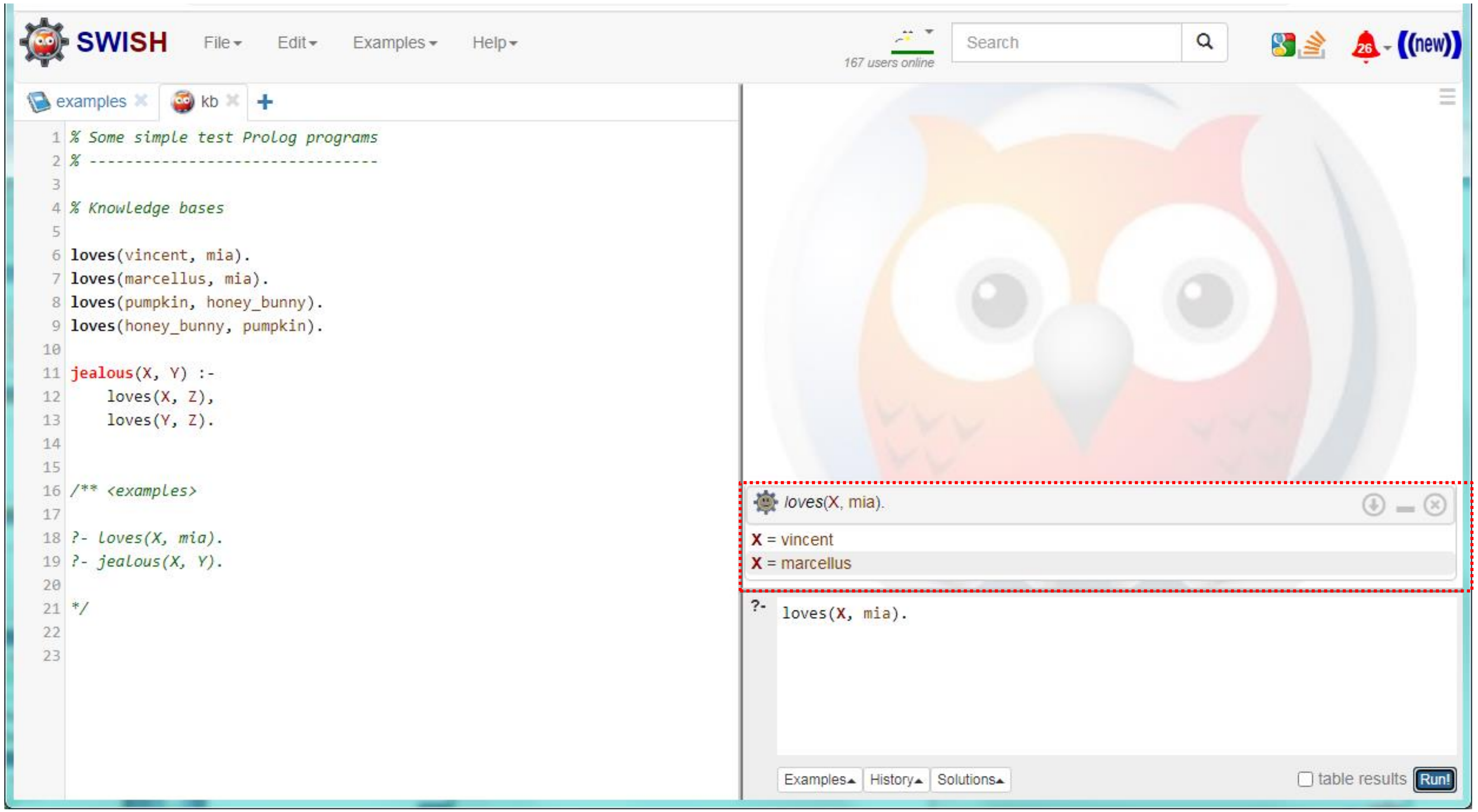


```
% Some simple test Prolog programs
% ---------------------------------

% 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).

*/
```

```
loves(X, mia).
X = vincent
```

```
loves(X, mia).
```

Example Program

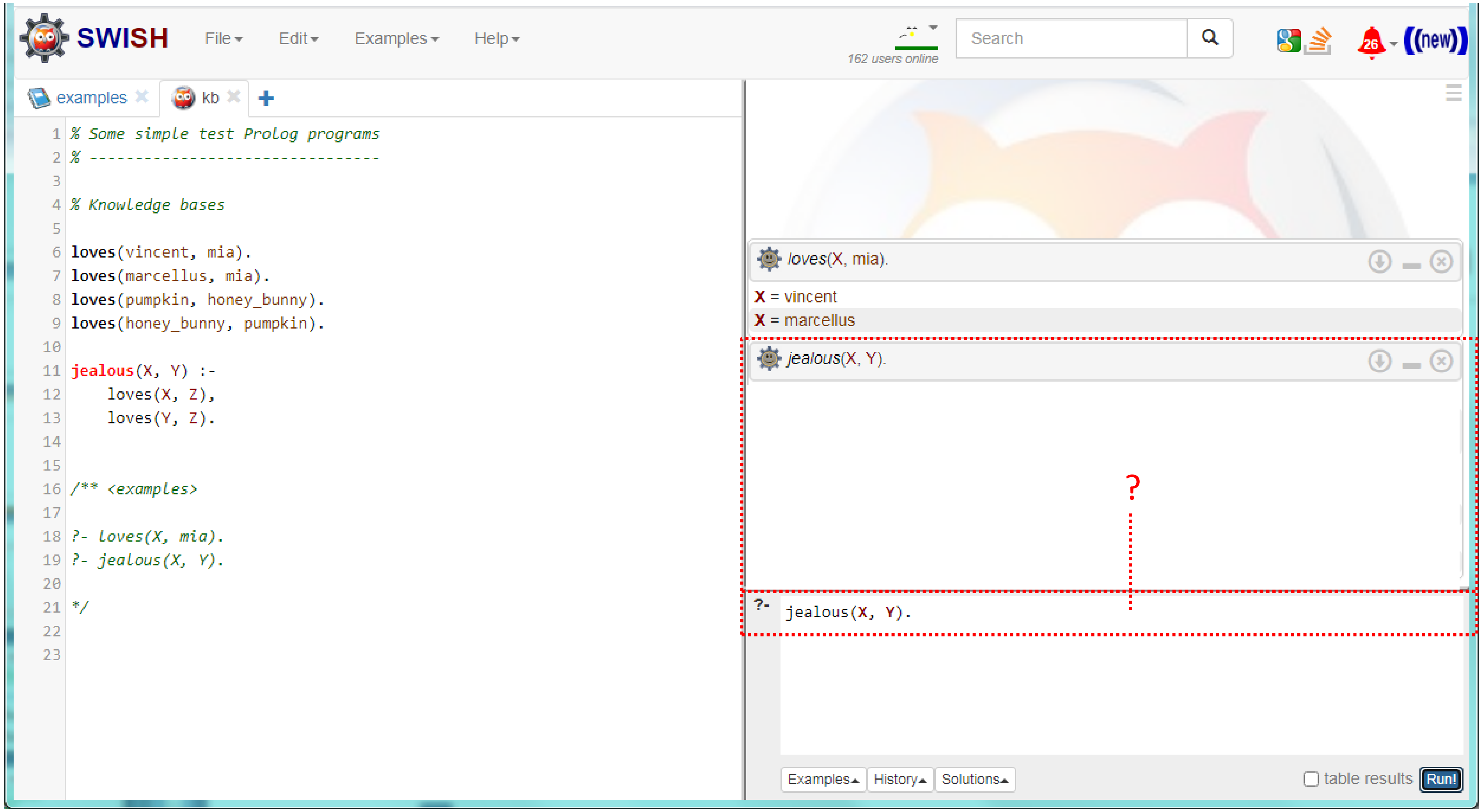


```prolog
% Some simple test Prolog programs
% --------------------------------

% 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).

*/
```

```
loves(X, mia).
X = vincent
X = marcellus
```

```
?- loves(X, mia).
```

Example Program



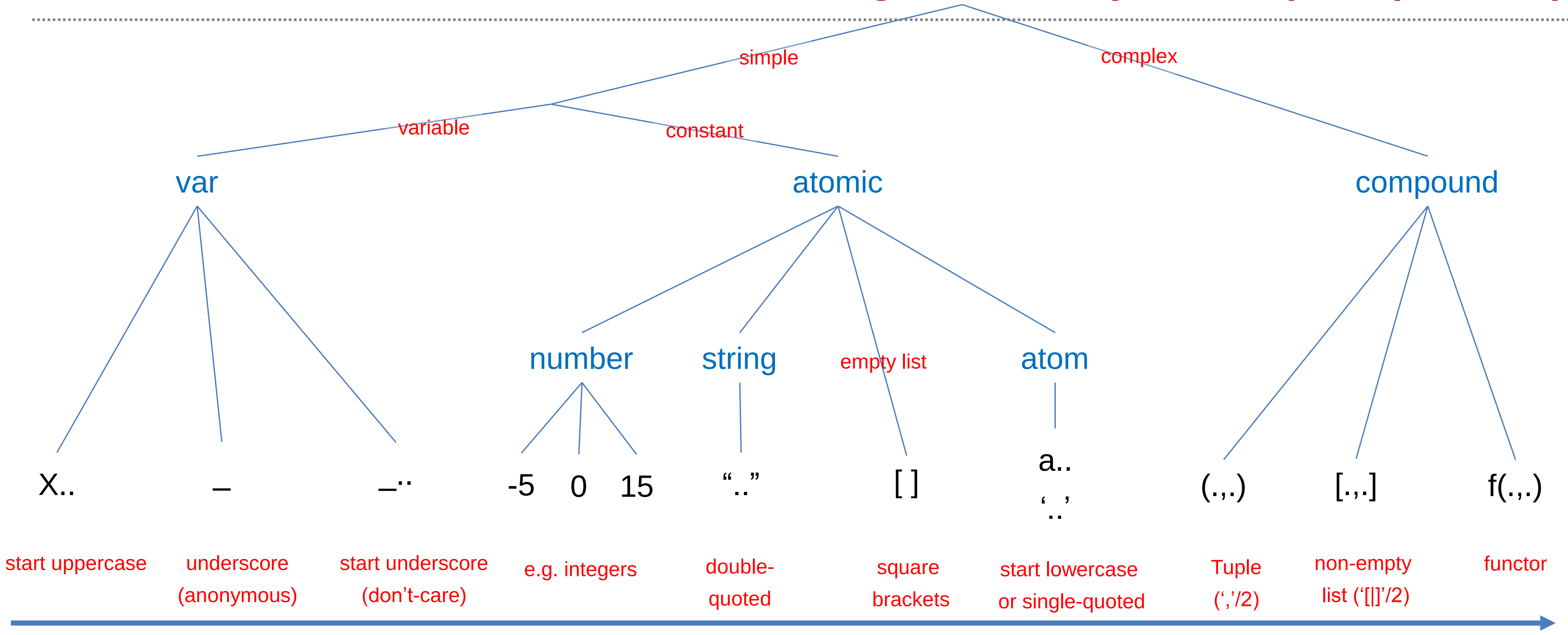
```
% Some simple test Prolog programs
% --------------------------------

% 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).

*/
```

```
loves(X, mia).
X = vincent
X = marcellus
jealous(X, Y).
```

?

```
?- jealous(X, Y).
```

Overview of Prolog Term Syntax (simplified)



https://www.swi-prolog.org/pldoc/man?section=standardorder

comparison operators for (ground) numbers: < , > , =< , >= , == , \==

comparison operators for (arbitrary) terms: @< , @> , @=< , @>= , == , \== (take care if comparing variables!)

Actions!

- Have a look around the SWI Prolog website and try to download, install, and run the SWIPL engine and SWISH IDE locally on your computer (highly recommended!)
- This should be relatively easy, but in case of issues, for now try working through this web-hosted SWISH server or try and remotely run SWIPL on the CS lab machines.
- Work through chapters 1 and 3 of the excellent free on-line tutorial Learn Prolog Now! which will take you through the basics of Prolog very simply and effectively

University of BRISTOL

Thank you