

Artificial Intelligence: Logic Programming I

Oliver Ray

bristol.ac.uk



Remember to Check-in

The quick and easy way to register your attendance.

- Download the University of Bristol Check-in app
- Check location is enabled on device and app
- Log in
- Click on class in 'Check in Now'
- And then click Check-in and Finish

Make sure your attendance is registered for all your on-campus classes



App



**About -
webpage**

Functional: evaluate function definitions on given arguments
Denotational style: input \rightarrow output

- Procedural:** execute instruction sequences from a given state
- Operational style:** state \rightarrow state

- Logical:** search for answers to queries with respect to relational constraints
- Axiomatic** style: program \models consequences

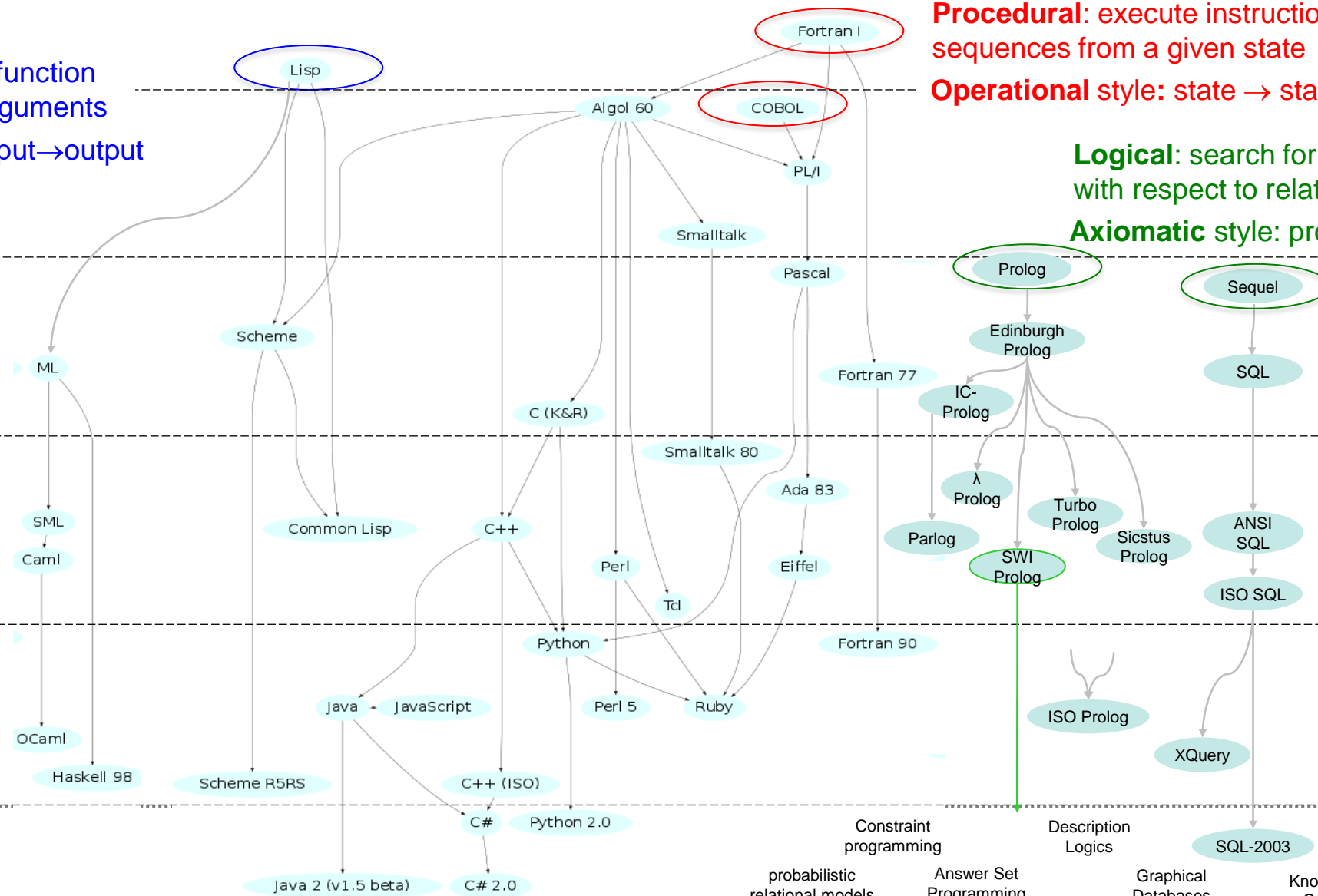


Diagram illustrating the relationship between various database paradigms and SQL-2003:

- Constraint programming** (includes probabilistic relational models and Answer Set Programming)
- Description Logics** (includes Graphical Databases and Knowledge Graphs)
- SQL-2003** (highlighted as a central concept)

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 prototyping 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), ...

We begin our exploration of Prolog starting from a very simple database perspective (**Datalog**) with an example

| ACTOR | | | ACTRESS | | |
|-----------------|--------------|----------------|-----------------|----------------|-----------------|
| <u>Title</u> | <u>Name</u> | <u>Role</u> | <u>Title</u> | <u>Name</u> | <u>Role</u> |
| american_beauty | kevin_spacey | lester_burnham | american_beauty | annette_bening | carolyn_burnham |
| ... | ... | ... | ... | ... | ... |

| MOVIE | |
|-----------------|-------------|
| <u>Title</u> | <u>Year</u> |
| american_beauty | 1999 |
| anna | 1987 |
| ... | ... |

| DIRECTOR | |
|-----------------|------------------|
| <u>Title</u> | <u>Director</u> |
| 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

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

- one predicate per table; one fact per row

- Prolog rules → relational views

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

- intentional definition; materialised when needed

- Prolog queries → relational algebra (RA)

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

- but Prolog is ***much*** easier than RA! (or SQL)

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

project

union

join

select

bristol.ac.uk

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



SWI for Sharing (SWISH)

load

Menu Bar

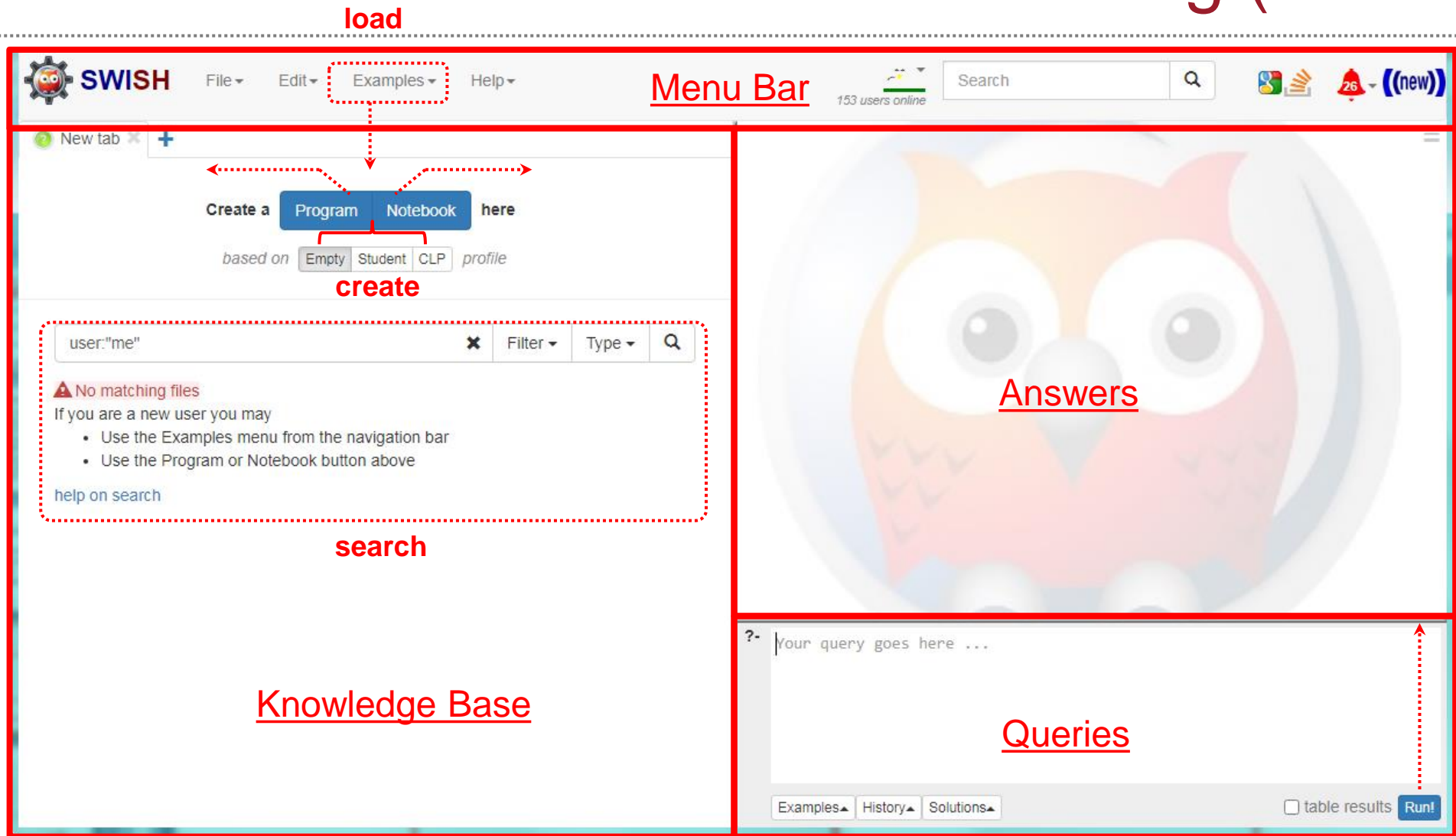
create

search

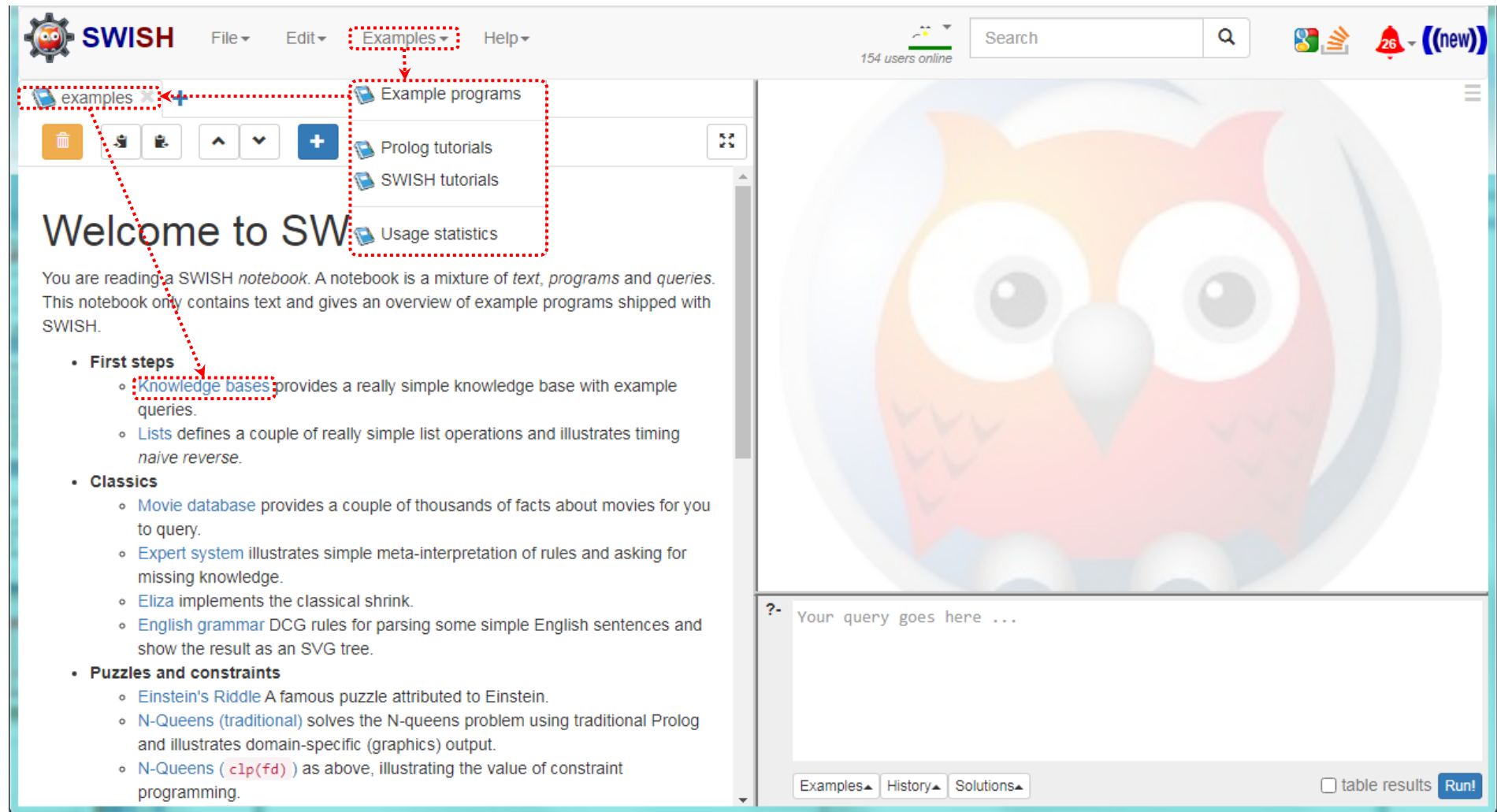
Knowledge Base

Answers

Queries



Example Program



The screenshot shows the SWISH web interface. The top navigation bar includes 'File', 'Edit', 'Examples', and 'Help'. The 'Examples' menu is open, showing options: 'Example programs', 'Prolog tutorials', 'SWISH tutorials', and 'Usage statistics'. The 'examples' tab is selected in the left sidebar. The main content area displays 'Welcome to SWISH' and a description of a notebook. Below this, there are three sections: 'First steps', 'Classics', and 'Puzzles and constraints'. The 'First steps' section includes links to 'Knowledge bases', 'Lists', and 'naive reverse'. The 'Classics' section includes links to 'Movie database', 'Expert system', 'Eliza', and 'English grammar'. The 'Puzzles and constraints' section includes links to 'Einstein's Riddle', 'N-Queens (traditional)', and 'N-Queens (c1p(fd))'. The right sidebar features a large owl illustration and a query input field with the placeholder text 'Your query goes here ...'. At the bottom right, there are buttons for 'Examples', 'History', 'Solutions', a checkbox for 'table results', and a 'Run!' button.

SWISH File Edit Examples Help

154 users online Search

examples Example programs

Prolog tutorials

SWISH tutorials

Usage statistics

Welcome to SWISH

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.

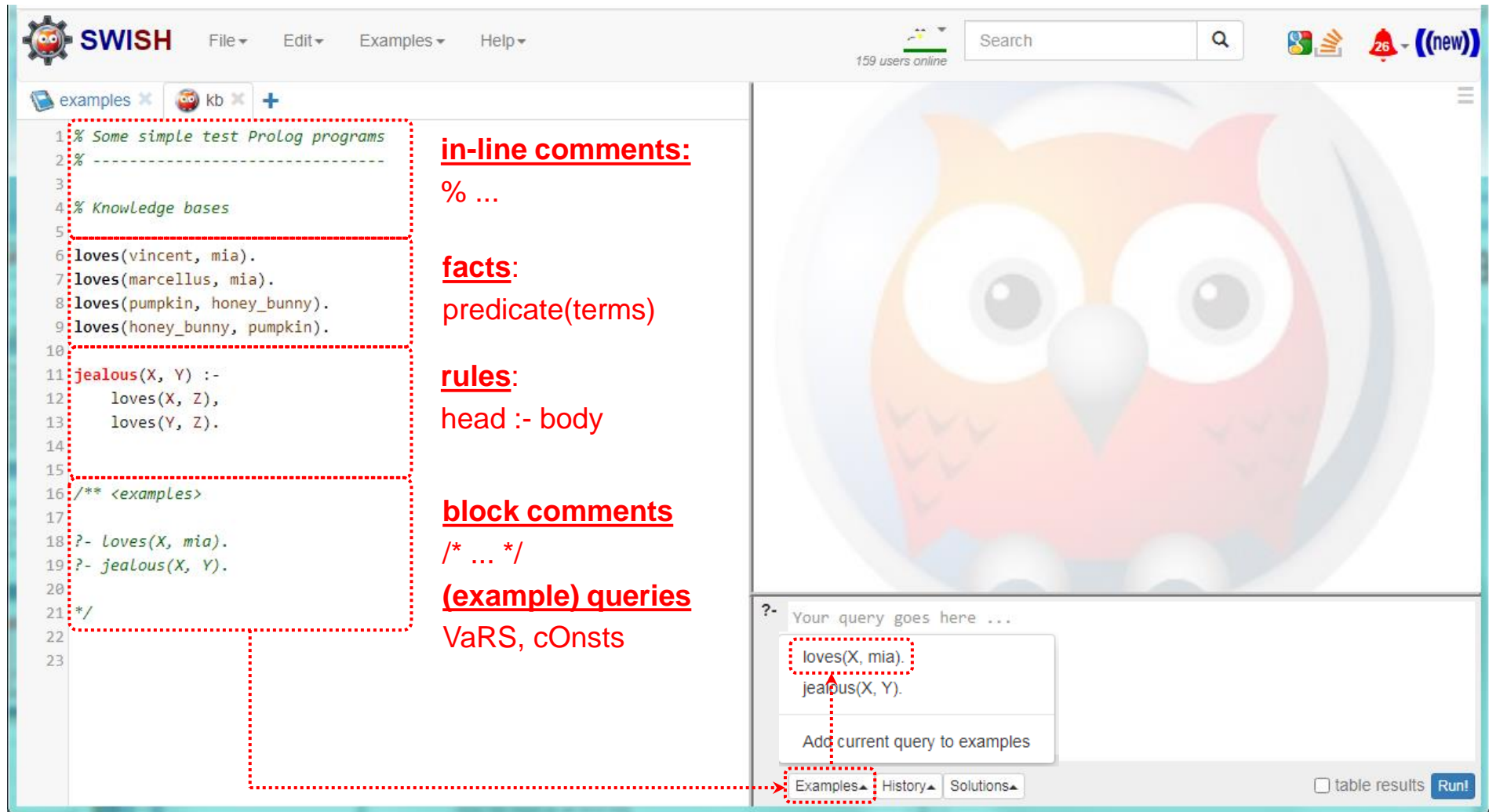
- **First steps**
 - [Knowledge bases](#) provides a really simple knowledge base with example queries.
 - [Lists](#) defines a couple of really simple list operations and illustrates timing *naive reverse*.
- **Classics**
 - [Movie database](#) provides a couple of thousands of facts about movies for 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 \(c1p\(fd\)\)](#) as above, illustrating the value of constraint programming.

?- Your query goes here ...

Examples History Solutions

☐ table results Run!

Example Program



The screenshot shows the SWISH Prolog IDE interface. The left pane displays a Prolog program with the following code:

```
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, Z),
13     loves(Y, Z).
14
15
16 /** <examples>
17
18 ?- loves(X, mia).
19 ?- jealous(X, Y).
20
21 */
22
23
```

Red dashed boxes highlight specific parts of the code, with arrows pointing to explanatory text on the right:

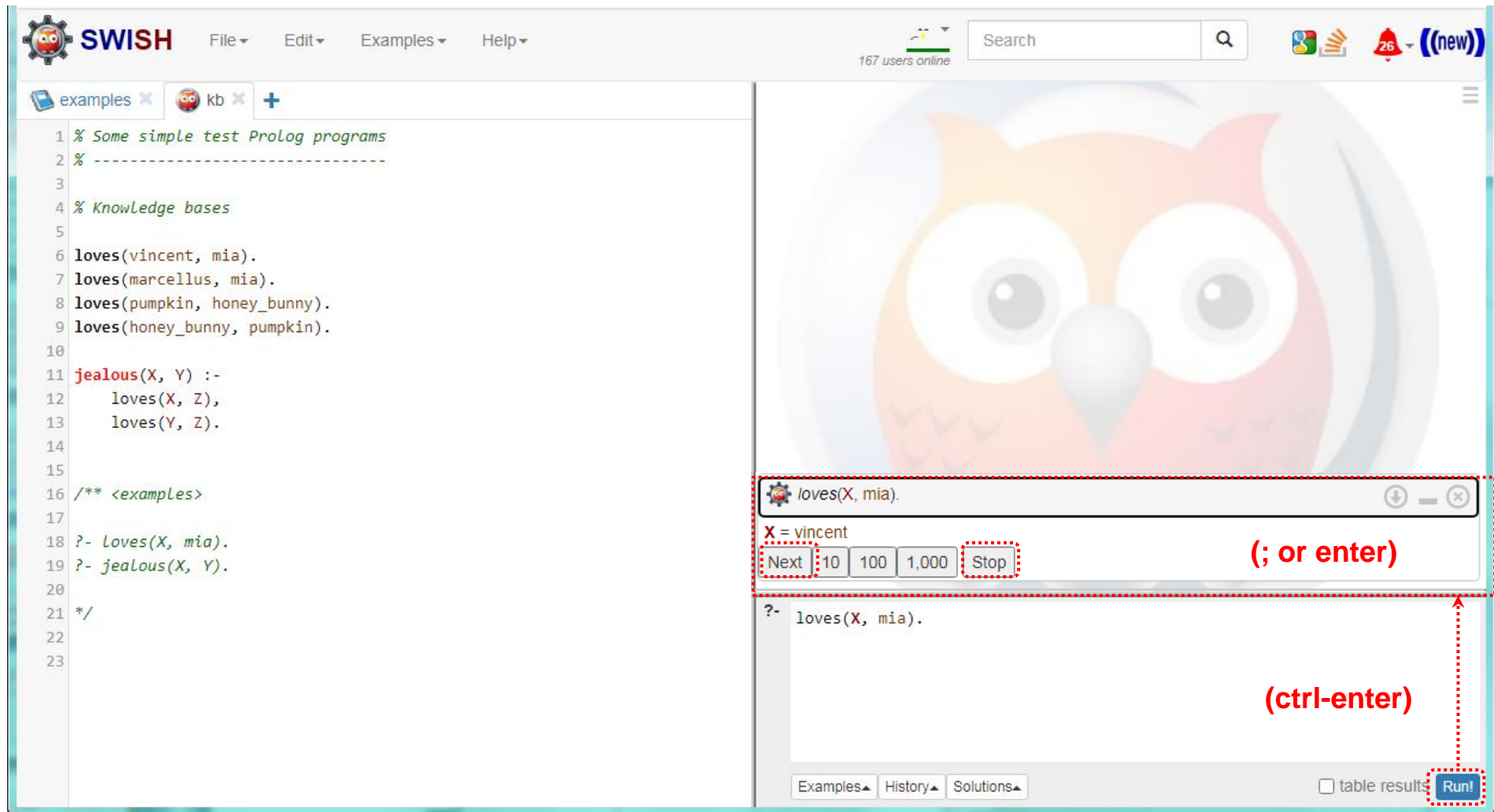
- in-line comments:** % ... (points to line 1)
- facts:** predicate(terms) (points to lines 6-9)
- rules:** head :- body (points to lines 11-13)
- block comments** /* ... */ (points to lines 16-21)
- (example) queries** VaRS, cOnsts (points to lines 18-19)

The right pane shows a large owl graphic. Below it, the query input area contains:

```
?- Your query goes here ...
loves(X, mia).
jealous(X, Y).
```

Below the query input, there is a button labeled "Add current query to examples". At the bottom of the right pane, there are tabs for "Examples", "History", and "Solutions", with "Examples" selected. A checkbox for "table results" and a "Run!" button are also visible.

Example Program

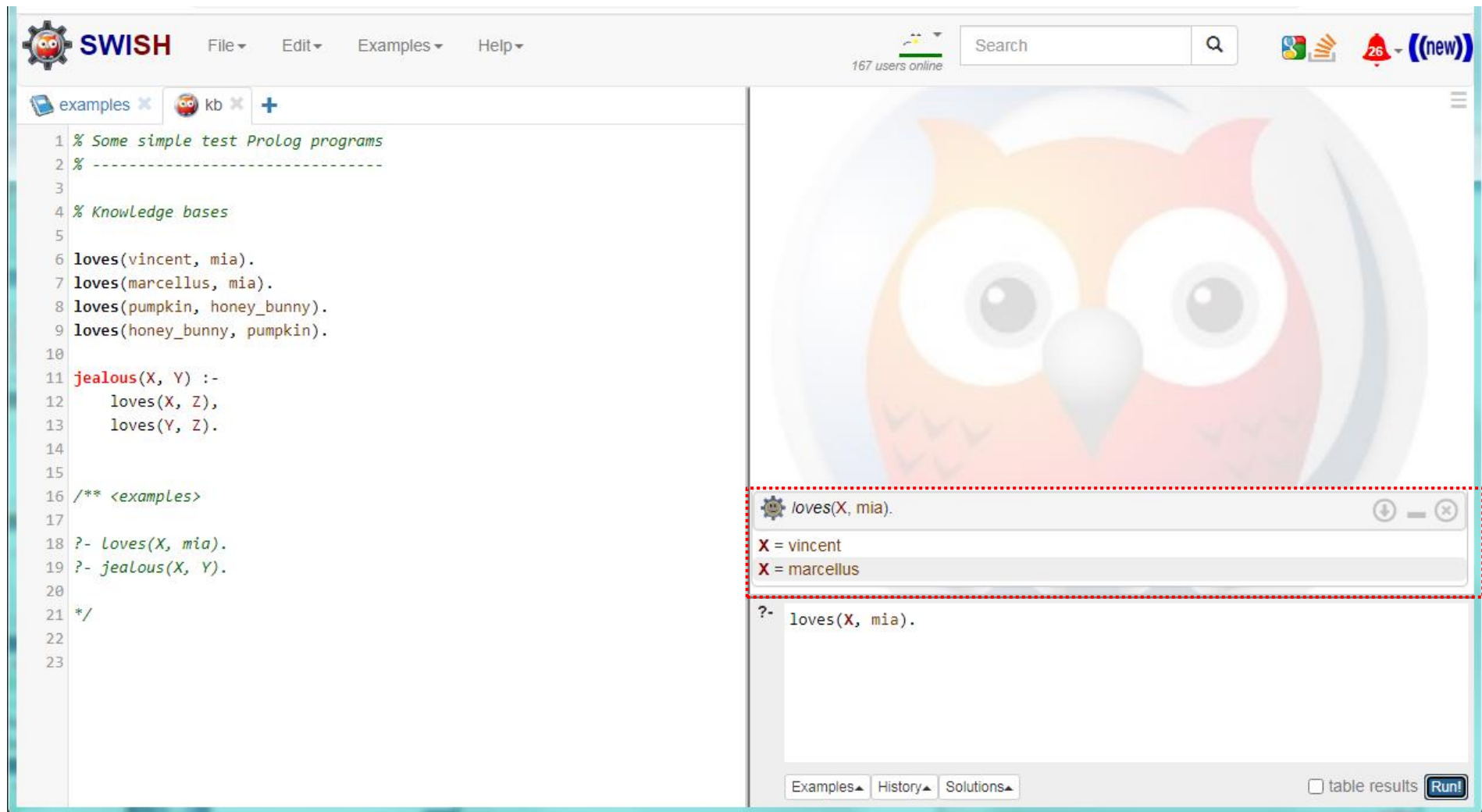


The screenshot shows the SWISH Prolog IDE interface. The left pane contains a Prolog program with the following code:

```
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, Z),
13     loves(Y, Z).
14
15
16 /** <examples>
17
18 ?- loves(X, mia).
19 ?- jealous(X, Y).
20
21 */
22
23
```

The right pane shows the execution of the query `loves(X, mia).`. The result is `X = vincent`. Below the result, there are buttons for `Next`, `10`, `100`, `1,000`, and `Stop`. A red dashed box highlights the `Next` button and the `Stop` button. A red arrow points from the `Run!` button at the bottom right to the `Next` button, with the text `(ctrl-enter)` next to it. Another red arrow points from the `Next` button to the `Stop` button, with the text `(; or enter)` next to it.

Example Program



The image shows the SWISH Prolog IDE interface. The left pane contains a Prolog program with the following code:

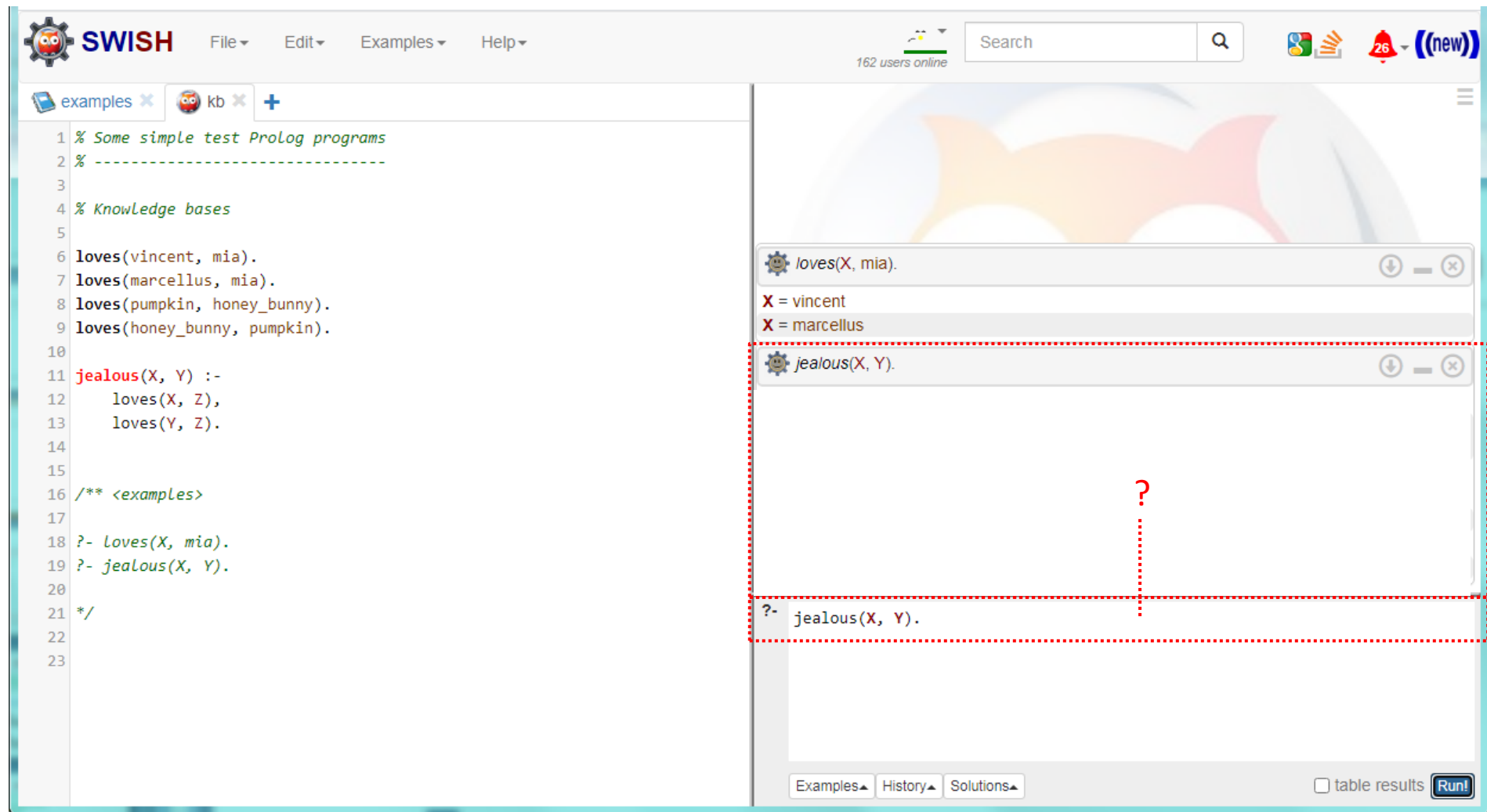
```
1 % Some simple test Prolog programs
2 % -----
3
4 % Knowledge bases
5
6 loves(vincent, mia).
7 loves(marcellus, mia).
8 loves(pumpkin, honey_bunny).
9 loves(honey_bunny, pumpkin).
10
11 jealous(X, Y) :-
12     loves(X, Z),
13     loves(Y, Z).
14
15
16 /** <examples>
17
18 ?- loves(X, mia).
19 ?-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

The right pane displays a large owl illustration. Below it, a query window shows the query `loves(X, mia).` and its solutions:

```
X = vincent
X = marcellus
```

At the bottom of the right pane, there is a query input field with `?- loves(X, mia).` and buttons for `Examples`, `History`, `Solutions`, `table results`, and `Run!`.

Example Program

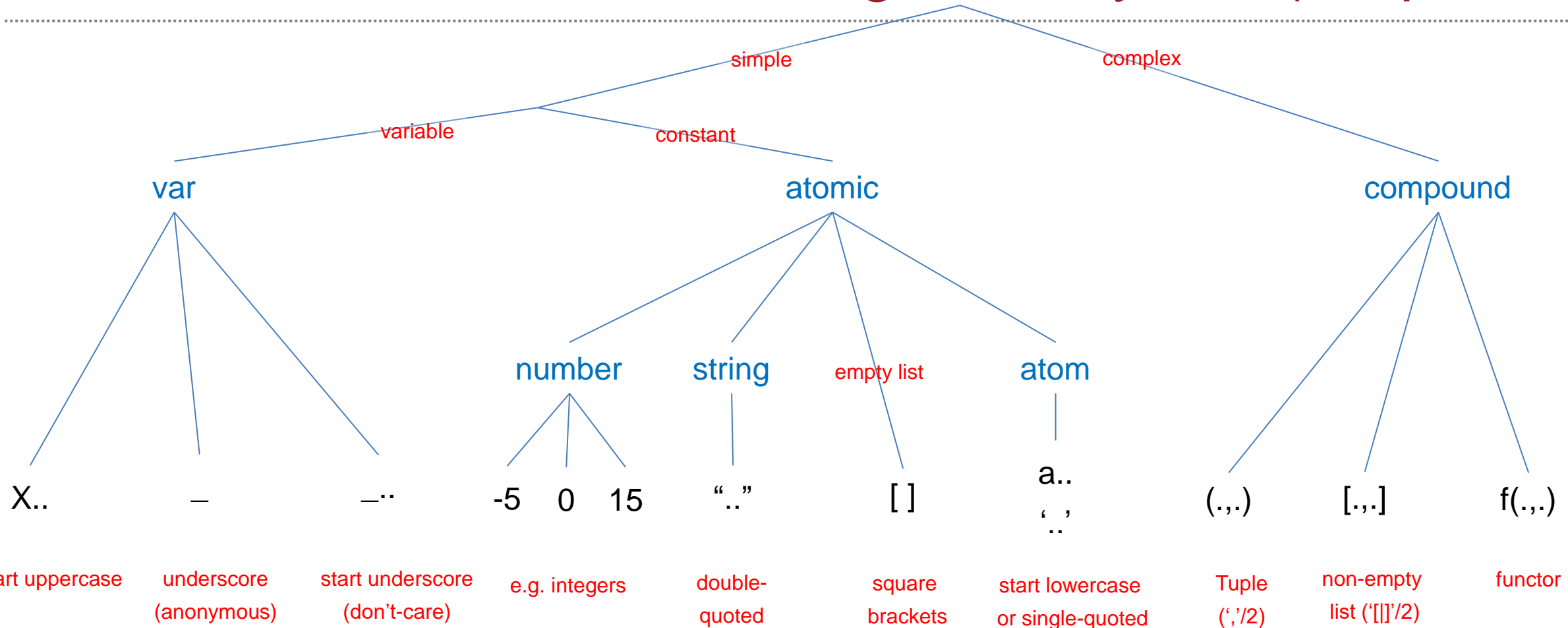


The screenshot shows the SWISH Prolog IDE interface. The left pane contains a Prolog program with the following code:

```
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, Z),
13     loves(Y, Z).
14
15
16 /** <examples>
17
18 ?- loves(X, mia).
19 ?- jealous(X, Y).
20
21 */
22
23
```

The right pane shows the execution results. The first query is `loves(X, mia).`, which has two solutions: `X = vincent` and `X = marcellus`. The second query is `jealous(X, Y).`, which is currently showing a red question mark, indicating that no solutions have been found yet. The bottom of the interface includes buttons for "Examples", "History", "Solutions", a checkbox for "table results", and a "Run!" button.

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: @< , @> , @=< , @>= , == , \==

- 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

| | | | |
|------------|--|-----------------|-----|
| Chapter 1 | Facts, Rules, and Queries | wk1 | |
| Chapter 2 | Unification and Proof Search | | wk3 |
| Chapter 3 | Recursion | wk2 | |
| Chapter 4 | Lists | wk2 | |
| Chapter 5 | Arithmetic | wk2 | |
| Chapter 6 | More Lists | | wk3 |
| Chapter 7 | Definite Clause Grammars | not examinable! | |
| Chapter 8 | More Definite Clause Grammars | not examinable! | |
| Chapter 9 | A Closer Look at Terms | wk1+ | |
| Chapter 10 | Cuts and Negation | wk2 | |
| Chapter 11 | Database Manipulation and Collecting Solutions | wk2 | |
| Chapter 12 | Working With Files | not examinable! | |

Optional self-study topics (courtesy of Seth Bullock):

- Week 1: Alan Turing and The Turing Test ('50s)
- Week 2: Newell & Simon's Physical Symbol Systems Hypothesis ('70s)
- Week 3: The Chinese Room ('80s)
- Week 4: Brooks and “Nouvelle AI” ('90s)

Thank you