Lesson #1: Getting Acquainted with Prolog

What's It All About?

1. The first thing you will need to do in order that you can do some Prolog programming is to download a Prolog interpreter. The first part of this lesson points you in the right direction to do just that.
2. Then, you will want to get acquainted with the nature of Prolog. Two sample sessions will serve to help you make this acquaintance, the first of which illustrates how to add bits of knowledge dynamically to a knowledge base, the second of which illustrates how to statically establish a knowledge base, by loading the knowledge base from a file.
3. As a side effect of studying these two "Prolog programs", you will gain some understanding of *terms* and *facts* and *rules* and `queries`.

The software

You will need a Prolog interpreter in order to do some Prolog programming. The SWI Prolog interpreter is **highly recommended**! You can find the software here: `https://www.swi-prolog.org/`

You will also need a good text editor with which to create your Prolog source code. I like emacs, but you might like something else.

First Prolog Experience: Simple Interaction (Three Colors)

The following session features the establishment of a **relation** consisting of three **facts**. The relation represents the facts that blue is a color, that red is a color, and that green is a color. The primitive `assert` can, as is seen three times in the session, be used to place a fact in the knowledge base.

The Session

```
bash-3.2$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.2.5)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- color(blue).
ERROR: toplevel: Undefined procedure: color/1 (DWIM could not correct goal)
?- assert(color(blue)).
```

```
true.

?- color(blue).
true.

?- assert(color(red)).
true.

?- assert(color(green)).
true.

?- color(red).
true.

?- color(green).
true.

?- color(yellow).
false.

?- listing(color).
:- dynamic color/1.

color(blue).
color(red).
color(green).

true.

?- halt.
bash-3.2$
```

Second Prolog Experience: Consulting a KB in a File (Six Colors)

A Prolog **knowledge base** is placed in a file called `colors.pro`. This knowledge base consists of 3 **relations**. The `primary` relation consists of 3 facts, and represents primary colors. The `secondary` relation consists of 3 facts, and represents secondary colors. The `color` relation consists of two rules and represents both the primary colors and the secondary colors.

The accompanying session incorporates a number of "closed" queries, in which the machine seeks merely to establish that a fact is or is not in the knowledge base. The session also incorporates a number of "open" queries, in which information is sought from the knowledge base through the use of variables.

What is with the semicolon in the session? If the system has met with some success, and there are alternate routes that it might have taken to find success, it will wait for instruction from you with regard to what to do next. If you type in a semicolon, it will try to succeed in an alternate way. If you type the "Enter" key, it will simply return you to the top-level prompt.

KB: colors.pro

```
% ----------------------------------------------------------------------------
% File: colors.pro
% Line: Six color facts, structured into primaries and secondaries

% ----------------------------------------------------------------------------
% primary(P) :: P is a primary color

primary(blue).
primary(red).
primary(yellow).

% ----------------------------------------------------------------------------
% secondary(S) :: S is a secondary color

secondary(green).
secondary(orange).
secondary(purple).

% ----------------------------------------------------------------------------
% color(C) :: C is a color

color(C) :- primary(C).
color(C) :- secondary(C).
```

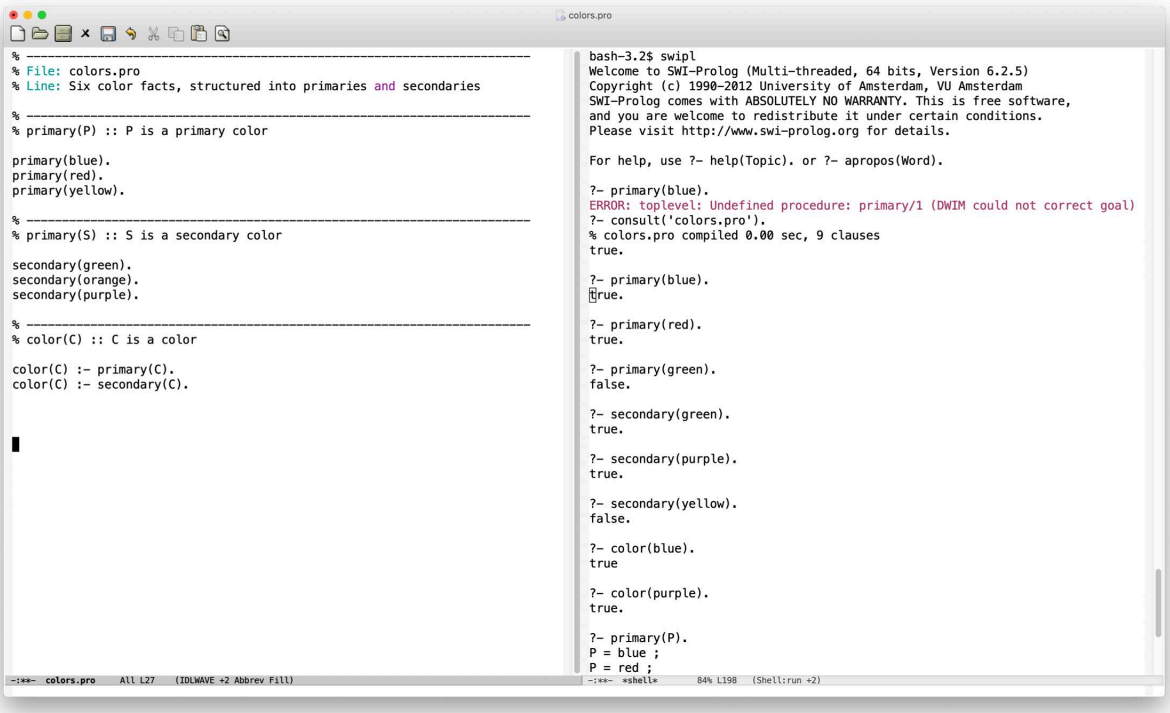
The Session

```
bash-3.2$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.2.5)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- primary(blue).
ERROR: toplevel: Undefined procedure: primary/1 (DWIM could not correct goal)
?- consult('colors.pro').
% colors.pro compiled 0.00 sec, 9 clauses
true.

?- primary(blue).
true.

?- primary(red).
true.
```

```
?- primary(green).
false.

?- secondary(green).
true.

?- secondary(purple).
true.

?- secondary(yellow).
false.

?- color(blue).
true

?- color(purple).
true.

?- primary(P).
P = blue ;
P = red ;
P = yellow.

?- secondary(S).
S = green ;
S = orange ;
S = purple.

?- color(C).
C = blue ;
C = red ;
C = yellow ;
C = green ;
C = orange ;
C = purple.

?- listing(primary).
primary(blue).
primary(red).
primary(yellow).

true.

?- listing(secondary).
secondary(green).
secondary(orange).
secondary(purple).

true.

?- listing(color).
color(A) :-
primary(A).
color(A) :-
```

```
secondary(A).

true.

?- halt.
bash-3.2$
```

Snapshot of Emacs with a KB and a Shell



colors.pro

```
% ------------------------------------------------------------------------
% File: colors.pro
% Line: Six color facts, structured into primaries and secondaries

% ------------------------------------------------------------------------
% primary(P) :: P is a primary color

primary(blue).
primary(red).
primary(yellow).

% ------------------------------------------------------------------------
% primary(S) :: S is a secondary color

secondary(green).
secondary(orange).
secondary(purple).

% ------------------------------------------------------------------------
% color(C) :: C is a color

color(C) :- primary(C).
color(C) :- secondary(C).
```

```
bash-3.2$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.2.5)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- primary(blue).
ERROR: toplevel: Undefined procedure: primary/1 (DWIM could not correct goal)
?- consult('colors.pro').
% colors.pro compiled 0.00 sec, 9 clauses
true.

?- primary(blue).
true.

?- primary(red).
true.

?- primary(green).
false.

?- secondary(green).
true.

?- secondary(purple).
true.

?- secondary(yellow).
false.

?- color(blue).
true

?- color(purple).
true.

?- primary(P).
P = blue ;
P = red ;
```

```
-:**-  colors.pro     All L27    (IDLWAVE +2 Abbrev Fill)
-:**-  *shell*        84% L198   (Shell:run +2)
```