

pLisp User Manual

1. Introduction

pLisp is a Linux-based integrated development environment (IDE) for Lisp. While it aims to be a friendly Lisp development system for beginners, its feature-set is comprehensive enough to address the needs of a small-to-medium sized Lisp project. Please refer to the README file for more details. This document is the user manual for pLisp. It is not a language reference manual. Email me at rajesh.jayaprakash@gmail.com for comments and suggestions.

2. Getting pLisp

You can get pLisp from [here](#). Right now you can either use git to check out the code or download a single zip file.

3. Installing pLisp

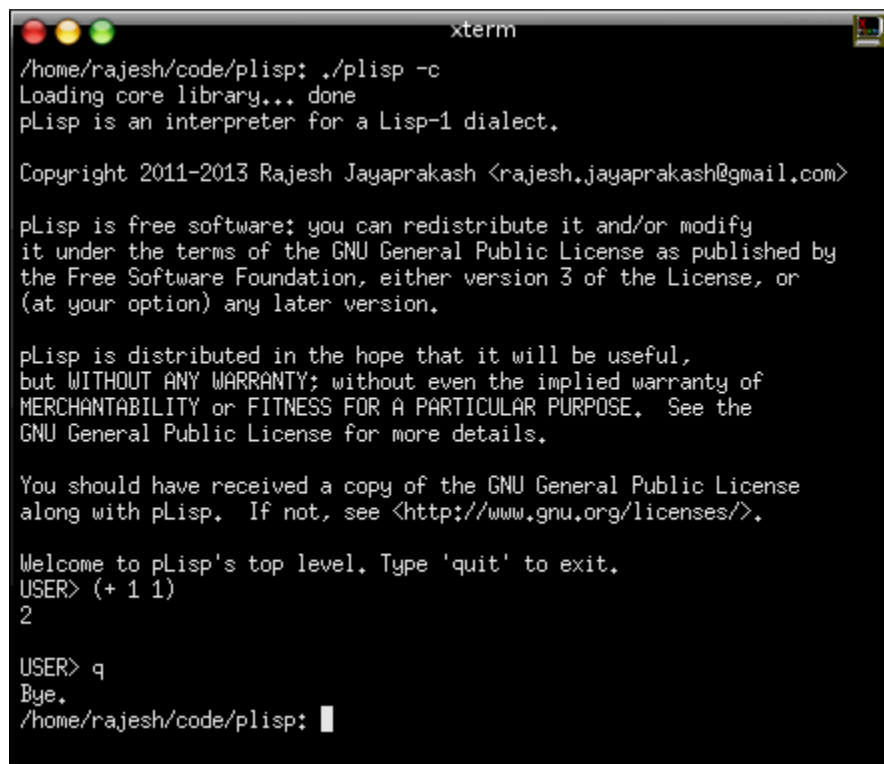
To install pLisp, simply type 'make' after navigating to the directory to which you downloaded the pLisp files. Before you do this, however, the following dependencies need to be taken care of (in addition to having GCC and make in your system, of course):

- a. GTK 3.0 development libraries (libgtk-3-dev). These files can be installed using the package manager that comes with your Linux distro.
- b. The [Tiny C Compiler](#) development files (libtcc.h and libtcc.a). You can either install them using the standard installation procedure for TCC, or simply copy these two files and include their locations via the -I and -L flags in the makefile supplied with pLisp.
- c. [libffi](#). You can install this by downloading the tar.gz file and invoking the standard 'configure', 'make' and 'make install' commands.

4. Invoking pLisp

Alright, so you've installed pLisp by successfully following the steps in the previous section. The next step is to start pLisp. You have three options while starting pLisp:

- a. **plisp -c**: Run pLisp in command-line mode, i.e., without any of the user interface bells and whistles. You will be presented with just a prompt, and you have to type your Lisp expressions at this prompt (Figure 1). The results of evaluating the expression will be displayed below the prompt (more later in Section 11 on what the prompt signifies). After you're done with all the expressions, type 'q' to exit pLisp. Then do a hundred push-ups and take a cold shower.
- b. **plisp -e <expression>**: Use pLisp to just evaluate a single expression, print the results and

A terminal window titled 'xterm' showing the execution of the pLisp script. The prompt is '/home/rajesh/code/plisp: ./plisp -c'. The output includes: 'Loading core library... done', 'pLisp is an interpreter for a Lisp-1 dialect.', copyright information for Rajesh Jayaprakash (2011-2013), the GNU General Public License notice, and a welcome message. The user enters '(+ 1 1)' and the output is '2'. The user then enters 'q' and the output is 'Bye.'.

```
xterm
/home/rajesh/code/plisp: ./plisp -c
Loading core library... done
pLisp is an interpreter for a Lisp-1 dialect.

Copyright 2011-2013 Rajesh Jayaprakash <rajesh.jayaprakash@gmail.com>

pLisp is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

pLisp is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

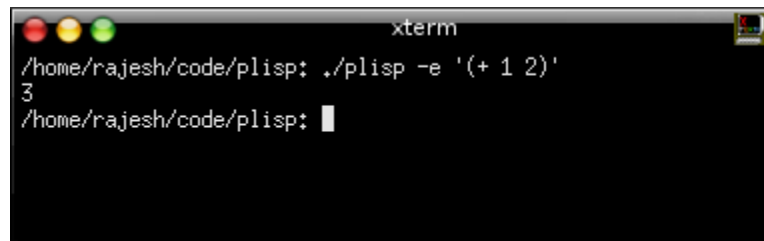
You should have received a copy of the GNU General Public License
along with pLisp. If not, see <http://www.gnu.org/licenses/>.

Welcome to pLisp's top level. Type 'quit' to exit.
USER> (+ 1 1)
2

USER> q
Bye.
/home/rajesh/code/plisp: █
```

Figure 1

quit (see Figure 2). This is useful for both running the programs that you have created using pLisp, and for using pLisp as a calculator [**plisp -e '(+ 1 2)'**]. Note that the expression has to be enclosed in single or double quotes.

A terminal window titled 'xterm' showing the execution of the pLisp script with the -e option. The prompt is '/home/rajesh/code/plisp: ./plisp -e '(+ 1 2)'. The output is '3'. The prompt returns to '/home/rajesh/code/plisp: █'.

```
xterm
/home/rajesh/code/plisp: ./plisp -e '(+ 1 2)'
3
/home/rajesh/code/plisp: █
```

Figure 2

c. **plisp -i <image file>**: Start pLisp by loading an existing image. An image is a file on your hard disk that stores a snapshot of your pLisp system – all the functions, variables, and other objects that you created in the course of your development. Image-based development is not only an incredibly powerful way to build code, but also has other benefits (easier debugging and persistent objects, to name a few). Also, the cool thing about pLisp images is that they remember the details of the UI elements, so if you open an image, you will be presented with the same windows (even the Profiler and the Debugger windows) that were open when you saved the image last.

These options can be used alone or in combination: you can start with an image in command line mode, or evaluate a single expression on an image and quit. But you cannot use both the '-c' and '-e' options together: if you're going to quit after evaluating a single expression, it doesn't

matter whether you are in command-line mode or GUI mode.

The order of these options also does not matter; for example, you can either say **plisp -e <expression> -i <image file>** or **plisp -i <image file> -e <expression>**.

5. The Transcript Window

So you have invoked pLisp in GUI mode. You will be presented with the main window in the pLisp UI, namely the Transcript window (Figure 3).

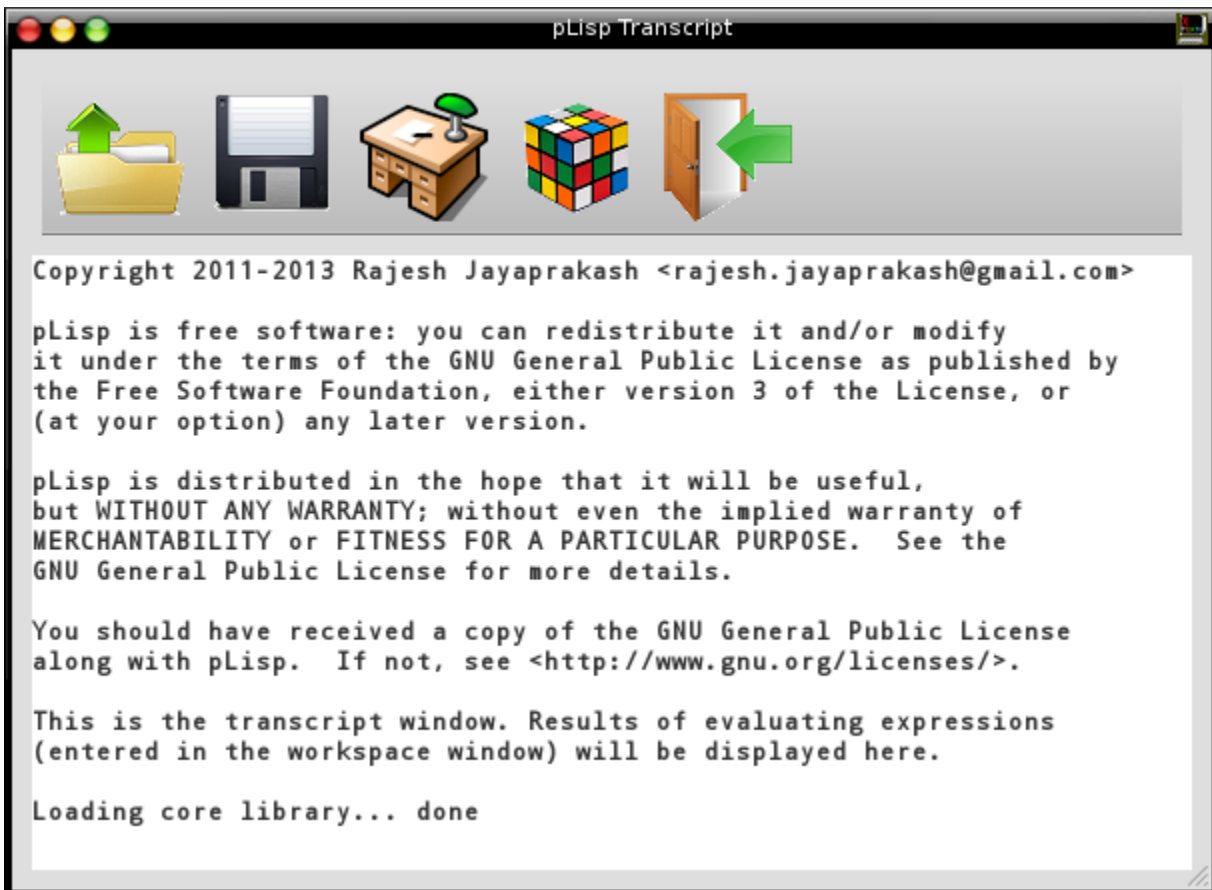


Figure 3

The Transcript window displays the results of evaluating expressions entered in the Workspace window. In addition to this, it's also the primary means of controlling pLisp, through the toolbar actions. The toolbar has five buttons:



Click on this button if pLisp has been invoked without an image file and you would like to load an image file, or if you would like to switch to another image file. Keyboard shortcut: Control-L



Click on this button either to save an already loaded image file or to create a fresh image. Keyboard shortcut: Control-S.



This button opens the Workspace window or, if it's already open, brings it to the front. Keyboard shortcut: F7.



This button opens the System Browser window or, if it's already open, brings it to the front. Keyboard shortcut: F9.



Exits pLisp. If you have made changes to an image, you will be prompted to save the changes. Keyboard shortcut: Control-W.

6. The Workspace Window

As mentioned in the previous section, pressing F7 or clicking on the Workspace toolbar button in the Transcript window brings up the Workspace window (Figure 4).

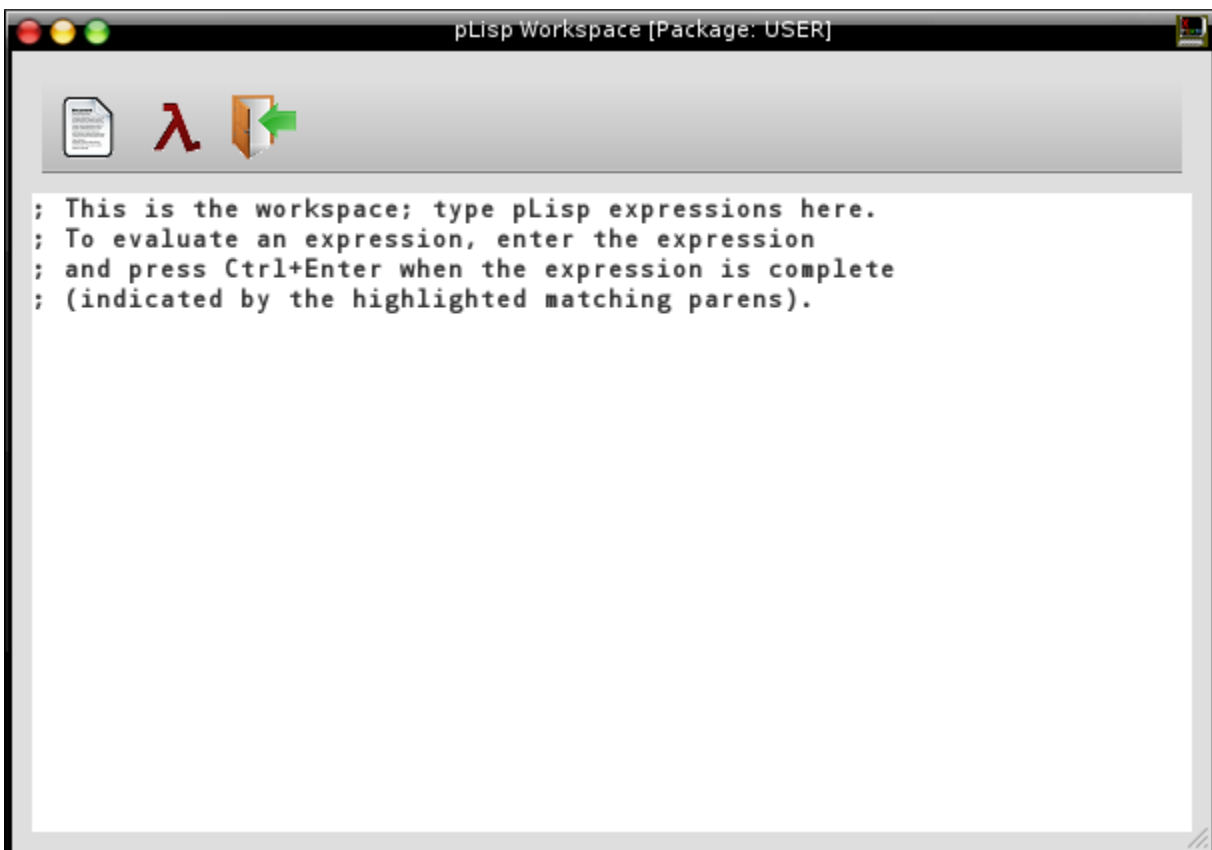


Figure 4

As the text in the window helpfully indicates, type the pLisp expressions here that you would like evaluated. The result will appear in the Transcript window if all goes well. The Workspace window has three toolbar buttons that can be used to control its behaviour:



Click on this button to load a file containing pLisp source expressions. Useful if you would like to feed multiple (previously saved) expressions in one shot. Keyboard shortcut: Control-O.



Click on this button to evaluate a selected expression entered into the workspace. Keyboard shortcut: Control-Enter.



Click on this button to close the Workspace Window. Keyboard shortcut: Control-W.

7. System Browser Window

Pressing F9 or clicking on the System Browser toolbar button in the Transcript window brings up the System Browser Window (Figure 5).

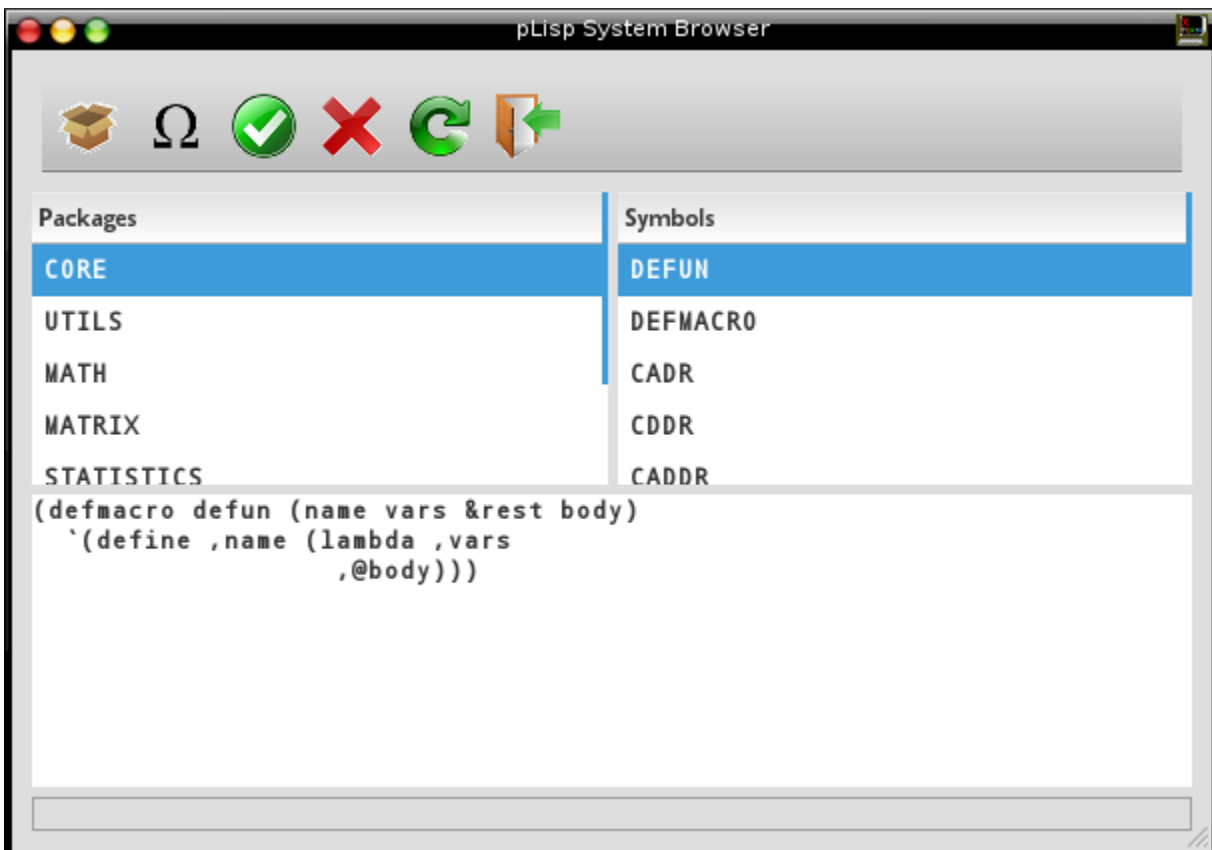


Figure 5

The System browser has three panels: the Packages panel, the Symbols panel, and the code panel. The Packages panel lists all the packages defined in the system (see Section 11). The Symbols panel lists all the symbols belonging to the selected package. Finally, the code panel displays the code corresponding to the selected symbol if it's a closure or a macro, or the value (integer, float, character, etc.) that is bound to the symbol.

The System Browser has these six toolbar buttons:



Click on this button to create a new package.
Keyboard shortcut: Control-K.



Click on this button to create a new symbol in the selected package. Keyboard shortcut: Control-N.



Click on this button to accept the changes made in the code panel. Keyboard shortcut: Control-S.



Click on this button to delete the selected symbol.
Keyboard shortcut: Control-X.



Click on this button to refresh the contents of the System Browser (for example, if you've made changes by typing expressions in the Workspace window and would like to sync the System Browser with these changes). Keyboard shortcut: F5.



Click on the button to close the System Browser window. Keyboard shortcut: Control-W.

8. Debugger Window

Debugging pLisp code entails placing '(BREAK)' statements judiciously in your expressions and inspecting the state of the system by means of the Debugger window. The Debugger window displays the current calling stack and the variables associated with each frame in the stack.

A (contrived) example will clarify things.

```
(let ((x 0))  
  (print x)  
  (break)  
  (incf x)  
  (break))
```

This code creates a local variable `x`, initialized to zero, prints it, then increments it. There are

break statements after the print and the increment. Evaluating this expression brings up the Debugger window twice (Figures 6 and 7) , once for each break statement (pressing the Resume button in the window resumes the evaluation). Pressing the Abort button aborts the evaluation.

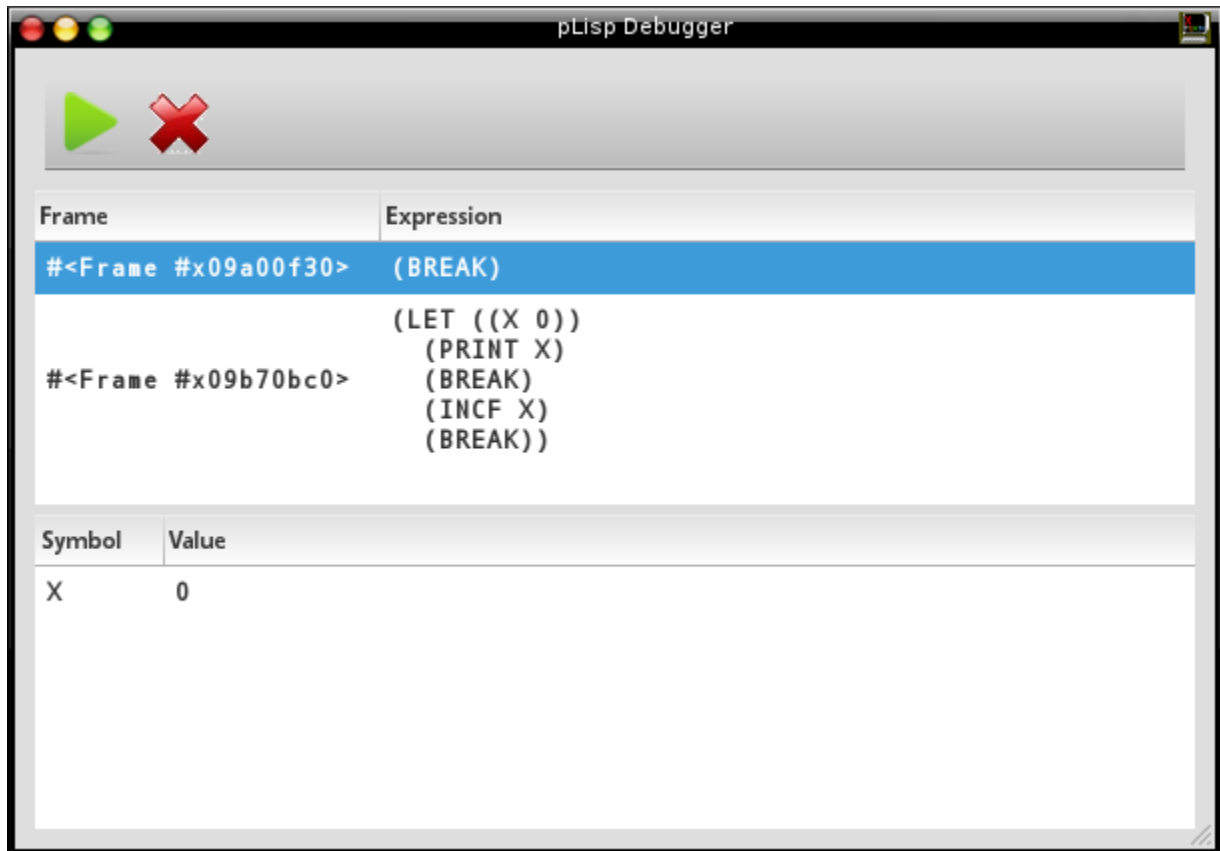


Figure 6

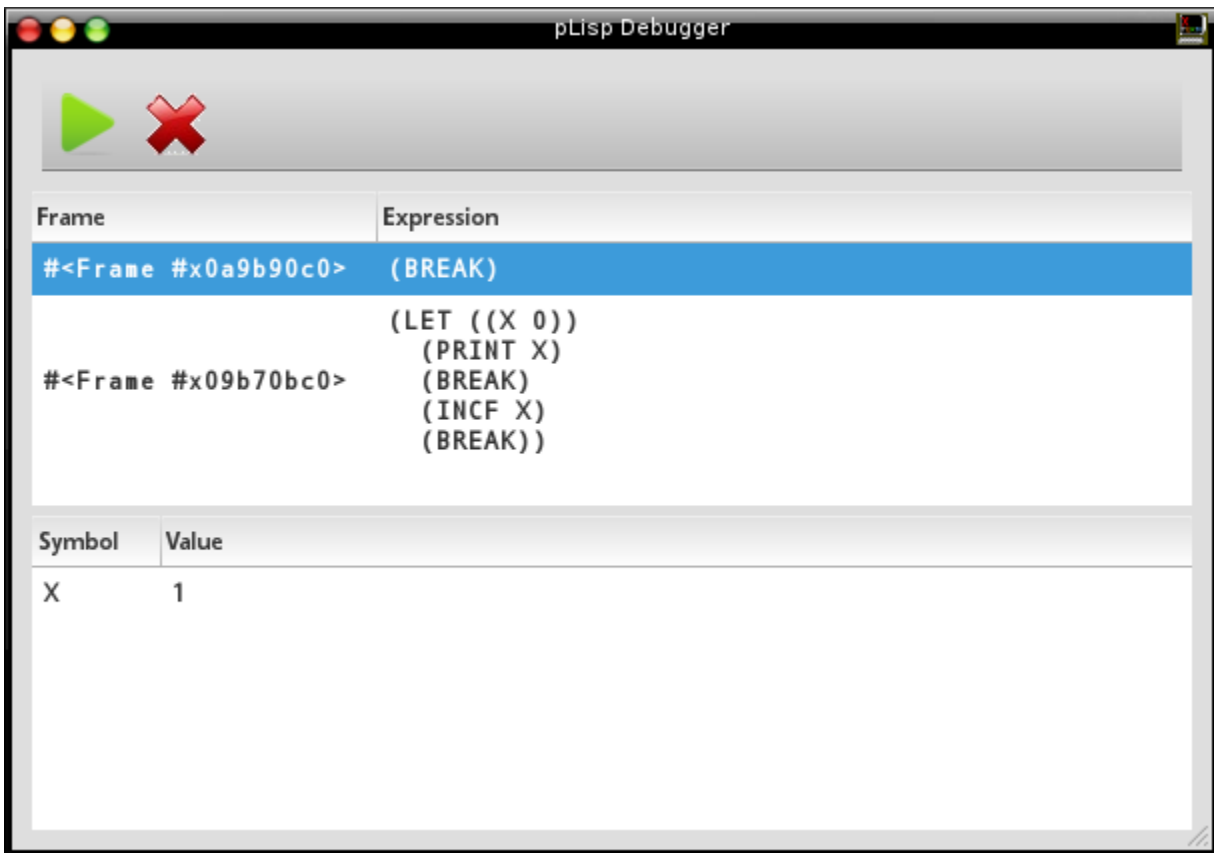


Figure 7

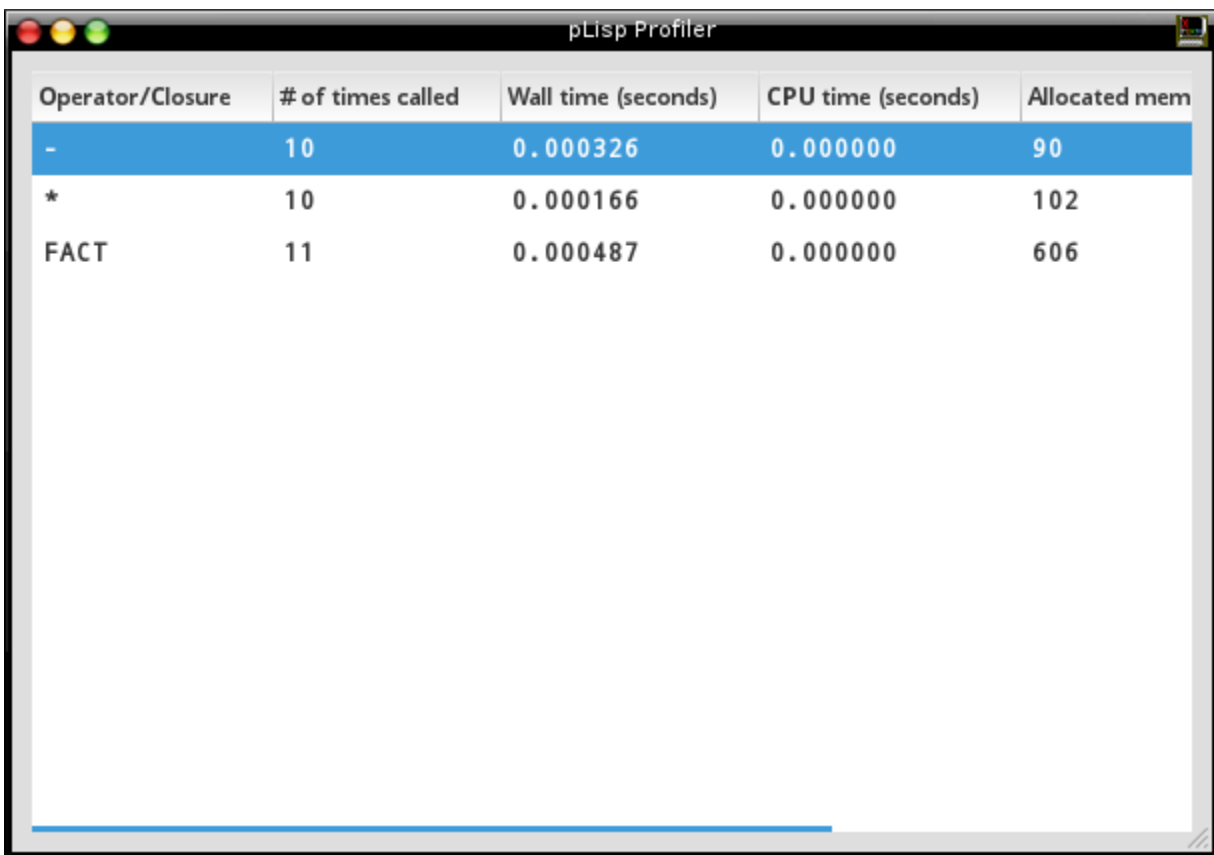
Astute readers would have noticed the different values of the symbol X in the two figures, corresponding to the execution of the two break statements.

9. The Profiler Window

pLisp has a special operator called 'profile' that can be used to investigate the resource utilization (time and memory) of expressions. Invoking this operator on an expression brings up the Profiler window that displays these statistics for each operator called (directly or indirectly) by this expression:

- Number of times called
- Elapsed wall time
- Elapsed CPU time
- Number of bytes allocated
- Number of bytes deallocated

The cumulative statistics for the whole expression are also displayed in the Transcript window. Figures 8 and 9 illustrate this with an example.



Operator/Closure	# of times called	Wall time (seconds)	CPU time (seconds)	Allocated mem
-	10	0.000326	0.000000	90
*	10	0.000166	0.000000	102
FACT	11	0.000487	0.000000	606

Figure 8

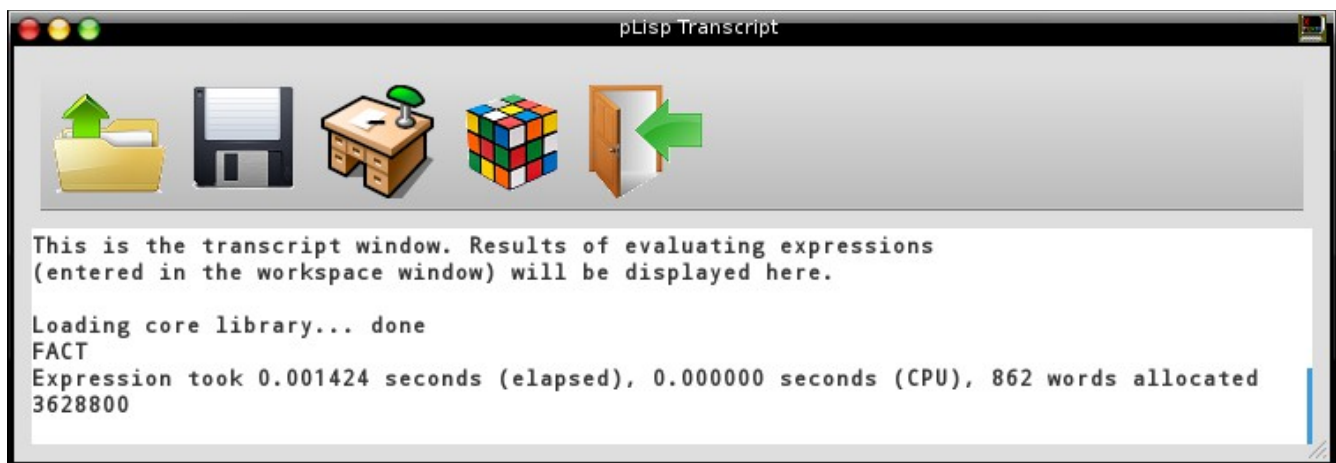


Figure 9

10. Exceptions

The Debugger window is also invoked automatically whenever your code hits an exception. The current stack is displayed, with the expression causing the exception on top; as before, the values of the variables belonging to each frame are also displayed (Figures 10, 11 and 12).

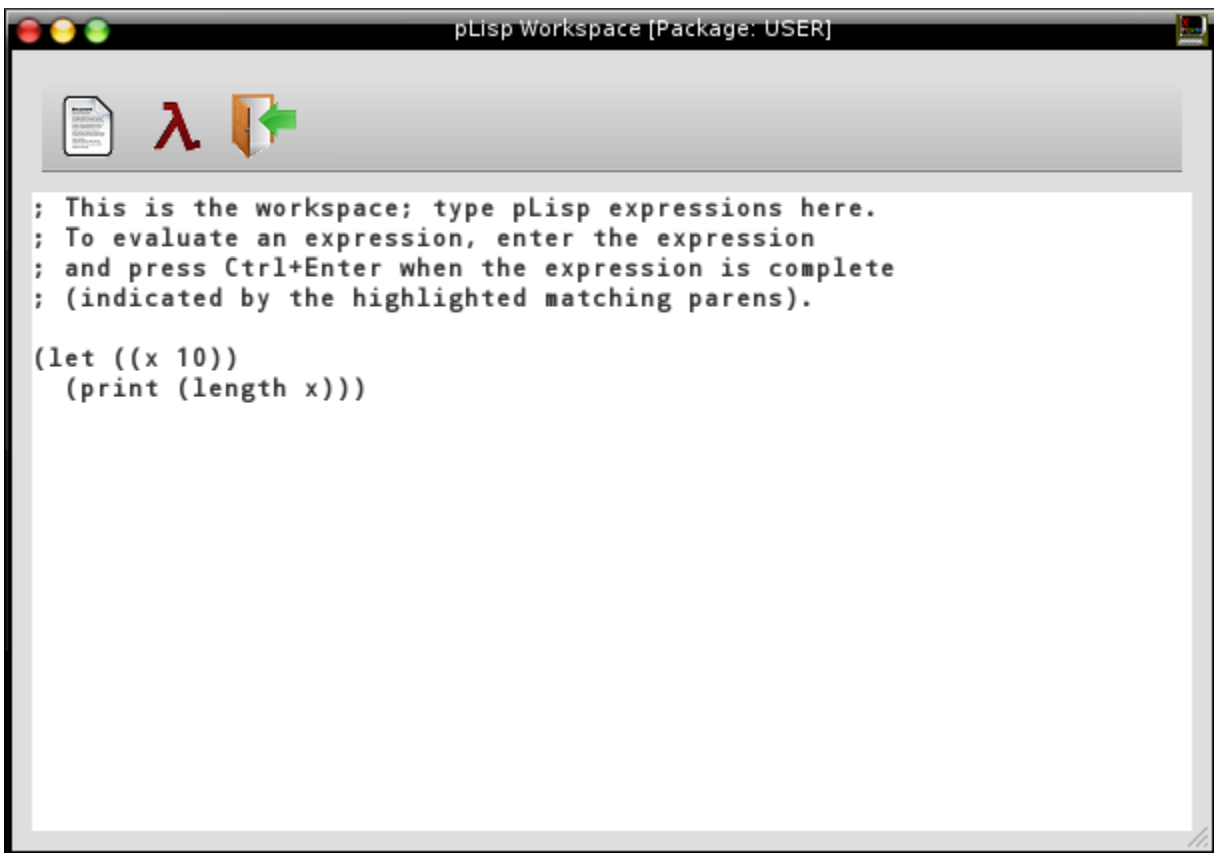


Figure 10

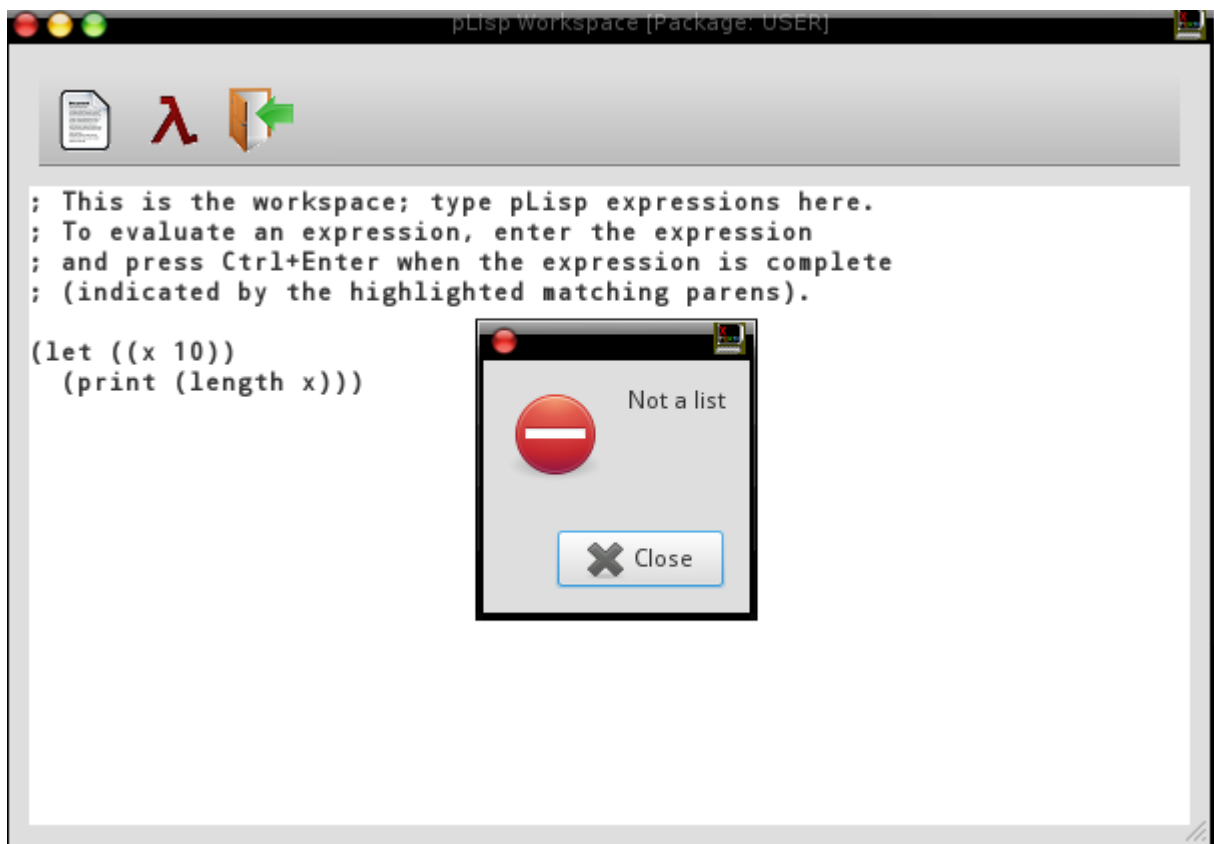


Figure 11

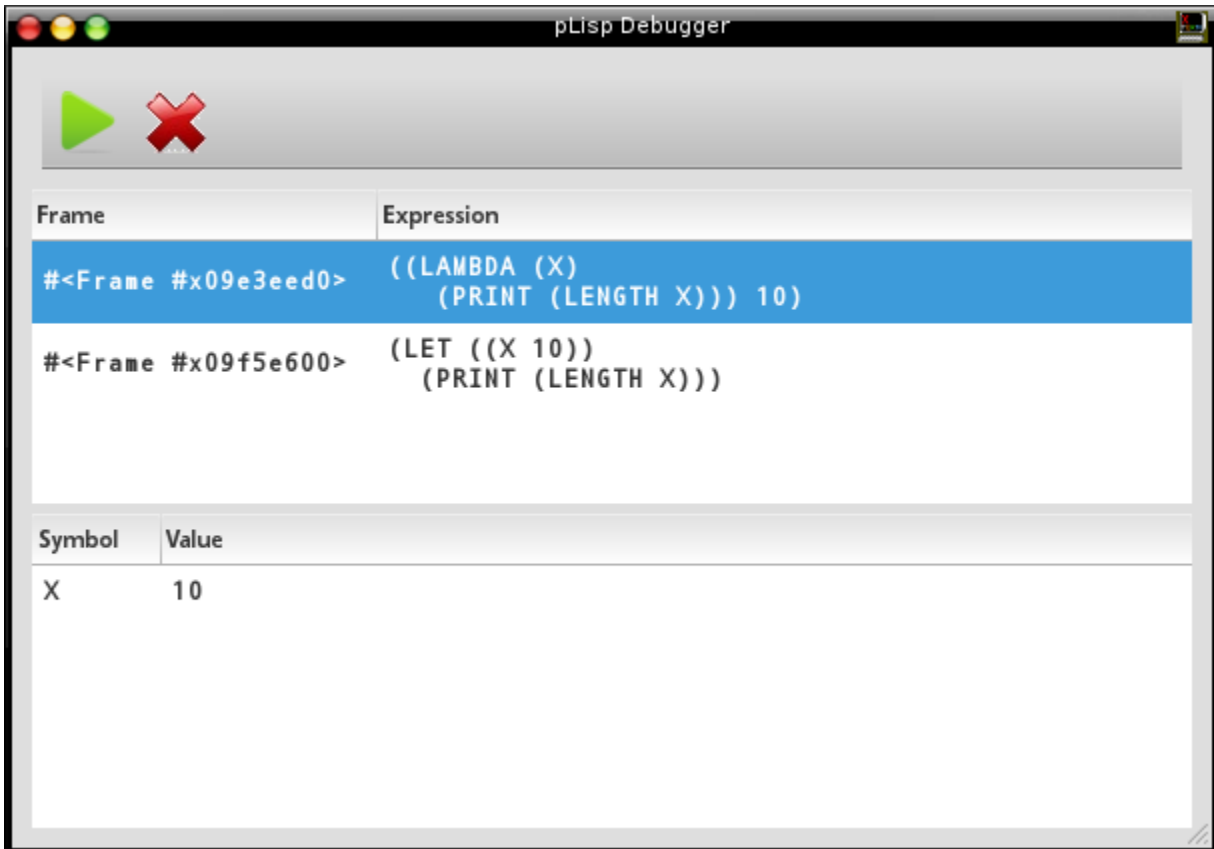


Figure 12

There are two frames displayed because the LET form is actually a macro that converts its body into a lambda expression.

11. A Note About Packages

pLisp objects are categorized into namespaces called packages. The CORE package contains all the predefined special operators (CAR, CDR, and so on), and it is not permitted to modify this package by adding or modifying any symbols in it. A package called USER is created by default, and the user is expected to define their code in this package. Additional packages can be created if needed, of course.

The evaluation of any expression is always in the context of the current package, indicated in the title of the Workspace window in GUI mode and in the prompt in the console mode.