

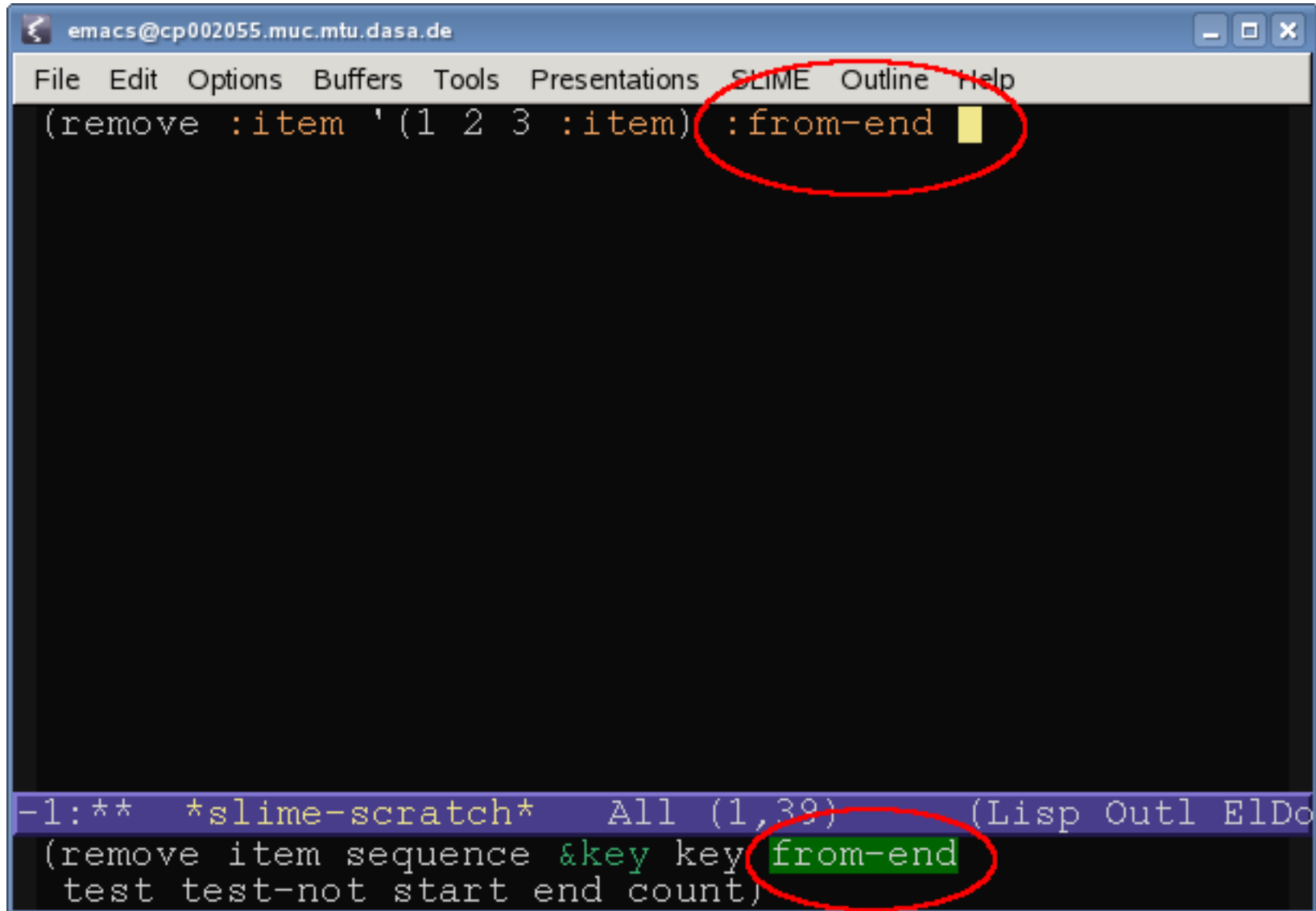


Tobias-Christian Rittweiler

September 9, 2009

Autodoc

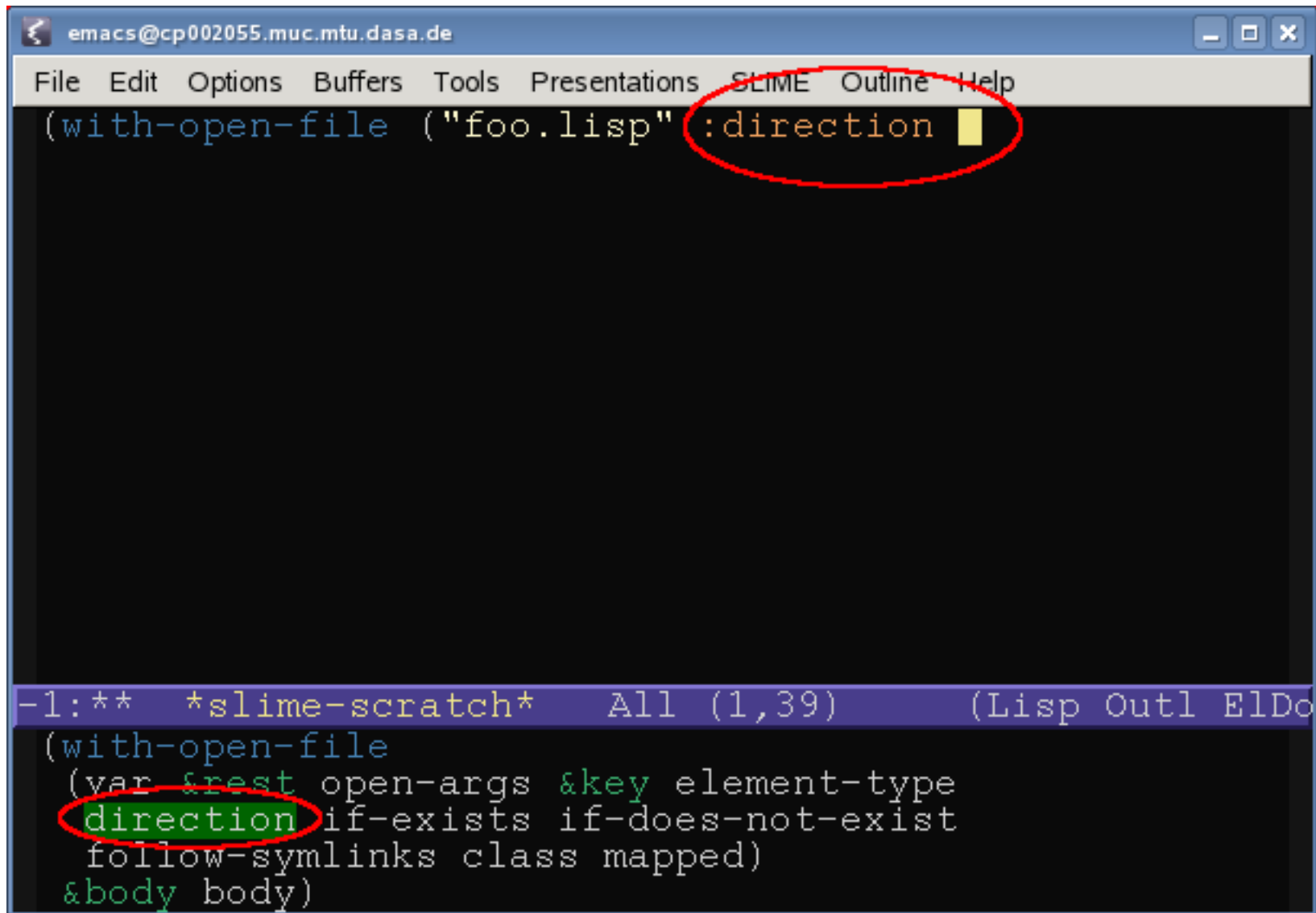
- highlighting of &KEY parameters



The screenshot shows an Emacs window with the title bar "emacs@cp002055.muc.mtu.dasa.de". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Presentations", "SLIME", "Outline", and "Help". The main text area contains the code `(remove :item '(1 2 3 :item) :from-end` with a cursor at the end. A red circle highlights the `:from-end` parameter. At the bottom, a status bar shows `-1:** *slime-scratch* All (1,39) (Lisp Outl ElDo`. Below the status bar, the autodoc documentation for `remove` is displayed: `(remove item sequence &key key from-end` and `test test-not start end count)`. A red circle highlights the `from-end` parameter in the documentation.

Autodoc (cont..)

- highlighting of parameters in nested arglists



The screenshot shows an Emacs window with the title bar "emacs@cp002055.muc.mtu.dasa.de". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Presentations", "SLIME", "Outline", and "Help". The main text area displays the code snippet `(with-open-file ("foo.lisp" :direction` with a cursor at the end of the line. A red circle highlights the `:direction` parameter. At the bottom, a status bar shows `-1:** *slime-scratch* All (1,39) (Lisp Outl ElDo`. Below the status bar, the autodoc documentation for `with-open-file` is shown, with the `direction` parameter highlighted in green and circled in red.

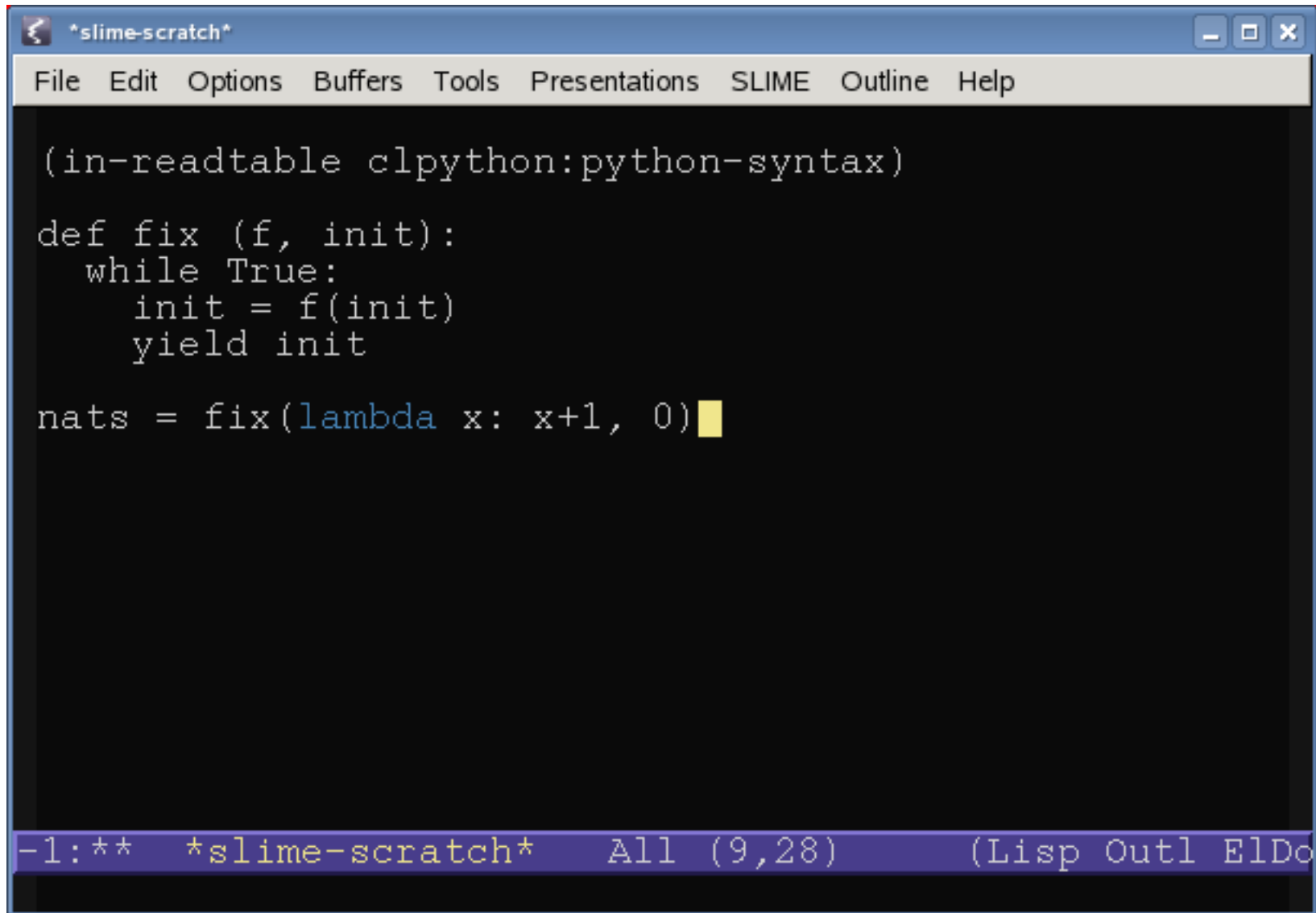
```
(with-open-file ("foo.lisp" :direction
```

```
-1:** *slime-scratch* All (1,39) (Lisp Outl ElDo
```

```
(with-open-file  
  (var &rest open-args &key element-type  
  direction if-exists if-does-not-exist  
  follow-symlinks class mapped)  
  &body body)
```

Named-Readtables

- package-like namespace for readtables



The screenshot shows a SLIME window titled `*slime-scratch*` with a menu bar containing `File Edit Options Buffers Tools Presentations SLIME Outline Help`. The main text area contains the following Python code:

```
(in-readtable clpython:python-syntax)

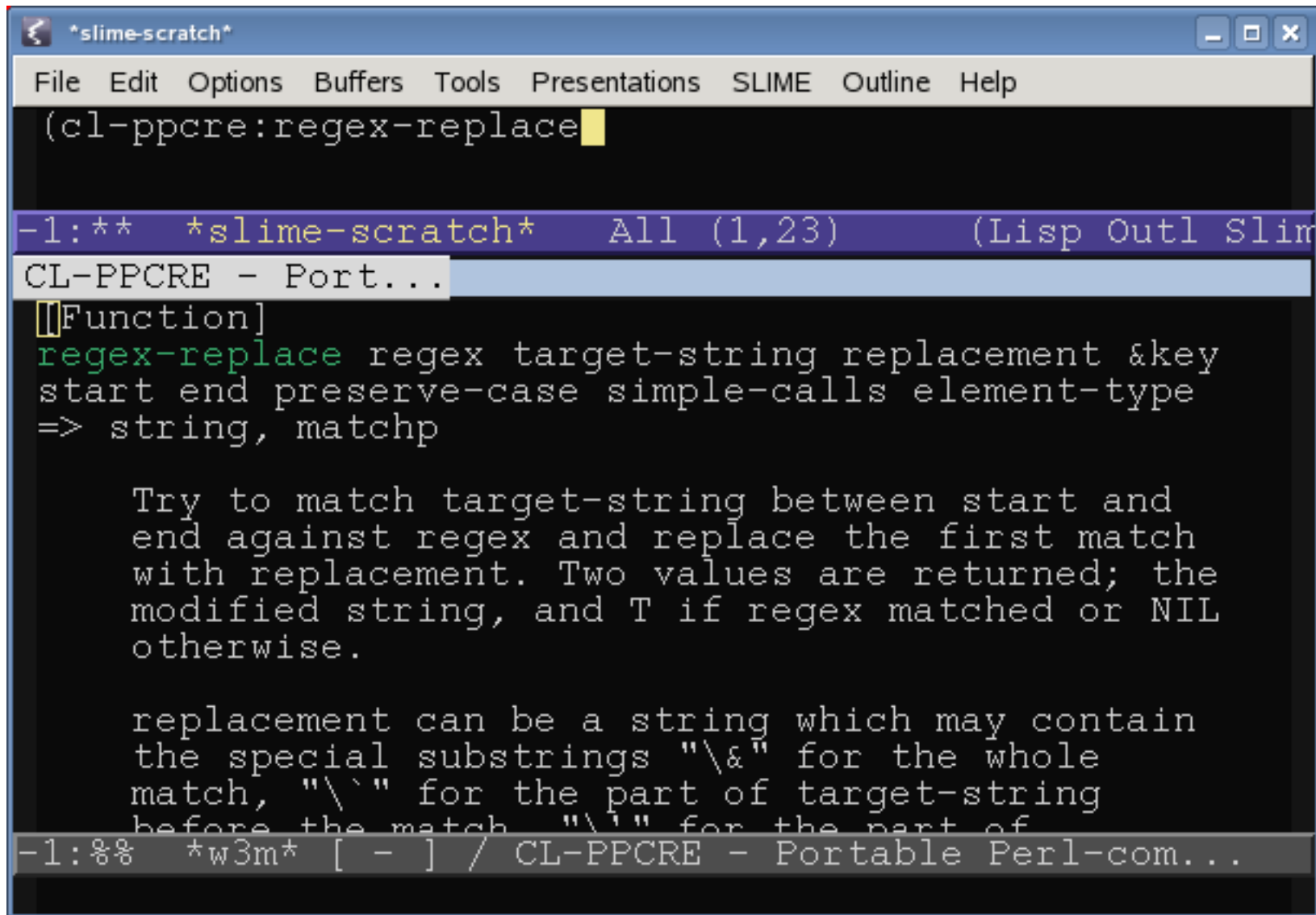
def fix (f, init):
    while True:
        init = f(init)
        yield init

nats = fix(lambda x: x+1, 0)█
```

The status bar at the bottom of the window displays `-1:** *slime-scratch* All (9,28) (Lisp Outl ElDo`.

Hyperdoc

- C-c C-d h on steroids!



The screenshot shows a SLIME window titled `*slime-scratch*` with a menu bar (File, Edit, Options, Buffers, Tools, Presentations, SLIME, Outline, Help). The main text area contains the code `(cl-ppcre:regex-replace` with a cursor. A hyperdoc is displayed below, with a purple header bar containing `-1:** *slime-scratch* All (1,23) (Lisp Outl Slim`. The main content of the hyperdoc is for the `CL-PPCRE - Port...` package, showing the `regex-replace` function signature and a detailed description of its behavior and arguments. A second hyperdoc bar is visible at the bottom with `-1:%% *w3m* [-] / CL-PPCRE - Portable Perl-com...`.

```
(cl-ppcre:regex-replace
```

`-1:** *slime-scratch* All (1,23) (Lisp Outl Slim`

`CL-PPCRE - Port...`

`[[Function]`
`regex-replace` regex target-string replacement &key
start end preserve-case simple-calls element-type
=> string, matchp

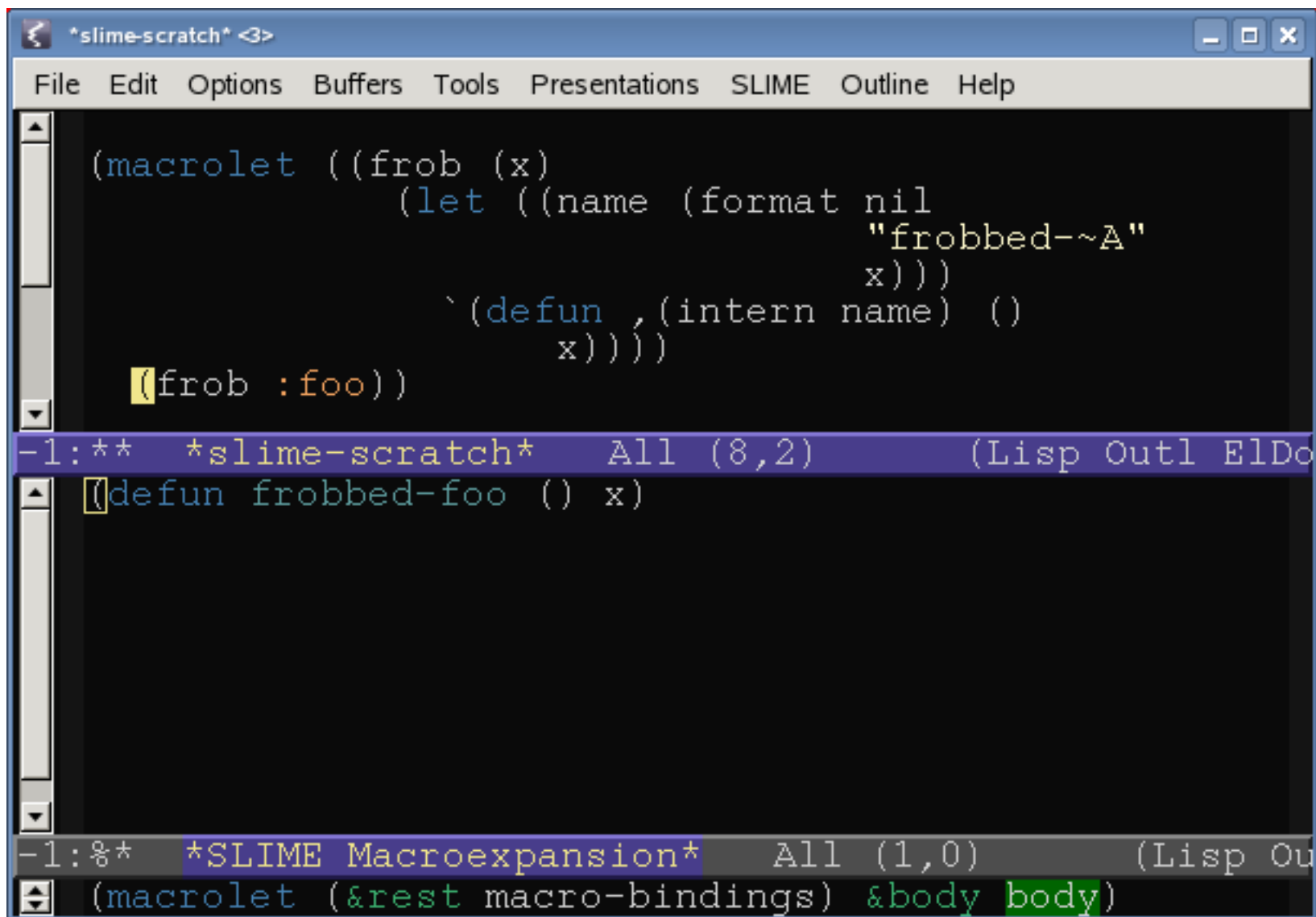
Try to match target-string between start and end against regex and replace the first match with replacement. Two values are returned; the modified string, and T if regex matched or NIL otherwise.

replacement can be a string which may contain the special substrings `"\&"` for the whole match, `"\`"` for the part of target-string before the match, `"\'"` for the part of

`-1:%% *w3m* [-] / CL-PPCRE - Portable Perl-com...`

Fancy Macroexpansion

- expanding in right lexical environment (think of MACROLET)
- retaining expansion in Lisp image, modified in-place.



```
^slime-scratch* <3>
File Edit Options Buffers Tools Presentations SLIME Outline Help

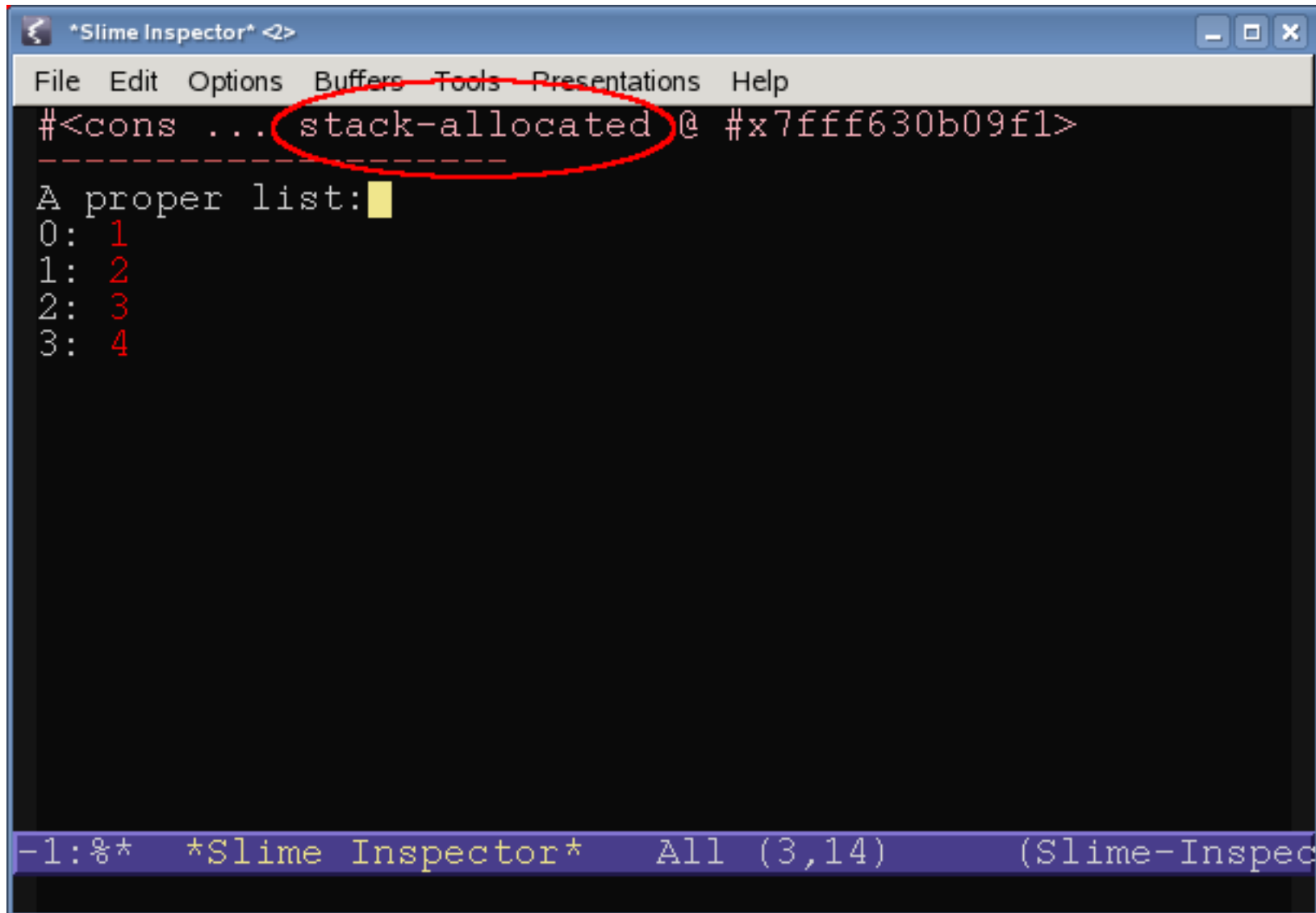
(macrolet ((frob (x)
            (let ((name (format nil
                                "frobbed-~A"
                                x)))
              `(defun ,(intern name) ()
                 x))))
  (frob :foo))

-1:** *slime-scratch* All (8,2) (Lisp Outl ElDo
(frobbed-foo)
(defun frobbed-foo () x)

-1:;%* *SLIME Macroexpansion* All (1,0) (Lisp Ou
(macrolet (&rest macro-bindings) &body body)
```

Unwinded dx objects

- you're in the Slime Debugger
- inspecting some frame variable which has been allocated on the stack



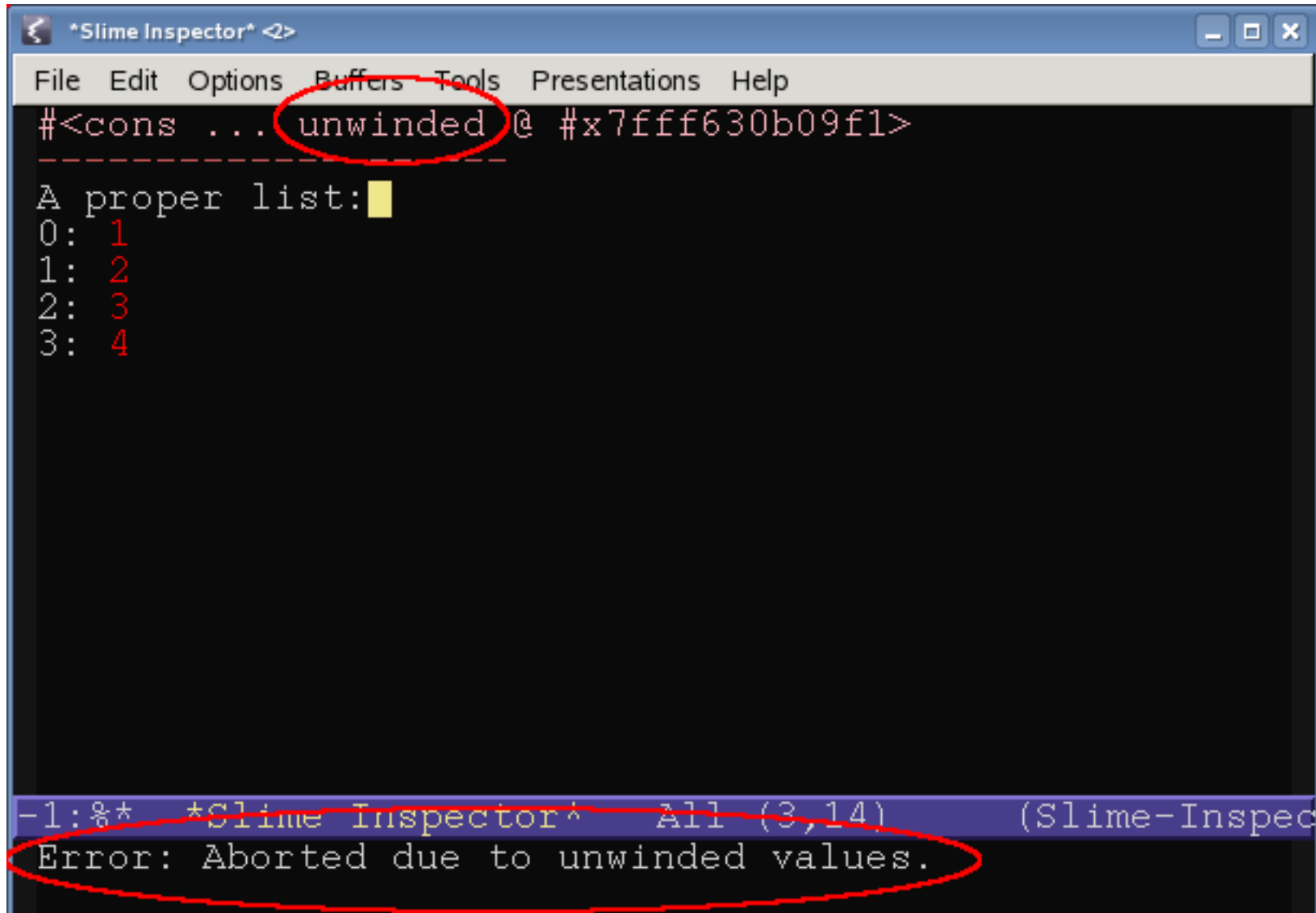
The screenshot shows the Slime Inspector window with the following content:

```
File Edit Options Buffers Tools Presentations Help
#<cons ... stack-allocated @ #x7fff630b09f1>
-----
A proper list:
0: 1
1: 2
2: 3
3: 4
```

The text "stack-allocated" in the first line is circled in red. The status bar at the bottom shows: -1:⌘* *Slime Inspector* All (3,14) (Slime-Inspect)

Unwinded dx objects (cont..)

- you close the debugger (→ stack unwinding)
- and reinspect the object in the Slime Inspector



The screenshot shows the Slime Inspector window with a menu bar (File, Edit, Options, Buffers, Tools, Presentations, Help) and a main text area. The text area contains the following content:

```
#<cons ... unwinded @ #x7fff630b09f1>
-----
A proper list:
0: 1
1: 2
2: 3
3: 4
```

At the bottom of the window, a status bar displays the following information:

```
-1: %* *Slime Inspector* All (3,14) (Slime-Inspe
```

An error message is displayed at the bottom of the window, circled in red:

```
Error: Aborted due to unwinded values.
```