local

<variable name(s)>	specify a local variable

If you name one or more variables in a **local** command within a program or interactively, Semper notes their current state or value and restores this at the end of the program or command line.

Examples

```
n=.2; ...
local n, m; ... ; n=5; library anyprog; ...
```

This sequence of commands leaves *n* set to 0.2, in spite of the assignment and program call.

```
n=.2 m=100; ...
for n=1,5; ... ; for m=n,5; ... ; library anyprog; loop; loop
```

This sequence of commands leaves *m* and *n* set to 0.2 and 100 in spite of the loops and program calls (the loop variables *m* and *n* are automatically made local).

Description

If you use **local** interactively, the value of a local variable is restored interactively at the end of the command line, provided that any **for** loops that you specified after the **local** command are closed using **loop**. The **local** command allows you to use variable names in programs without any risk of their clashing with names used at higher or lower calling levels.

Note that variables that are named in **for** loops are automatically treated as local variables, so you do not need to take any special action to preserve them.

Notes

see also:	**for...loop**
restrictions:	**local** may not be used within a loop