display() — Display text interpreting SMCL

Description

display(s) displays the string or strings contained in s.

display(s, asis) does the same thing but allows you to control how SMCL codes are treated. display(s, 0) is equivalent to display(s); any SMCL codes are honored.

display(s, asis), asis ≠ 0, displays the contents of s exactly as they are. For instance, when asis ≠ 0, "{it}" is just the string of characters {, i, t, and } and those characters are displayed; {it} is not given the SMCL interpretation of enter italic mode.

Syntax

void display(string colvector s)

void display(string colvector s, real scalar asis)

Remarks and examples

When s is a scalar, the differences between coding

: display(s)

and coding

: s

are

1. display(s) will not indent s; s by itself causes s to be indented by two spaces.

2. display(s) will honor any SMCL codes contained in s; s by itself is equivalent to display(s, 1). For example,

: s = "this is an {it:example}"
: display(s)
  this is an example
: s
  this is an {it:example}

3. When s is a vector, display(s) simply displays the lines, whereas s by itself adorns the lines with row and column numbers:
Another alternative to `display()` is `printf();` see [M-5] printf(). When `s` is a scalar, `display()` and `printf()` do the same thing:

```lisp
: display("this is an {it:example}")
this is an example
: printf("%s\n", "this is an {it:example}")
this is an example
```

`printf()`, however, will not allow `s` to be nonscalar; it has other capabilities.

**Conformability**

`display(s, asis)`

- **s:** \( k \times 1 \)
- **asis:** \( 1 \times 1 \) (optional)
- **result:** `void`

**Diagnostics**

None.

**Also see**

- [M-5] `displayas()` — Set display level
- [M-5] `displayflush()` — Flush terminal-output buffer
- [M-5] `printf()` — Format output
- [M-4] `IO` — I/O functions