**tokenize** — Divide strings into tokens

Syntax

```
tokenize [[`']][string][`'] [ , parse("pchars") ]
```

Description

`tokenize` divides `string` into tokens, storing the result in `1`, `2`, ... (the positional local macros). Tokens are determined based on the parsing characters `pchars`, which default to a space if not specified.

Option

```
parse("pchars")
```

specifies the parsing characters. If `parse()` is not specified, `parse(" ")` is assumed, and `string` is split into words.

Remarks and examples

`tokenize` may be used as an alternative or supplement to the `syntax` command (see [P] syntax) for parsing command-line arguments. Generally, it is used to further process the local macros created by `syntax`, as shown below.

```
program myprog
    version 13
    syntax [varlist] [if] [in]
    marksample touse
    tokenize `varlist'
    local first `1'
    macro shift
    local rest `*'
    ...
end
```

Example 1

We interactively apply `tokenize` and then display several of the numbered macros to illustrate how the command works.

```
. tokenize some words
. di "1=|'1'|, 2=|'2'|, 3=|'3'|"
   1=|somet|, 2=|words|, 3=|
. tokenize "some more words"
. di "1=|'1'|, 2=|'2'|, 3=|'3'|, 4=|'4'|"
   1=|somet|, 2=|more|, 3=|words|, 4=|
```
These examples illustrate that the quotes surrounding the string are optional; the space parsing character is not saved in the numbered macros; nonspace parsing characters are saved in the numbered macros together with the tokens being parsed; and more than one parsing character may be specified. Also, when called with no string argument, `tokenize` resets the local numbered macros to empty.

Also see

[P] foreach — Loop over items
[P] gettoken — Low-level parsing
[P] macro — Macro definition and manipulation
[P] syntax — Parse Stata syntax
[U] 18 Programming Stata