**tokenize —** Divide strings into tokens

### 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.

### Syntax

```
tokenize [[‘]”][`string’] [“] [ , parse("pchars") ]
```

### Option

`parse("pchars")` specifies the parsing characters. If `parse()` is not specified, `parse(" ")` is assumed, and `string` is split into words. Note that `pchars` is treated as a sequence of bytes. Any Unicode character in multibyte UTF-8 encoding, which applies to all Unicode characters except ASCII characters, is treated as a sequence of bytes rather than as a single character. For example, `parse()` will not work as expected when trying to break a string into tokens based on a Unicode whitespace character \u2000.

### 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.

```stata
program myprog
    version 16.1
    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
. display "1=|‘1’|, 2=|‘2’|, 3=|‘3’|"
1=|some|, 2=|words|, 3=|
. tokenize "some more words"
. display "1=|‘1’|, 2=|‘2’|, 3=|‘3’|, 4=|‘4’|"
1=|some|, 2=|more|, 3=|words|, 4=|
```

1
2  tokenize — Divide strings into tokens

. tokenize "Marcello Pagano" "Rino Bellocco"
. display "1=|'1'|, 2=|'2'|, 3=|'3'|"
1=|Marcello Pagano|, 2=|Rino Bellocco|, 3=|
. local str "A strange++string"
. tokenize 'str'
. display "1=|'1'|, 2=|'2'|, 3=|'3'|"
1=|A|, 2=|strange++string|, 3=|
. tokenize 'str', parse(" +")
. display "1=|'1'|, 2=|'2'|, 3=|'3'|, 4=|'4'|, 5=|'5'|, 6=|'6'|"
1=|A|, 2=|strange|, 3=|+|, 4=|+|, 5=|string|, 6=|
. tokenize 'str', parse("++")
. display "1=|'1'|, 2=|'2'|, 3=|'3'|, 4=|'4'|, 5=|'5'|, 6=|'6'|"
1=|A strange|, 2=|+|, 3=|+|, 4=|string|, 5=||, 6=|
. tokenize
. display "1=|'1'|, 2=|'2'|, 3=|'3'|"
1=||, 2=||, 3=||

These examples illustrate that the quotes surrounding the string are optional; the space parsing character is not saved in the numbered macros; non-space 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
[M-5] invtokens( ) — Concatenate string rowvector into string scalar
[M-5] tokenget() — Advanced parsing
[M-5] tokens() — Obtain tokens from string
[M-5] ustrsplit( ) — Split string into parts based on a Unicode regular expression
[U] 18 Programming Stata