Special Characters [ ] ( ) {} = ' . ... , ; : % ! @

Special characters

Syntax

```
[ ]
{ }
( )

=
'
.
.
.( )
..
...
,
;
:
%
%{ %}
!
@
```

Description

`[ ]` Brackets are used to form vectors and matrices. `[6.9 9.64 sqrt(-1)]` is a vector with three elements separated by blanks. `[6.9, 9.64, i]` is the same thing. `[1+j 2-j 3]` and `[1 +j 2 -j 3]` are not the same. The first has three elements, the second has five.

`[11 12 13; 21 22 23]` is a 2-by-3 matrix. The semicolon ends the first row.

Vectors and matrices can be used inside `[ ]` brackets. `[A B;C]` is allowed if the number of rows of `A` equals the number of rows of `B` and the number of columns of `A` plus the number of columns of `B` equals the number of columns of `C`. This rule generalizes in a hopefully obvious way to allow fairly complicated constructions.

`A = [ ]` stores an empty matrix in `A`. `A(m,:) = [ ]` deletes row `m` of `A`. `A(:,n) = [ ]` deletes column `n` of `A`. `A(n) = [ ]` reshapes `A` into a column vector and deletes the third element.

`[A1,A2,A3...] = function` assigns function output to multiple variables.

For the use of `[` and `]` on the left of an "=" in multiple assignment statements, see `lu`, `eig`, `svd`, and so on.

`{ }` Curly braces are used in cell array assignment statements. For example, `A(2,1) = {[1 2 3; 4 5 6]}`, or `A{2,2} = ('str')`. See `help paren` for more information about `{ }`.

`( )` Parentheses are used to indicate precedence in arithmetic expressions in the usual way. They are used to enclose arguments of functions in the usual way. They are also used to enclose subscripts of vectors and matrices in a manner somewhat more general than usual. If `X` and `V` are vectors, then `X(V)` is `[X(V(1)), X(V(2)), ..., X(V(n))]`. The components of `V` must be integers to be used as subscripts. An error occurs if any such subscript is less than 1 or greater than the size of `X`. Some examples are

- `X(3)` is the third element of `X`.
- `X([1 2 3])` is the first three elements of `X`.

See `help paren` for more information about `( )`.

If `X` has `n` components, `X(n:-1:1)` reverses them. The same indirect subscripting works in matrices. If `V` has `m` components and `W` has `n` components, then `A(V,W)` is the `m`-by-`n` matrix formed from the elements of `A` whose subscripts are the elements of `V` and `W`. For example, `A([1,5],:) = A([5,1],:)` interchanges rows 1 and 5 of `A`.

`=` Used in assignment statements. `B = A` stores the elements of `A` in `B`. == is the relational equals operator. See the Relational Operators < > <= >= == ~= page.

`'` Matrix transpose. `X'` is the complex conjugate transpose of `X`. `X.'` is the nonconjugate transpose.

Quotation mark. `'any text'` is a vector whose components are the ASCII codes for the characters. A quotation mark within the text is indicated by two quotation marks.

`.` Decimal point. `314/100`, `3.14`, and `.314e1` are all the same.

Element-by-element operations. These are obtained using `.*`, `.^`, `./`, or `.\`. See the Arithmetic Operators page.

`.` Field access. `S(m).f` when `S` is a structure, accesses the contents of field `f` of that structure.

`.( )` Dynamic Field access. `S.(df)` when `A` is a structure, accesses the contents of dynamic field `df` of that structure. Dynamic field names are defined at runtime.

`..` Parent directory. See `cd`.

`...` Continuation. Three or more periods at the end of a line continue the current function on the next line. Three or more periods before the end of a line cause MATLAB to ignore the remaining text on the current line and continue the function on the next line. This effectively makes a comment out of anything on the current line that follows the three periods. See Entering Long Statements (Line Continuation) for more information.

`,` Comma. Used to separate matrix subscripts and function arguments. Used to separate statements in multistatement lines. For multistatement lines, the comma can be replaced by a semicolon to suppress printing.

`;` Semicolon. Used inside brackets to end rows. Used after an expression or statement to suppress printing or to separate statements.

`:` Colon. Create vectors, array subscripting, and `for` loop iterations. See `colon (:)` for details.

`%` Percent. The percent symbol denotes a comment; it indicates a logical end of line. Any following text is ignored. MATLAB displays the first contiguous comment lines in a M-file in response to a `help` command.

`%{` `%}` Percent-brace. The text enclosed within the `%{` and `%}` symbols is a comment block. Use these symbols to insert comments that take up more than a single line in your M-file code. Any text between these two symbols is ignored by MATLAB.

`!` Exclamation point. Indicates that the rest of the input line is issued as a command to the operating system. See Running External Programs for more information.

`@` Function handle. MATLAB data type that is a handle to a function. See `function_handle (@)` for details.

Remarks

Some uses of special characters have M-file function equivalents, as shown:

Horizontal concatenation	`[A,B,C...]`	`horzcat(A,B,C...)`
Vertical concatenation	`[A;B;C...]`	`vertcat(A,B,C...)`
Subscript reference	`A(i,j,k...)`	`subsref(A,S)`. See help `subsref`.
Subscript assignment	`A(i,j,k...)= B`	`subsasgn(A,S,B)`. See help `subsasgn`.

Note For some toolboxes, the special characters are overloaded, that is, they perform differently in the context of that toolbox. To see the toolboxes that overload a given character, type `help` followed by the character name. For example, type `help transpose`. The toolboxes that overload `transpose` (`.'`) are listed. For information about using the character in that toolbox, see the documentation for the toolbox.

See Also

Arithmetic Operators + - * / \ ^ '

Relational Operators < > <= >= == ~=

Logical Operators: Elementwise & | ~,

Logical Operators: Short-circuit && || colon (:)