MATLABII

Data Types Available in MATLAB

MATLAB provides 15 fundamental data types. Every data type stores data that is in the form of a matrix or array. The size of this matrix or array is a minimum of 0-by-0 and this can grow up to a matrix or array of any size.

The following table shows the most commonly used data types in MATLAB −

|  |  |
| --- | --- |
| **Sr.No.** | **Data Type & Description** |
| 1 | **int8**  8-bit signed integer |
| 2 | **uint8**  8-bit unsigned integer |
| 3 | **int16**  16-bit signed integer |
| 4 | **uint16**  16-bit unsigned integer |
| 5 | **int32**  32-bit signed integer |
| 6 | **uint32**  32-bit unsigned integer |
| 7 | **int64**  64-bit signed integer |
| 8 | **uint64**  64-bit unsigned integer |
| 9 | **single**  single precision numerical data |
| 10 | **double**  double precision numerical data |
| 11 | **logical**  logical values of 1 or 0, represent true and false respectively |
| 12 | **char**  character data (strings are stored as vector of characters) |
| 13 | **cell array**  array of indexed cells, each capable of storing an array of a different dimension and data type |
| 14 | **structure**  C-like structures, each structure having named fields capable of storing an array of a different dimension and data type |
| 15 | **function handle**  pointer to a function |
| 16 | **user classes**  objects constructed from a user-defined class |
| 17 | **java classes**  objects constructed from a Java class |

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. MATLAB is designed to operate primarily on whole matrices and arrays. Therefore, operators in MATLAB work both on scalar and non-scalar data. MATLAB allows the following types of elementary operations −

* Arithmetic Operators
* Relational Operators
* Logical Operators
* Bitwise Operations
* Set Operations

Arithmetic Operators

MATLAB allows two different types of arithmetic operations −

* Matrix arithmetic operations
* Array arithmetic operations

Matrix arithmetic operations are same as defined in linear algebra. Array operations are executed element by element, both on one-dimensional and multidimensional array.

The matrix operators and array operators are differentiated by the period (.) symbol. However, as the addition and subtraction operation is same for matrices and arrays, the operator is same for both cases. The following table gives brief description of the operators −

[Show Examples](https://www.tutorialspoint.com/matlab/matlab_arithmetic_operators.htm)

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **+**  Addition or unary plus. A+B adds the values stored in variables A and B. A and B must have the same size, unless one is a scalar. A scalar can be added to a matrix of any size. |
| 2 | **-**  Subtraction or unary minus. A-B subtracts the value of B from A. A and B must have the same size, unless one is a scalar. A scalar can be subtracted from a matrix of any size. |
| 3 | **\***  Matrix multiplication. C = A\*B is the linear algebraic product of the matrices A and B. More precisely,  Matrix Multiplication  For non-scalar A and B, the number of columns of A must be equal to the number of rows of B. A scalar can multiply a matrix of any size. |
| 4 | **.\***  Array multiplication. A.\*B is the element-by-element product of the arrays A and B. A and B must have the same size, unless one of them is a scalar. |
| 5 | **/**  Slash or matrix right division. B/A is roughly the same as B\*inv(A). More precisely, B/A = (A'\B')'. |
| 6 | **./**  Array right division. A./B is the matrix with elements A(i,j)/B(i,j). A and B must have the same size, unless one of them is a scalar. |
| 7 | **\**  Backslash or matrix left division. If A is a square matrix, A\B is roughly the same as inv(A)\*B, except it is computed in a different way. If A is an n-by-n matrix and B is a column vector with n components, or a matrix with several such columns, then X = A\B is the solution to the equation *AX = B*. A warning message is displayed if A is badly scaled or nearly singular. |
| 8 | **.\**  Array left division. A.\B is the matrix with elements B(i,j)/A(i,j). A and B must have the same size, unless one of them is a scalar. |
| 9 | **^**  Matrix power. X^p is X to the power p, if p is a scalar. If p is an integer, the power is computed by repeated squaring. If the integer is negative, X is inverted first. For other values of p, the calculation involves eigenvalues and eigenvectors, such that if [V,D] = eig(X), then X^p = V\*D.^p/V. |
| 10 | **.^**  Array power. A.^B is the matrix with elements A(i,j) to the B(i,j) power. A and B must have the same size, unless one of them is a scalar. |
| 11 | **'**  Matrix transpose. A' is the linear algebraic transpose of A. For complex matrices, this is the complex conjugate transpose. |
| 12 | **.'**  Array transpose. A.' is the array transpose of A. For complex matrices, this does not involve conjugation. |

Relational Operators

Relational operators can also work on both scalar and non-scalar data. Relational operators for arrays perform element-by-element comparisons between two arrays and return a logical array of the same size, with elements set to logical 1 (true) where the relation is true and elements set to logical 0 (false) where it is not.

The following table shows the relational operators available in MATLAB −

[Show Examples](https://www.tutorialspoint.com/matlab/matlab_relational_operators.htm)

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **<**  Less than |
| 2 | **<=**  Less than or equal to |
| 3 | **>**  Greater than |
| 4 | **>=**  Greater than or equal to |
| 5 | **==**  Equal to |
| 6 | **~=**  Not equal to |

Logical Operators

MATLAB offers two types of logical operators and functions −

* Element-wise − These operators operate on corresponding elements of logical arrays.
* Short-circuit − These operators operate on scalar and, logical expressions.

Element-wise logical operators operate element-by-element on logical arrays. The symbols &, |, and ~ are the logical array operators AND, OR, and NOT.

Short-circuit logical operators allow short-circuiting on logical operations. The symbols && and || are the logical short-circuit operators AND and OR.

[Show Examples](https://www.tutorialspoint.com/matlab/matlab_logical_operators.htm)

Bitwise Operations

Bitwise operators work on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ are as follows −

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **p** | **q** | **p & q** | **p | q** | **p ^ q** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Set Operations

MATLAB provides various functions for set operations, like union, intersection and testing for set membership, etc.

The following table shows some commonly used set operations −

[Show Examples](https://www.tutorialspoint.com/matlab/matlab_set_operators.htm)

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Description** |
| 1 | **intersect(A,B)**  Set intersection of two arrays; returns the values common to both A and B. The values returned are in sorted order. |
| 2 | **intersect(A,B,'rows')**  Treats each row of A and each row of B as single entities and returns the rows common to both A and B. The rows of the returned matrix are in sorted order. |
| 3 | **ismember(A,B)**  Returns an array the same size as A, containing 1 (true) where the elements of A are found in B. Elsewhere, it returns 0 (false). |
| 4 | **ismember(A,B,'rows')**  Treats each row of A and each row of B as single entities and returns a vector containing 1 (true) where the rows of matrix A are also rows of B. Elsewhere, it returns 0 (false). |
| 5 | **issorted(A)**  Returns logical 1 (true) if the elements of A are in sorted order and logical 0 (false) otherwise. Input A can be a vector or an N-by-1 or 1-by-N cell array of strings. **A is considered to be sorted if A** and the output of sort(A) are equal. |
| 6 | **issorted(A, 'rows')**  Returns logical 1 (true) if the rows of two-dimensional matrix A is in sorted order, and logical 0 (false) otherwise. **Matrix A is considered to be sorted if A** and the output of sortrows(A) are equal. |
| 7 | **setdiff(A,B)**  Sets difference of two arrays; returns the values in A that are not in B. The values in the returned array are in sorted order. |
| 8 | **setdiff(A,B,'rows')**  Treats each row of A and each row of B as single entities and returns the rows from A that are not in B. The rows of the returned matrix are in sorted order.  The 'rows' option does not support cell arrays. |
| 9 | **setxor**  Sets exclusive OR of two arrays |
| 10 | **union**  Sets union of two arrays |
| 11 | **unique**  Unique values in array |

All variables of all data types in MATLAB are multidimensional arrays. A vector is a one-dimensional array and a matrix is a two-dimensional array.

We have already discussed vectors and matrices. In this chapter, we will discuss multidimensional arrays. However, before that, let us discuss some special types of arrays.

Special Arrays in MATLAB

In this section, we will discuss some functions that create some special arrays. For all these functions, a single argument creates a square array, double arguments create rectangular array.

The **zeros()** function creates an array of all zeros −

For example −

[Live Demo](http://tpcg.io/w1CS2v)

zeros(5)

MATLAB will execute the above statement and return the following result −

ans =

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

The **ones()** function creates an array of all ones −

For example −

[Live Demo](http://tpcg.io/cE3ZT3)

ones(4,3)

MATLAB will execute the above statement and return the following result −

ans =

1 1 1

1 1 1

1 1 1

1 1 1

The **eye()** function creates an identity matrix.

For example −

[Live Demo](http://tpcg.io/XLSqxZ)

eye(4)

MATLAB will execute the above statement and return the following result −

ans =

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1