# MATLAB for Python Users

The MATLAB language is designed primarily for math-intensive scientific computing. MATLAB combines a desktop environment tuned for iterative analysis with a programming language that expresses matrix and array mathematics directly. Understanding the philosophy and API design can help while learning MATLAB.

**Enhance Python with MATLAB**

Integrate MATLAB's advanced tools directly into your Python workflows.

» Learn more

## Table of Contents



## » General Behavior

| Python Syntax | MATLAB Syntax | Purpose | MATLAB Examples |
|---|---|---|---|
| `#` | `%` | Comment | `% hello` |
| `print` | `Do not terminate with ;` | Print output | `x` |
| `/` | `...` | Continue to next line | `x = 1+...2;` |
| `os` | `!` | Operating system command | `! echo hi` |
| `+ - * /` | `+ - * /` | Mathematical operators | `x = 1+2` |
| `**` | `^` | Exponent | `x = y^2` |
| `* / **` | `.* ./ .^` | Element-wise operators | `x = [1 2].* [3 4]` |
| `not, and, or` | `~ & \|` | NOT, AND, OR logical operators | `if x<2 & x>2` |
| `del` | `clear` | Clear variable from memory | `clear x y` |
| `clear` | `clc` | Clear command window | `clc` |

## » Referencing

| MATLAB Syntax | Purpose | Example |
|---|---|---|
| `( )` | Index (copy-on-write) | `x(1,1)` |
| `[ ]` | Create array | `x = [1 2 3]` |
| | Join arrays | `z = [x ; y]` |
| `{ }` | Create cell arrays | `x = {42; "hello world"}` |
| | Extract contents from a container | `x{1,1}` |
| `.` | Access class property or method | `obj.Data` |
| | Reference table or struct field | `t.FieldName` |

- Beginning element has an index of 1.
- Indexing is left and right inclusive.
- Indexing options include N-D indexing (row,col), linear indexing (element number), and logical indexing (conditional statement).

## » Functions

| | | |
|---|---|---|
| **Creating functions** | You can declare functions within a function file. Input arguments are captured in parentheses. | `function z = foo(x,y)`<br>`  ...`<br>`end` |
| | Multiple outputs are captured with square brackets. | `function [a,b] = foo(x,y)`<br>`  ...`<br>`end` |
| **Calling functions with input arguments and name-value pairs** | | `y = foo(x,y,"Name",Value)` |

## » Data Types

Similar data types:

| Python | MATLAB |
|---|---|
| `float` | `double, single` |
| `complex` | `complex single, complex double` |
| `int` | `(u)int8, (u)int16, (u)int32, (u)int64` |
| `float(nan)` | `NaN` |
| `float(inf)` | `inf` |
| `str` | `str, char` |
| `bool` | `logical` |
| `dict` | `struct` |
| `list, tuple` | `cell` |
| `pandas.dataframe` | `table` |

MATLAB defaults to store all numeric values as double-precision floating-point numbers. Python stores some numbers as integers and others as floating-point numbers. In MATLAB, `for x=4` and `y=4.0`, `x` is always equal to `y`.

## » Control Flow

| Statement | Example |
|---|---|
| for | `for i = 1:10`<br>`  ...`<br>`end` |
| if | `if x<`<br>`  ...`<br>`  elseif x == 2`<br>`  else`<br>`  ...`<br>`end` |
| while | `while x<3`<br>`  ...`<br>`end` |
| switch-case | `switch switch_arg`<br>`  ...`<br>`  case case_arg`<br>`  ...`<br>`end` |
| try-catch | `try`<br>`  ...`<br>`  catch`<br>`  ...`<br>`end` |

## » Objects

**Define a class**

```
classdef MyClass
   properties
      MyProp
   end
   methods
      function obj = MyClass(val)
      end
      function y = MyMethod(obj,x)
      end
   end
end
```

**Use a class**

- Save the class definition with the same name as the class
  `MyClass.m`
- Create an object of the class
  `a = MyClass`
- Access the properties
  `a.MyProp`
- Call methods to perform operations
  `b = MyMethod(a,val)`
- To pass-by-reference, create a "handle" class

```
classdef myclass < handle
   ...
end
```


© 1994-2025 The MathWorks, Inc.