

This table was modified/updated from the table [here](#).

Modifications by [Jenni Rinker](#) with contributions from Nikolay Dimitrov. (Thanks!)

Python code	MATLAB code
<pre># numeric variables # are double precision by default  a = 5.0 # this is a float a = 5   # this is an int</pre>	<pre>% numeric variables % are double precision by default  a = 5.0;</pre>
<pre># arrays are defined in NumPy package # array indexes start at 0 in Python # structures are defined by #   indentation, no 'end'  import numpy as np  A = np.empty(10) # initialize array A for i in range(10): # 0 to 9     A[i] = i + 1 print(A)</pre>	<pre>% array indexes start at 1 in Matlab % indentation is for readability only  for i=1:10     A(i) = i; % no need to initialize A end A % display contents of A</pre>
<pre># get a range of values with skips  for i in range(0, 11, 2):     print(i)</pre>	<pre>for i=0:2:10     fprintf(' %i \n', i) end</pre>
<pre># initialize an identity matrix  import numpy as np  B = np.identity(3)</pre>	<pre>% MATLAB has built-in functions for % common array initializations  B = eye(100);</pre>
<pre># declare and initialize an array  import numpy as np  C = np.array([1, 2, 3])</pre>	<pre>C = [1, 2, 3]; % or C = [1 2 3];</pre>
<pre># numpy arange with skips  import numpy as np  C = np.arange(2, 10, 2) print(C)</pre>	<pre>% array name = [start:increment:end];  C = [2:2:8] % leave off ; to display value</pre>

<pre># print an array element on screen # array indexes start at 0  print(C[1])  # prints 4 using C from array defined #   above # note square brackets C[1]</pre>	<pre>% array indexes start at 1  C(2)  % prints 4 using C from above table cell % note parentheses C(2)</pre>
<pre># declare and initialize an array # with fixed interval between values  import numpy as np  C = np.linspace(2, 8, 4)  # third param is opt: num of points #   between and including 1st two #   points # if third param left off, default # is 50 points</pre>	<pre>C = linspace(2,8,4);  % third param is optional and = # points % between and including 1st two points % if third param left off, default % is 100 points</pre>
<pre># initialize a 2D array  import numpy as np  D = np.array([[1, 2, 3],               [4, 5, 6],               [7, 8, 9]])</pre>	<pre>% these three examples accomplish the % same thing  D = [1 2 3; 4 5 6; 7 8 9]; D = [1:3; 4:6; 7:9]; D = [1 2 3      4 5 6      7 8 9];</pre>
<pre># print element of 2D array # array indexes start at 0  print(D[1, 1]) # row 2, column 2  # prints 5 using D defined above</pre>	<pre>% array indexes start at 1  D(2,2) % row 2, column 2  % prints 5 using D from above table cell</pre>
<pre># print selected sub array # e.g., first two rows of 1st column # Note indexing starts at 0, ends at #   end value - 1  print(D[:2, 0])</pre>	<pre>D(1:2, 1) % rows 1 to 2 of column 1</pre>
<pre># print all rows of 1st column  print(D[:, 1])</pre>	<pre>D(:,1) % all rows, column 1</pre>

<pre># logical expression # for Booleans, can use 'or' or ' '  a = 1 b = 2 if (a == 1)   (b == 3):     print('a = 1 or b = 3')</pre>	<pre>a = 1; b = 2; if a == 1    b == 3     fprintf('a = 2 or b = 3 \n'); end</pre>
<pre># if structure  if (a == 1) and (b != 3):     print('a=1 and b not 3')     print('OK?')</pre>	<pre>if a == 1 &amp;&amp; b ~= 3     fprintf('a=1 and b not 3 \n');     fprintf('OK? \n'); end</pre>
<pre># if, else structure  if a != 1:     print('a is not 1') elif b != 3:     print('b is not 3') else:     print('huh?')</pre>	<pre>a ~= 1     fprintf('a is not 1 \n') elseif b ~= 3     fprintf('b is not 3 \n') else     fprintf('huh? \n') end</pre>
<pre># switch structure  # Python doesn't have a switch structure  # any switch structure can be # written as an if-else structure  # switch structures may be quicker to # read and write for applications such # as menus</pre>	<pre>switch menuChoice     case 1         % can do any actions in a case, e.g.,         % call a user-defined function          myMenuFunc01();     case 2         myMenuFunc02();     case 3         myMenuFunc03();     otherwise         fprintf('invalid selection, try again') end</pre>

<pre> # Non-Anonymous functions # program that calls a user-defined #   function called 'myfunc'  def myfunc(x, y):     return x**y # ** is power  # call function  z = myfunc(2, 3) print(z) # prints '8' type(z) # prints 'int' </pre>	<pre> % Non-Anonymous functions % main program and function definition must % be in separate files and function file % must have same name as function name  z = myfunc(2,3) % prints 8 for this input  ----- LISTING OF FILE myfunc.m -----  function returnValue = myfunc(x,y)     returnValue = x^y; % ^ is exp operator      % function is a keyword     % returnValue is arbitrary varbl name </pre>
<pre> # Anonymous functions  myfunc = lambda x, y: x ** y z = myfunc(2, 3) # z is 8 </pre>	<pre> % Anonymous functions  myfunc = @(x,y) x^y z = myfunc(2,3) % z is 8 </pre>
<pre> # matrix multiplication  import numpy as np  A = np.array([[2,3],               [3, 5]]) B = np.array([[1,2],               [5, -1]])  C = A * B print(C) </pre>	<pre> A = [2, 3; 3, 5]; B = [1, 2; 5, -1];  C = A * B </pre>

```
# plotting

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.ylabel('sin(x)')
plt.xlabel('x')

plt.show() # sometimes need to call
# this function to show the plot
```

```
x = linspace(0,2*pi,100);
y = sin(x);

plot(x,y)
ylabel('sin(x)')
xlabel('x')
```