The Matlab Command Window (>>)

>>
The Matlab Command Window (>>)

>> 1+2
The Matlab Command Window (>>)

>> 1+2

ans =

3
The Matlab Command Window (>>)

>> 1+2

ans =

    3

>> x = 1 + 2

x =

    3
The Matlab Command Window (>>)

>> 1+2
ans =
    3

>> x = 1 + 2
x =
    3

>> x = 1 + 2; % semicolon suppresses output
>>
The Matlab Command Window (>>)

>> 1+2

ans =

   3

>> x = 1 + 2

x =

   3

>> x = 1 + 2;  % semicolon suppresses output
>> x

x =

   3

>>
Data Types

- there are 15 fundamental data types
- like Python, you don’t have to declare data types
- just about every data type can be thought of as matrix
  - a scalar is a 1x1 matrix
  - a string is a 1xN matrix
  - a vector is a 1xN or Nx1 matrix
Name Space

>> x = 1; % "global" variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous
Name Space

>> x = 1; % “global” variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> pi
    ans = 3.1416

>> pi = 'foo';

>> pi
    ans = 'foo'
Name Space

>> x = 1; % "global" variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> pi
   ans = 3.1416

>> pi = 'foo';

>> pi
   ans = 'foo'

>> i
   ans = 0.0000 + 1.0000i

>> i = 1;

>> i
   ans = 1
Name Space

>> x = 1; % "global" variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> max(1,2)
   ans = 2
Name Space

>> x = 1; % "global" variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> max(1,2)
ans = 2

>> max = 10;
Name Space

>> x = 1; % "global" variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> max(1,2)
ans = 2

>> max = 10;

>> max(1,2); % max is now a 1x1 matrix
Index exceeds matrix dimensions.
Name Space

>> x = 1; % “global” variable

>> x = 1.14; % overwrites previous value

>> x = 'foo'; % convenient and dangerous

>> max(1,2)
    ans = 2

>> max = 10;

>> max(1,2); % max is now a 1x1 matrix
    Index exceeds matrix dimensions.

>> clear; % clear all variables in global name space

>> max(1,2)
    ans = 2
Basic (scalar) Arithmetic

addition $+$
subtraction $-$
multiplication $*$
division $/$ or \ [10/2 = 5 and 10\2 = 2/10 = 0.2]
exponentiation $^\wedge$
Vectors and Matrices

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}
\]

\[
\begin{bmatrix}
\end{bmatrix}
\]
Vectors and Matrices

>> u = [1 2 3] % a 1 x 3 matrix (row vector)
   u =     1     2     3

>> u' % transpose (row -> column vector)
   ans =
     1
     2
     3
Vectors and Matrices

\[
>> v = [1 \; 2 \; 3] \quad \text{% 3 x 1 matrix (column vector)}
\]
\[
v =
\begin{align*}
1 \\
2 \\
3
\end{align*}
\]

\[
>> v' \quad \text{% transpose (column -> row vector)}
\]
\[
\text{ans} = \begin{array}{c}
1 \\
2 \\
3
\end{array}
\]
Vectors and Matrices

>> m = [1 2 3 ; 4 5 6] % a 2 x 3 matrix
m =
    1     2     3
    4     5     6

>> m’ % transpose (2 x 3 matrix --> 3 x 2 matrix)
ans =  1     4
       2     5
       3     6
Vectors and Matrices

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix
Vectors and Matrices

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> u(1) % matlab vectors/matrices are indexed on 1
ans = 1
Vectors and Matrices

>> s = 5;  \% a scalar value or a 1x1 matrix
>> u = [1 2 3];  \% a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3];  \% 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6];  \% a 2 x 3 matrix

>> u(1)  \% matlab vectors/matrices are indexed on 1
    ans = 1

>> v(1)  \% row/column vectors are index the same
    ans = 1
Vectors and Matrices

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> u(1) % matlab vectors/matrices are indexed on 1
   ans = 1

>> v(1) % row/column vectors are index the same
   ans = 1

>> m(1,1) % index matrices as (row,column)
   ans = 1
Vectors and Matrices

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> u(1) % matlab vectors/matrices are indexed on 1
   ans = 1

>> v(1) % row/column vectors are index the same
   ans = 1

>> m(1,1) % index matrices as (row,column)
   ans = 1

>> m(1,2)
   ans = 2
Vectors and Matrices

>> s = 5;  % a scalar value or a 1x1 matrix
>> u = [1 2 3];  % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3];  % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6];  % a 2 x 3 matrix

>> u(1)  % matlab vectors/matrices are indexed on 1
ans = 1

>> v(1)  % row/column vectors are index the same
ans = 1

>> m(1,1)  % index matrices as (row,column)
ans = 1

>> m(1,2)
ans = 2

>> m(2,1)
ans = 4
Vectors and Matrices

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> m(1,:)
ans = 1 2 3

>> m(:,3)
ans = 3
   6

>> m(1:2,1:2)
ans = 1 2
   4 5
Vector and Matrix Initialization

```matlab
>> v = zeros(1,10); % [0 0 0 0 0 0 0 0 0 0]
>> v = ones(10,1); % [1 ; 1 ; 1 ; 1 ; 1 ; 1 ; 1 ; 1 ; 1 ; 1]
>> v = [1:10]; % [1 2 3 4 5 6 7 8 9 10]
>> v = [1 : 2 : 10]; % [1 3 5 7 9]
>> m = zeros(2,3); % 2 x 3 matrix of zeros
>> m = eye(3); % 3 x 3 identity matrix (1 on diagonal)
>> m = []; % empty matrix (of size 0 x 0)
```
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix
Vector and Matrix Dimensions

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> size(s)
   ans = 1 1
Vector and Matrix Dimensions

\[
\begin{align*}
\text{>> } s & = 5; \quad \% \text{ a scalar value or a } 1 \times 1 \text{ matrix} \\
\text{>> } u & = [1 \ 2 \ 3]; \quad \% \text{ a } 1 \times 3 \text{ matrix (row vector)} \\
\text{>> } v & = [1; \ 2; \ 3]; \quad \% \text{ a } 3 \times 1 \text{ matrix (column vector)} \\
\text{>> } m & = [1 \ 2 \ 3; \ 4 \ 5 \ 6]; \quad \% \text{ a } 2 \times 3 \text{ matrix}
\end{align*}
\]

\[
\begin{align*}
\text{>> } \text{size}(s) \\
\text{ans} & = 1 \quad 1 \\
\text{>> } \text{size}(u) \\
\text{ans} & = 1 \quad 3 \\
\text{>> } \text{size}(v) \\
\text{ans} & = 3 \quad 1
\end{align*}
\]
Vector and Matrix Dimensions

```
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
>> v = [1 ; 2 ; 3]; % 3 x 1 matrix (column vector)
>> m = [1 2 3 ; 4 5 6]; % a 2 x 3 matrix

>> size(s)
    ans = 1     1

>> size(u)
    ans = 1     3

>> size(v)
    ans = 3     1

>> size(m)
    ans = 2     3
```
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] – point-wise addition (no loops)
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] — point-wise addition (no loops)

>> v * 2; % [2 4 6 8] — point-wise multiplication (no loops)
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] — point-wise addition (no loops)

>> v * 2; % [2 4 6 8] — point-wise multiplication (no loops)

>> v^2; % v * v — ERROR
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5;  \% [6 7 8 9] — point-wise addition (no loops)

>> v * 2;  \% [2 4 6 8] — point-wise multiplication (no loops)

>> v^2;  \% v * v — ERROR

>> v.^2  \% [1 4 9 16] — point-wise exponentiation
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] — point-wise addition (no loops)

>> v * 2; % [2 4 6 8] — point-wise multiplication (no loops)

>> v^2; % v * v — ERROR

>> v.^2 % [1 4 9 16] — point-wise exponentiation

>> w = [5 6 7 8];

>> v + w; % [6 8 10 12] — point-wise addition
Matrix Arithmetic

>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] — point-wise addition (no loops)

>> v * 2; % [2 4 6 8] — point-wise multiplication (no loops)

>> v^2; % v * v — ERROR

>> v.^2 % [1 4 9 16] — point-wise exponentiation

>> w = [5 6 7 8];

>> v + w; % [6 8 10 12] — point-wise addition

>> v * w; % ERROR
Matrix Arithmetic

```matlab
>> v = [1 2 3 4];

>> v + 5; % [6 7 8 9] — point-wise addition (no loops)

>> v * 2; % [2 4 6 8] — point-wise multiplication (no loops)

>> v^2; % v * v — ERROR

>> v.^2 % [1 4 9 16] — point-wise exponentiation

>> w = [5 6 7 8];

>> v + w; % [6 8 10 12] — point-wise addition

>> v * w; % ERROR

>> v .* w; % [5 12 21 32] — point-wise multiplication
```
Matrix Arithmetic

>> v = [1 2 3];

>> w = [4 5 6];

>> v * w';  % inner-product: (1 x 3) * (3 x 1) = 1 x 1

>> v' * w;  % outer-product: (3 x 1) * (1 x 3) = 3 x 3
Matrix Arithmetic

>> v = [1 2 3];
>> w = [4 5 6];

>> v * w'; % inner-product: (1 x 3) * (3 x 1) = 1 x 1
>> v' * w; % outer-product: (3 x 1) * (1 x 3) = 3 x 3

>> m = [1 2 ; 3 4]; % 2 x 2 matrix

>> p = m * m; % matrix multiplication

>> inv(m) * m; % matrix inverse
Print to console

```matlab
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)
```
Print to console

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)

>> s

s =

      5
Print to console

```
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)

>> s

s =

    5

>> disp(s);
  5
```
Print to console

>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)

>> s

s =

    5

>> disp(s);

    5

>> disp(u);

    1     2     3
Print to console

```matlab
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)

>> s

s =
  
      5

>> disp(s);
    
      5

>> disp(u);
   
      1     2     3

>> fprintf( '
', s );

5
```
Print to console

```matlab
>> s = 5; % a scalar value or a 1x1 matrix
>> u = [1 2 3]; % a 1 x 3 matrix (row vector)

>> s

s =

   5

>> disp(s);

5

>> disp(u);

     1     2     3

>> fprintf( '%d\n', s );

5

>> fprintf( '%d ', u );

1 2 3 >>
```
Matlab Workspace

>> x = 4;
>> y = 8;
>> z = 'foo';
Matlab Workspace

>> x = 4;
>> y = 8;
>> z = 'foo';

>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>1x3</td>
<td>6</td>
<td>char</td>
<td></td>
</tr>
</tbody>
</table>
Matlab Workspace

>> x = 4;
>> y = 8;
>> z = 'foo';

>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>1x3</td>
<td>6</td>
<td>char</td>
<td></td>
</tr>
</tbody>
</table>

>> save ws.mat % save all variables to file
>> save ws.mat x y % save specific variables to file
>> load ws.mat % load pre-saved variables
Matlab Workspace

```matlab
>> x = 4;
>> y = 8;
>> z = 'foo';

>> whos
    Name      Size            Bytes  Class     Attributes
          x         1x1                 8  double
          y         1x1                 8  double
          z         1x3                 6  char

>> save ws.mat % save all variables to file
>> save ws.mat x y % save specific variables to file
>> load ws.mat % load pre-saved variables

>> clear x; % clear one variable
>> clear; % clear entire workspace
```