Previous class:
- User-defined function
- Nested loops

Now:
- Working with colors
- 1-dimensional array—vector

Generating tables and plots

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.784</td>
<td>0.707</td>
</tr>
<tr>
<td>1.571</td>
<td>1.000</td>
</tr>
<tr>
<td>3.142</td>
<td>0.000</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.707</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.000</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.707</td>
</tr>
<tr>
<td>6.283</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Note: x, y are shown in columns due to space limitation; they should be rows.

Built-in function linspace

```matlab
x = linspace(1,3,5);
x = linspace(0:pi, 9);
y = sin(x);
plot(x,y)
```

How did we get all the sine values?

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1.57</td>
<td>1.00</td>
</tr>
<tr>
<td>3.14</td>
<td>0.00</td>
</tr>
<tr>
<td>4.71</td>
<td>-1.00</td>
</tr>
<tr>
<td>6.28</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Built-in functions accept arrays

```
0.00 1.57 3.14 4.71 6.28
```

and return arrays

```
0.00 1.00 0.00 -1.00 0.00
```

Vectorized addition

```matlab
x = [2 1.5 8];
y = [1 2 0 1];
z = x + y
```

Matlab code: `z = x + y`
Vectorized multiplication

\[ \begin{array}{c}
a & 2 & 1.5 & 8 \\
x & \\
\times & b & 1 & 2 & 0 & 1 \\
\end{array} \]

\[ = c & 2 & 2 & 0 & 8 \]

Matlab code: \( c = a .* b \)

Vectorized element-by-element arithmetic operations on arrays

\[ \begin{array}{c}
a & 2 & 1.5 & 8 \\
x & \times & b & 1 & 2 & 0 & 1 \\
\end{array} \]

\[ = c & 2 & 2 & 0 & 8 \]

A dot (.) is necessary in front of these math operators

Shift

\[ \begin{array}{c}
a & 2 & 1.5 & 8 \\
x & \times & b & 1 & 2 & 0 & 1 \\
\end{array} \]

\[ = c & 2 & 2 & 0 & 8 \]

Matlab code: \( c = a .* b \)

Reciprocate

\[ \begin{array}{c}
a & 2 & 1.5 & 8 \\
x & \times & b & 1 & 2 & 0 & 1 \\
\end{array} \]

\[ = c & 2 & 2 & 0 & 8 \]

Matlab code: \( c = a ./ b \)

Vectorized element-by-element arithmetic operations between an array and a scalar

\[ \begin{array}{c}
a & 2 & 1.5 & 8 \\
x & \times & b & 1 & 2 & 0 & 1 \\
\end{array} \]

\[ = c & 2 & 2 & 0 & 8 \]

A dot (.) is necessary in front of these math operators

Color is a 3-vector, sometimes called the RGB values

- Any color is a mix of red, green, and blue
- Example: \( \text{colr} = [0.4 \ 0.6 \ 0] \)
- Each component is a real value in [0,1]
- \([0 \ 0 \ 0]\) is black
- \([1 \ 1 \ 1]\) is white
- \([2 \ 2 \ 2]\) is dark gray
- \([4 \ .6 \ .1]\) is a colorized hue
Mix two colors
Implement this function:

```matlab
function newc = mixEqual(c1,c2)
% Average colors c1 and c2.
% c1, c2, and newc are vectors.
% Display the three colors.
```

Let's show the “paint chips” from white to black

Name the script `white2black`

---

**I-d array: vector**
- An array is a named collection of like data organized into rows or columns
- A 1-d array is a row or a column, called a vector
- An index identifies the position of a value in a vector

<table>
<thead>
<tr>
<th>score</th>
<th>93</th>
<th>92</th>
<th>87</th>
<th>0</th>
<th>90</th>
<th>82</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**Array index starts at 1**

Let \( k \) be the index of vector \( x \), then
- \( k \) must be a positive integer
- \( 1 \leq k \leq \text{length}(x) \)
- To access the \( k \)th element: \( x(k) \)

---

**Accessing values in a vector**

<table>
<thead>
<tr>
<th>score</th>
<th>93</th>
<th>99</th>
<th>87</th>
<th>80</th>
<th>85</th>
<th>82</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Given the vector `score` ...

```matlab
score(4)= 80;
score(5)= (score(4)+score(5))/2;
k= 1;
score(k+1)= 99;
```

---

**A few different ways to create a vector**
(More later!)

```matlab
count= zeros(1,6)
```

```matlab
x= linspace(10,30,5)
```

```matlab
y= [3 7 2 1]
```

```matlab
z= [3; 7; 2]
```
Drawing a single line segment
\[
\begin{align*}
\text{a} &= 0; & \text{% x-coord of pt 1} \\
\text{b} &= 1; & \text{% y-coord of pt 1} \\
\text{c} &= 5; & \text{% x-coord of pt 2} \\
\text{d} &= 3; & \text{% y-coord of pt 2} \\
\text{plot}([a \ c], [b \ d], '-*')
\end{align*}
\]

Drawing polygon (multiple line segments)
\[
\begin{align*}
% & \text{Draw a rectangle with the lower-left} \\
% & \text{corner at (a,b), width w, height h.} \\
x &= [\text{ ]; } & \text{% x data} \\
y &= [\text{ ]; } & \text{% y data} \\
\text{plot}(x, y)
\end{align*}
\]

Example
- Write a program fragment that calculates the cumulative sums of a given vector \(v\).
- The cumulative sums should be stored in a vector of the same length as \(v\).

A twinkling constellation
- Write a script that generates 9 random positions—the configuration of my constellation
- Simulate 10 rounds of twinkling
  - In each round, each star is equally likely to be lit or black
  - Can you add some random adjustment to the color of the star?