Tables

More matrix fun

CS112 Scientific Computation
Department of Computer Science
Wellesley College

Remember from last time…

We refer to individual locations of a matrix using two indices that specify the row and column

```
nums = [ 1   2   3   4   5;  6   7   8   9  10; ...
          11  12  13   0  15;  16  17  18  19  20];
val = nums(2, 3);
nums(3, 4) = 14;
```

nums	1	2	3	4	5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	0	15
4	16	17	18	19	20

val

Indexing with colon notation

To refer to an *entire column* of a matrix, provide **:** as the first index and the column number as the second index

```
>> nums(:, 3)
ans =
      3
      8
     13
     18
```

nums	1	2	3	4	5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20

To refer to an *entire row* of a matrix, provide **:** as the second index and the row number as the first index

```
>> nums(2, :)
ans =
     6  7   8   9  10
```


Analyzing table data

level	1998	1999	2000	2001	2002	2003	2004	2005
advanced	7	9	15	18	20	24	29	35
proficient	17	15	18	27	24	27	28	27
needs improvement	24	23	22	30	31	29	28	24
failing	52	53	45	25	25	20	15	15

Table 1. Statewide results for MCAS Test in Mathematics, Grade 10

Plotting trends in performance levels

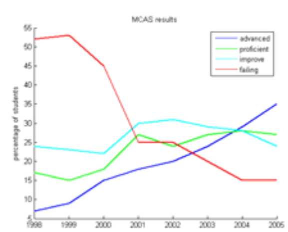
We begin our analysis by plotting the data for each performance level over the 8 years

```
% create matrices that store data and years
results =  [ 7   9  15  18  20  24  29  35; ...
            17  15  18  27  24  27  28  27; ...
            24  23  22  30  31  29  28  24; ...
            52  53  45  25  25  20  15  15];
years = [1998 1999 2000 2001 2002 2003 2004 2005];
```

Each row of the table corresponds to a performance level. How do we plot the resulting trend over the given years?

Plotting the data

```
% plot the data for each performance level vs. years
hold on
plot(years, results(1,:), 'b', 'LineWidth', 2);
plot(years, results(2,:), 'g', 'LineWidth', 2);
plot(years, results(3,:), 'c', 'LineWidth', 2);
plot(years, results(4,:), 'r', 'LineWidth', 2);
hold off
```

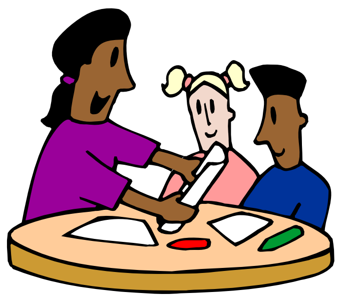



```
xlabel('year')
ylabel('percentage of students')
title('MCAS results')
legend('advanced', 'proficient', 'improve', 'failing' );
```

Finally, ...

Suppose we want to print the *change in results* between 1998 and 2005 for each performance level...

How do we do this?



Printing changes in results

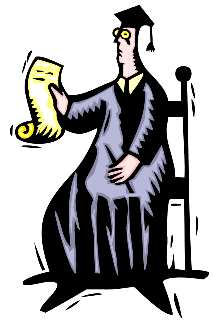
```
% print total change in results between 1998 and 2005
totalChange = results(:, end) - results(:, 1);
disp('Change in performance between 1998 and 2005:');
disp(['advanced: ' num2str(totalChange(1)) '%']);
disp(['proficient: ' num2str(totalChange(2)) '%']);
disp(['needs improvement: ' num2str(totalChange(3)) '%']);
disp(['failing: ' num2str(totalChange(4)) '%']);
```

```
Change in performance between 1998 and 2005:
advanced: 28%
proficient: 10%
needs improvement: 0%
failing: -37%
```

Time-out exercise

For each year, compute a *weighted sum* of the four percentages, using a weight of 1 for "advanced", 2 for "proficient", 3 for "needs improvement" and 4 for "failing"*

overallPerformance =



Add a new row to the **results** matrix that stores these weighted sums

* The resulting sum can range from 100 (great!) to 400 (not so good…)

More indexing with colon notation

We can use colon notation to refer to a *range of indices* within a column or row of a matrix

```
>> nums(1:3, 4)
ans =
      4
      9
     14
>> nums(3, 3:5)
ans =
     13  14  15
>> nums(2:3, 2:4)
ans =
      7   8   9
     12  13  14
```

nums	1	2	3	4	5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20

Conditional operations on matrices

A conditional expression can be *applied to an entire matrix all at once* producing a new matrix of the same size that contains logical values

```
ages = [13  52  19  21;  18  47  23  15;  60  38  16  12];
teens = (ages >= 13) & (ages <= 19);
```

ages

13	52	19	21
18	47	23	15
60	38	16	12

teens

1	0	1	0
1	0	0	1
0	0	1	0

Using logical vectors

```
>> ages(teens) = 0
ages =
     0   52    0   21
     0   47   23    0
    60   38    0   12

>> overTheHill = ages(ages>40)
overTheHill =
    60
    52
    47
```

ages

13	52	19	21
18	47	23	15
60	38	16	12

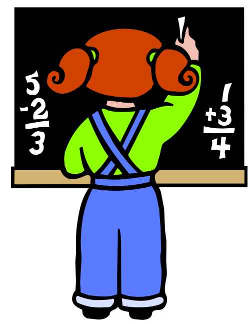
teens

1	0	1	0
1	0	0	1
0	0	1	0

Time-out exercise

Given the original **ages** matrix, write two statements that each assign the variable **numAdults** to the total number of age values that are 18 or over

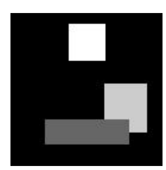
One statement should use **sum** and the other should use **length**



Creating synthetic images

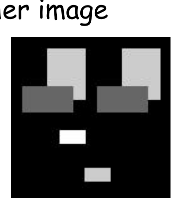
Using colon notation, we can create a synthetic image that contains patches of constant brightness

```
image = zeros(128, 128);
image(10:40, 50:80) = 1.0;
image(60:100, 80:115) = 0.8;
image(90:110, 30:100) = 0.4;
```



Copy rectangular regions of one image into another image

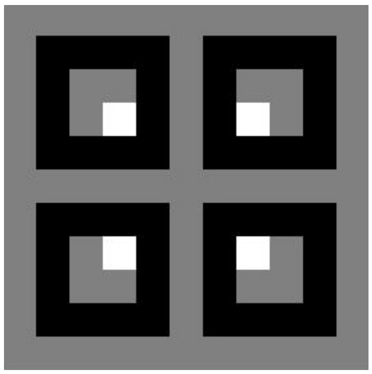
```
patch = image(60:110, 60:110);
newImage = zeros(128,128);
newImage(10:60, 10:60) = patch;
newImage(10:60, 70:120) = patch;
```



```
newImage(75:115, 40:80) = image(30:70, 60:100);
```

Let's make a quilt

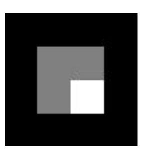
Suppose we want to create the following image -
How do we *plan* the code?



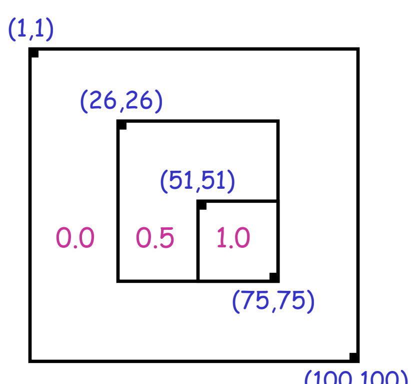
We can begin by making a single patch:

Making the patch

First draw a picture of the pattern, with coordinates of key points and brightness values:



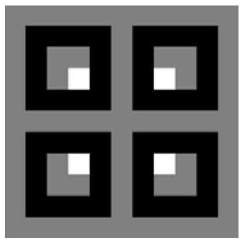
patch



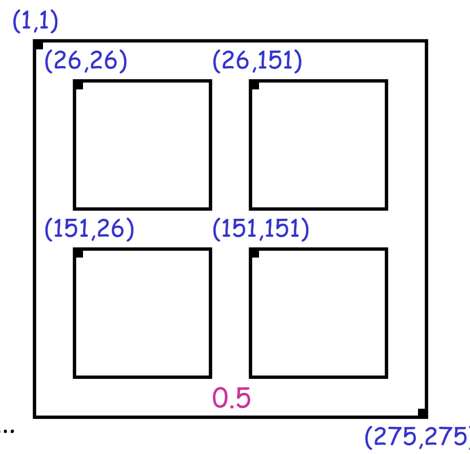
Then write the code…

Planning the full quilt

Again, draw a picture of the pattern first:

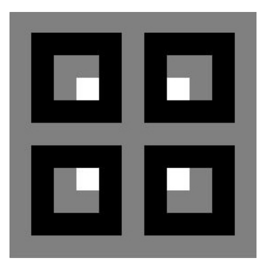


quilt



Then write the code…

The final code



```
% create the patch
patch = zeros(100,100);
patch(26:75, 26:75) = 0.5;
patch(51:75, 51:75) = 1.0;

% assemble the quilt
quilt = 0.5 * ones(275, 275);
quilt(26:125, 26:125) = patch;
quilt(26:125, 151:250) = patch(:, 100:-1:1);  % flip columns
quilt(151:250, 26:125) = patch(100:-1:1, :);  % flip rows
quilt(151:250, 151:250) = patch(100:-1:1, 100:-1:1);

% display the quilt
imshow(quilt);
```