Day 1:
* The development environment
* Variables
* Arithmetic operators
* Logical operators
* The command window
* Writing and running scripts
* if else statements
* for loops
While loops

while <expression>
  <statements>
end
Same result with for loop and while loop

```matlab
% while loop execution
while( a < 20 )
  fprintf('value of a: %d\n', a);
  a = a + 1;
end
% for loop execution
for b=10:20
  fprintf('value of b: %d\n', b);
end
```
break statements

whileLoop.m

1. a = 10;
2. % while loop execution
3. while (a < 20)
4.   fprintf('value of a: %d\n', a);
5.   a = a+1;
6.   if (a > 15)
7.     % terminate the loop using break statement
8.     break;
9.   end
10. end

Command Window

>> whileLoop
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15

Reference the ‘i’th element of a vector with v(i)
A little more about vectors

v(:)
Gets all elements in the vector

V(3:6)
Gets the 3\textsuperscript{rd} to the 6\textsuperscript{th} element
v(m,n)
Gets the element at $m^{th}$ row and $n^{th}$ column
A little more about matrices

Indexing can be combined with ‘:’
Convenient matrix creation

```matlab
>> zeros(3)
ans =
  0   0   0
  0   0   0
  0   0   0

>> ones(2,4)
ans =
  1   1   1   1
  1   1   1   1
```
Higher dimensional Matrices

```matlab
>> myMatrix=zeros(4,4,3)

myMatrix(:,:,1) =
    0   0   0   0
    0   0   0   0
    0   0   0   0
    0   0   0   0

myMatrix(:,:,2) =
    0   0   0   0
    0   0   0   0
    0   0   0   0
    0   0   0   0

myMatrix(:,:,3) =
    0   0   0   0
    0   0   0   0
    0   0   0   0
    0   0   0   0
```
How would you change this element to a 7?
Higher dimensional Matrices

What does myMatrix look like after this command?

```matlab
>> myMatrix(4,3,2)=7
myMatrix(:,1,1) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
myMatrix(:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 7 0
myMatrix(:,3) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
>> myMatrix(:,:,3)=ones(4)
```
Higher dimensional Matrices

```matlab
>> myMatrix(:, :, 3) = ones(4)
myMatrix(:, :, 1) =
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0

myMatrix(:, :, 2) =
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
    0     0     7     0     0

myMatrix(:, :, 3) =
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
    1     1     1     1     1
```
Sorting

>> sortingTest

v =

23  45  12  9  5  0  19  17

ans =

0   5   9  12  17  19  23  45

m =

2   6   4
5   3   9
2   0   1

ans =

2   0   1
2   3   4
5   6   9

ans =

2   4   6
3   5   9
0   1   2
Cell Arrays – like matrices but more awesome

We’ve seen an example before!

Why did we use a cell array for gene names here?
Cell Arrays – like matrices but more awesome

Matrices are bad for Strings

```matlab
>> geneNames = ['IL1A', 'NFKBIA', 'BCL2', 'STAT5A', 'CCL5']
geneNames =
IL1ANFKBIABCL2STAT5ACCL5

>> geneNames = ['IL1A'; 'NFKBIA'; 'BCL2'; 'STAT5A'; 'CCL5']
Error using `vertcat`
Dimensions of matrices being concatenated are not consistent.

>> geneNames = ['IL1A', '; 'NFKBIA'; 'BCL2', '; 'STAT5A'; 'CCL5 ']
geneNames =
IL1A
NFKBIA
BCL2
STAT5A
CCL5
```
Cell Arrays – like matrices but more awesome

Can be initiated with
\[ \text{cell}(m,n) \]

```matlab
>> c = cell(2, 5);
>> c = {'IL1A', 'NFkBIA', 'BCL2', 'STAT5A', 'CCL5'; 4, 56, 21, 12, 39}

\[
\begin{array}{cccccc}
\text{'IL1A'} & \text{'NFkBIA'} & \text{'BCL2'} & \text{'STAT5A'} & \text{'CCL5'} \\
\end{array}
\]
```
Cell Arrays – beware of the difference between () and {}.

c(m,n) refers to sets of cells

c{m,n} refers to the data within the cells

```
c = [ 'IL1A' 'NFKBIA' 'BCL2' 'STAT5A' 'CCL5' ;
     [ 4]    [ 56]    [21]    [12]    [39] ]
>> c(:,2:3)
ans =
    'NFKBIA'    'BCL2'
    [ 56]    [ 21]

>> c{:,2:3}
ans =
    NFKBIA
    56
    BCL2
    21
```
String Comparisons

```matlab
exp1Genes = {'IL1A', 'NFKBIA', 'BCL2', 'STAT5A', 'CCL5', 'MYC', 'IRF1'};
exp1Results = [4, 56, 21, 12, 39, 21, 3];
threshold = 20;
geneOfInterest = 'IRF1';
for i=1:length(exp1Results)
    if(exp1Results(i)>threshold)
        fprintf('Gene %s expression %d\n', exp1Genes{i}, exp1Results(i));
    end
    if(strcmp(exp1Genes{i}, geneOfInterest))
        fprintf(strcat('Expression of ', geneOfInterest, ' %d\n'), exp1Results(i));
    end
end
```

Command Window

```bash
>> forLoops
Gene NFKBIA expression 56
Gene BCL2 expression 21
Gene CCL5 expression 39
Gene MYC expression 21
Expression of IRF1 3
```
Time for Functions!

* 3 Ways of running code
  * Command Window
  * Scripts
  * Functions

m-files
function [outputs] = functionName (inputs)

Sometimes inputs are called “arguments”.
Our First Function

```matlab
function outSeq = reverseSeq(seq)
    %This function returns the reverse of an input sequence
    outSeq = seq(end:-1:1);
```

Command Window

```matlab
>> edit reverseSeq.m
>> reverseSeq('TCGCG')
```

```
ans =
GCGCT
```
Convert our Script to a Function

```
forLoops.m

exp1Genes = {'IL1A', 'NFKBIA', 'BCL2', 'STAT5A', 'CCL5', 'MYC', 'IRF1'};
exp1Results = [4, 56, 21, 12, 39, 21, 3];
threshold = 20;
geneOfInterest = 'IRF1';
for i = 1:length(exp1Results)
    if(exp1Results(i) > threshold)
        fprintf('Gene %s expression %d \n', exp1Genes{i}, exp1Results(i));
    end
    if(strcmp(exp1Genes{i}, geneOfInterest))
        fprintf(strcat('Expression of ', geneOfInterest, ', %d \n'), exp1Results(i));
    end
end
```

printGeneExpression.m

```
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)

    % Function prints gene expression information for a list of genes over a given threshold
    % Returns the expression value from a gene of interest
    out = 'Gene Not Found';
    for i = 1:length(exp1Results)
        if(exp1Results(i) > threshold)
            fprintf('Gene %s expression %d \n', exp1Genes{i}, exp1Results(i));
        end
        if(strcmp(exp1Genes{i}, geneOfInterest))
            fprintf(strcat('Expression of ', geneOfInterest, ', %d \n'), exp1Results(i));
            out = exp1Results(i);
        end
    end
end
```
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)

    % Function prints gene expression information for a list of genes over a given threshold
    % Returns the expression value from a gene of interest
    out='Gene Not Found';
    for i=1:length(exp1Results)
        if(exp1Results(i)>threshold)
            fprintf('Gene %s expression %d\n',exp1Genes{i},exp1Results(i));
        end
        if(strcmp(exp1Genes{i},geneOfInterest))
            fprintf(strcat('Expression of ',geneOfInterest,' %d\n'),exp1Results(i));
            out = exp1Results(i);
        end
    end
end

>> genes = {'IL1A','NFKBIA','BCL2','STAT5A','CCL5','MYC','IRF1'};
>> values = [4,56,21,12,39,21,3];
>> geneOfInterestValue = printGeneExpression(genes,values,'IRF1',20)
Gene NFKBIA expression 56
Gene BCL2 expression 21
Gene CCL5 expression 39
Gene MYC expression 21
Expression ofIRF1 3

geneOfInterestValue =

3
Getting Help

```python
>> help printGeneExpression
Function prints gene expression information for a list of genes over a given threshold
Returns the expression value from a gene of interest
```

```python
>> help max
max    Largest component.
    For vectors, max(X) is the largest element in X. For matrices, max(X) is a row vector containing the maximum element from each column. For N-D arrays, max(X) operates along the first non-singleton dimension.
```
Working with files

Command Window

```matlab
>> url='http://www.signalingsystems.ucla.edu/users/Simon/example.jpg';
>> imageFile='example.jpg';
>> myImage=urlwrite(url,imageFile);
>> imshow(myImage);

>> clear all;
>> A=importdata('example.jpg');
>> imshow(A);
```
Working with (more sensible) files

Download example file: http://www.signalingsystems.ucla.edu/users/Simon/experiment1.txt

```matlab
function out = printGeneExpressionFromFile(inputFileName, geneOfInterest, threshold)
    %Function prints gene expression information for a list of genes from a file
    %Prints gene information for genes over a given threshold
    %Returns the expression value from a gene of interest
    delimiter='\t'; %tab delimiter
    headerLines=1; %Titles are on first line

    A=importdata(inputFileName,delimiter,headerLines);
    exp1Genes=A.textdata(2:end,1);
    exp1Results=A.data;

    fprintf('\%s loaded. \%d genes.',inputFileName,length(A.data));

    %Call function to print gene information
    out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold);
```
Working with (more sensible) files

Download example file:
http://www.signalingsystems.ucla.edu/users/Simon/experiment1.txt

```
function out = printGeneExpressionFromFile(inputFileName, geneOfInterest, threshold)
    %Function prints gene expression information for a list of genes from a file
    %prints gene information for genes over a given threshold
    %Returns the expression value from a gene of interest
    delimiter='\t'; %tab delimiter
    headerLines=1; %titles are on first line

    A=importdata(inputFileName,delimiter,headerLines);
    exp1Genes=A.textdata(2:end,1);
    exp1Results=A.data;

    fprintf('\s loaded. %d genes.\n',inputFileName,length(A.data));

    %call function to print gene information
    out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold);
```

```
>> printGeneExpressionFromFile('experiment1.txt','CCL5',20)
```
Tidying up output

Gene IL-23A (p19) expression 2.761789e+01
Gene IL-27 (p28) expression 2.827399e+01
Gene EBI3/IL-27B expression 3.333691e+01
Gene beta-Interferon expression 2.658134e+01

Gene IL-23A (p19) expression 27.62
Gene IL-27 (p28) expression 28.27
Gene EBI3/IL-27B expression 33.34
Gene beta-Interferon expression 26.58

---

```matlab
function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)
    
    % Function prints gene expression information for a list of genes over a given threshold
    % Returns the expression value from a gene of interest
    out = 'Gene Not Found';
    for i=1:length(exp1Results)
        if(exp1Results(i)>threshold)
            fprintf('Gene %s expression %.2f\n',exp1Genes{i},exp1Results(i));
        end
        if(strcmp(exp1Genes{i},geneOfInterest))
            fprintf(strcat('Expression of ',geneOfInterest,' %.2f\n'),exp1Results(i));
            out = exp1Results(i);
        end
    end
```
Writing to files

```c
fileIdentifier = fopen('filename','w');

fprintf(fileIdentifier, 'someText');

fclose(fid);
```
Changing our functions to write to a file

```matlab
function out = printGeneExpressionFromFile(inputFileName, outputFileName, geneOfInterest, threshold)

% Function prints gene expression information for a list of genes from a file
% prints gene informaton for genes over a given threshold
% Returns the expression value from a gene of interest
delimiter='\t'; % tab delimiter
headerLines=1; % titles are on first line

A=importdata(inputFileName,delimiter,headerLines);
exp1Genes=A.textdata(2:end,1);
exp1Results=A.data;

fprintf('%s loaded. %d genes.\n',inputFileName,length(A.data));

% call function to print gene information
out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold, outputFileName);
```
Changing our functions to write to a file

function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold, outputFileName)
    %Function prints gene expression information for a list of genes over a
    %given threshold
    %Returns the expression value from a gene of interest
    fid=fopen(outputFileName,'w');
    out='Gene Not Found';
    for i=1:length(exp1Results)
        if(exp1Results(i)>threshold)
            fprintf(fid,'%s,%.2f
',exp1Genes(i),exp1Results(i));
        end
        if(strcmp(exp1Genes{i},geneOfInterest))
            fprintf(strcat('Expression of ',geneOfInterest,' %.2f
'),exp1Results(i));
            out = exp1Results(i);
        end
    end
    fclose(fid);

>> printGeneExpressionFromFile('experiment1.txt','experimentOutput.txt','CCL5',20)
experiment1.txt loaded. 59 genes.
Expression ofCCL5 3.84

ans =

    3.84

Open ‘experimentOutput.txt’ in excel