

An Introduction to MATLAB

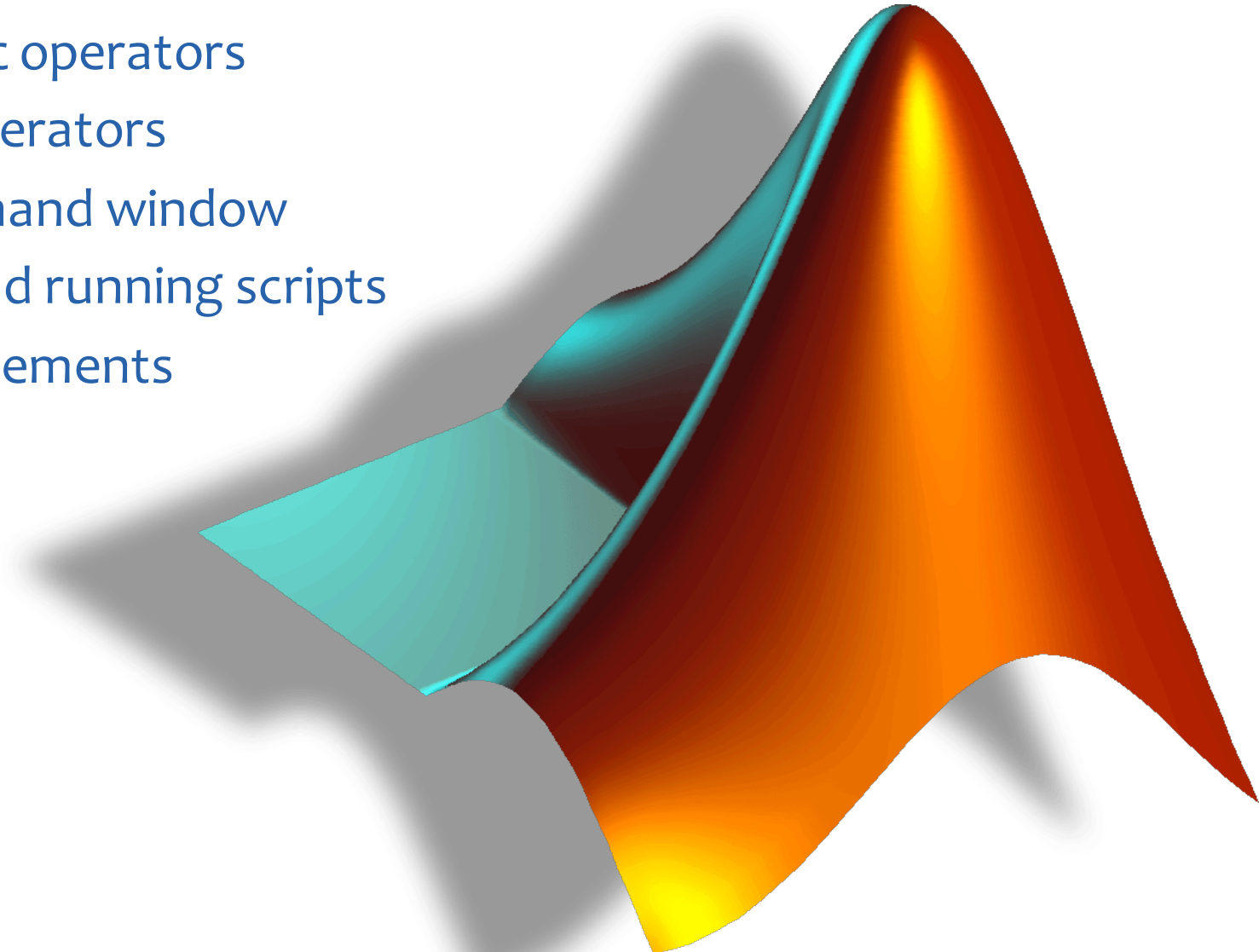
Day 2

Simon Mitchell

Simon.Mitchell@ucla.edu

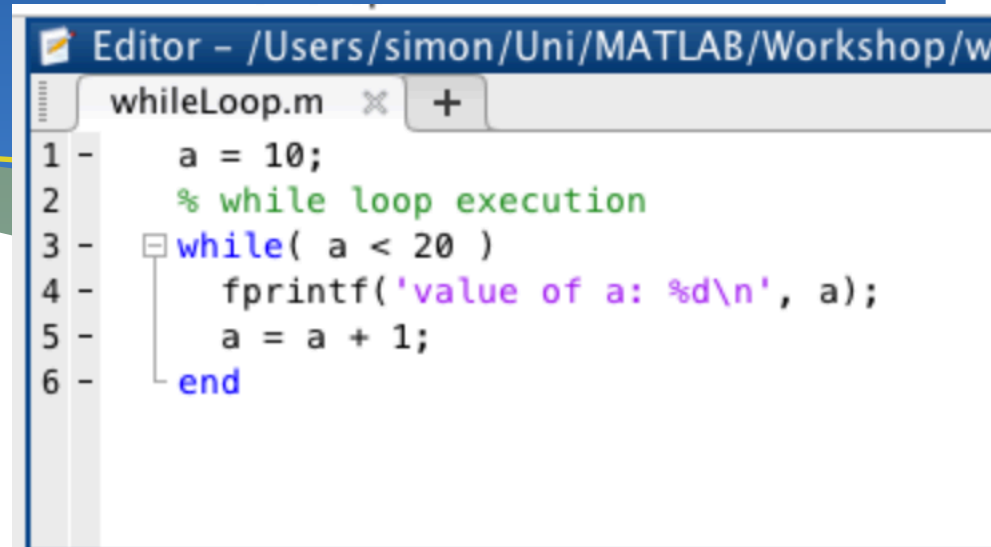
Day 1:

- * The development environment
- * Variables
- * Arithmetic operators
- * Logical operators
- * The command window
- * Writing and running scripts
- * if else statements
- * for loops



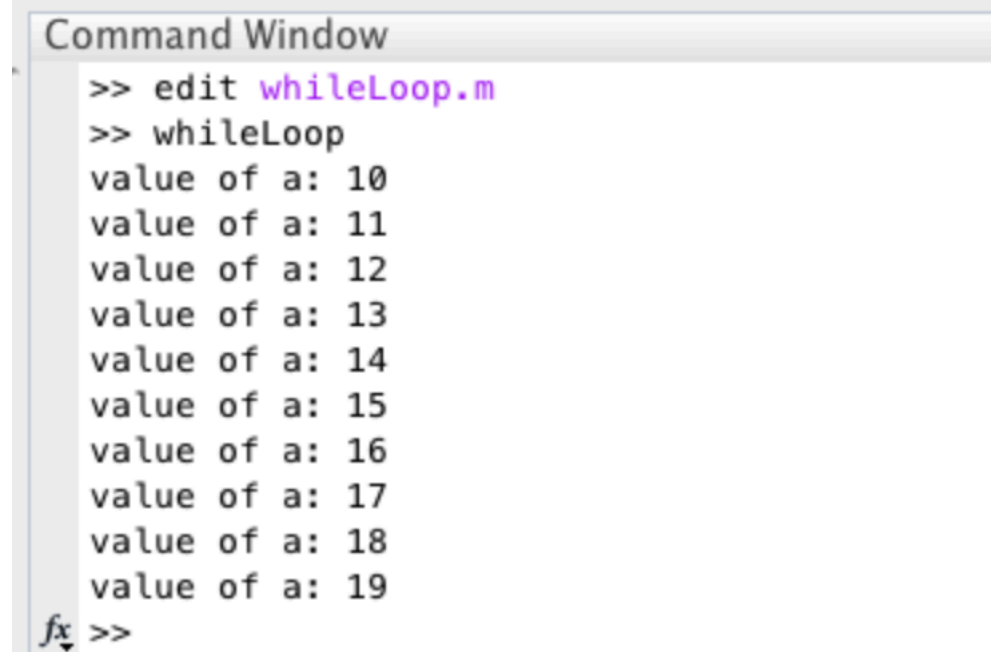
While loops

```
while <expression>  
    <statements>  
end
```



The image shows a MATLAB Editor window with a single tab titled 'whileLoop.m'. The code inside the tab is as follows:

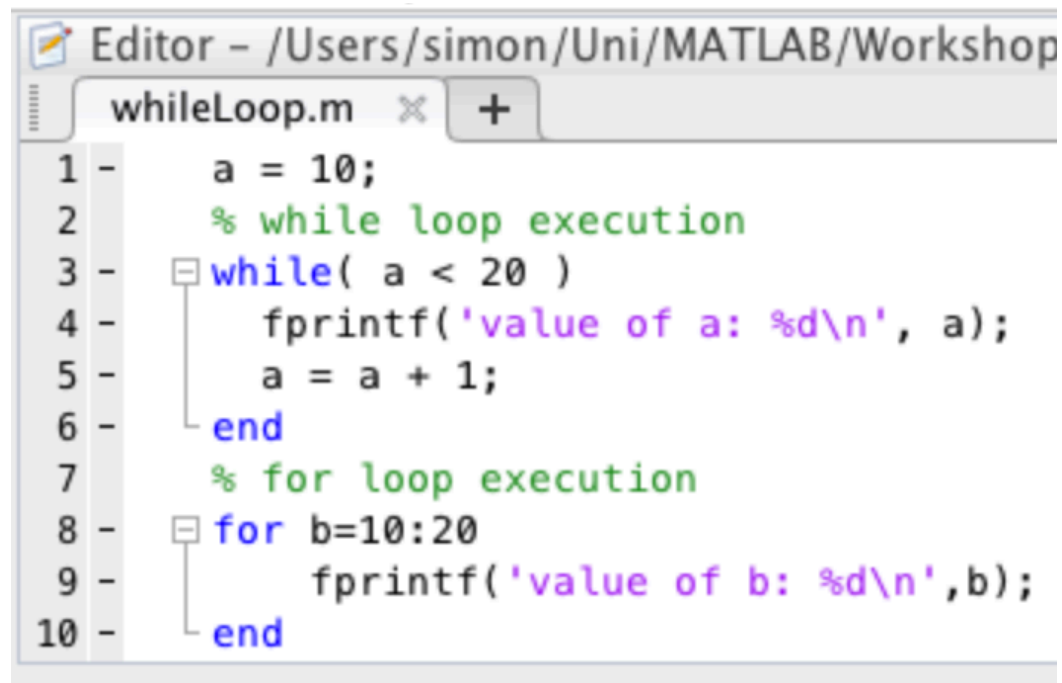
```
1 - a = 10;  
2 - % while loop execution  
3 - while( a < 20 )  
4 -     fprintf('value of a: %d\n', a);  
5 -     a = a + 1;  
6 - end
```



The image shows the MATLAB Command Window with the following text:

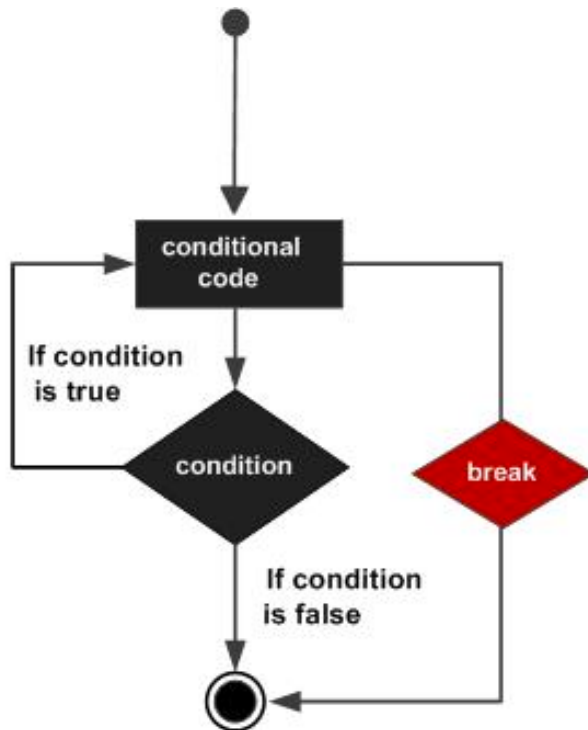
```
>> edit whileLoop.m  
>> whileLoop  
value of a: 10  
value of a: 11  
value of a: 12  
value of a: 13  
value of a: 14  
value of a: 15  
value of a: 16  
value of a: 17  
value of a: 18  
value of a: 19  
fx >>
```

Same result with for loop and while loop



```
Editor - /Users/simon/Uni/MATLAB/Workshop
whileLoop.m x +
1 - a = 10;
2 - % while loop execution
3 - while( a < 20 )
4 -     fprintf('value of a: %d\n', a);
5 -     a = a + 1;
6 - end
7 - % for loop execution
8 - for b=10:20
9 -     fprintf('value of b: %d\n',b);
10 - end
```

break statements



Editor - /Users/simon/Uni/MATLAB/Workshop/whileLoop.m

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
```

```
fx >> |
```

A little more about vectors

Reference the 'i'th
element of a vector
with
 $v(i)$

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

```
v =
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
>> v(3)
```

```
ans =
```

?

A little more about vectors

`v(:)`

Gets all elements in the vector

`v(3:6)`

Gets the 3rd to the 6th element

Command Window

```
>> v(:)
```

```
ans =
```

```
1  
2  
3  
4  
5  
6
```

```
>> v(3:6)
```

```
ans =
```

```
3  
4  
5  
6
```

fx >>

A little more about matrices

$v(m,n)$

Gets the element at
 m^{th} row and n^{th} column

Command Window

```
>> a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
```

```
>> a
```

```
a =
```

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 |
| 3 | 4 | 5 | 6 | 7 |
| 4 | 5 | 6 | 7 | 8 |

```
>> a(2,5)
```

```
ans =
```

```
6
```

```
fx >> |
```


A little more about matrices

Command Window

a =

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 |
| 3 | 4 | 5 | 6 | 7 |
| 4 | 5 | 6 | 7 | 8 |

```
>> a(:,4)
```

ans =

| |
|---|
| 4 |
| 5 |
| 6 |
| 7 |

```
>> a(4,:)
```

ans =

| | | | | |
|---|---|---|---|---|
| 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|

fx >> |

Indexing can be combined
with ':'

```
>> a(2:3,2:3)
```

ans =

| | |
|---|---|
| 3 | 4 |
| 4 | 5 |

Convenient matrix creation

```
>> 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

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

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

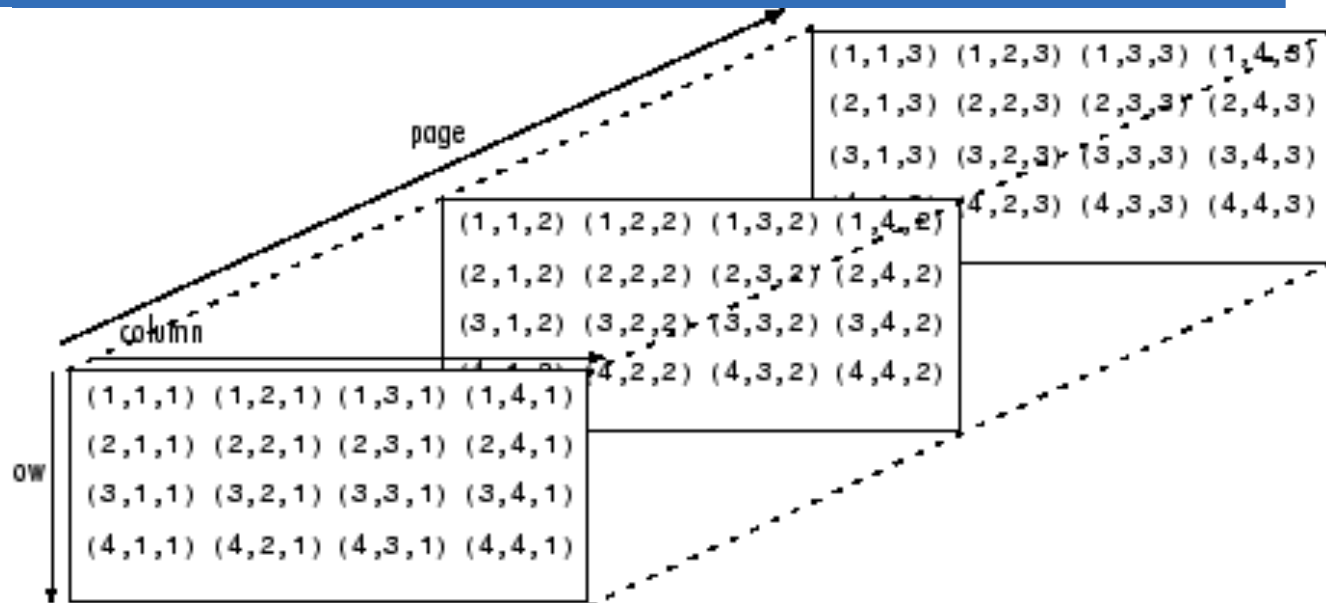
```
0    0    0    0
0    0    0    0
0    0    0    0
0    0    0    0
```

```
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
```



Higher dimensional Matrices

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

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

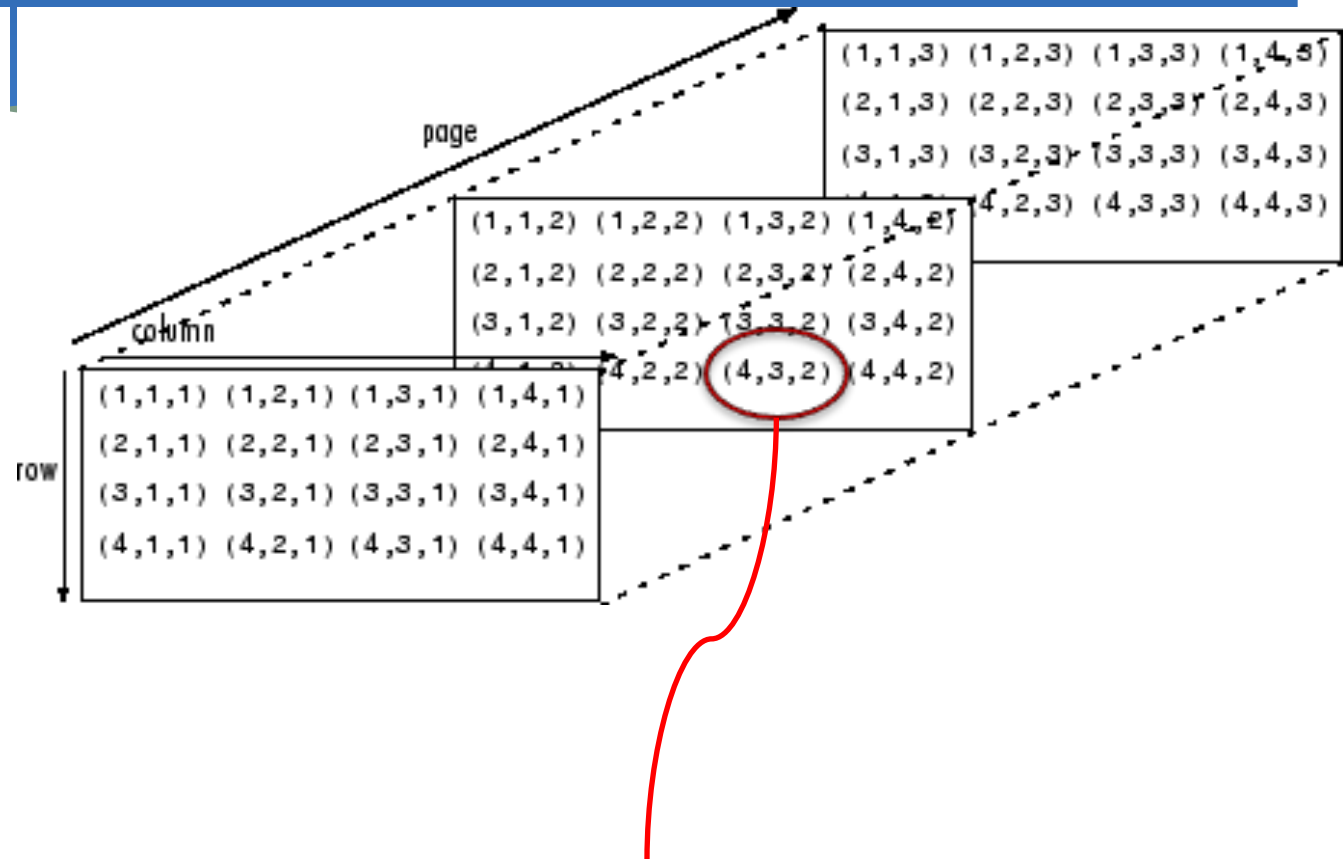
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

```
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

```
>> myMatrix(4,3,2)=7
```

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

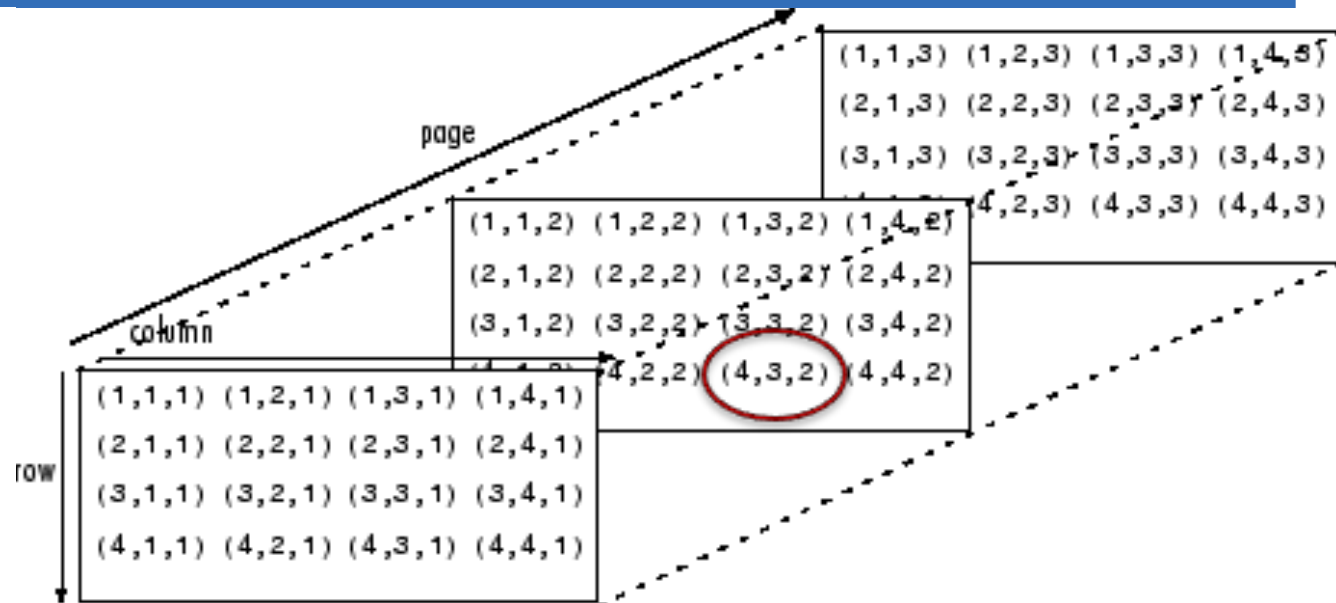
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

```
myMatrix(:,:,2) =
```

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 7 | 0 |

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

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |



What does myMatrix look like after this command?

```
>> myMatrix(:,:,3)=ones(4)
```

Higher dimensional Matrices

```
>> myMatrix(:,:,3)=ones(4)
```

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

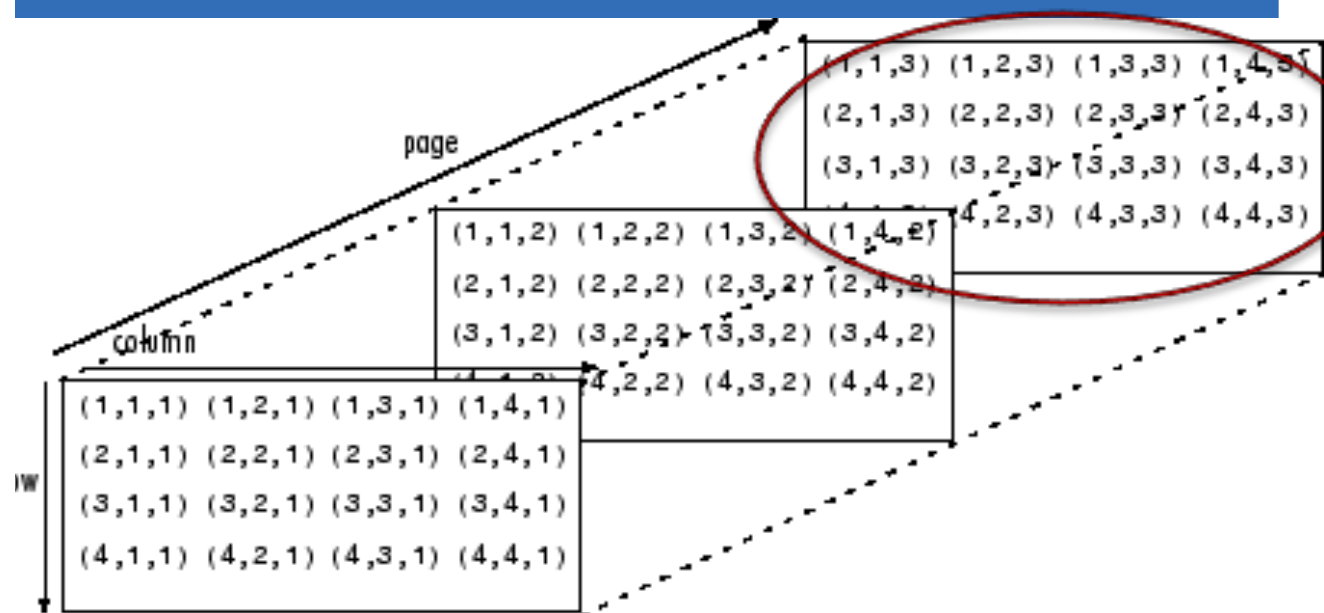
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

```
myMatrix(:,:,2) =
```

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 7 | 0 |

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

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |



Sorting

```
sortingTest.m ✕ +
1 - v = [ 23 45 12 9 5 0 19 17]
2 - sort(v)
3 - m = [2 6 4; 5 3 9; 2 0 1]
4 - sort(m, 1)
5 - sort(m, 2)
```

```
>> 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 =
```

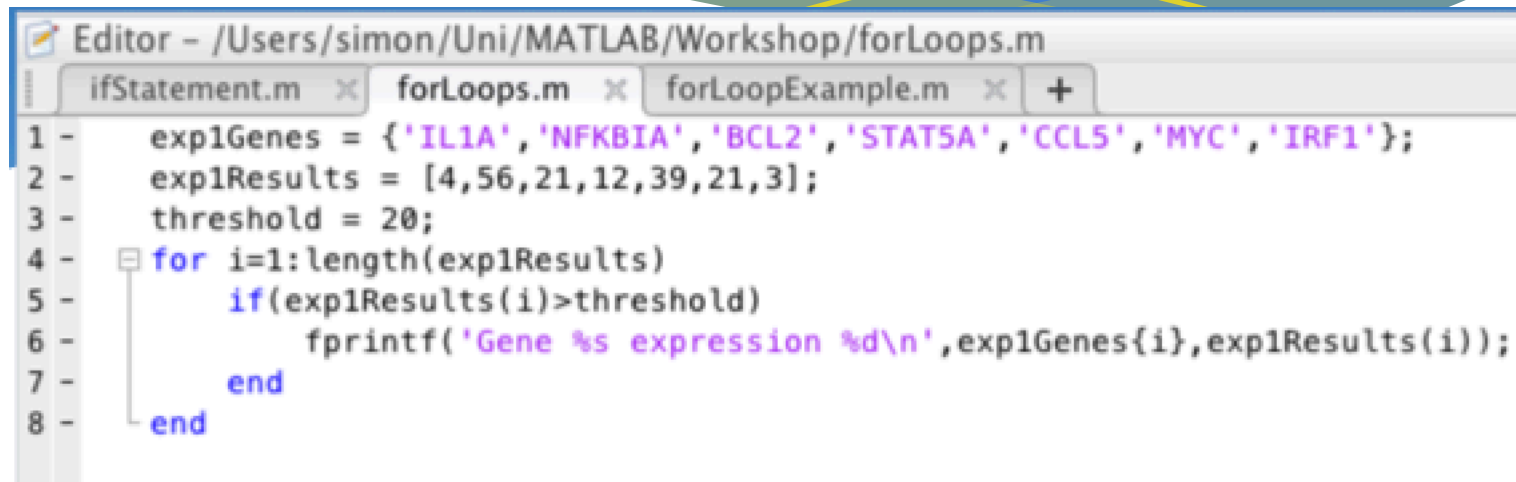
```
     2     0     1
     2     3     4
     5     6     9
```

```
ans =
```

```
     2     4     6
     3     5     9
     0     1     2
```

Cell Arrays – like matrices but more awesome

We've seen an example before!



The image shows a MATLAB Editor window with the title bar 'Editor - /Users/simon/Uni/MATLAB/Workshop/forLoops.m'. There are three tabs open: 'ifStatement.m', 'forLoops.m' (which is the active tab), and 'forLoopExample.m'. The code in the 'forLoops.m' tab is as follows:

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

Why did we use a cell array for gene names here?

Cell Arrays – like matrices but more awesome

Matrices
are bad
for Strings

```
>> 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
`cell(m,n)`

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

```
c =
```

| | | | | |
|--------|----------|--------|----------|--------|
| 'IL1A' | 'NFKBIA' | 'BCL2' | 'STAT5A' | 'CCL5' |
| [4] | [56] | [21] | [12] | [39] |

Cell Arrays – beware of the difference between () and {}

```
c =  
  
    'IL1A'    'NFKBIA'    'BCL2'    'STAT5A'    'CCL5'  
    [    4]    [    56]    [    21]    [    12]    [    39]
```

$c(m,n)$ refers to
sets of cells

```
>> c(:,2:3)
```

```
ans =
```

```
    'NFKBIA'    'BCL2'  
    [    56]    [    21]
```

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

```
>> c{: ,2:3}
```

```
ans =
```

```
NFKBIA
```

```
ans =
```

```
56
```

```
ans =
```

```
BCL2
```

```
ans =
```

```
21
```

String Comparisons

```
Editor - /Users/simon/UniNew/MATLAB/Workshop/forLoops.m
+8 SolveBigAllSpecies.m x initial_parameters8.m x MatFile.m x mlprintjob.m x forLoops.m x +
1 - exp1Genes = {'IL1A', 'NFKBIA', 'BCL2', 'IRF1', 'STAT5A', 'CCL5', 'MYC'};
2 - exp1Results = [4,56,21,3,12,39,21];
3 - threshold = 20;
4 - geneOfInterest='IRF1';
5 - for i=1:length(exp1Results)
6 -     if(exp1Results(i)>threshold)
7 -         fprintf('Gene %s expression %d\n',exp1Genes{i},exp1Results(i));
8 -     end
9 -     if(strcmp(exp1Genes{i},geneOfInterest))
10 -        fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
11 -    end
12 - end
13 - fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
14
```

Command Window

```
>> forLoops
Gene NFKBIA expression 56
Gene BCL2 expression 21
Expression of IRF1: 3
Gene CCL5 expression 39
Gene MYC expression 21
Expression of IRF1: 21
fx >>
```

Time for Functions!

- * 3 Ways of running code

- * Command Window ✓

- * Scripts ✓

- * Functions

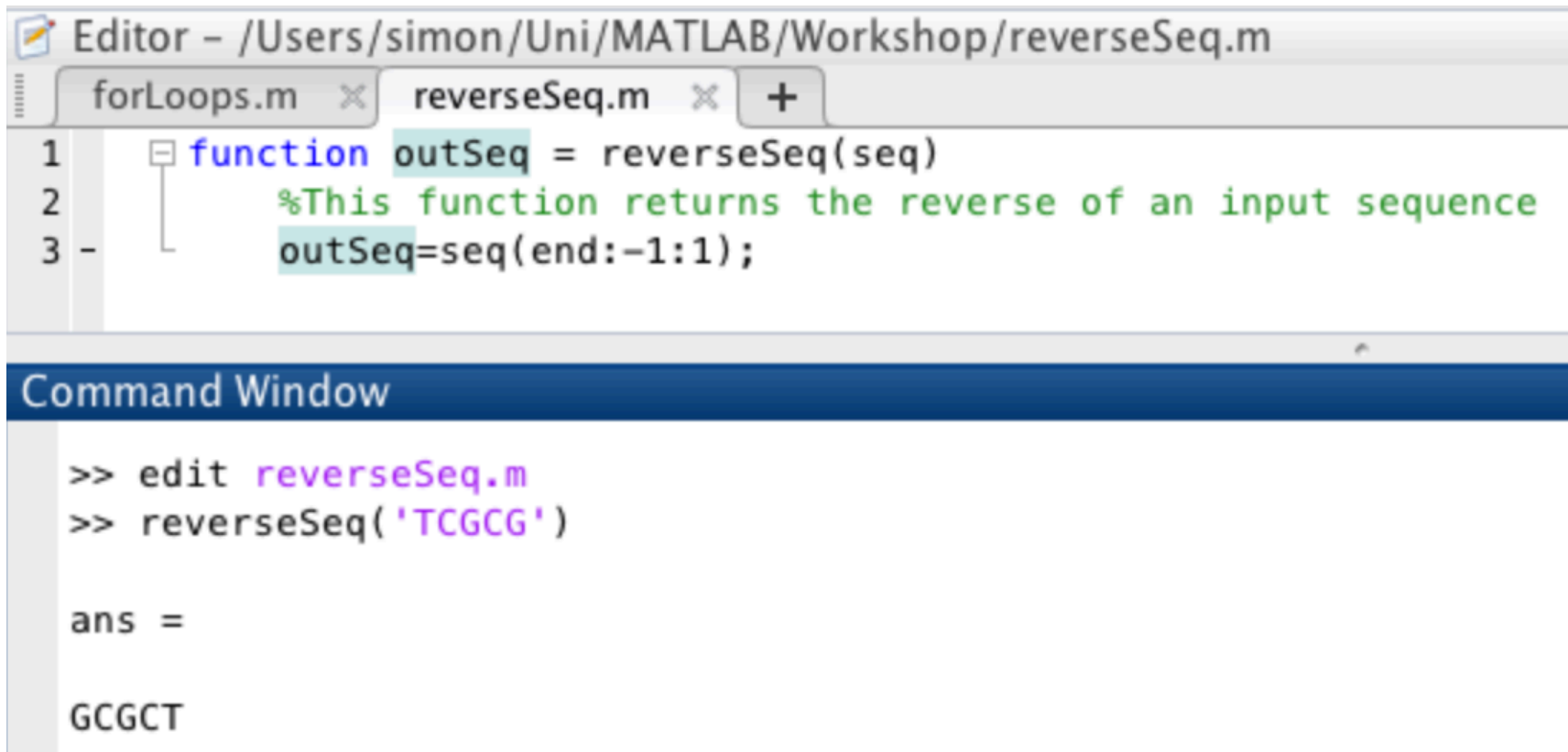
m-files

Defining Functions

`function [outputs] = functionName (inputs)`

Sometimes inputs are called “arguments”.

Our First Function



The image shows a MATLAB environment. The top part is the Editor window, which has two tabs: 'forLoops.m' and 'reverseSeq.m'. The 'reverseSeq.m' tab is active, showing a function definition. The function is named 'reverseSeq' and takes an input 'seq'. It returns 'outSeq', which is the reverse of the input sequence. A comment line explains the function's purpose. The bottom part is the Command Window, which shows the execution of the function. The user enters 'edit reverseSeq.m' to open the file, then 'reverseSeq('TCGCG')' to call the function. The output is 'ans = GCGCT', which is the reverse of the input string 'TCGCG'.

```
Editor - /Users/simon/Uni/MATLAB/Workshop/reverseSeq.m
forLoops.m  reverseSeq.m  +
1  function outSeq = reverseSeq(seq)
2      %This function returns the reverse of an input sequence
3  -  outSeq=seq(end:-1:1);

Command Window

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

ans =

GCGCT
```

```
+8 SolveBigAllSpecies.m x initial_parameters8.m x MatFile.m x mlprintjob.m x forLoops.m x +
1 - exp1Genes = {'IL1A', 'NFKBIA', 'BCL2', 'IRF1', 'STAT5A', 'CCL5', 'MYC'};
2 - exp1Results = [4,56,21,3,12,39,21];
3 - threshold = 20;
4 - geneOfInterest='IRF1';
5 - for i=1:length(exp1Results)
6 -     if(exp1Results(i)>threshold)
7 -         fprintf('Gene %s expression %d\n',exp1Genes{i},exp1Results(i));
8 -     end
9 -     if(strcmp(exp1Genes{i},geneOfInterest))
10 -        fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
11 -    end
12 - end
13 - fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
14 -
```

```
printGeneExpression.m* x printGeneExpressionFromFile.m x forLoops.m x +
1 - function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)
2 -     %Function prints gene expression information for a list of genes over a
3 -     %given threshold
4 -     %Returns the expression value from a gene of interest
5 -     out='Gene Not Found';
6 -     for i=1:length(exp1Results)
7 -         if(exp1Results(i)>threshold)
8 -             fprintf('Gene %s expression %d\n',exp1Genes{i},exp1Results(i));
9 -         end
10 -        if(strcmp(exp1Genes{i},geneOfInterest))
11 -            fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
12 -            out = exp1Results(i);
13 -        end
14 -    end
```



```
1 function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)
2     %Function prints gene expression information for a list of genes over a
3     %given threshold
4     %Returns the expression value from a gene of interest
5     out='Gene Not Found';
6     for i=1:length(exp1Results)
7         if(exp1Results(i)>threshold)
8             fprintf('Gene %s expression %d\n',exp1Genes{i},exp1Results(i));
9         end
10        if(strcmp(exp1Genes{i},geneOfInterest))
11            fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
12            out = exp1Results(i);
13        end
14    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 =
```

Getting Help

```
>> 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
```

```
>> 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.
```

Google!!!

Working with files

```
>> url='http://signalingsystems.ucla.edu/users/Simon/example.jpg';  
>> imageFile='example.jpg';  
>> urlwrite(url,imageFile);  
>> A=imread(imageFile);  
>> image(A);
```

Working with (more sensible) files

Download example file:

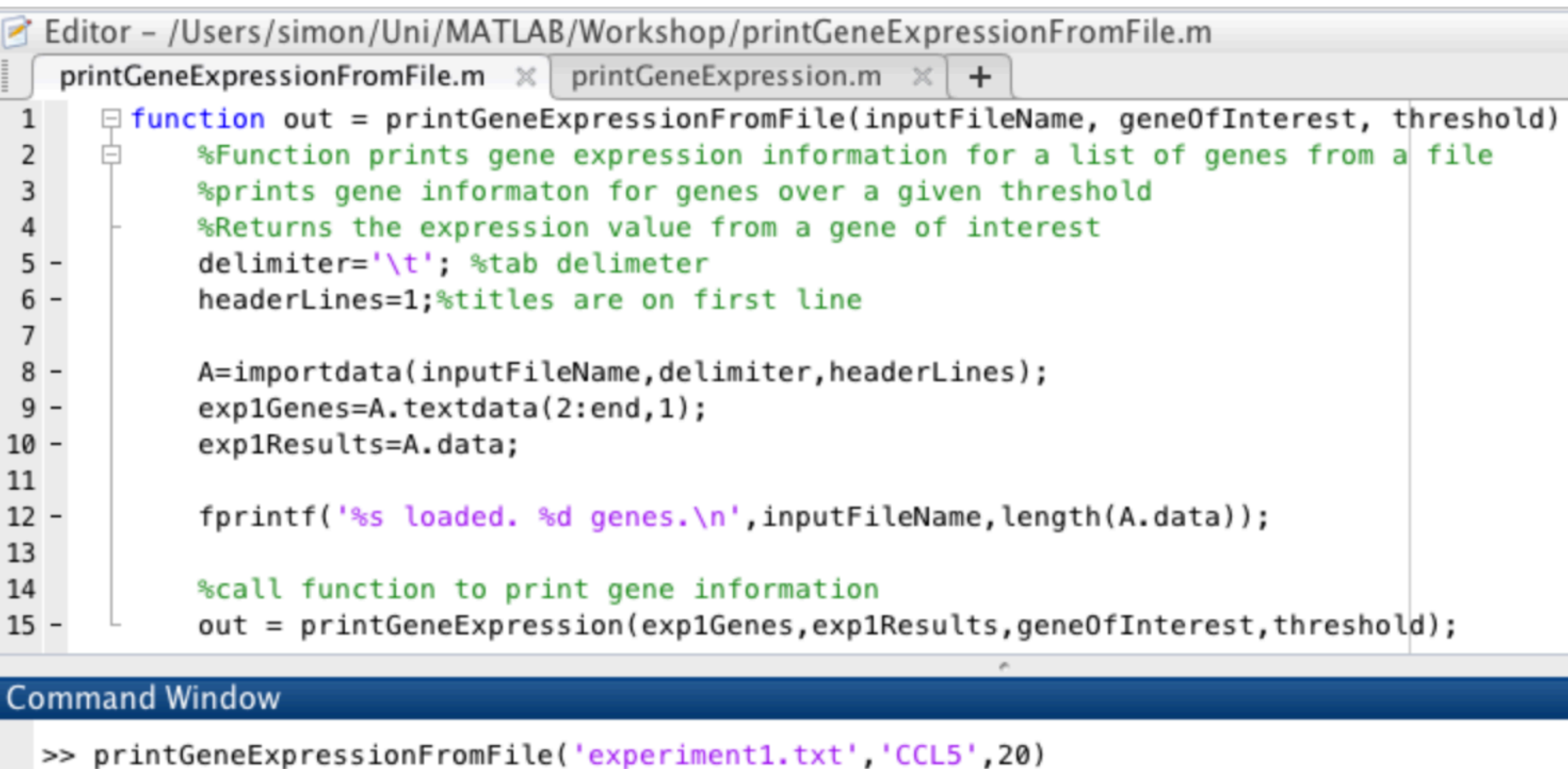
<http://www.signalingystems.ucla.edu/users/Simon/experiment1.txt>

```
Editor - /Users/simon/Uni/MATLAB/Workshop/printGeneExpressionFromFile.m
printGeneExpressionFromFile.m x printGeneExpression.m x +
1 function out = printGeneExpressionFromFile(inputFileName, geneOfInterest, threshold)
2     %Function prints gene expression information for a list of genes from a file
3     %prints gene informaton for genes over a given threshold
4     %Returns the expression value from a gene of interest
5     delimiter='\t'; %tab delimiter
6     headerLines=1;%titles are on first line
7
8     A=importdata(inputFileName,delimiter,headerLines);
9     exp1Genes=A.textdata(2:end,1);
10    exp1Results=A.data;
11
12    fprintf('%s loaded. %d| genes.',inputFileName,length(A.data));
13
14    %call function to print gene information
15    out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold);
```

Working with (more sensible) files

Download example file:

<http://www.signalingystems.ucla.edu/users/Simon/experiment1.txt>



The image shows a MATLAB Editor window with two tabs: 'printGeneExpressionFromFile.m' and 'printGeneExpression.m'. The 'printGeneExpressionFromFile.m' tab is active, displaying a function definition. The function 'printGeneExpressionFromFile' takes three inputs: 'inputFileName', 'geneOfInterest', and 'threshold'. It includes comments explaining its purpose and usage. The function uses 'importdata' to load data from the input file, processes it, and calls 'printGeneExpression' to output the results. The Command Window at the bottom shows the command to run the function with specific arguments.

```
Editor - /Users/simon/Uni/MATLAB/Workshop/printGeneExpressionFromFile.m
printGeneExpressionFromFile.m x printGeneExpression.m x +
1 function out = printGeneExpressionFromFile(inputFileName, geneOfInterest, threshold)
2 %Function prints gene expression information for a list of genes from a file
3 %prints gene information for genes over a given threshold
4 %Returns the expression value from a gene of interest
5 delimiter='\t'; %tab delimiter
6 headerLines=1;%titles are on first line
7
8 A=importdata(inputFileName,delimiter,headerLines);
9 exp1Genes=A.textdata(2:end,1);
10 exp1Results=A.data;
11
12 fprintf('%s loaded. %d genes.\n',inputFileName,length(A.data));
13
14 %call function to print gene information
15 out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold);

Command Window
>> 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

printGeneExpression.m*

printGeneExpressionFromFile.m

forLoops.m

+

```
1 function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold)
2 %Function prints gene expression information for a list of genes over a
3 %given threshold
4 %Returns the expression value from a gene of interest
5 out='Gene Not Found';
6 for i=1:length(exp1Results)
7     if(exp1Results(i)>threshold)
8         fprintf('Gene %s expression %.2f\n',exp1Genes{i},exp1Results(i));
9     end
10    if(strcmp(exp1Genes{i},geneOfInterest))
11        fprintf('Expression of %s: %d\n',geneOfInterest,exp1Results(i));
12        out = exp1Results(i);
13    end
14 end
```

Writing to files

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

```
fprintf(fileIdentifier, 'someText');
```

```
fclose(fid);
```

Changing our functions to write to a file

Editor - /Users/simon/UniNew/MATLAB/Workshop/printGeneExpressionFromFile.m

printGeneExpression.m x printGeneExpressionFromFile.m x +

```
1 function out = printGeneExpressionFromFile(inputFileName, outputFileName, geneOfInterest, threshold)
2     %Function prints gene expression information for a list of genes from a file
3     %prints gene informaton for genes over a given threshold
4     %Returns the expression value from a gene of interest
5 -     delimiter='\t'; %tab delimiter
6 -     headerLines=1;%titles are on first line
7
8 -     A=importdata(inputFileName,delimiter,headerLines);
9 -     exp1Genes=A.textdata(2:end,1);
10 -     exp1Results=A.data;
11
12 -     fprintf('%s loaded. %d genes.\n',inputFileName,length(A.data));
13
14     %call function to print gene information
15 -     out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold, outputFileName);
```



```

1  function out = printGeneExpressionFromFile(inputFileName, outputFileName, geneOfInterest, threshold)
2      %Function prints gene expression information for a list of genes from a file
3      %prints gene informaton for genes over a given threshold
4      %Returns the expression value from a gene of interest
5      delimiter='\t'; %tab delimiter
6      headerLines=1;%titles are on first line
7
8      A=importdata(inputFileName,delimiter,headerLines);
9      exp1Genes=A.textdata(2:end,1);
10     exp1Results=A.data;
11
12     fprintf('%s loaded. %d genes.\n',inputFileName,length(A.data));
13
14     %call function to print gene information
15     out = printGeneExpression(exp1Genes,exp1Results,geneOfInterest,threshold, outputFileName);

```

```

1  function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold, outputFileName)
2      %Function prints gene expression information for a list of genes over a
3      %given threshold
4      %Returns the expression value from a gene of interest
5      fid=fopen(outputFileName,'w');
6      out='Gene Not Found';
7      for i=1:length(exp1Results)
8          if(exp1Results(i)>threshold)
9              fprintf(fid,'%s\t%.2f\n',exp1Genes{i},exp1Results(i));
10             end
11             if(strcmp(exp1Genes{i},geneOfInterest))
12                 fprintf(fid,'Expression of %s: %d\n',geneOfInterest,exp1Results(i));
13                 out = exp1Results(i);
14             end
15         end
16     fclose(fid);

```

Changing our functions to write to a file

```
printGeneExpression.m  printGeneExpressionFromFile.m  +
1  function out = printGeneExpression(exp1Genes, exp1Results, geneOfInterest, threshold, outputFileName)
2      %Function prints gene expression information for a list of genes over a
3      %given threshold
4      %Returns the expression value from a gene of interest
5      fid=fopen(outputFileName,'w');
6      out='Gene Not Found';
7      for i=1:length(exp1Results)
8          if(exp1Results(i)>threshold)
9              fprintf(fid,'%s\t%.2f\n',exp1Genes{i},exp1Results(i));
10         end
11         if(strcmp(exp1Genes{i},geneOfInterest))
12             fprintf(fid,'Expression of %s: %d\n',geneOfInterest,exp1Results(i));
13             out = exp1Results(i);
14         end
15     end
16     fclose(fid);

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

ans =

    3.84
```

Open 'experimentOutput.txt' in excel