

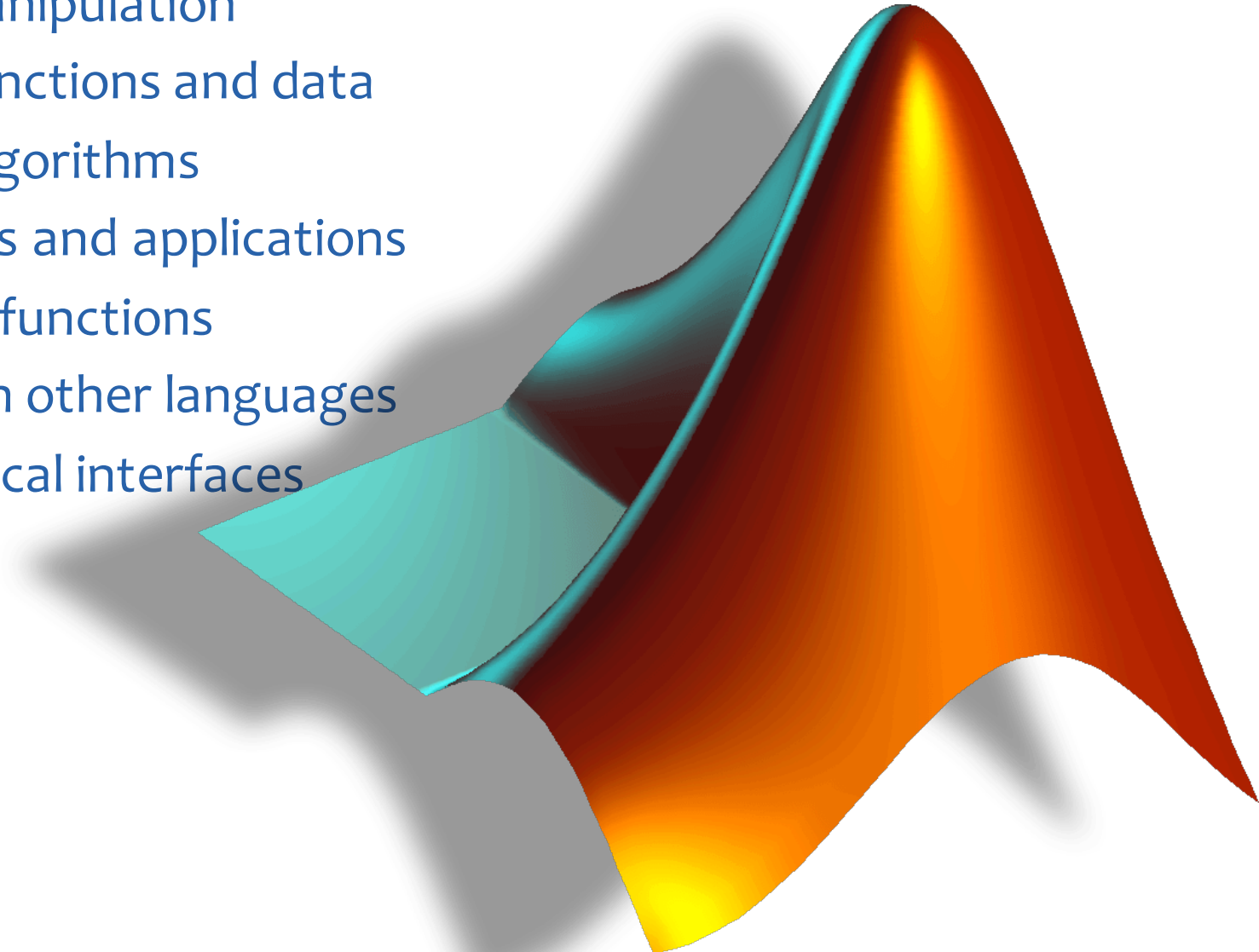
An Introduction to MATLAB

Day 1

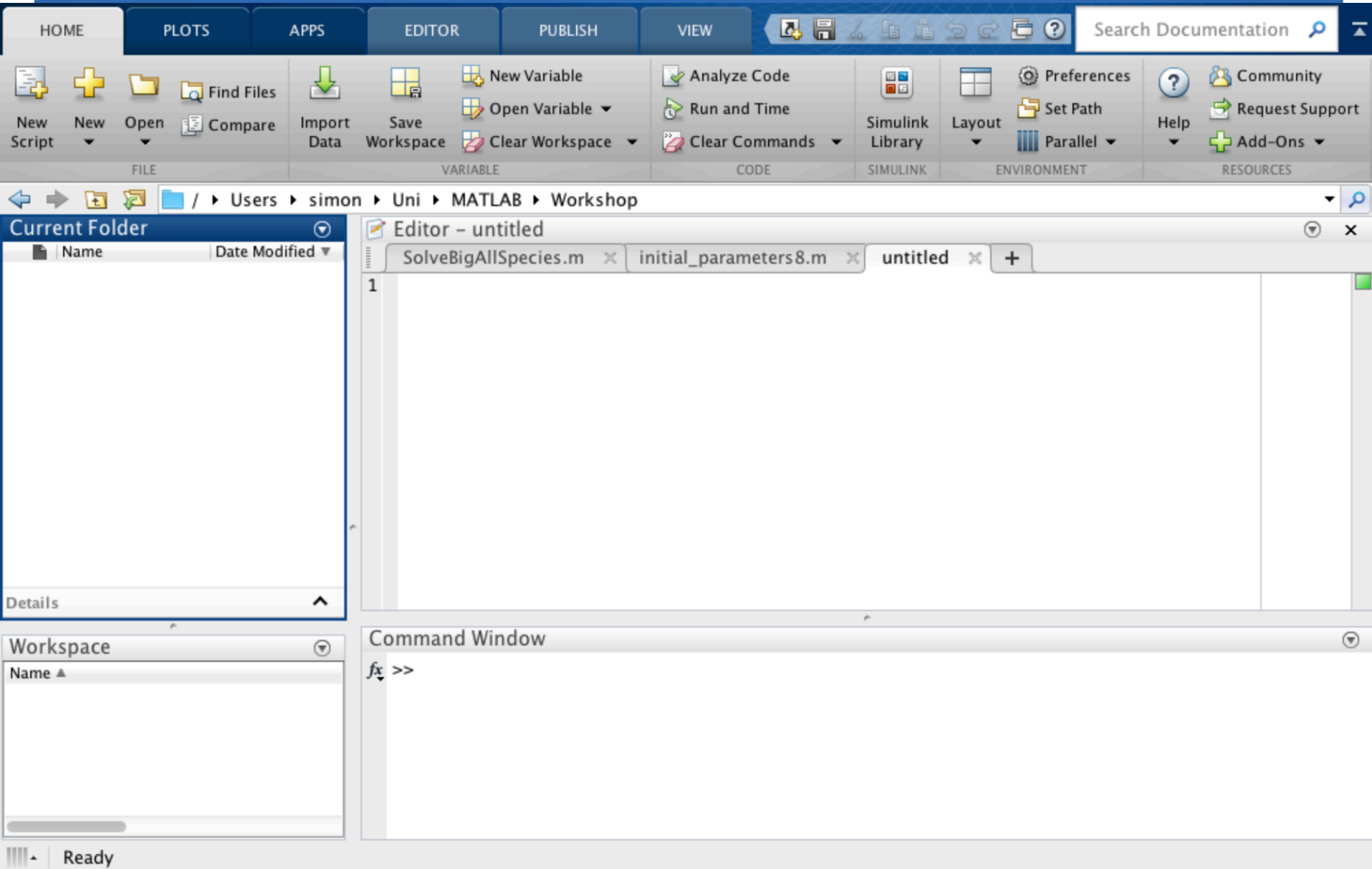
Simon Mitchell

Simon.Mitchell@ucla.edu

- * High level language
- * Programing language and development environment
- * Built-in development tools
- * Numerical manipulation
- * Plotting of functions and data
- * Implement algorithms
- * Create models and applications
- * Many built in functions
- * Interface with other languages
- * Create graphical interfaces



The MATLAB Environment

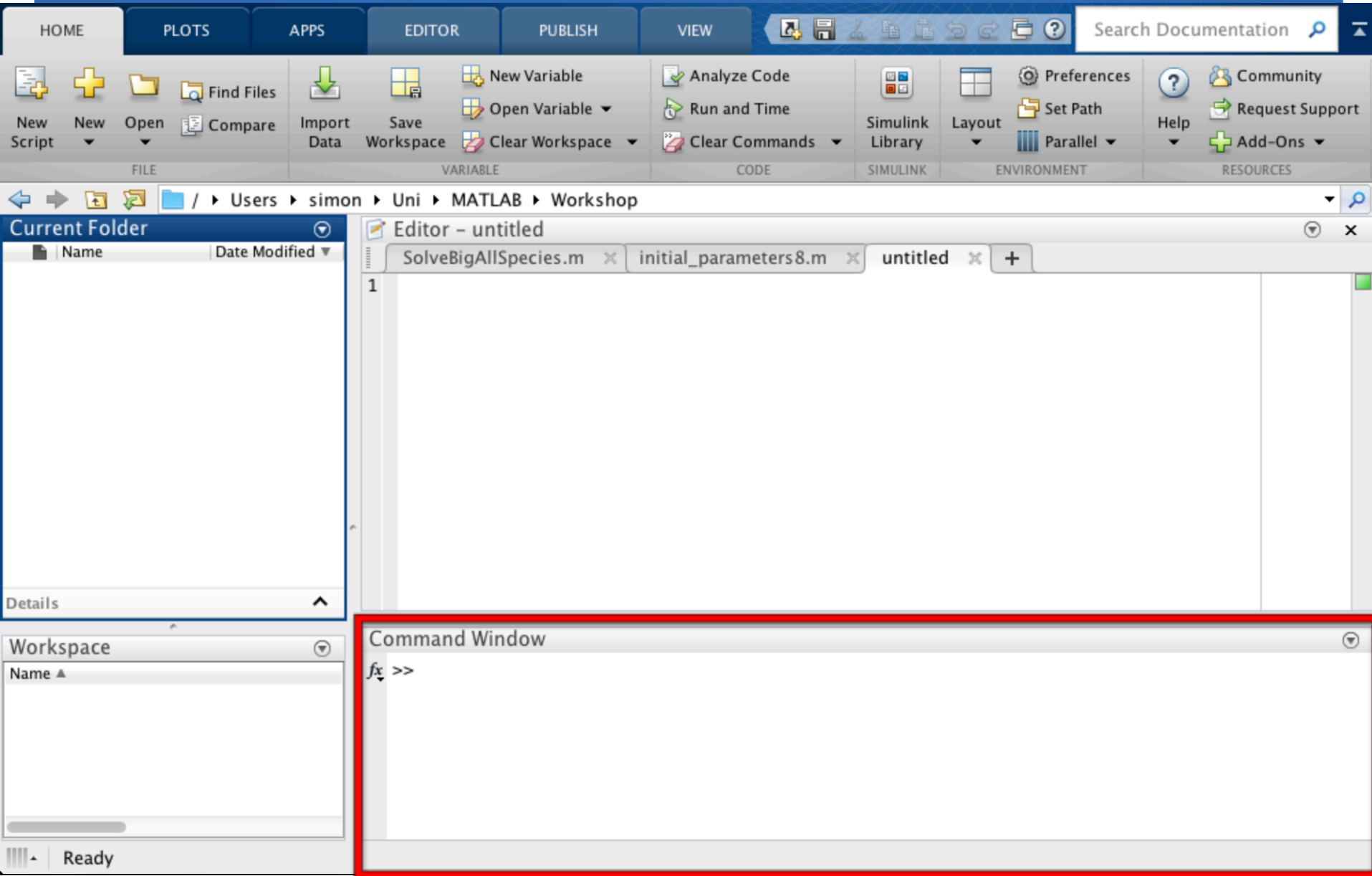


Current Folder

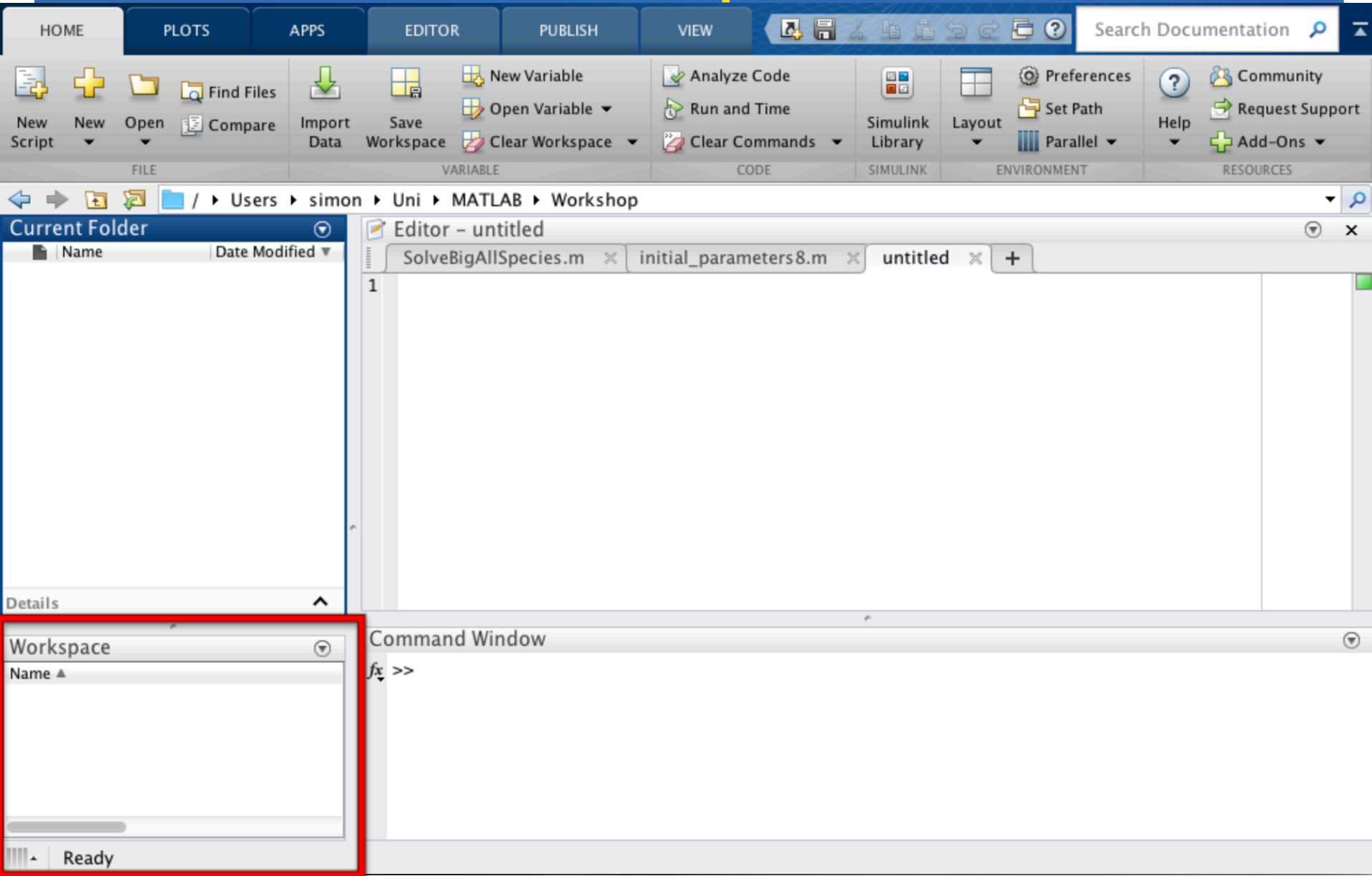
The image displays the MATLAB software interface. At the top, a blue banner contains the title "Current Folder" in large yellow text. Below this is the MATLAB ribbon menu with tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The ribbon contains various tool icons and dropdown menus for file operations, workspace management, code execution, and environment settings. A search bar for "Search Documentation" is located on the right side of the ribbon.

The main workspace area is divided into several panes. On the left, the "Current Folder" browser window is highlighted with a red rectangular border. This window shows the current directory path as "/ > Users > simon > Uni > MATLAB > Workshop". It includes a table with columns "Name" and "Date Modified", which is currently empty. Below the table is a "Details" section. To the right of the Current Folder pane is the "Editor - untitled" pane, which shows a list of open files: "SolveBigAllSpecies.m", "initial_parameters8.m", and "untitled". The editor area is currently blank. Below the editor is the "Command Window" pane, which shows the prompt "fx >>". At the bottom of the interface, a status bar indicates the system is "Ready".

Command Window >>



Workspace



A screenshot of the MATLAB software interface. The top ribbon contains tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below the ribbon are toolbars for FILE, VARIABLE, CODE, SIMULINK, ENVIRONMENT, and RESOURCES. The main workspace area is divided into four panes: Current Folder, Editor, Command Window, and Workspace. The Current Folder pane shows the path / > Users > simon > Uni > MATLAB > Workshop. The Editor pane shows three open files: SolveBigAllSpecies.m, initial_parameters8.m, and untitled. The Command Window pane shows the prompt fx >>. The Workspace pane, which is highlighted with a red border, shows a table with columns Name and Date Modified, and a status bar at the bottom that says Ready.

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder

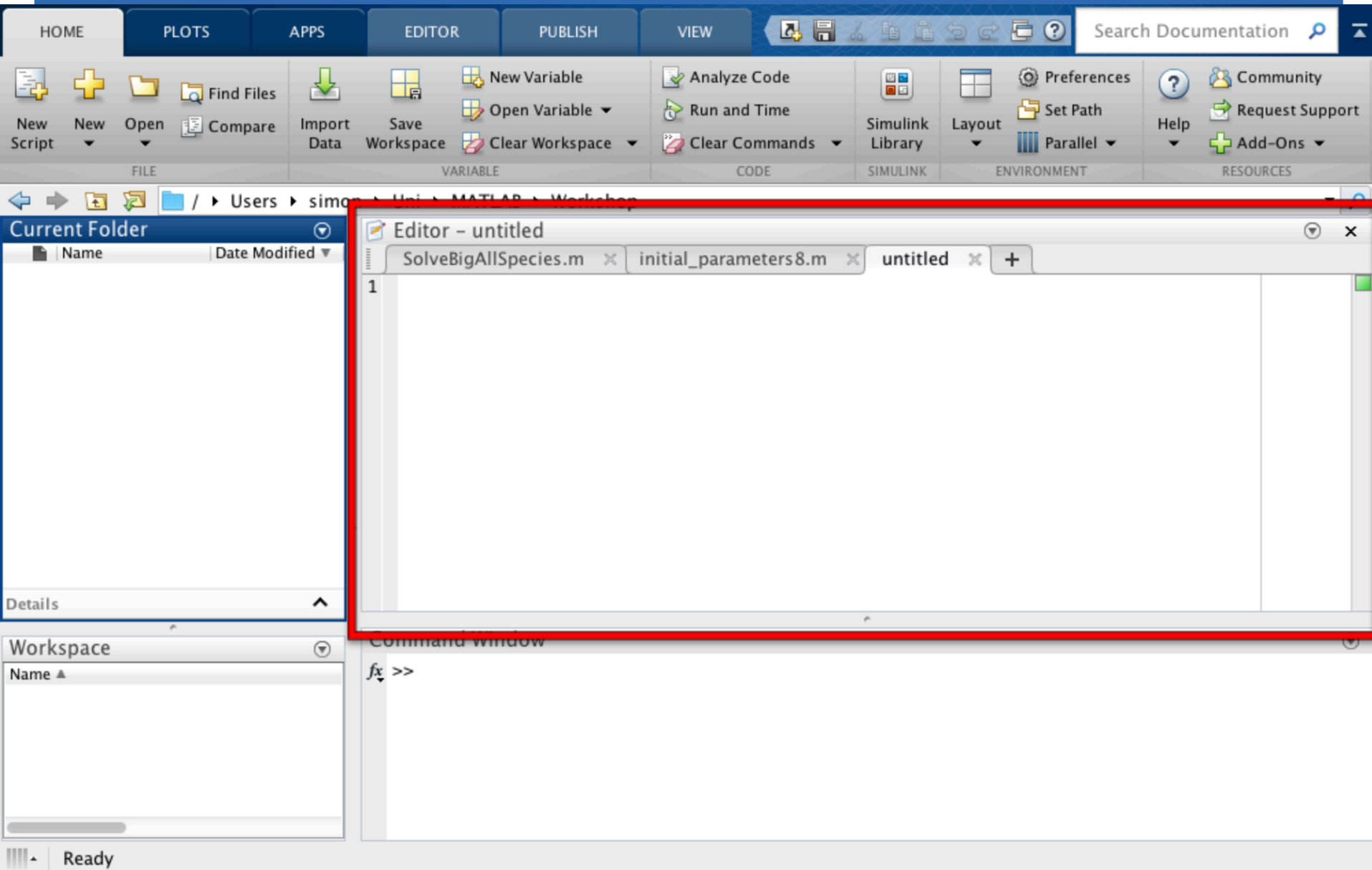
Editor - untitled

Command Window

Workspace

Ready

Editor



Command Window Basics

Command Window

```
>> 2+2
```

```
ans =
```

```
4
```

```
>> 3 ^ 2
```

```
ans =
```

```
9
```

```
>> sin(pi/2)
```

```
ans =
```

```
1
```

```
>> ans
```

```
ans =
```

```
1
```

```
fx >> |
```

Command Window

```
>> 7/0
```

```
ans =
```

```
Inf
```

```
>> (10-3)/(12-(6*2))
```

```
ans =
```

```
Inf
```

```
>> 1e+04*1e+06
```

```
ans =
```

```
1.0000e+10
```

```
fx >>
```


Common Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponential
()	Order operations

Semicolons in MATLAB

Suppress the output from a MATLAB expression



The image shows a screenshot of the MATLAB interface. On the left is the 'Workspace' window, which contains a table with the following data:

Name ▲	Value	Min
ans	9	9

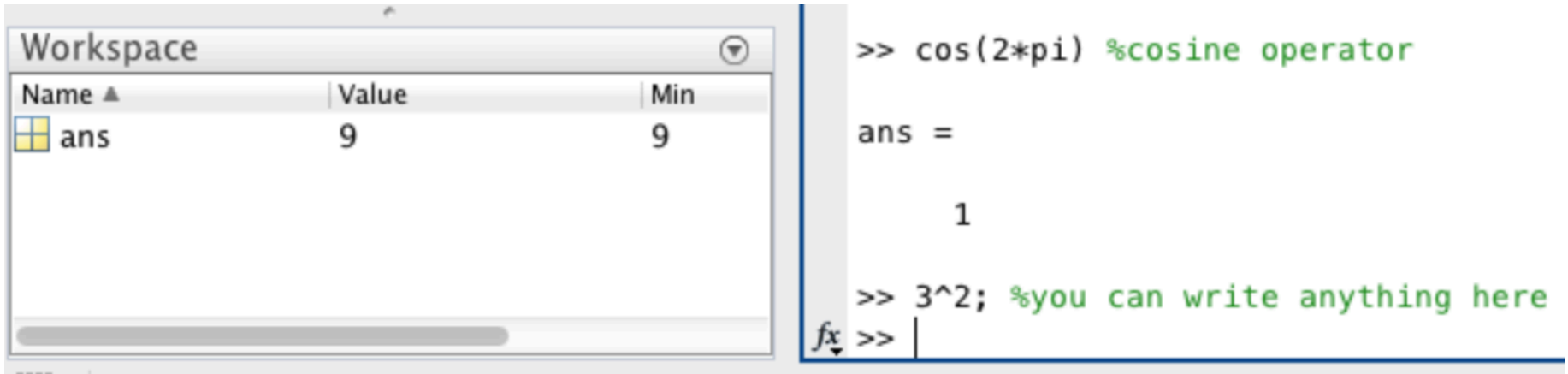
On the right is the Command Window, which displays the following text:

```
>> 3^2  
  
ans =  
  
    9  
  
>> 3^2;  
fx >>
```

The Command Window shows the result of the expression 3^2 as 9. The second line shows the expression 3^2 followed by a semicolon, which suppresses the output.

Comments %

Suppress the output from a MATLAB expression



The image shows a screenshot of the MATLAB environment. On the left is the 'Workspace' window, which contains a table with the following data:

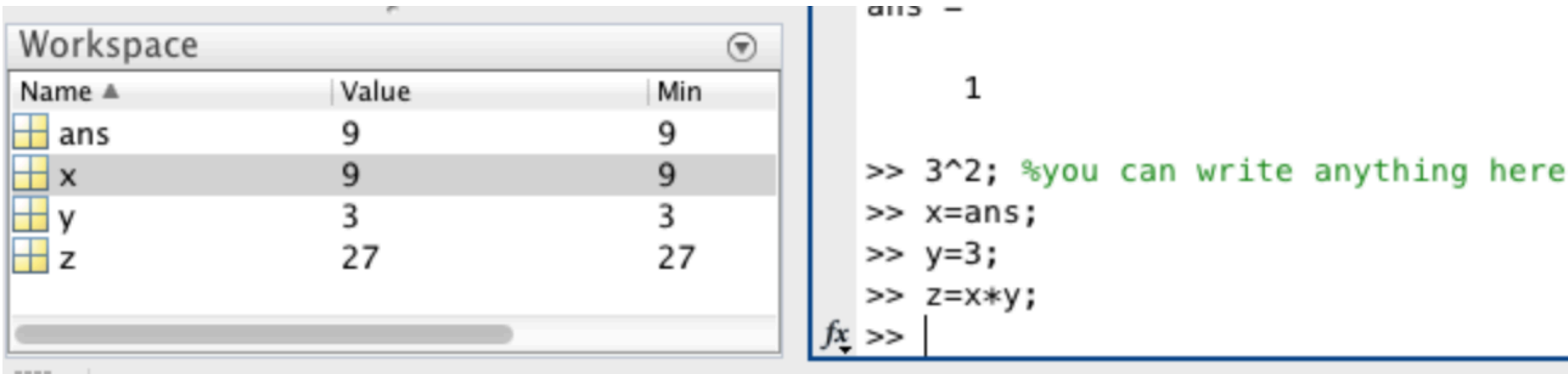
Name ▲	Value	Min
ans	9	9

On the right is the Command Window, showing the following MATLAB code and output:

```
>> cos(2*pi) %cosine operator  
ans =  
1  
  
>> 3^2; %you can write anything here  
fx >> |
```

Variables

Variable name = variable value



The image shows a screenshot of the MATLAB interface. On the left is the 'Workspace' window, which displays a table of variables. On the right is the Command Window, showing a sequence of MATLAB commands and their outputs.

Name ▲	Value	Min
ans	9	9
x	9	9
y	3	3
z	27	27

```
ans =  
1  
  
>> 3^2; %you can write anything here  
>> x=ans;  
>> y=3;  
>> z=x*y;  
fx >> |
```

Be careful of i,j

```
>> i  
  
ans =  
  
0.0000 + 1.0000i
```

Variables 2

MATLAB is CASE SENSITIVE

```
>> x
```

```
x =
```

```
9
```

```
>> X
```

```
Undefined function or variable 'X'.
```

```
Did you mean:
```

```
>> x
```

```
x =
```

```
9
```

```
>>
```

File Explorer showing a folder named **myFile.mat** with a date modified of **23/04/2016 16:57**.

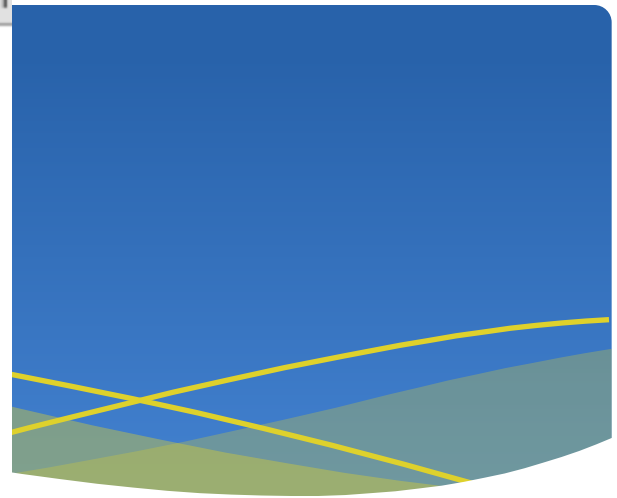
Details

Workspace

Name ▲	Value	Min
ans	0.0000 + 1.0000i	0.0000
x	9	9
y	3	3
z	27	27

SolveBigAllSpecies.m

1



Close MATLAB

Reopen MATLAB

Command Window

```
x =  
  
    9  
  
>> save myFile  
fx >> |
```

Current Folder

Name	Date Modified
myFile.mat	23/04/2016 16:57

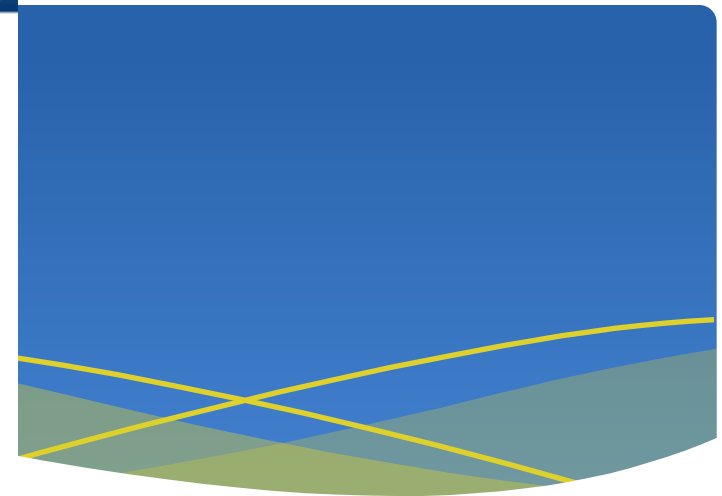
Details

Workspace

Name	Value	Min
------	-------	-----

Command Window

fx >> |



Initially Workspace
is empty

Current Folder

Name	Date Modified
myFile.mat	23/04/2016 16:57

Details

Workspace

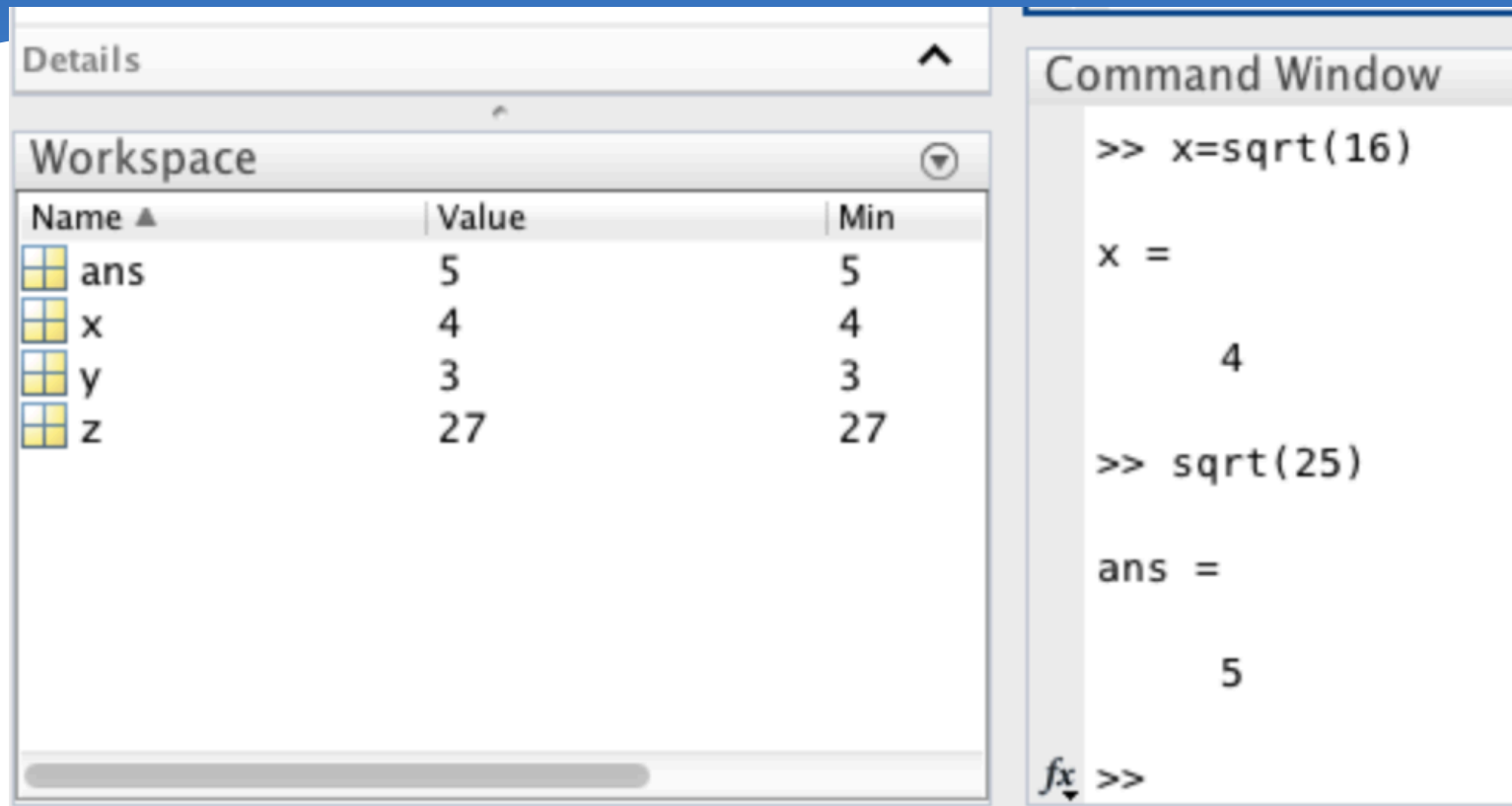
Name	Value	Min
ans	0.0000 + 1.0000i	0.0000
x	9	9
y	3	3
z	27	27

Command Window

```
>> load myFile.mat  
fx >> |
```

Load your
workspace

Variables to result of expression



The image displays the MATLAB interface with two main panels: the Workspace and the Command Window.

Workspace Panel: This panel shows a table of variables currently in the workspace. The table has three columns: Name, Value, and Min. The variables listed are 'ans', 'x', 'y', and 'z'.

Name ▲	Value	Min
ans	5	5
x	4	4
y	3	3
z	27	27

Command Window Panel: This panel shows the command history and the current command. The commands entered are:

```
>> x=sqrt(16)  
  
x =  
  
    4  
  
>> sqrt(25)  
  
ans =  
  
    5  
  
fx >>
```

If an expression is not stored as a variable it will be stored as 'ans'

Multiple assignments and 'who'

Command Window

```
>> a=4;b=2;c=a*b
```

```
c =
```

```
8
```

```
>> who
```

```
Your variables are:
```

```
a      ans  b      c      x      y      z
```

```
fx >>
```

whos

Command Window

```
>> whos
```

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	
ans	1x1	8	double	
b	1x1	8	double	
c	1x1	8	double	
x	1x1	8	double	
y	1x1	8	double	
z	1x1	8	double	

```
>> clear a
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	
b	1x1	8	double	
c	1x1	8	double	
x	1x1	8	double	
y	1x1	8	double	
z	1x1	8	double	

```
>> clear
```

```
>> whos
```

Command Window

```
>> x=pi
```

```
x =
```

```
3.1416
```

```
>> format long
```

```
>> x=pi
```

```
x =
```

```
3.141592653589793
```

```
>> format short
```

```
>> x
```

```
x =
```

```
3.1416
```

```
>> format bank
```

```
>> pi
```

```
ans =
```

```
3.14
```

format

```
>> format rat
```

```
>> pi
```

```
ans =
```

```
355/113
```

```
>>
```

‘format short’
to get back to normal

Numbers are actually 1x1 Matrices

```
>> format short  
>> x
```

```
x =
```

```
3.1416
```

```
>> size(x)
```

```
ans =
```

```
1    1
```

```
>> rowVector = [1,2,3,4,5]
```

```
rowVector =
```

```
1    2    3    4    5
```

```
>> rowVector = [1:5]
```

```
rowVector =
```

```
1    2    3    4    5
```

```
>> size(rowVector)
```

```
ans =
```

```
1    5
```

Operations apply to matrices

```
>> rowVector  
  
rowVector =  
  
      1      2      3      4      5  
  
>> secondVector = [4,3,2,1,0]  
  
secondVector =  
  
      4      3      2      1      0  
  
>> rowVector+secondVector  
  
ans =  
  
      5      5      5      5      5
```

Operations apply to matrices

```
>> rowVector+secondVector
```

```
ans =
```

```
5    5    5    5    5
```

```
>> columnVector = [1;2;3;4;5]
```

```
columnVector =
```

```
1  
2  
3  
4  
5
```

```
>> columnVector + rowVector
```

```
Error using +  
Matrix dimensions must agree.
```

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

```
myMatrix =
```

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

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

```
mySecondMatrix =
```

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

```
>> myMatrix + mySecondMatrix
```

```
ans =
```

```
    7    7    7
    7    7    7
```

```
>> myMatrix * mySecondMatrix
```

```
Error using *
Inner matrix dimensions must agree.
```

```
>> myMatrix .* mySecondMatrix
```

```
ans =
```

```
    6   10   12
   12   10    6
```

Put . before an operator to make it element-wise

```
>> ans
```

```
ans =
```

```
    6   10   12
   12   10    6
```

```
>> disp(ans)
```

```
    6   10   12
   12   10    6
```

disp(variable) displays the contents of a variable

fprintf – print something to command window

```
>> numberOfDays = 3;
>> nameOfInstructor = 'Simon';
>> fprintf('We are learning MATLAB with %s over %d days',nameOfInstructor, numberOfDays)
We are learning MATLAB with Simon over 3 days>>
>> fprintf('We are learning MATLAB with %s over %d days\n',nameOfInstructor, numberOfDays)
We are learning MATLAB with Simon over 3 days
>>
```

%s	Format as a string.
%d	Format as an integer.
%f	Format as a floating point value.
%e	Format as a floating point value in scientific notation.
%g	Format in the most compact form: %f or %e.
\n	Insert a new line in the output string.
\t	Insert a tab in the output string.

Concatenating matrices

```
>> myMatrix = [0,1,2,3];  
>> myOtherMatrix=[4,5,6,7];  
>> cat(1,myMatrix,myOtherMatrix)
```

ans =

0	1	2	3
4	5	6	7

```
>> cat(2,myMatrix,myOtherMatrix)
```

ans =

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

```
>> [myMatrix,myOtherMatrix]
```

ans =

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

```
>> [myMatrix;myOtherMatrix]
```

ans =

0	1	2	3
4	5	6	7

ans =

0	1	2	3
4	5	6	7

```
>> find(ans)
```

ans =

2
3
4
5
6
7
8

Concatenating matrices

```
>> [myMatrix;myOtherMatrix]
```

```
ans =
```

0	1	2	3
4	5	6	7

```
>> length(ans)
```

```
ans =
```

4

```
>> size(ans)
```

```
ans =
```

1	1
---	---

Why is this 1,1?



```
>> size([myMatrix;myOtherMatrix])
```

```
ans =
```

2	4
---	---

Many Functions work on columns

```
myBigMatrix =
```

0	1	2	3
4	5	6	7

```
>> max(myBigMatrix)
```

```
ans =
```

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

```
>> min(myBigMatrix)
```

```
ans =
```

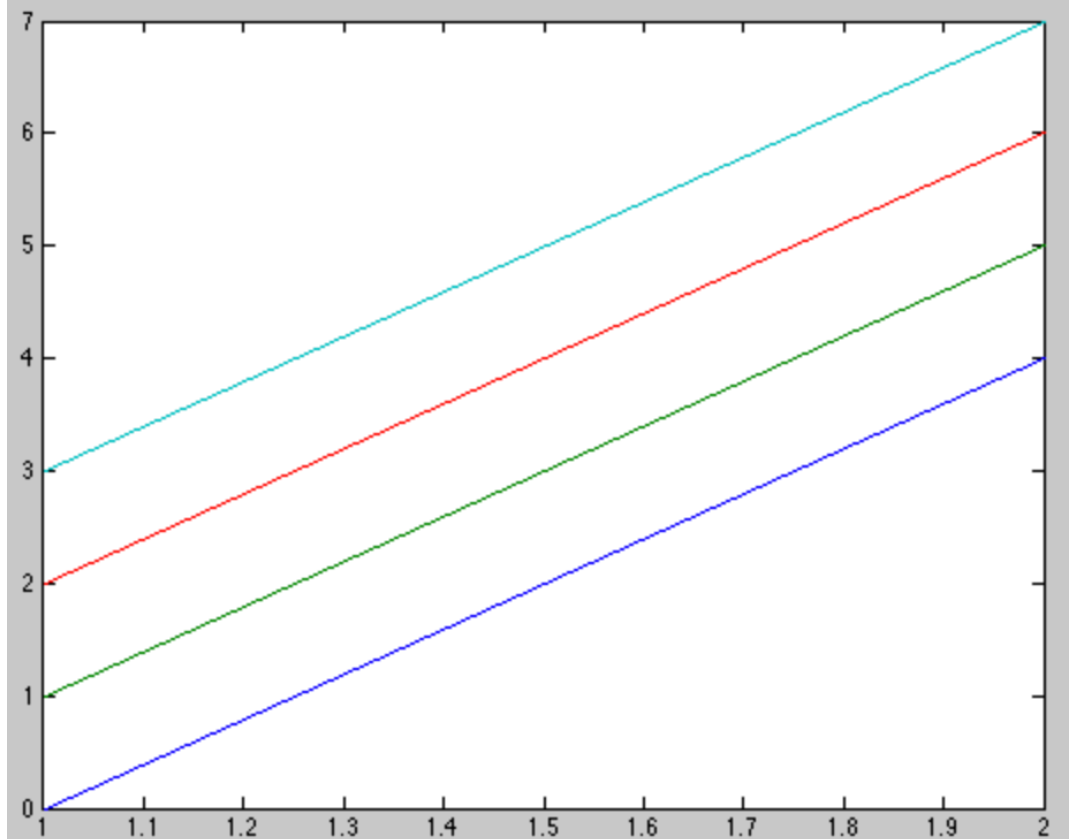
0	1	2	3
---	---	---	---

```
>> sum(myBigMatrix)
```

```
ans =
```

4	6	8	10
---	---	---	----

```
>> plot(myBigMatrix)
```



Working with m-files

- * 3 Ways of running code

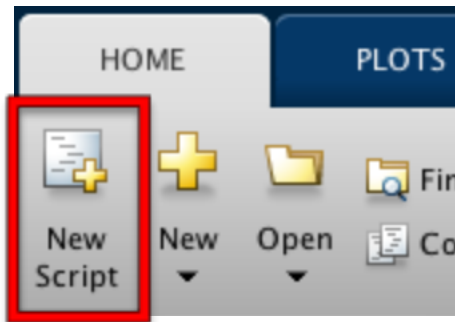
- * Command Window ✓

- * Scripts

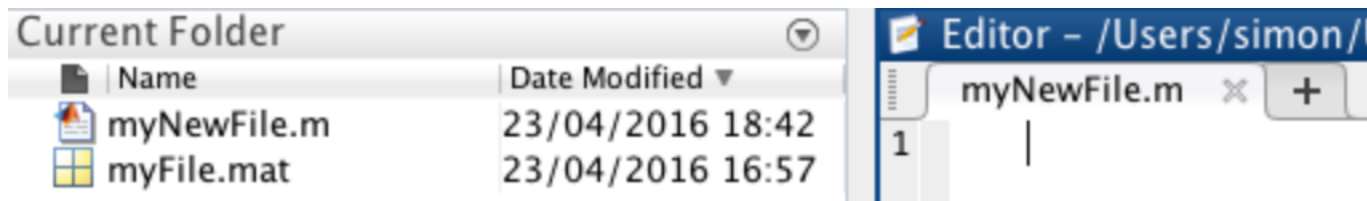
- * Functions

m-files

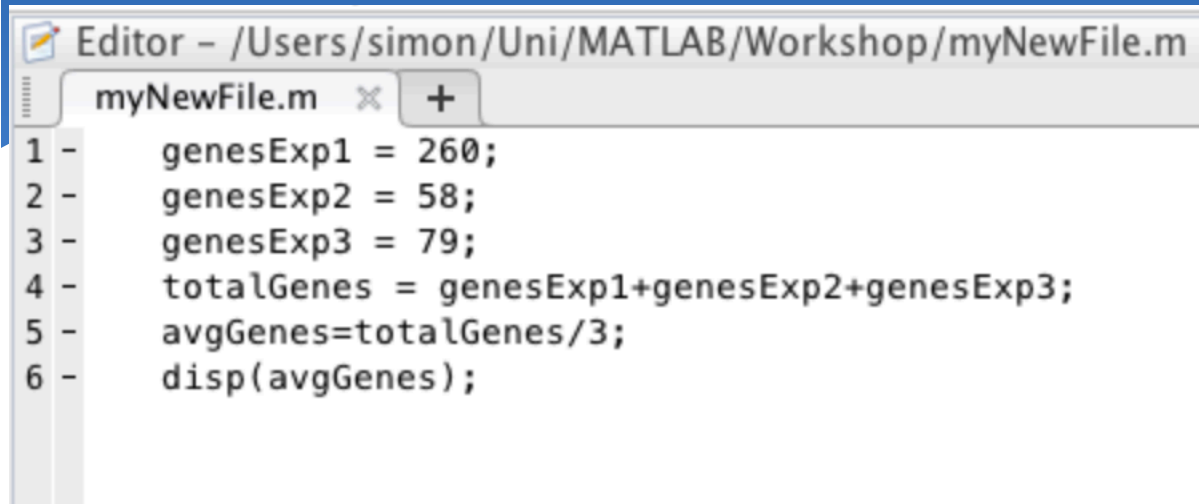
New Script



```
>> edit  
>> edit myNewFile.m
```



Our First Script



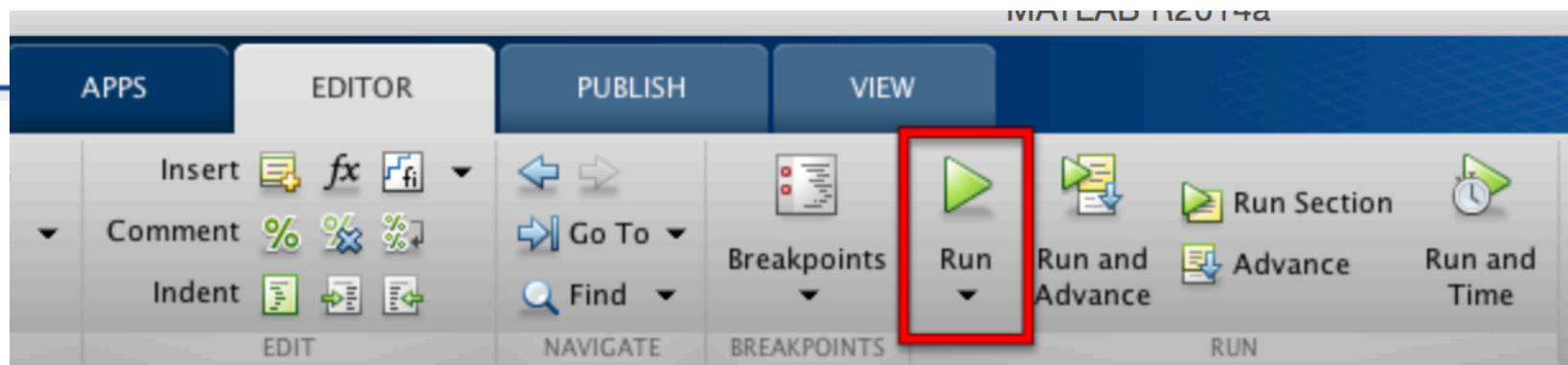
Editor - /Users/simon/Uni/MATLAB/Workshop/myNewFile.m

```
1 - genesExp1 = 260;  
2 - genesExp2 = 58;  
3 - genesExp3 = 79;  
4 - totalGenes = genesExp1+genesExp2+genesExp3;  
5 - avgGenes=totalGenes/3;  
6 - disp(avgGenes);
```

Command Window

```
>> edit  
>> edit myNewFile.m  
>> myNewFile  
132.3333
```

```
fx >>
```



Scripts can create variables in the workspace

Current Folder

Name	Date Modified
myNewFile.m	23/04/2016 18:48
myNewFile.m~	23/04/2016 18:46
myFile.mat	23/04/2016 16:57

Workspace

Name	Value	Min
avgGenes	132.3333	132.3333
genesExp1	260	260
genesExp2	58	58
genesExp3	79	79
totalGenes	397	397

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

```
myNewFile.m  x  +
1 - genesExp1 = 260;
2 - genesExp2 = 58;
3 - genesExp3 = 79;
4 - totalGenes = genesExp1+genesExp2+genesExp3;
5 - avgGenes=totalGenes/3;
6 - disp(avgGenes);
```

Command Window

```
>> clear
>> myNewFile
    132.3333
```


Data Types

- * Type declarations are not necessary in MATLAB
- * MATLAB automatically decides data type

Command Window

```
>> myVar = 7.3;  
>> class(myVar)
```

```
ans =
```

```
double
```

```
>> myVar = 'test'
```

```
myVar =
```

```
test
```

```
>> class(myVar)
```

```
ans =
```

```
char
```

```
fx >> |
```

Data Types

single - single precision numerical data

double - double precision numerical data

logical - logical values of 1 or 0, represent true and false respectively

char - character data (strings are stored as vector of characters)

cell array - array of indexed cells, each capable of storing an array of a different dimension and data type

structure - named fields capable of storing an array of a different dimension and data type

function handle - pointer to a function

user classes - objects constructed from a user-defined class

Int8 uint8 int16 uint16 int32 uint32 int64 uint64 – don't worry about these

Data Types

```
>> edit dataTypes.m
```

```
Editor - /Users/simon/Uni/MATLAB/Work
myNewFile.m x dataTypes.m x +
1 - geneName = 'nfkbia';
2 - disp(geneName);
3 - geneExp = 367.54323;
4
5 - doubleVal = double(geneExp);
6 - intVal = uint32(geneExp);
7
8 - disp(doubleVal);
9 - disp(intVal);
10
11 - disp(isinteger(doubleVal));
12 - disp(isinteger(intVal));
13

Command Window
>> dataTypes
nfkbia
367.5432

368

0

1

fx >>
```

```
1 - x = [1 2 3]
2 - isinteger(x)
3 - isfloat(x)
4 - isvector(x)
5 - isscalar(x)
```

Command Window

```
x =
     1     2     3
```

```
ans =
     0
```

```
ans =
     1
```

```
ans =
     1
```

```
ans =
     0
```

Testing Data Types

Relational Operators

<	Less than
>=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

```
>> x=6; y=9;  
>> x>y
```

```
ans =
```

```
0
```

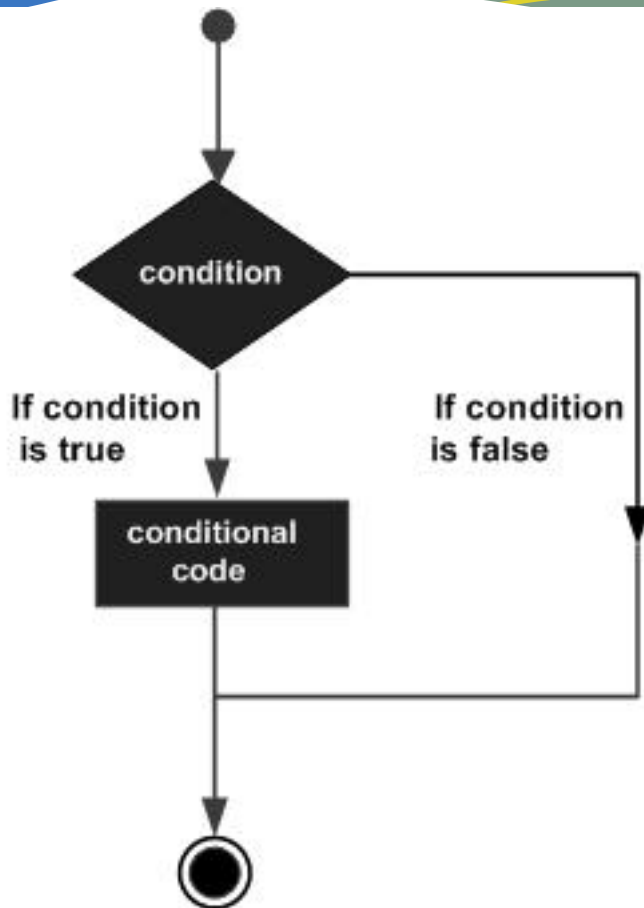
```
>> x<=y
```

```
ans =
```

```
1
```

Useful for if statements!

If Statement



The image shows a screenshot of a MATLAB environment. The top window is the "Editor" titled "Editor - /Users/simon/Un", showing a file named "ifStatement.m". The code in the editor is as follows:

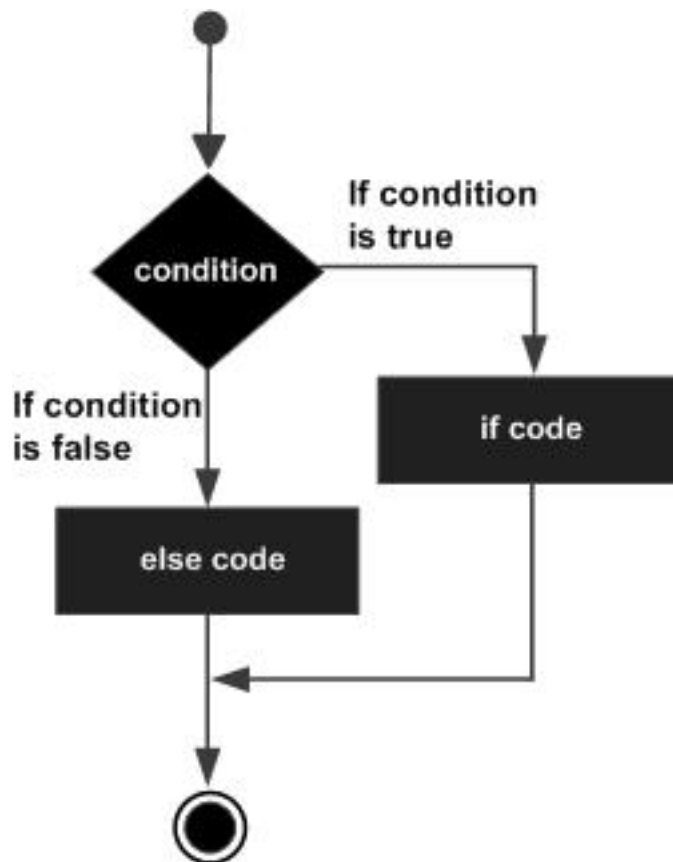
```
1 - exp1 = 400;  
2 - exp2 = 500;  
3 - if (exp1 >= exp2)  
4 -     max == exp1  
5 - end
```

Below the editor is the "Command Window", which contains the command:

```
>> ifStatement
```

} Never run

If else Statement



The image shows a MATLAB Editor window with a file named 'myNewFile.m' and a Command Window below it.

Editor - /Users/simon/Uni/...

```
1 - exp1 = 400;
2 - exp2 = 500;
3 - if (exp1 >= exp2)
4 -     max = exp1
5 - else
6 -     max = exp2
7 - end
```

Command Window

```
>> edit ifStatement.m
>> ifStatement

max =

    500

fx >>
```

If elseif Statement

```
ifStatement.m  x  +
1 -   exp1 = 400;
2 -   exp2 = 400;
3 -   if (exp1 > exp2)
4 -       max == exp1
5 -   elseif (exp1==exp2)
6 -       max == 'Both are equal'
7 -   else
8 -       max == exp2
9 -   end
```

Command Window

```
max =  
  
Both are equal
```


Nested if Statement

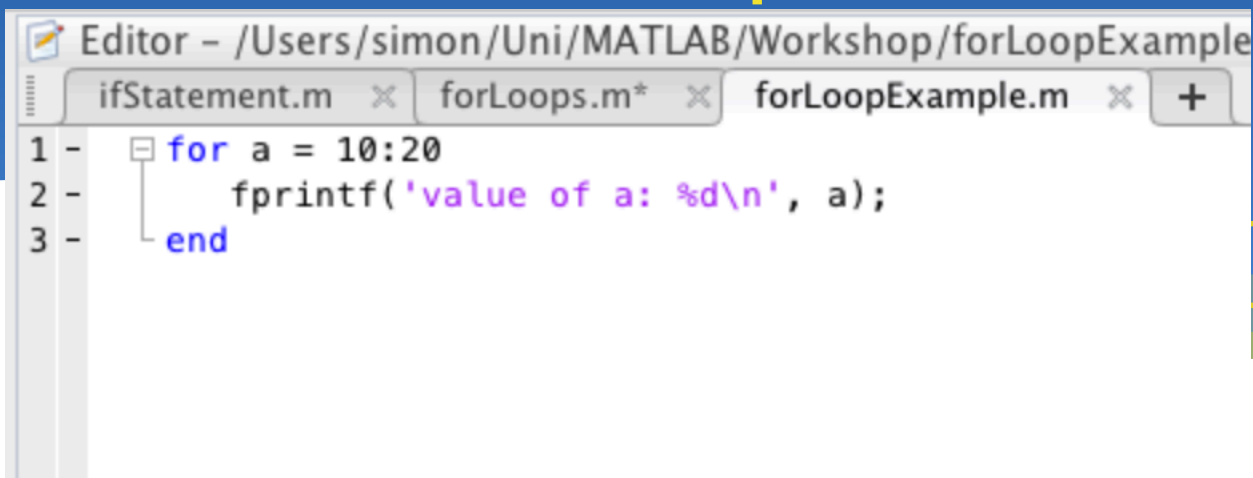
```
Editor - /Users/simon/Uni/MATLAB/Workshop/ifStatem
ifStatement.m
1 - exp1 = 440;
2 - exp2 = 400;
3 - if (exp1 > exp2)
4 -     if ((exp1-exp2)>50)
5 -         fprintf('exp1 is much bigger')
6 -     else
7 -         fprintf('exp1 is slightly bigger')
8 -     end
9 -     max = 'exp1'
10 - elseif (exp1==exp2)
11 -     max = 'Both are equal'
12 - else
13 -     max = exp2
14 - end
```

Command Window

```
>> ifStatement
exp1 is slightly bigger
max =

exp1
```

For loops



The image shows a MATLAB Editor window with the title bar "Editor - /Users/simon/Uni/MATLAB/Workshop/forLoopExample". There are three tabs open: "ifStatement.m", "forLoops.m*", and "forLoopExample.m". The "forLoops.m" tab is active, showing a for loop that iterates from 10 to 20, printing the value of 'a' for each iteration. The code is as follows:

```
1 - for a = 10:20
2 -     fprintf('value of a: %d\n', a);
3 - end
```

Command Window

```
>> edit forLoops.m
>> edit forLoopExample.m
>> forLoopExample
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
value of a: 20
```

```
fx >>
```

Combining What We've Learned

Write a **script** that takes a list of gene names and gene expression values and outputs only those gene names over a threshold.

PSUEDO CODE:

For each gene in a list:

 if its expression value is over the threshold

 print the gene name and the expression value

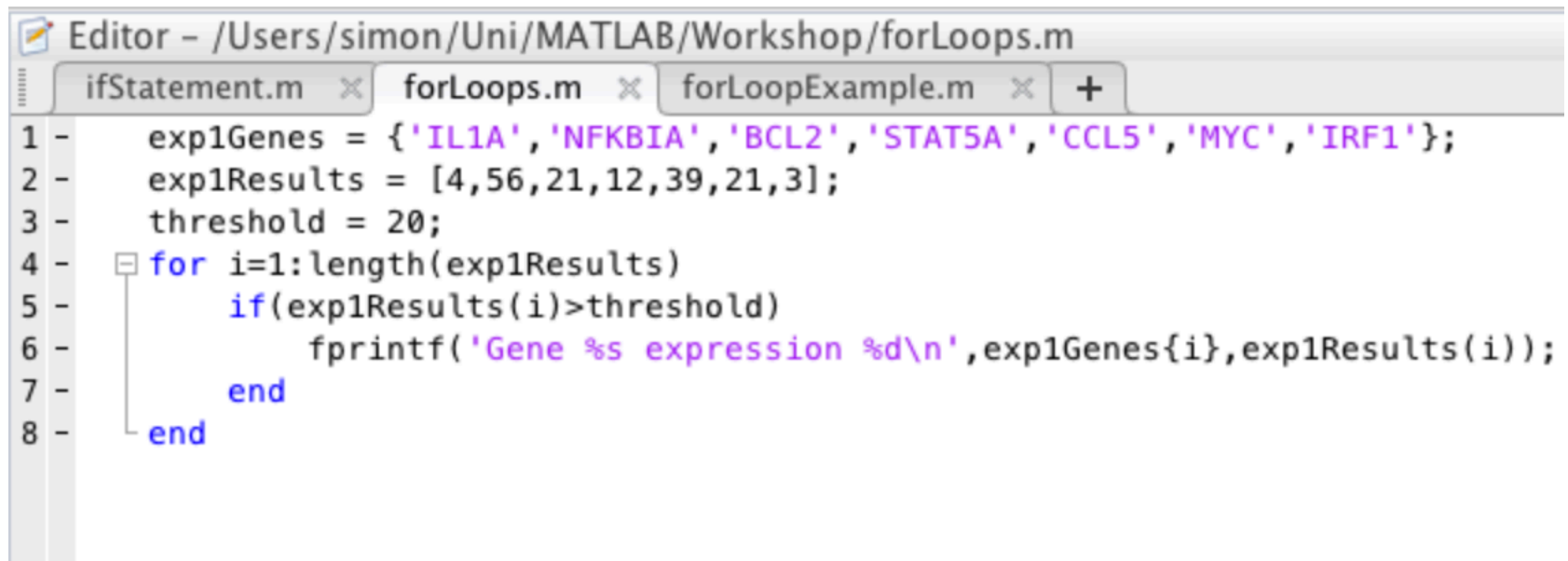
Combining What We've Learned

PSUEDO CODE:

For each gene in a list:

if its expression value is over the threshold

print the gene name and the expression value



The image shows a MATLAB Editor window with the title bar "Editor - /Users/simon/Uni/MATLAB/Workshop/forLoops.m". The window contains three tabs: "ifStatement.m", "forLoops.m", and "forLoopExample.m". The "forLoops.m" tab is active, displaying the following MATLAB code:

```
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
```

Combining What We've Learned

```
Editor - /Users/simon/Uni/MATLAB/Workshop/forLoops.m
ifStatement.m x forLoops.m x forLoopExample.m x +
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
```

Command Window

```
>> edit forLoops
>> forLoops
Gene NFKBIA expression 56
Gene BCL2 expression 21
Gene CCL5 expression 39
Gene MYC expression 21
fx >> |
```