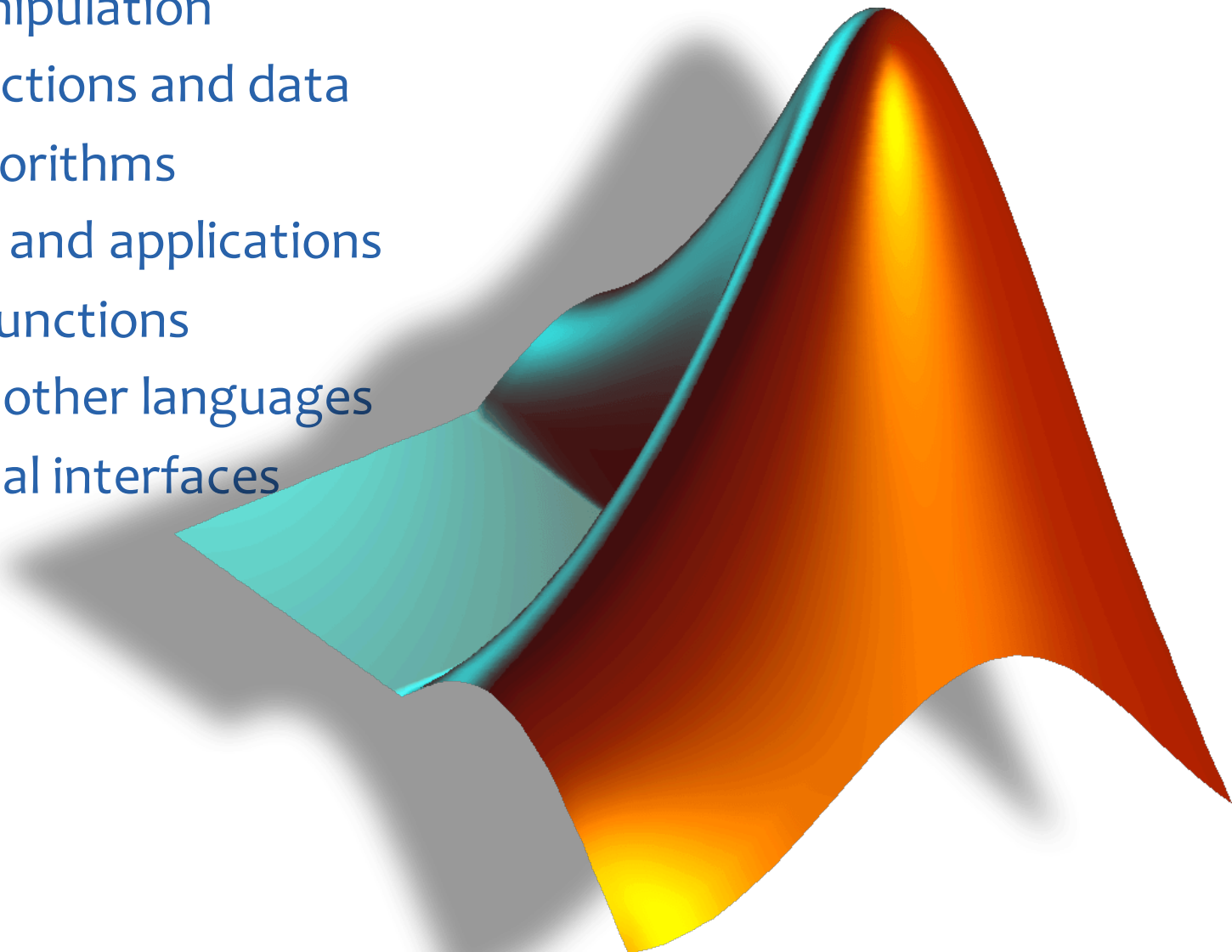
An Introduction to MATLAB

Day 1

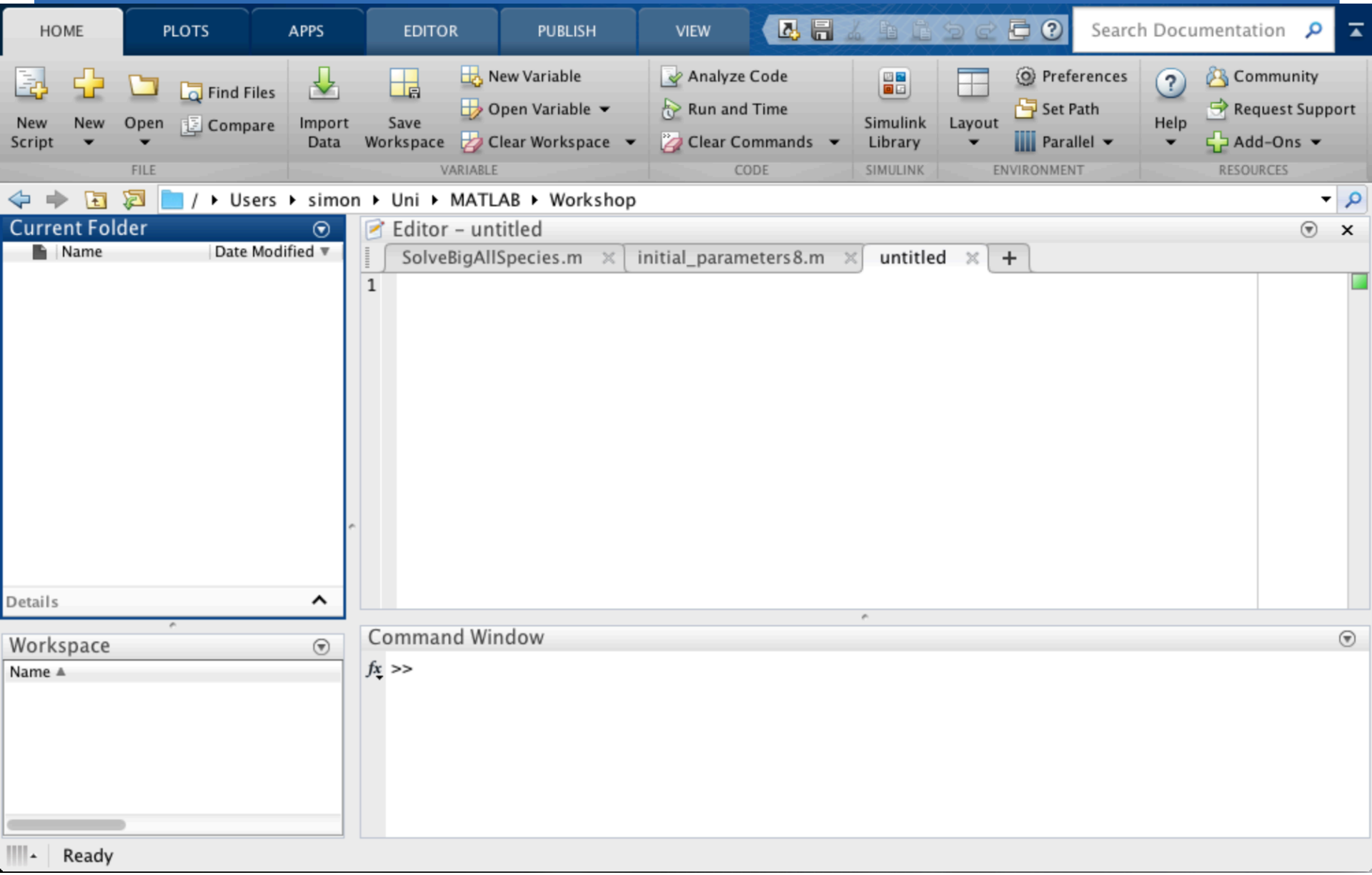
Simon Mitchell

Simon.Mitchell@ucla.edu

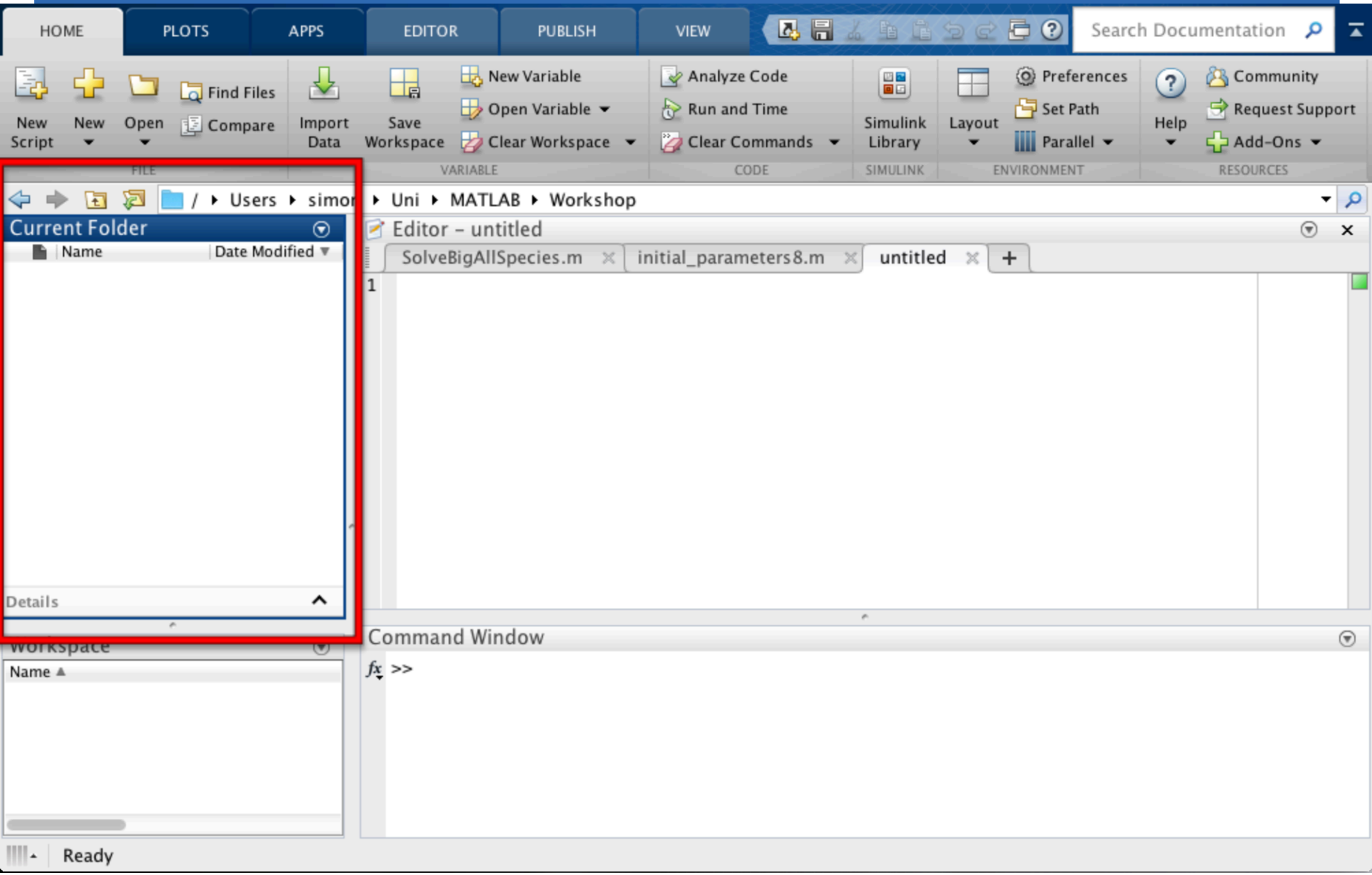
* High level language
* Programing language and development environment
* Built-in development tools
* Numerical manipulation
* Plotting of functions and data
* Implement algorithms
* Create models and applications
* Many built in functions
* Interface with other languages
* Create graphical interfaces



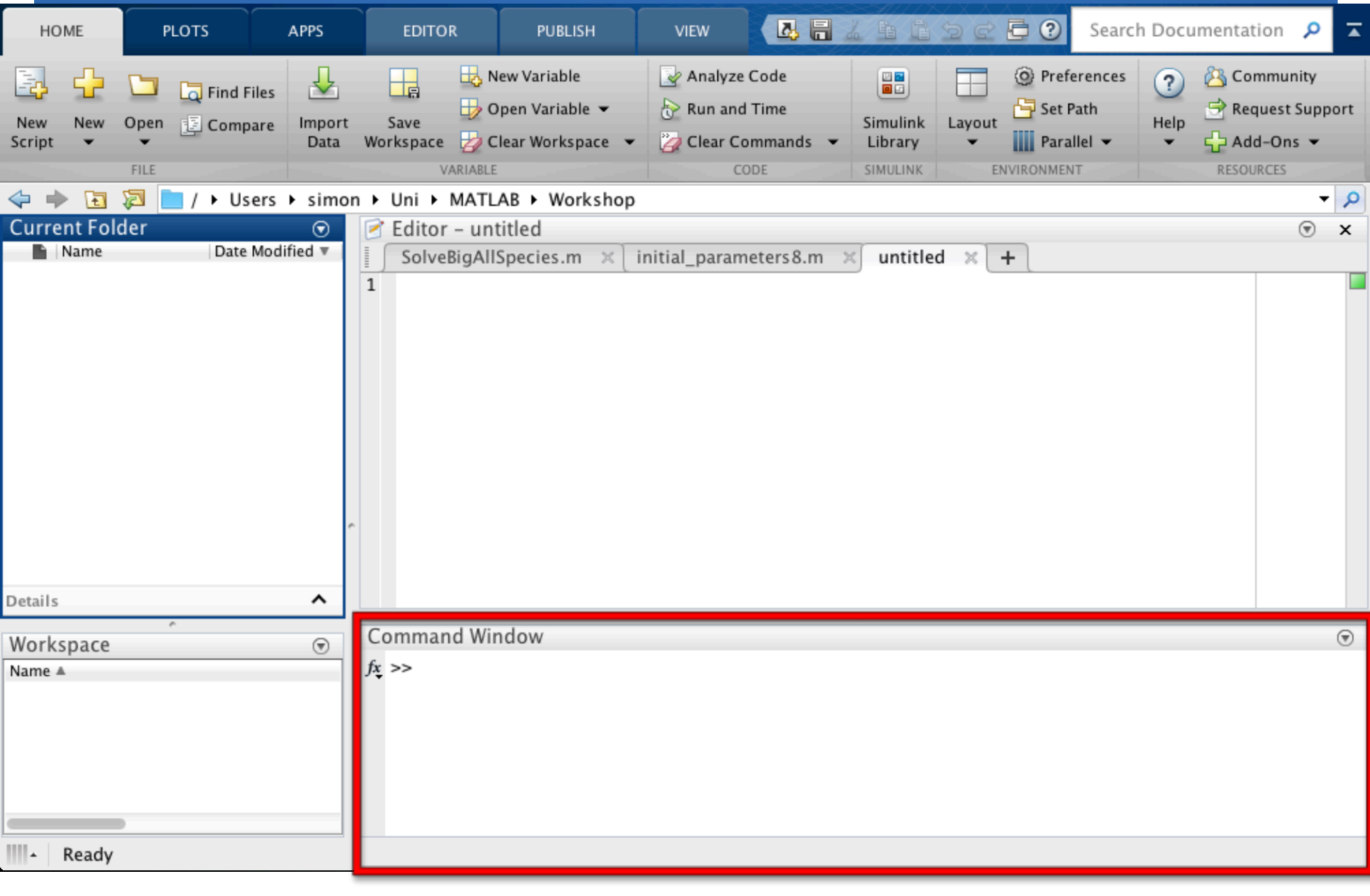
The MATLAB Environment



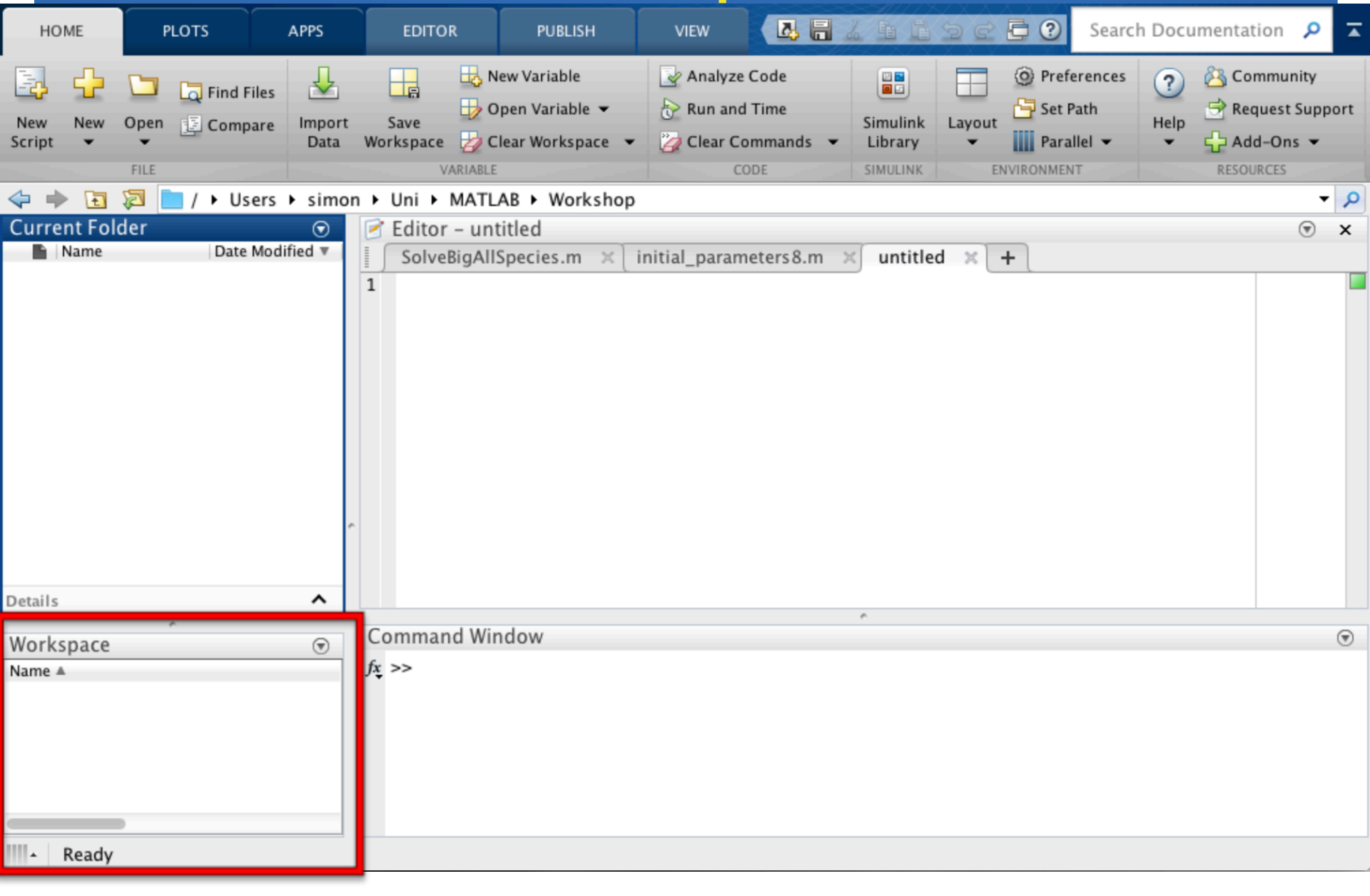
Current Folder



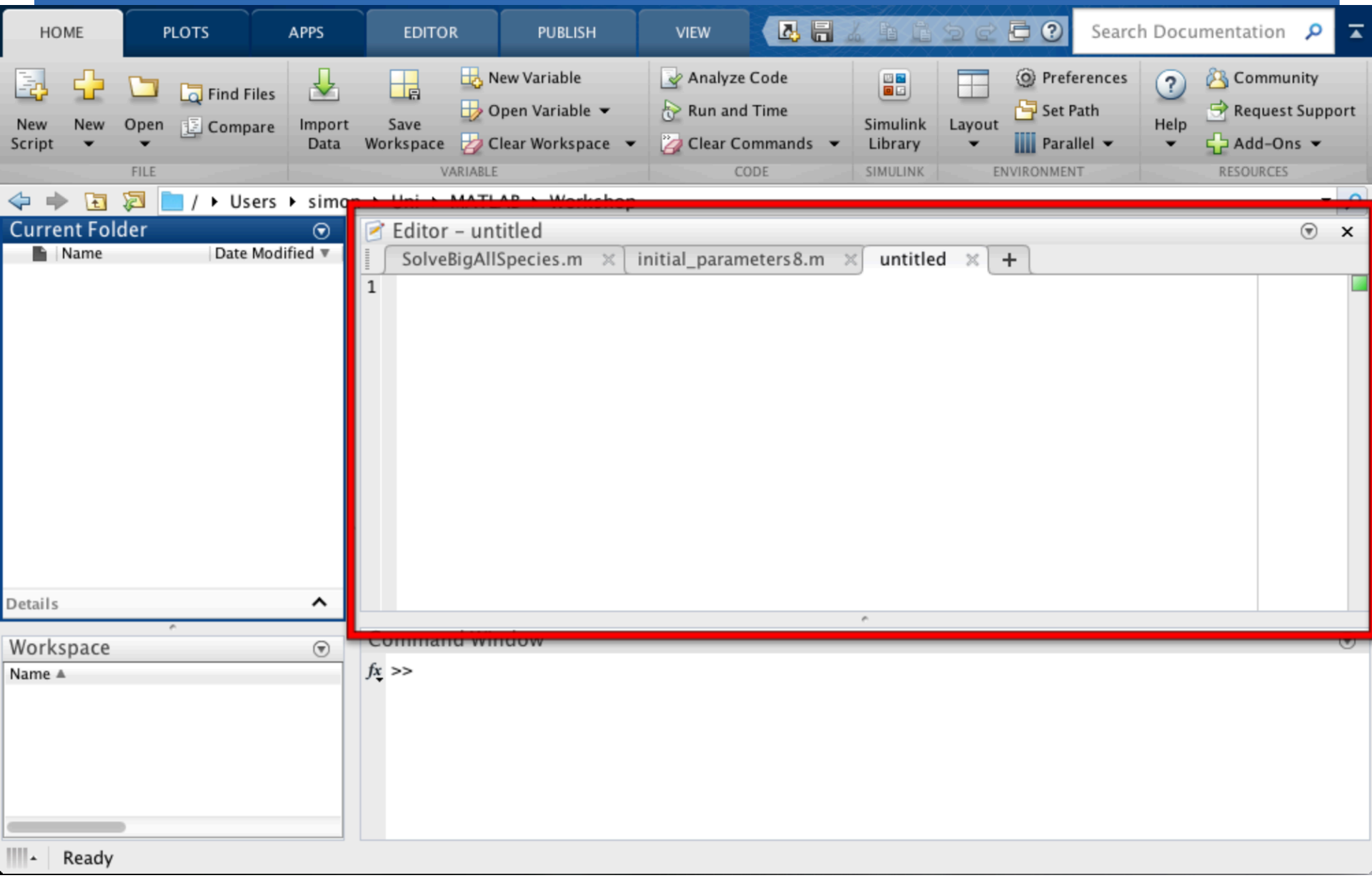
Command Window >>



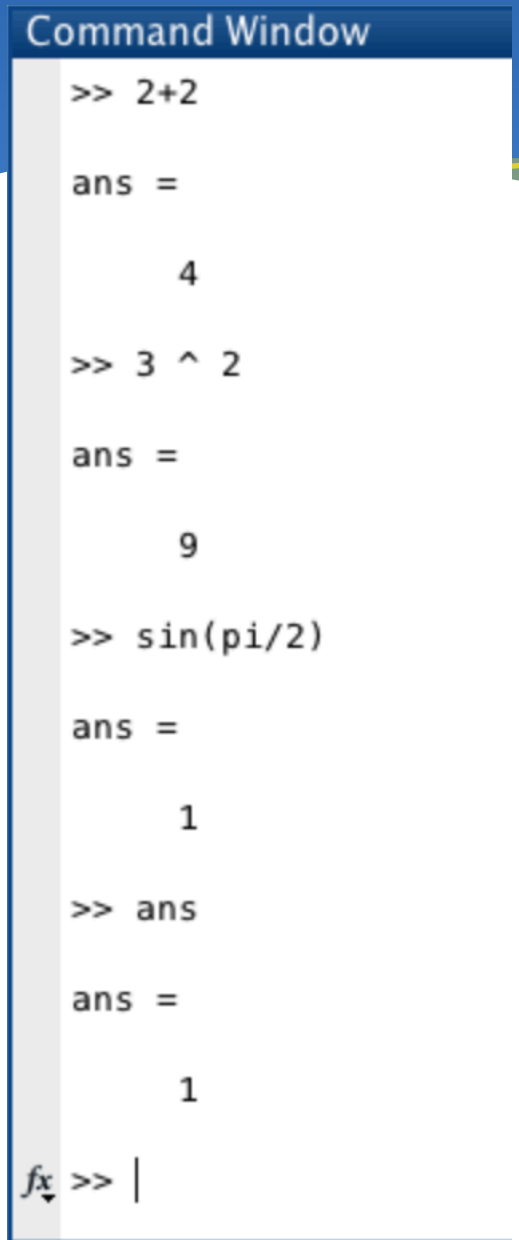
Workspace

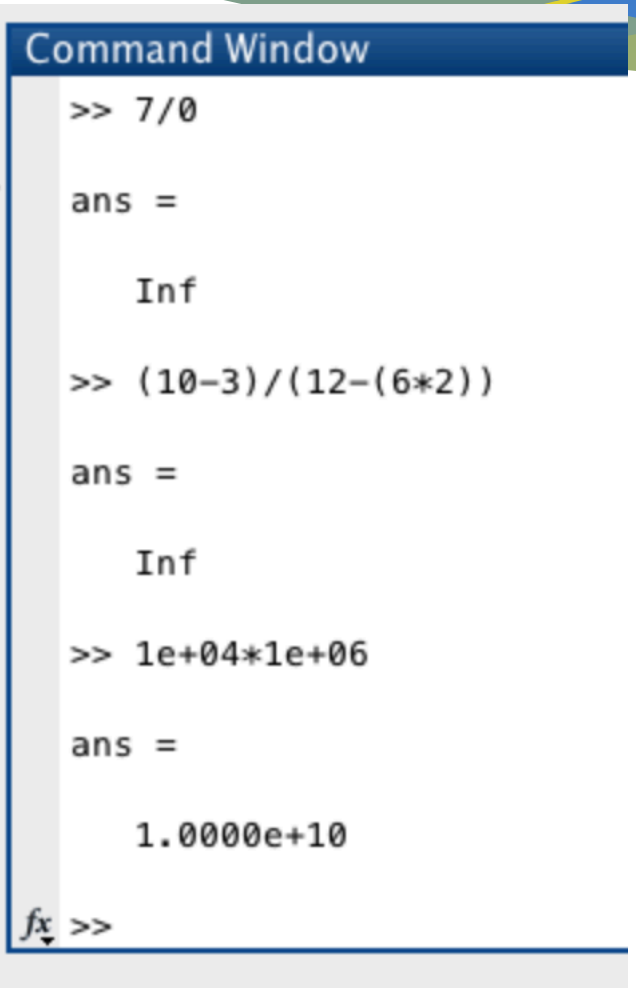

HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
/ ▸ Users ▸ simon ▸ Uni ▸ MATLAB ▸ Workshop
Current Folder
Name
Date Modified
Details
Editor - untitled
SolveBigAllSpecies.m
initial_parameters8.m
untitled
1
Workspace
Name
Command Window
>>
Ready

Editor


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
Users
Current Folder
Name
Date Modified
Details
Editor - untitled
SolveBigAllSpecies.m
initial_parameters8.m
untitled
1
Workspace
Name
>>
Ready

Command Window Basics





Common Arithmetic Operators

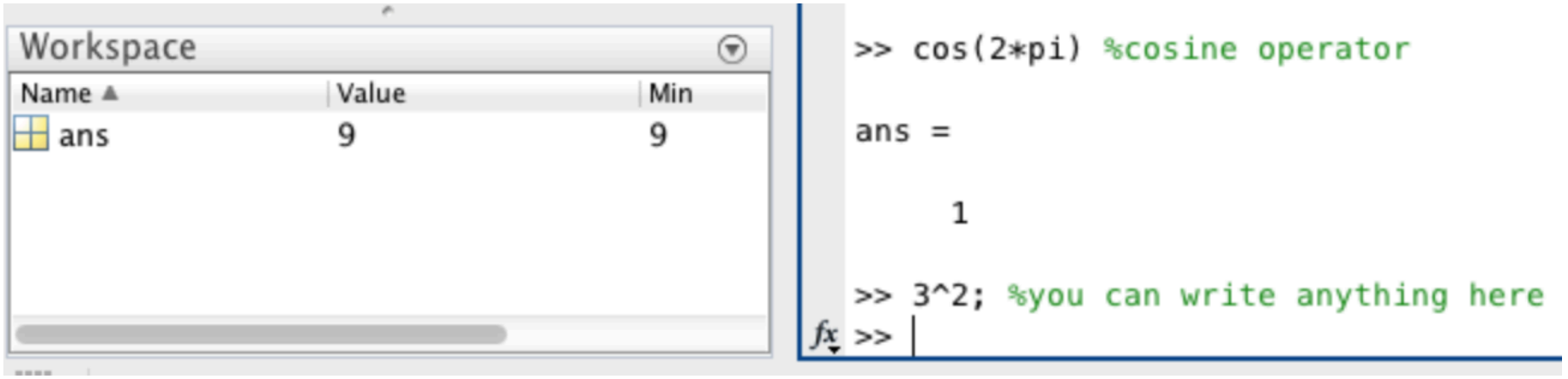
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponential
()	Order operations

Semicolons in MATLAB

Suppress the output from a MATLAB expression

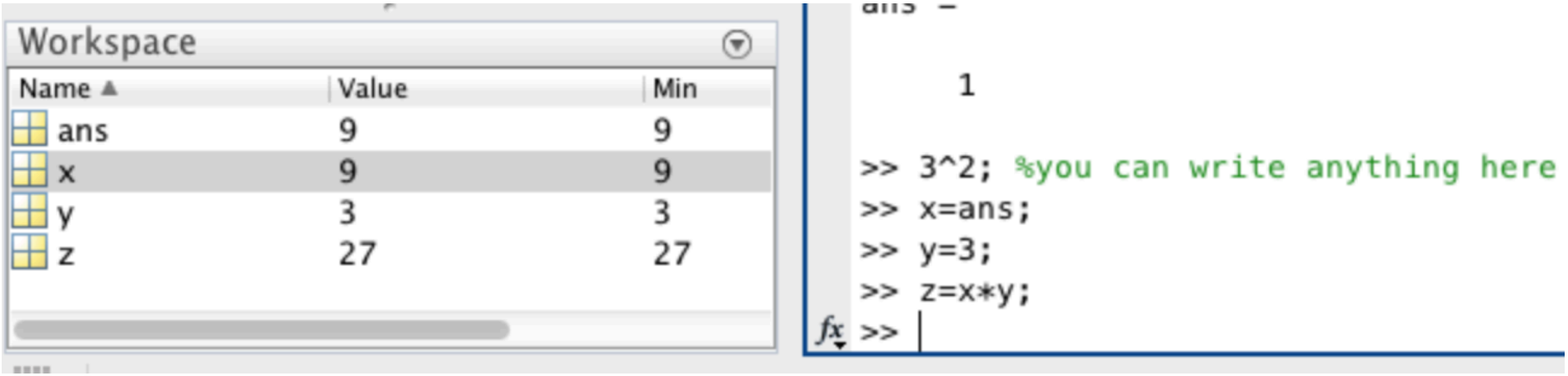
Comments %

Suppress the output from a MATLAB expression



Variables

Variable name = variable value



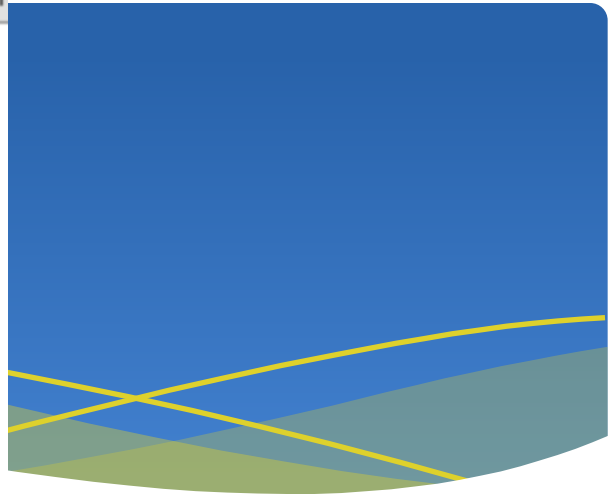
Be careful of i,j

```
>> i

ans =

   0.0000 + 1.0000i
```

Variables 2

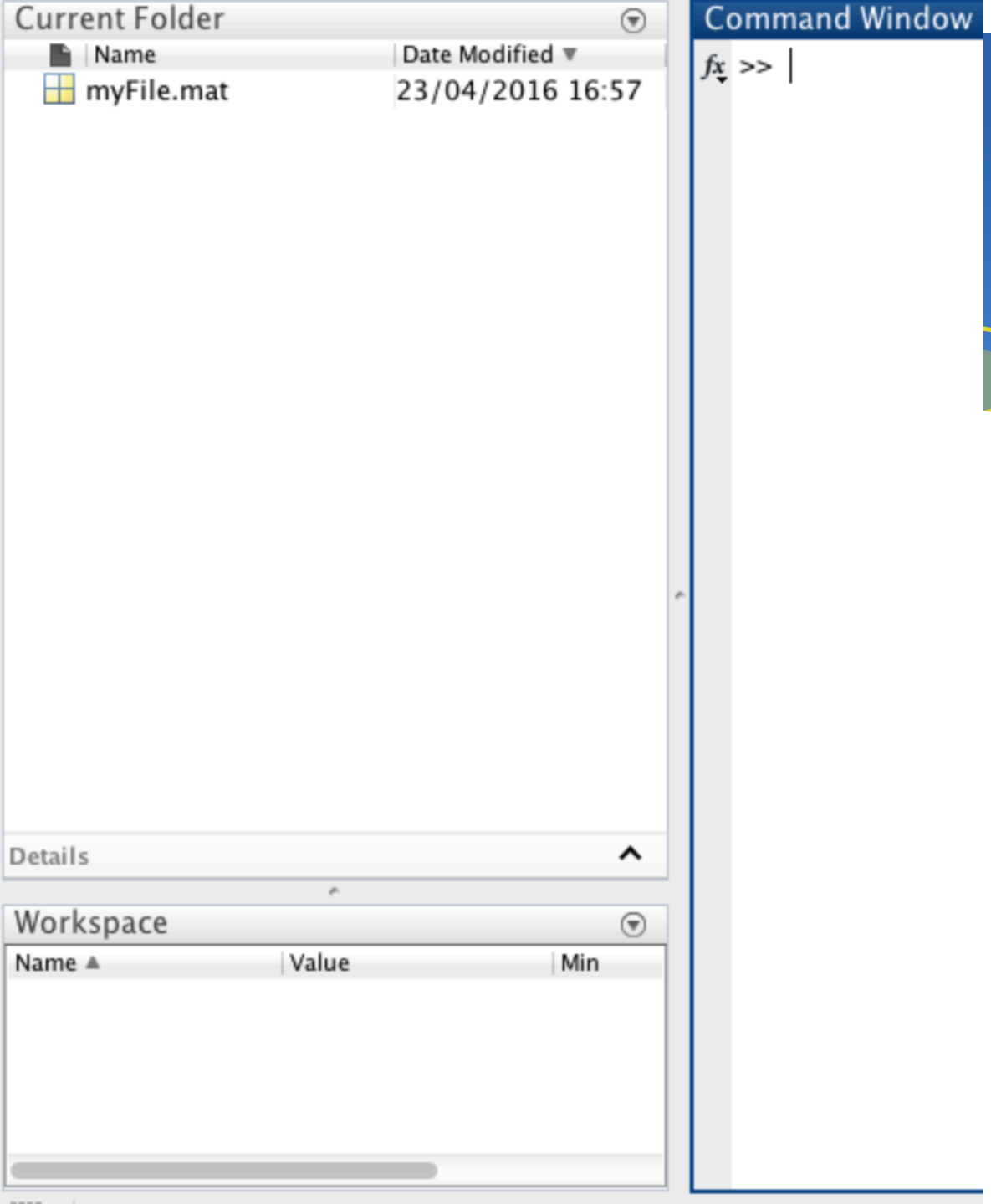
MATLAB is CASE SENSITIVE

```
>> x

x =

     9

>> X
Undefined function or variable 'X'.

Did you mean:
>> x

x =

     9

>>
```

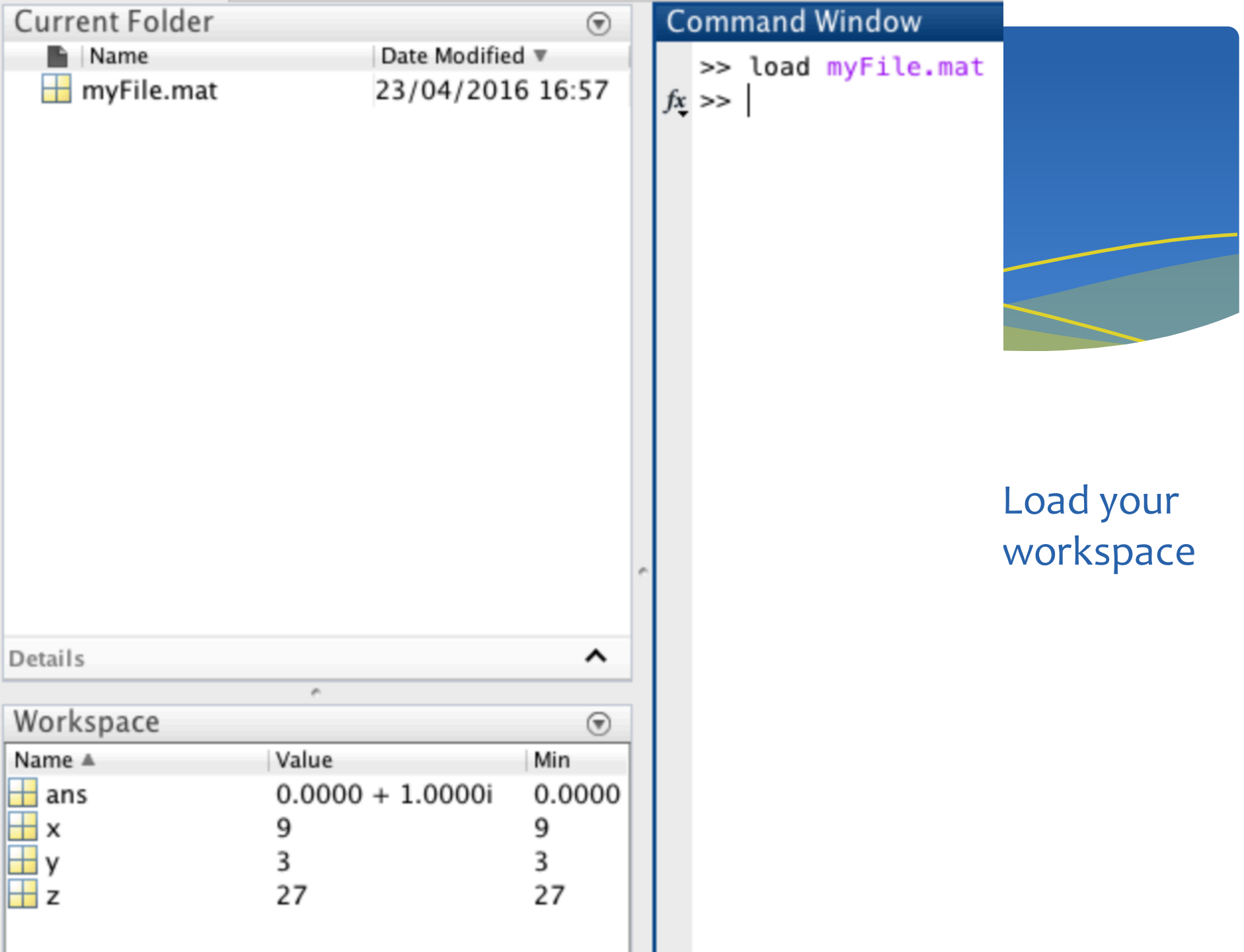


Close MATLAB

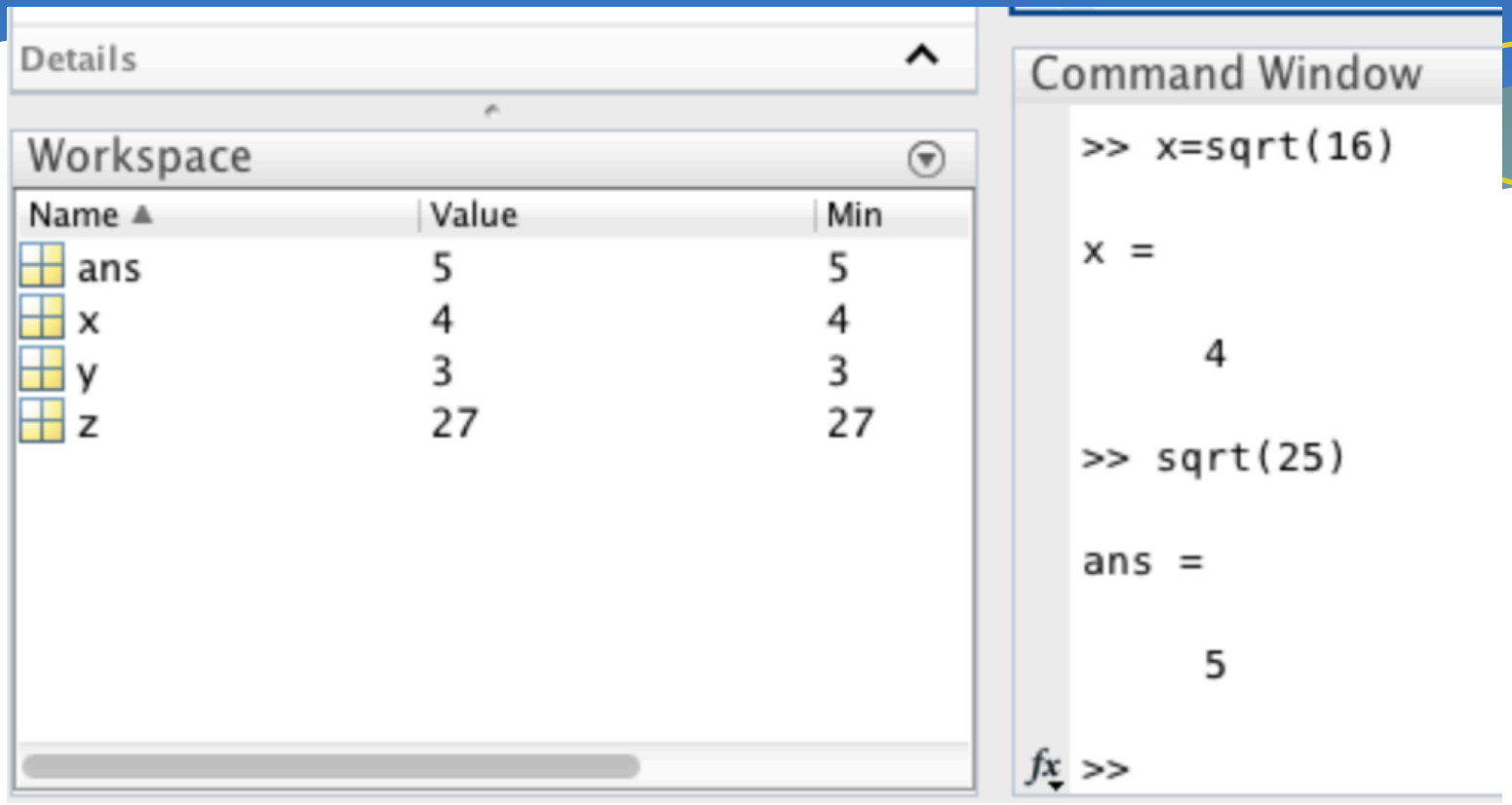
Reopen MATLAB



Initially Workspace is empty

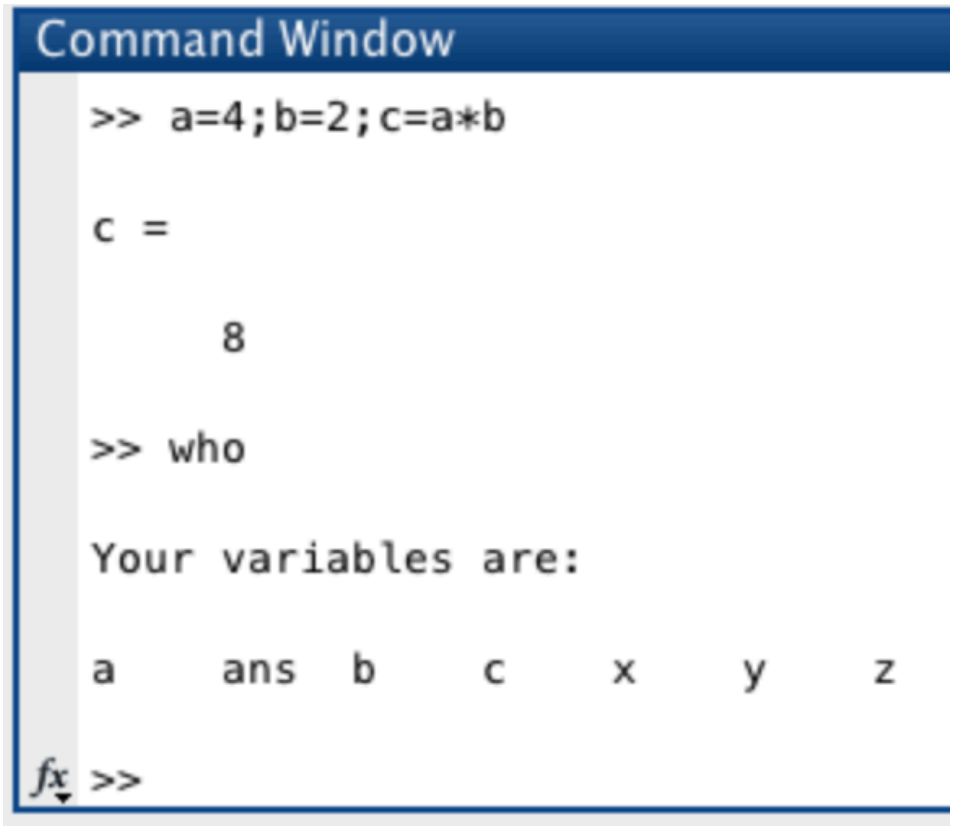

Current Folder
Name
Date Modified
myFile.mat
23/04/2016 16:57
Details
Workspace
Name
Value
Min
ans
0.0000 + 1.0000i
0.0000
x
9
9
y
3
3
z
27
27
Command Window
>> load myFile.mat
>>
Load your workspace

Variables to result of expression



If an expression is not stored as a variable it will be stores as ‘ans’

Multiple assignments and ‘who’



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
```

format

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 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.
```

Put . before an operator to make it element-wise

```
Command Window
>> 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
```

```
>> 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

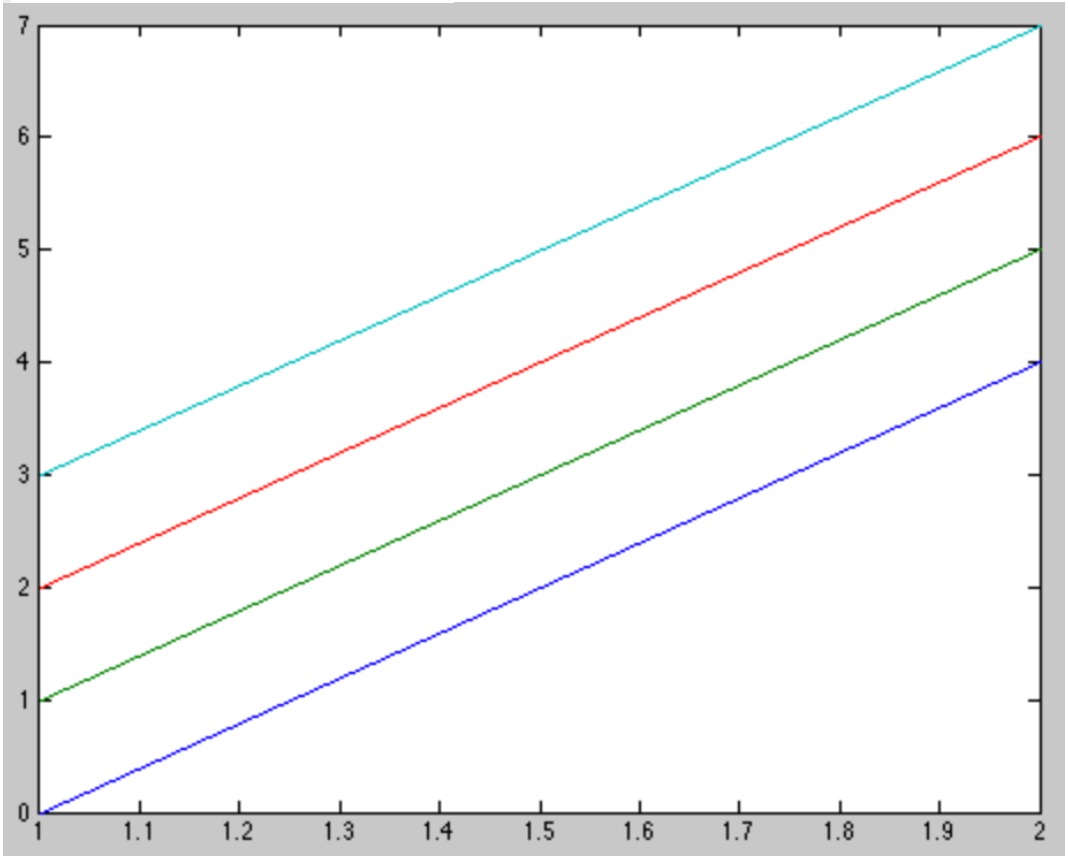
```
>> [myMatrix;myOtherMatrix]

ans =

     0     1     2     3
     4     5     6     7

>> length(ans)

ans =

     4

>> size(ans)

ans =

     1     1

>> size([myMatrix;myOtherMatrix])

ans =

     2     4
```

Why is this 1,1?

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)
```

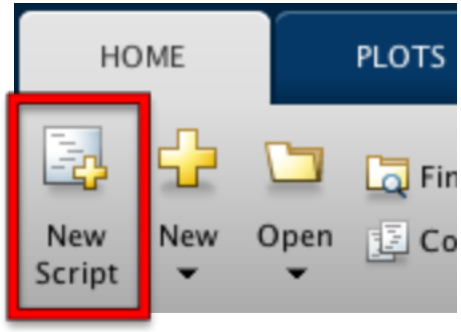


Working with m-files

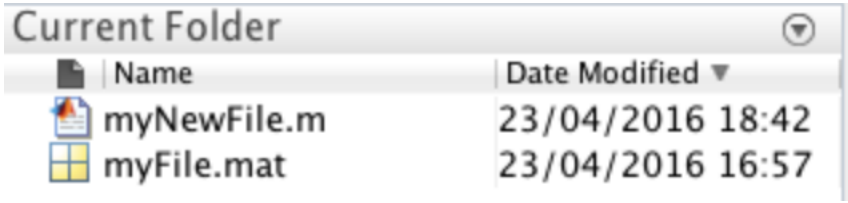
* 3 Ways of running code
  * Command Window ✓
  * Scripts
  * Functions

m-files

New Script




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



Our First Script

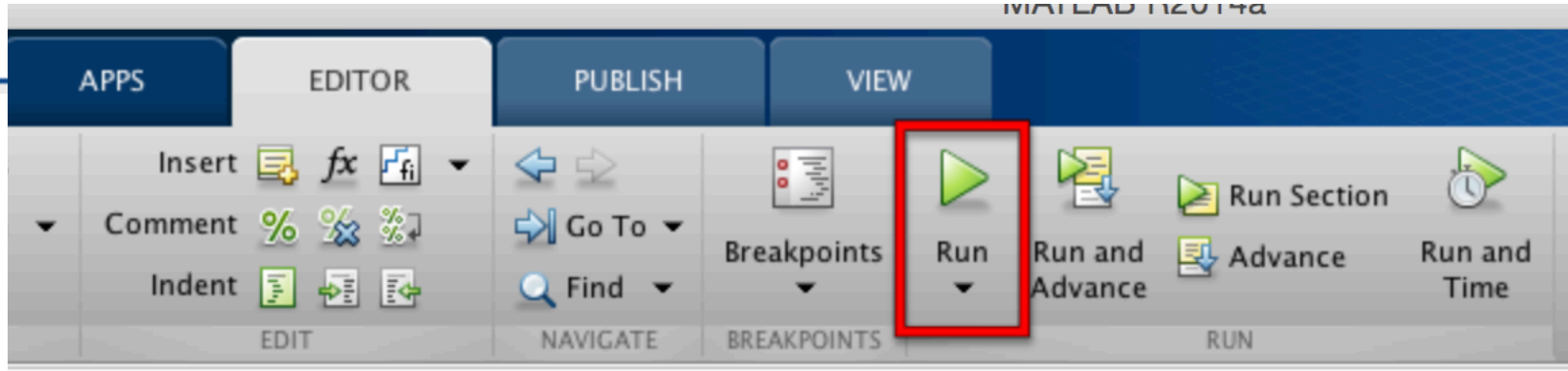
Editor - /Users/simon/Uni/MATLAB/Workshop/myNewFile.m

myNewFile.m

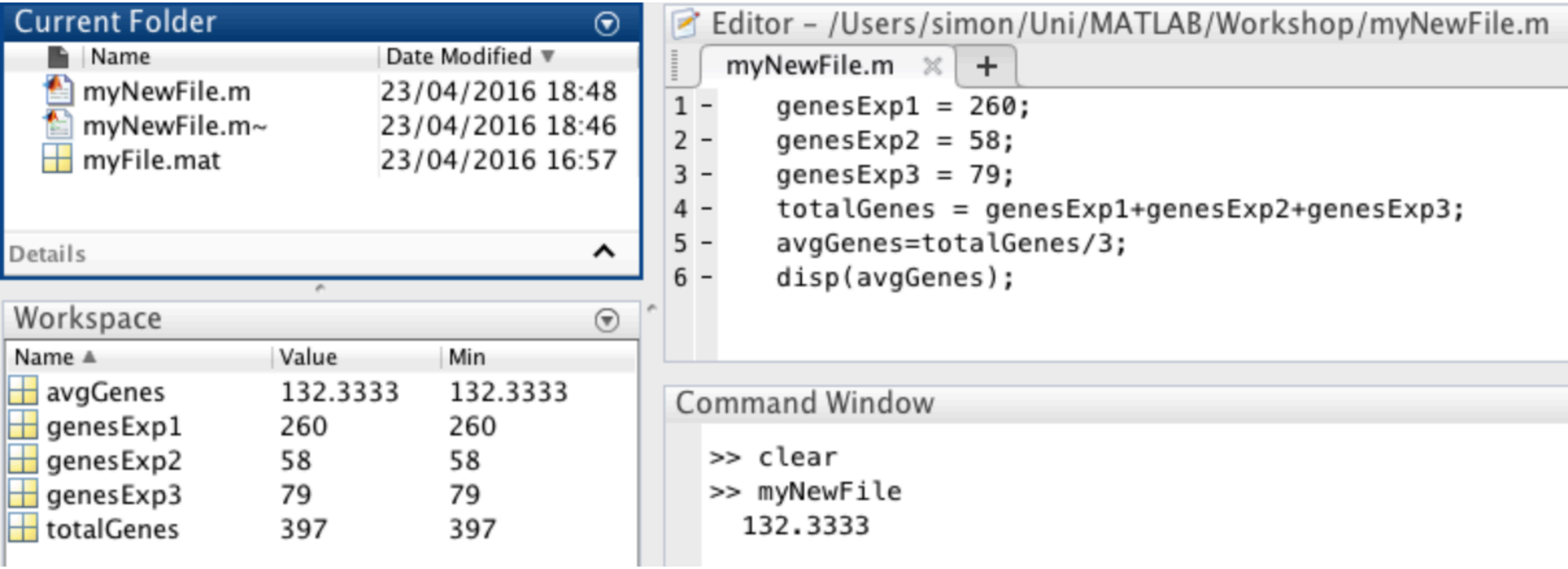
```
1 -    genesExp1 = 260;
2 -    genesExp2 = 58;
3 -    genesExp3 = 79;
4 -    totalGenes = genesExp1+genesExp2+genesExp3;
5 -    avgGenes=totalGenes/3;
6 -    disp(avgGenes);
```

Command Window

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

fx >>
```

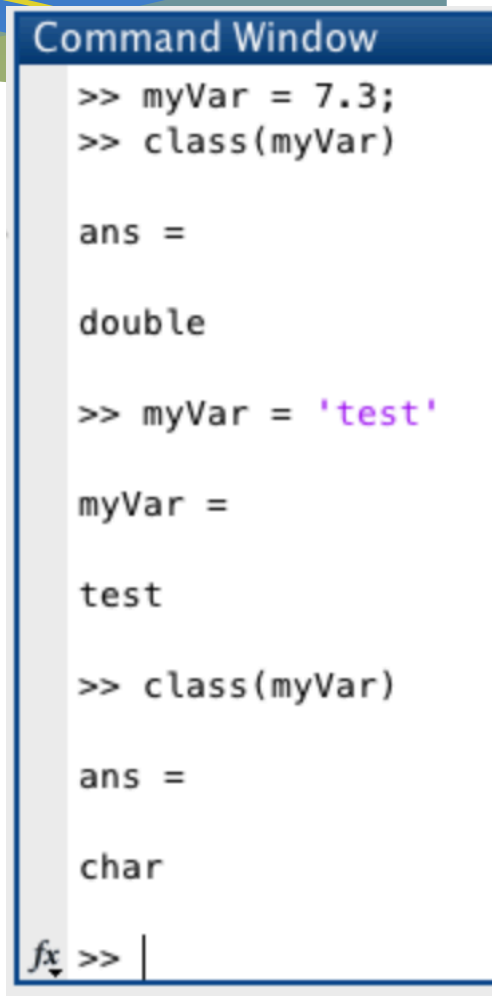


Scripts can create variables in the workspace



Data Types

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



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 | dataTypes.m

```
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 >>
```

Testing Data Types

Editor - /Users/simon/Uni/MATLAB/Workshop/dataTypes2.m

myNewFile.m | dataTypes.m | dataTypes2.m

```
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
```

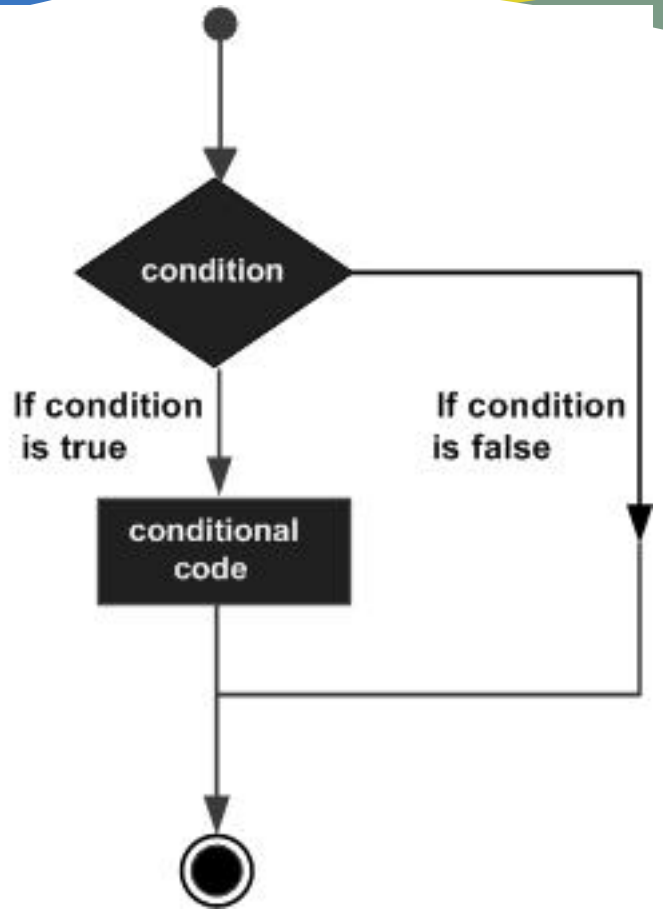
Relational Operators

<	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



Editor – /Users/simon/Un

ifStatement.m

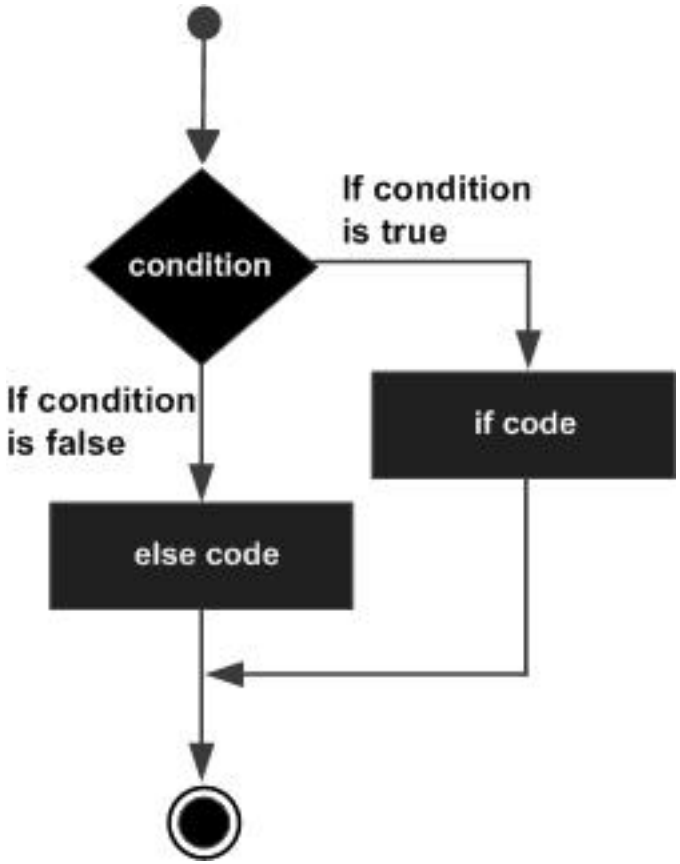
```
1 -     exp1 = 400;
2 -     exp2 = 500;
3 -     if (exp1 >= exp2)
4 -         max = exp1
5 -     end
```

Never run

Command Window

```
>> ifStatement
```

If else Statement



Editor – /Users/simon/Uni/

myNewFile.m | dataType

```
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

```
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

Editor - /Users/simon/Uni/MATLAB/Workshop/forLoopExample

ifStatement.m | forLoops.m* | forLoopExample.m

```
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
>>
```

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

Editor – /Users/simon/Uni/MATLAB/Workshop/forLoops.m

ifStatement.m | forLoops.m | forLoopExample.m | +

```
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 | forLoops.m | forLoopExample.m | +

```
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 >>
```