Unit-1

Basics of MATLAB

- MATLAB is an interactive program for numerical computation and data visualization; it is used extensively by control engineers for analysis and design
- The material provides a gentle introduction to the MATLAB computing environment, and designed to give a basic understanding of MATLAB. No prior programming experience or knowledge of MATLAB is assumed
- The name **MATLAB stands for MATrix LABoratory**. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

- MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming.
- These factors make MATLAB an excellent tool for teaching and research.

Features of MATLAB

- MATLAB is a high-level programming language with data structures, control flow statements, functions, output/input, and object-oriented programming
- It permits both, rapidly creating speedy throw-away programs, and creating complete, complex and large application programs.
- It provides an interactive environment that allows iterative exploration, design, and problem-solving.
- It is a bunch of tools that a programmer can use. It includes abilities for handling the variables in the workspace & importing/exporting data

Features of MATLAB

- It also contains tools for development, handling, debugging, and profiling MATLAB files and offers built-in graphics useful for data visualizing, and tools for generating custom plots.
- It offers a huge library of mathematical functions needed for computing statistics, linear algebra, numerical integration, filtering, Fourier analysis, optimization and solving regular differential equations.
- MATLAB Application Program Interfaces (APIs) allow users to write C/C++ and Fortran programs that directly interact with MATLAB

Features of MATLAB

- A Toolbox is a set of functions designed for a specific purpose and compiled as a package. These Toolboxes include MATLAB code, apps, data, examples and the documentation which helps users to utilize each Toolbox. Users can compile MATLAB files to create toolboxes if they require sharing with others
- There are separate Toolboxes available from Mathworks, to be used for specific purposes, for example, text analytics, image processing, signal processing, deep learning, statistic & machine learning, and many more.

Language Fundamentals

- *MATLAB* is an abbreviation for "matrix laboratory." While other programming languages usually work with numbers one at a time, MATLAB® operates on whole matrices and arrays.
- Language fundamentals include basic operations, such as creating variables, array indexing, arithmetic, and data types.

MATLAB PROGRAMMING BASICS

- There are some important windows inside MatLab GUI, in default view, use and work of each window is described below:
- Current directory
- Command window
- Workspace
- Command history

Current directory

- Current directory: This is the window from where; access of files and folders is easy. The locations shown in this window is called present working directory. MatLab program files under present working directory are directly accessed and executed.

Command window

- Command window: This is the main window of MatLab, from where all the commands and MatLab program file (scripts) are executed. Results are also displayed on this window.

Workspace

- Workspace: This is the place, where the entire variables are visible with their size, type and values. This window can be further explored with each variable as Variable Editor Window.

Command history

- Command history: This is the window, where all the previous commands are visible. It is the type of log record of previous commands executed.

CALCULATIONS IN COMMAND WINDOW

- Examples and exercise of simple calculations with mathematical functions over constants and variables are given below,(double arrow “>>” symbol represents the command prompt terminal).
- SIMPLE CALCULATION
- USE OF VARIABLES
- FLOATING POINT NUMBER PRECISION CONTROL

SIMPLE CALCULATION

- Calculations can be directly and easily performed on command prompt.
- >> **-4/(2.9+6.13)^3**
- ans = -0.0054
- **>> (2+5i)*(2-5i)**
- ans = 29
- **>>cos(pi/4)**
- ans = 0.7071
- **>>exp(tan(0.5))**
- ans = 1.7269

USE OF VARIABLES

- Calculations with variables and constants can also be directly and easily performed on command prompt as given below,
- >> m=7;
- >> n=9;
- >>m+n
- ans = 16
- r=pi/2 (pi is constant)
- r = 1.5708
- >> s=sin(r)
- s =1

FLOATING POINT NUMBER PRECISION CONTROL

- When displaying the results of calculations or values of variables, “format” command is used to control the precision of numbers.
- >>format short
- >> 3.4
- ans = 3.4000 (only 4 decimals in the result)
- >>format long
- >> 3.4
- ans = 3.400000000000000 (15 decimals in the result)

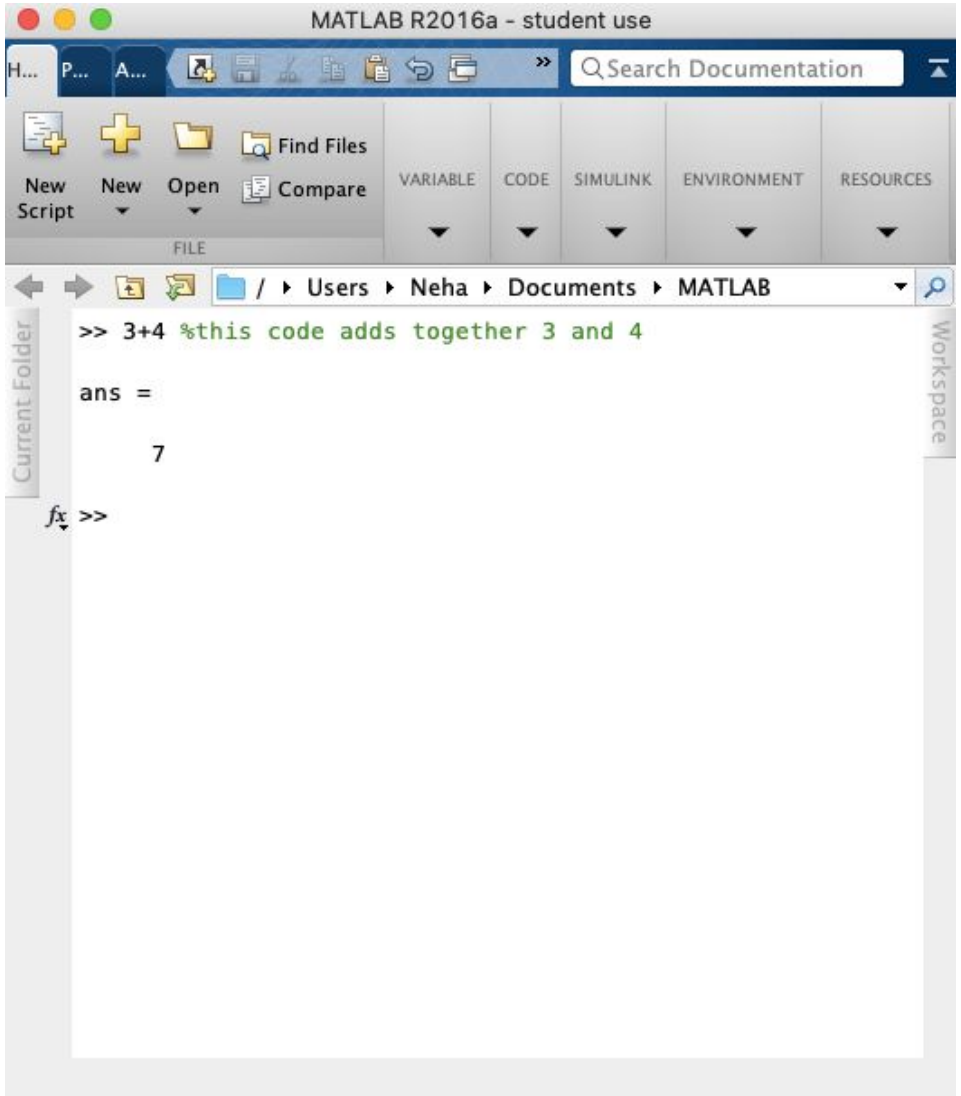
ARITHMETIC OPERATORS

- Since all types of data in MatLab are stored in the form of arrays, MatLab provides two different kinds of arithmetic operations, i.e. array operations and matrix operations. MatLab array operations are just ordinary arithmetic operations that supports multidimenstional arrays for processing element by elemt operations.
- While MatLab matrix operation are ordinary matrix operations that following the rules of linear algebra. Both array operations and matrix operations share the similar symbols of operation, a period characater, "." is used to distinguish the array operations from the matrix operations.
- However, as the addition and subtraction for both matrix operation and array operation are the same, the period characater, "." is not necessary and the character pairs ".+" and ".-" are not used. Besides, the 1-by-1 array, scalar, is also a special type of MatLab array.
- A 1-by-1 array, scalar can have array operation with an array of any size also. A 1-by-1 matrix, scalar can also have martrix operation with a matrix of any size, but limited by the matrix multiplication, the 1-by-1 matrix, scalar can only be the divisor of the right and left division.

Arithmetic Operators	Matrix Arithmetic Operations		Element-Wise Array Arithmetic Operations	
	Syntax	**Description**	**Syntax**	**Description**
Addition +	A+B	Addition	A+B	Addition
	+A	Unary Plus	+A	Unary Plus
Subtraction -	A-B	Subtraction	A-B	Subtraction
	-A	Unary Minus	-A	Unary Minus
Multiplication *	A*B	Matrix Multiplication	A.*B	Array Multiplication
Right Division /	A/B	Forward Slash or Matrix Right Division	A./B	Array Right Division
Left Division \	A\B	Backslash or Matrix Left Division	A.\B	Array Left Division
Power ^	A^B	Matrix Power	A.^B	Array Power
Transpose '	A'	Matrix Transpose or Complex Conjugate Transpose	A.'	Array Transpose

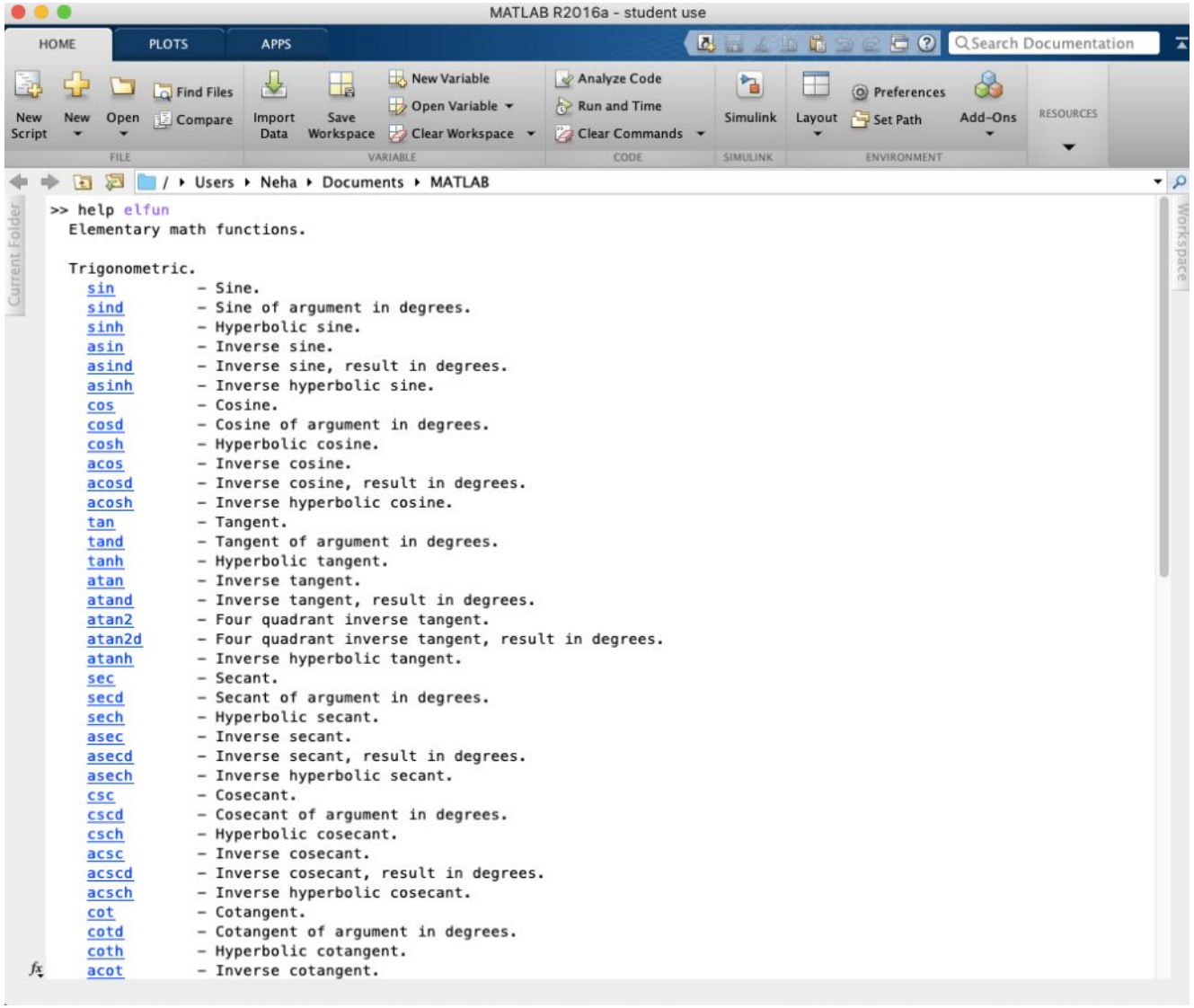
Adding Comments to Code

- Comments are any words or code that you don't want to put into your actual code. You will use the percent sign, **%**, before you begin typing, and MATLAB will color these in green.
- They can go anywhere in a code. Usually it is used when we want to describe each line of code without that description actually being a part of the code itself. For example, I have put a comment in green below explaining what my code does:



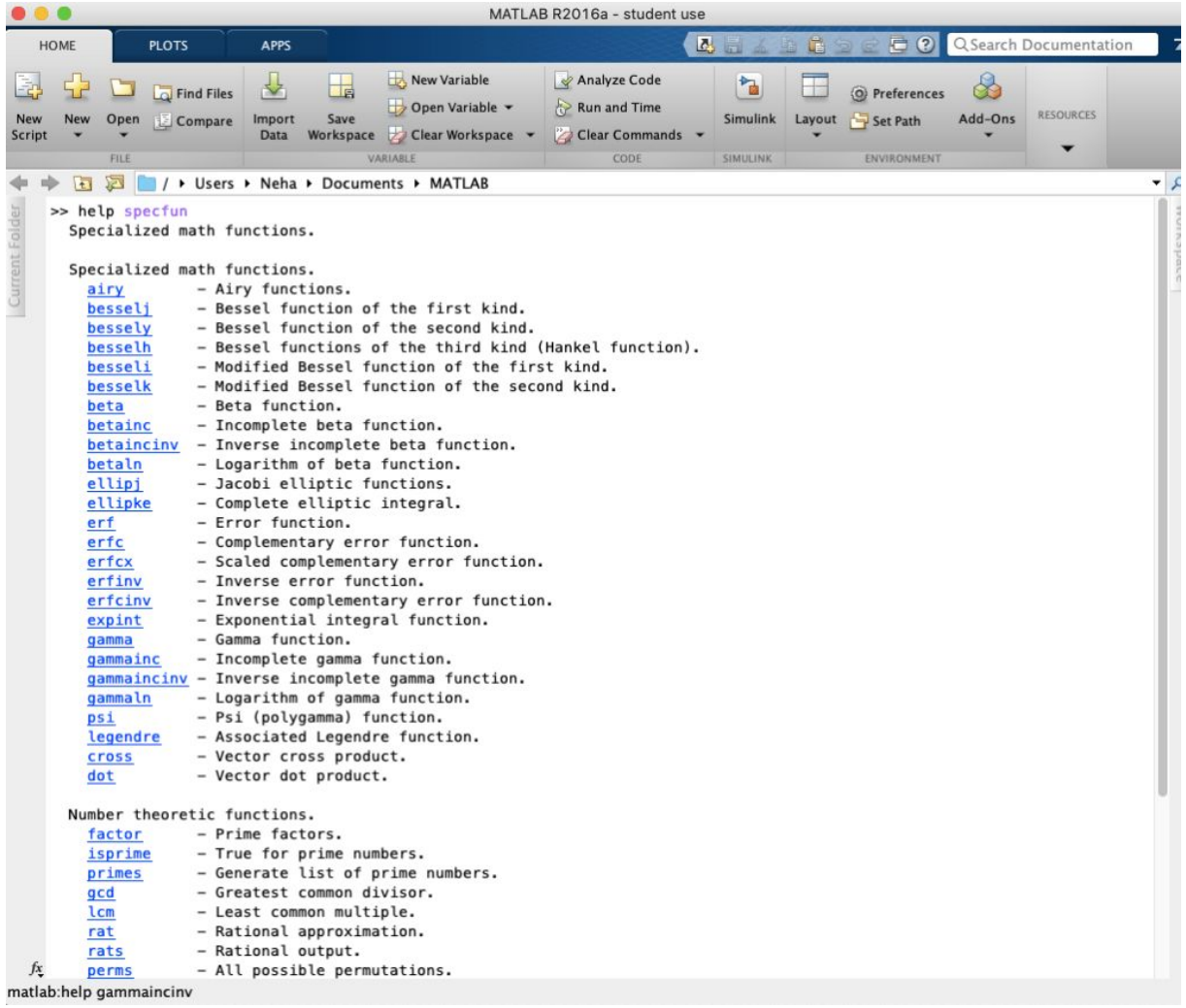
Command : help elfun

- **elp elfun" :** Typing "help elfun" in the command window gives a list of **el**ementary **fun**ction s, many of which I'm talking about below (eg. trig functions, square root function, etc). This list goes on for a while but here's a screenshot of the first part of it:



Command : help specfun

“help specfun” : Typing “help specfun” in the command window gives a list of **spec**ial **fun**ctions, many of which are not commonly used except in higher level calculations. This list goes on for a while but here’s a screenshot of the first part of it:



MANAGING THE WORKSPACE AND MISCELLANEOUS COMMANDS

- Some miscellaneous commands are given in Table .These command will help to speed-up the programming, and also helps in managing the workspace.

Commands	Operation
Clc	This will clear the Command Window
Clear	To empty workspace, delete all the variables
Who	To know variables names in workspace
Whos	To know variables' name with size and memory consumed
Diary	diary on, diary off, diary , this will create a text file of command history of command prompt.
ctrl + c	**To abort a MatLab computation**
Help	help , explore about the given command
Lookfor	Lookfor , search the given key word and suggest matched commands

Common Built-In Functions and Predefined Constants

Here are two tables below. The first table has a list of some common built-in elementary functions.

The second table has a list of some predefined constants you can enter into MATLAB. "Predefined" meaning that MATLAB will know what the value of these are when you enter them – for example, if you put "pi" it will know the value is 3.14159…. and so on.

Table 2.1: Elementary functions

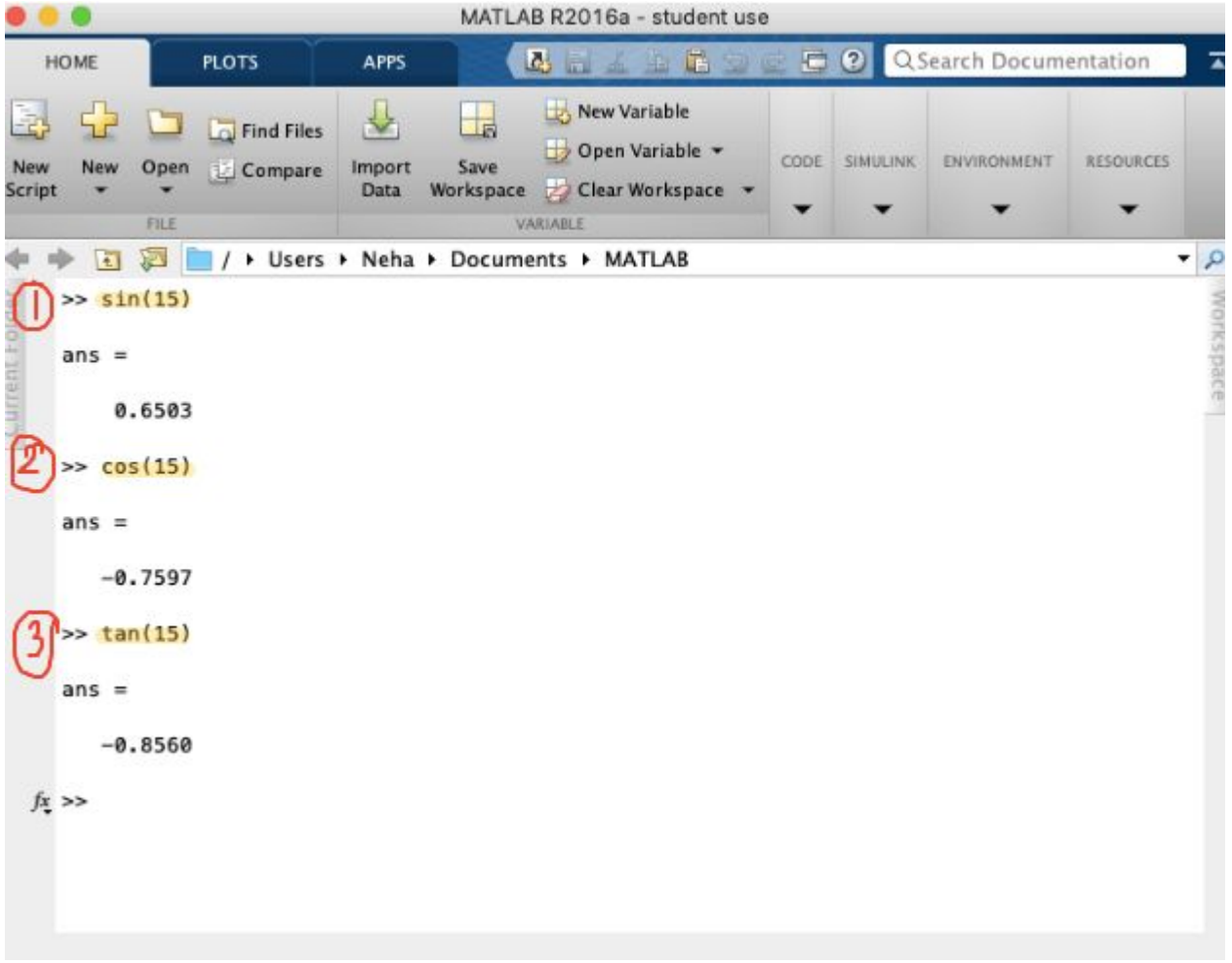
`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

Table 2.2: Predefined constant values

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

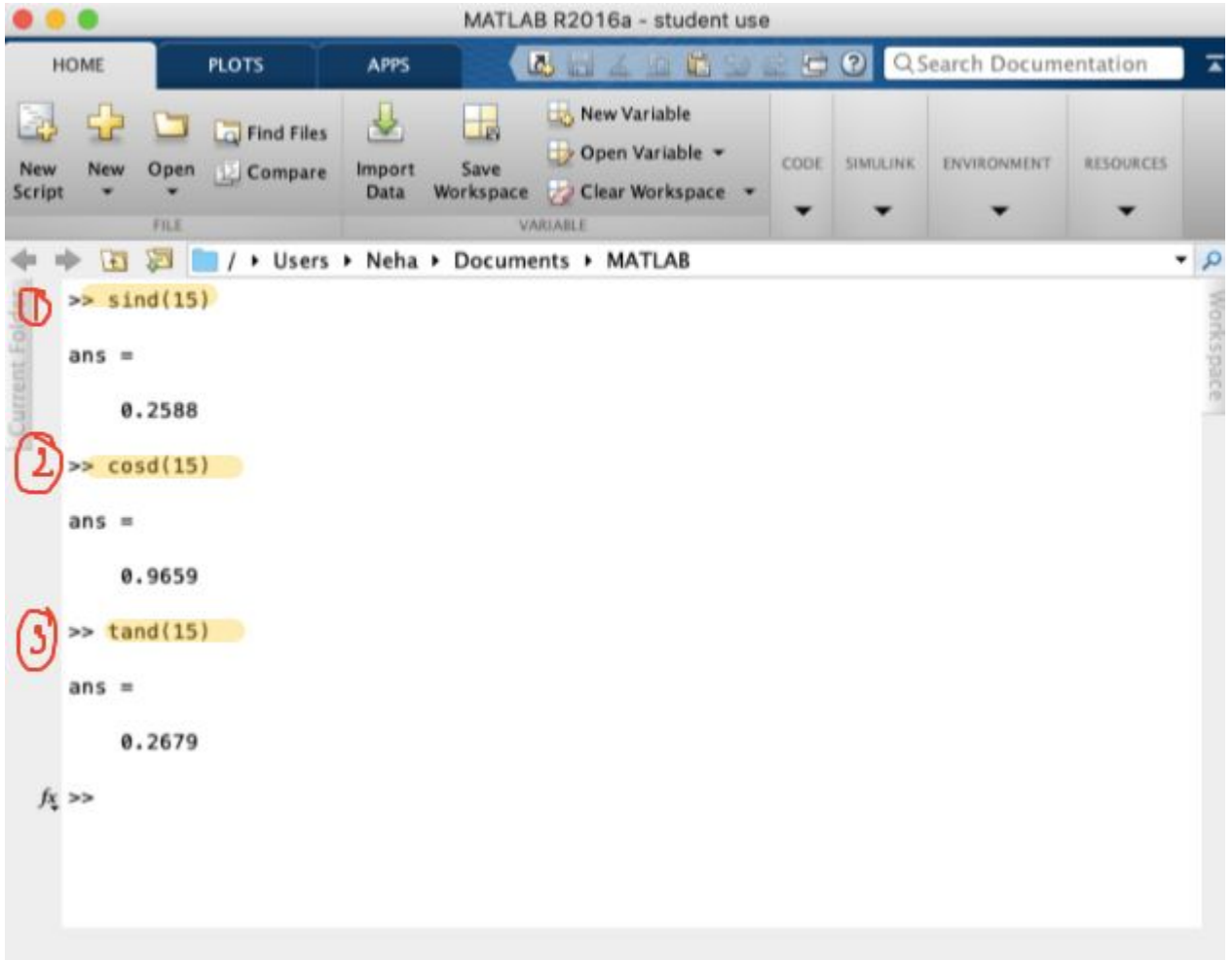
Built-in functions

Trig Functions IN RADIANS (sin, cos, tan): Here I have done examples using the trig functions: sin(x), cos(x) and tan(x). I have highlighted each example. You have to put a number in the brackets rather than x (you can even use “pi” or other predefined constants in the brackets that have number values). These values of x are read in Radians:

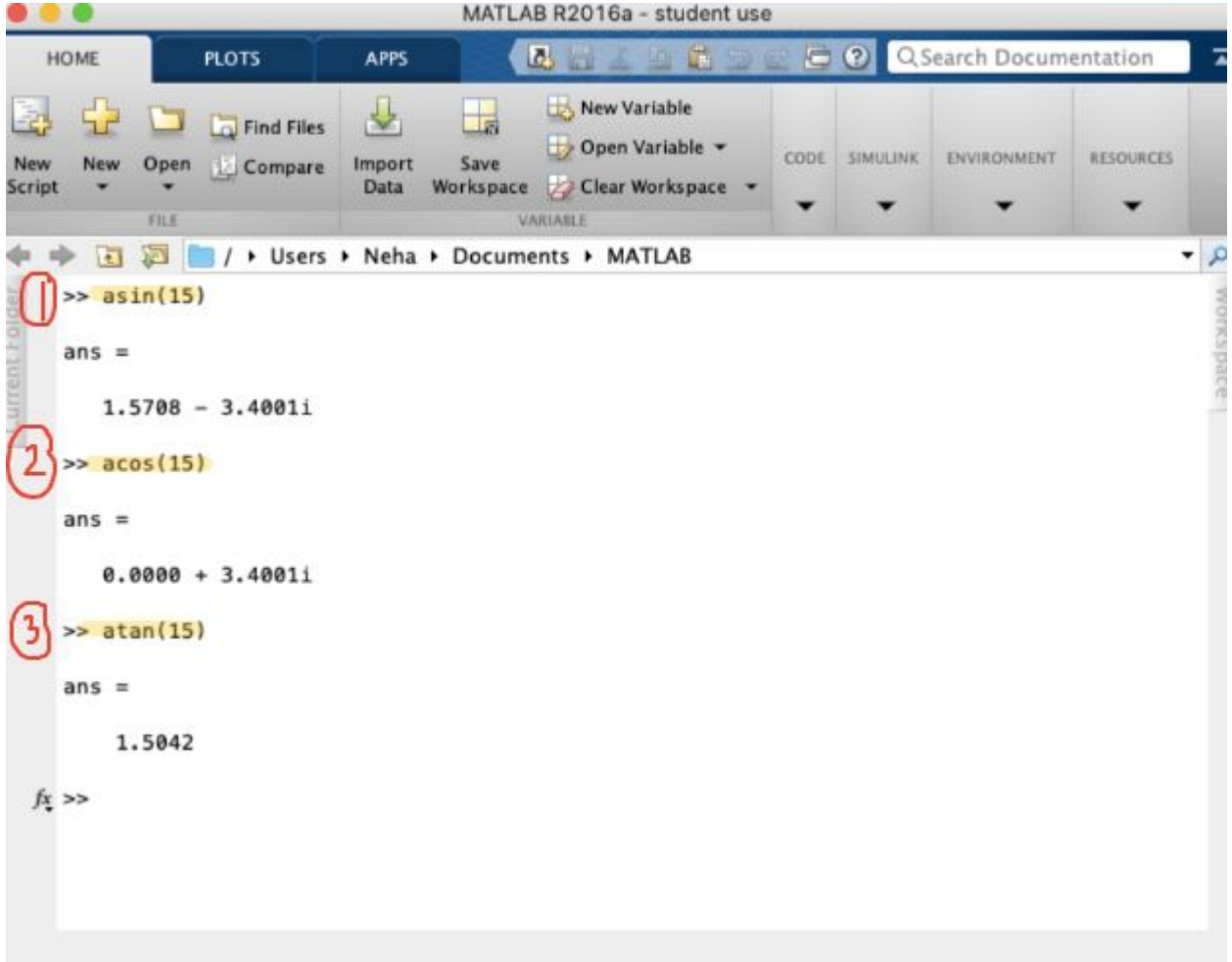


Trig Functions IN DEGREES: Part (a) above explained how to use the trigonometric functions sin(x), cos(x) and tan(x). But MATLAB will read the numbers you put in for "x" in RADIANS.

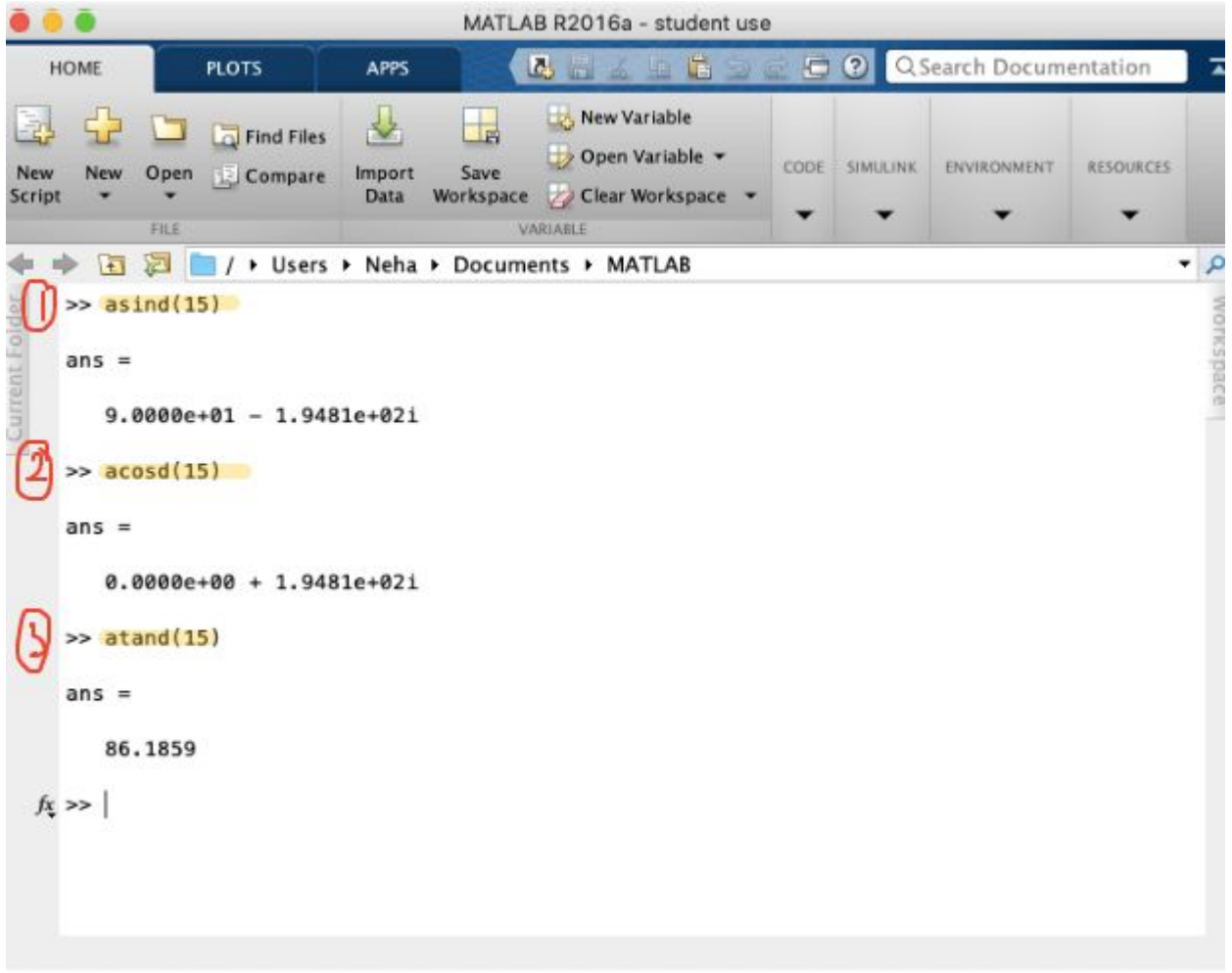
To do this in degrees , you need to use "sind" "cosd" and "tand." Otherwise everything works the same way as it did for part (a). See the example below and note how the answers here are different than they are in part (a), even though I'm still using 15 in the brackets.



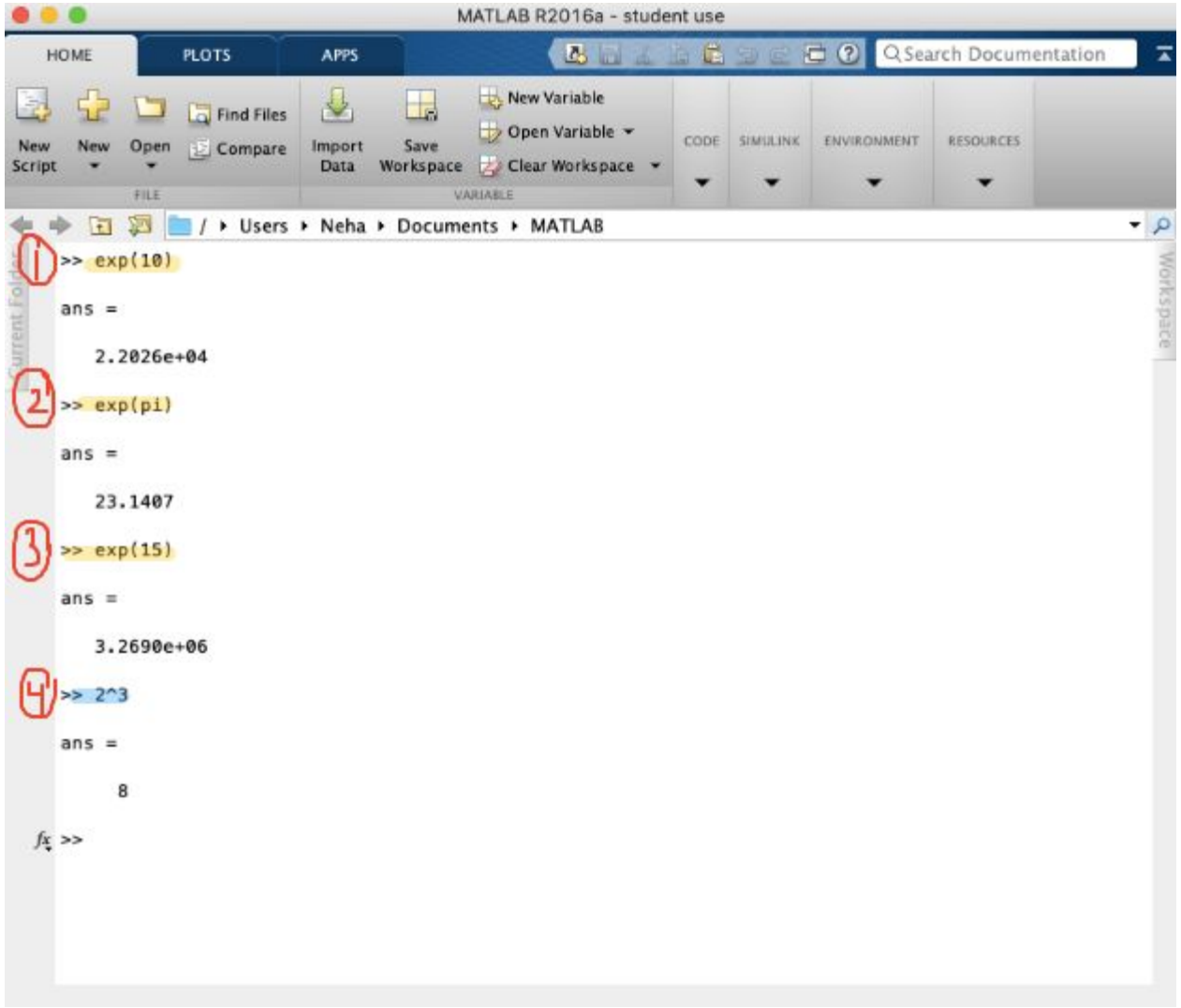
Inverse Trig Functions IN RADIANS (arcsin, arccos, arctan): Usually, the inverse of the trig functions sin, cos, and tan are done using the exponent of -1 on a calculator. But when there's an "arc" in front of the words, that also means to take the inverse. So arcsin(x), arccos(x), and arctan(x) respectively use the functions asin(x), acos(x), and atan(x). See this example below:



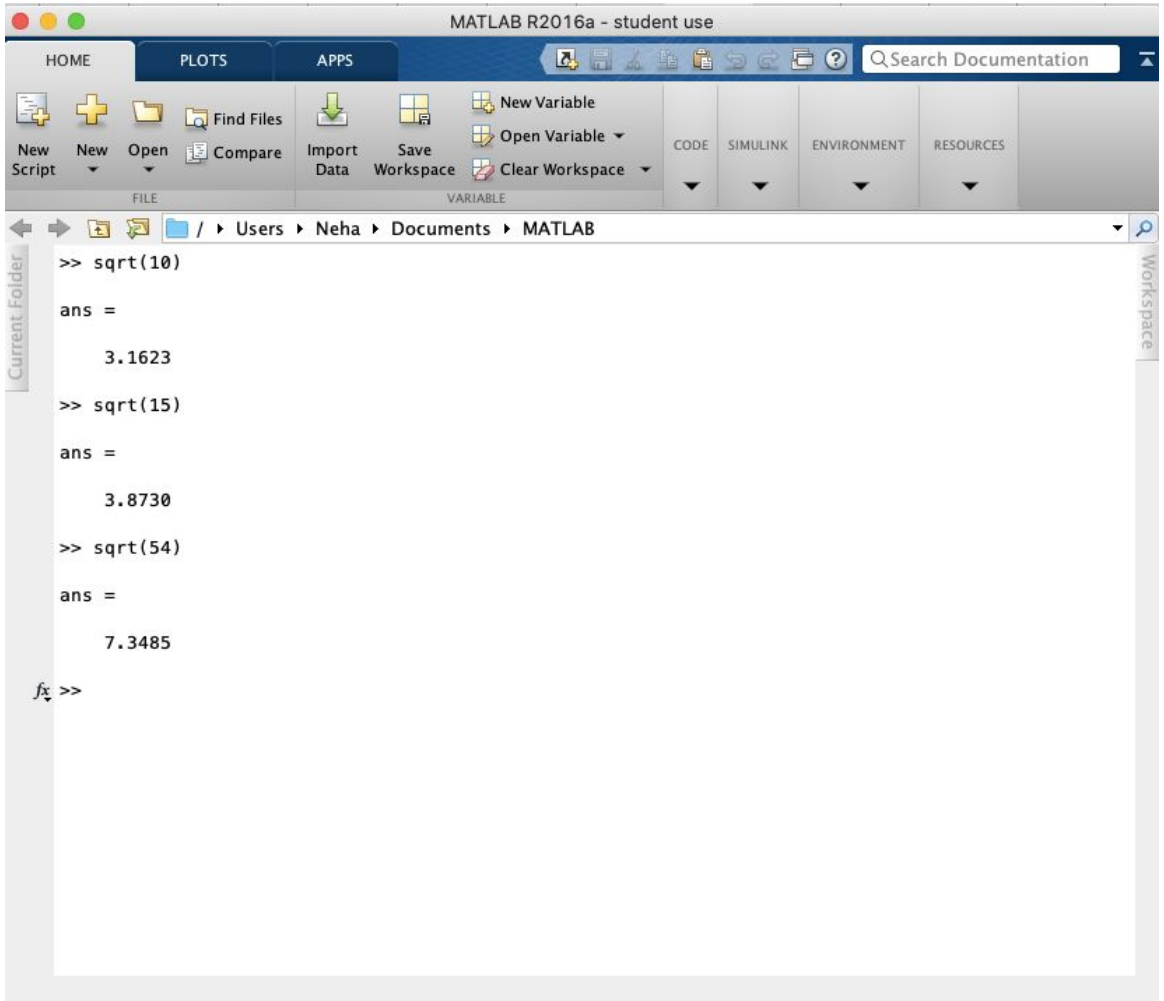
Inverse Trig Functions IN DEGREES: This is the same reasoning as I explained in part (b). To use this function in degrees, we need “asind” “acosd” and “atand.” Take a look at this example:



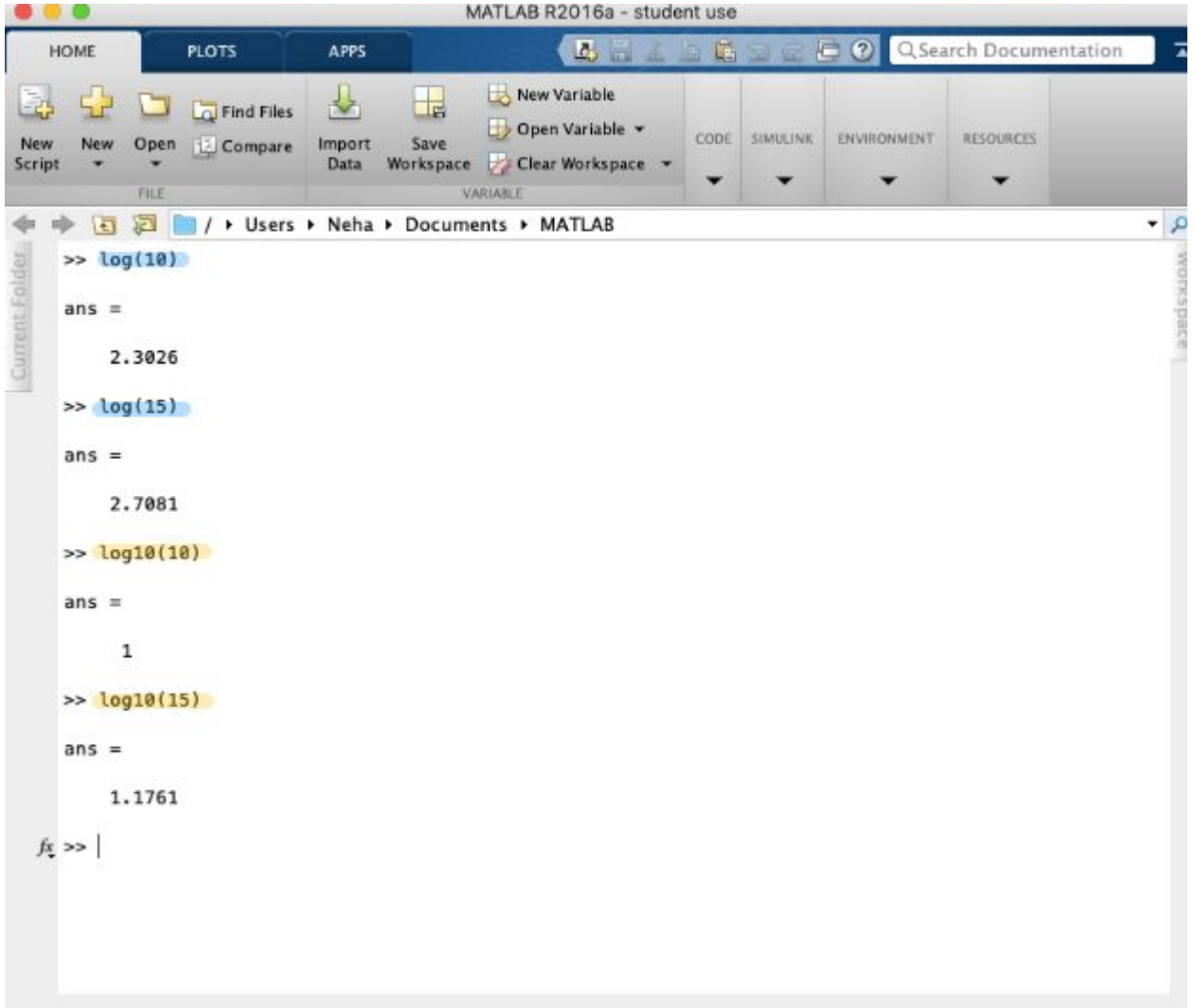
Exponential Function (and Exponents in general): To do exponents, eg. 2 to the power of 3, you need the ^ key (eg. 2^3 = 8).
To use the Exponential Function (e^x) use “exp(x)” where ‘x’ can be any value. I’ve done examples of both the exponential functions (in yellow) and using exponents in general (highlighted in blue):



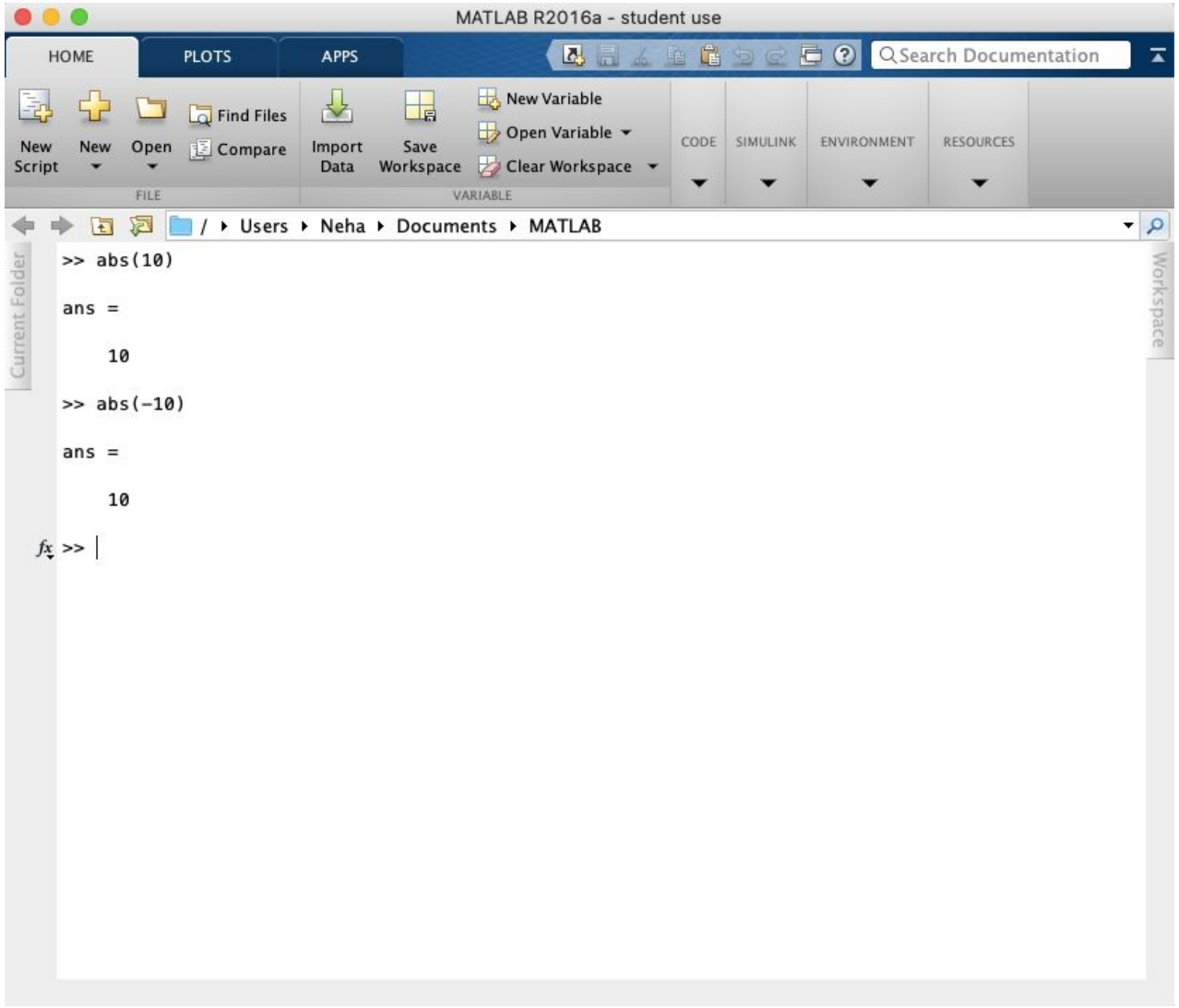
Square Root Function: To find the square root of a number, use “sqrt(x)”, where x can be any value. I’ve done a couple examples using it below:



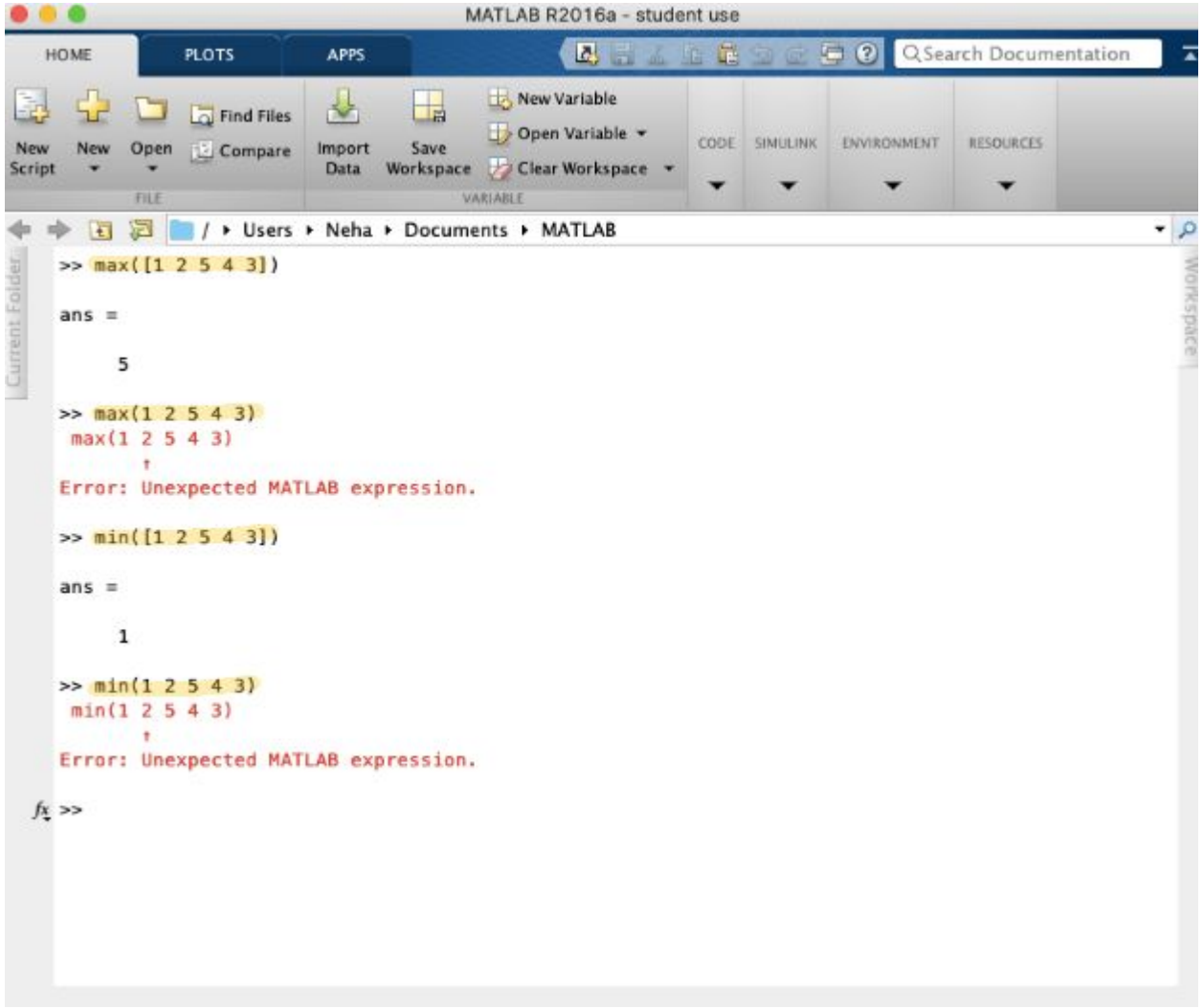
Natural Logarithm (which is base ‘e’ log) and Common Logarithm (base 10 log) – Natural Logarithm is more commonly seen as **“ln,”** and the examples are in blue. Common Log is usually seen as just **“log,”** and the examples are in yellow:



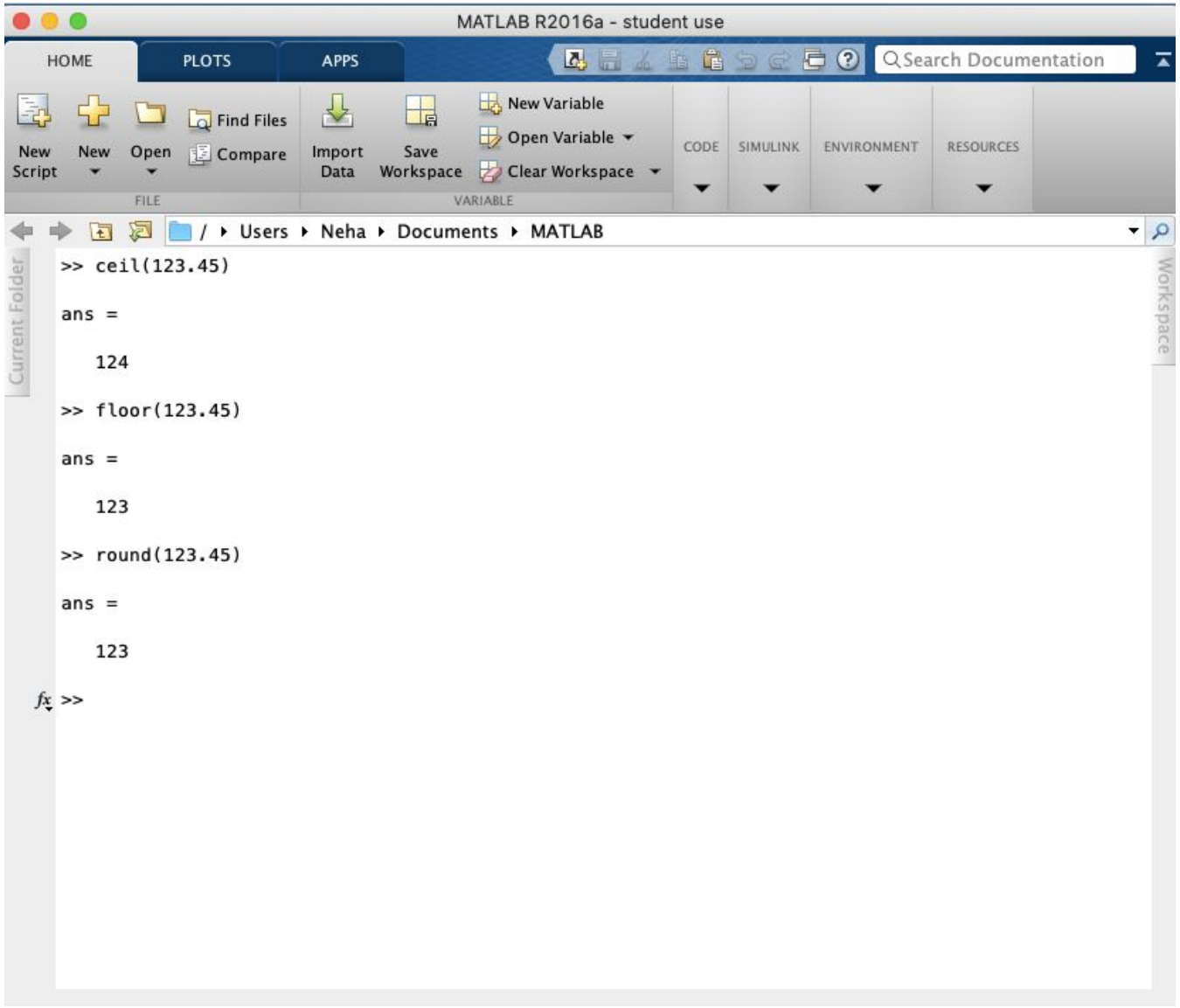
Absolute Value Function: In math, absolute value functions make all numbers positive. So in MATLAB, if you enter a negative number, you'll get the same number back but positive. If you enter a positive number to begin with, you'll get the same number back, positive still. Use "abs(x)" for this. I've done two examples below using 10 and -10.



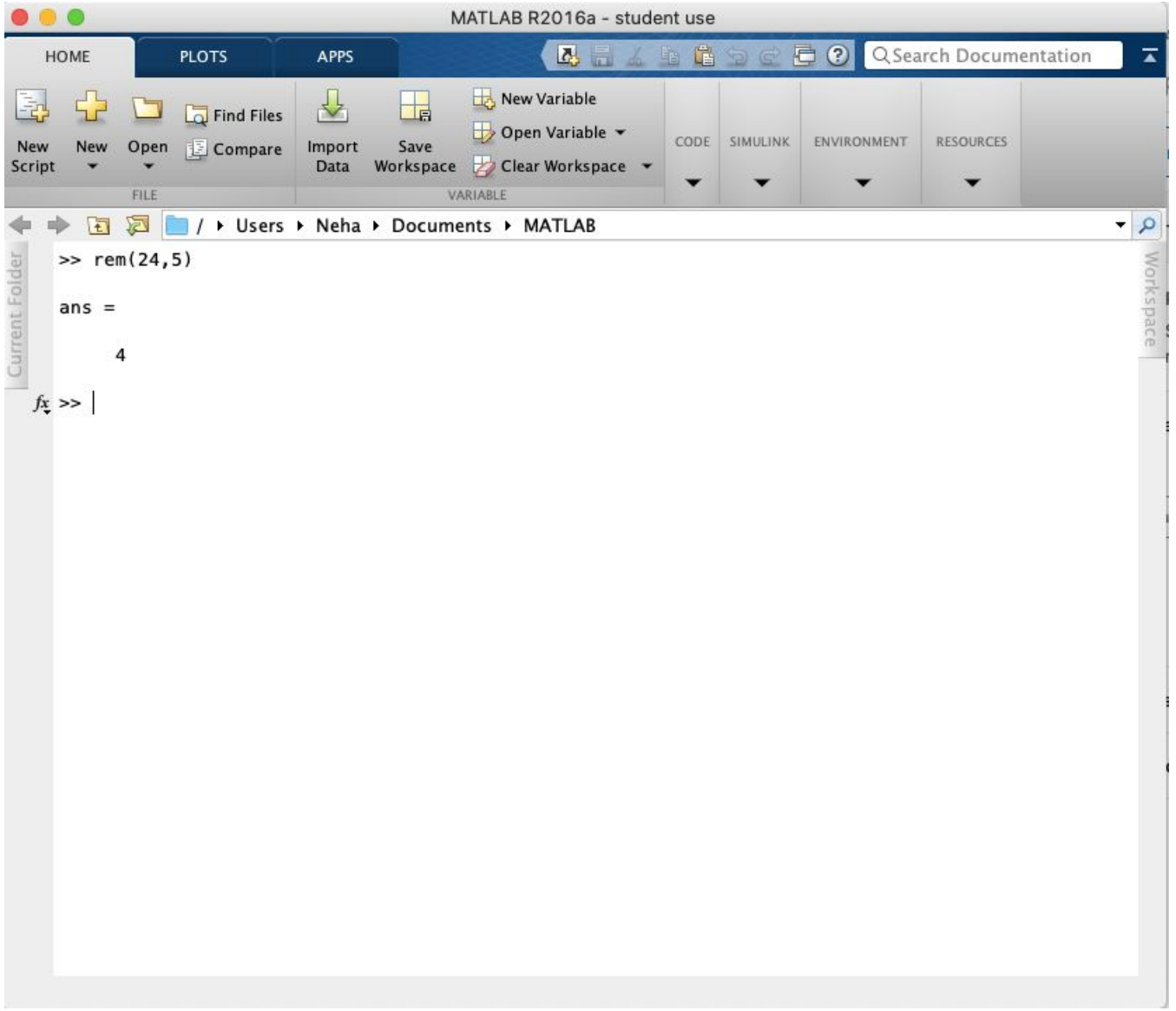
Finding Maximum and Minimum Value: max([x]) and min([x]) are used to find the maximum and minimum values of a set of numbers, where [x] is the set of numbers. Note you need to enter the list of numbers in these square brackets [ ] which are inside the round brackets. When I didn't use the [ ], it gave me an error. (A list of numbers in square brackets are vectors, but vectors are discussed in a future lessons):



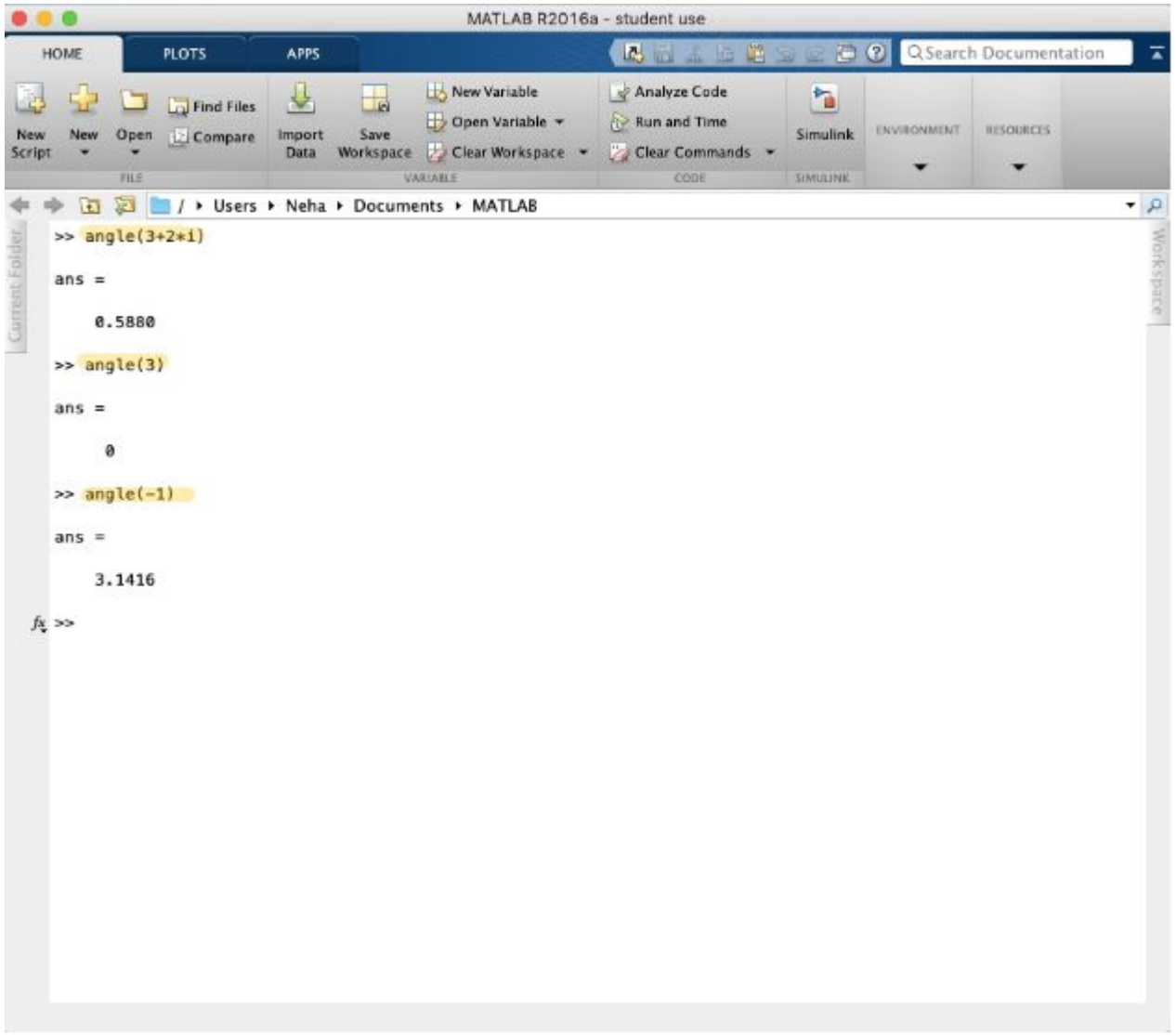
Rounding Functions – Ceiling, Floor and Round: ceil(x), floor(x) and round(x) will round numbers. ceil(x) rounds up, floor(x) rounds down, and round(x) rounds to the nearest integer, whether that's rounding up or down. See the example below, where I input 123.45 into all three functions and you can notice how each function rounds the same value differently:



Remainder After Divison: To find the remainder after dividing two values, use rem(x,y) if you are dividing x/y and finding the remainder. In this example I am dividing 24/5, so the remainder is 4:



Finding the Phase Angle: To find the phase angle of an equation, use angle(x), where x is in radians by default. The first example I have highlighted is one where I used a complex number as x, then I have an example using a positive number 3, then a negative number -1.



Finding the Complex Conjugate: To find the complex conjugate of a complex number (a complex number is one that has an imaginary part to it), use conj(x). It won't work with numbers that don't have 'i' in them, which is the part that makes a complex number imaginary!