LEARNING OBJECTIVES:
1. Call functions
2. Use zeros() function
3. Explore RGB image matrices in MATLAB
4. Work with subplots

REMINDER: Name your file Lab3_LastName. You must complete this Lab#3 assignment individually. You may ask only the course TA and Tutors for assistance. You must complete this assignment without looking at other student's code or copying solutions from any source.

Description:
This assignment consists of following a series of instructions and reporting on outcomes. Some questions may ask for the code to achieve a certain result while others will ask for the result of some code.

Lab Instructions:
Login and set up same environment as Lab#1 (cs7wXX ieng6 home directory, Notepad++, MATLAB). Refer to Lab#1.

PART ONE: INTRO TO FUNCTIONS (AND HOW DIFFER FROM SCRIPTS)
1. In this Lab, we will learn how functions work and how to use them in various ways (don’t worry; we will be learning how to write them soon too). Functions encapsulate a specific task in MATLAB – they can combine many instructions into a single line of code (like with scripts) but provide more flexibility in that the user is allowed to call them with certain input to get the desired output (NOTE: not all functions have input/output).

2. In the very first lab assignment, we used built-in MATLAB functions like sqrt() and gcd(). Let's review how we called them in the interpreter:
   a. sqrt() took in a number, and returned the squared root of that input as its output:

   Example:
   ```
   >> sqrt(25)
   ans =  
   5
   Example (if we want to store the result into a variable, like 'x'):
   >> x = sqrt(25)
   x =  
   5
   ```
**Example** (if we provide two parameters; causes an error)

```matlab
>> sqrt(25, 36)
Error using sqrt
Too many input arguments.
```

b. `gcd()` took in two numbers, and returned the greatest common divisor of these numbers:

**Example:**  
```matlab
>> gcd(25,10)
ans =
  5
```

**Example** (if we want to store the result into a variable, like 'a'):

```matlab
>> a = gcd(25,10)
a =
  5
```

**Example** (if we provide only one parameter; causes an error):

```matlab
>> gcd(25)
Error using gcd (line 27)
Not enough input arguments.
```

As you can see here (and already viewed in Lab#1), each function has specific requirements for its input and, if it has output, it can store the returning value(s) into a variable(s) in the workspace for future use.

**Question #1:** What is another built-in function have we used so far this quarter?

3. In MATLAB, we are also able to nest functions (use them in each other).

**Example:**

```matlab
>> sqrt(gcd(100,175))
ans =
  5
```

In the line of code above, `gcd()` was called first with 100 and 175 and the result (25) was used in `sqrt()`, leading to a final value of 5.

**Question #2:** Give an example of another set of nested functions that gives valid output (no errors).

4. In MATLAB, a function can also have a list of outputs. An example of this would be the `size()` function (we will be using this function a lot this quarter), which takes in a matrix, and returns two outputs – the height and the width of the matrix (multiple outputs depending on the matrix).
Example:

```matlab
>> matrix = [1,2,3;4,5,6;7,8,9;10,11,12]
```

```
matrix =

    1     2     3
    4     5     6
    7     8     9
   10    11    12
```

```matlab
>> size(matrix)
```

```
ans =

    4     3
```

If we wanted to store these values separately into two variables, we can define a list (with square brackets []) and place the two variable names accordingly.

Example:

```matlab
>> [rows, cols] = size(matrix)
```

```
rows =

    4
```

```
cols =

    3
```

You can use

```matlab
>> doc size
```

to read more about how the size function works.

**Question#3:** What happens if we do not use a list to store the outputs of the `size()` function? For example, if we do:

```matlab
>> output3 = size(matrix)
```

what would be assigned into the variable `output3`?

**PART TWO: PRACTICE WITH BUILT-IN FUNCTIONS**

1. Now that we know how to call these functions, we will explore other built-in ones that are unfamiliar to us. To figure out what functions we can use and how to use them, go to [http://www.mathworks.com/](http://www.mathworks.com/) or simply use google to get the functions that you need. This is a very important skill to learn, because someone will rarely tell you exactly what commands to type (except in this class). Instead, they will pose a problem or a question and ask you to solve it. You will practice this in the next couple questions. As always, feel free to ask a TA/tutor for help/clarification.

Example: How do you calculate 5 to the power of 3 and store this value into a variable called `a`?
Solution: \[ a = \text{power}(5,3) \]

**Question#4:** How do you find a list of all the prime numbers less than or equal to 20, and store the whole list as `primes_20`? (Hint: The list should contain the numbers 2, 3, 5, 7, 11, 13, 17, and 19.)

**Question#5:** How would you find the base 2 logarithm of 256, i.e., \( \log_2(256) \)? Store this value into `log_256`. (Hint: The answer should be 8.)

**Question#6:** How would you find the absolute value of -100, divide that number by 39, and then round this quotient to the nearest integer? Store this value as `q6_solution`. (Hint: The answer should be 3. If you use nested functions, this can be done with one line of code.)

**PART THREE: ZEROS() FUNCTION**

1. A function we will be using numerous times for images will be the `zeros()` function, which provides a matrix filled with zeros. Type in the interpreter:
   
   ```matlab
   >> doc zeros
   or
   >> help zeros
   or look it up on Mathworks.com for a more detailed explanation of how this function works. It basically creates a matrix filled completely with zeros and the size of this matrix will be determined by the input parameters used with it. (Note: If only one parameter was used, which we will call `n`, the resulting matrix will be of size \( n \times n \).)
   
   Example:
   ```
   
   ```matlab
   >> zeros(4,5)
   ans =
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
   ```

   Example: (Think three dimensional for this one, as in 3 sheets of 4x5)  
   ```matlab
   >> zeros(4,5,3)
   ans(:,:,1) =
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
   ans(:,:,2) =
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
    0   0   0   0   0
    ```
ans(:,:,3) =

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Example: (Using only 1 parameter)
>> zeros(4)

ans =

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

As you can see, (with the exception of a single input) the number of dimensions in the resulting matrix will be the number of inputs used.

2. As mentioned briefly before, since imshow() (the function we use to display images) just requires a three-dimensional, uint8 matrix as input, try this:
>> imshow(uint8(zeros(40,50,3)))

NOTE: a uint8 type is used because color values are represented in the range of 0 to 255, and the uint type can represent exactly this range. Other types store other ranges of numbers. For example a double type represents numbers between about \(-1.8\times10^{308}\) and \(+1.8\times10^{308}\). This range is much larger than what we need so uint0 is used instead.

Question#7: What did the above line of code show? Does it make sense? (Hint: Think about images are represented in MATLAB and the concept of RGB color.)

PART FOUR: PLOTS AND SUBPLOTS
1. We will now work with plots. Like imshow(), there is a function plot() that takes in two equal-sized vectors and displays the plot like below:
>> x = [0:6];
>> y = x;
>> plot(x,y);
2. Before closing this plot, if we call the title() function, we can title this plot:

```matlab
>> title('Positive slope');
```

![Positive slope graph](image)

3. Now, we can also use subplots to display multiple plots into one figure window.

   a. Subplots allow users to partition the figure into parts. For example, the command
   ```matlab
   >> subplot(3,2,1)
   ```
   Indicates that in a setting of a $3 \times 2$ grid, this particular subplot will be placed in the spot number 1 (see below)

   ![Subplot grid](image)

   So, after using the command subplot(3,2,1), any graph you plot, for example calling
   ```matlab
   >> plot(x,y)
   ```
   from the earlier example would place that plot in spot number 1 in the grid. If you then continued and typed:
   ```matlab
   >> subplot(3,2,2)
   >> plot(x,y)
   ```

   You would see that in the figure, another plot would be placed at position number 2 according to the $3 \times 2$ grid.
4. Make a script called `subplots.m`. You will be pasting the lines of code from this script into Question#8. Note that this is a 2×2 grid, rather than the 3×2 grid from the example.
   a. The first plot will be a negative slope, \( y = -x \).
   b. The second plot will be the sine function, \( y = \sin(x) \).
   c. The third plot will be the exponential, \( e^x \).
   d. The fourth will be a plot of your choice (something other than positive slope).

5. Once completed, the plot should look something like this:

   ![Plot Diagram]

   a. Save your figure as a jpg image: Lab3_subplot.jpg.
LAB #3 CHECKOFF CHECKLIST

To receive credit for this lab you need to:

- Write your name on the whiteboard when you are finished and ready to get checked off.
- Show your TA/Tutor your files are in your ieng6 cs7wXX home directory folder.
- Be prepared to show the TA/Tutor the subplots.m file and the Lab3_subplot.jpg image, and be able to explain what you did.
- Be prepared to show the TA/Tutor the Notepad++ document with all 8 questions answered.
- Be able to answer questions about functions, zeros() function, and subplot() function.

Do not leave until you have seen a TA/Tutor mark your name down in autograder.

It is your responsibility to make sure you get credit for each lab!

HOMEWORK #3:

- The Homework#3 assignment is due next week in YOUR lab section at the BEGINNING of Lab.
- You are free to go, or you are welcome to stay and work on Homework#3 (posted on the website) portion in the remainder of your enrolled lab section.