Project Requirements:

Comments in Code: You must include comments in every script file submitted that cites any resources outside of course materials used (e.g., hyperlinks for blogs, hyperlinks for MathWorks), acknowledges any additional help you received from anyone other than your own professor with the person's name (e.g., Professor X, Taylor the tutor, my roommate Jordan), and an explanation of how the material was used and/or the person helped with the code. If you are using lines of code substantially written by others to enhance your code, you must clearly comment these lines of code and their source; these may be used in your code, but they will NOT count for any points in the assessment of your final project (e.g., if your only programmer-defined function is one that someone else wrote, you will receive a 0 for "create at least one programmer-defined function").

Programming Techniques: All of the programming techniques in the table below must be demonstrated in your code for reasonable uses that make sense for your project. *(refer to final project rubric for more details)*

INPUTS:
User inputs/interface (e.g., input, dialog boxes, GUI)
File input (e.g., xlsread, dlmread, getcsv)
Random numbers (e.g., rand, randi)
DATA TYPES:
Numeric data and strings
Arrays (i.e., vectors, matrices, cell arrays)
DATA PROCESSING:
Rounding (i.e., round, ceil, floor)
Counting/calculating (e.g., running total, sum, mean, multiplication)
Organizing/analyzing (e.g., sorting, searching)
Array manipulation (i.e., referencing, slicing, augmenting, and diminution)
LOGIC:
Conditions (i.e., relational operators and Boolean operators)
Conditionals (i.e., if statement, switch/case)
Loops (i.e., while loop, for loop)
Nesting (loops and conditionals)
Error checking (all inputs appropriately error checked)
Programmer-defined functions (input parameters and return values)
String functions (e.g., sprintf, strcat, strfind, strcmp)
Other built-in function not discussed in class
OUTPUTS/DISPLAY:
Display relevant outputs to user/interface (e.g., fprintf, dialog boxes, GUI)
File output (e.g., xlsread, dlmread, putcsv)
Plotting (with appropriate data, formatting, and labeling)

Scoring Markers in Code: You must add score markers (%<SM:___>) as defined in the final project rubric for one occurrence of each programming technique. You will lose 1 point for each missing scoring marker. You will lose 3 points for each occurrence of mislabeled scoring markers (e.g., a for loop labeled as a while loop).

Overall Coding: You must also make your interface user-friendly, strive for efficiency in your code, and format your code neatly. Refer to the final project rubric for more details about grading for the final project submission.

List of Inappropriate Code/Techniques:

These guidelines refer to coding practices that you must NOT implement in your code. Use of any of these will result in 10 point reduction for each category unless explicitly stated elsewhere (example: use of GUIDE is an automatic 100% reduction and results in a 0, as stated below). For example, if there is a variable called i, a variable called ans, 4 uses of break, and one use of exit in your final project, you will lose 30 points on your final project.

Most of these are tools which can be useful (and not necessarily detrimental) when used in special situations. New programmers will generally use them instead to bypass good design practices, or the tools allow students to avoid using the techniques we are attempting to teach. As such, they are BANNED.

- GUIDE and App-Designer for code creation will result in a 0 on the final project. This includes all .fig files.

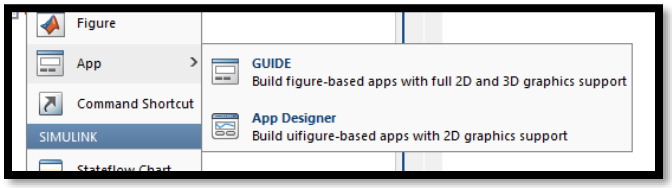
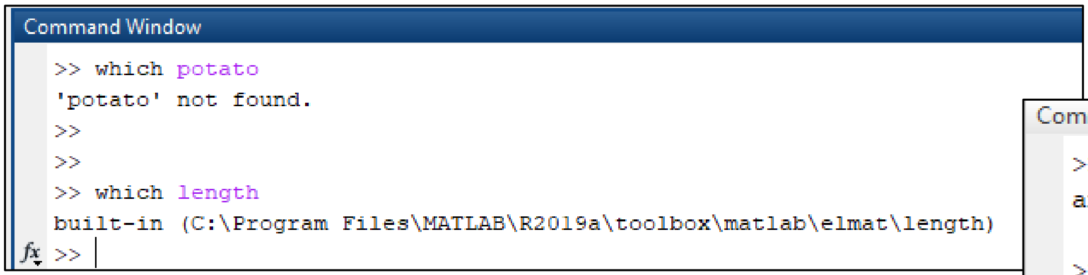


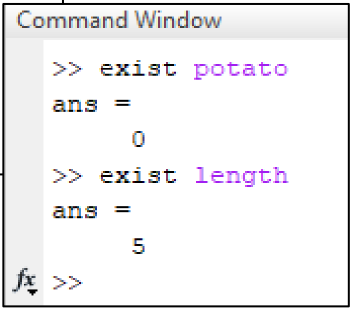
Figure
App
Command Shortcut
SIMULINK
Stateflow Chart
GUIDE
Build figure-based apps with full 2D and 3D graphics support
App Designer
Build uifigure-based apps with 2D graphics support

* If you are going down this path, you will get a **ZERO**. **DO NOT** use GUIDE or App Designer.

- `global`, `persistent`, and `static` variables are NOT allowed. (This includes the use of `evalin` and `assignin`.) `setappdata` and `getappdata` are also NOT allowed. For levels of coding in this course, this is a lazy way to avoid transferring variables/data. *For other methods, consider using struct (structures) for GUIs instead. Refer to https://www.mathworks.com/help/matlab/matlab_prog/create-a-structure-array.html*
- Calling a program. *(This is not referring to calling a programmer-defined function or other built-in function.)*. This means typing the name of the file to execute it, but this file is not a function file. To fix this, turn the file into a function without return-values or parameters then call the function!
- Defining a programmer-defined function inside another programmer-defined function.
- `break`, `continue`, `return`, and `run` functions are NOT allowed. For levels of coding in this course, this is a lazy way to avoid writing a well-though-out algorithm.
- `quit`( ), `exit`( ), and `error`( ) are NOT allowed. These will terminate the MATLAB program.
- You many not create a variable or function with a name that is already used in MATLAB (e.g., i, j, ans, edit, help*). To check and see if a name exists in MATLAB, use either method below:*
    - *the* ***`which`*** *function will tell you 'not found' or a path to an existing file.*


Command Window

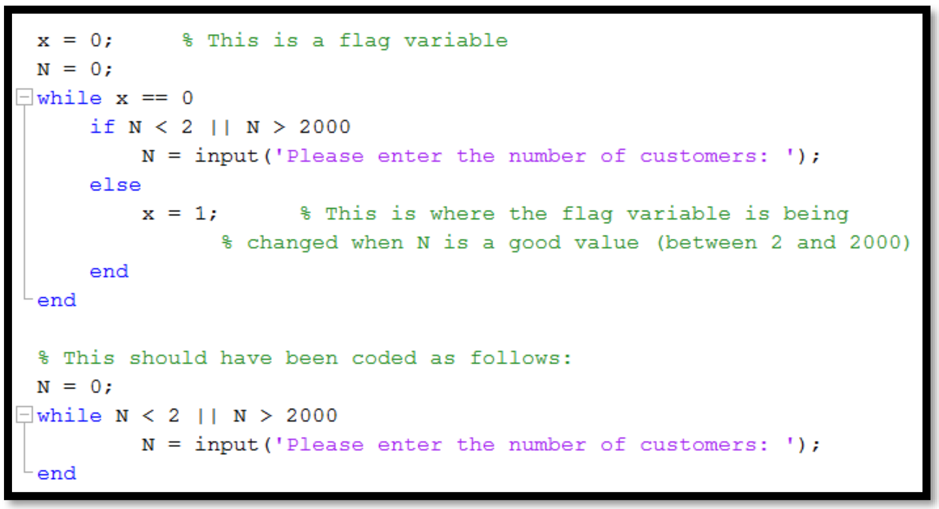
```
>> which potato
'potato' not found.
>>
>>
>> which length
built-in (C:\Program Files\MATLAB\R2019a\toolbox\matlab\elmat\length)
>>
```


Command Window

```
>> exist potato
ans =
     0
>> exist length
ans =
     5
>>
```

    - *the* ***`exists`*** *function will tell you 0 if* name does not exist or cannot be found for other reasons", or a number if it exists somewhere

- Inappropriate flag variables. *Flag variables are variables used to track events. In very complex situations they can be useful – in most of our class, they are not needed. An example of an inappropriate flag variable is below, as well as how to remove it:*



```
x = 0;      % This is a flag variable
N = 0;
while x == 0
    if N < 2 || N > 2000
        N = input('Please enter the number of customers: ');
    else
        x = 1;       % This is where the flag variable is being
              % changed when N is a good value (between 2 and 2000)
    end
end

% This should have been coded as follows:
N = 0;
while N < 2 || N > 2000
        N = input('Please enter the number of customers: ');
end
```

- The use of `disp`, `display`, `table`, `summary`, `celldisp`, and other print functions are NOT allowed. You can use `fprintf()` to format any data that you want to display to your user in the command window. *(Avoiding these by omitting semi-colons to display information will result in this same deduction; this deduction applies to any unsuppressed lines of code in your final project.)*
- Evaluating string inputs as numeric inputs is a NO. *An example of this is shown here:*

```
%define Yes and No variables
Yes=1;
No=2;

%ask User to do something
%BAD: this is missing 's' at the end so input() is really scanning for a
%NUMERICAL value. When use enters Yes, MATLAB is forced to search for a
%variable Yes to associate it to the value 1.
response = input('Enter Yes to play, No to quit: ');
```

```
%GOOD: do this instead!
response = input('Enter Yes to play, No to quit: ','s');
%or simply:
response = input('Press 1 to play, 2 to Quit: ');
```

- The use of `.mat` files are forbidden.
- The use of use of `"strings"` instead of `'character vectors'` (strings taught in course). Always use single quotes/apostrophes.