MATLAB Tutorial

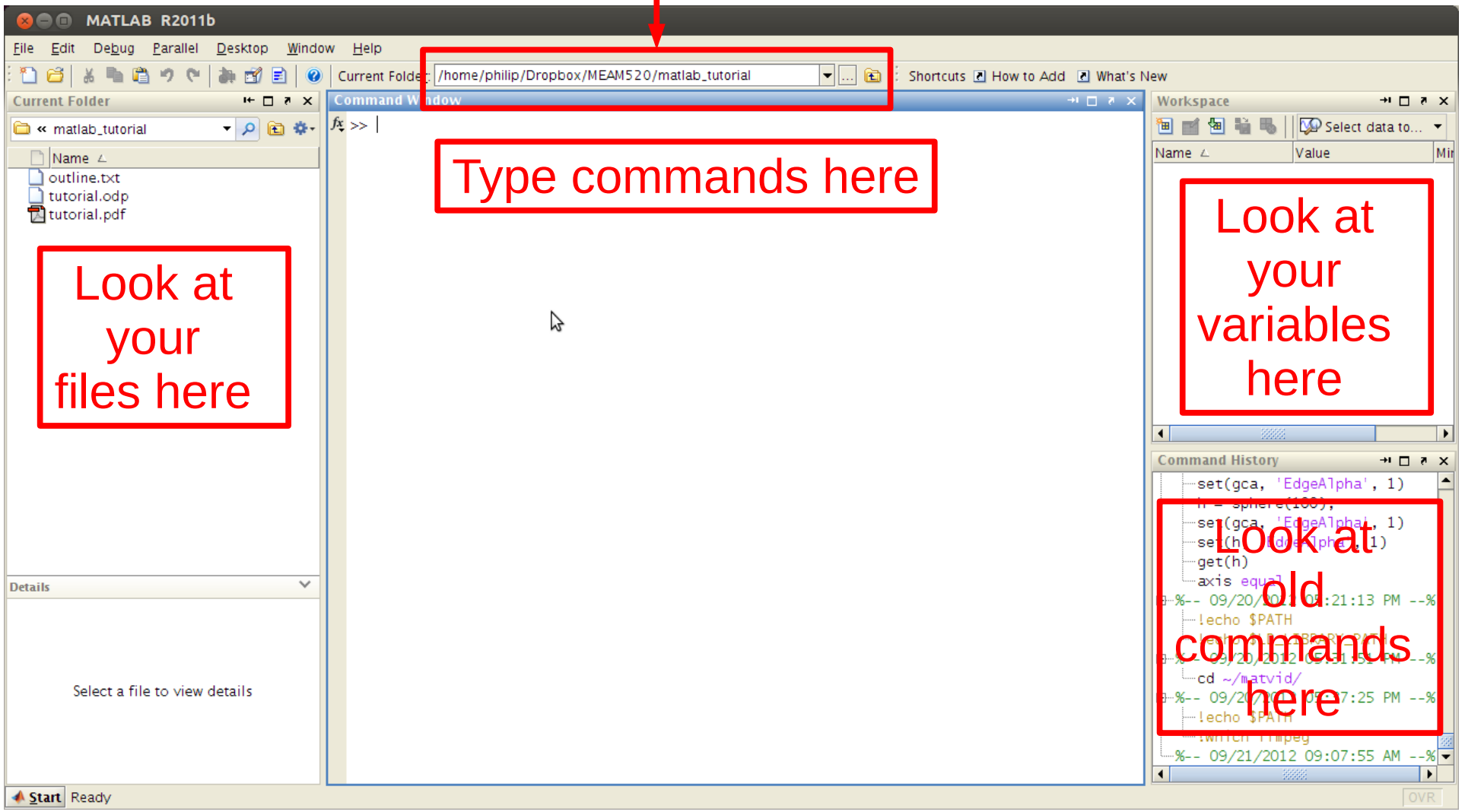
MEAM 520

Philip Dames
Denise Wong

Sept. 21, 2012

Interface

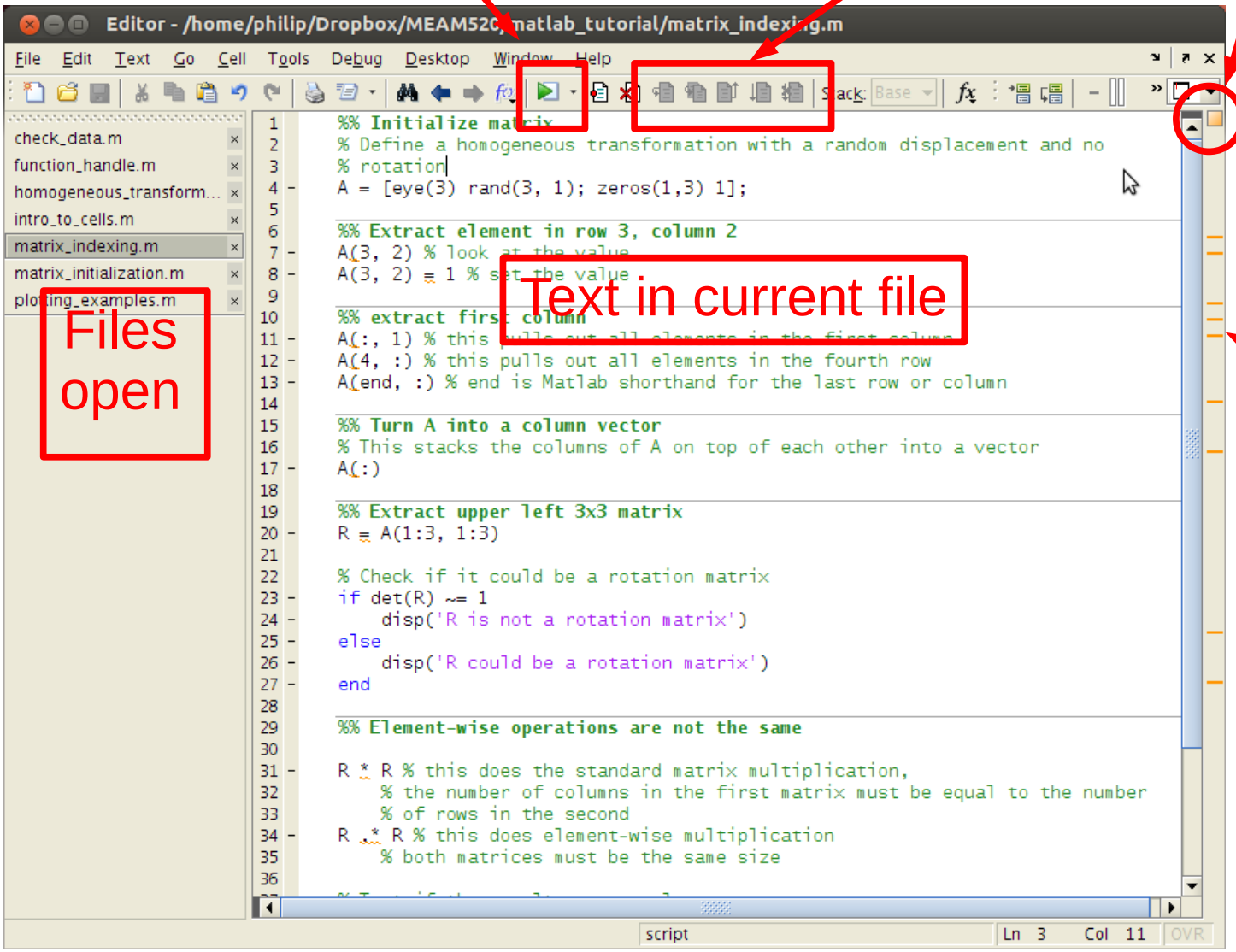
Change folders here



Editor

Run current file (same as F5)

Step through cells



Green
Everything is good
Orange
Code will run, but you can do things better
Red
Code will not run

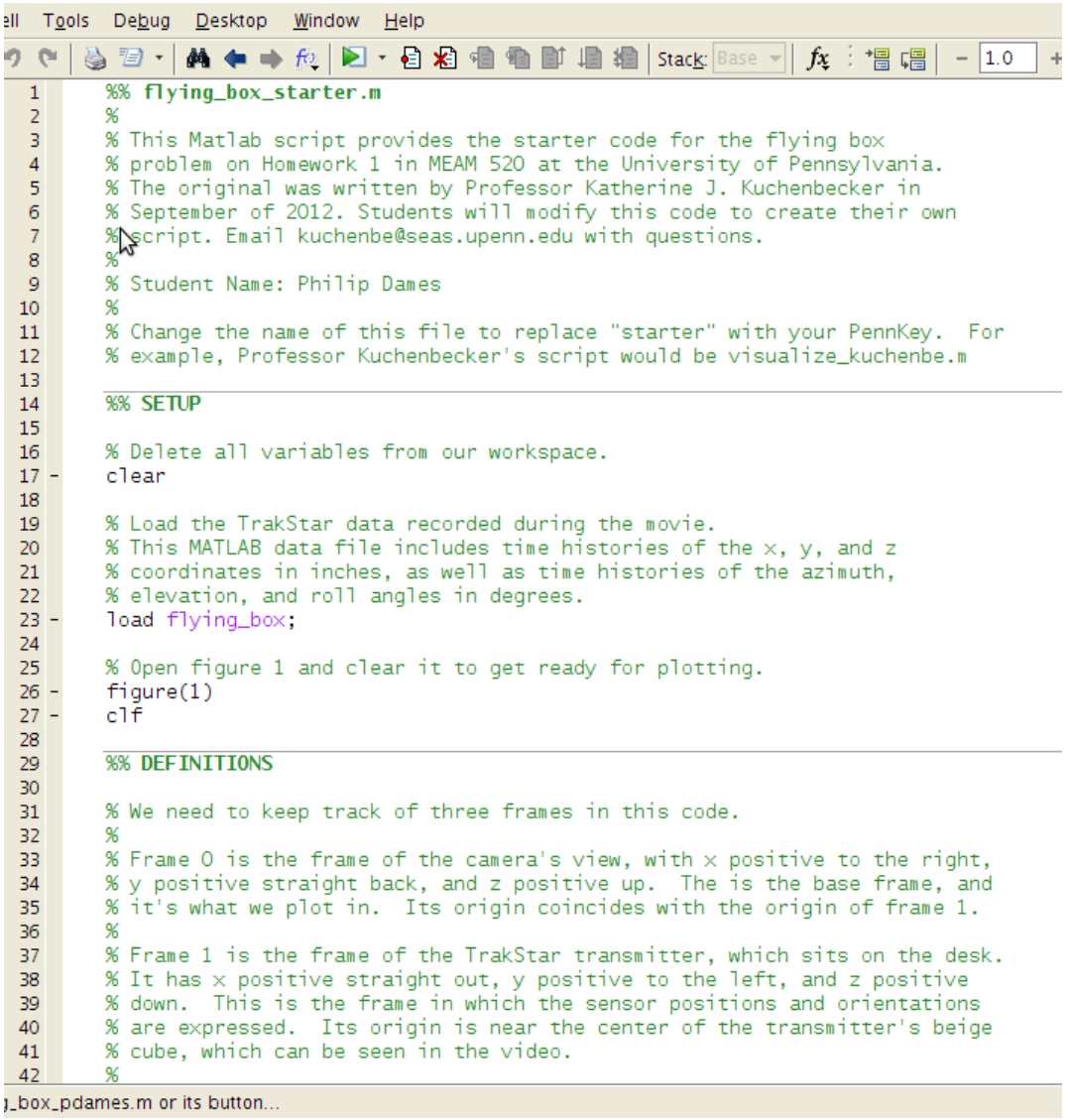
Matlab hightlights lines that you should look at

Writing Code

- Use descriptive variable/function names
- Write comments
- Go slowly
- Test each little piece, not the whole system
- Write modular code
  - If you copy/paste something many times, turn it into a function

Programming with Cells

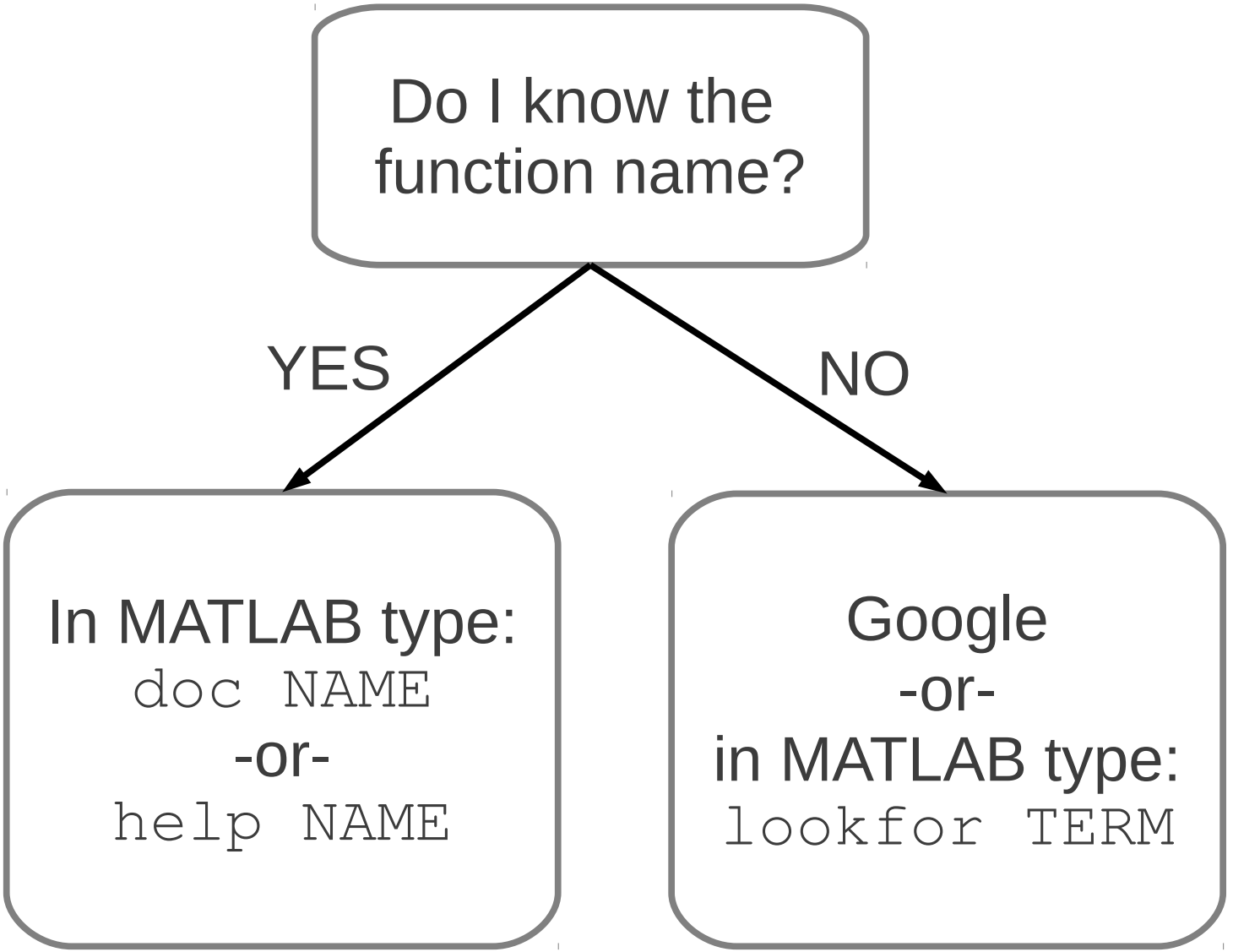
- Begin cell with %%
- Allows you to run small pieces of code
- Useful for:
  - Testing/debugging
  - Organization
- Used in the homework 1 files



Helpful Hotkeys

- Ctrl+r – comment blocks of code
- Ctrl+t – uncomment blocks of code
- F5 – run script
- Ctrl+Enter – run cell

Help: From MATLAB



Help: From Instructors

- Don't say:

  “My code doesn't work”

- Do say:

  “I am trying to get X and I did Y
  but got Z instead.”

- The more detail you give us, the more quickly we can help you out

Look at Your Data

- `disp` – displays something in Command Window
- `keyboard` – pause execution of code and enter Command Window in calling workspace
- `plot` – visualize your data
- `;` – leave off the semicolon to display output

Check Your Data

- `if` – checks if something is true
- `assert` – checks if something is true, if not then kill the program
- `warning` – displays a custom warning in the Command Window, but keep running program
- `error` – displays a custom error in the Command Window, kills the program

Matrix Initialization

- `zeros(n,m)` – creates an `n x m` matrix of 0's
- `ones(n,m)` – creates an `n x m` matrix of 1's
- `eye(n)` – creates an `n x n` identity matrix
- `1:n` – creates the row vector `[1, 2, …, n]`
- `[A B],[A; B]` – create matrix from other matrices
- `[]` – create empty matrix
- `rand(n,m)` – create an `n x m` random matrix
- `size(A)` – returns size of matrix A

Matrix Indexing

- `A(r,c)` – takes element in row `r` and column `c`
- `A(:,c)` – takes all of column `c`
- `A(1:3, 4)` – takes elements in rows 1-3 and column 4
- `A(B>1)` – logical indexing, takes all elements of A where B>1 is true
- `A(:)` – turns matrix A into a column vector by stacking columns

Matrix Manipulation

- `A'` – (conjugate) transpose of matrix A
- `A.'` – transpose of matrix A
- `.*  ./  .^` – element-wise operations on a matrix
- `det(A)` – determinant of A
- `trace(A)` – trace of A

Trigonometric Functions

- MATLAB has built-in functions for both radians and degrees
  - `cos(theta)` – cosine of theta, in radians
  - `cosd(theta)` – cosine of theta, in degrees
  - Same for `sin, tan, sec,` etc
- Inverse functions begin with “a”
  - `acos(x)` – arccos of x, in radians
  - `asind(x)` – arcsin of x, in degrees

Functions

- Begin with
```
function [output1, output2, …] =
   FUNCTION_NAME(input1, input2, ...)
```
- Variables defined in the function exist only in the function (unless output)
- Cannot access variables defined outside the function (unless input)
- Save in separate file `FUNCTION_NAME.m`

Workspaces

- A workspace is something used by Matlab to store variables
  - Functions create their own workspaces that are deleted after the function finishes running
  - The command windows deals with the Base workspace (unless you use `keyboard`)
- `clc` – clear all commands from the Command Window
- `clear all` – deletes all variables from the workspace
  - `clear A` – deletes only the variable A from the workspace
- `close all` – close all open figures

Function Handle

- Like a mini-function
- Examples

```
f = @(x) x^3;
c = @(theta) cos(theta);
c(pi)
```

- Useful for simple things you will use repeatedly
- Not as powerful as functions

Basic Plotting

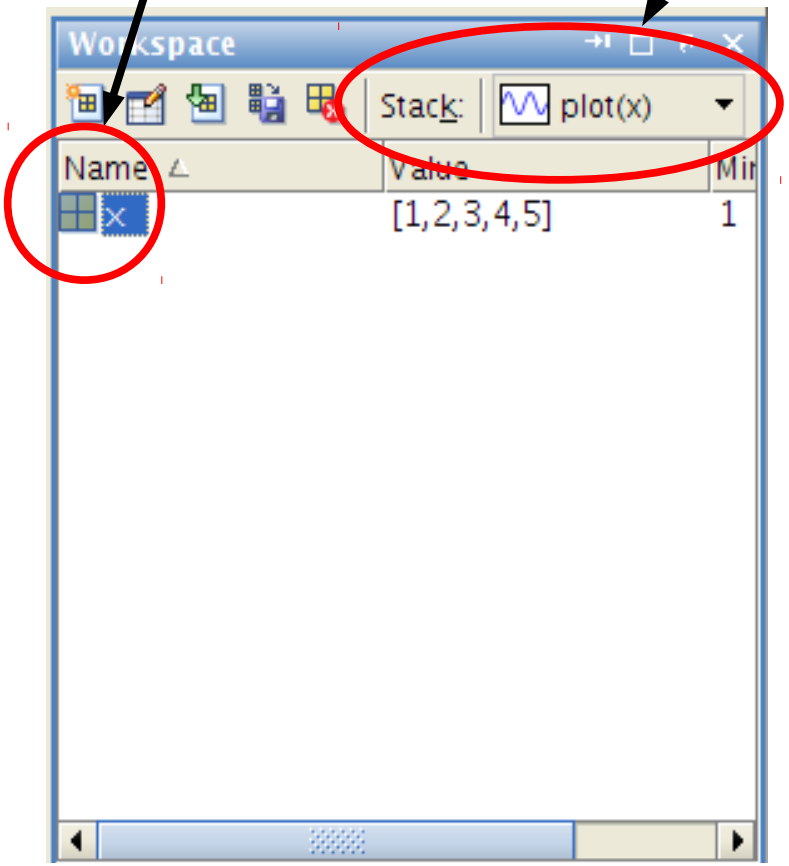
- `plot(x,y,'LineSpec')` – plots the data in `x` and `y`, using the optional argument `'linespec'` to change the color and style of the plot
  - See `doc linespec` for information on the available colors and styles
  - Example, `'r--'` is a dashed red line
- `hold on` – holds the data on the plot until `hold off` is called, this lets you plot multiple things on the same axis

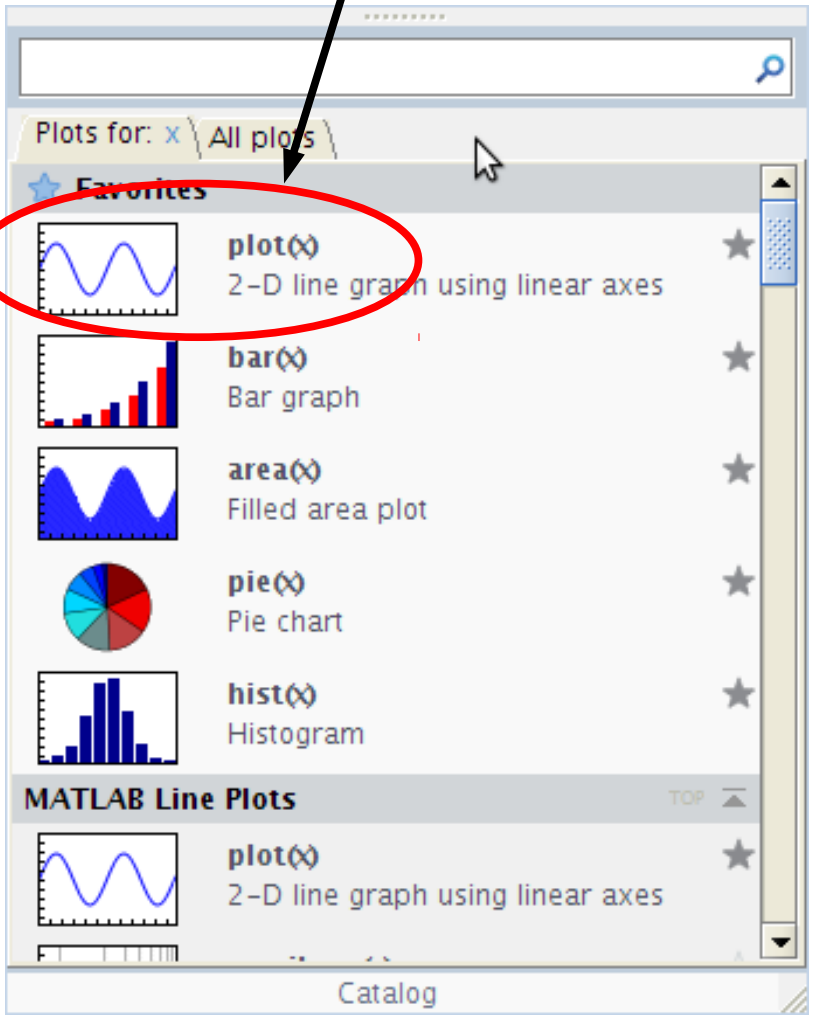
Quick Plotting

1) Select variable

2) Click on this

3) Select plot type





Plot Annotations

- `xlabel('text')` – labels the x-axis
- `title('text')` – displays a title above the plot area
- `legend('line1', 'line2', ...)` – displays a legend of the different line in the plot, useful when plotting multiple things
- `axis([xmin xmax ymin ymax])` – sets the x and y limits on the axis
- `figure` – creates a new figure window

3D Plotting

- `plot3(x,y,z,'LineSpec')` – very similar to plot, but does 3d
- `patch(X,Y,Z,C)` – creates filled polygonal patch with vertices specified by x, y, z and color c

Figure Handles

- `h=plot(x,y)` – h is a figure handle
  most plotting functions can return one
- `get(h)` – returns a list of figure properties
- `set(h,'PropertyType','Value')` –
  sets property PropertyType to Value
- `gca` – returns handle to current axis
- `gcf` – returns handle to current figure
- Much faster to use handles for animations since MATLAB does not redraw everything
- Used in the homework 1 files

Animations

- Use figure handles to update plots for animations
- `pause(t)` – pause for t seconds, this gives Matlab time to update the plot
  - Also useful for slowing down animations
  - `pause` – without a time input Matlab waits until a key is pressed
- `drawnow` – like pause, but Matlab will only wait for as long as it needs to plot