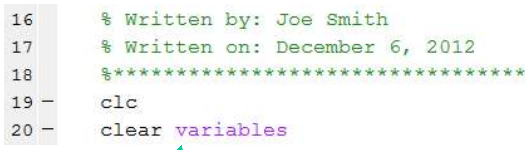
Day 7

- (Concept Question)
- A good start for all programs
- Two kinds of programming errors
- Debugger tools
  - Breakpoints
  - Examining variables
  - Command window
  - Step
  - Add/Remove, Continue
  - Quit debugging
- (Demo)
- (Exercises)

A good start for all programs

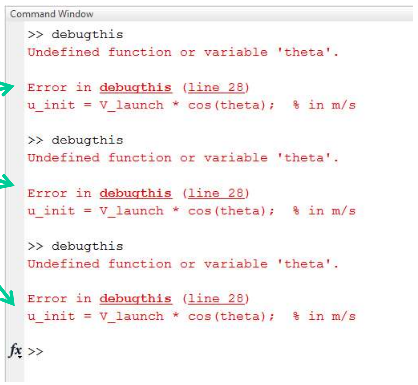
Be sure that your program always starts with `clc` and `clear variables`



```
% Written by: Joe Smith
% Written on: December 6, 2012
%***********************************
clc
clear variables
```

`clc` will clear your screen so that you can more easily see which error message go with the latest run of your program.

`clear variables` will clear all the variables in your workspace—without this you may get confusing results because of variables left over from a previous run of your code.



```
>> debugthis
Undefined function or variable 'theta'.

Error in debugthis (line 28)
u_init = V_launch * cos(theta);  % in m/s

>> debugthis
Undefined function or variable 'theta'.

Error in debugthis (line 28)
u_init = V_launch * cos(theta);  % in m/s

>> debugthis
Undefined function or variable 'theta'.

Error in debugthis (line 28)
u_init = V_launch * cos(theta);  % in m/s

fx >>
```

Two kinds of programming errors

There are two kinds of programming errors:

- Syntax errors
- Logical errors

Syntax errors are the ones that make RED in the command window

- You must fix all of the syntax errors exactly the way we have done so far

Two kinds of programming errors

Typical Logical errors

- Formulas typed incorrectly
- For loops with bad limits or increments
- Counters that don’t increment correctly

Matlab does not find logical errors– it doesn’t know what logic you want

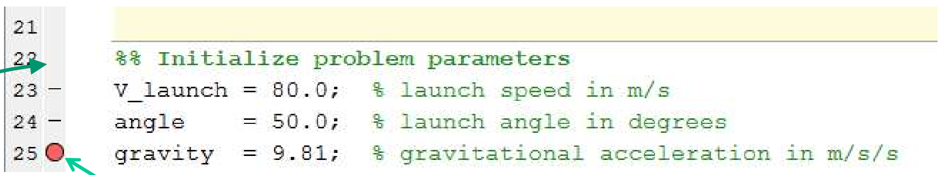
Two kinds of programming errors

Matlab has a “Debugger” to help *you* find logical errors in your code.

- You must fix all syntax errors before you can get to the Debugger.
- No magic: the Debugger simply lets you go one step at a time through your code.
- Then you must figure out which formulas are wrong.

Debugger tools: Breakpoints

Start the Debugger by setting a breakpoint on an executable statement near the start of your script.



```
21
22     %% Initialize problem parameters
23 -   V_launch = 80.0;  % launch speed in m/s
24 -   angle    = 50.0;  % launch angle in degrees
25 ●   gravity  = 9.81;  % gravitational acceleration in m/s/s
```

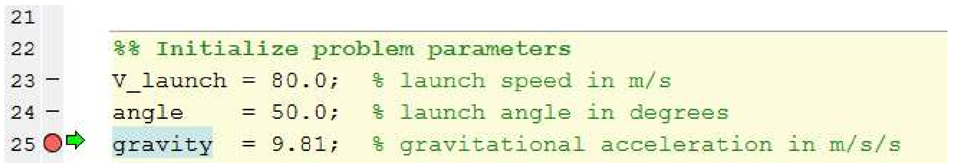
Set a breakpoint by clicking on the small dash by the line number. You will get a red circle—like a “stop sign”.

If your circle is gray, you probably didn’t save your file since your latest changes. Just save your file, and it will turn red.

Comments (%) don’t execute. You can’t put breakpoints there.

Debugger tools: Breakpoints

When you click Run now, you will see a green arrow by your breakpoint—MATLAB has *paused execution* of your program at your breakpoint. It is now in "debug mode."

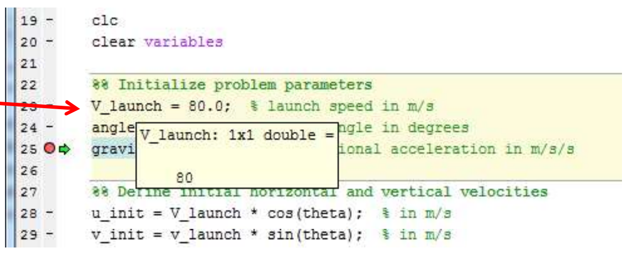


```
%% Initialize problem parameters
V_launch = 80.0;  % launch speed in m/s
angle    = 50.0;  % launch angle in degrees
gravity  = 9.81;  % gravitational acceleration in m/s/s
```

Debugger tools: Examining variables

Pausing execution lets you examine all variables that are defined so far. Two ways:

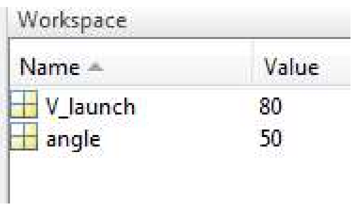
- Hover your mouse over variable names



```
clc
clear variables

%% Initialize problem parameters
V_launch = 80.0;  % launch speed in m/s
angle                          ngle in degrees
gravi                          ional acceleration in m/s/s

%% Define initial horizontal and vertical velocities
u_init = V_launch * cos(theta);  % in m/s
v_init = v_launch * sin(theta);  % in m/s
```

```
V_launch: 1x1 double =

    80
```

- Look in the Workspace



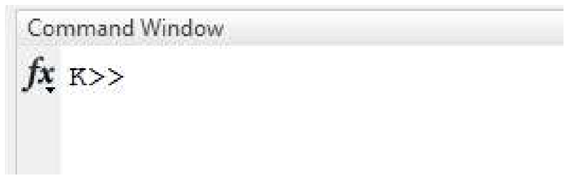
Workspace

Name	Value
V_launch	80
angle	50

NOTE: You will only see values for variables that have been assigned in program lines *before* your breakpoint.

Debugger tools: Command window

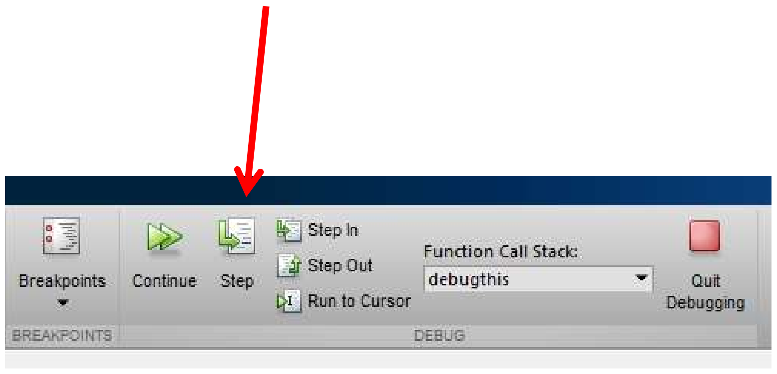
When your program has paused, you can also test variable values and run commands from the command window—this debug mode has a special prompt that looks like this:



```
K>>
```

Debugger tools: Step

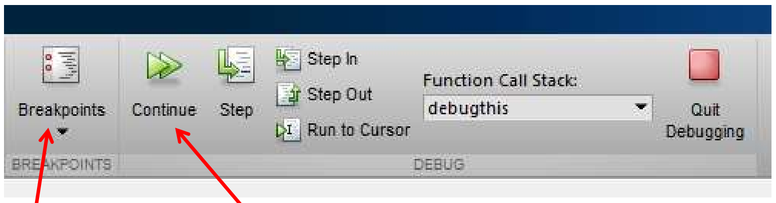
“Step” lets you execute one line of your program at a time.



When you “step” like this, MATLAB will execute one line and pause again.

Debugger tools: Add/Remove, Continue

Other tools allow you to add/remove breakpoints or continue running your program.

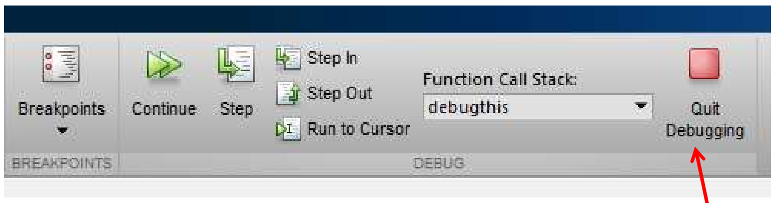


If you click here, your program will run to the end or until it hits another breakpoint.

This Breakpoints button allows you to add more breakpoints, clear breakpoints, or clear ALL breakpoints.

Debugger tools: Quit debugging

If you've made changes to your code while debugging, it is best to quit the debugging mode and click Run to start your program over from the beginning.



Clicking here stops execution of your program.

When you're done debugging, you should use the **'Clear All'** under the 'Breakpoints' menu to get rid of all the stop signs. Otherwise your program will continue to go into debug mode.