Learning Objectives:

- Review variable types of MATLAB *classes*
- Learn the structure of a script
- Learn how to write your own functions
- Learn how to use global variables
- Learn the Linear Regression algorithm

MATLAB Scripts

- The standard coding environment
- More useful for long codes
- Cannot embed texts and pictures so comments are mandatory
- Output appears in command window
- Variables are still stored in workspace
- Almost identical to live scripts but more practical for longer programs

MATLAB Functions

- You have already been using built-in functions BUT you can write your own too
- User-defined functions are useful for bits of code:
  - you plan to use over and over in the program
  - code you plan to use in multiple programs
  - Code that is boring and interrupts the conceptual flow of your program (like code that makes plots or imports data)
- Can be saved in a separate file or embedded in a script
- If functions are embedded in a script, they must appear at the end of the script
- Embedded functions cannot be accessed by another program

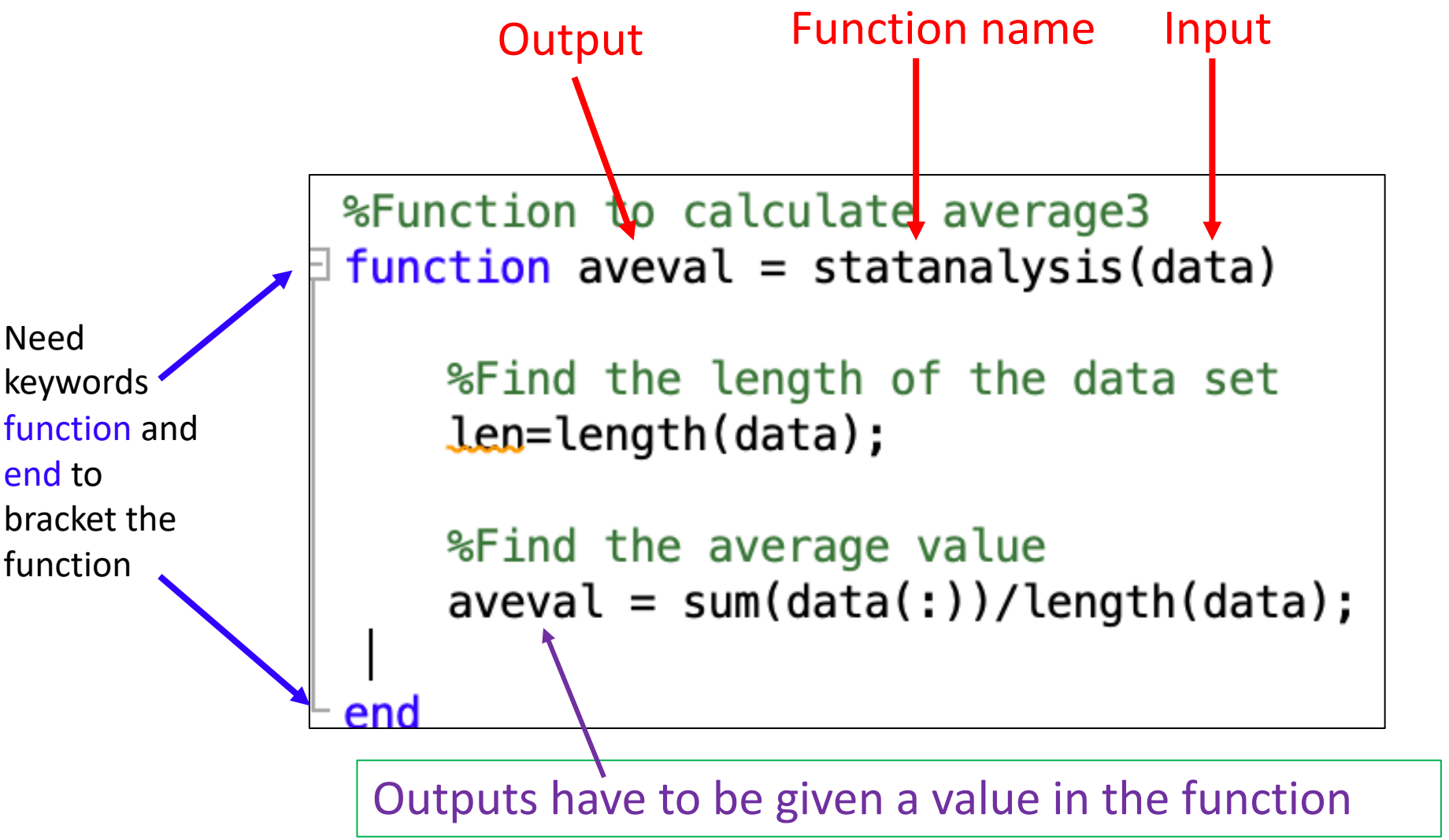
An example of a program that calculates the mean of a data set

```
%Generate some fake data
sample_data=rand(10,1);

%Find the number of data points
len=length(sample_data);

%calculate the average
average=sum(sample_data(:))/len;

fprintf('The average is %5.2f.\n',average);
```

User-defined function with one input and one output

```
%Statistical Analysis Program

%Generate some sample data
sample_data = rand(10,1);

%Call the function in the program
ave = statanalysis(sample_data);

%output the average
fprintf('The average is %5.2f .\n',ave);


%Function to calculate average3
function aveval = statanalysis(data)

    %Find the length of the data set
    len=length(data);

    %Find the average value
    aveval = sum(data(:))/length(data);

end
```

Defining a user-defined function with one input and one output

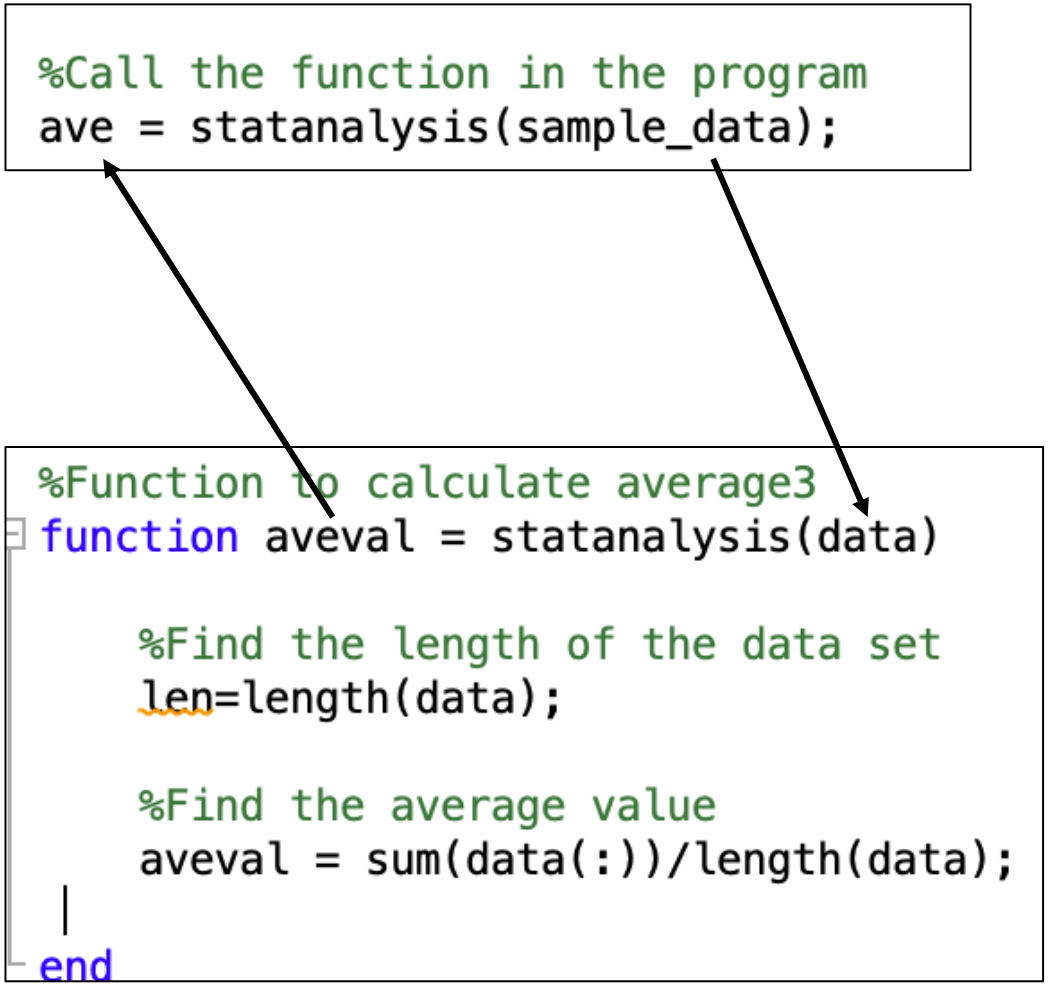


Calling a user-defined function

- Call it just like a built-in function
- Must use the correct name

```
%Call the function in the program
ave = statanalysis(sample_data);
```

Mapping variables defined in the function to the main program

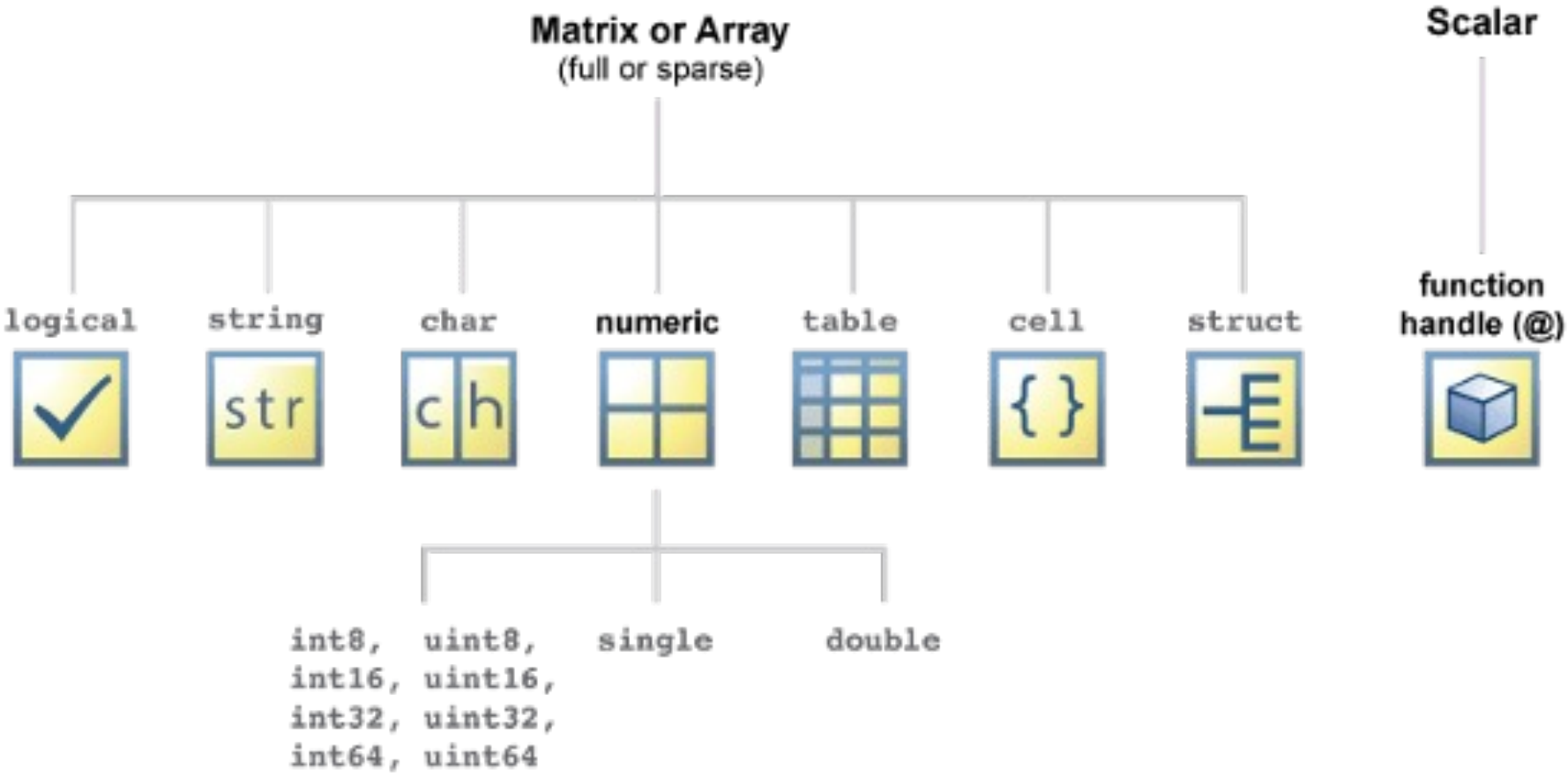


```
%Call the function in the program
ave = statanalysis(sample_data);
```

```
%Function to calculate average3
function aveval = statanalysis(data)

    %Find the length of the data set
    len=length(data);

    %Find the average value
    aveval = sum(data(:))/length(data);

end
```

- Must have same number of inputs and outputs
- Inputs and outputs must be the same type or class of variable (e.g., can't map a string to a number)
- Inputs and outputs **DO NOT** need to have the same variable names

MATLAB Variable Types or Classes



Placing the functions in the main program

```
%Statistical Analysis Program

%Generate some sample data
sample_data = rand(10,1);

%Call the function in the program
ave = statanalysis(sample_data);

%output the average
fprintf('The average is %5.2f .\n',ave);


%Function to calculate average3
function aveval = statanalysis(data)

    %Find the length of the data set
    len=length(data);

    %Find the average value
    aveval = sum(data(:))/length(data);

end
```

Declare function(s) *after* the main program or the code where you will call the functions.

Functions declared within a script cannot be used in the command line or in other scripts.

Typed in Command Window:

```
>> statanalysis(sample_data)
Unrecognized function or variable 'statanalysis'.
```

User-defined function with one input and two output

```
%Statistical Analysis Program

%Generate some sample data
sample_data = rand(10,1);

%Call the function in the program
[ave,standard_dev] = statanalysis(sample_data);

%output the average
fprintf('The average is %5.2f and the standard devation is %5.2f.\n',ave,standard_dev);


%Function to calculate average3
function [aveval,stdev] = statanalysis(data)

    %Find the length of the data set
    len=length(data);

    %Find the average value
    aveval = sum(data(:))/length(data);

    %Find the standard deviation
    stdev = sqrt(sum((data(:)-aveval).^2/len));

end
```

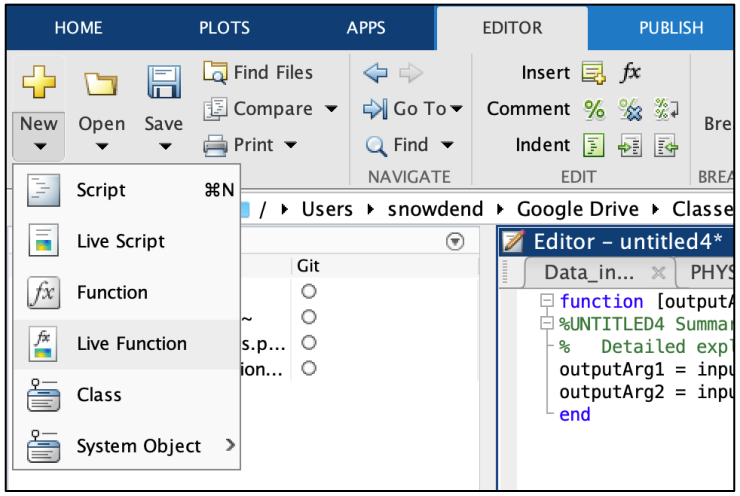
- Use square brackets to define multiple outputs
- Give each output a value in the function

User-defined function with multiple inputs and outputs

- Similarly you can have multiple inputs
- Note variables are mapped in order!

```
%Find Maximum Percentage Error

%Generate some sample data
sample_data1 = rand(10,1);
sample_data2 = rand(10,1);

%Call the function in the program
[PercError,maxPercError,imaxPercError] = percentError(sample_data1,sample_data2)


%Function to calculate average
function [pError,max_pError,imaxpError] = percentError(data1,data2)

    %Find percent error between all points
    pError=abs((data1-data2)./data1);

    %Find max percent error and index of max percent error
    [max_pError,imaxpError]=max(pError);

end
```

User-defined function that can be called outside of a script



You can have a .m file that only contains a function.

```
function [outputArg1,outputArg2] = untitled4(inputArg1,inputArg2)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
outputArg1 = inputArg1;
outputArg2 = inputArg2;
end
```

User-defined function that can be called outside of a script

Editor – /Users/snowdend/Google Drive/Classe

statanalysis.m

```
function aveval = statanalysis(data)

    %Find the length of the data set
    len=length(data);

    %Find the average value
    aveval = sum(data(:))/length(data);

end
```

Current Folder

Name	Git
statanalysis.m	
fuction_examples.m	
Scripts and Functions.p...	
~$Scripts and Function...	

Command Window

```
>> sample_data=rand(10,1);
>> statanalysis(sample_data)

ans =

    0.5280

fx >>
```

- The .m file defining the function should have the same name as the function.
- This function can be called in the **Command Line and in other m-scripts** as long as the .m file defining the function is in the **Current Folder**
- This is handy for user-defined functions you will use often in multiple programs

Local and Global Variables

You **declare** a variable when you assign it a value.

A **local variable** declared in the main program of a script cannot be accessed inside of function automatically. It has to be passed in as an input.

areaofcylinder.m

```
pie=3.14;

radius = 1;
height = 2;

A = surfAreaCylinder(radius,height);


function area = surfAreaCylinder(r,h)
    area = 2*pie*r*h;
end
```

Command Window

```
>> clear all
>> areaofcylinder
Error using pie (line 65)
Not enough input arguments.

Error in areaofcylinder>surfAreaCylinder (line 11)
    area = 2*pie*r*h;

Error in areaofcylinder (line 7)
A = surfAreaCylinder(radius,height);
```

Local and Global Variables

A **local variable** declared in a function cannot be accessed in the main program or in other scrips, unless it is passed out as output.

areaofcylinder.m

```
clear all;

radius = 1;
height = 2;

A = surfAreaCylinder(radius,height);

pie;

function area = surfAreaCylinder(r,h)

    pie=3.14;
    area = 2*pie*r*h;
end
```

Command Window

```
>> areaofcylinder
Error using pie (line 65)
Not enough input arguments.

Error in areaofcylinder (line 8)
pie;
```

Local and Global Variables

You **declare** a variable when you assign it a value.

A **global variable** is one that is "declared" outside of the but can be accessed or is assigned the same value inside the functions without being passed in as input.

You have to tell MATLAB what variables you want to be global before you define their value.

areaofcylinder.m

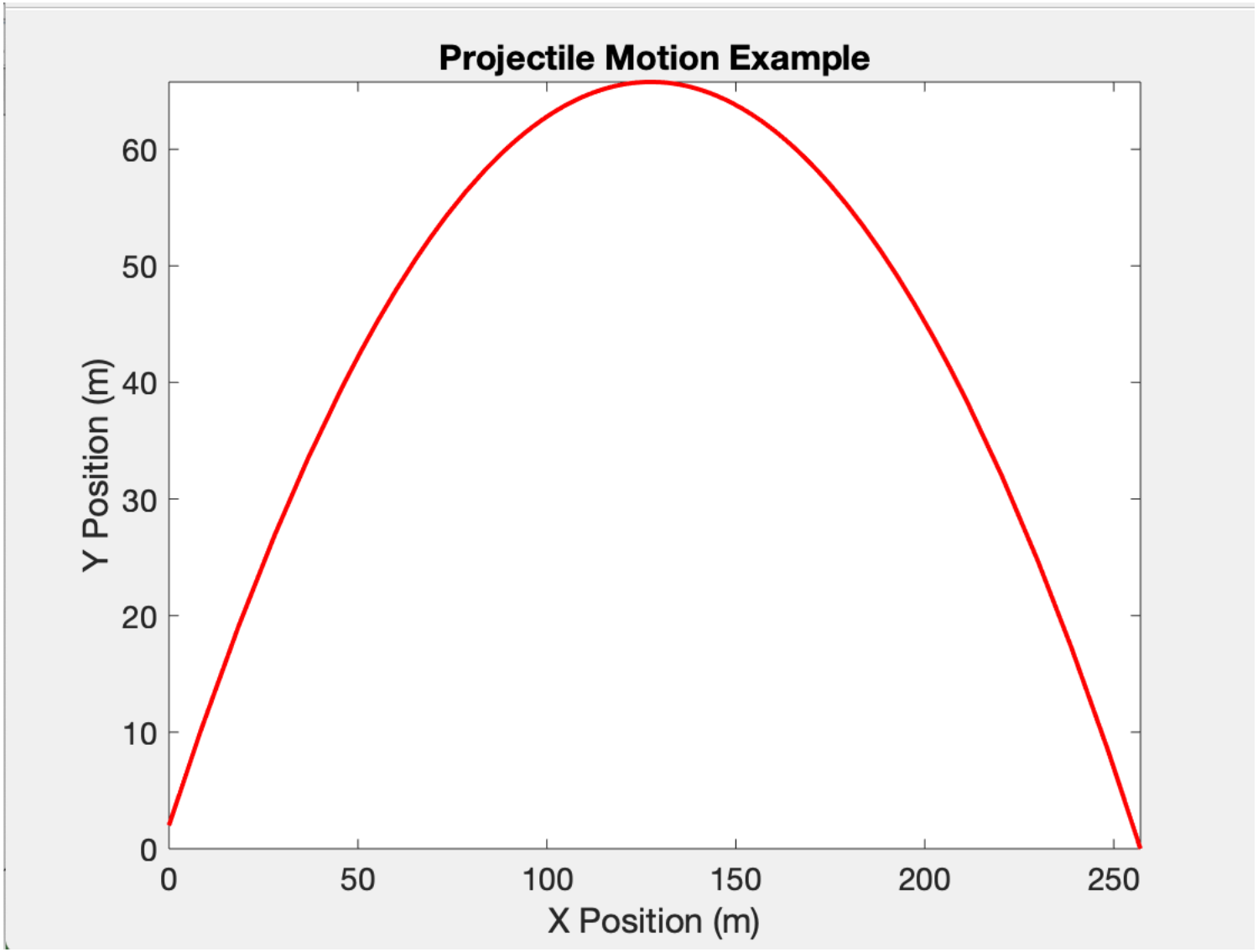
```
clear all;

global pie;
pie=3.14;

radius = 1;
height = 2;

A = surfAreaCylinder(radius,height);

function area = surfAreaCylinder(r,h)

    global pie;

    area = 2*pie*r*h;
end
```

Projectile Motion Example

Notice how I declare g inside the main program and use it in the functions.

```
%Define the acceleration of gravity
global g;
g=9.8;  %acceleration of gravity, m/s^2

%Ask the user for their input
angle_launch=input('What is the launch angle in degrees? ');
initial_v=input('What is the initial velocity in m/s? ' );
initial_h=input('What is the initial height in m? ' );

%Determine the impact
tfin = findImpactTime(initial_h,initial_v,angle_launch);

%Make a plot of the trajectory
drawProjMotPlot(initial_h,initial_v,angle_launch,tfin);


function drawProjMotPlot(y0,v0,launchAng,tfin)
    global g;

    %Define functions for graph
    xpos=@(t) v0*cosd(launchAng)*t;
    ypos=@(t) y0+v0*sind(launchAng)*t-1/2*g*t^2;

    %Plot the function defining the x position vs. the y position
    fplot(xpos,ypos,[0 tfin],'r','LineWidth',2.);
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    title('Projectile Motion Example');
    set(gca,'FontSize',14);

end

function t_impact = findImpactTime(y0,v0,launchAng)

    global g;

    %Find the initial velocity
    vy0=v0*sind(launchAng);

    %Use the quadratic formula to find the impact time
    t_impact=(-vy0-sqrt(vy0^2+2*g*y0))/(-g);

end
```

Screenshot




```
%Define the acceleration of gravity as a global variable
global g;
g=1.6;  %acceleration of gravity on the Moon, m/s^2
```