1) Using the top-down design approach, write an algorithmfor making a sandwich.

- Get the ingredients
- Get the utensils
- Assemble the sandwich

Get the ingredients:

- Get the bread
- Get the cheese
- Get the condiments

Get the bread:

- Open the bread box
- Select desired bread
- Open bag and remove2 slices

etc.

2) Write a simple scriptthat will calculatethe volume of a hollowsphere,

$$\frac{4\pi}{3}\left(r_o^3 - r_1^3\right)$$

where ri is the inner radius and ro is the outer radius. Assign a value to a variable for the inner radius, and also assign a value to another variable for the outer radius. Then, using these variables, assign the volume to a third variable. Include comments in the script. Use **help** toview the commentin your script.

```
% This script calculates the volume of a hollow sphere

% Assign values for the inner and outer radii
ri = 5.1;
ro = 6.8;
% Calculate the volume
vol = (4*pi)/3*(ro^3-ri^3)
```

```
vol = 761.4425
```

3) Write a statement that prompts the user for his/her favorite number.

```
favnum = input('What is your favorite number: ')
```

```
favnum = 810
```

4) Write a statement that prompts the user for his/her name.

```
uname = input('What is your name: ', 's')
```

```
uname =
'Varun'
```

5) Write an input statement that will prompt the user for a real number,and store it in a variable. Then, use the fprintf function to print the value of this variable using 2 decimalplaces.

```
realnum = input('Enter a real number: ');
fprintf('The number is %.2f\n', realnum)
```

```
The number is 810.00
```

6) Experiment, in the Command Window, with using the fprintf function for real numbers. Make a note of what happens for each. Use fprintf to print the real number 12345.6789.

```
realnum = 12345.6789;
```

- without specifying any field width

```
fprintf('The number is %f\n', realnum)
```

```
The number is 12345.678900
```

- in a field width of10 with 4 decimalplaces

```
fprintf('The number is %10.4f\n', realnum)
```

```
The number is 12345.6789
```

- in a field width of10 with 2 decimalplaces

```
fprintf('The numberis %10.2f\n', realnum)
```

```
The numberis   12345.68
```

- in a field width of 6 with 4 decimal places

```
fprintf('The number is %6.4f\n', realnum)
```

```
The number is 12345.6789
```

- in a field width of 2 with 4 decimal places

```
fprintf('The number is %2.4f\n', realnum)
```

```
The number is 12345.6789
```

7) Experiment, in the CommandWindow, with using the fprintf function for integers. Make a note of what happens for each. Use fprintf to print the integer 12345.

```
intnum = 12345;
```

- without specifying any field width

```
fprintf('The number is %d\n', intnum)
```

```
The number is 12345
```

- in a field width of 5

```
fprintf('The number is %5d\n', intnum)
```

```
The number is 12345
```

- in a field width of 8

```
fprintf('The number is %8d\n', intnum)
```

```
The number is    12345
```

- in a field width of 3

```
fprintf('The number is %3d\n', intnum)
```

```
The number is 12345
```

8) When would you use disp instead of fprintf? When would you use fprintf instead of disp? The disp function is used when no formatting is required. It is also easier to print vectors and matrices using disp. The fprintf function is used for formatted output.

9) Write a script called*echostring* that will promptthe user for a string,and will echo print the string in quotes:

```
>> echostring

Enter your string: hi there

Your string was: 'hi there'
```

```
% Prompt the user and print a string in quotes
str = input('Enter your string: ', 's');
fprintf('Your string was: ''%s''\n',str)
```

```
Your string was: 'Varun Khadayate'
```

10) If the lengths of two sides of a triangle and the angle between them are known, the length of the third side can be calculated. Given the lengths of two sides (b and c) of a triangle, and the angle between them α in degrees, the third side a is calculated as follows:

$$a^2 = b^2 + c^2 - 2bc\cos(\alpha)$$

Write a script *thirdside* thatwill prompt the user and read in values for b, c, and α (in degrees), and then calculate and print the value of a with 3 decimal places. The format of the output from the script should look exactlylike this:

```
>> thirdside

Enter the first side: 2.2

Enter the second side: 4.4

Enter the angle between them: 50

The third side is 3.429
```

```
% Calculates the third side of a triangle, given
% the lengths of two sides and the angle between them
b = input('Enter the first side: ');
c = input('Enter the second side: ');
alpha = input('Enter the angle between them: ');
a = sqrt(b^2 + c^2 -2*b*c*cosd(alpha));
fprintf('\nThe third side is %.3f\n', a)
```

```
The third side is 3.429
```

11) Write a script that will prompt the user for a character, and will print it twice; once left-justified in a field width of 5, and again right-justified in a field width of 3.

```
mych = input('Enter a character: ', 's');
fprintf('Here it is: %-5c and again: %3c\n',mych,mych)
```

```
Here it is: v     and again:   a
Here it is: r     and again:   u
Here it is: n     and again:
Here it is: k     and again:   h
Here it is: a     and again:   d
Here it is: a     and again:   y
Here it is: a     and again:   t
Here it is: e     and again:   v
Here it is: a     and again:   r
Here it is: u     and again:   n
Here it is:       and again:   k
Here it is: h     and again:   a
Here it is: d     and again:   a
Here it is: y     and again:   a
Here it is: t     and again:   e
```

12) Write a script *lumin* that will calculate and print the luminosity L of a star in Watts. The luminosity L is given by $L = 4\pi d^2 b$ where d is the distance from the sun in meters and b is the brightness in Watts/meters2. Here is an example of executing the script:

```
>> lumin

This scriptwill calculate the luminosity of a star.

When prompted, enter the star'sdistance from the sun in meters, and its brightness in
W/meters squared.

Enter the distance: 1.26e12 Enter the brightness: 2e-17

The luminosity of this star is 399007399.75 watts
```

```
% Calculates the luminosity of a star
disp('This script will calculate the luminosity of a star.')
```

```
This script will calculate the luminosity of a star.
```

```
disp('When prompted, enter the star''s distance from the sun')
```

```
When prompted, enter the star's distance from the sun
```

```
fprintf(' in meters, and its brightness in W/meters squared.\n\n')
```

```
 in meters, and its brightness in W/meters squared.
```

```
d = input('Enter the distance: ');
b = input('Enter the brightness: ');
L = 4*pi*d^2*b;
fprintf('The luminosity of this star is %.2f watts\n', L)
```

```
The luminosity of this star is 399007399.75 watts
```

13) A script *iotrace* has been written. Here's what the desired output looks like:

```
>>iotrace

Please enter a number: 33 Please enter a character: x Your numberis 33.00

Your char is      x!
```

Fix this script so that it works as shown above:

```
mynum = input('Please enter a number:\n ');
mychar = input('Please enter a character: ', 's');
fprintf('Your number is %.2f\n', mynum)
```

```
Your number is 12.00
```

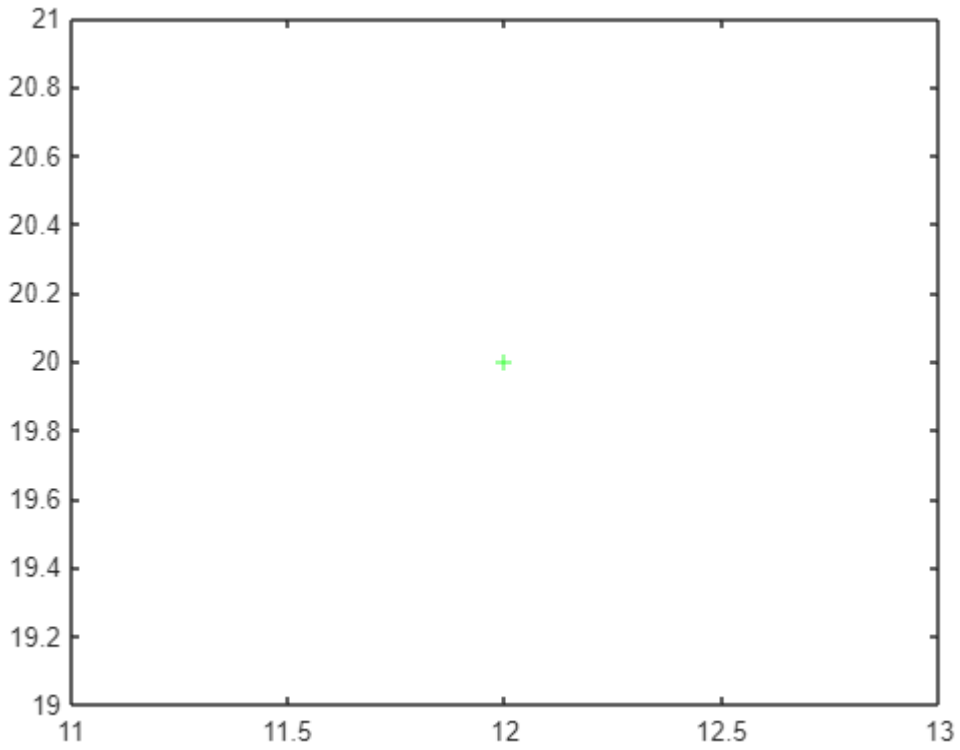
```
fprintf('Your char is %6c!\n', mychar)
```

```
Your char is      V!
Your char is      a!
Your char is      r!
Your char is      u!
Your char is      n!
```

14) Write a script that assigns values for the x coordinateand then y coordinate of a point, and then plots this using a green +.

```
% Prompt the user for the coordinates of a point and plot
% the point using a green +

x = input('Enter the x coordinate: ');
y = input('Enter the y coordinate: ');
plot(x,y, 'g+')
```

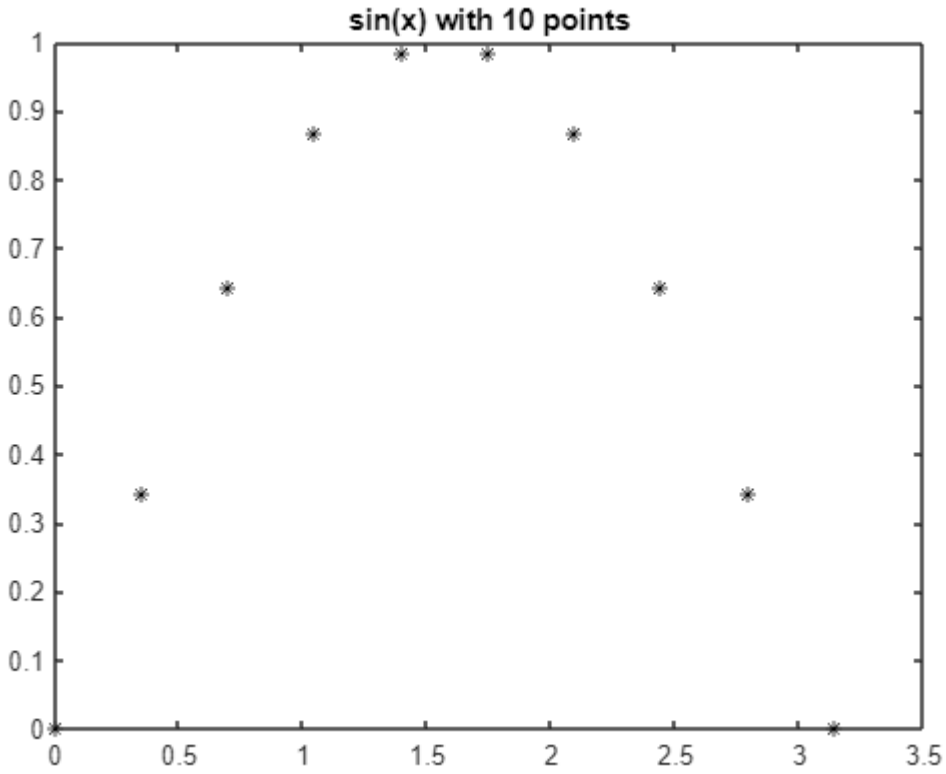


15) Plot sin(x) for x values ranging from 0 to π (in separate Figure Windows):

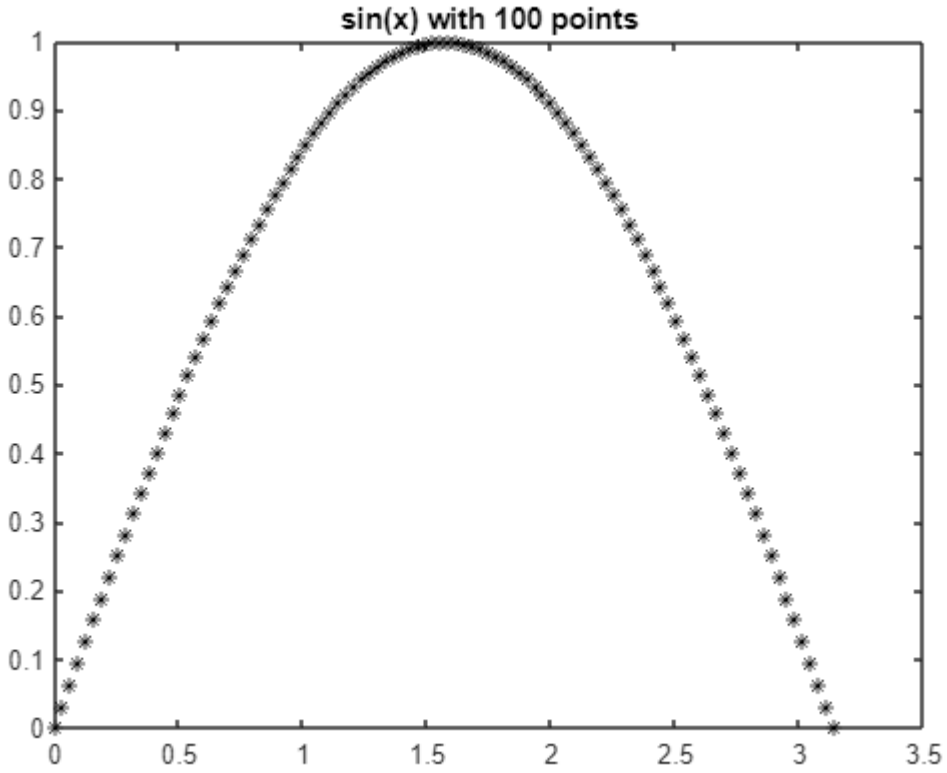
- using 10 points in this range
- using 100 points in this range

```
% Plots sin(x) with 10 points and 100 points in range 0 to pi
x = linspace(0,pi,10);
y = sin(x);
```

```
clf
figure(1)
plot(x,y,'k*')
title('sin(x) with 10 points')
```




```
figure(2)
x = linspace(0,pi); y = sin(x);
plot(x,y,'k*')
title('sin(x) with 100 points')
```



16) When would it be important to use legend in a plot?

When you have more than one plot in a single Figure Window.

17) Why do we always suppress all assignment statements in scripts?

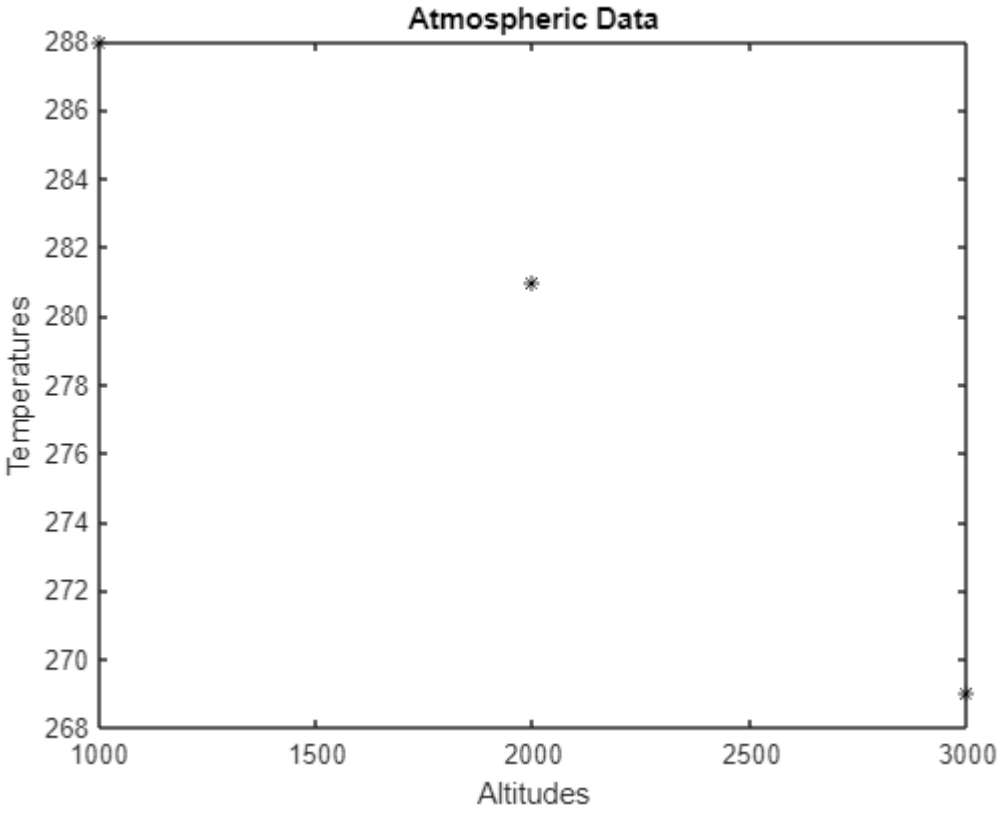
- So we don't just see the variable = and then the value.

18) Atmospheric properties such as temperature, air density, and air pressure are important in aviation. Create a file that stores temperatures in degrees Kelvin at variousaltitudes. The altitudes are in the first column and the temperatures in the second. For example, it may look like this:

```
1000     288
2000     281
3000     269
```

```
% Read altitudes and temperatures from a file and plot
row1 = [1000 288];
row2 = [2000 281];
row3 = [3000 269];
mat = [row1;row2;row3];
% Save the matrix as an ASCII file.
```

```
save alttemps.dat mat -ascii
load alttemps.dat
altitudes = alttemps(:,1);
temps = alttemps(:,2);
plot(altitudes,temps,'k*')
xlabel('Altitudes')
ylabel('Temperatures')
title('Atmospheric Data')
```

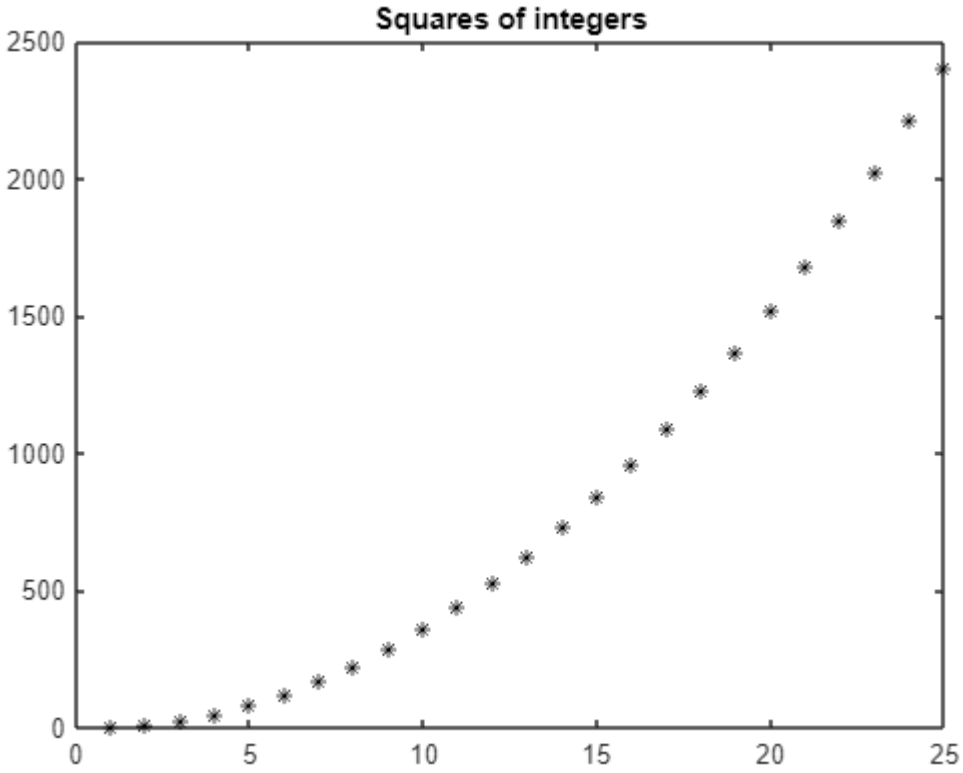


19) Generate a random integer*n*, create a vector of the integers 1 through *n* in steps of 2, square them, andplot the squares.

```
% Create a vector of integers 1:2:n where n is random
% square them and plot the squares
n = randi([1,50])
```

```
n = 49
```

```
vec = 1:2:n;
vecsq = vec .^ 2;
plot(vecsq,'k*')
title('Squares of integers')
```



20) Create a *3 x 6* matrix of random integers,each in the range of 50 - 100. Write this to a file called*randfile.dat*. Then, create a new matrixof random integers,but this time make it a *2 x 6* matrix of random integers,each in the range of 50 - 100. Append this matrix to the original file. Then, read the file in (which will be to a variable called *randfile*) just to make sure that worked!

```
mat = randi([50,100], 3,6)
```

```
mat = 3×6
    92    62    60    74    79    64
    62    97    62    67    78    88
    91    67    81    92    96    88
```

```
save randfile.dat mat -ascii
newmat = randi([50,100], 2,6)
```

```
newmat = 2×6
    69    53    77    97    79    50
    78    52    89    56    73    67
```

```
save randfile.dat newmat -ascii -append
load randfile.dat
```

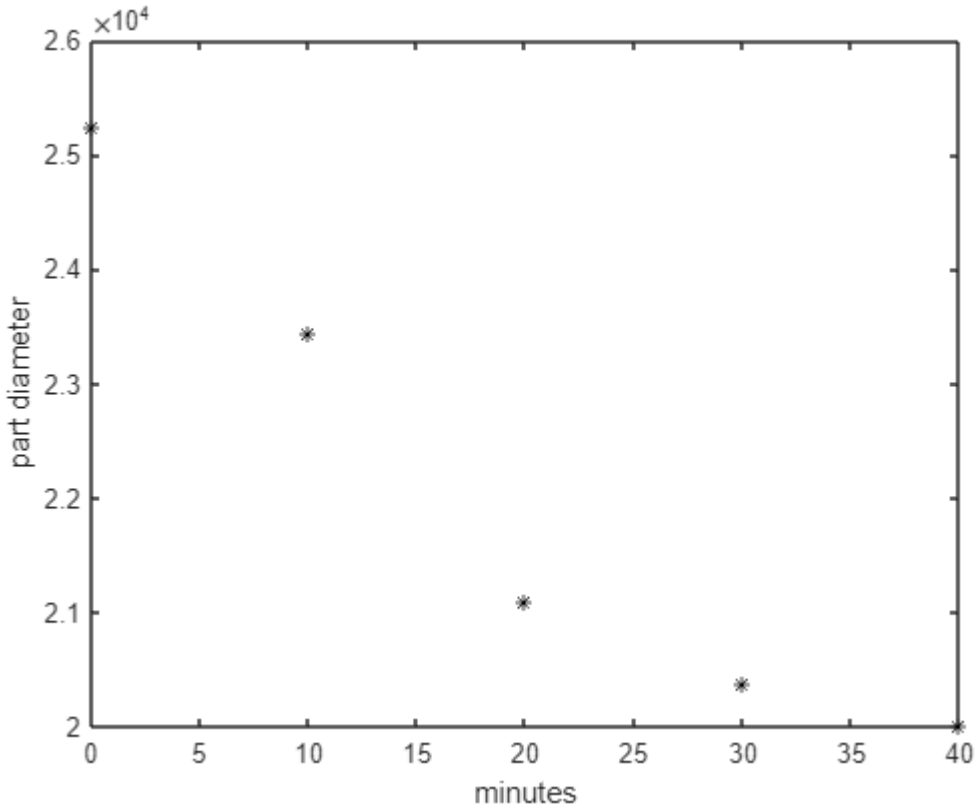
21) A particular part is being turned on a lathe. The diameter of the part is supposed to be 20,000 mm. The diameter is measured every 10 minutes and the results are stored in a file called partdiam.dat. Create a data file to

simulate this. The file will store the time in minutes and the diameter at each time. Plot the data.

```
row1 = [0 25233];
row2 = [10 23432];
row3 = [20 21085];
row4 = [30 20374];
row5 = [40 20002];
mat = [row1;row2;row3;row4;row5]
```

```
mat = 5×2
     0       25233
    10       23432
    20       21085
    30       20374
    40       20002
```

```
save partdiam.dat mat -ascii
load partdiam.dat
mins = partdiam(:,1);
diams = partdiam(:,2);
plot(mins,diams,'k*')
xlabel('minutes')
ylabel('part diameter')
```



22) Create a file called “testtan.dat” comprised of two lines with three real numbers on each line (some negative, some positive, in the -1 to 3

range). The file can be created from the Editor, or saved from a matrix. Then, load the file into a matrix and calculate the tangent of every element in the resulting matrix.

```
mat = rand(2,3)*4-1
```

```
mat = 2×3
   -0.3513    0.2449   -0.3374
    2.1771    1.1141    1.4079
```

```
save testtan.dat mat -ascii
load testtan.dat
tan(testtan)
```

```
ans = 2×3
   -0.3665    0.2499   -0.3508
   -1.4420    2.0354    6.0855
```

23) Write a function calcrectarea that will calculate and return the area of a rectangle. Pass the length and width to the function as input arguments.

```
% function area = MIDTERM_2_PREP(length, width)
% % This function calculates the area of a rectangle
% % Format of call: calcrectarea(length, width)
% % Returns the area
% area = length * width;
% end
% calcrectarea(12,5)
%
% ans =
%
%     60
```

Renewableenergy sources such as biomassare gaining increasing attention. Biomass energy units include megawatt hours (MWh) and gigajoules(GJ). One MWh is equivalent to 3.6 GJ. For example,one cubic meterof wood chips produces 1 MWh.

24) Write a function *mwh_to_gj* that will convert from MWh to GJ.

```
% function out = mwh_to_gj(mwh)
% % Converts from MWh to GJ
%
% % Format of call: mwh_to_gj(mwh)
% % Returns gigajoules
% out = mwh * 3.6;
% end
% mwh_to_gj(34)
%
% ans =
%
%   122.4000
```

25) List some differences between a script and a function.

- A function has a header whereas a script does not.
- A function typically has end at the **end** ofthe file.
- A function is called whereasa script is executed.
- Arguments are passed to functions but not to scripts.
- Functions can return argumentswhereas scripts cannot.
- The block comment is typically in the beginningof a script but under the functionheader.
- The scope of variables is different: scripts use the baseworkspace, whereas functions have their own workspaces.

26) In quantum mechanics, the angular wavelengthfor a wavelength λ is defined as $\frac{\lambda}{2\pi}$. Write a function named *makeitangular* that will receive the wavelength as an input argument, and will return the angular wavelength.

```
% function angwave = makeitangular(wavelength)
% angwave = wavelength/(2*pi);
% end
% makeitangular(250)
%
% ans =
%
%    39.7887
```

27) Write a fives function that will receive two arguments for the number of rows and columns, and will return a matrix with that size of all fives.

```
% function five = fives(r,c)
% % Returns a matrix of fives of specified size
% % Format of call: fives(rows, cols)
% % Returns a rows by cols matrix of all fives
%
% % Initialization
% five = zeros(r,c) + 5;
% end
% ans =
%
%      5     5     5     5
%      5     5     5     5
%      5     5     5     5
```

28) Write a function *isdivby4* that will receive an integer input argument, and will return logical 1 for true if the input argumentis divisible by 4, or logical false if it is not. Write a function *isdivby4* that will receive an integer input argument, and will return logical 1 for true if the input argumentis divisible by 4, or logical false if it is not.

```
% function out = isdivby4(inarg)
% % Returns 1 for true if the input argument is
% % divisible by 4 or 0 for false if not
% % Format of call: isdivby4(input arg)
% % Returns whether divisible by 4 or not
% out = rem(inarg,4) == 0;
% end
% isdivby4(12)
%
% ans =
%
%   logical
%
%    1
%
% isdivby4(1)
%
% ans =
%
%   logical
%
%    0
```

29) Write a function *isint* that will receive a number input argument *innum*, and will return 1 for true if this number is an integer, or 0 for false if not. Use the fact that *innum* should be equal to int32(innum) if it is an integer. Unfortunately, due to round-off errors, it should be noted that it is possible to get logical 1 for true if the input argument is close to an integer. Therefore the output may not be what you mightexpect, as shown here.

```
>> isint(4)

ans =

1

>> isint(4.9999)

ans =

0

>> isint(4.99999999999999999999999999)

ans =
```

1

```
% function out = isint(innum)
% % Returns 1 for true if the argument is an integer
% % Format of call: isint(number)
% % Returns logical 1 iff number is an integer
% out = innum == int32(innum);
% end
% isint(0810)
%
% ans =
%
%   logical
%
%    1
%
% isint(0810.1172)
%
% ans =
%
%   logical
%
%    0
```

30) A Pythagorean triple is a set of positive integers (a,b,c) such that $a^2 + b^2 = c^2$. Write a function ispythag that will receive three positive integers (a, b, c in that order) and will return logical 1 for true if they form a Pythagorean triple, or 0 for false if not.

```
% function out = ispythag(a,b,c)
% % Determines whether a, b, c are a Pythagorean triple or not
% % Format of call: ispythag(a,b,c)
% % Returns logical 1 if a Pythagorean triple
% out = a^2 + b^2 == c^2;
% end
% ispythag(12,24,2)
%
% ans =
%
%   logical
%
%    0
%
% ispythag(3,4,5)
%
% ans =
%
%   logical
%
```

```
%    1
```

31) A function can return a vector as a result. Write a function vecout that will receive one integer argument and will return a vector that increments from the value of the input argument to its value plus 5, using the colon operator. For example,

>> vecout(4)

ans =

4 5 6 7 8 9

```
% function outvec = vecout(innum)
% % Create a vector from innum to innum + 5
% % Format of call: vecout(input number)
% % Returns a vector input num : input num+5
% outvec = innum:innum+5;
% end
% vecout(20)
%
% ans =
%
%     20    21    22    23    24    25
```

32) Write a function called pickone, which will receive one input argument x, which is a vector, and will return one random element from the vector. For example,

>> pickone(4:7)

ans =

5

>> disp(pickone(-2:0))

-1

>> help pickone

pickone(x) returns a random element from vector x

```
% function elem = pickone(invec)
% % pickone(x) returns a random element from vector x
% % Format of call: pickone(vector)
% % Returns random element from the vector
% len = length(invec);
% ran = randi([1, len]);
% elem = invec(ran);
```

```
% end
% pickone(4:7)
%
% ans =
%
%      7
```

33) The conversion depends on the temperature and other factors, but an approximation is that 1 inch of rain is equivalent to 6.5 inches of snow. Write a script that prompts the user for the number of inches of rain, calls a function to return the equivalent amount of snow, and prints this result. Write the function, as well!

```
% Prompt the user for a number of inches of rain
% and call a function to calculate the
% equivalent amountof snow
rain = input('How much rain in inches: ');
snow = rainToSnow(rain);
fprintf('%.1f inches of rain would be ', rain)
```

```
25.0 inches of rain would be
```

```
fprintf('%.1f inches of snow\n', snow)
```

```
162.5 inches of snow
```

34) In thermodynamics, the Carnot efficiency is the maximum possible efficiency of a heat engine operating between two reservoirs at different temperatures. The Carnot efficiency is given as

$$\eta = 1 - \frac{T_C}{T_H}$$

where T_C and T_H are the absolute temperatures at the cold and hot reservoirs, respectively. Write a script "carnot" that will prompt the user for the two reservoir temperatures in Kelvin, call a function to calculate the Carnot efficiency, and then print the corresponding Carnot efficiency to 3 decimal places. Also write the function.

```
% Calculates the Carnot efficiency, given the temps
% of cold and hot reservoirs, error-checking both
Tc = input('Enter the cold reservoir temperature: ');
Th = input('Enter the hot reservoir temperature: ');
fprintf('The Cold and Hot tempreature of reservoir is %d and %d respectively\n',Tc,Th)
```

```
The Cold and Hot tempreature of reservoir is 35 and 180 respectively
```

```
carnotEff = calcCarnot(Tc, Th);
```

```
fprintf('The Carnot efficiency is %.3f\n',carnotEff)
```

```
The Carnot efficiency is 0.806
```

35) Many mathematical models in engineering use the exponential function. The general form of the exponential decay function is:

$$y(t) = Ae^{-\tau t}$$

where A is the initial value at t=0, and τ is the time constant for the function. Write a script to study the effect of the time constant. To simplify the equation, set A equal to 1. Prompt the user for two different values for the time constant, and for beginning and ending values for the range of a t vector. Then, calculate two different y vectors using the above equation and the two time constants, and graph both exponential functions on the same graph within the range the user specified. Use a function to calculate y. Make one plot red. Be sure to label the graph and both axes. What happens to the decay rate as the time constant gets larger?

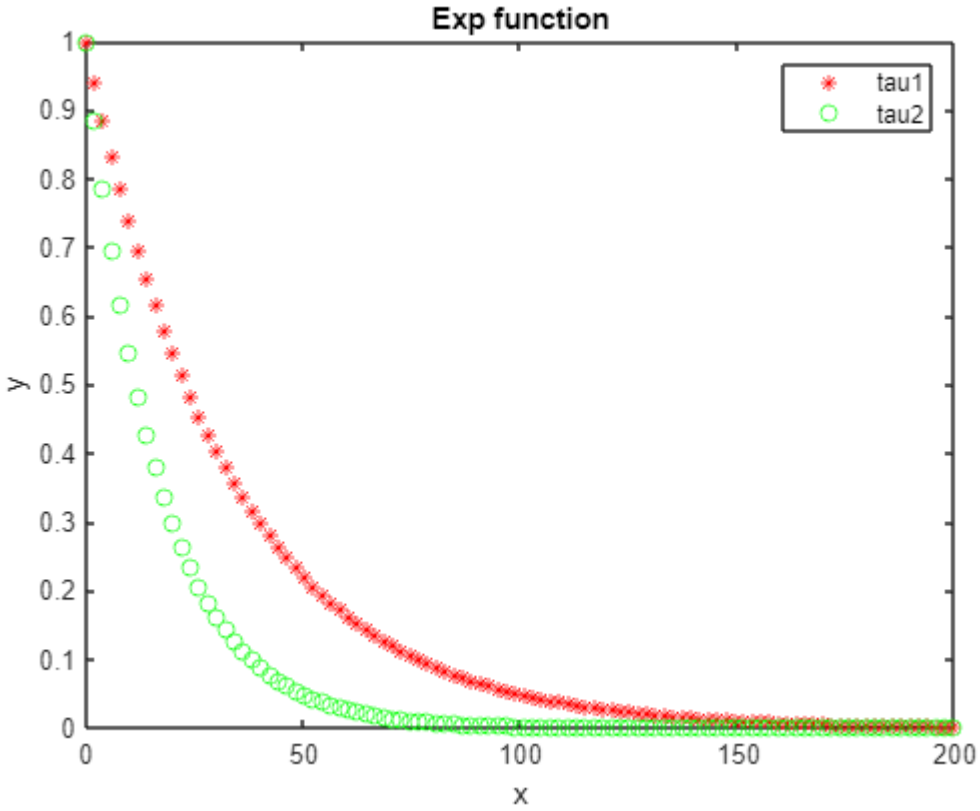
```
A = 1;
tau1 = input('Enter a time constant: ');
tau2 = input('Enter another time constant: ');
tstart = input('Enter the beginning t: ');
tend = input('Enter the end of t: ');
fprintf("The time constant 1 and 2 are %f and %f respectively",tau1,tau2)
```

```
The time constant 1 and 2 are 0.030000 and 0.060000 respectively
```

```
fprintf("The beginning and end of time t are %d and %d respectively",tstart,tend)
```

```
The beginning and end of time t are 0 and 200 respectively
```

```
t = linspace(tstart,tend);
y1 = expfn(A, t, tau1);
y2 = expfn(A, t, tau2);
plot(t,y1,'r*',t,y2,'go')
xlabel('x')
ylabel('y')
title('Exp function')
legend('tau1','tau2')
```


Exp function
tau1
tau2
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200