```
>> clear all
>> exercise1script

x =

     3


y =

   27.1411

>> x = 4

x =

     4

>> exercise1script

y =

   63.2432

>> x = 5

x =

     5

>> exercise1script

y =

  124.0411

>> x = 6

x =

     6

>> exercise1script

y =

  215.7206

>> exercise1func(3)

p =
```

```
   27.1411


ans =

   27.1411

>> exercise1func

x =

     3     4     5     6


p =

   27.1411   63.2432  124.0411  215.7206


ans =

   27.1411   63.2432  124.0411  215.7206

>> a = 12

a =

    12

>> b = 4

b =

     4

>> lec_16
Error using rem
Not enough input arguments.

Error in lec_16 (line 1)
y = rem(a/b) %Assigns the remainder of a/b

>> help abs
 abs    Absolute value.
    abs(X) is the absolute value of the elements of X. When
    X is complex, abs(X) is the complex modulus (magnitude) of
    the elements of X.

    See also sign, angle, unwrap, hypot.
```

```
    Documentation for abs
    Other functions named abs

>> help rem
 rem    Remainder after division.
    rem(x,y) returns x - fix(x./y).*y if y ~= 0, carefully computed to
    avoid rounding error. If y is not an integer and the quotient x./y is
    within roundoff error of an integer, then n is that integer. The inputs
    x and y must be real and have compatible sizes. In the simplest cases,
    they can be the same size or one can be a scalar. Two inputs have
    compatible sizes if, for every dimension, the dimension sizes of the
    inputs are either the same or one of them is 1.

    By convention:
       rem(x,0) is NaN.
       rem(x,x), for x~=0, is 0.
       rem(x,y), for x~=y and y~=0, has the same sign as x.

    Note: MOD(x,y), for x~=y and y~=0, has the same sign as y.
    rem(x,y) and MOD(x,y) are equal if x and y have the same sign, but
    differ by y if x and y have different signs.

    See also mod.

    Documentation for rem
    Other functions named rem

>> lec_16

y =

     0

>> x = 6

x =

     6

>> lec_16

y =

     6

>> x = -6

x =

    -6
```

```
>> lec_16

y =

     6

>> x = [3 4 5 6]

x =

     3     4     5     6

>> y = [0 2 4 6]

y =

     0     2     4     6

>> lec_16
Unrecognized function or variable 'xArray'.

Error in lec_16 (line 3)
plot(xArray,yArray) % Description: Plots a graph, given an array of x-coordinates,↙
xArray, and an array of y-coordinates,yArray.

>> xArray = [3 4 5 6]

xArray =

     3     4     5     6

>> yArray = [0 2 4 6]

yArray =

     0     2     4     6

>> lec_16
>> round(a)_
 round(a)_
         ↑
Error: Invalid text character. Check for unsupported symbol, invisible character, or↙
pasting of non-ASCII characters.

>> round(a)

ans =

    12

>> help round
```

```
 round  rounds towards nearest decimal or integer

    round(X) rounds each element of X to the nearest integer.

    round(X, N), for positive integers N, rounds to N digits to the right
    of the decimal point. If N is zero, X is rounded to the nearest integer.
    If N is less than zero, X is rounded to the left of the decimal point.
    N must be a scalar integer.

    round(X, N, 'significant') rounds each element to its N most significant
    digits, counting from the most-significant or left side of the number.
    N must be a positive integer scalar.

    round(X, N, 'decimals') is equivalent to round(X, N).

    round(..., TieBreaker=DIRECTION) breaks ties as specified by DIRECTION.
    Ties are rare. A tie occurs only when 10^N*X is within roundoff error
    of a point halfway between two consecutive integers. DIRECTION must be
    one of:
         "fromzero" - round ties away from zero (default)
           "tozero" - round ties towards zero
             "even" - round ties to nearest even integer
              "odd" - round ties to nearest odd integer
          "plusinf" - round ties to the right, towards +infinity
         "minusinf" - round ties to the left, towards -infinity

    For complex X, the imaginary and real parts are rounded independently.

    Examples
    --------
    % Round pi to the nearest hundredth
    round(pi, 2)
    %   3.14

    % Round the equatorial radius of the Earth, 6378137 meters,
    % to the nearest kilometer.
    round(6378137, -3)
    %   6378000

    % Round to 3 significant digits
    format shortg;
    round([pi, 6378137], 3, 'significant')
    %   3.14     6.38e+06

    % If you only need to display a rounded version of X,
    % consider using fprintf or num2str:

    fprintf('%.3f\n', 12.3456)
    %   12.346
    fprintf('%.3e\n', 12.3456)
    %   1.235e+01
```

```
   See also floor, ceil, fprintf.

    Documentation for round
    Other functions named round

>> a = 12.3345567889

a =

   12.3346

>> lec_16

ans =

    12

>> help myRand
--- myRand not found. Showing help for rand instead. ---

 rand Uniformly distributed pseudorandom numbers.
    R = rand(N) returns an N-by-N matrix containing pseudorandom values drawn
    from the standard uniform distribution on the open interval(0,1).  rand(M,N)
    or rand([M,N]) returns an M-by-N matrix.  rand(M,N,P,...) or
    rand([M,N,P,...]) returns an M-by-N-by-P-by-... array.  rand returns a
    scalar.  rand(SIZE(A)) returns an array the same size as A.

    Note: The size inputs M, N, P, ... should be nonnegative integers.
    Negative integers are treated as 0.

    R = rand(..., CLASSNAME) returns an array of uniform values of the
    specified class. CLASSNAME can be 'double' or 'single'.

    R = rand(..., 'like', Y) is an array of uniform values with the same
    data type and complexity (real or complex) as the numeric variable Y.

    The sequence of numbers produced by rand is determined by the settings of
    the uniform random number generator that underlies rand, RANDI, and RANDN.
    Control that shared random number generator using RNG.

    Examples:

       Example 1: Generate values from the uniform distribution on the
       interval (a, b).
          r = a + (b-a).*rand(100,1);

       Example 2: Use the RANDI function, instead of rand, to generate
       integer values from the uniform distribution on the set 1:100.
          r = randi(100,1,5);
```

```
       Example 3: Reset the random number generator used by rand, RANDI, and
       RANDN to its default startup settings, so that rand produces the same
       random numbers as if you restarted MATLAB.
          rng('default')
          rand(1,5)

       Example 4: Save the settings for the random number generator used by
       rand, RANDI, and RANDN, generate 5 values from rand, restore the
       settings, and repeat those values.
          s = rng
          u1 = rand(1,5)
          rng(s);
          u2 = rand(1,5) % contains exactly the same values as u1

       Example 5: Reinitialize the random number generator used by rand,
       RANDI, and RANDN with a seed based on the current time.  rand will
       return different values each time you do this.  NOTE: It is usually
       not necessary to do this more than once per MATLAB session.
          rng('shuffle');
          rand(1,5)

    See Replace Discouraged Syntaxes of rand and randn to use RNG to replace
    rand with the 'seed', 'state', or 'twister' inputs.

    See also randi, randn, rng, RandStream, RandStream/rand,
             sprand, sprandn, randperm.

    Documentation for rand
    Other functions named rand

>> myRand(1,10)

ans =

    8.3325

>> myRand(100,100+1)

ans =

  100.9058

>> myRand(3,pi)

ans =

    3.0180

>> myRand(20)
Not enough input arguments.
```

```
Error in myRand (line 2)
scale = maxRand - minRand;

>> myRand(20,1)

ans =

    2.6459

>> twoTo8 = twoN(8)

twoTo8 =

   256

>> newNumber = twoN(5)

newNumber =

    32

>> squareOfTwo = twoN(2)

squareOfTwo =

     4

>> twoN(9)

ans =

   512

>> rootOfPower = twoN(5)^(1/2)

rootOfPower =

    5.6569

>> twoN
Not enough input arguments.

Error in twoN (line 6)
y = 2^n;

>> twoN % This wont work since w=the value of n os not defined
Not enough input arguments.

Error in twoN (line 6)
y = 2^n;
```

```
>> quadRoots(1,3,2)

ans =

    -1
    -2

>> quadRoots(1,6,10)

ans =

  -3.0000 + 1.0000i
  -3.0000 - 1.0000i

>> quadRoots(1,6,13)

ans =

  -3.0000 + 2.0000i
  -3.0000 - 2.0000i

>> myCubic(-5)

ans =

   -58

>> myCubic(5)

ans =

   142

>> x = [-5:5]

x =

    -5    -4    -3    -2    -1     0     1     2     3     4     5

>> cubicExercise
Error using  ^
Incorrect dimensions for raising a matrix to a power. Check that the matrix is square and ↙
the power is a scalar. To operate
on each element of the matrix individually, use POWER (.^) for elementwise power.

Error in myCubic (line 7)
y = x^3 + 2*x^2 - 5*x - 8;

Error in cubicExercise (line 3)
A = plot(myCubic(x))
```

```
>> cubicExercise

A =

  Line with properties:

            Color: [0 0.4470 0.7410]
        LineStyle: '-'
        LineWidth: 0.5000
           Marker: 'none'
       MarkerSize: 6
  MarkerFaceColor: 'none'
            XData: [1 2 3 4 5 6 7 8 9 10 11]
            YData: [-58 -20 -2 2 -2 -8 -10 -2 22 68 142]

  Show all properties

Error using matlab.internal.math.isLocalExtrema>parseInputs
First argument must be a numeric or logical array, a table, or a timetable.

Error in matlab.internal.math.isLocalExtrema (line 11)
        parseInputs(A, varargin{:});

Error in islocalmin (line 105)
    tf = matlab.internal.math.isLocalExtrema(A, false, varargin{:});

Error in cubicExercise (line 5)
B = islocalmin(A)

>> x = [0:5]

x =

     0     1     2     3     4     5

>> cubicExercise

B =

  1×6 logical array

   0   1   0   0   0   0

>> cubicExercise

ans =

    2.1249
   -2.7616
   -1.3633
```

```
>> cubicExercise
Not enough input arguments.

Error in myCubic (line 7)
y = x.^3 + 2*x.^2 - 5*x - 8;

Error in cubicExercise (line 7)
roots(myCubic)

>> cubicExercise

ans =

    2.1249
   -2.7616
   -1.3633

>> x = [-5:5]

x =

    -5    -4    -3    -2    -1     0     1     2     3     4     5

>> cubicExercise
Execution of script cubicExercise as a function is not supported:
E:\College-Codes\Fourth Year\SEM VII\IT  WS\Practical Work\cubicExercise.m

Error in quadl (line 62)
y = feval(f,x,varargin{:}); y = y(:).';

Error in cubicExercise (line 10)
quadl(@cubicExercise,-5,5)

>> cubicExercise
Execution of script cubicExercise as a function is not supported:
E:\College-Codes\Fourth Year\SEM VII\IT  WS\Practical Work\cubicExercise.m

Error in integralCalc/iterateScalarValued (line 314)
                fx = FUN(t);

Error in integralCalc/vadapt (line 132)
            [q,errbnd] = iterateScalarValued(u,tinterval,pathlen);

Error in integralCalc (line 75)
        [q,errbnd] = vadapt(@AtoBInvTransform,interval);

Error in integral (line 87)
Q = integralCalc(fun,a,b,opstruct);

Error in cubicExercise (line 10)
integral(@cubicExercise,-5,5)
```

```
>> cubicExercise

ans =

   86.6667

>>
```