```
>> help rand
 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) returns an array of uniform values of the
    same class as 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.

    Reference page for rand
    Other functions named rand

>> doc rand
>> A = [2, 0, -1; 0, 1, 2]

A =

     2     0    -1
     0     1     2

>> B = rand(5, 10)

B =

  Columns 1 through 5

    0.8147    0.0975    0.1576    0.1419    0.6557
    0.9058    0.2785    0.9706    0.4218    0.0357
    0.1270    0.5469    0.9572    0.9157    0.8491
    0.9134    0.9575    0.4854    0.7922    0.9340
    0.6324    0.9649    0.8003    0.9595    0.6787

  Columns 6 through 10

    0.7577    0.7060    0.8235    0.4387    0.4898
    0.7431    0.0318    0.6948    0.3816    0.4456
    0.3922    0.2769    0.3171    0.7655    0.6463
    0.6555    0.0462    0.9502    0.7952    0.7094
    0.1712    0.0971    0.0344    0.1869    0.7547

>> B(3, 7)

ans =

    0.2769

>> % it means 3rd row 7th column
>> B(3, 7) = 0;
>> B(3,7)

ans =

     0

>> % also we can altering the element in metrics
>> % we can't remove the element. only we can remove entire col or row
>> B(3,:) = [];
>> B

B =
  Columns 1 through 5

    0.8147    0.0975    0.1576    0.1419    0.6557
    0.9058    0.2785    0.9706    0.4218    0.0357
    0.9134    0.9575    0.4854    0.7922    0.9340
    0.6324    0.9649    0.8003    0.9595    0.6787

  Columns 6 through 10

    0.7577    0.7060    0.8235    0.4387    0.4898
    0.7431    0.0318    0.6948    0.3816    0.4456
    0.6555    0.0462    0.9502    0.7952    0.7094
    0.1712    0.0971    0.0344    0.1869    0.7547

>> % simply whole raw can be remove
>> B(:, 5) = [];
>> B

B =

  Columns 1 through 5

    0.8147    0.0975    0.1576    0.1419    0.7577
    0.9058    0.2785    0.9706    0.4218    0.7431
    0.9134    0.9575    0.4854    0.7922    0.6555
    0.6324    0.9649    0.8003    0.9595    0.1712

  Columns 6 through 9

    0.7060    0.8235    0.4387    0.4898
    0.0318    0.6948    0.3816    0.4456
    0.0462    0.9502    0.7952    0.7094
    0.0971    0.0344    0.1869    0.7547

>> % simply we can remove whole column
>> size(B)

ans =

     4     9

>> % we can see the size of the array that B has 4 rows and 9 columns
>> [r, c] = size(B)

r =

     4


c =

     9

>> % simply we can assign row and columns size to the two variables
>> row = size(B, 1);
>> col = size(B, 2)

col =

     9

>> row

row =

     4

>> col

col =

     9

>> % transpose
>> B'

ans =

    0.8147    0.9058    0.9134    0.6324
    0.0975    0.2785    0.9575    0.9649
    0.1576    0.9706    0.4854    0.8003
    0.1419    0.4218    0.7922    0.9595
    0.7577    0.7431    0.6555    0.1712
    0.7060    0.0318    0.0462    0.0971
    0.8235    0.6948    0.9502    0.0344
    0.4387    0.3816    0.7952    0.1869
    0.4898    0.4456    0.7094    0.7547

>> size(B')

ans =

     9     4

>> B(3,:)

ans =

  Columns 1 through 5

    0.9134    0.9575    0.4854    0.7922    0.6555

  Columns 6 through 9

    0.0462    0.9502    0.7952    0.7094

>> % access the third row only
>> B(:, 2)

ans =
    0.0975
    0.2785
    0.9575
    0.9649

>> % access the 2nd column
>> size(B)

ans =

     4     9

>> B(2:3 , 4,6)
Index exceeds matrix dimensions.

>> B(2:3, 3:5)

ans =

    0.9706    0.4218    0.7431
    0.4854    0.7922    0.6555

>> C = B(2:3, 3:5)

C =

    0.9706    0.4218    0.7431
    0.4854    0.7922    0.6555

>> C

C =

    0.9706    0.4218    0.7431
    0.4854    0.7922    0.6555

>>
```