Patrick McMillin

PHYS 365

9/19/2016

Assignment 4\_1

**5)**

**Function:**

function [sumstep2] = sumsteps2(n)

% Takes an integer argument 'n' to sum from 1 to 'n' in steps of 2. Meaning

% summing all odd numbers from 1 to n.

for ind = 1:n

values = 1:2:n;

sumstep2 = sum(values);

end

end

**Command:**

>> sumsteps2(9)

ans =

25

**9)**

**Script:**

% Preallocates space for a vector of zeros of length 4. Then prompts the user for four

% numbers, which will then replace the zeros in the vector, and print the

% vector to the users screen through the command prompt.

vector = zeros(1,4);

fprintf('You will be prompted for four numbers, which will be taken and placed into a vector.\n')

for ind = 1:4

n = input('Please enter a number: \n');

vector(ind) = n ;

end

fprintf('Your vector is: \n')

disp(vector);

**Command:**

>> make\_vec

You will be prompted for four numbers, which will be taken and placed into a vector.

Please enter a number:

1

Please enter a number:

2

Please enter a number:

3

Please enter a number:

4

Your vector is:

1 2 3 4

**10)**

**Script:**

% Prompts the user for a vector. The script will then take the vector and

% display the index of each vector element, and the value of the element in

% sentence form.

vector = input('Please enter your vector: \n');

for index = 1:length(vector)

n = vector(index);

fprintf('Element %d of your vector is %f. \n', index, n);

end

**Command:**

>> vector\_index\_values

Please enter your vector:

1:2:20

Element 1 of your vector is 1.000000.

Element 2 of your vector is 3.000000.

Element 3 of your vector is 5.000000.

Element 4 of your vector is 7.000000.

Element 5 of your vector is 9.000000.

Element 6 of your vector is 11.000000.

Element 7 of your vector is 13.000000.

Element 8 of your vector is 15.000000.

Element 9 of your vector is 17.000000.

Element 10 of your vector is 19.000000.

**17)**

**Function:**

function [average] = matrix\_average(mat)

% Takes a two dimensional matrix, and uses 'for' loops to calculate the average of all the

% values in the matrix.

number\_of\_values = size(mat,1) \* size(mat,2);

total\_sum = 0;

for index\_1 = 1:size(mat,1)

for index\_2 = 1:size(mat,2)

total\_sum = total\_sum + mat(index\_1,index\_2);

end

end

average = total\_sum / number\_of\_values;

end

**Command:**

>> matrix = rand(5)

matrix =

0.3012 0.2259 0.9234 0.4389 0.2622

0.4709 0.1707 0.4302 0.1111 0.6028

0.2305 0.2277 0.1848 0.2581 0.7112

0.8443 0.4357 0.9049 0.4087 0.2217

0.1948 0.3111 0.9797 0.5949 0.1174

>> average = matrix\_average(matrix)

average =

0.4225

**20)**

**Script:**

% Loops through values of n until (1-(1/n))^n = e^(-1) up to four decimal

% places. Uses the built-in value of e^(-1) for this condition, and prints

% the value of n needed for such an approximation, and prints the value of

% e^(-1) used for the comparison.

n = 0;

while abs(round(exp\_est\*10000)/10000 - round((exp(1)^(-1))\*10000)/10000) > 0.00001

exp\_est = (1-(1/n))^n;

n = n+1;

end

fprintf('The value of n needed to approximate e^(-1) to four decimal places is %d. \n', n);

fprintf('The estimated value of e^(-1) using this n is %f. \n', exp\_est)

fprintf('The value of e^(-1) is %f. \n', exp(1)^(-1));

**Command:**

>> inv\_expon\_estimation

The value of n needed to approximate e^(-1) to four decimal places is 6250.

The estimated value of e^(-1) using this n is 0.367850.

The value of e^(-1) is 0.367879.