Workshop 3 - Working with loops in MATLAB

- Due 9 Aug by 23:59
- Points 1

Workshop 3 - Working with loops in MATLAB

This workshop is designed to give you some practice with using loops in MATLAB and working through some key concepts from this week's video content and interactive lecture. To prepare for this workshop you should create a folder in your account or on your computer called **workshop3**.

Each question starts with an **example test and program - complete with answers** and follows with **some actual questions** that you should attempt to do.

If you are sharing a terminal make sure that take turns at working during this practical so every group member has a chance to save some working code.

When you have finished your workshop you should make sure that you show the work that you have done to the laboratory supervisor so your participation can be marked. Be sure to check your participation is recorded before you leave your workshop session or you won't receive credit.

Question 1 - Building vectors

In this question we will look at code to build vectors.

Example: Building a vector

Definition

In this question we read a non-negative number *n* and then use a for-loop to build a vector, called *result*, containing the elements from 1 to n.

Testing

These are some example test cases :

```
n   result
0   []
1   [1]
2   [1 2]
5   [1 2 3 4 5]
```

(we would save these in a file called q1Example1.txt)

Coding

To start this workshop navigate to your workshop 3 folder in MATLAB and type

```
clear all
```

into the MATLAB console. This is a way to clear previous definitions that might get in the way of code working as required. In general, it is good to type this command at the beginning of a coding session and you can also add it as the first line of your MATLAB script.

One possible solution for the program is the script

```
clear all

% read in length of vector
n = input('enter vector length ');

% start with empty result vector
result = [];

% build the vector
for i = [1:n]
    % add a new vector element by concatenation
    result = [result i];
end

%display the result
disp(result)
```

(we would save this script in a file called q1Example1.m). Note the use of concatenation to add a new element to a vector in the line:

```
 result = [result i];
```

Another example: Building a vector of even numbers

Definition

In this question we read a non-negative number n and then use a for-loop to build a vector, called *result*, containing the first n even numbers (starting at the number 2).

Testing

These are some example test cases :

```
n   result
0   []
1   [2]
2   [2 4]
5   [2 4 6 8 10]
```

(we would save these in a file called q1Example2.txt)

Coding

One solution for this program is the script

```
% read in length of vector
n = input('enter vector length ');

% start with empty result vector
result = [];

% build the vector
for i = [1:n]
    % add a new vector element by concatenation
```

```
    result = [result i*2];
end

%display the result
disp(result)
```

(we would save this script in a file called q1Example2.m). Note, again the use of indexing to build a vector in the line:

```
   result = [result i*2];
```

Before progressing, ensure you are able to trace through the code and understand how it generates the expected vector.

Question 1a: **(write answers to this question)**

Definition

In this question we write code that reads in a non-negative number *n* and uses a for-loop to put the first n *odd* numbers into a vector called result.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
n      result
1     [ 1 ]
2     [ 1 3 ]
```

if you are using a text file save the file with the name q1a.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q1a.m that reads in a non negative number n and uses a for-loop to put the first n **odd** numbers into a vector called result.

Question 1b: **(write answers to this question)**

Definition

In this question we write code that reads in a non-negative number n and uses a for-loop to put the first n **square** numbers into a vector called result.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
n      result
1     [ 1 ]
```

```
3     [ 1 4 9]
```

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q1b.m that reads in a non-negative number n and uses a for-loop to put the first n square numbers into a vector called result.

end of question 1

Question 2 - Transforming vectors

In this question we will look at how to use for-loops to transform vectors. In all examples we assume that we are starting with a vector A that contains some numbers.

Example: adding one to each vector value:

Definition

Write a MATLAB script, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

in the script, use a for-loop to add one to each element of A.

Testing

These are some example test cases :

```
A(before).         A(afterwards)
[-2 4 9 -5 0 -1]   [-1 5 10 -4 1 0]
[ 7 8 9]           [ 8 9 10]
[-8]               [-7]
```

(we would save these in a file called q2Example1.txt if we were writing these)

Coding

One solution for this program is the script

```
% definition of A
A = [-2 4 9 -5 0 -1];

% loop to change A
for i = [1:length(A)]
    % add one to the current element
    A(i) = A(i)+1;
end

%display the new values in A
disp(A)
```

(we would save this script in a file called q2Example1.m):

Question 2a: **(write answers to this question)**

Definition

In this question we write code, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

and then uses a for-loop to change each element of A to its *absolute value*. **Hint**: Check the functions information to see if MATLAB has a builtin absolute value function you can use.

Testing

On paper, or in a text file, write three new tests for the above program As an example, three tests for this program might be.

```
A(before).       A(afterwards)
[-2 4 9 -5 0 -1] [2 4 9 5 0 1]
[ 7 8 9]         [ 7 8 9]
[-8]             [8]
```

if you are using a text file save the file with the name q2a.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q2a.m that uses a for-loop to change each element of a vector A to its *absolute value.*

end of question 2

Question 3 - Accumulating Vectors

In this question we will look at how to use for-loops to accumulate values in vectors. In all examples we assume that we are starting with a vector A that contains some numbers.

Example: summing vector values:

Definition

Write a MATLAB script, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

in the script, use a for-loop to calculate the sum of the elements of A.

Example Test Cases

These are some example test cases :

```
A.                sum
[-2 4 9 -5 0 -1]  5
[ 7 8 9]          24
[-8]              -8
```

(we would save these in a file called q3Example1.txt if we were writing these)

Example Program Code

One solution for this program is the script

```
% definition of A
A = [-2 4 9 -5 0 -1];

% initialise sum
sum = 0;

% loop to sum A
for i = 1:length(A)
    % add one sum
    sum = sum + A(i);
end

%display the sum
disp(sum);
```

(we would save this script in a file called q2Example1.m):

Queston 3a: **(write answers to this question)**

Definition

For this question we write code, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

in the script, use a for-loop to calculate the *mean* value of the values in A.

Testing

On paper, or in a text file, write three new tests for the above program As an example, three tests for this program might be.

```
A               Mean
[-2 4 9 -5 0 -1] 0.83
[ 7 8 9]         8.00
[-8]             -8.00
```

if you are using a text file save the file with the name q3a.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q3a.m that uses a for-loop to calculate the mean of a vector A.

Queston 3b: **(write answers to this question)**

Definition

For this question we write code, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

in the script, use a for-loop to calculate the *product* of the values in A. The product is the result of multiplying all of the numbers together.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
A(before).       A(afterwards)
[-2 4 9 -5 0 -1] 0
[ 7 8 9]         504
[-8 9]           -72
```

if you are using a text file save the file with the name q3b.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q3b.m that uses a for-loop to calculate the the product of the vector A.

Question 4 -More Vector Transformations

In this question we will look at how to use for-loops to transform values in one vector and copy them into another.

Example: selecting vector elements less than 3:

Definition

Write a MATLAB script, that, at the top of the program has a definition of a vector A such as:

```
A = [-2 4 9 -5 0 -1];
```

in the script, use a for-loop to produce a new vector B that contains the squares of the elements of A.

Testing

These are some example test cases :

```
A.                B
[-2 4 9 -5 0 -1]  [4 16 81 25 0 1]
[ 7 8 9]          [49 64 81]
[-8]              [64]
```

(we would save these in a file called q4Example1.txt if we were writing these)

Coding

One solution for this program is the script

```
% produce a new vector B that contains the squares of A
A = [ 1 3 7 2 8 5];
B = []; % start with empty B
```

```
% loop over A and build up B
for i = [1:length(A)]
    % append square of A(i) to B
    B = [B A(i)*A(i)];
end

%display B
disp(B);
```

The line responsible for concatenating A(i)*A(i) onto the end of B is:

```
 B = [B A(i)*A(i)];
```

Queston 4a: **(write answers to this question)**

Definition

For this question we write code script, that, at the top of the program has a definition of a vector A such as:

```
A=[-2 4 9 -5 2 0 2 -1];
```

in the script, use a for-loop to produce a new vector B that contains the negative of each element of A. Thus, after the script runs B should contain:

```
[2 -4 -9 5 -2 0 -2 1]
```

Testing

On paper, or in a text file, write three new tests for the above program As an example, three tests for this program might be.

```
A.                    B
[-2 4 9 -5 2 0 2 -1] [2 -4 -9 5 -2 0 -2 1]
[ 7 8 9]             [-7 -8 -9]
[2]                  [-2]
```

if you are using a text file save the file with the name q4a.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q4a.m implements the program defined in the definition above.

Question 4b: Calculating a running average (moving average) **(write answers to this question)**

Definition

In this question you will write code to calculate the running *average* of a vector of numbers stored in a vector A and store these averages in vector B.

Testing

On paper, or in a text file, write three new tests for the above program As an example, two tests for this program might be.

```
A             B
[ 1 2 3]      [ 1.0 1.5 2.0 ]
[ 3 5 2 4 1 ] [ 3.0 4.0 3.3 3.5 3.0 ]
```

if you are using a text file save the file with the name q3.txt

Coding

If you are sharing a computer, swap a new group member to the terminal. Write a MATLAB script called q4b.m that calculates the running averages as described above. **Hint**: Try to apply your knowledge of loops and vectors to write the code, but if you get stuck, you may find the **running_sum.m (https://myuni.adelaide.edu.au/courses/95033/files/15473287?wrap=1)** **(https://myuni.adelaide.edu.au/courses/95033/files/15473287/download?download_frd=1)** example as a good starting point (since it already calculates running sums).

When your code is working, define a vector A as:

```
A = rand(1,100);
```

You should find that the running average converges to a value near 0.5.

If you run:

```
close all;
plot(B);
```

you can see how this convergence occurs.

Finishing up.

Make sure you let a practical supervisor see your work for this session. If there is time work with your neighbour on the practice questions below.

If there is time..

If there is time in your session work with your neighbour to develop solutions to the following problems. Your answers to each should involve for-loops. Write down three test cases before writing each.

- Write a script that defines two vectors A and B and uses a for loop to add together corresponding elements of A and B to produce a new vector C.
- Write a script that defines a vector A and uses a for-loop to set all values in A except the last three values to zero.

- Write a script that initialises a vector A and creates a new vector B that contains the running sum of the elements of A. Thus if A contains the elements [1 2 3 4] then, after your script runs then B will contain the elements [1 3 6 10].

End of Workshop.