Computer Programming
Bachelor in Biomedical Engineering
Bachelor in Applied Mathematics and Computing
Course 2020 / 2021

Exercise Sheet 1 - SOLUTIONS
MATLAB Syntax

Content Table

Objectives

This collection of exercises is intended to help you start learning MATLAB. At the end of the session you will have practiced:

- Using MATLAB help
- Using MATLAB fundamental datatypes
- Using vectors and matrices

Help

MATLAB offers many different functions and commands of many different types. Before carrying out any task it is worth having a look at the MATLAB help function and checking which functions MATLAB may offer to help you. The simplest way to access help is to type help on the command line followed by the name of the command that you want to query.
See for example:

>> help clear

```
clear  Clear variables and functions from memory.
    clear removes all variables from the workspace.
```

>> help input

```
input  Prompt for user input.
    RESULT = input(PROMPT) displays the PROMPT string on the screen, waits
    for input from the keyboard, evaluates any expressions in the input,
    and returns the value in RESULT. To evaluate expressions, input accesses
    variables in the current workspace. If you press the return key without
    entering anything, input returns an empty matrix.

    STR = input(PROMPT,'s') returns the entered text as a MATLAB string,
    without evaluating expressions.

    To create a prompt that spans several lines, use '\n' to indicate each
    new line. To include a backslash ('\') in the prompt, use '\\'.

    Example:

       reply = input('Do you want more? Y/N [Y]:','s');
       if isempty(reply)
          reply = 'Y';
       end
```

Exercises: VARIABLES AND OPERATORS

Exercise 1

Execute the following commands in the command window, line by line

```
num=128
op1=1
op2=0
rdo=0
```

Now perform the following operations:

- Increment op1 by 1 and multiply it by 12
- Store the result of adding op1 to num in op2
- Store the result of dividing op2 by op1 in rdo
- Show the current values of op1, op2 and rdo on screen

SOLUTION

- Increment op1 by 1 and multiply it by 12
  `(op1+1)*12`
  If you want to store result in op1: `op1 = (op1+1)*12`

- Store the result of adding op1 to num in op2
  `op2 = op1 + num`

- Store the result of dividing op2 by op1 in rdo
  `rdo = op2 / op1`

- Show the current values of op1, op2 and rdo on screen
  ```
  op1
  op2
  rdo
  ```

Exercises: VECTORS AND MATRICES

Exercise 2

Create the following vector in the command window

```
x = [1 9 7 5 1 2]
```

Now try to anticipate what the following MATLAB commands will produce. Then check your answers by carrying them out in the command window.

- `x(6)`
- `x(2:3)`
- `x(1:7)`
- `x(3:end)`
- `min(x)`
- `max(x)`

Note: you are going to use two new functions *min, max* and *length*. To read more information about how they work you can use the command *help* followed by the name of the function

SOLUTION

- `x(6)`
  ```
  ans = 2
  ```
- `x(2:3)`
  ```
  ans = 9 7
  ```
- `x(1:7)`
  ```
  ??? Index exceeds matrix dimensions.
  ```
- `x(3:end)`
  ```
  ans =  7     5     1     2
  ```
- `min(x)`
  ```
  ans = 1
  ```
- `max(x)`
  ```
  ans = 9
  ```
- `length(x)`
  ```
  ans = 6
  ```

Exercise 3

Use the operator : to create the following vectors and store them in a variable named *vect*

- A vector containing all the numbers between 1 and 20
- A vector containing all the numbers between 20 and 1 in reverse order
- A vector containing all the odd numbers between 15 and 50
- A vector containing all the even numbers between 16 and 50

SOLUTION

- A vector containing all the numbers between 1 and 20
  ```
  vect = [1:20]
  ```
- A vector containing all the numbers between 20 and 1 in reverse order
  ```
  vect = [20:-1:1]
  ```
- A vector containing all the odd numbers between 15 and 50
  ```
  vect = [15:2:50]
  ```
- A vector containing all the even numbers between 16 and 50
  ```
  vect = [16:2:50]
  ```

Exercise 4

You can compare and perform operations with all the elements in a vector at once. To practice this introduce the following vectors

```
x= [2 4 7 8 9 0 2 5]
y= [1 7 0 9 1 5 1 9]
```

Perform the following operations and analyse the results

- `x > y`
- `y <= x`
- `x == y`
- `x | y`
- `x & y`
- `x & (~y)`

Note: remember that although MATLAB represents the boolean values TRUE and FALSE using 1 and 0, in practice it considers anything different from 0 as the value TRUE

SOLUTION

- `x > y`
  ```
  ans = 1     0     1     0     1     0     1     0
  ```
- `y <= x`
  ```
  ans = 1     0     1     0     1     0     1     0
  ```
- `x == y`

```
   ans = 0     0     0     0     0     0     0     0
```

- `x | y`

```
   ans = 1     1     1     1     1     1     1     1
```

- `x & y`

```
   ans = 1     1     0     1     1     0     1     1
```

- `x & (~y)`

```
   ans = 0     0     1     0     0     0     0     0
```

Exercise 5

Create the following variables

```
A = 1
B = 2
C = 3
D = 4
```

Now try to anticipate what the following MATLAB commands will produce. Then check your answers by carrying them out in the command line. If the command doesn't execute correctly, try to explain why.

- `[A B C]`
- `[C B A]`
- `[A B`
  `C D]`
- `[A B ; C D]`

SOLUTION

- `[A B C]`

```
   ans = 1     2     3
```

- `[C B A]`

```
   ans = 3     2     1
```

- `[A B`
  `C D]`

```
   ans = 1     2
         3     4
```

- `[A B ; C D]`

```
   ans = 1     2
         3     4
```

Exercise 6

- Introduce a 4x4 matrix
- Display the content of the whole matrix on screen
- Display the second row of the matrix on screen
- Display the third column of the matrix on screen
- Display the element (2,3) of the matrix on screen
- Display the elements between the elements (2,2) and (4,4) on screen

SOLUTION

- Introduce a 4x4 matrix
```
A = [1 3 5 6; 3 5 7 2; 1 3 9 19; 12 5 2 8]
```

- Display the content of the whole matrix on screen
```
A
```

- Display the second row of the matrix on screen
```
A(2,:)
```

- Display the third column of the matrix on screen
```
A(:,3)
```

- Display the element (2,3) of the matrix on screen
```
A(2,3)
```

- Display the elements between the elements (2,2) and (4,4) on screen
```
A(2:4,2:4)
```
Or
```
A(2:end,2:end)
```