

Arrays & Debugging

LAB 3

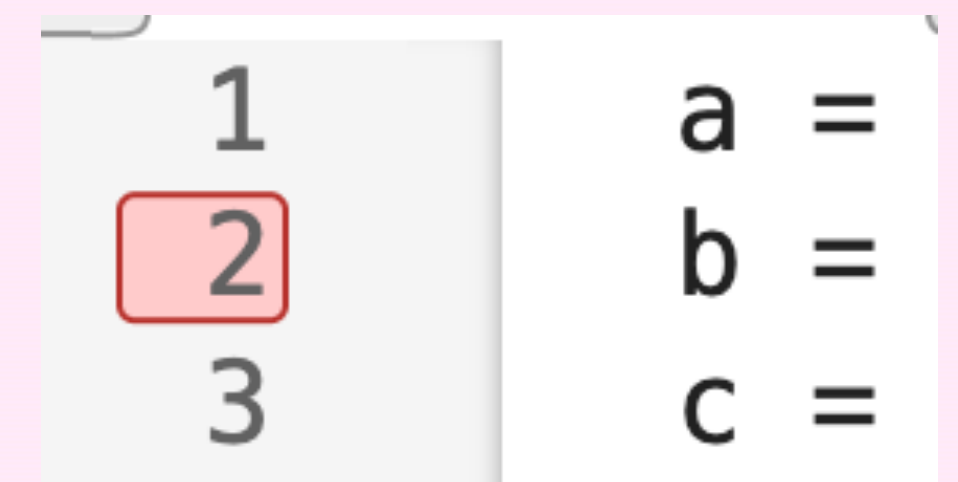
ENGG100 - Spring 2024

Lab 3 Objectives

- Creating 2D numerical and cell arrays with MATLAB
- Editing arrays using workspace and command window
- Understanding, debugging and resolving errors in MATLAB

Debugging

- When encountering errors in MATLAB, it's important to read the errors carefully to solve the problem
- Most common errors include:
 - Not including enough arguments or having too many arguments inside a function
 - Not defining variables properly, so MATLAB cannot recognize them
 - Not using proper brackets (incomplete code)
- With MATLAB, you can set "breakpoints" to execute one line at a time to find and debug a suspected error. You can do this by clicking on the line number to turn it red. Then click on "Run" or "Run Section" and MATLAB will execute all the commands before the breakpoint
- The command window will show K>> to remind you that you are working in the debugger

A screenshot of the MATLAB command window. On the left, there is a small icon with the letters 'fx' in a stylized font. To the right of the icon, the text 'K>>' is displayed in a black, monospaced font, indicating that the MATLAB debugger is active.A screenshot of the MATLAB editor interface. On the left, a vertical list of line numbers (1, 2, 3) is shown. The number '2' is highlighted with a red square, indicating a breakpoint has been set at that line. To the right of the line numbers, the code 'a =', 'b =', and 'c =' is visible on lines 1, 2, and 3 respectively.

TASK 1 - SCRIPT

- Create a new script with the below code, debug and fix the errors.
- Explain in comments how you fixed the error

```
a = [1 2 3];  
b = [1 2];  
c = a+b;  
disp(c);
```

- Hint: Array addition is between the corresponding elements. e.g. the value of 1 from a will be added to the value of 1 from b

RESULT

2	4	8
---	---	---

TASK 2 - SCRIPT

- Create a new script with the below code, debug and fix the errors.
- Explain in comments how you fixed the error

```
d=[0,pi/4,pi/2];
```

```
g = sin(d, d);
```

```
disp(g);
```

- Hint: We are trying to display d twice as a concatenated value

RESULT

0	0.7071	1.0000	0	0.7071	1.0000
---	--------	--------	---	--------	--------

Creating Numerical Arrays

- Numerical arrays can be created using square brackets, with values either being separated by a space or a comma. New rows can be created by separating values with a semicolon.
- `[1, 2, 3]` is not the same as `[1; 2; 3]` - the first is a 1x3 matrix while the second is a 3x1 array
 - 1x3 array \rightarrow `matrix = [1 2 3]` or `matrix = [1, 2, 3]`
 - 2x2 array \rightarrow `matrix = [2 3; 4 5]` or `matrix = [2, 3; 4, 5]`
- An array of zeros can be created using the `zeros` function
 - 5x1 array \rightarrow `matrix = zeros(5, 1)`
- Functions and arithmetic operations can be applied to all values in an array
 - `matrix + 1`
 - `cos(matrix)`
 - `matrix * 3`

TASK 3 - SCRIPT

- Use MATLAB help & documentation to figure out how to create an identity matrix
- Create an identity matrix of size 4x4 and multiply all the values by 6
- Your script should contain comments explaining all lines of code

Magic Square Matrix & Editing Array Values

- You can create an equal-sized matrix using the `magic(N)` function, which constructs it using integers from 1 to N^2 with equal rows, columns, and diagonal sums
- The workspace window allows you to view and edit the matrix real-time, while dynamically adjusting the column widths, and vertical window space
- The command `matrix(row, column) = value`, allows to change the value using the command window
- To double-check if the value has been changed, use the `disp(matrix)` command
- **Try it out:**
 1. Create a variable called `matrix` and use the `magic` function to create a matrix of size 3
 2. Change the value of (2,3) to 7 using the workspace
 3. Change the value of (3,1) to 1 using the command window

TASK 4 - COMMAND WINDOW

- Do this task in the **command window**
- Create a 6x6 `matrix` using the `magic()` command
- Update the value of $(4, 6) = 30$ using the command window
- Update the value of $(1, 3) = 12$ using the workspace browser
- Verify the matrix has been updated using the `disp()` function

In your lab report, make sure to include:

- Screenshot of your original magic matrix
- Screenshot of the value changed using command window
- Screenshot of the value changed using workspace browser
- Screenshot of your final updated magic matrix after using the `disp()` function

Creating Cell Arrays

- Cell arrays can be used instead of numerical arrays when wanting to store data that is not solely comprised of numbers
- Cell arrays can be created using curly braces, with values separated by a space or a comma - e.g:

```
my_array = { 'age', 12 }
```

- New rows can be created using semicolons - e.g:

```
my_array = { 'student id', 1234567; 'semester', 'spring' }
```

- Try the below in the command window:
 - `a = { 'This is an array of', 3, 'values' }`
 - Change a to be a 3x1 array
- An empty cell array can be created using the `cell()` function
 - 2x3 array —> `my_array = cell(2,3)`

TASK 5 - SCRIPT

- This task must be done in a script. Make sure to use `clear` and `clc` before you start
- For this task, you will create a cell array to create a weekly timetable template

```
timetable = { '', 'Monday', 'Tuesday' ...;  
'8:30-9:30', '', '', ...;  
'9:30-10:30', '', '', ...;  
...  
'17:30-18:30', '', '', ... }
```

	1	2	3	4	5	6
1		Monday	Tuesday	Wednesday	Thursday	Friday
2	8:30-9:30					
3	9:30-10:30					
4	10:30-11:30					
5	11:30-12:30					
6	12:30-13:30					
7	13:30-14:30					
8	14:30-15:30					
9	15:30-16:30					
10	16:30-17:30					
11	17:30-18:30					

- Fill in the timetable template with your own class schedule using either command window or workspace
- In your lab report, add a screenshot of the script, along with timetable template from workspace or command window

TASK 6 (BONUS) - SCRIPT

- Write a script that asks the user for the coordinates of 2 points $A = (x_1, y_1, z_1)$ and $B = (x_2, y_2, z_2)$ in a 3D space and returns the distance between the points using the formula below

$$AB = \sqrt{[x_2 - x_1]^2 + [y_2 - y_1]^2 + [z_2 - z_1]^2}$$

- Make sure you add comments to each line in the script, explaining what each line does
- Start by creating the below variables
- `point_A` = use the input function to get 3 coordinates from the user and store it in an array
- `point_B` = use the input function to get 3 coordinates from the user and store it in an array
- Assign values to `x1`, `y1`, `z1`, `x2`, `y2`, `z2` from the arrays
- `distance` = use the values assigned above to calculate the distance
- `disp()` = display the final value to the user
- Test the script with the coordinates $A = [2, 3, 4]$ and $B = [5, 6, 7]$ and verify you get the answer as **5.1962**