MATLAB

MATLAB Working Environment

In your MATLAB environment you have 3 major parts:

- Current Folder
- Command Window
- Workspace

You can see them as soon as you open MATLAB and can freely adjust the orientation and size of each section

1. The current folder should be set to a folder in your computer that contains the data you want to work with.

All files and subfolders will be visible under the Current Folder area once you select the folder you want to work in.

2. The Command Window is where you can write and execute your code instantenously.

3. The Workspace is where all of the variables you have created will be displayed and collected.

Variables and Data Types

Using MATLAB as a calculator

You can use Matlab as a calculator by creating variables in the Command Window

Lets start with numerical variables...

In order to create a variable called "age" and assign the value of "your own" age write the following code to the Command Window

```
age = 25 %your own age
```

```
age = 25
```

After typing this code and pressing "Enter" you will see the variable "a" in your Workspace and can view the value by double clicking the name.

We can assign values to a variable. We can then use these variables in calculations.

Assing another variable containing the age of the person sitting next to you...

```
age_next = 20 %age of the person next to you
```

```
age_next = 20
```

Try using the mathematical operators to do simple calculations using your variable

```
age_sum = age + age_next
```

```
age_sum = 45
```

```
age_rat = age/age_next
```

```
age_rat = 1.2500
```

```
age_mult = age*age_next
```

```
age_mult = 500
```

```
age_diff = age-age_next
```

```
age_diff = 5
```

Rules for variable names:age1

Variable names **must begin** with a **letter**, and **can contain numbers or underscores.**

If you do not want the variable to printed on the Command Window you can suppress the output by adding a semicolon ";" at the end

```
suppressed_output = 13; % The semicolon will supress the output being printed
```

We can assign more than one type of data to a variable. It doesn't always have to be one number. We do it in the same way we assign numbers. Here are all the data types available to us.

- Vectors and Matrices: "[ ]"
- Cells and Structures: "{ }", ". or struct"
- Tables: "table"
- Characters and Strings: ' ', " "

Vectors and Matrices

A vector or matrix are both groups of numbers (or elements: in MATLAB you can have sets of other types as well)

Lets create a Vector called 'ages' and a Matrix called 'math_with_ages'

```
ages = [age, age_next] % or without the commas
```

```
ages = 1×2
    25    20
```

```
math_with_ages = [age age_next; age_sum age_rat; age_mult age_diff]
```

```
math_with_ages = 3×2
   25.0000   20.0000
   45.0000    1.2500
  500.0000    5.0000
```

The square brackets "[ ]" allow you to create vectors (arrays) or matrices

using a space " " or comma "," between elements allow you to create new columns

using a semicolon ";" between elements allow you to create new rows (Experiment!!!)

- Bonus: The transpose operator ' (apostrophe)

Characters and Strings

Characters and strings are basically vectors of special elements which do not support mathematical operations.

They are mostly useful to represent non numerical elements.

Anything placed between two apostrophes ' ' is considered a character and anything placed between quatation marks " " is considered a string.

Lets create a character array variable called name and put your name as the value.

```
name = "Doruk"
```

```
name =
"Doruk"
```

And create a string variable called next_name and put the name of the person next to you as the value.

```
name_next = 'Darcy'
```

```
name_next =
'Darcy'
```

Cells and Structures

```
% Lets create a Cell called 'Name_Age'
Name_Age = {name age; name_next age_next}
```

```
Name_Age = 2×2 cell array
    {["Doruk"]}    {[25]}
    {'Darcy'  }    {[20]}
```

Unlike vectors and matrices, cells can hold different element types within: for example the cell named "Name_Age" has both numbers and letter characters

Cells can also hold arrays or matrices as elements:

```
Name_Age_Num = { [name name_next]; ages; [7 11]} % contains the names array, ages array and fav
```

```
Name_Age_Num = 3×1 cell array
    {1×2 string}
    {1×2 double}
    {1×2 double}
```

A structure is like a cell as it can also contain different element types.

But it has fields instead of columns and each field is accessed using the "dot" notation as structure_name.field_name

To create our new structure "ID" with a field we are going to call "Names"

```
% And a structure called
ID.Names = [name name_next] %ID is a struct element with only one field called Names
```

```
ID = struct with fields:
    Names: ["Doruk"    "Darcy"]
     Ages: [25 20]
     More: {2×2 cell}
```

```
ID.Ages = ages %adding a second field
```

```
ID = struct with fields:
    Names: ["Doruk"    "Darcy"]
     Ages: [25 20]
     More: {2×2 cell}
```

```
ID.More = Name_Age % we have named the 3rd field "bla" as we can give any name
```

```
ID = struct with fields:
    Names: ["Doruk"    "Darcy"]
     Ages: [25 20]
     More: {2×2 cell}
```

```
% this final field holds the cell we have created previously called "c"
```

Another way to create Structures is to use the "struct()" function

For more information on this function type "help struct" in your command window

```
% another way to create a structure is:
S = struct('field1', 1, 'field2', 'c', 'bla', Name_Age) %3rd field is named bla to show that ar
```

```
S = 2×2 struct array with fields:
    field1
    field2
    bla
```

Tables

Tables are similar to Structures since they also have an ordered structure to store data with certain titles. They are very useful for holding experimental data.

```
Height = [1.68; 1.99; 1.30];
Weight = [61; 95; 45]
```

```
Weight = 3×1
    61
    95
    45
```

```
Age = [43; 40; 13]
```

```
Age = 3×1
    43
    40
    13
```

```
T1 = table(Height, Weight, Age)% A table is created for each persons values from the given data
```

T1 = 3×3 table

	Height	Weight	Age
1	1.6800	61	43
2	1.9900	95	40
3	1.3000	45	13

```
% Lets create a different table
Family1 = {'Height', 'Weight', 'Age'; 1.68 61 43; 1.99 95 40; 1.30 45 13};
Family2 = {'Height', 'Weight', 'Age'; 1.74 80 27; 1.70 76 34; 2.10 100 17};
T2 = table(Family1, Family2)
```

T2 = 4×2 table

	Family1			Family2		
1	'Height'	'Weight'	'Age'	'Height'	'Weight'	'Age'
2	1.6800	61	43	1.7400	80	27
3	1.9900	95	40	1.7000	76	34
4	1.3000	45	13	2.1000	100	17

Using Scripts

Scripts are essential for organising and writing "neat" blocks of code, making changes and locating your errors.

You can create a new script by:

- pressing the "New Script" button on the top left of the MATLAB screen in the Home Tab
- or by pressing the "New" dropdown button next to it and selecting "Script"

We will be doing the indexing part of the workshop in scripts.

Indexing

Different data types are indexed using different notations. Indexing is a vital part of using MATLAB and one of the greatest advantages.

Vector and Matrix index

Vectors and Matrices are indexed using regular brackets "()".

For any 2D array the first element of the inex is the row number and the second is the column number: (row, column).

```
%Indexing the vector ages that was created earlier to obtain the second element
ages(2) %Because a vector is 1 dimesional single index is enough to adress the element you want
```

```
ans = 20
```

```
%Indexing the matrix math_with_ages that was created earlier to obtain the element in the secon
math_with_ages(2,1)
```

```
ans = 45
```

One important notation is the colon ":" operator which is used to define a range.

```
1:10 % will give you a vector containing numbers from 1 to 10 with increments of 1
```

```
ans = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```
1:2:10 % will give you a vector containing numbers from 1 to 10 with increments of 2
```

```
ans = 1×5
     1     3     5     7     9
```

When the colon operator is used in indexing, a range of elements with different increments can be selected.

```
math_with_ages(2,1:2) % this notation points to the second row, columns 1 and 2
```

```
ans = 1×2
   45.0000    1.2500
```

```
%similarly
math_with_ages(1:2:end,1:2) % will give columns 1 and 2 for the first and last rows (noted by e
```

```
ans = 2×2
    25    20
   500     5
```

```
%Whereas
math_with_ages(:,1) % will give the first column of all rows (all rows being noted by just putt
```

```
ans = 3×1
    25
    45
   500
```

Since defining a range is actually the same as inputting an array of numbers as indexes, any array can also be indexed using another array of index values.

A note remember is that any index should actually be available (i.e. you cant access the 5th row of an arra with 3 rows)

```
% Replicating the indexing that was achieved by defining ranges using arrays
math_with_ages(2,[1 2])
```

```
ans = 1×2
   45.0000    1.2500
```

```
math_with_ages([1 3],[1 2])
```

```
ans = 2×2
    25    20
   500     5
```

```
math_with_ages([1 2 3],1)
```

```
ans = 3×1
    25
    45
   500
```

```
% Check if the results are the same
```

Logic Statements in MATLAB

Another important data type to remeber are logic values.

They can either be true (1) or false (0).

Logic values can be used to make statements and write codes that make decisions depending on inputs, results of calculations, properties of data, etc.

For more on logic statements and the uses of logic values see the making decisions sheet included. "MakingChoices.mlx"

Functions in MATLAB-A Simple Intro

Functions in MATLAB are written in scripts as well.

They are specialised scripts which are declared to be functions by adding the word function in the beginning of a script.

The characteristic properties of a function are:

- Function Name-needs to be unique and has to be saved under a file with the same name
- Function Inputs-the values that a function takes in from a user in order to carry out its purpose
- Function Outputs- the resulting values that are returned after the function has complerted its purpose and the desired outcome is found.

For an example template function please see the template function file included. "normalize_temp.m"

To find the code of any other function or what it does you can:

- type: help "function name" (e.g. help mean), to the command window and a description will be shown as the output
- right click on a function in any script and select "open 'functionname'" and the code for the function will open in the editor

- or google the functions name+matlab to see numerous descriptions, examples and tutorials

I hope this workshop and summary notes will be helpful in solving your MATLAB related issues. Thanks for your interest and I hope to meet with you in our future events.