Unit 3. Matlab Syntax

3.1 Variables
3.2 Expressions
3.3 Fundamental data types
3.3 Operators
3.4.Screen output, input and comments

Syntax

- The **syntax** of a language defines how to use keywords, operators, and variables to build and evaluate expressions.
- In this first part of the **Matlab Language syntax** we specify how to write the following elements:
  - Variables
  - Expressions
  - Operators
  - Data types

Variables

- Some possible definitions:
  - “A symbol or name that stands for a value”
  - “A variable is a container which holds values”
  - “A variable is the name for a place in the computer's memory where you store some data.”

- A variable is a symbolic name given to an unknown data that permits the name to be used independently of the information that represents
  - Variables are associated with data storage locations
  - Values of a variable are normally changed during the course of program execution

Variables

- In MATLAB variables can be used:
  - In the command window
  - Within the code of a program
  - As parameters of functions

- We can **assign a value** of a variable, **retrieve its value** and **operate** with its value

- A variable is made of
  - Identifier/Name: list of characters used to reference the variable
  - Type: states the kind of values that will be stored in the variable
  - Value: its current data

Variables: naming a variable

- Naming variables
  - The names are case sensitive (*myvariable* and *myVariable* are different variables)
  - The names should start by a letter followed by any combination of letters, digits and underscores
  - Avoid using too long variable names
  - Never use names of existing functions or MATLAB keywords (*break*, *case*, etc..). You can verify by using the function *'isvarname'*

Always use meaningful names

Variables: assignment/creation statement

- It is used to set value to a variable.

variable = expression

- Examples:

  guests = 20

  vocal = 'a'

  amount = 240.78 + 5

Note that this **equal sign represents an assignment** and not an arithmetic equality

You can modify the value of a variable as many times as you want

Variables: retrieving its value

- The current value of a variable can be obtained writing the name of the variable in the command window. For example:

```
>> guest
    guest = 20

>> vocal
    vocal = a

>> amount
    amount = 245.78
```

Variables: retrieving its value

- As part of an expression:
  - An expression is a construction composed by variables, values, operators and function calls
  - MATLAB evaluates an expression an returns a value
  - Examples:

```
>> guest * 5
    ans = 100
>> (guest - 2) * 20
    ans = 360
>> cans = (guest * 3)
    cans = 60
```

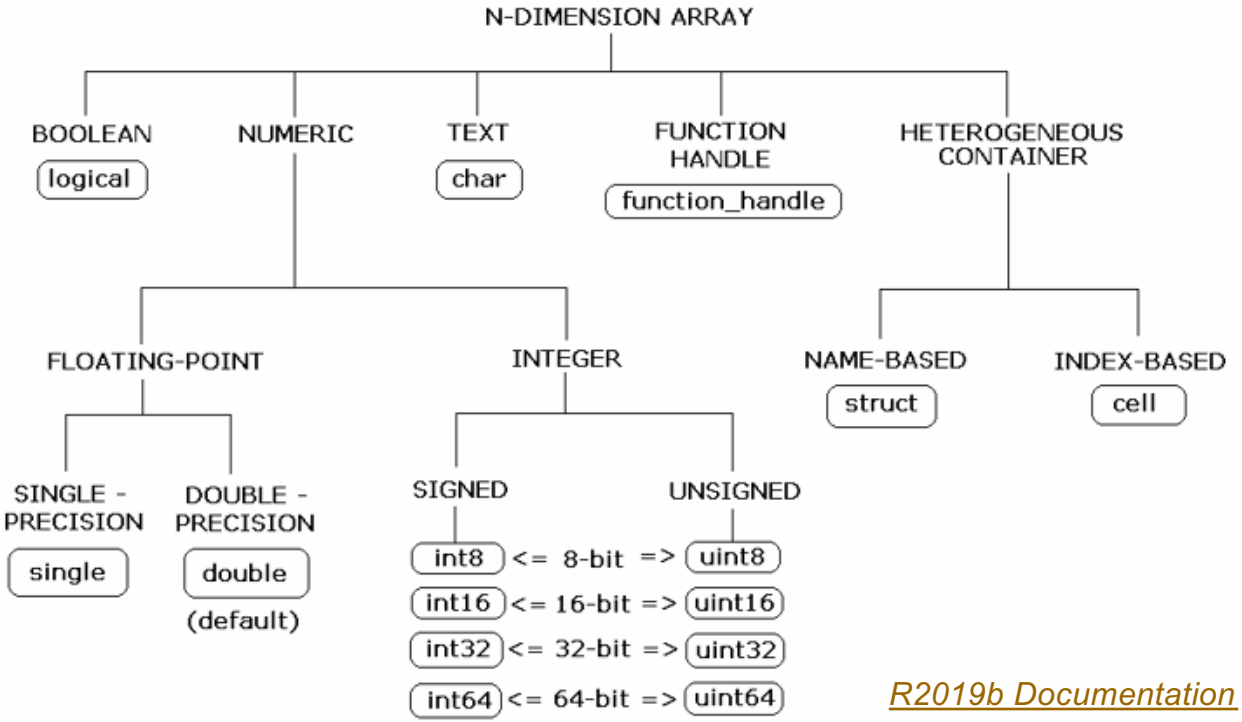
MATLAB evaluates the expressions from left to right . If the expression has not been explicitly assigned to a variable MATLAB automatically stores the result in the special variable *ans.*

Variables: type of variable

- A **type** **restricts the values** that a variable can include, **restricts the operation** supported by these values and **determines the meaning** of the operation.

- Matlab includes two categories of data types:
  - Fundamental data types: integers, chars, booleans..
  - User-defined types (MATLAB interface to java, *not used in this course)*

Fundamental data types



R2019b Documentation → *MATLAB*

- There are 15 fundamental data types in MATLAB (lowercase in the diagram)

Fundamental data types

- Boolean data types:
  - **logical** : Represents a logical TRUE or FALSE state using the numbers 0 and 1. 0 represents FALSE and 1 represents TRUE
- Integers:
  - **uint8, uint16, uint32, uint 64:** Unsigned integers. Size of 8,16, 32 and 64 bits respectively.

    Ej: intmin('uint8') intmax('uint8')

    ans = 0 ans = 255
  - **int8, int16, int32, int 64:** Signed integers. Size of 8,16, 32 and 64 bits respectively.

    Ej: intmin('int8') intmax('int8')

    ans = -128 ans = 127
- Floating point numbers
  - **single**: Floating point numbers of 32 bits
  - **double**: Floating point numbers of 64 bits

Fundamental data types

- **BY DEFAULT MATLAB STORES NUMERIC DATA AS DOUBLE.**

  A = 56 - The type of A is double

  To store the number as a different type you must specify it

  A = int8(56) - The type of A is int8
  A = single(3.67) - The type of A is single

Fundamental data types

- Character data types:
  - **char** : Characters. Size16-bits. Unicode.
  - TO SPECIFY THAT SOMETHING IS A CHAR YOU SHOULD PUT THE CHARACTER WITHIN **SINGLE QUOTATION MARKS**
  - Example: var = ‘T’
  - Special type of array(vector) is a character string, it is a text surrounded by single quotes. Example: str = ‘Hello’

Fundamental data types

- Cells: Array of indexed cells, each capable of storing an array of a different dimension and data type.

  A = { ‘Hello’ , 0.23, [0 1 2 3]}

- Structures: provide the means to store hierarchical data together in a single entitity by asociating named fields to different information.

  s = struct(‘a’ , ‘Hello’ , ‘b’ , ‘0.23’ , ‘c’ , [0 1 2 3]);

s.a	s.b	s.c
ans = ‘Hello’	ans = 0.23	ans = 0 1 2 3

Variables: types

- In a strongly-typed programming language the datatype of the variable is defined as soon as the variable is declared.

- **Matlab is NOT strongly typed**. Variables don't need to be declared prior to use.
  - When MATLAB encounters a new variable name, it automatically creates the variable and assign a type based on the type of data is going to store.
  - When MATLAB encounters an assignment of an existing variable, the value and type that the variable had before the assignment is lost.

Variables: declaration (creation)

- Example:

  >> X = 50

  >> Y = ‘b’

  >> X = -23.4

  >> X = ‘c’

Variables: declaration (creation)

- Example:

>> X = 50	MATLAB creates a variable X of type Double and stores the value 50
>> Y = ‘b’	MATLAB creates a variable Y of type Char and stores the character b
>> X = -23.4	MATLAB stores the value -23.4 in the variable X
>> X = ‘c’	MATLAB modifies the type of the variable X. Now the variable has the type Char and stores the character ‘c’

Variables: declaration (creation)

- Be careful. You cannot operate with variables for which you haven't specify a value yet.

- Example:

```
>> A = 50

>> B = A * 2

>> C = A + D
```

What would be the answer from MATLAB to these commands?

Variables: declaration (creation)

- Be careful. You cannot operate with variables for which you haven't specify a value yet.
- Example:

```
>> A = 50
A = 50
>> B = A * 2
B = 100
>> C = A + D
Error. Undefined function
or variable 'D'.
```

MATLAB computes the value of B by replacing A in the expression for its current value (50)

The variable D has not been created previously. MATLAB cannot replace it by any value.. therefore you get an error!

Operators

- They are characterized by:
  - Numbers of operands (unary, binary, or ternary)
  - The type of operands (i.e. numeric or boolean)
  - The type of generated result

Aritmetic Operators

- Aritmetic operators when working with variables with one single value (no matrices)
  - + Addition
  - - Subtraction
  - * Multiplication
  - / Division
  - ^ Power

Relational Operators

- Relational Operators
  - < Less than
  - <= Less than or equal to
  - > Greater than
  - >= Greater than or equal to
  - == Equal to
  - ~= Not equal to

- Example:
  A = 7
  B = 10

A > B	A < B	A == B
0	1	0

- Relational operators **compare values**

- The result is a boolean value:
  - 0 when false
  - 1 when true

Logical Operators

- Logical operators:
  - & and
  - | or
  - ~ not
  - xor exclusive or

- For Matlab the 0 value corresponds to the logical value **False**, and any value different from 0 corresponds to **True**

A	B	A&B	A|B	xor(A,B)	~A
0	0				
0	1				
1	0				
1	1				

Logical Operators

- Logical operators:
  - & and
  - | or
  - ~ not
  - xor exclusive or

- For Matlab the 0 value corresponds to the logical value **False**, and any value different from 0 corresponds to **True**

A	B	A&B	A|B	xor(A,B)	~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Logical Operators

- Short-circuit operators
  - && and
  - || or
- Example:
  A && B

  if A equals zero returns zero
  if A is not equals zero it evaluates B
  and returns the correspondent value

- They work exactly in the same way as & and | , but they evaluate their second operand only when the result is not fully determined by the first operand

In summary: they are equivalent to the operators & and |

Order of Operations

- Associativity ( )
- Transpose, Power: .‘,.^, ‘,^
- Logical negation: ~
- Multiplication, division: *, /, \, .*, ./, .\
- Addition, subtraction: +, -
- Colon: :
- Less, greater, equal: < > <= >= == ~=
- Element wise And: &
- Element wise Or: |
- Short circuit And: &&
- Short circuit Or: ||

Example:
2+3*5 == 17
(2+3)*5 == 25

It is a good idea to use parentheses to explicitly specify the intended precedence

My first Matlab program

- Exercise: We want to create a program to automatically obtain the shopping list for our party

  - *Number of bags of ice cubes will be obtained by dividing the number of guests by 4*
  - *Number of pizzas will be obtained by dividing the number of guests by 3*
  - *Number of cans of coke will be obtained by multiplying the number of guests by 4*

My first Matlab program

- Exercise: We want to create a program to automatically obtain the shopping list for our party
  - *Solution (using what we know so far…)*
    - *We are going to store the number of guest in a variable*
    - *Then we are going to perform operations with the variable to obtain the number of ice bags, cans of coke and pizzas*

My first MATLAB program

- **Creating the source file:**
  - The source should include the MATLAB language code.
  - A text editor can be used to create and edit the source files.
  - The extension of the file should be .m

- **Running the program:**
  - Type the name of the file in the command window.
  - The program should be placed in the current directory or in any directory of the variable *path*

Remember: You do not need to explicitely compile the program. Since MATLAB is an interpreted language the compilation is carried out automatically everytime you execute the program.

My first Matlab program

- Solution:

File name: shoppingList.m

```
guests = 20
guests / 4
guests / 3
guests * 4
```

Execution of the program
Command line:

```
>> run shoppingList
guest = 20
ans = 5
ans = 6.6667
ans = 80
```

My first Matlab program

- Solution:

File name: shoppingList.m

```
guests = 20
guests / 4
guests / 3
guests * 4
```

Execution of the program
Command line:

```
>> run shoppingList
guest = 20
ans = 5
ans = 6.6667
ans = 80
```

It's ok …. but we have to modify the program each time we want to modify the number of guests

User keyboard input

- The command for asking the user to introduce some data via the keyboard during the execution is input.
  - For numerical inputs (you want the user to introduce a number)

variableName = input('*any sentence*')

  - For character inputs (you want the user to introduce a character)

variableName = input('*any sentence* ', 's')

Remember this. It is a very common mistake for begginners not to put the , 's' when working with text entries. If you don't include it your program will not work!

My first Matlab program

- Solution:

File name: shoppingList.m

```
guests = input('Introduce the number of guests: ')
guests / 4
guests / 3
guests * 4
```

EXECUTION

Command line:

```
>> run shoppingList
Introduce the number of guests: 20
ans = 5
ans = 6.6667
ans = 80
```

Exercise

- Exercise: Write a program named *converter* for changing euros to pounds. The exchange rate is: 1€ = 0.799£

  - Example of execution:

    Introduce a quantity: 5

    ans = 3.9950

Exercise

- Solution:

FILE: converter.m

```
euros = input('Introduce a quantity: ')
euros * 0.799
```

Exercise

- Exercise: Modify the *converter* program so it also asks the user to introduce the exchange rate
  - Example of execution:

    Introduce a quantity: 5

    Introduce the change rate: 0.799

    ans = 3.9950

Exercise

- Solution:

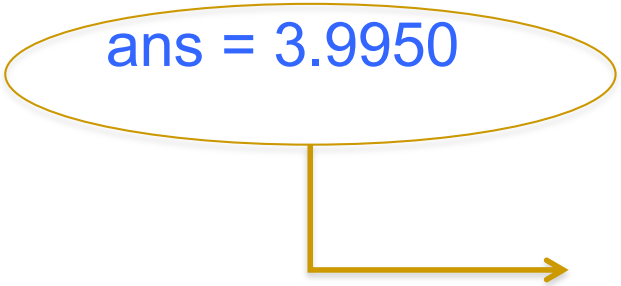
FILE: converter.m

```
euros = input('Introduce a quantity: ')
change = input('Introduce the euros: ')
euros * change
```

Exercise

- Example of execution:

```
Introduce a quantity: 5
Introduce the change rate: 0.799
ans = 3.9950
```



Not an elegant way of displaying the result….
… the next week we will learn how to do it better

Exercise

- Exercise: Write a program which asks the user to introduce two numbers and returns the sum of the two values
  - Example of execution:

    Introduce a number: 8

    Introduce another number: 2

    ans = 10

Exercise

- Solution:

```
number1 = input('Introduce a number: ')
number2 = input('Introduce another number: ')
number1 + number2
```

Exercise

- Exercise: Modify the previous program so it asks two numbers and prints their sum. Next asks the user to introduce another number and divides the previous sum by it
  - Example of execution:

    Introduce a number: 8

    Introduce another number: 2

    ans =10

    Introduce another number: 2

    ans = 5

Exercise

- Solution:

FILE: division.m

```
number1 = input('Introduce a number: ')
number2 = input('Introduce another number: ')
mySum = (number1 + number2)
number3 = input('Introduce another number: ')
mySum / number3
```

Exercises

1. Write a program that asks the user to introduce the coordinates *x, y* of two points and computes their distance
   - Remember the operator for power is ^
   - To compute the square root use the MATLAB funcion ***sqrt(X)***
2. Write a program that asks the user to introduce an student's marks in five exams. The program should:
   - Compute the average mark

Solutions

- 1.- Solution

```
x1 = input('Introduce the x coordinate of the first point: ');
y1 = input('Introduce the y coordinate of the first point: ');
x2 = input('Introduce the x coordinate of the second point: ');
y2 = input('Introduce the y coordinate of the second point: ');
distance = sqrt((x1 - x2)^2 +(y1 - y2)^2)
```

Solutions

- 2.- Solution

```
m1 = input('Introduce the first mark: ');
m2 = input('Introduce the second mark: ');
m3 = input('Introduce the third mark: ');
m4 = input('Introduce the fourth mark: ');
m5 = input('Introduce the fifth mark: ');
average = (m1+m2+m3+m4+m5) / 5
```