

Computer Science  
FYS

**MATLAB**

Essential MATLAB for Scientists

Dr. Delphine Syvilay  
Dr. Madhurima Panja

# Why computer science?

## What is AI?

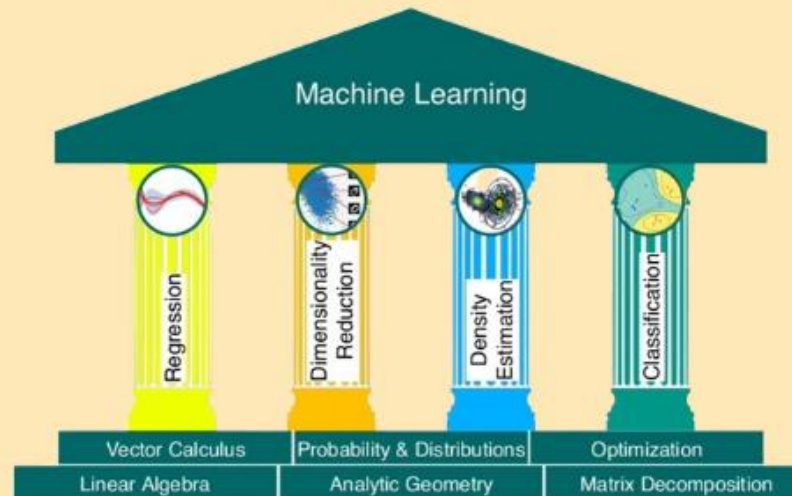
Artificial intelligence, or AI, is a simulation of intelligent human behavior.

It's a computer or system designed to perceive its environment, understand its behaviors, and take action.

Consider self-driving cars: AI-driven systems like these integrate AI algorithms, such as machine learning and deep learning, into complex environments that enable automation.

## Key Components to an AI Workflow

## Pillars of ML



## Softwares

Matlab is used to teach introductory mathematics such as calculus and algebraic computations...

Both Python and R can be used to make decisions involving big data....



# Objectives of this course

- The objectives of this course are to enable you to use some simple MATLAB commands from the Command Window and to examine various MATLAB desktop and editing features.
- Learn the exact rules for writing MATLAB statements.
- Develop a logical plan of attack for solving particular problems.
- Create your own program



# During the semester

- 10 CM (2h)

Chap 1: Introduction to algorithm

Chap 2: Introduction to Matlab

Chap 3: Fundamentals (if decision, for loop)

Chap 4: Functions

Chap 5: Matrices

- 5 TD (2h)

TD1: Exercises on Chap 3

TD2: Exercises on Chap 4

TD3: Exercises on Chap 5

TD4: Mini project

TD5: Mini project

⇒ Bring your own laptop for each session (CM & TD)

# Evaluation

- 1 mid-term written exam (30%) 10/10/23
- 1 oral presentation (mini-project 20%)
- 1 final written exam (50%)

Computer Science  
FYS

**MATLAB**

Essential MATLAB for Scientists

Chap 1 : Introduction to algorithm

Dr. Delphine Syvilay  
Dr. Madhurima Panja



# Concept and definition

What is an algorithm?

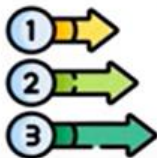
Definition:

An algorithm is a finite set of instructions that accomplishes a particular task, for solving a problem

Algorithms are easy to understand.



Algorithm is a step-by-step solution to a problem.



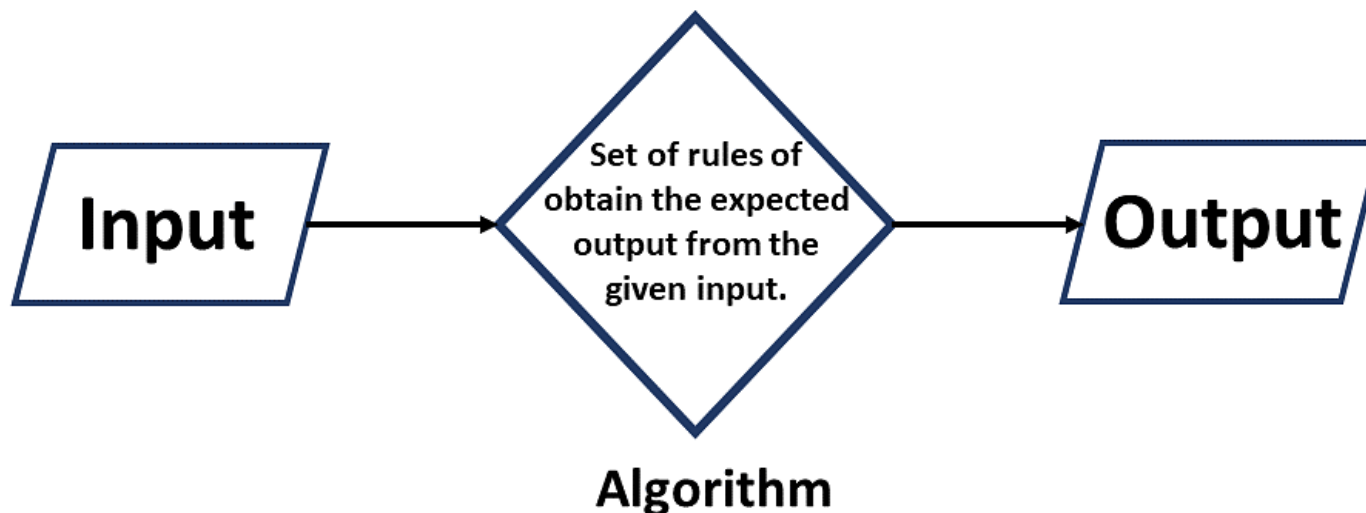
Algorithm divides problem into smaller steps



# Characteristic of algorithm

Criteria:

- Input: zero or more quantities that are externally supplied
- Output: at least one quantity is produced
- Effectiveness: instruction is basic enough to be carried out
- Finiteness: terminate after a finite number of steps
- Definiteness: clear and unambiguous





# Characteristic of algorithm

An algorithm has certain characteristics:

- A complex system may be divided into smaller units called modules. Modularity enhances design clarity (knowledge ordering)

# Characteristic of algorithm

An algorithm has certain characteristics:

- A complex system may be divided into smaller units called modules. Modularity enhances design clarity (knowledge ordering)
- Algorithms are generally created independent of underlying languages. An algorithm can be implemented in more than one programming language



# Characteristic of algorithm

An algorithm has certain characteristics:

- A complex system may be divided into smaller units called modules. Modularity enhances design clarity (knowledge ordering)
- Algorithms are generally created independent of underlying languages. An algorithm can be implemented in more than one programming language
- Each instruction should be unique and concise



# Characteristic of algorithm

An algorithm has certain characteristics:

- A complex system may be divided into smaller units called modules. Modularity enhances design clarity (knowledge ordering)
- Algorithms are generally created independent of underlying languages. An algorithm can be implemented in more than one programming language
- Each instruction should be unique and concise
- Repetition of same task should be avoided

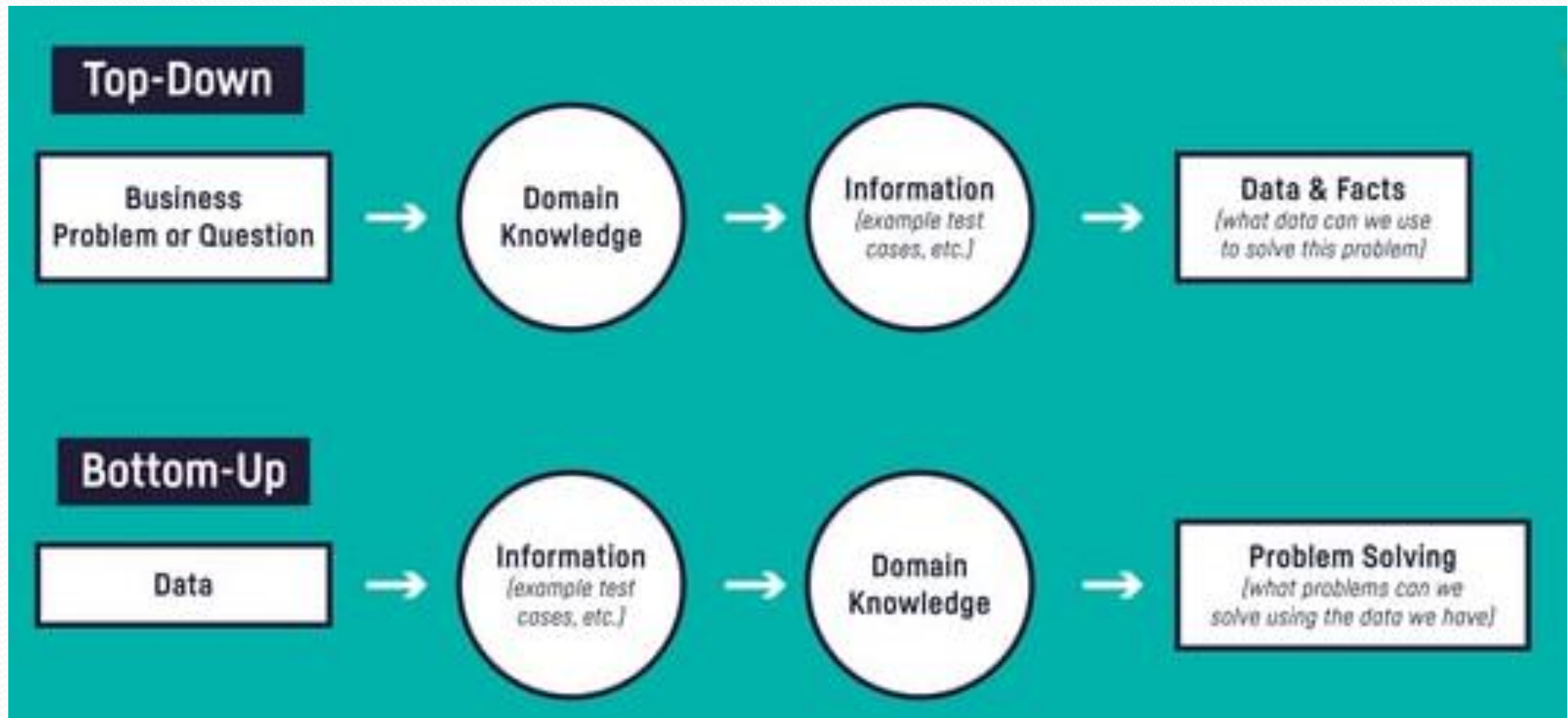
# Characteristic of algorithm

An algorithm has certain characteristics:

- A complex system may be divided into smaller units called modules. Modularity enhances design clarity (knowledge ordering)
- Algorithms are generally created independent of underlying languages. An algorithm can be implemented in more than one programming language
- Each instruction should be unique and concise
- Repetition of same task should be avoided
- The result should be available to the user after algorithm terminates

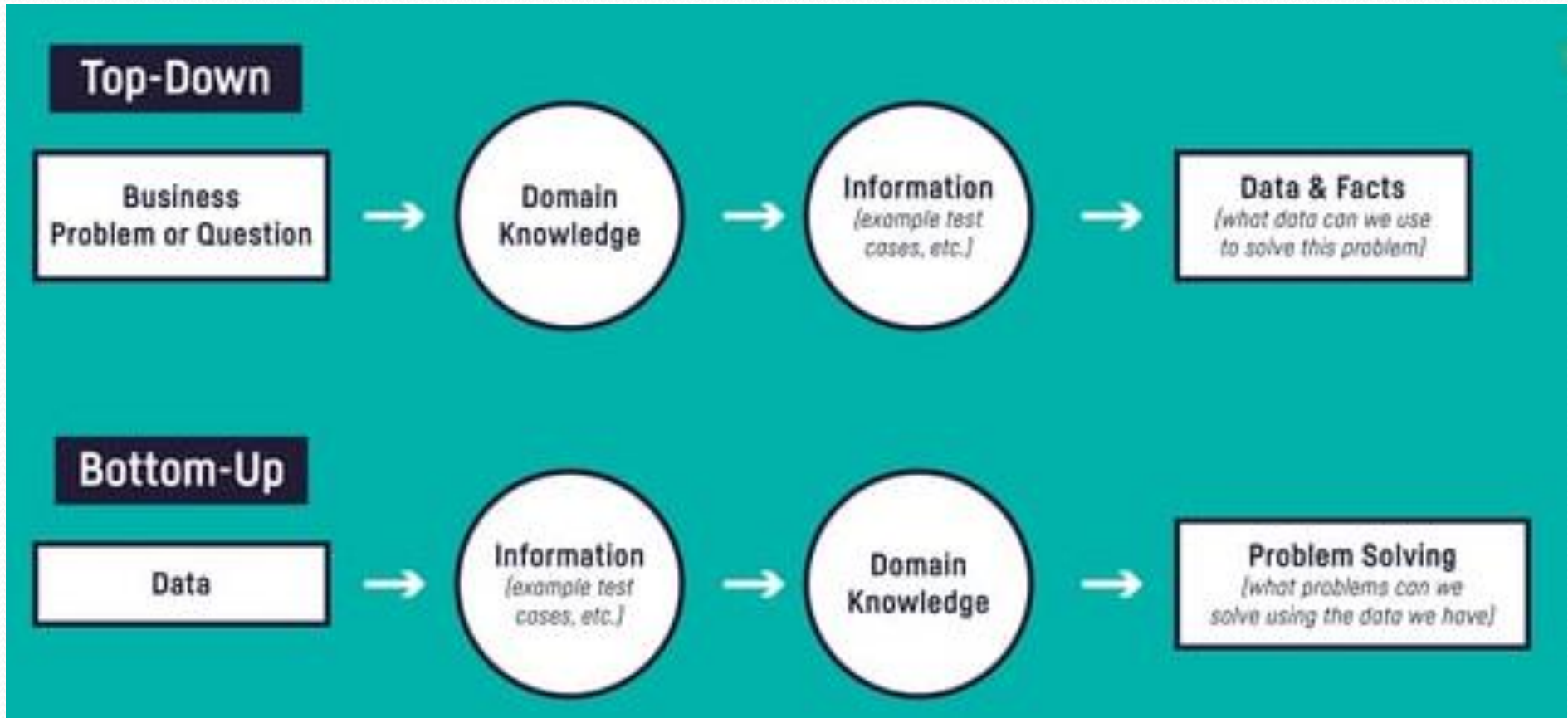


# Design of algorithm





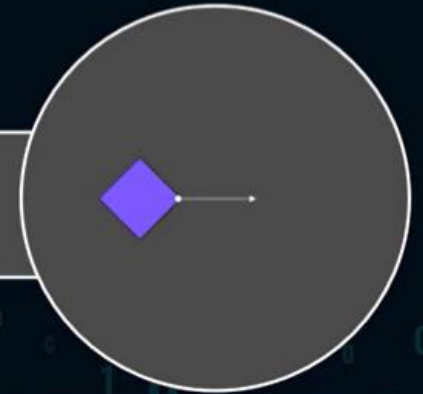
# Design of algorithm



⇒ The bottom-up method to data science tends to be unstructured and exploratory. It lets the data lead to a result, while the top-down method defines a problem to be solved and constructs an experiment to solve it.

# Algorithm vs Programming

An algorithm is more like a concept, a technique to solve a problem.



Programming

vs

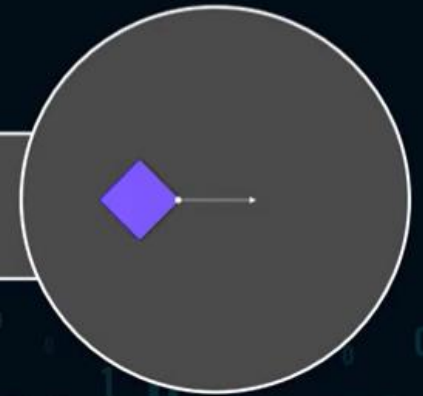
Algorithm



A programmer is related to executing one or more tasks by a computer

# Algorithm vs Programming

An algorithm may be run by a human.



Programming

vs

Algorithm

Programs are run by the compilers of the computer.





# Algorithm vs Programming

An algorithm is designed, and we can analyze the algorithm.



Programming

## Algorithm



Programming is in the implementation phase, so we test the programs.

# Algorithm vs Programming

An algorithm is written using a natural human vocabulary.



Programming

vs

Algorithm



Programs are written in any programming language like C, C++, Java etc.



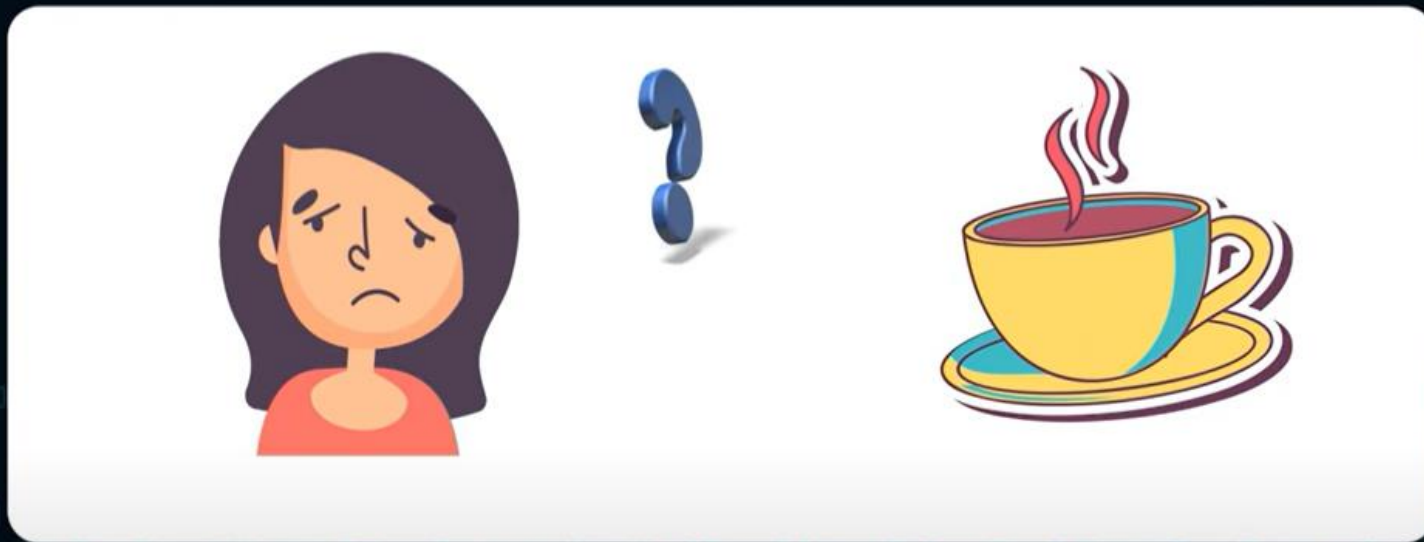
# Algorithm vs Programming

⇒ Algorithm – It is a well-defined, systematic logical approach that comes with a step-by-step procedure for computers to solve any given program.

⇒ Program – It refers to the code (written by programmers) for any program that follows the basic rules of the concerned programming language.



# Example of Algorithm



She wants to prepare tea. To do so, she is following a series of steps

# Example of Algorithm

STEP

1

Take a pan, fill it with water, and place it on the gas stove.

STEP

2

After the water has come to a boil, add the tea leaves and sugar.

STEP

3

Allow for full leaf expansion and milking after that.





# Example of Algorithm

STEP

4

Then wait for the tea to come to a boil.

STEP

5

Turn off the gas after it has finished cooking.

STEP

6

You're all set with your tea now.



# Example of Algorithm

STEP 1

Take a pan, fill it with water, and place it on the gas stove.

STEP 2

After the water has come to a boil, add the tea leaves and sugar.

STEP 3

Allow for full leaf expansion and milking after that.

STEP 4

Then wait for the tea to come to a boil.

STEP 5

Turn off the gas after it has finished cooking.

STEP 6

You're all set with your tea now.

← Input (water)

← Input (tea leaves and sugar)

← Output (tea)

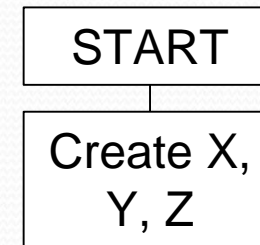


# Example of Algorithm

Create an algorithm that multiplies two numbers and displays the output

Step 1: Start

Step 2: Declare three integers X, Y and Z



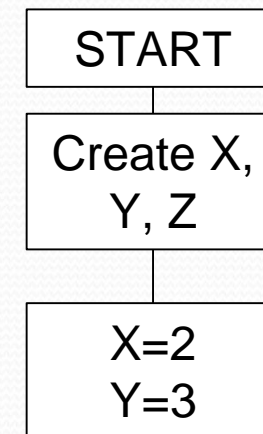
# Example of Algorithm

Create an algorithm that multiplies two numbers and displays the output

Step 1: Start

Step 2: Declare three integers X, Y and Z

Step 3: Define values of X and Y





# Example of Algorithm

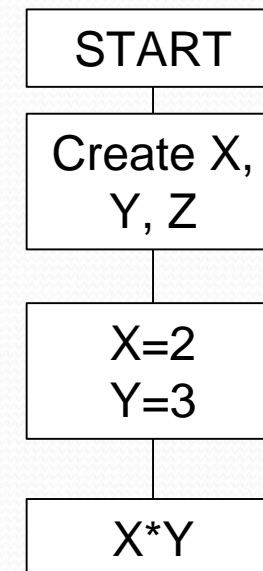
Create an algorithm that multiplies two numbers and displays the output

Step 1: Start

Step 2: Declare three integers X, Y and Z

Step 3: Define values of X and Y

Step 4: Multiply values of X and Y



# Example of Algorithm

Create an algorithm that multiplies two numbers and displays the output

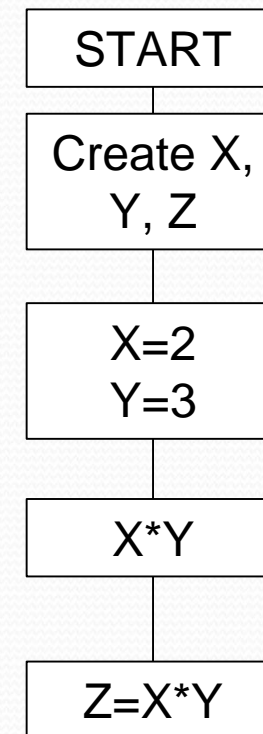
Step 1: Start

Step 2: Declare three integers X, Y and Z

Step 3: Define values of X and Y

Step 4: Multiply values of X and Y

Step 5: Store result of step 4 to Z





# Example of Algorithm

Create an algorithm that multiplies two numbers and displays the output

Step 1: Start

Step 2: Declare three integers X, Y and Z

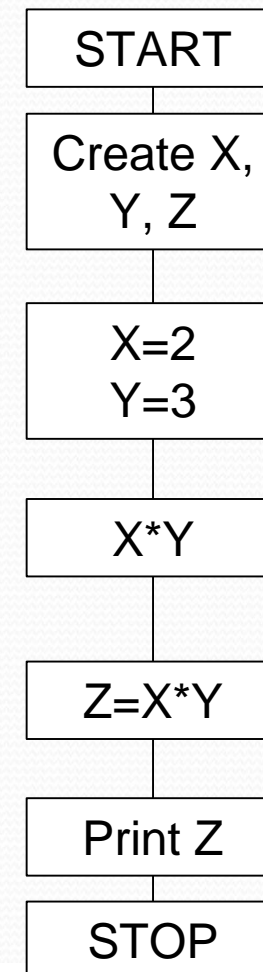
Step 3: Define values of X and Y

Step 4: Multiply values of X and Y

Step 5: Store result of step 4 to Z

Step 6: Print Z

Step 7: Stop



Computer Science  
FYS

**MATLAB**


Essential MATLAB for Scientists

Chap 2 : Introduction and overview

Dr. Delphine Syvilay  
Dr. Madhurima Panja



- 
- MATLAB stands for MATrix LABoratory.

- 
- MATLAB stands for MATrix LABoratory.
  - MATLAB is a powerful computing system for handling the calculations involved in scientific and engineering problems.
  - MATLAB can be used interactively





# 1. Using MATLAB

- Understand the logic
- Basic commands
- Basic functions

# 2. The MATLAB desktop

- Different tools
- How a program works

# 1. Using MATLAB

- Start MATLAB on your desktop
  - The window that concerns us for this chapter is the Command Window.
  - The >> prompt means that MATLAB is waiting for a command.



# 1. Using MATLAB

- Start MATLAB on your desktop
  - The window that concerns us for this chapter is the Command Window.
  - The >> prompt means that MATLAB is waiting for a command.
  - Once can quit MATLAB at any time with one of the following:
    - Select **Exit MATLAB** from the desktop **File** menu.
    - Enter *quit* or *exit* at the Command Window prompt.

# 1. Using MATLAB

- Start MATLAB on your desktop
  - The window that concerns us for this chapter is the Command Window.
  - The >> prompt means that MATLAB is waiting for a command.
  - Once can quit MATLAB at any time with one of the following:
    - Select **Exit MATLAB** from the desktop **File** menu.
    - Enter *quit* or *exit* at the Command Window prompt.
  - Do not click on the close box in the top right corner of the MATLAB desktop. This does not allow MATLAB to terminate properly and, on rare occasions, may cause problems with your computer operating software.



# 1. Using MATLAB

- Let us start by examining if MATLAB does arithmetic correctly
  - Type  $2 + 3$  after the `>>` prompt, followed by **Enter**. Differently presented, execute the following:
    - `>> 2 + 3 <Enter>`
    - Commands are only carried out when you press **Enter**. The answer in this case is, of course, 5.

# 1. Using MATLAB

- Let us start by examining if MATLAB does arithmetic correctly
  - Type `2 + 3` after the `>>` prompt, followed by **Enter**. Differently presented, execute the following:
    - `>> 2 + 3 <Enter>`
    - Commands are only carried out when you press **Enter**. The answer in this case is, of course, 5.
  - Try the following:
    - `>> 3 - 2 <Enter>`
    - `>> 2 * 3 <Enter>`
    - `>> 1/2 <Enter>`
    - `>> 2^3 <Enter>`
    - `>> 2\1 <Enter>`



# 1. Using MATLAB

- Let us start by examining if MATLAB does arithmetic correctly
  - The backslash means that the denominator is to the left of the symbol and the numerator is to the right of the symbol; the result for the last command is 0.5. This operation is equivalent to  $1/2$ .
- Try the following:
  - `>> 2 .* 3 <Enter>`
  - `>> 1 ./ 2 <Enter>`
  - `>> 2 .^3 <Enter>`

# 1. Using MATLAB

- Hints on creating and editing command lines
  - The line with the >> prompt is called the *command line*.
  - You can edit a MATLAB command before pressing **Enter** by using various combinations of the **Backspace**, **Left-arrow**, **Right-arrow** and **Del** keys. This is helpful feature is called *command line editing*.



# 1. Using MATLAB

- Hints on creating and editing command lines
  - The line with the >> prompt is called the *command line*.
  - You can edit a MATLAB command before pressing **Enter** by using various combinations of the **Backspace**, **Left-arrow**, **Right-arrow** and **Del** keys. This helpful feature is called *command line editing*.
  - You can select and edit previous commands you have entered using **Up-arrow** and **Down-arrow**. But remember to press **Enter** to get the *command carried out* i.e. to *run* or *execute* the command.
  - MATLAB has a useful editing feature called *smart recall*. Just type the first few characters of the command you want to recall, e.g. type the characters 2 \* and press the **Up-arrow** key. This action recalls the most recent command starting with 2 \*.

# 1. Using MATLAB

- Anticipating some errors.

Indeed, type the following:

- `>> 0/1 <Enter>`
- `>> 1/0 <Enter>`

- The answer for the first calculation is, of course, 0. However, for the second calculation, MATLAB gives the answer *Inf*.



# 1. Using MATLAB

- Anticipating some errors.

Indeed, type the following:

- `>> 0/1 <Enter>`
- `>> 1/0 <Enter>`

- The answer for the first calculation is, of course, 0. However, for the second calculation, MATLAB gives the answer *Inf*.
- If you insist on using  $\infty$  in a calculation, then type the symbol *Inf* (short for infinity). Try the following:
  - `>> 8 + Inf <Enter>`
  - `>> 13/Inf <Enter>`

# 1. Using MATLAB

- Anticipating some errors.

Indeed, type the following:

- `>> 0/1 <Enter>`
- `>> 1/0 <Enter>`

- The answer for the first calculation is, of course, 0. However, for the second calculation, MATLAB gives the answer *Inf*.
- If you insist on using  $\infty$  in a calculation, then type the symbol *Inf* (short for infinity). Try the following:
  - `>> 8 + Inf <Enter>`
  - `>> 13/Inf <Enter>`
- Another special value that you might meet is *NaN*, which stands for Not-a-Number. It is the answer for calculations like 0/0.



# 1. Using MATLAB

- Variables
  - Try the following:
    - `>> a = 2 <Enter>`
    - This operation assigns the value 2 to the variable  $a$ .

# 1. Using MATLAB

- Variables
  - Try the following:
    - `>> a = 2 <Enter>`
    - This operation assigns the value 2 to the variable  $a$ .
  - Now, try the following:
    - `>> a = a + 7 <Enter>`
    - `>> a = a * 10 <Enter>`
  - Note that each time you clicked on **Enter**, MATLAB displayed the value of the variable  $a$ .



# 1. Using MATLAB

- Variables
  - Try the following:
    - `>> b = 3; <Enter>`
    - This operation assigns the value 3 to the variable *b*.
    - The “;” prevents the value of *b* from being displayed.

# 1. Using MATLAB

- Variables

- Try the following:
  - `>> b = 3; <Enter>`
  - This operation assigns the value 3 to the variable *b*.
  - The “;” prevents the value of *b* from being displayed.
- Now, try the following:
  - `>> b <Enter>`
  - The value of *b* is displayed as *b* is not followed by “;”.



# 1. Using MATLAB

- Variables

- Try the following:

- `>> b = 3; <Enter>`
- This operation assigns the value 3 to the variable *b*.
- The “;” prevents the value of *b* from being displayed.

- Now, try the following:

- `>> b <Enter>`
- The value of *b* is displayed as *b* is not followed by “;”.

- Now, try the following:

- `>> x = 2; y = 3; <Enter>`
- `>> z = x + y <Enter>`
- Note that, in addition to doing the arithmetic with variables with assigned values, several commands separated by semicolons (or by commas) can be put on one line.

# 1. Using MATLAB

- Some functions

- *sqrt(pi)* is for  $\sqrt{\pi}$ .
- *sin(x)* expects the argument “*x*” to be in radians.
- *exp(x)* is the exponential function  $e^x$ .
- *date*.
- *calendar*.
- Note that because of the numerous built-in functions like *pi* or *sin*, care must be exercised in the naming of user-defined variables. The name of user-defined variables should not duplicate the name of a built-in function unless it is deliberately done for a good reason. The problem can be illustrated as follows:



# 1. Using MATLAB

- Some functions
  - Try the following:
    - `>> pi = 17 <Enter>`
    - `>> sqrt(pi) <Enter>`
    - `>> whos (or who) <Enter>`
    - `>> clear pi <Enter>`
    - `>> whos (or who) <Enter>`
    - `>> sqrt(pi) <Enter>`
    - `>> clear <Enter>`
    - `>> whos (or who) <Enter>`

# 1. Using MATLAB

- Some functions

- Try the following:
  - `>> pi = 17 <Enter>`
  - `>> sqrt(pi) <Enter>`
  - `>> whos (or who) <Enter>`
  - `>> clear pi <Enter>`
  - `>> whos (or who) <Enter>`
  - `>> sqrt(pi) <Enter>`
  - `>> clear <Enter>`
  - `>> whos (or who) <Enter>`
- Note that *clear* executed by itself clears all of the local variables in the workspace.
- *whos* is executed to determine the list of local variables or commands presently in the workspace.



# 1. Using MATLAB

- Some commands
  - *clc* for clear command window
  - Note that the difference between functions and commands is that functions usually return with a value, e.g. the date, while commands tend to change the environment in some way, e.g. by clearing the screen, or saving some statements to the workspace (on the disk).

# 1. Using MATLAB

- Scalars and Vectors
  - Variables such as  $a$  and  $b$  are called scalars as they are single-valued.
  - MATLAB also handles vectors (arrays), which are the key to many powerful features of the language.



# 1. Using MATLAB

- Scalars and Vectors

- Variables such as  $a$  and  $b$  are called scalars as they are single-valued.
- MATLAB also handles vectors (arrays), which are the key to many powerful features of the language.
- The following example defines a vector where the elements (components) increase by the same amount “1”:
  - `>> x = 0 : 10; <Enter>`
  - $x$  is a row vector, i.e. a single row by 11 columns array.

# 1. Using MATLAB

- Scalars and Vectors

- Variables such as  $a$  and  $b$  are called scalars as they are single-valued.
- MATLAB also handles vectors (arrays), which are the key to many powerful features of the language.
- The following example defines a vector where the elements (components) increase by the same amount “1”:
  - `>> x = 0 : 10; <Enter>`
  - $x$  is a row vector, i.e. a single row by 11 columns array.
- Now, try the following:
  - `>> size(x) <Enter>`
  - `>> x <Enter>`



# 1. Using MATLAB

- Scalars and Vectors
- Part of the real power of MATLAB is illustrated by the fact that other vectors can be defined in terms of the vector  $x$  just defined.
- Try the following:
  - `>> y = 2 .* x <Enter>`
  - `>> w = y ./ x <Enter>`
  - `>> z = sin(x) <Enter>`

# 1. Using MATLAB

- Scalars and Vectors

- Let us plot the graph of  $\sin(x)$  through the following lines:
  - `>> x = 0 : 0.1 : 10; <Enter>`
  - `>> z = sin(x); <Enter>`
  - `plot(x,z), grid <Enter>`



# 1. Using MATLAB

- Scalars and Vectors

- Let us plot the graph of  $\sin(x)$  through the following lines:
  - `>> x = 0 : 0.1 : 10; <Enter>`
  - `>> z = sin(x); <Enter>`
  - `plot(x,z), grid <Enter>`
- Note that the first command line above has three numbers after the equal sign. When there are three numbers separated by two colons in this way, the middle number is the *increment*. The increment of 0.1 was selected to give a reasonably smooth graph. The command *grid* following the comma in the last command line adds a grid to the graph.

## 2. The MATLAB desktop

- Help
  - MATLAB has an extremely useful online Help system.
  - To get into the **Help** menu, either click on the help button (?) in the desktop toolbar, or select the **Help menu** in any tool.
  - Open the Help browser



## 2. The MATLAB desktop

- Programs
  - We saw some simple examples on how to use MATLAB by entering simple commands or statements at the MATLAB prompt.
  - What if we want to solve problems which MATLAB cannot do in one line, like finding the roots of a quadratic equation while taking into account all the special cases. Well, in that case we need a collection of statements. That collection of statement is called a program.

## 2. The MATLAB desktop

- Programs

- Cut and Paste (example 1)

- Suppose you want to draw the graph of  $e^{-0.2x} \sin(x)$  over the domain 0 to  $6\pi$ . Proceed as follows:

- From the MATLAB desktop, select **File** → **New** → **M-file**, or click the new file button on the desktop toolbar. This action opens an *untitled* window in the Editor/Debugger. For the time being, you may consider this new window as a scratch pad.

- Type the following:

- $x = 0 : \pi/20 : 6 * \pi;$
- $\text{plot}(x, \exp(-0.2 * x) .* \sin(x), 'r'), \text{grid}$

- Use **Copy** these two lines from the M-file and **Paste** them into the Command Window then press <**Enter**>.



## 2. The MATLAB desktop

- Programs

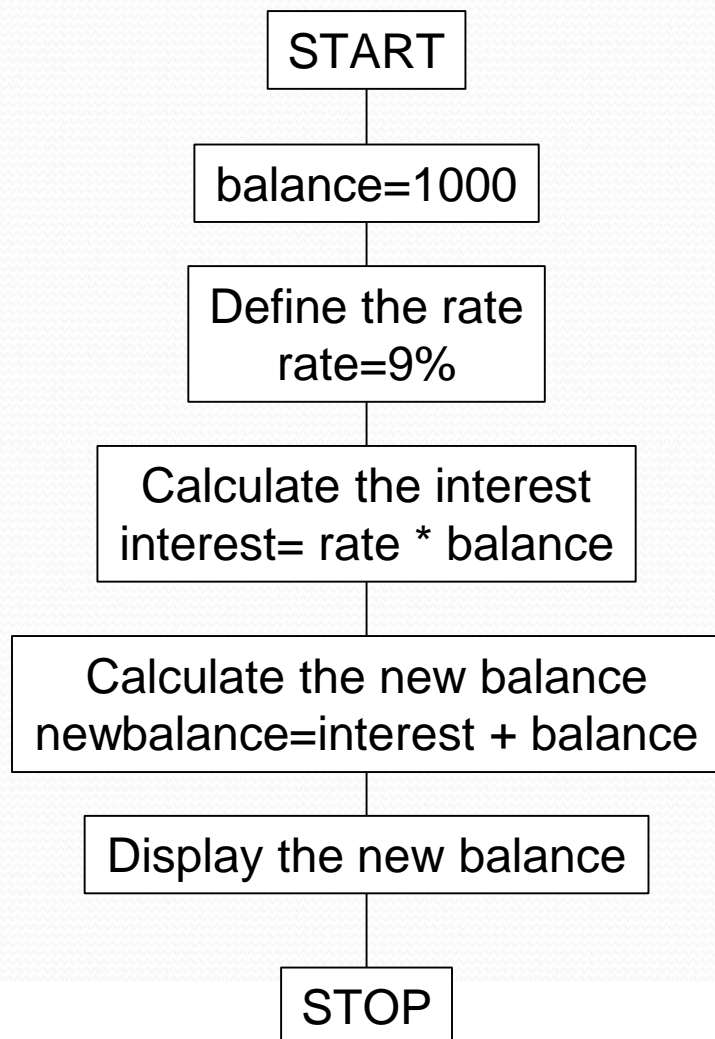
- Suppose you have \$1000 saved in the bank. Interest is compounded at the rate of 9 percent per year. What will your bank balance be after one year?
- Note: it is always advised to write down a rough *structure plan*/algorithm before you program your problem regardless of its simplicity.

## 2. The MATLAB desktop

- Programs

**Algorithm**

**Matlab code**

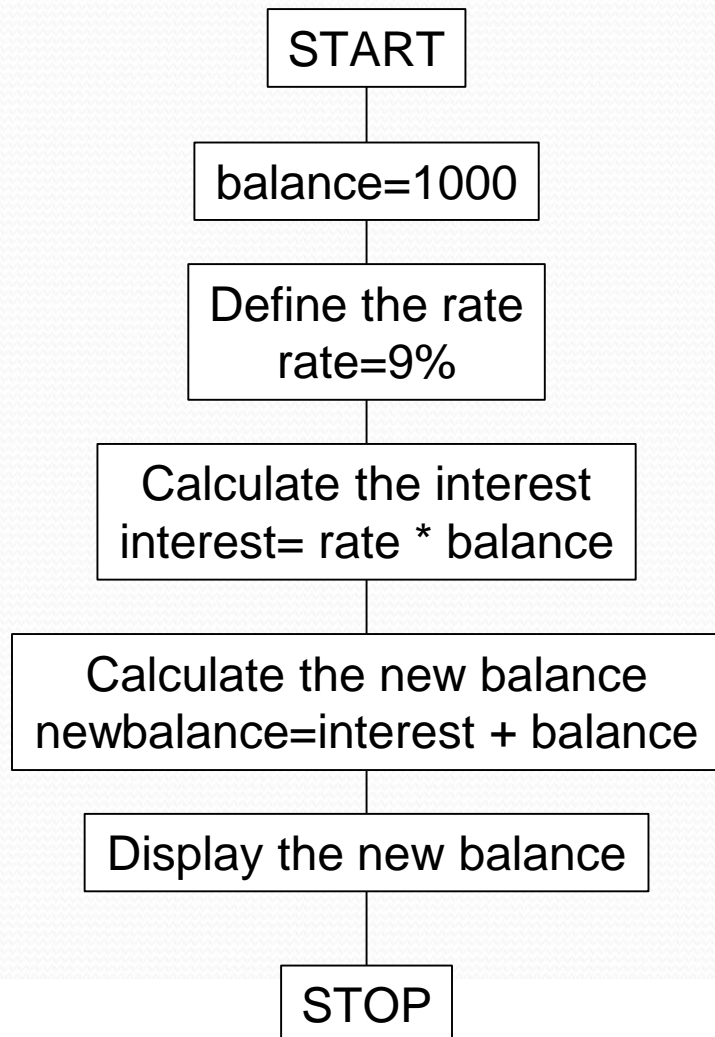




## 2. The MATLAB desktop

- Programs

### Algorithm



### Matlab code

```
%% PROGRAM TO CALCULATE INTEREST  
balance = 1000;
```

```
rate = 0.09;
```

```
interest = rate * balance;
```

```
newbalance = balance + interest;  
or balance = balance + interest;
```

```
disp('New balance:');
```

```
disp(balance);
```

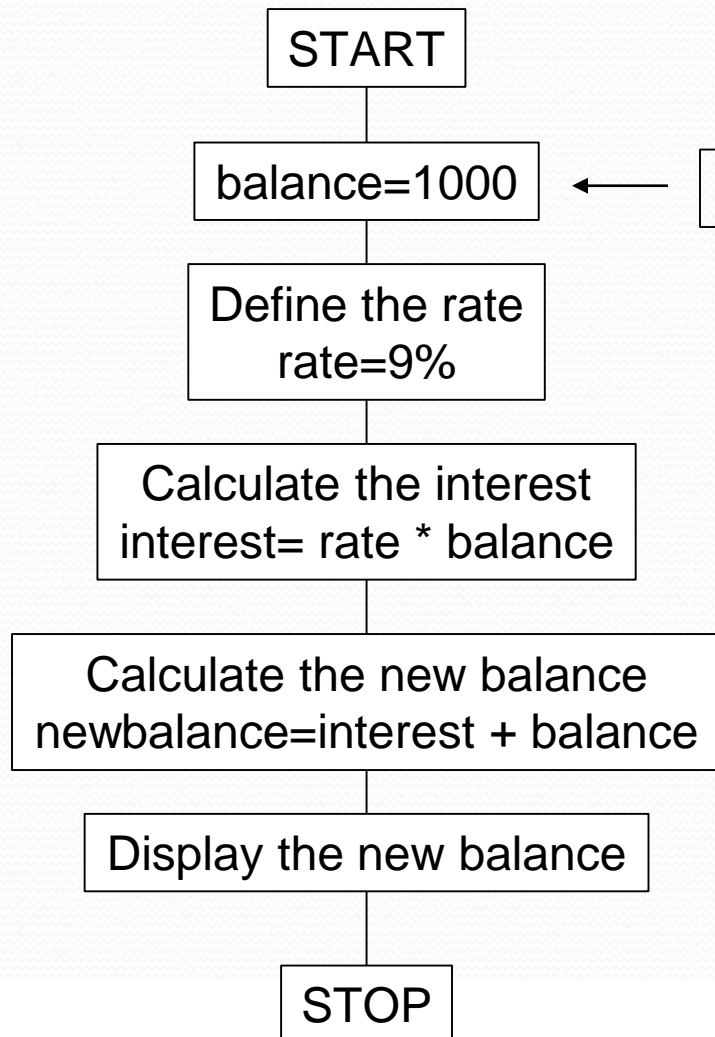
```
%% END OF THE PROGRAM
```

## 2. The MATLAB desktop

- Programs

### Algorithm

### Matlab code



Get initial balance

`balance = input('What is the balance? ');`



## 2. The MATLAB desktop

- Programs

- Saving a program: script files

- Saving your work is essential especially if you want to use it again later.
- To save the contents of the Editor, select **File** → **Save** from the Editor menu bar. A **Save file as:** dialogue box appears. Select a directory and enter a file name which *must* have the extension *.m*, in the **File name:** box, e.g. *test.m*.

Click on **Save**. The Editor window now has the title *test.m*.

## 2. The MATLAB desktop

- Programs

- Saving a program: script files

- Saving your work is essential especially if you want to use it again later.
- To save the contents of the Editor, select **File** → **Save** from the Editor menu bar. A **Save file as:** dialogue box appears. Select a directory and enter a file name which *must* have the extension *.m*, in the **File name:** box, e.g. *test.m*.

Click on **Save**. The Editor window now has the title *test.m*.

If you make subsequent changes to *test.m* in the Editor, an asterisk appears next to its name at the top of the Editor until you save the changes.



## 2. The MATLAB desktop

- Programs
  - Saving a program: script files
    - You just save a *script* file. The very significance of a script file is that, if you enter its name at the command-line prompt, MATLAB carries out each statement in the script file as if it were entered at the prompt. It is equivalent to copying all the lines of the script and pasting them into the Command Window then pressing <Enter>.

## 2. The MATLAB desktop

- Programs
  - Saving a program: script files
    - You just save a *script* file. The very significance of a script file is that, if you enter its name at the command-line prompt, MATLAB carries out each statement in the script file as if it were entered at the prompt. It is equivalent to copying all the lines of the script and pasting them into the Command Window then pressing **<Enter>**.
    - As an example, save the compound interest program above in a script file under the name *compint.m*. Then simply enter the name *compint* at the prompt (**>>**) in the Command Window followed by **<Enter>**.
    - Note: the rules for script file names are the same as those for MATLAB variable names.



## 2. The MATLAB desktop

- Programs
  - How a program works
    - The MATLAB system is technically called an *interpreter* (as opposed to a *compiler*). This means that each statement presented to the command line is translated (interpreted) into a language the computer understands better, and then *immediately* carried out.

## 2. The MATLAB desktop

- Programs

- How a program works

- The MATLAB system is technically called an *interpreter* (as opposed to a *compiler*). This means that each statement presented to the command line is translated (interpreted) into a language the computer understands better, and then *immediately* carried out.
- A fundamental concept in MATLAB is how numbers are stored in the computer's *random access memory* (RAM). If a MATLAB statement needs to store a number, space in the RAM is set aside for it.



## 2. The MATLAB desktop

- Programs

- How a program works

- The MATLAB system is technically called an *interpreter* (as opposed to a *compiler*). This means that each statement presented to the command line is translated (interpreted) into a language the computer understands better, and then *immediately* carried out.
- A fundamental concept in MATLAB is how numbers are stored in the computer's *random access memory* (RAM). If a MATLAB statement needs to store a number, space in the RAM is set aside for it.
- The statement *balance* = 1000 allocates the number to the memory location named *balance*. Since the content of *balance* may be changed during a session, it is called a *variable*.

## 2. The MATLAB desktop

- Programs

- How a program works

The statements in our program are therefore interpreted by MATLAB as follows:

1. Put the number 1000 into the variable balance.
2. Put the number 0.09 into the variable rate.
3. Multiply the contents of rate by the contents of balance and put the answer/result in the variable interest.
4. Add the contents of balance to the contents of interest and put the answer in balance.
5. Display (in the Command Window) the message given in the single quotes.
6. Display the contents of the variable balance.

- When the program has finished running, the variables used will have the following values:

balance : 1090

interest : 90

rate : 0.09



## 2. The MATLAB desktop

- Programs
  - How a program works
    - Try the following exercises:
      1. Run the program as it stands.
      2. Change the first statement in the program to read *balance = 2000*;  
Make sure that you understand what happens now when the program runs.

## 2. The MATLAB desktop

- Programs

- How a program works

- Try the following exercises:

1. Run the program as it stands.

2. Change the first statement in the program to read *balance = 2000;*  
Make sure that you understand what happens now when the program runs.

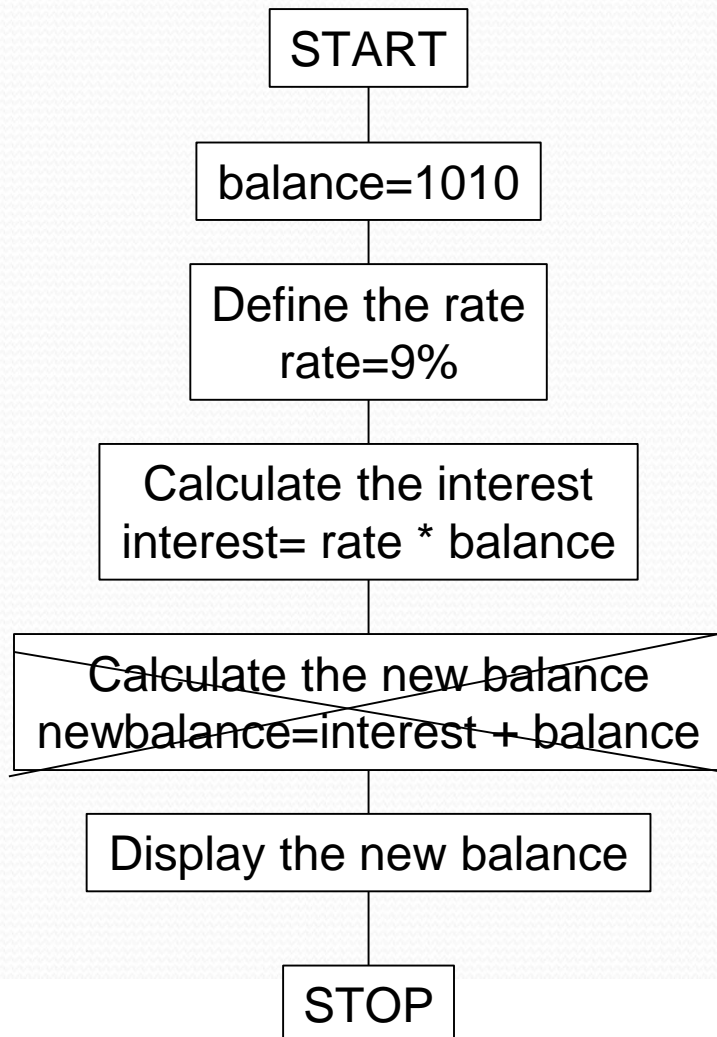
3. Leave out the line *balance = balance + interest;*  
and rerun.



## 2. The MATLAB desktop

- Programs

### Algorithm



### Matlab code

```
%% PROGRAM TO CALCULATE INTEREST  
balance = 1000;
```

```
rate = 0.09;
```

```
interest = rate * balance;
```

```
disp( 'New balance: ' );  
disp( balance );
```

```
%% END OF THE PROGRAM
```

## 2. The MATLAB desktop

- Programs
  - How a program works
    - A number of questions have probably occurred to you by now, such as:
      - What names may be used for variables?
      - How can numbers be represented?
      - What happens if a statement would not fit on one line?
      - How can we organize the output more neatly?



## 2. The MATLAB desktop

- Programs

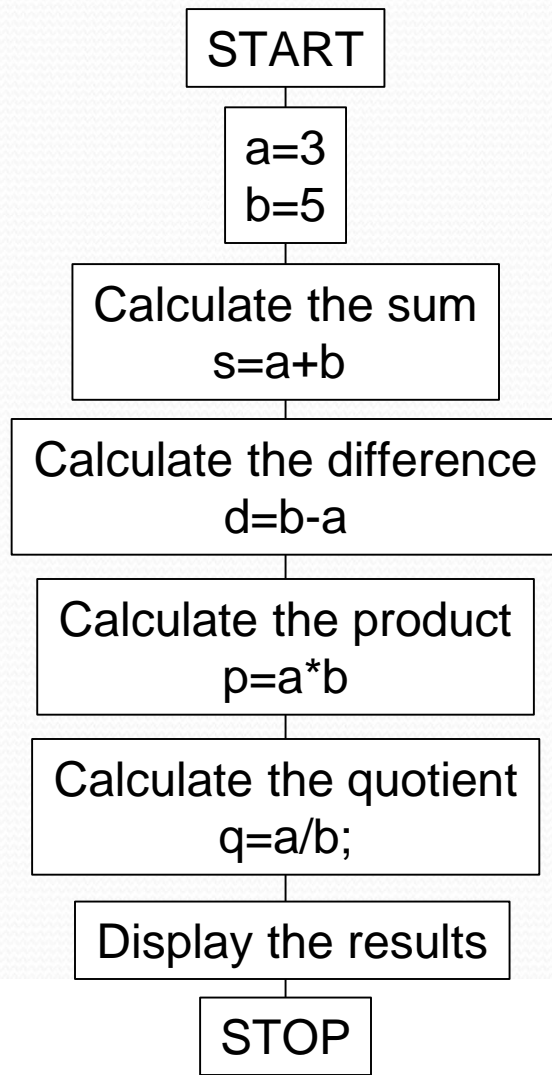
You may try the following exercise on your own:

- Give values to variables  $a$  and  $b$  on the command line, e.g.,  $a = 3$  and  $b = 5$ .
- Write some statements to find the sum, difference, product and quotient of  $a$  and  $b$ .

## 2. The MATLAB desktop

- Programs

**Algorithm**



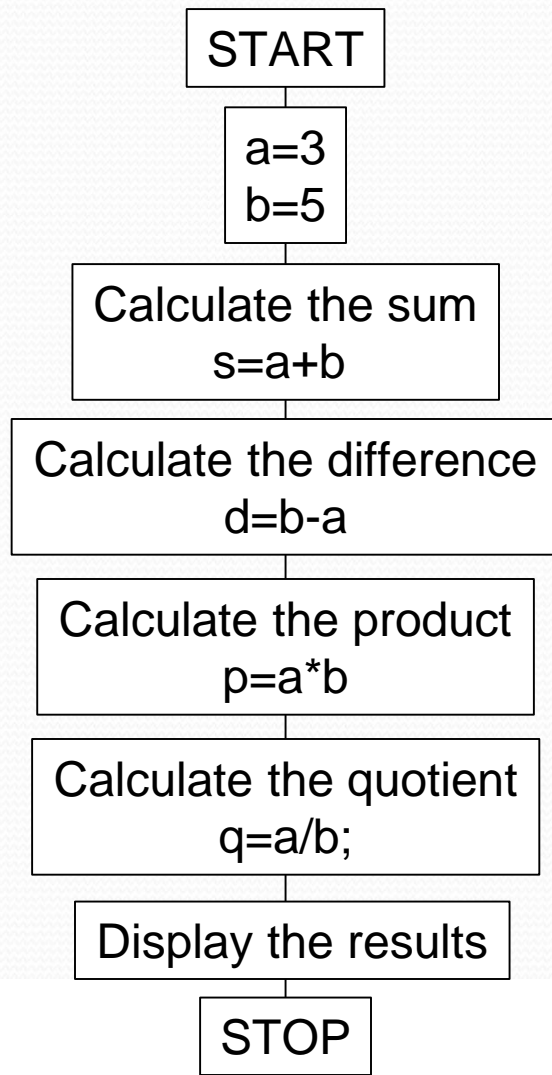
**Matlab code**



## 2. The MATLAB desktop

- Programs

### Algorithm



### Matlab code

```
%% PROGRAM FOR MATHEMATICAL OP  
a=3; b=5;  
  
s=a+b;  
  
d=b-a;  
  
p=a*b;  
  
q=a/b;  
  
disp(s);disp(d);disp(p);disp(q);  
%% END OF THE PROGRAM
```

# To summarize

## 1. Using MATLAB

- Arithmetic (+, \*, ./, .\* Inf NaN)
- Basic commands (; clear clc whos, assign value to a variable)
- Basic functions (size(), sqrt(), sin(), exp())
- Scalars and Vectors

## 2. The MATLAB desktop

- Definition of a program, running a program
- Script file
- How a program works