Lab 1

Contents

1 Command line basics

1.1 The most important windows in Matlab

- Workspace
- Command window
- Script editor

1.2 Changing working directory

Use "Current folder" window or `cd` command (for more information see `https://se.mathworks.com/help/matlab/ref/cd.html`).

1.3 Simple Matlab commands

```
3 + 5
3 * 4
4 ^ 2

sqrt(4)

disp('Hello, World!')
```

1.4 Creating variables

```
a = 5
b = 10/2

salary = 1200;
taxes = 110;
salary_after_taxes = salary - taxes

num_employees = 10;
total_salary = salary * num_employees;

disp(total_salary);

text = 'Hello, World!';
disp(text);

courseName = 'TNK055';
```

Variables can be observed in the Workspace pane.

1.5 Semicolons

Semicolon suppresses output of the code line.

Without a semicolon Matlab will print the result of the command execution.

```
>> a = sqrt(4)

a =

     2

>>
```

When we use a semicolon Matlab suppresses the output.

```
>> a = sqrt(4);
>>
```

1.6 Special variable ans

```
5 * 5
disp(ans);
```

1.7 Variables names

Requirements for naming a variable are as follows:

- first character must be a LETTER
- after that, any combination of letters, numbers and _
- CASE SENSITIVE! (var1 is different from Var1)

1.8 Do not use these variables names

- i, j - they can be used for complex numbers
- pi
- ans
- Inf
- NaN

1.9 clear command

Clears the current workspace (i.e., deletes all variables from it)

```
clear;
```

1.10 lookfor function

Searches for the specified keyword. Helps to find the name of the function. For example: `lookfor sine`

1.11 help function

Shows information about functions. For example: `help sin`

1.12 doc function

The same as help, but looks better. For example `doc sin`

2 Scripts

2.1 Write a script

```
% everything after percent sign until the end of the line is ignored by Matlab
clear; % clears workspace (deletes all variables from it)

profit = 1200; % Profit from selling an item in US dollars
sold = 20; % Number of sold items
owners = 3; % How many people share profit

profit_per_owner = (profit * sold)/owners; % Profit of each owner

disp('The profit per owner is:');
disp(profit_per_owner);
```

Matlab executes all commands line by line from top to bottom each time you run the script.

2.2 Importance of clear

Unlike many other programming languages, Matlab keeps all variables in the workspace even after the end of the script execution, unless `clear` command is called. If we forget to use `clear`, it may lead to potential mistakes. For example, try to initialize variable `a = 10;` and then try to run the following script several times

```
a = a + 1;
disp(a);
```

You may observe that variable `a` saves the value of the variable between runs and increases it each time.

2.3 Debugging (breakpoints)

You can put breakpoint on a line and Matlab will pause before executing this line. You can check the state of all variables, change them, etc. In order to continue, you can use either "Continue" button or "Step" button. "Continue" button (shortcut F5) allows to run through breakpoints, so Matlab continues execution of code until the next breakpoint. "Step" button (shortcut F10) executes only one line and stops before executing the next line. "Step" button can be used to run the code line by line.

3 Receiving user input

In Matlab, it is possible to interactively ask for user input using command `input()`. It is possible to print some text as an invitation for a user. The user is expected to write some value using his/her keyboard and hit Enter when done.

```
clear;
price = input('Please write the price of an item and
   press Enter: '); % The text between parenthesis
   will be printed as an invitation
amount = input('Please write amount of items to order
    and press Enter: '); % The text between
   parenthesis will be printed as an invitation

total = price*amount
disp('The total price is:')
disp(total)
```

4 Conditional statements

4.1 Boolean data type

Boolean data type takes one of two possible logical values: "true" or "false". Matlab stores "false" and "true" as 0 and 1 respectively.

4.2 Conditional expressions

The result of a conditional expression is of boolean type.

Examples of conditional expressions

```
a = 10;

a < 30 % true
a > 20 % false
a == 10 % true
a == 11 % false
```

4.3 if statement

`if` statement allows you to perform different computations or actions depending on whether a condition evaluates to true of false.

Basic syntax is:

```
if condition
    code
end
```

Example:

```
a = 10; % try to change the value!

if a > 9
    disp('a is greater than 9');
end


if a < 20
    disp('a is less than 20');
end
```

4.4 else statement

Code in the `else` block will be executed if `condition` is false.

Basic syntax:

```
if condition
    code1
else
    code2
end
```

Example:

```
a = 20;

if a < 40
    disp('a is less than 40');
else
    disp('a is greater than 40');
end
```

4.5 elseif statement

`elseif` statement allows to combine several conditions. Only the code following the first condition that is found to be true will be executed. All other code will be skipped.

Basic syntax:

```
if condition1
    code1
elseif condition2
    code2
elseif condition3
    code3
...
else
    code
end
```

Examples:

```
a = 10; % try to set a = 4; a = 5; a = 6;

if a > 5
    disp('a > 5');
elseif a < 5
    disp('a < 5');
else
    disp('a == 5');
end
```

4.6 and/or/not operators

If you want to have complex conditions which consist of more than one logical statement, you can use logical "and", "or" and "not" operators.

4.6.1 And

The "and" of two or more conditions is true if each of the conditions is true. For example, `a and b` is true only if `a` and `b` are both true.

In Matlab, logical "and" is written as `&&`.

Example:

```
a = 10;

if a > 5 && a < 15
    disp('a > 5 and a < 15');
else
    disp('a <= 5 or a >= 15');
end
```

4.6.2 Or

The "or" of two or more conditions is true if at least one of the conditions is true. For example, `a or b` is true if either `a` or `b` (or both) are true.

In Matlab, logical "or" is written as `||`.

Example:

```
a = 10;

if a < 5 || a > 9
    disp('a < 5 or a > 9');
else
    disp('9 => a >= 5');
end
```

4.6.3 Not

`not` operator negates the condition. If `a` is true, then `not a` is false. If `a` is false, then `not a` is true.

In Matlab, logical "not" is written as `~`.

Example:

```
a = 10

if ~(a > 0)
    disp('a <= 0');
else
    disp('a > 0');
end
```

5 Switch statement

```
clear;
n = input('Enter a number: ');

switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

6 For loops

For loop is used for a known number of iterations.

6.1 Hello world 20 times

```
clear;

for i=1:20
    disp('Hello, world!');
end
```

6.2 Hello world with numbers

```
clear;

for i=1:20
    disp('Hello, world!');
    disp(i);
end
```

6.3 Sum of numbers from 1 to 10

```
clear;

to = 10;
sum = 0;

for i = 1:to
    sum = sum + i;
end
```

7 While loops

While loop repeats code while the condition is true.

```
clear;

n = 5;

while n > 1
    n = n-1;
    disp(n);
end
```

7.1 Infinite loops

Be careful, an infinite loop (it is a loop which never ends on its own) is possible with `while`, for example

```
% This loop will be running forever
clear;

n = 5;

while n > 1 % initially n > 1
    n = n+1; % and we increasing n each iteration,
       hence the loop will never end
    disp(n);
end
```

7.2 break statement

`break` statement stops the loop immediately. No further iterations will be done. This statement works with both `while` and `for` loops.

In the following example we use `break` to exit the infinite cycle.

```
clear;

n = 5;
while n > 1
    n = n+1;

    if n > 100 % when n > 100
        break % we stop the loop
    end
    disp(n);
end
```

The following code will stop printing after 3 because the loop is terminated when `a == 4`.

```
clear;

for a=1:5
    if a == 4
        break
    end

    disp(a);
```

```
end
```

In the following example the loop stops when the user chooses 0 as the input.

```
clear;
secret = 3;
guess = 0;

while guess ~= secret
    guess = input('Guess my secret number between 1
       and 10 (to exit enter 0 ):');

    if guess == 0
        disp('You chose to exit.');
        break
    end

    if guess == secret
        disp('Correct!');
    else
        disp('Try again >>');
    end
end
```

Also this program can be implemented using infinite loops:

```
clear;
secret = 3;
guess = 0;

while 1 == 1 %force the loop to be infinite
    guess = input('Guess my secret number between 1
       and 10 (to exit enter 0 ):');

    if guess == 0
        disp('You chose to exit.');
        break
    end

    if guess == secret
        disp('Correct!');
```

```
        break
    else
        disp('Try again >>');
    end
end
```

7.3 continue statement

`continue` statement allows to skip the rest part of the code in current iteration and to go to the next iteration of the loop. This statement works with both `while` and `for` loops.

In the following example "Hello, world!" text will not be printed, because `continue` is the first statement in the for loop.

```
clear;

for a=1:5
    continue

    disp('Hello, world!');
end
```

And the following code will print "Hello, world!" 5 times and "Good bye, world!" only 3 times because of the `continue` command before the second `disp` command.

```
clear;

for a=1:5
    disp('Hello, world!');

    if a > 3
        continue
    end

    disp('Good bye, world!');
end
```

8 Homework 1

Task 1

Write a program that computes the area of a triangle or a parallelogram. Let the user input the type of their figure (1 for triangle, 2 for parallelogram), the height, and the base. Print the result or an error message in case of wrong user input.

Task 2

Write a script to input electricity unit charges and calculate total electricity bill according to the given conditions:

- For the first 1000 units - 1.5 SEK/unit
- For the units above 1000 - 2.0 SEK/unit
- If units number is negative, display an error message

You can test your program on the following data:

- If number of units is -100, an error message is printed
- If number of units is 100, total amount to pay is 150 SEK
- If number of units is 3000, total amount to pay is 5500 SEK

Task 3

Write a script which asks user to enter a number from 1 to 5 and prints it using words (like "one" for 1). Use `switch` statement. In case of wrong user input display an error message.

Task 4

Modify Task1 so, that the script does not stop until the user enters 0, when asked for the figure type. So, 1 means triangle, 2 - parallelogram, 0 - exit the script, any other digit - error message.

Task 5

Write a program that calculates the total price for several movie tickets. Each ticket price depends on the age of the ticket's owner.

- If the owner is over 65 years old then the ticket costs 96 SEK
- For children younger than 12 years the price is 105 SEK
- Otherwise, the owner pays 120 SEK

Your program should perform the following task by the given order.

1. Ask the number of tickets to be bought. This number should be between 1 and 10. Assume that the user enters an integer value, but validate the input to be within the range. If the number of tickets given is incorrect then the script should display an error message and request the user to enter the number of tickets again.
2. Request the age of each ticket owner. Assume that the user always gives a non-negative integer for each age, i.e. no need to validate the input.
3. Display the total price to be paid.

A running example is shown below (user input shown in red).

```
Welcome to our Filmstaden!
Enter number of tickets (1-10): 11
Invalid number of tickets!
Enter number of tickets (1-10): 5
Enter age for person 1: 40
Enter age for person 2: 42
Enter age for person 3: 67
Enter age for person 4: 8
Enter age for person 5: 10
Total price = 546 SEK
```