Making Choices

Learning Objectives

- Learn about logical operators
- Learn about if, elseif and else
- Learn to test equality, AND, and OR conditions
- Learn to nest loops
- Learn about conditional vectors

Part 1: Logical operators

Intro to logical operators available in MATLAB

- Less than, greater than, equal to, less than / greater than, not equal to

What happens when you type:

```
1<2
```

```
ans = logical
   1
```

```
1>=2
```

```
ans = logical
   0
```

```
1==2
```

```
ans = logical
   0
```

```
35==35
```

```
ans = logical
   1
```

```
35==70
```

```
ans = logical
   0
```

By observing the outputs of the above statements you can see that a logical value of "0" corresponds to "false" statements and a logical value of "1" corresponds to "true" statements

Also notice that in order to generate logical statements "==" is used instead of "=" since using only one equals sign corresponds to creating a variable.

- AND and OR (&& and ||)

```
% AND logic table: only when both statements are true the output of an AND statement is true
1<2 && 3==3 % true AND true
```

```
ans = logical
   1
```

```
1>2 && 3==3 % false AND true
```

```
ans = logical
   0
```

```
1<2 && 3~=3 % true AND false
```

```
ans = logical
   0
```

```
1>2 && 3~=3 % false AND false
```

```
ans = logical
   0
```

```
% OR logic table: only when both statements are false the output of an OR statement is false
1<2 || 3==3 % true AND true
```

```
ans = logical
   1
```

```
1>2 || 3==3 % false AND true
```

```
ans = logical
   1
```

```
1<2 || 3~=3 % true AND false
```

```
ans = logical
   1
```

```
1>2 || 3~=3 % false AND false
```

```
ans = logical
   0
```

- True, false

```
true
```

```
ans = logical
   1
```

```
false
```

```
ans = logical
   0
```

Challenge

The logical function converts a statement to its logical value - either a 1 or a 0

Read the following statements - try and guess what the answer will be before you check

```
logical(0 == false)
```

```
ans = logical
   1
```

```
logical(1 == true)
```

```
ans = logical
   1
```

```
logical(42 == true)
```

```
ans = logical
   0
```

```
logical(42)
```

```
ans = logical
   1
```

```
logical(0 && (8 || 1))
```

```
ans = logical
   0
```

What happens if you use a vector with an if statement?

```
if([1 2 4 6])
    disp('it worked')
end
```

```
it worked
```

What about a string?

```
if('hello')
```

```
    disp('it worked')
end
```

```
it worked
```

Part 2: If statements

The programs we have written so far always do the same things, regardless of what data they're given.

We want programs to make choices based on the values they are manipulating.

```
num = 200; % create an arbitrary number and hold it in the variable called "num"

% Check "if" num is larger than 100
if num>100
    disp('number is greater than 100');
elseif num == 100 % if num is not greater than 100 check if it equal to 100
    disp('number is equal to 100')
else % "if" num is not equal to or larger than 100 then "else" it must be smaller
    disp('number is not greater than 100');
end
```

```
number is greater than 100
```

```

disp('done')
```

```
done
```

Mini Challenge: Change num and see what happens.

AND, OR tests

OR statement

AND statement

The if statement above to check if num is larger or equal to 100 can be condensed by using logical statements (AND, OR)

```
if num > 100 || num == 100 % Using OR (||)
    disp('number is greater or equal to 100')
else
    disp('number is not greater or equal to 100');
end
```

```
number is greater or equal to 100
```

```
% another way is
if num >= 100
    disp('number is greater or equal to 100')
else
    disp('number is not greater or equal to 100');
```

```
end
```

```
number is greater or equal to 100
```

The "NOT" logic operator in MATLAB is "~".

The NOT operator means the opposite of the logic value:

"~true" is false

"~false" is true

"~=" means not equal to etc.

```
~true
```

```
ans = logical
   0
```

```
% Another way to form the if statement is
if num < 100 && num ~= 100 % Using AND (&&)
    disp('number is not greater or equal to 100')
else
    disp('number is greater or equal to 100');
end
```

```
number is greater or equal to 100
```

Challenge

Write code to work out which number is bigger (and try changing the values of num1 and num2 to make sure your code always works)

```
num1 = 40;
num2 = 25;

% write your code here
```

Extension Challenge

Write code to decide whether a number is 10% greater than a threshold value.

Modify the code below

```
threshold = 37;
number = 40;
```

Define a new threshold that's 10% greater than the old threshold

```
%new_threshold =
```

Write code here to see if the number is greater than the new threshold.

```

```

Check your code with some different values of number – does it still work? (try negative numbers too)

Part 3: Nesting Loops

Combining if statements with loops.

I want to get the sum of all the negative numbers.

```
numbers = [-5, 3, 2, -1, 9, 6];     % list of numbers to test
total = 0;      % initialise value to zero
```

HINT:

- You will need a for loop to go through each element in the "numbers" vector
- You will also need an if statement to check if each element is negative or positive
- Also a variable (called total in this example) needs to update the sum of each negative element

A template for the code is given below

```
for n = 1:x % instead of x write the number of elements to checked for negativity
    if numbers(n) < y % instead of y write a number that wil check if the value of nth element
        total = total + numbers(n); % add the value to the total only if the value negative
    end
end
% in the end display total to see what the total sum if negative numbers in the list is, check
total
```

```
total = -6
```

Challenge

Edit the above code so it also displays the sum of positive numbers.

Part 4: Conditional Vectors

One of the great things about MATLAB is we don't always need to nest conditional statements in a loop. MATLAB can condition vectors directly.

```
vector1 = [1 0 0 1 0 1];
vector2 = [1 0 1 0 0 0];
```

Try using logic operators between vectors:

```
vector1 == vector2 % are the two vectors equal
```

```
ans = 1×6 logical array
```

```
     1     1     0     0     1     0
```

It is seen that the output has given the result of the condition for each element of both vectors

The first two elements are the same so the 1st element of the output is 1 since the statement is true for the first element

The 3rd element of the two vectors are NOT equal so the output vector has a 0 for false etc.

Same output could be obtained with vectors that have values other than 1 or 0's

```
samplevect1 = [3 5 12 -7 100];
samplevect2 = [3 5 -1 -7 99.5];
samplevect1 == samplevect2 % are the two vectors equal
```

```
ans = 1×5 logical array
   1   1   0   1   0
```

Using vector1 and vector2, logical operations can be done

```
vector1 & vector2 % AND
```

```
ans = 1×6 logical array
   1   0   0   0   0   0
```

Try other logical operators on the two vectors (OR, NOT and other combinations)

We can do this vector conditioning with all our logical operators. So if we want the sum of all the positive and negative numbers, we can do this without a loop.

```
numbers = [-5, 3, 2, -1, 9, 6];

numbers< 0 % finding the indexes of numbers smaller than 0 in the numbers vector
```

```
ans = 1×6 logical array
   1   0   0   1   0   0
```

```
numbers>= 0 % finding the indexes of numbers larger or equal to 0 in the numbers vector
```

```
ans = 1×6 logical array
   0   1   1   0   1   1
```

```
numbers(numbers < 0) % using the logical indexes to obtain the actual values fitting the logica
```

```
ans =
    -5    -1
```

```
numbers(numbers >= 0)
```

```
ans =
     3     2     9     6
```

```
pos_total = sum(numbers(numbers >= 0));
neg_total = sum(numbers(numbers < 0));

disp(['sum of positive values: ', num2str(pos_total)]);
```

```
sum of positive values: 20
```

```
disp(['sum of negative values: ', num2str(neg_total)]);
```

```
sum of negative values: -6
```