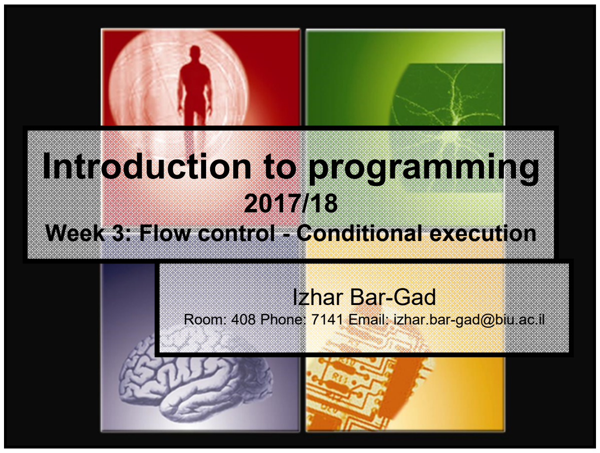
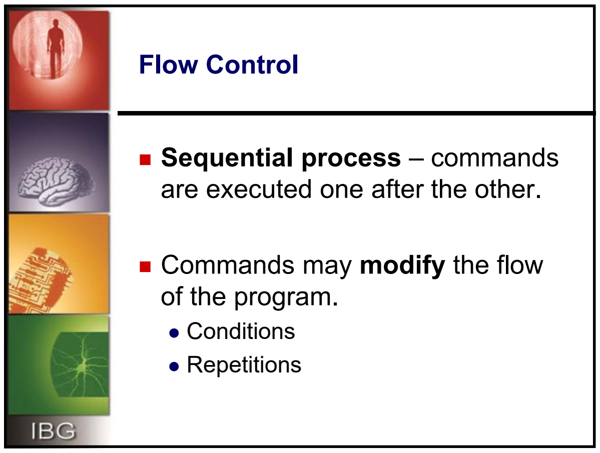


Introduction to programming

2017/18

Week 3: Flow control - Conditional execution

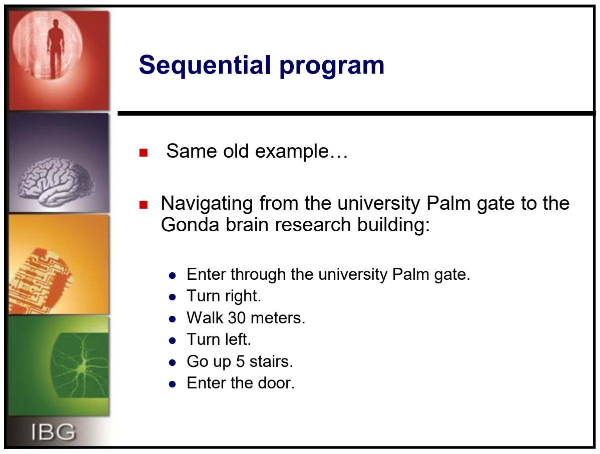
Izhar Bar-Gad
Room: 408 Phone: 7141 Email: izhar.bar-gad@biu.ac.il



Flow Control

- **Sequential process** – commands are executed one after the other.
- Commands may **modify** the flow of the program.
  - Conditions
  - Repetitions

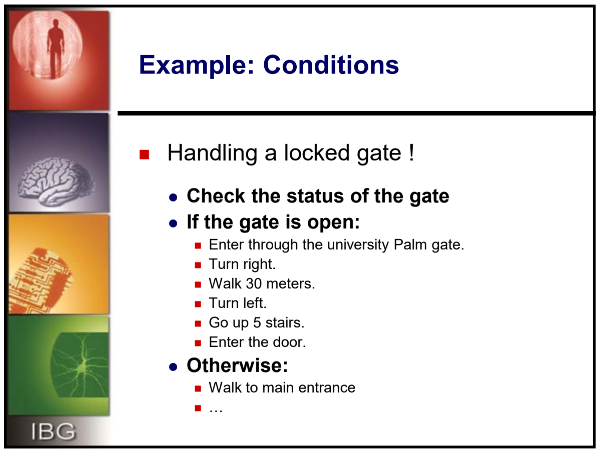
IBG



Sequential program

- Same old example…
- Navigating from the university Palm gate to the Gonda brain research building:
  - Enter through the university Palm gate.
  - Turn right.
  - Walk 30 meters.
  - Turn left.
  - Go up 5 stairs.
  - Enter the door.

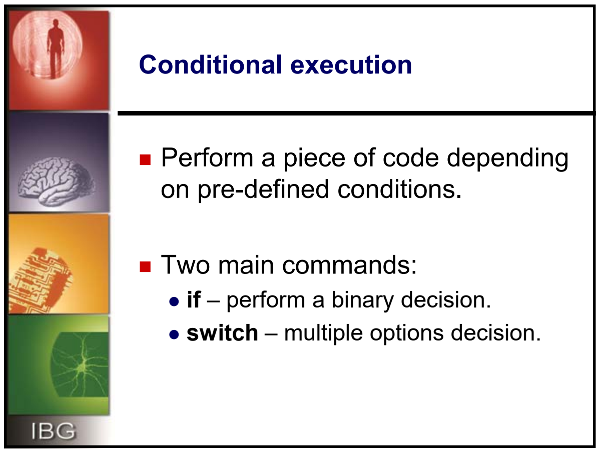
IBG



Example: Conditions

- Handling a locked gate !
  - **Check the status of the gate**
  - **If the gate is open:**
    - Enter through the university Palm gate.
    - Turn right.
    - Walk 30 meters.
    - Turn left.
    - Go up 5 stairs.
    - Enter the door.
  - **Otherwise:**
    - Walk to main entrance
    - …

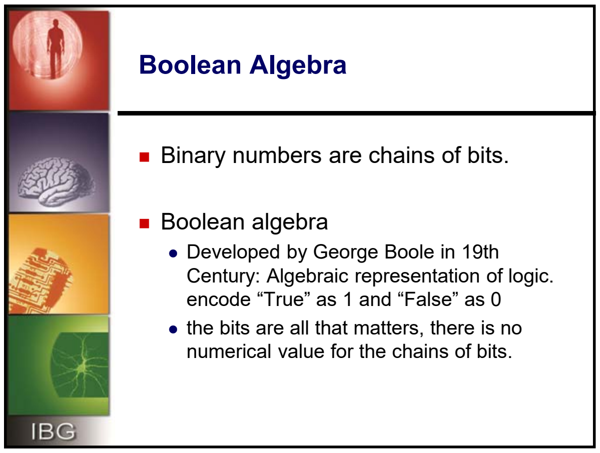
IBG



Conditional execution

- Perform a piece of code depending on pre-defined conditions.
- Two main commands:
  - **if** – perform a binary decision.
  - **switch** – multiple options decision.

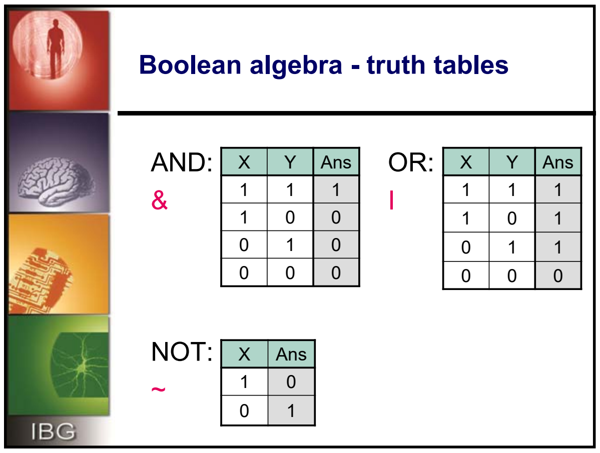
IBG



Boolean Algebra

- Binary numbers are chains of bits.
- Boolean algebra
  - Developed by George Boole in 19th Century: Algebraic representation of logic. encode “True” as 1 and “False” as 0
  - the bits are all that matters, there is no numerical value for the chains of bits.

IBG



Boolean algebra - truth tables

AND: &

X	Y	Ans
1	1	1
1	0	0
0	1	0
0	0	0

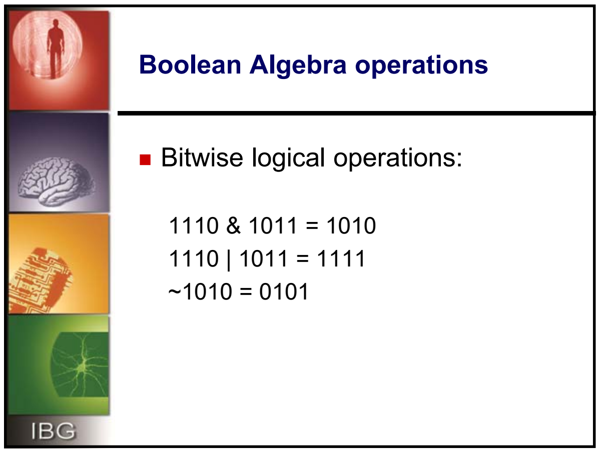
OR: |

X	Y	Ans
1	1	1
1	0	1
0	1	1
0	0	0

NOT: ~

X	Ans
1	0
0	1

IBG

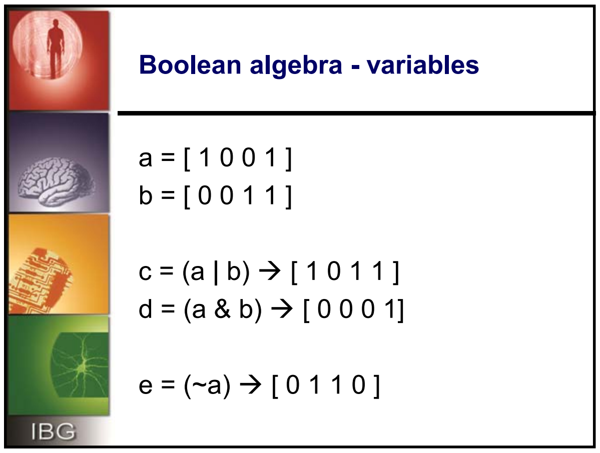


Boolean Algebra operations

- Bitwise logical operations:

  1110 & 1011 = 1010
  1110 | 1011 = 1111
  ~1010 = 0101

IBG



Boolean algebra - variables

a = [1 0 0 1]
b = [0 0 1 1]

c = (a | b) $\rightarrow$ [1 0 1 1]
d = (a & b) $\rightarrow$ [0 0 0 1]

e = (~a) $\rightarrow$ [0 1 1 0]

IBG

Command: *if*

- Evaluation of a **logical** expression and execution of a group of commands when **condition** is TRUE.

Syntax:	**Example**

if condition
    commands…
end
	if a > 5
	    a = a + 1;
	end

Example: *if*

```
y=0;
if x > 10
  y = x;
end
```

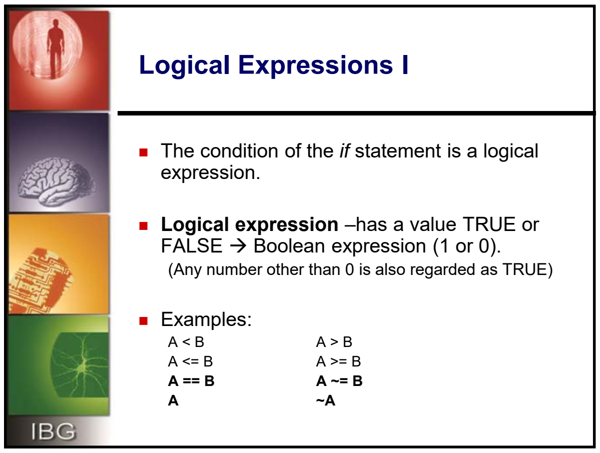
- What is *y* when x is 2 ?
- What is *y* when x is 13 ?

Indentation

- **Indentation** → placing text farther to the right to separate it from surrounding text, used to format source code in order to improve its readability.

- Anything inside a conditional statement or a repetitive statement is pushed in.

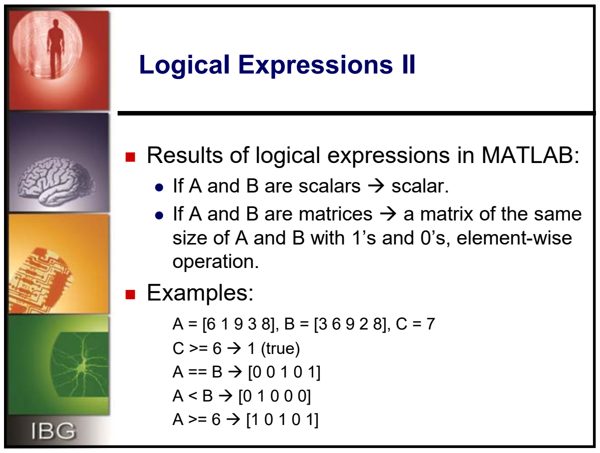
```
if a < 1          if a < 1
b=0;                  b=0;
c=3        →          c=3;
end               end
d=4;              d=4;
```



Logical Expressions I

- The condition of the *if* statement is a logical expression.
- **Logical expression** –has a value TRUE or FALSE → Boolean expression (1 or 0).
  (Any number other than 0 is also regarded as TRUE)
- Examples:

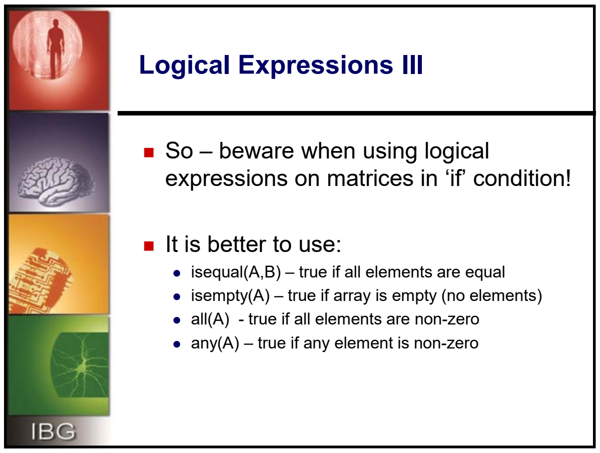
A < B	A > B
A <= B	A >= B
A == B	**A ~= B**
A	**~A**



Logical Expressions II

- Results of logical expressions in MATLAB:
  - If A and B are scalars → scalar.
  - If A and B are matrices → a matrix of the same size of A and B with 1's and 0's, element-wise operation.
- Examples:

  A = [6 1 9 3 8], B = [3 6 9 2 8], C = 7
  C >= 6 → 1 (true)
  A == B → [0 0 1 0 1]
  A < B → [0 1 0 0 0]
  A >= 6 → [1 0 1 0 1]



Logical Expressions III

- So – beware when using logical expressions on matrices in 'if' condition!
- It is better to use:
  - isequal(A,B) – true if all elements are equal
  - isempty(A) – true if array is empty (no elements)
  - all(A) - true if all elements are non-zero
  - any(A) – true if any element is non-zero

Logical Operators

And: &&	X && Y
Or: ||	X || Y

- Can be applied only on 2 scalar variables (or logical expressions), not on arrays.
- Examples:

```
A = 3; B = -4; C = 0;
(A < 5) && (B < 5) → 1
A && C → 0
A || C → 1
(A > 2) && (C < 3) → 1
```

Conditions

- A complex logical expression:

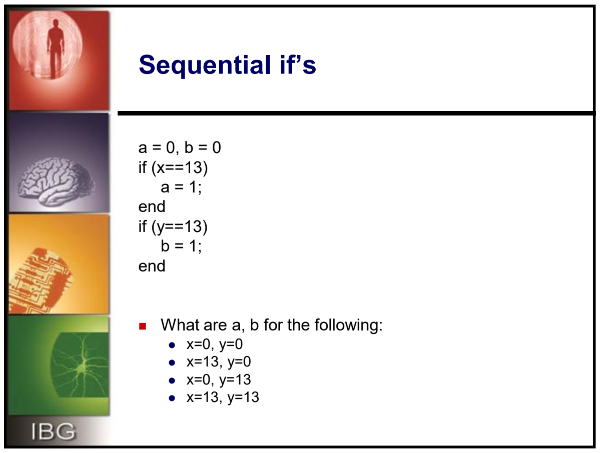
```
if A > 2 && C >=4 || D>A
   commands…
end
```

- **Use** () for clarification.

```
if ((A > 2) && (C >=4)) || (D>A)
     commands…
end
```

Precedence

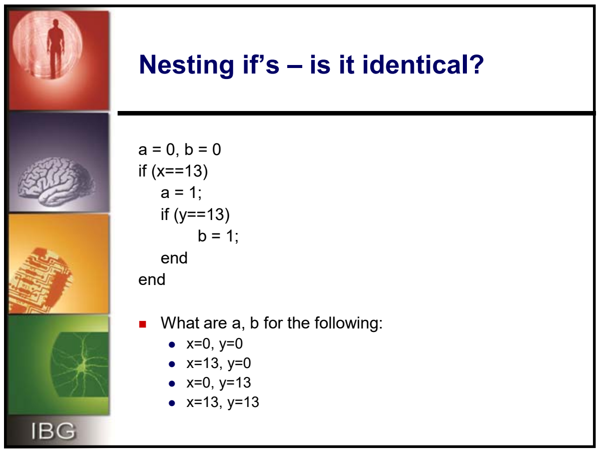
1. Parentheses ()
2. power (.^ or ^)
3. Unary plus (+), unary minus (-), logical negation (~)
4. Multiplication (.* or *), division (./ or /)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8. Element-wise AND (&)
9. Element-wise OR (|)
10. Short-circuit AND (&&)
11. Short-circuit OR (||)



Sequential if’s

```
a = 0, b = 0
if (x==13)
    a = 1;
end
if (y==13)
    b = 1;
end
```

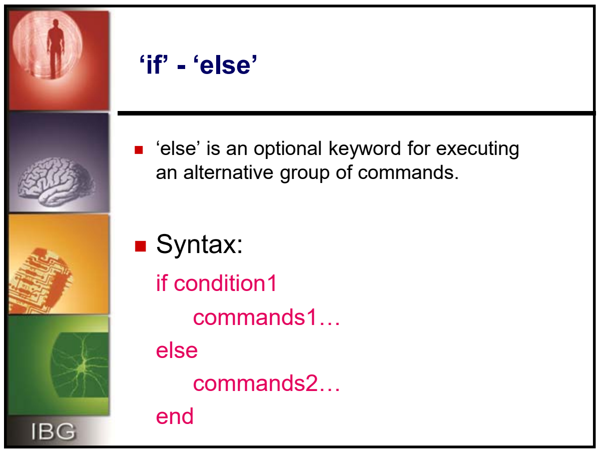
- What are a, b for the following:
  - x=0, y=0
  - x=13, y=0
  - x=0, y=13
  - x=13, y=13



Nesting if’s – is it identical?

```
a = 0, b = 0
if (x==13)
    a = 1;
    if (y==13)
          b = 1;
    end
end
```

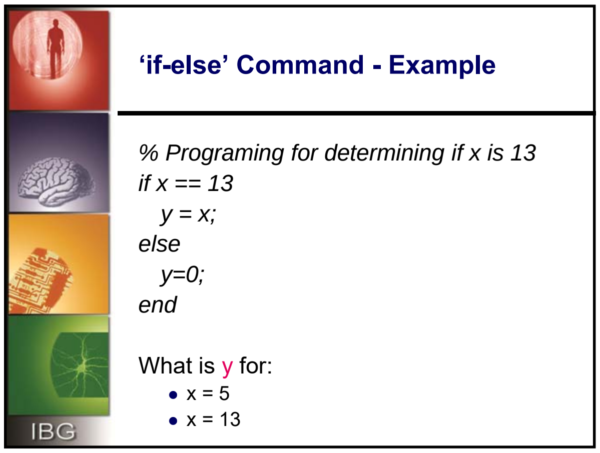
- What are a, b for the following:
  - x=0, y=0
  - x=13, y=0
  - x=0, y=13
  - x=13, y=13



‘if’ - ‘else’

- ‘else’ is an optional keyword for executing an alternative group of commands.
- Syntax:

```
if condition1
      commands1…
else
      commands2…
end
```

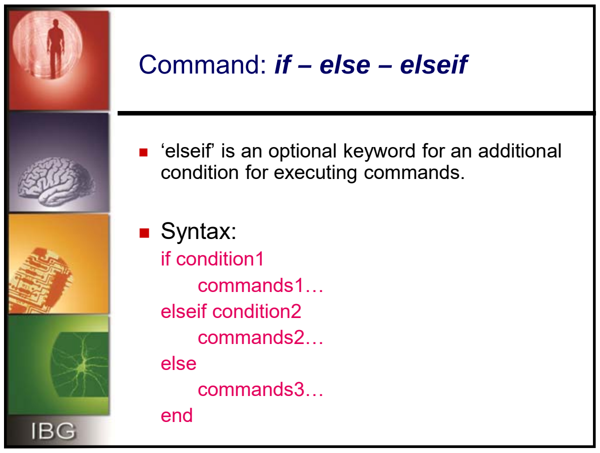


‘if-else’ Command - Example

```
% Programing for determining if x is 13
if x == 13
   y = x;
else
   y=0;
end
```

What is y for:

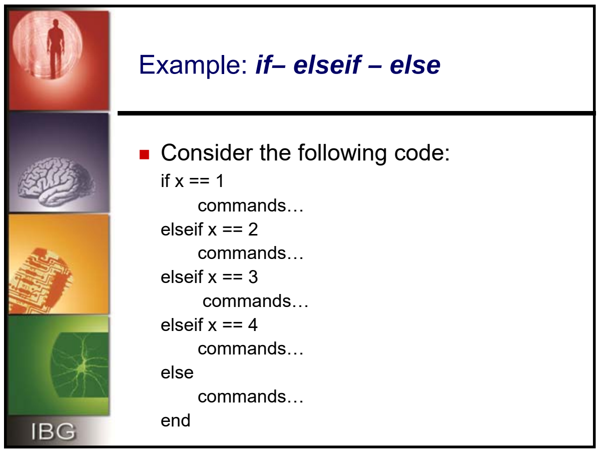
- x = 5
- x = 13



Command: *if – else – elseif*

- ‘elseif’ is an optional keyword for an additional condition for executing commands.
- Syntax:

```
if condition1
     commands1…
elseif condition2
     commands2…
else
     commands3…
end
```



Example: *if– elseif – else*

- Consider the following code:

```
if x == 1
     commands…
elseif x == 2
     commands…
elseif x == 3
      commands…
elseif x == 4
     commands…
else
     commands…
end
```

Command: ***Switch - Case***

- The ‘switch’ statement executes groups of commands based on the value of variable or expression.
- Syntax:

```
switch expression
      case 1
               commands…
      case 2
               commands…
      otherwise
               commands…
end
```

Example: ***Switch - Case***

```
switch x
    case 1
          y = 4;
    case 2
          y = 3;
    case 3
          y = 5;
    case 4
          y = 1;
    otherwise
          y = 0;
end;
```

Logical indexes
Back to variables…

- In addition to regular indexes, a logical index may be used, using true/false values for each element.

- Example:

```
>> myVec = [3 5 1 7 2 8];
>> midInd = (myVec > 2 & myVec< 8);
midInd =
   1    1    0    1    0    0
>> midVec = myVec(midInd)
midVec =
   3    5    7
```