SIT105 - Critical Thinking and Problem Solving for IT

Class 07

Part 1 – Building and Checking Algorithms

Part 2 – Selection Statements

A/Prof. Xiao Liu

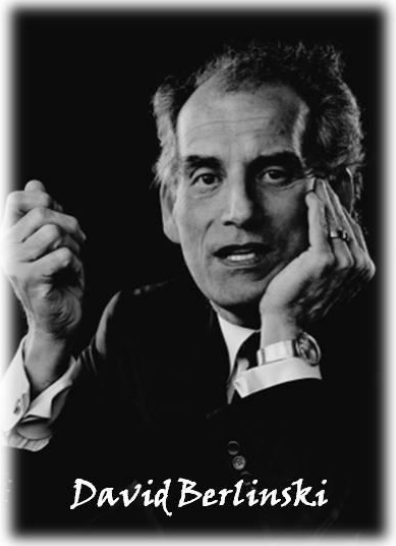
xiao.liu@deakin.edu.au

QUOTE OF TODAY

More than sixty years ago, mathematical logicians, defined the concept of an algorithm, and gave content to the ancient human idea of an effective calculation. Their definitions led to the creation of the digital computer.

Quote by: David Berlinski
American philosopher, educator, and author.

Author of the book:
The Advent of the Algorithm: The 300-Year Journey from an Idea to the Computer.



Part 1 Content

1. Defining Diagram
2. Algorithm Design
3. Desk Check

Reminder: What is an Algorithm?

Algorithms have been commonly defined in simple terms as:

"instructions for completing a task"

They've also been called "recipes". In *The Social Network movie*, an algorithm is what Zuckerberg needed to make Facemash work.

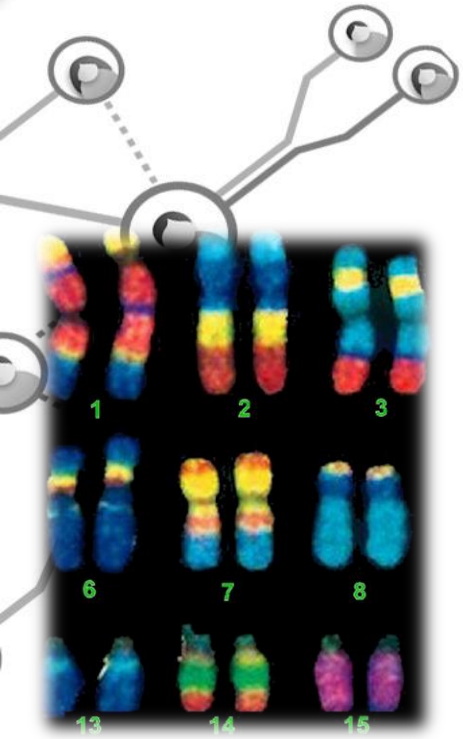
If you saw the movie, you probably remember seeing what looked like a scribbly equation on a window in Mark's dorm room.

These were patterns for completing a specific task in an efficient way!

In this class we will use Pseudocode to write algorithms! This is a high level, structured English approach.

Why Study Algorithms?

They are used everywhere!



- **Internet**. Web search, packet routing, distributed file sharing, ...
- **Biology**. Human genome project, protein folding, …
- Computers. Circuit layout, databases, caching, networking, compilers, …
- **Computer graphics**. Movies, video games, virtual reality, …
- **Security**. Cell phones, e-commerce, voting machines, …
- **Multimedia**. MP3, JPG, DivX, HDTV, face recognition, …
- **Social networks**. Recommendations, news feeds, advertisements, …
- **Physics**. N-body simulation, particle collision simulation

Source: Jon Kleinberg And Éva Tardos. Princeton University

Forget the Mother of All Bombs — fear the Mother of All Algorithms

The Mother of All Bombs made news last week after the U.S. military dropped its most powerful non-nuclear bomb at a site in Afghanistan.

The Pentagon has put artificial intelligence at the center of its strategy to maintain the United States' position as the world's dominant military power. It is spending billions of dollars to develop what it calls autonomous and semiautonomous weapons and to build an arsenal stocked with the kind of weaponry.

The US Defense Department is designing robotic fighter jets that would fly into combat alongside manned aircraft. It has tested missiles that can decide what to attack, and it has built ships that can hunt for enemy submarines, stalking those it finds over thousands of miles, without any help from humans. ...

Question: If future presidents and Pentagons trusted algorithms to make such decisions regarding conflicts between two nations, is there a chance that war's could start without human involvement?

Defining Diagrams

Where to begin? Defining the Problem!

- We talked last week about the 7 steps to developing a program.
- The **first step** is defining the problem!
  - ***Investigate it until you know the requirements***
- Often we need to find out additional specific information to work out the problem entirely.
- To assist with analysis of your problem we can use a **defining diagram**.
  - Inputs
  - Outputs
  - Processing

Kindly, slowly, here, often, and very are examples of adverbs. Adverbs modify verbs, adjectives, or other adverbs.

Defining Diagram

1. We identify these from the problem statement:
   - Inputs (nouns and adjectives)
   - Outputs (nouns and adjectives)
   - Processing (verbs and adverbs)
2. Tabulate them!

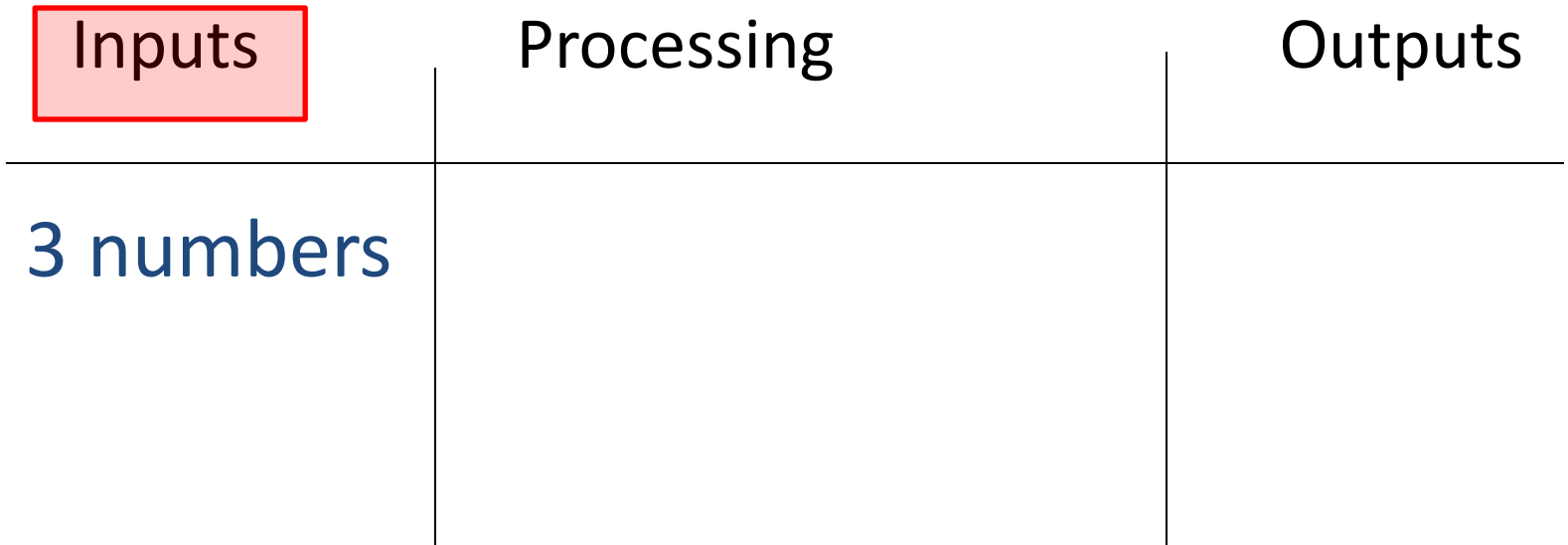
Inputs	Processing	Outputs

Defining Diagram (example - inputs)

Problem statement

A program is required to read three numbers, add them together, and print their total.

Defining Diagram



Defining Diagram (example - processing)

Problem statement

A program is required to read three numbers, add them together, and print their total.

Defining Diagram

Inputs	Processing	Outputs
3 numbers	- read 3 numbers - add these numbers - print the total	

Defining Diagram (example - outputs)

Problem statement

A program is required to read three numbers, add them together, and print their total.

Defining Diagram

Inputs	Processing	Outputs
3 numbers	- read 3 numbers - add these numbers - print the total	total

Algorithm Design

- the algorithm is based on the Defining Diagram
- the algorithm is written in Pseudocode

Template:

- Beginning: start the algorithm with its **name**
- Middle: **statements**; indented for readability
- End: finish using the **END** statement

```
module OuterMod
  module InnerMod
    def self.outermethod
    end
    module AnotherInnerMod
      def self.z
      end
      class InnerClass
        class AnotherClass
          def x
          end
          def y
          end
        end
      end
    end
  end
end
```

Algorithm Design (**example 1**)

Algorithm (Version 1)

```
ADD_THREE_NUMBERS
        READ n1
        READ n2
        READ n3
        total = n1 + n2 + n3
        PRINT total
END
```

Algorithm Design (**example 2**)

Algorithm (Version 2)

```
ADD_THREE_NUMBERS
        READ n1, n2, n3
        total = n1 + n2 + n3
        PRINT total
END
```

Algorithm Design (**example 3**)

Algorithm (Version 3)

```
ADD_THREE_NUMBERS
        READ n1, n2, n3
        PRINT n1 + n2 + n3
END
```

Which version (1, 2 or 3) is “better”?

Algorithm Design (**example 3**)

Which version (1, 2 or 3) is “better”?

Time? If they are written into computer code. The compiler will turn all 3 of these examples into the same result, so processing time generally will be the same.

Maintenance? You don’t want to mix up your logical concepts.

In version 3: its doing 2 things, in just 2 lines.

Try to separate them out as much as possible, so its easier to fix bugs.

The first example is best.

Desk Check

Desk Check			
Line Number	x	Conditions	Input / Output
1			
2			
3	1		
4		1 < 3 is True	
5			Hello
6	1 + 1 = 2		
4		2 < 2 is True	
5			Hello
6	2 + 1 = 3		
4		3 < 3 is False	
8			

Steps in desk checking!

1. Choose good test data (valid and invalid)
2. Determine **expected results** (without using algorithm)
3. Tabulate all variable names to record their values
4. Step through the algorithm one statement at a time and record all **actual results** in this table
5. **Compare** actual results to expected results
6. If errors detected, fix them and go-to 4

Desk Check (example)

Desk check - ***for adding 3 numbers***

Test ID	Test Data				Expected Results
1	0	6	15		21
2	-1	4	-2.5		0.5
3	one	two	three		error
4	1	2	three		error
5	1	2	3	4	6
6	10	11			program blocked, waiting for input

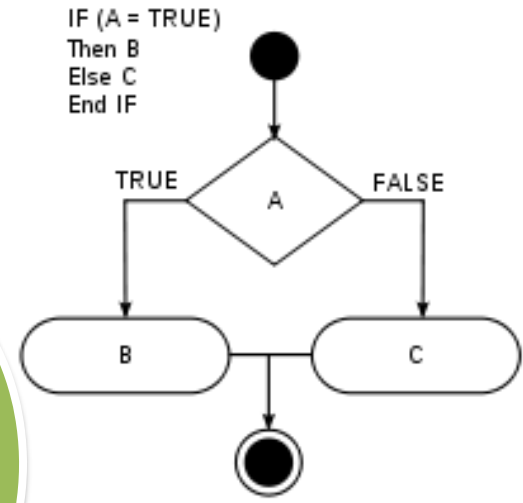
Desk Check (example) – Algorithm Test

Test 1 data: 0, 6, 15

statement	n1	n2	n3	total	output
READ	0				
READ	0	6			
READ	0	6	15		
assignment	0	6	15	21	
PRINT	0	6	15	21	21

Compare

expected result (21) **equals** actual result (21)



Part 2 Content

1. Selection Concept
2. IF Statement
3. IF … ELSE Statement
4. Nested IF Statements
5. CASE Statement
6. CASE Statement with flexibility

Bill Gates explains If statements! (video)

Selection Concept

- Selection statements are used to:
  - **select and evaluate one embedded group of statements**
  - and so ignore other embedded groups.

- The selection depends on the value of:
  - an **explicit** Boolean expression as used in **IF** statements
  - an **implicit** Boolean expression as used in **CASE** statements

You don't see the Boolean expression here, its hidden away!

Boolean data type is a data type, having two values (usually denoted true and false)

IF Statement

- Format

```
IF Boolean expression THEN
    statements
ENDIF
```

The green block is executed after the blue block, but only when the blue block has a TRUE value

- The value of a Boolean expression is True or False.

IF Statement - Example 1

```
age = 21
IF age >= 18 THEN
        PRINT “voter”
ENDIF
```

The Age variable is set to 21.
The expression checks if that value is
greater than or equal to 18.
If it is then Print out some text to the screen.

IF … ELSE Statement

- Format

```
IF Boolean expression THEN
        statement1
ELSE
        statement2
ENDIF
```

The green block is executed after the blue block, but only when the blue block has a TRUE value

The yellow block is executed after the blue block, but only when the blue block has a FALSE value

IF … ELSE Statement - Example 1

```
age = 15

IF age >= 13 AND age <=19 THEN
        PRINT “Teenager.”
ELSE
        PRINT “Not a teenager.”
ENDIF
```

The Age variable is set to 15.
The expression checks if that value is greater than or equal to 13 or less than or equal to 19.
If it is then Print out some text to the screen if it meets those values.
If its not then Prints out some alternative text.
Then ends the statement.

Nested IF Statements

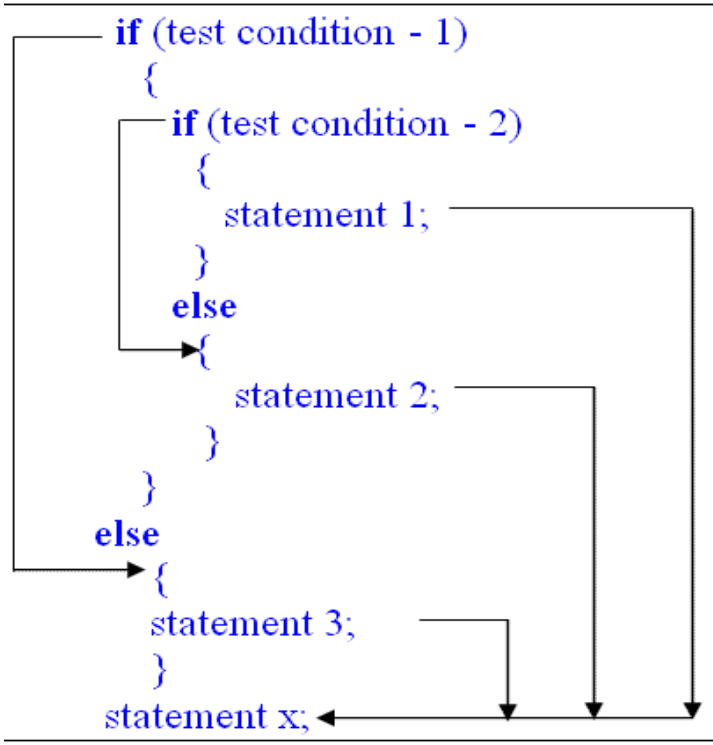
- In general, the **statements** clause of an IF statement can be any kind of statement.

  IF Boolean expression THEN

  statements

  ENDIF

- When this statement is another IF statement,
  it is said to be **nested** within the outer IF statement.



Nested IF Statements - Example 1

```
IF Boolean expression1 THEN
    IF Boolean expression2 THEN
        statement 1
    ELSE
        statement 2
    ENDIF
ENDIF
```

The nested IF section is in green in the diagram.

Nested IF Statements - Example 2

```
IF Boolean expression1 THEN
    IF Boolean expression2 THEN
        IF Boolean expression3 THEN
            statement1
        ELSE
            statement2
        ENDIF
    ELSE
        statement3
    ENDIF
ENDIF
```

Here we have 2 IF statements in total. Each one is dependant on the Boolean result.

Nested IF Statements - Example 3


```
IF Boolean expression 1 THEN
    IF Boolean expression 2 THEN
        IF Boolean expression 3 THEN
            statement 1
        ELSE
            statement 2
        ENDIF
    ELSE
        statement 3
    ENDIF
ELSE
    IF Boolean expression 4 THEN
        statement 4
    ELSE
        statement 5
    ENDIF
ENDIF
```

CASE Statement

- A CASE statement:
  - is just a special form of **several IF statements!**
  - usually provides **more readability** than its equivalent IF statements.

- **Any CASE statement can be easily rewritten using IF statements.**
- **However most IF statement's can't be easily rewritten using a CASE statement.**

CASE Statement

Format:

```
CASE variable
    value1 : statements_1
    value2 : statements_2
    value3 : statements_3
    ...
    OTHER : statements_N
ENDCASE
```

Working from top to bottom, a block of **statements** is executed when the value of the **variable** matches a **value**.

Once a match is found, do not look for additional matches.

For no matches, execute the **statements** in the **OTHER** clause.

CASE Statement – Example 1

```
x = 5
CASE  x
            0 :       PRINT “ZERO”
            1 :       PRINT “ONE”
            10 :      PRINT “10, 11 or 12”
            11 :      PRINT “10, 11 or 12”
            12 :      PRINT “10, 11 or 12”
            OTHER:    PRINT “x is not 0, 1, 10, 11 or 12”
ENDCASE
```

CASE Statement – Example 2

```
IF x = 0 THEN
        PRINT "ZERO"
ELSE
        IF x = 1 THEN
                PRINT "ONE"
        ELSE
                IF x >= 10 AND x <= 12
                        PRINT "10, 11 or 12"
                ELSE
                        PRINT "x is not 0, 1, 10, 11 or 12"
                ENDIF
        ENDIF
ENDIF
```

This example is equivalent to the CASE statement in Example 1 but written with nested IF's

Cases could be based on ranges or sets, instead of a single value.

CASE Statement with Flexibility

Format:

```
CASE expression
        values1 :       statements_1
        values2 :       statements_2
        values2 :       statements_3
        ...
        OTHER :         statements_N
ENDCASE
```

Example

Cases could be based on ranges or sets, instead of a single value.

CASE Statement with Flexibility - Example

An equation instead of a set value

```
CASE 2x² + y²
        0 :          PRINT "ZERO"
        1 :          PRINT "ONE"
        2, 4, 5, 8 : PRINT "2, 4, 5 or 8"
        5* :         PRINT "Number begins with a 5"
        10 to 12 :   PRINT "10, 11 or 12"
        OTHER :      PRINT "an unexpected value"
ENDCASE
```

ARRAYS – Find the Max

Data_array

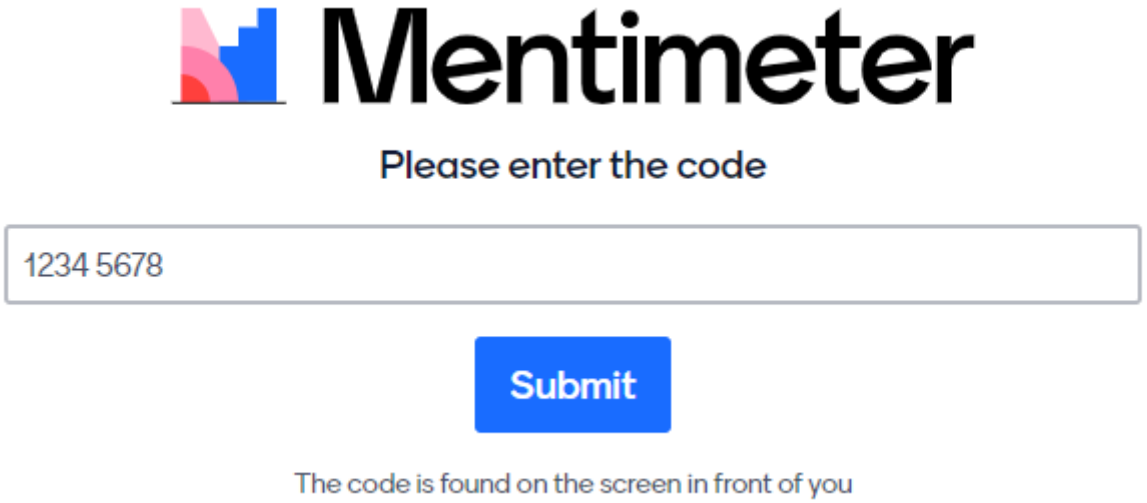
Columns

Rows	0	1	2	3
0	1	4	1	7
1	2	6	7	1
2	7	1	6	4
3	2	8	3	6

value = data_array[rowIndex, colIndex]

Let’s go to www.menti.com

- 5mins! Work with your group for the solution



ARRAYS – Find the Max (General)

Data_array

Rows \ Columns	0	1	2	3
0	1	4	1	7
1	2	6	7	1
2	7	1	6	4
3	2	8	3	6

```
INITIALIZE row_max = 0 // Store the current max value of the row
INITIALIZE row_min = 0  // Store the current min value of the row
INITIALIZE rowCount = 3 // Store how many rows there are
INITIALIZE colCount = 3  // Store how many columns there are
INITIALIZE data_array = array[colCount, rowCount] // Our array to store our dataset
INITIALIZE value = 0 // Store the current value we are looking at
INITIALIZE rowIndex = 0 // Keep track of which row we are on
INITIALIZE colIndex = 0  // Keep track of which column we are on

FOR rowIndex TO rowCount // Loop through each row
    colIndex = 0 // Reset the colIndex back to 0
    FOR colIndex TO colCount THEN // Loop through each column
            value = data_array[rowIndex, colIndex] // Get the value from the array
            IF value > row_max THEN
                row_max = value
            END IF
        END FOR

        // What can we do here with our row_max and row_min?

END FOR
```

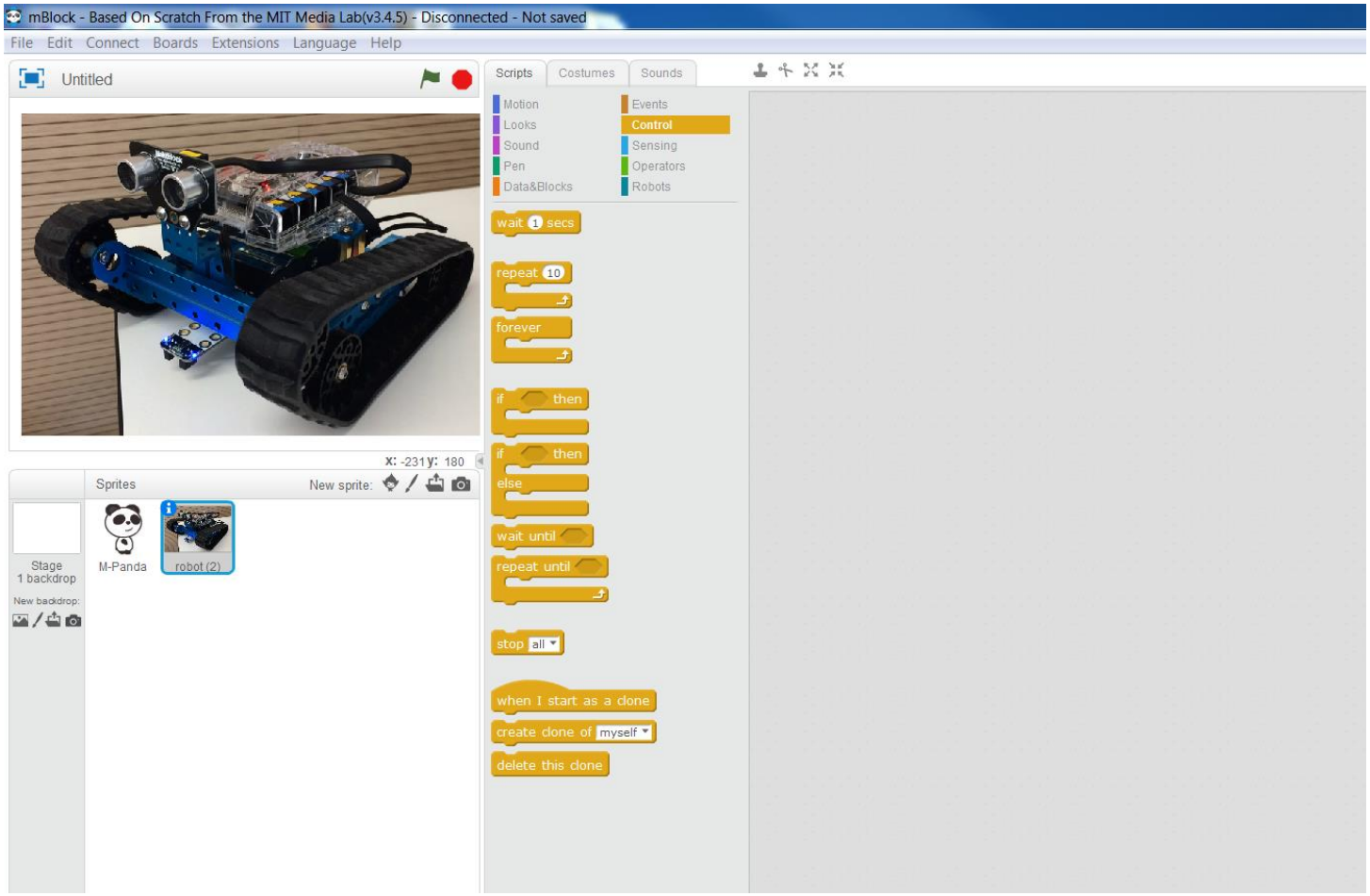
For next week…

Lets try to code Wally with some simple Algorithms!

2. Following on from #1 make me repeat this 3 times! (without copying the code 3 times)

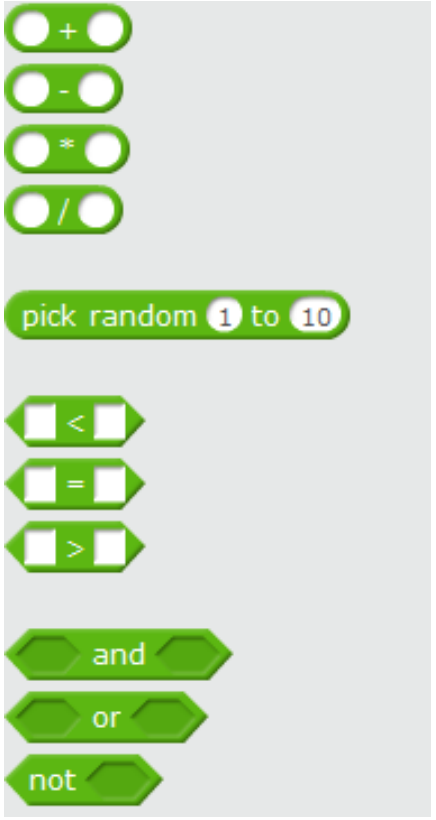
1. Using sequential steps make me go around the front of the room in a square formation!

Wally Bot – How to code it?
mBlock (Based on Scratch from MIT Media Lab)



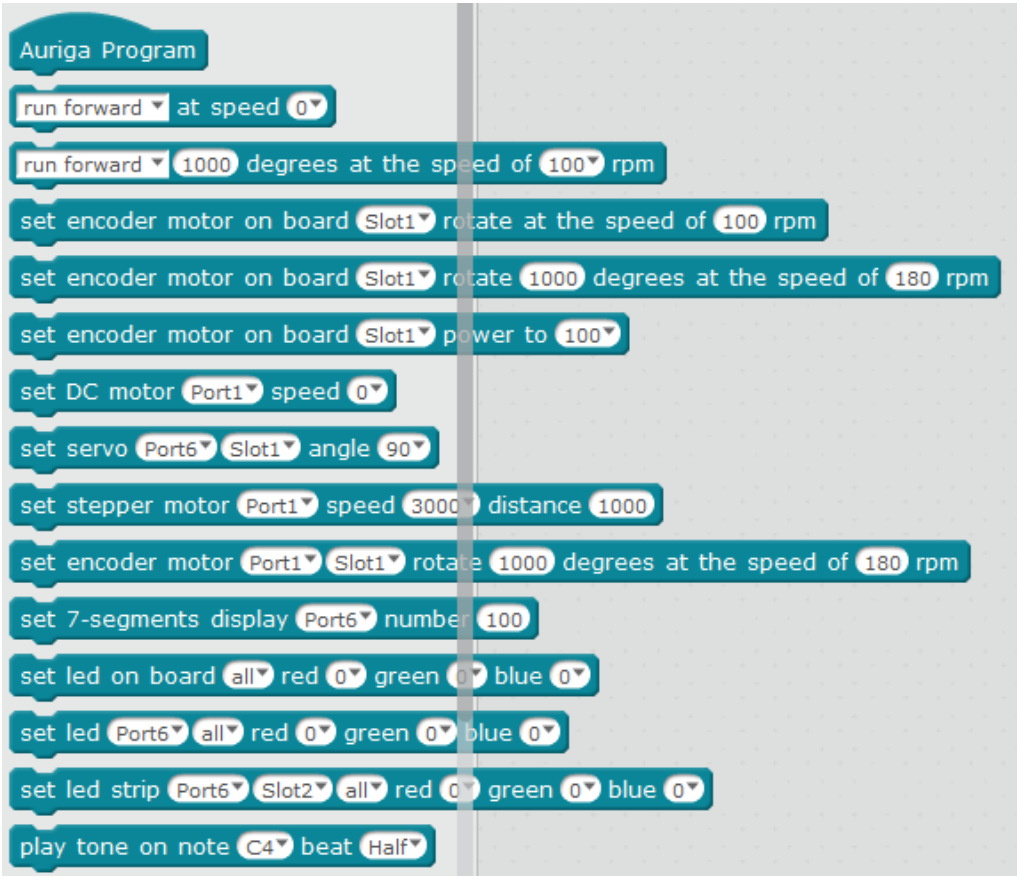
Wally Bot – How to code it?

Operators



Control

Robotic functions



Demo