

CISC 1003 - EXPLORING ROBOTICS

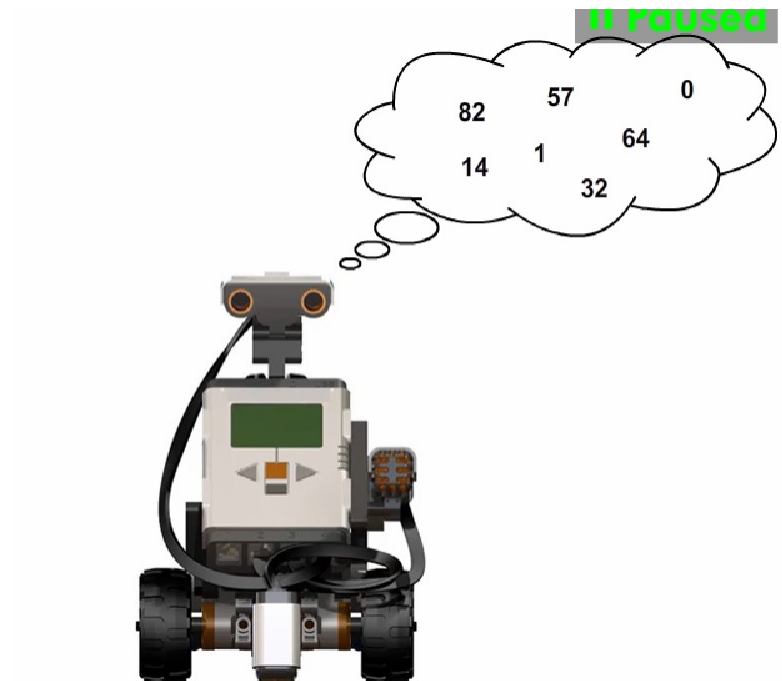


ROBOT DECISION MAKING

CISC1003

How a Robot Thinks

- Robot can 'learn' the world using their sensors
- Sensors return data in number format



How a Robot Thinks

- The robot can answer 'yes' or 'no' questions
- Example:
 - Is the touch sensor bumped?
 - Is the audio level in the room above 50%?
- This ability is based on a special logic
 - Called '***Boolean logic***'

How a Robot Thinks

- Programmers can give the robot its decision-making capability
 - By combining the numbers provided by the sensor with robot ability to answer questions
- This requires the following:
 - Robot is programmed to ask questions
 - Act one way if the answer is 'yes'
 - and another if the answer is 'no'

How a Robot Thinks

- Boolean operators are used when asking the questions, such as:
 - $<$ 'less than' ,
 - $>$ 'more than',
 - $==$ "equal to"
 - etc.

How a Robot Thinks

- Example: We want the robot to stop moving before it runs into a wall
 - Use the feedback from an ultrasonic sensor
 - Use 'less than' operator
 - with a certain distance threshold
 - E.g., 10 inch
- This will result in a program that moves the robot until it detects an obstacle
 - Within the distance specified (10 inch)

How a Robot Thinks

- How does the program work?
- The robot moves forward
- It repeatedly asks the questions:
 - “Am I 10 inch away from anything?”
- If the answer is no, the robot continues moving forward
 - If the answer is ‘yes’, it stops

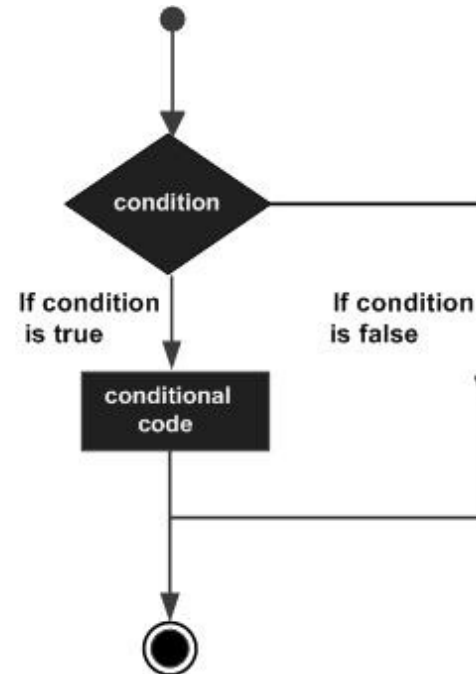
Conditional Statements

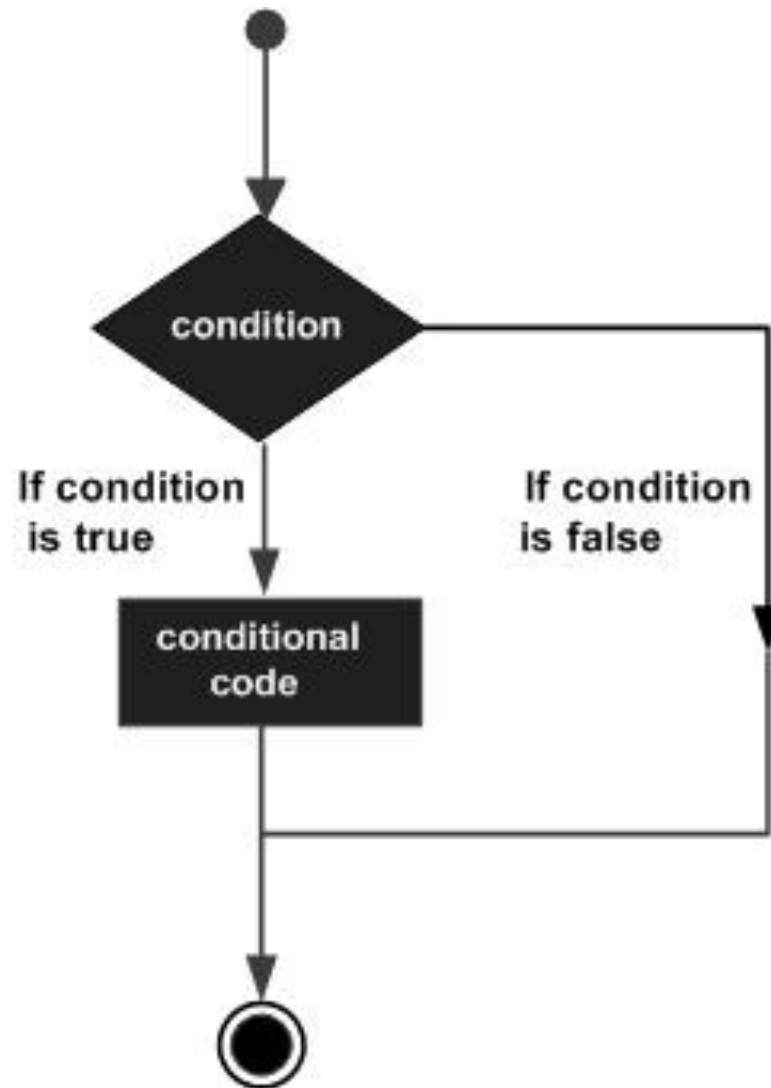
- The parts of the program where the robot choose an action
 - Depending on a certain condition
 - Typically expressed as Boolean values

Decision Making

- Decision making structures require one or more conditions to be evaluated by the program
 - along with a statement or statements to be executed if the condition is determined to be true
 - and optionally, other statements to be executed if the condition is determined to be false

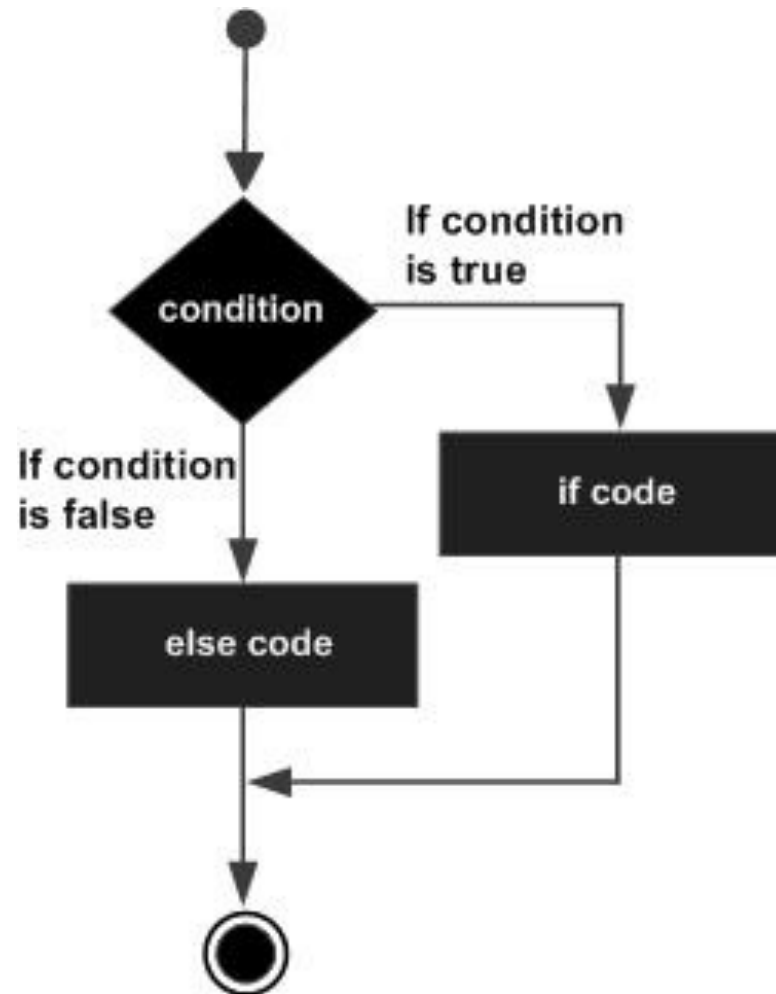
- below is the general form of a typical decision making structure
 - found in most of the programming languages





Conditional Statements

- An **if** statement can be followed by an optional **else** statement
 - which executes when the Boolean expression is false
- If the Boolean expression evaluates to **true**, then the **if block** will be executed
 - otherwise, the **else block** will be executed.

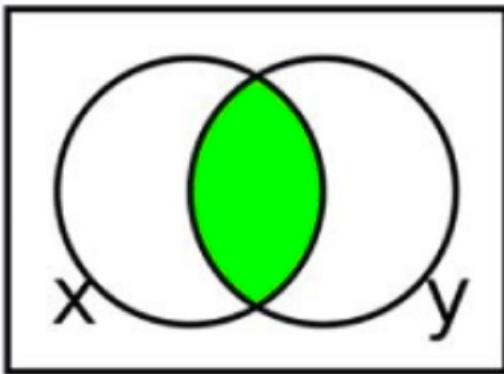


Boolean Expressions

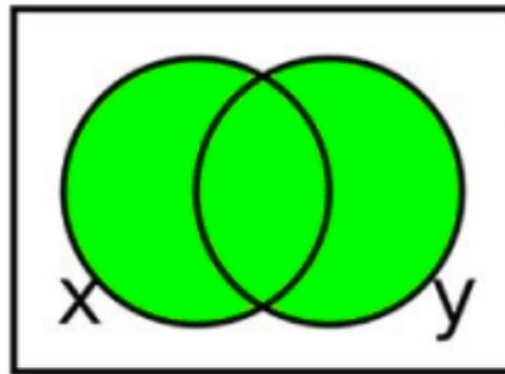
- Boolean expression: A logical statement that results in a boolean value
 - either be True or False
- Boolean values are expressed by:
 - True
 - Can also be expressed by the number 1 and 'Yes' value
 - False
 - Can also be expressed by the number 0 and 'No' value

Boolean Algebra Operations

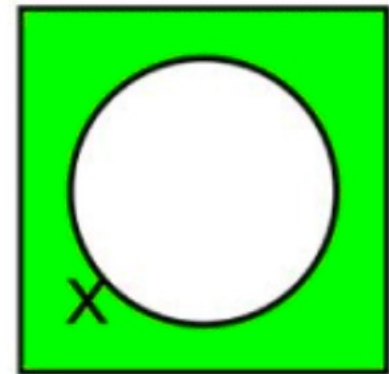
- There are three basic operations:
 - OR operation
 - AND operation
 - NOT operation



$$x \wedge y$$



$$x \vee y$$

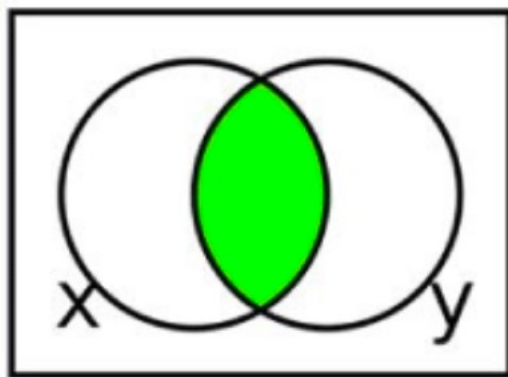


$$\neg x$$

Boolean Algebra Operations

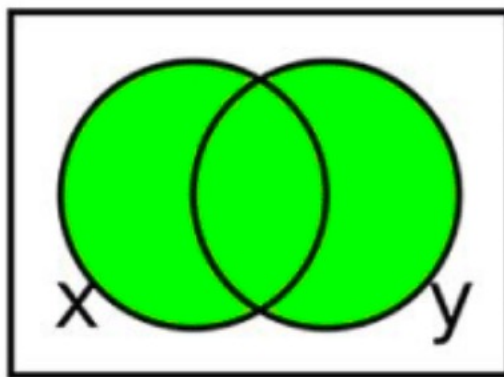
- There are three basic operations:

AND



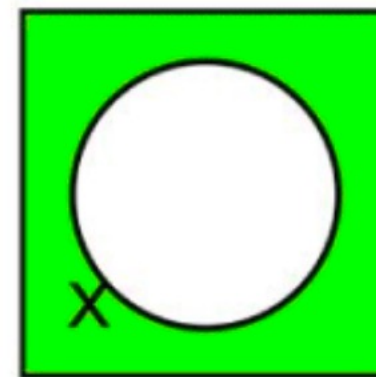
$$x \wedge y$$

OR



$$x \vee y$$

NOT



$$\neg x$$

Truth Tables

A	$\neg A$
True	False
False	True

A	B	$A \cap B$ (AND)	$A \cup B$ (OR)
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Precedence of operations

- Operators are used in order of precedence
 - To change order, use brackets for operations that should be done first

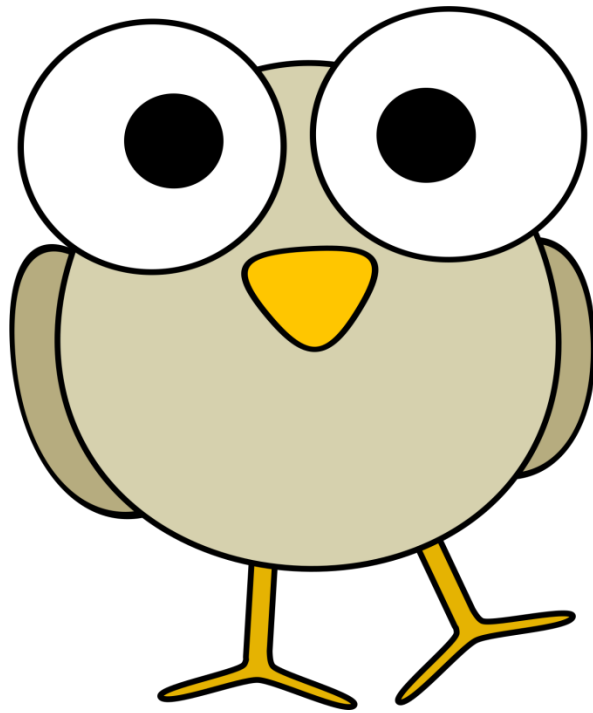
Operator	Symbol	Precedence
NOT	' (or) \neg	Highest
AND	. (or) \wedge	Middle
OR	+ (or) \vee	Lowest

Precedence of operations

- Example:
 - $\neg 0 \cup 1 = ?$
 - $\neg(0 \cup 1) = ?$

Precedence of operations

- Example:
 - $\neg 0 \cup 1 = 1 \cup 1 = ?$
 - $\neg(0 \cup 1) = \neg 1 = ?$



??

Precedence of operations

- Example:
 - $\neg 0 \cup 1 = 1 \cup 1 = 1$
 - $\neg(0 \cup 1) = \neg 1 = 0$

Summary

- We can create conditional statements
 - by combining sensor output and Boolean operators
- This allows the robot to make decisions

How a robot thinks

- What kinds of questions can a robot ask?
 - “yes” or “no” questions
 - Questions that have only two possible answers
- What can a robot do with the answer to the question?
 - Use the answers to choose between two different actions
 - E.g., move forward or stop



LOOPS

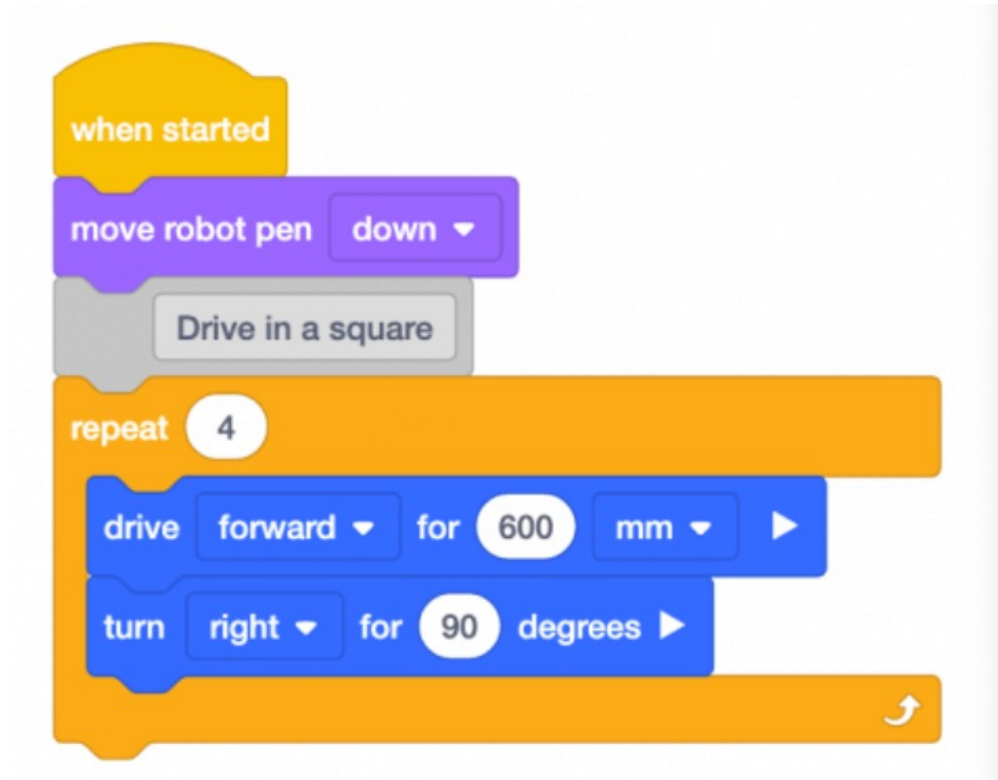
Loops

- We started seeing loops in Lab 4
- Almost all the programming languages provide a concept called **loop**,
 - which helps in executing one or more statements up to a desired number of times.
 - High-level programming languages provide various forms of loops
 - which can be used to execute one or more statements repeatedly.

Loops in VR.VEX

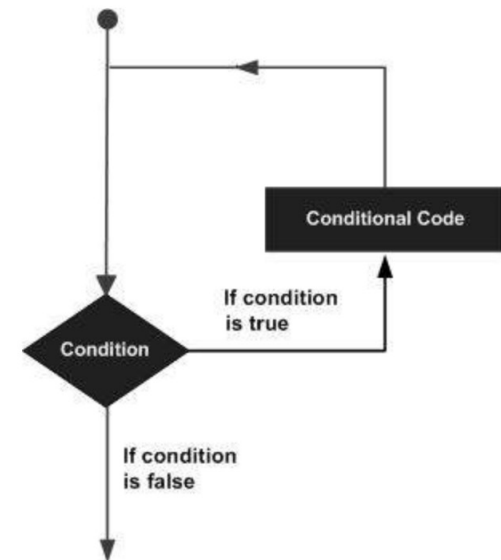
- Blocks from the Control category such as the [Repeat], [Repeat until], and [Forever] blocks
- These blocks repeat behaviors on a loop.

Example

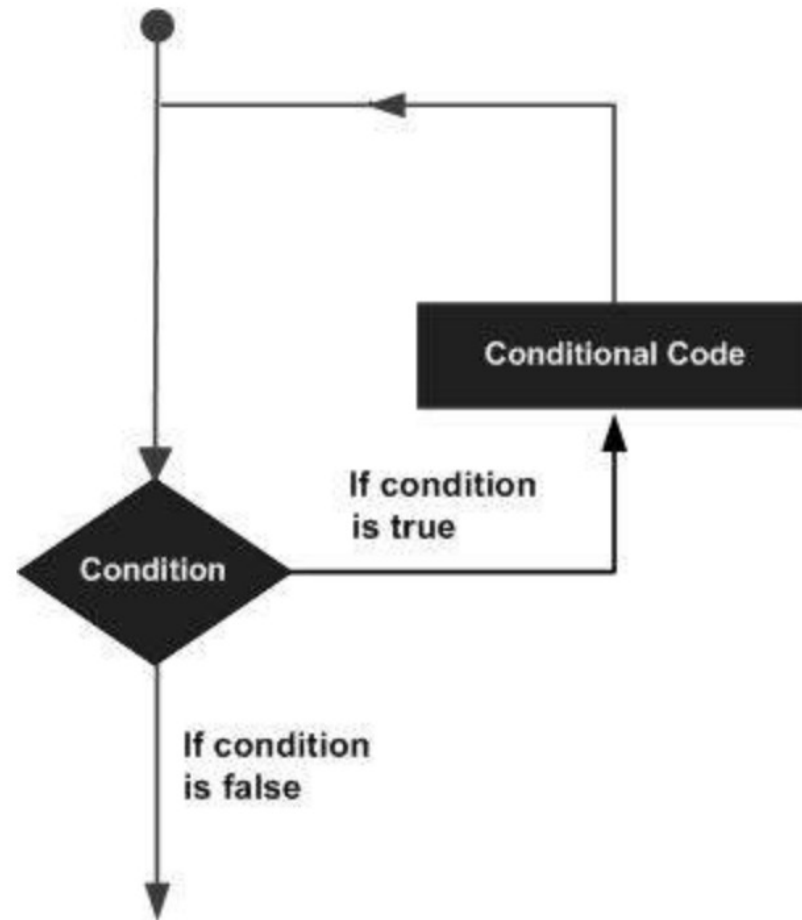


Loops

- a loop statement allows us to execute a statement or group of statements multiple times.
- Below is the general form of a loop statement in most of the programming languages



Loops

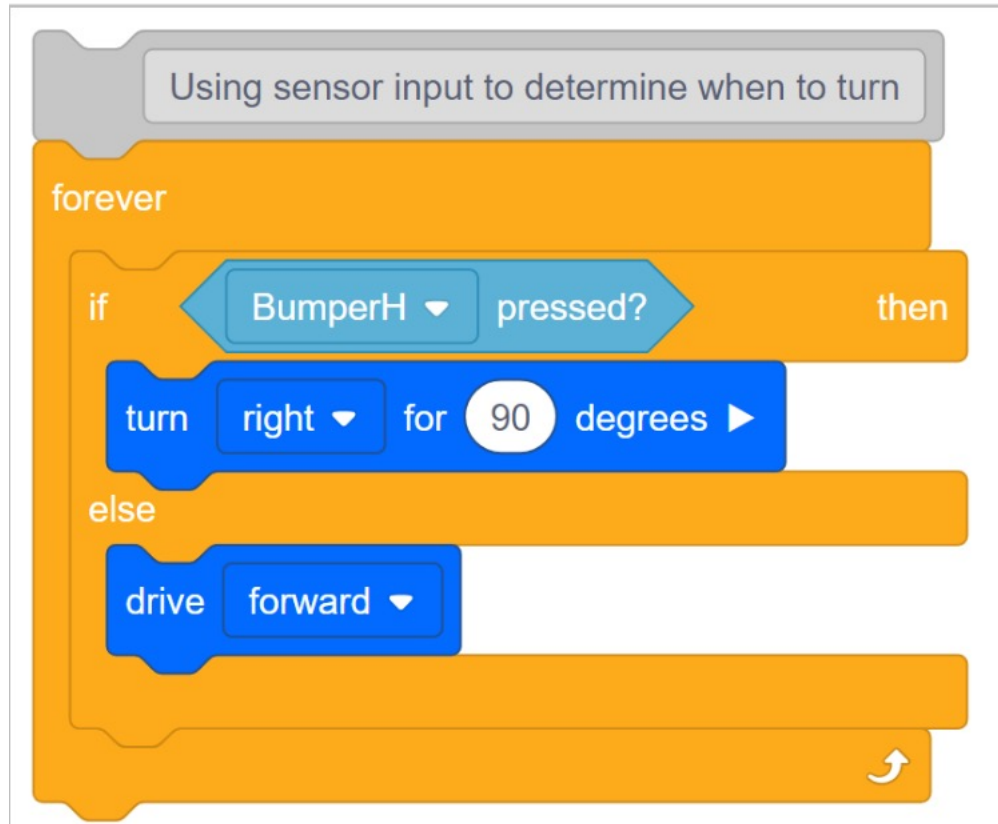


Forever Loop

- A [Forever] block, for example, repeats the blocks inside of it on a forever loop.
- The arrow at the bottom of the block indicates the behaviors inside will be repeated on a loop



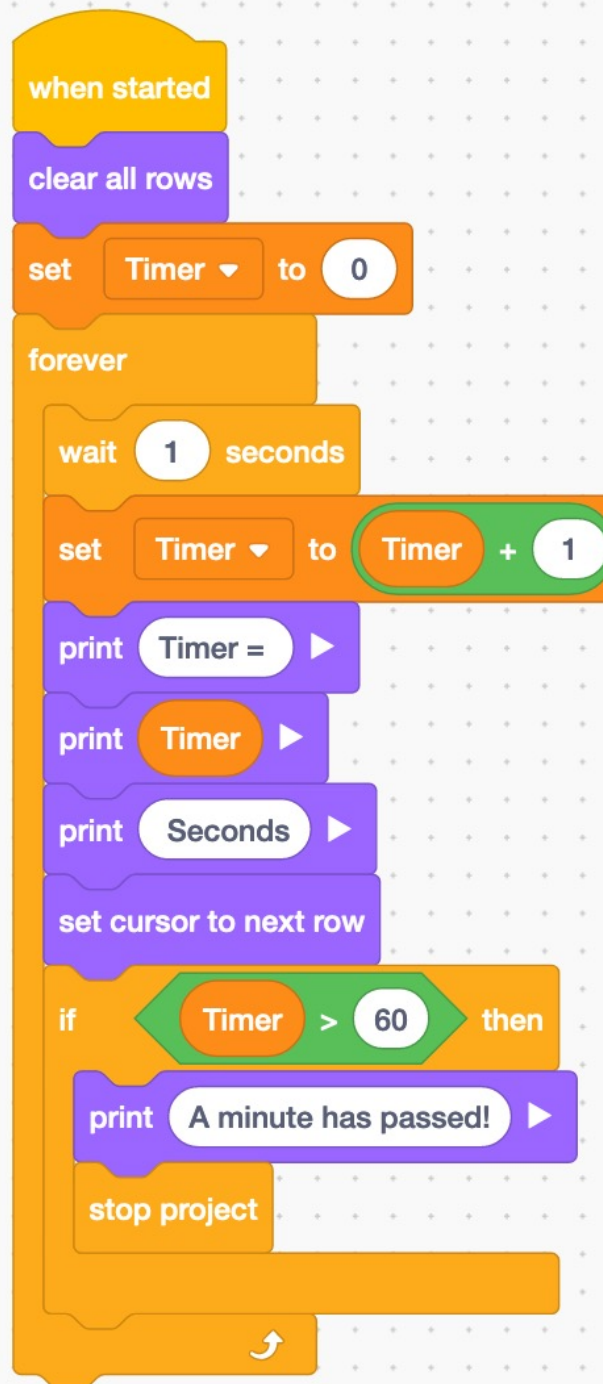
Forever Loops



Forever Loops

- How do use forever loop?
 - Rarely do we want a behavior to repeat forever
 - Add a condition to break or stop the project



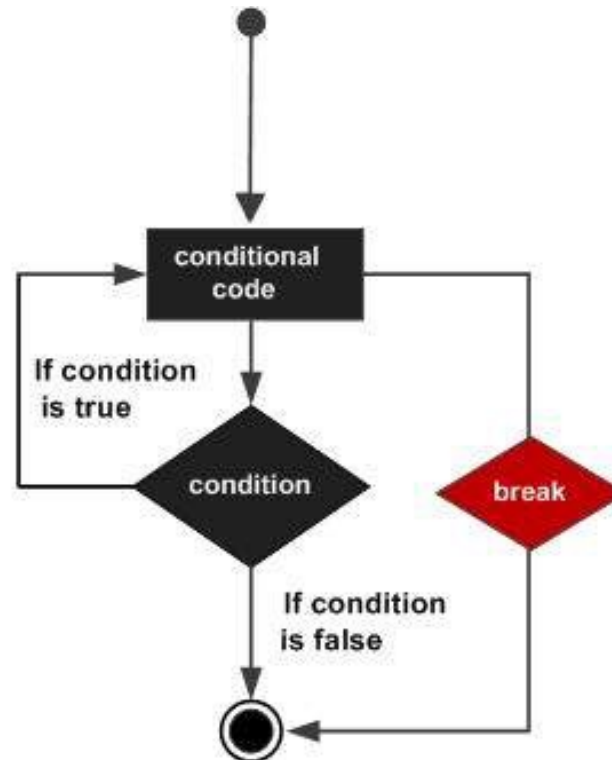


The Break Statement

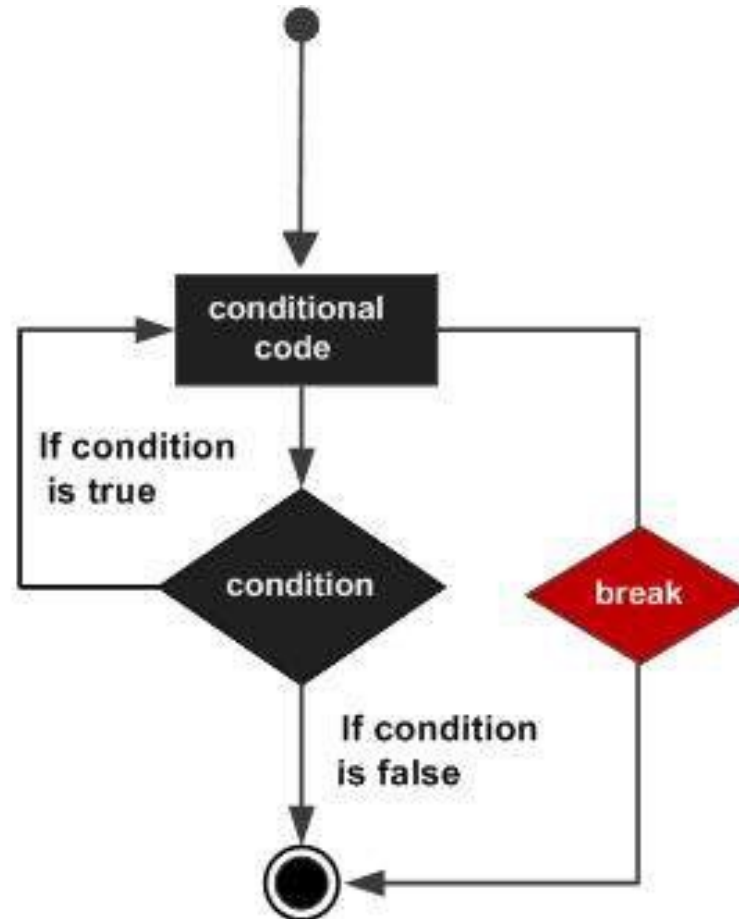
- When the **break** statement is encountered inside a loop, the loop is immediately terminated
- The program control resumes at the next statement following the loop.

The Break Statement

- A **break** statement can be represented in the form of a flow diagram as shown below



The Break Statement





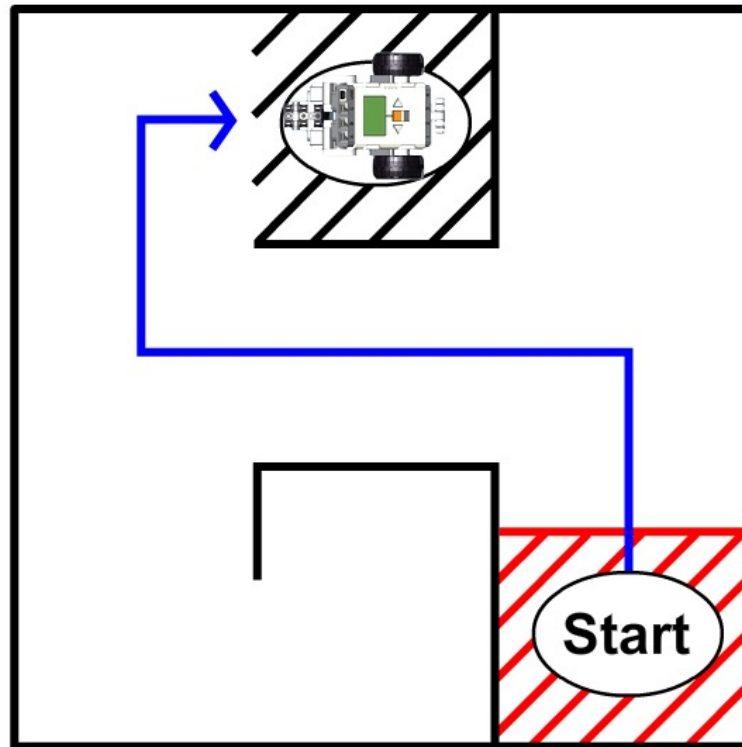
PSEUDOCODE

Planning and Behavior

- What is the problem?
 - Identify the behavior you need

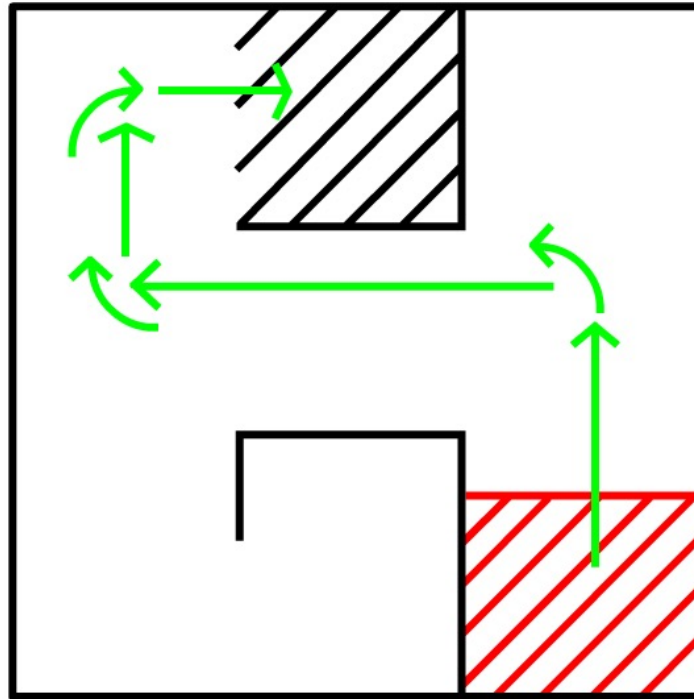
Planning and behavior

- Example: follow the path



Planning and behavior

- Break the main path into smaller paths:

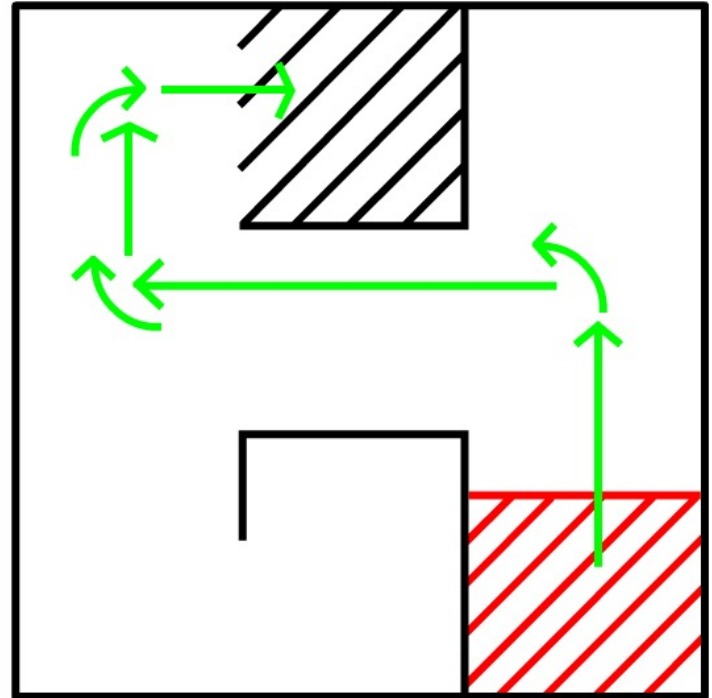


Planning and behavior

- Each of the smaller paths is called a behavior
- Write down the sequence of behaviors that is needed

Planning and behavior

- Follow the path:
 - Move forward
 - Turn left
 - Move forward
 - Turn right
 - Move forward
 - Turn right
 - Move forward

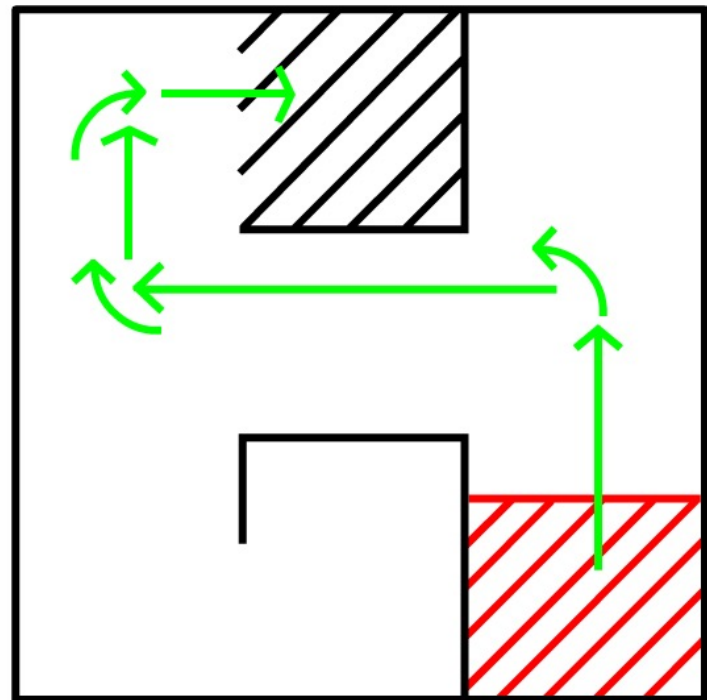


Planning and behavior

- Can we break these into smaller tasks?

Planning and behavior

- Follow the path:
 - Move forward
 - Left motor forward
 - Right motor forward
 - Wait 2 seconds
 - Turn left
 - Left motor reverse
 - Right motor forward
 - Wait 1 second
- Etc...



Pseudocode

- As we increase the level of details, we will reach commands we can express directly
 - in programming language
- This is the plan the robot needs to follow
- The steps are written in English
 - So can be understood by the human programmer
- This is called *Pseudocode*

Pseudocode Example

- Goal: prepare PB&J sandwich

Pseudocode Example

- Take exactly two pieces of bread.
- Take one piece of bread that is not covered with peanut butter on any side
- use a knife to spread peanut butter on one side
- Take a second piece of bread that is not covered with jelly on any side
- use a knife to spread jelly on one side

Pseudocode Example

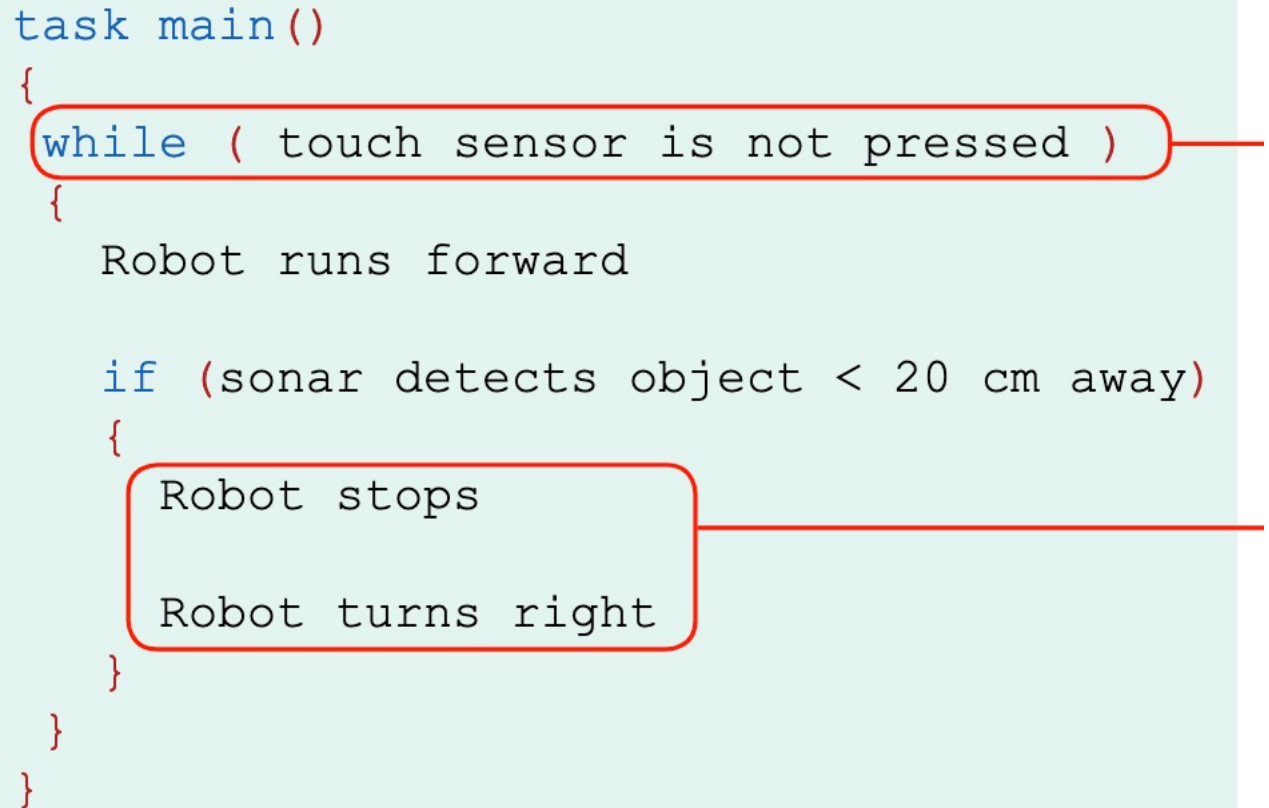
- Place the jelly side of the second piece of bread against the peanut butter side of the first piece of bread.
- Place the combined pieces of bread on plate

Pseudocode Example

- Robot runs forward. If an obstacle is detected within 20 cm distance, robot stops and turns right, and then continues moving forward. Robots deactivates when the touch sensor is pressed.

Pseudocode Example

```
task main()  
{  
  while ( touch sensor is not pressed )  
  {  
    Robot runs forward  
  
    if (sonar detects object < 20 cm away)  
    {  
      Robot stops  
      Robot turns right  
    }  
  }  
}
```



Pseudocode Example

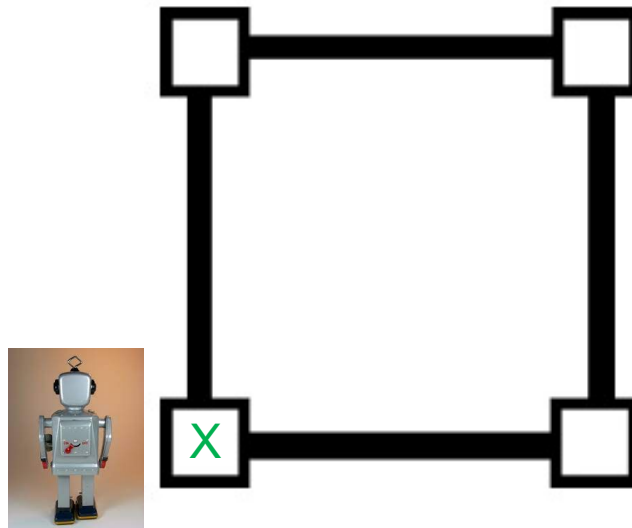
- Can we improve this/suggest alternative solution?
 - Sense before moving?
 - Making sure we don't advance into a wall before sensor result is received.

Pseudocode Example

- task main()
- {
 - While (touch sensor is not pressed)
 - {
 - If (sonar detects object < 20 cm away)
 - {
 - Robot Stops;
 - Robot turns right;
 - }
 - Robot moves forward
 - }
 - }
- }

Pseudocode Exercise

- Robot needs to go once around a square box with length 10 cm. It starts at the beginning line and faces north. It will end on the line facing north





Pseudocode Exercise

- Step 1: Go forward 10 cm
Step 2: Turn left 90 degrees
Step 3: Repeat steps 1 and 2 three more times
- Alternatively, you can add the repeat at the beginning:
 - Repeat 4 times
 - Go forward 10 inches
 - Turn left 90 degrees
 - End of program

