

CS1102

Lecture 3

Boolean Logic, Conditionals and Loops

**Not to be redistributed
to Course Hero or any
other public websites**

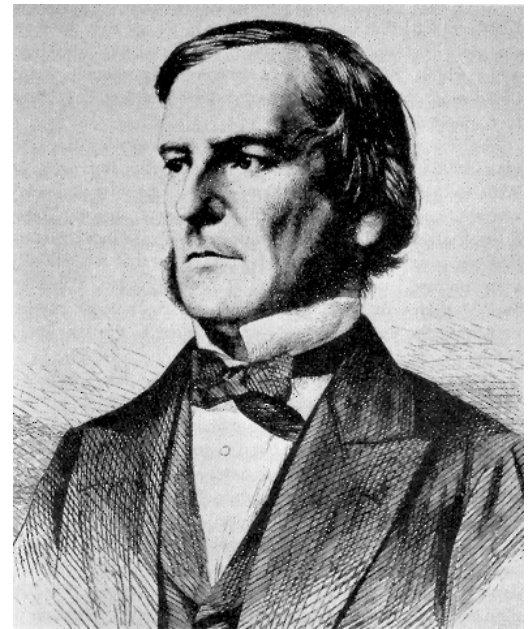


Semester A, 2020-2021
Department of Computer Science
City University of Hong Kong



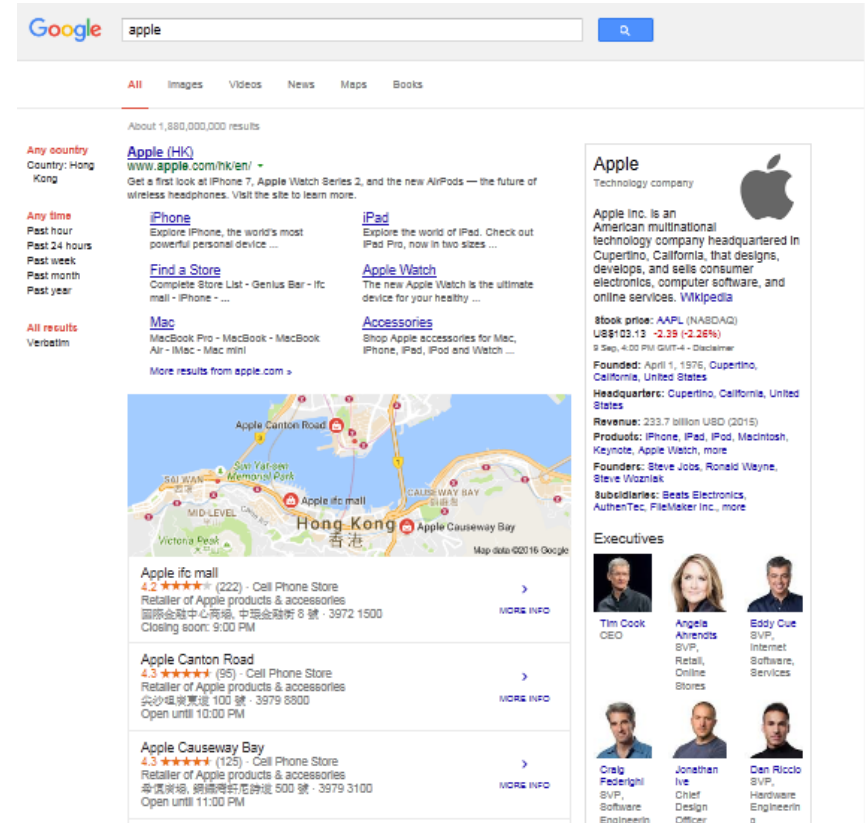
Boolean Logic

- Boolean Logic
 - A form of algebra with operations to work with values that are either True or False, often denoted by 1 or 0 respectively
 - Introduced by George Boole in 1800s
 - You probably have already been using some common Boolean operators: AND, OR, NOT



Boolean Operator Example in Daily Life

- Let's say your Biology teacher asks you to research about different kinds of apples
- If you go online and use the keyword *apple* in the search engine, you would get results about electronic devices or newspaper which also has the word Apple in their company name. So the top results do not contain information that you are looking for



Boolean Operator Example in Daily Life (2)

- So you may refine your search result say using the Google Advanced Search



Google Search

I'm Feeling Lucky

Google offered in: [中文\(繁體\)](#) [中文\(简体\)](#) [मराठी](#) [日本語](#)

Search settings

Advanced search

History

Search Help

Send feedback

Privacy

Terms

Settings

Advanced Search

Find pages with...

all these words:

this exact word or phrase:

any of these words:

none of these words:

numbers ranging from:

to

Boolean Operator Example: AND

- You can enter the keywords *apple* and *fruit* in the field “all these words” and the search engine will look for webpages that contain both *apple* AND *fruit*

Advanced Search

Find pages with...

all these words:

this exact word or phrase:

any of these words:

none of these words:


numbers ranging from:

to

[All](#) [Images](#) [Videos](#) [News](#) [Maps](#) [Books](#)

About 127,000,000 results

[Images for apple fruit](#)



[Apple - Wikipedia, the free encyclopedia](#)
<https://en.wikipedia.org/wiki/Apple>

The apple tree is a deciduous tree in the rose family best known for its sweet, pomaceous fruit, the apple. It is cultivated worldwide as a fruit tree, and is the most ...

[Apple Inc. - Malus sieversii - Apple \(disambiguation\) - Malus](#)

[Apple fruit nutrition facts and health benefits - Nutrition and You](#)
www.nutrition-and-you.com/apple-fruit.html

Delicious and crunchy apple fruit is one of the popular fruits containing an impressive list of antioxidants and essential nutrients required for good health...

[Apple fruit | History of Apples | Nutrition Facts - Fruits](#)
www.fruitsinfo.com/apples.php

Apple fruit was the king of all fruits originated in the past 4000 years ago. It has the best nutritional facts and we have a few tips about the selection & storage ...

[15 Health Benefits of Apples | Best Health Magazine Canada](#)
www.besthealthmag.ca/best.../15-health-benefits-of-eating-apples/

Many of us forget that sometimes, the simplest answers are the best. Better health could be as easy as reaching for the fruit bowl for some apples next time you ...

Boolean Operator Example: OR

- You may want to look for specific types of apples such as Fuji apple or Washington apple to study their characteristics. You can enter the keywords *Fuji* and *Washington* in the field “any of these words” and the search engine will look for webpages that contain *Fuji* OR *Washington* (or both)

Advanced Search

Find pages with...

all these words:

apple fruit

this exact word or phrase:

any of these words:

Fuji Washington

none of these words:

numbers ranging from:

to

apple fruit Fuji OR Washington

All Images Videos News Maps Books

About 15,500,000 results

[Fuji \(apple\) - Wikipedia, the free encyclopedia](#)

[https://en.wikipedia.org/wiki/Fuji_\(apple\)](https://en.wikipedia.org/wiki/Fuji_(apple))

The Fuji apple is an apple hybrid developed by growers at the Tohoku Research Station in
Grapple (fruit)—a Concord grape-flavor-infused Fuji apple ...

[Gala - Ralls Genet - Grapple](#)

[Apple - Wikipedia, the free encyclopedia](#)

<https://en.wikipedia.org/wiki/Apple>

The apple tree is a deciduous tree in the rose family best known for its sweet, pomaceous fruit, In the 20th century, irrigation projects in Eastern Washington began and allowed the development of the multibillion-dollar fruit industry, of which ...

[Images for apple fruit Fuji OR Washington](#)



[Washington Fruit Apples | Washington Fruit & Produce Co](#)

www.washfruit.com/produce/apples.html

Apples are unique. They differ from other foods because they're ...

[The Yummy Fruit Company | Apples - Ambrosia, Braeburn, Fuji ...](#)

www.yummyfruit.co.nz/apples

YUMMY Royal Gala is our most popular apple. It is a high-colour version of the original gala strain. Very popular in the 1960's and even more popular now, it is ...

Boolean Operator Example: NOT

- From the previous search results, you find that the top 2 results are from Wikipedia. If you want to exclude all results from Wikipedia so that you can check from other sources, you can enter the keyword Wikipedia under “none of these words”

Advanced Search

Find pages with...

all these words:

this exact word or phrase:

any of these words:

none of these words:

numbers ranging from: to

apple fruit Fuji OR Washington -Wikipedia

All Images Videos News Maps Books

About 13,600,000 results

[Images for apple fruit Fuji OR Washington -Wikipedia](#)



[Washington Fruit Apples | Washington Fruit & Produce Co](#)
www.washfruit.com/produce/apples.html

Apples are unique. They differ from other foods because they're ...

[Apple - Fuji - tasting notes, identification, reviews - Orange Pippin](#)
www.orangepippin.com/apples/fuji

A very attractive modern apple, crisp, sweet-flavoured, and keeps well. ... Slow to start bearing early crops tend to have poor flavor; Fruit bearing: Spur-bearer ...

[Apple Suppliers - Washington Apple Commission](#)
bestapples.com/wa-apple-suppliers/apple-suppliers/

Double Diamond Fruit (Morgans of Washington) 1801 F Street SW Quincy, WA 98848. 509-787-4644. Sales: Columbia Marketing International (CMI). Douglas ...

[Borton Fruit: Premium Quality Apples, Cherries and Pears Since 1912](#)
<https://www.bortonfruit.com/>

Borton and Sons is a premium grower, packer and shipper of the highest quality apples, pears and cherries in Washington State.

[Washington State Fruit | Apple - State Symbols USA](#)

Logic Gate

- A tiny device that computes a Boolean operation
- Often implemented as (small) electronic circuits
- Provide the building blocks from which computers are constructed
- Specific logic gates:
 - AND, OR, NOT, XOR (exclusive or), NAND, NOR, XNOR (exclusive nor)

AND Gate

AND gate



The output is True when both inputs are True; otherwise, the output is False.

Truth Table

Input		Output
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

False AND False is False

False AND True is False

True AND False is False

True AND True is True

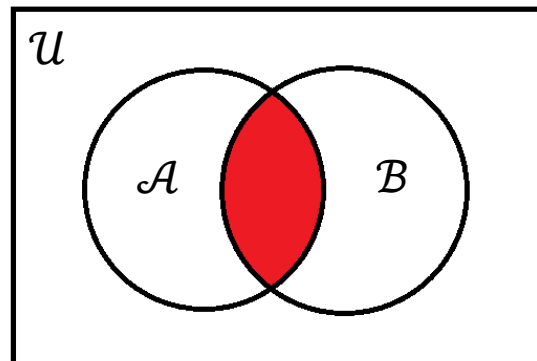
A truth table lists all combinations of the input and provides the output values under each input combination

0 = False

1 = True

- If *any* input is false, then the output of an AND gate is false
- If **ALL** input is true, then the output of an AND gate is true

This is known as a Venn Diagram and can be used to visualize the relationships between sets of data



\mathcal{U} : all webpages

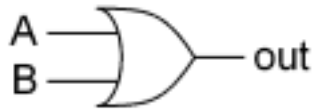
\mathcal{A} : webpages containing apple

\mathcal{B} : webpages containing fruit

Color region - \mathcal{A} AND \mathcal{B} :
webpages containing both
apple and fruit

OR Gate

OR gate



The output is False if both inputs are False; otherwise, the output is True.

Truth Table

Input		Output
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

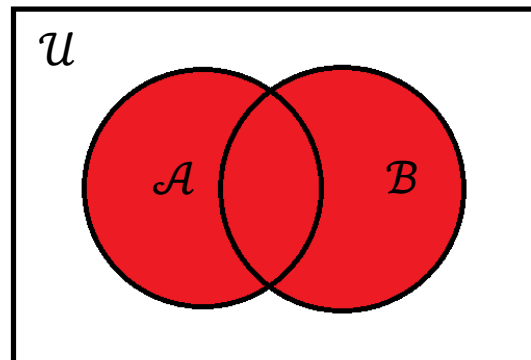
False OR False is False

False OR True is True

True OR False is True

True OR True is True

- If *any* input is true, then the output of an OR gate is true
- If ALL input is false, then the output of an OR gate is false



\mathcal{U} : all webpages

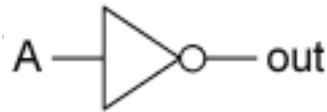
\mathcal{A} : webpages containing Fuji

\mathcal{B} : webpages containing
Washington

Color region - \mathcal{A} OR \mathcal{B} :
webpages containing Fuji or
Washington (or both)

NOT Gate

NOT gate or inverter



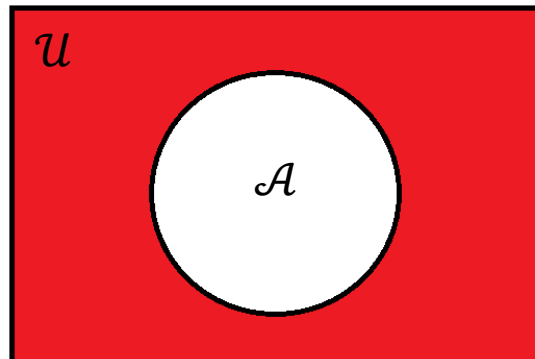
Truth Table

Input	Output
A	NOT A
1	0
0	1

NOT True is False

NOT False is True

- The bit is flipped from 0 to 1 and from 1 to 0, i.e., taking compliment



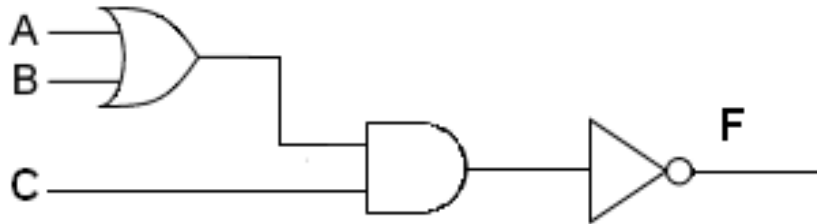
\mathcal{U} : all webpages

\mathcal{A} : webpages containing
Wikipedia

Color region – NOT \mathcal{A} :
webpages that do not contain
Wikipedia

Combining Gates to Form Circuit

- Different circuits can be formed by combining gates



- The relationship between input and output of a circuit can be specified by either a truth table or a Boolean expression
- The following notations are often used in representing Boolean expressions:

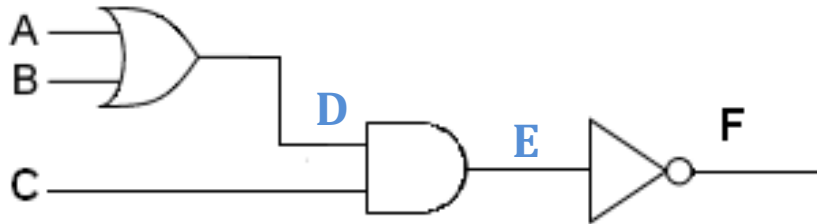
AB : A AND B

$A+B$: A OR B

\bar{A} : NOT A

Circuit Example

- Express the output F in terms of the input A, B, C



- To make it easier to analyze the circuit, intermediate signals can be labeled, e.g., D and E
- The above circuit can be characterized by the following equations:

$$D = A + B$$

$$E = CD$$

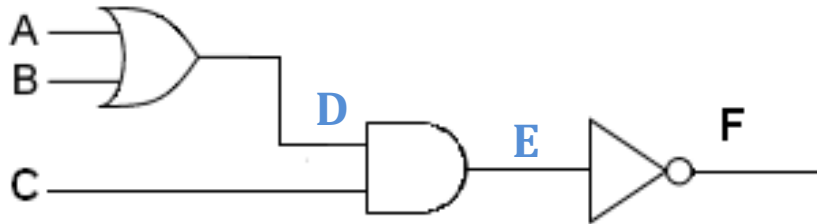
$$F = \overline{E}$$

Combining the above 3 equations, the following can be obtained:

$$F = \overline{C(A+B)}$$

Circuit Example (cont.)

- Write down the truth table of the following circuit



With 3 input (A, B, C),
there are 8 entries in
the truth table.

In general, with N input,
there are 2^N entries in
the truth table

Input			Intermediate		Output
A	B	C	$D = A+B$	$E = CD$	$F = \bar{E}$
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	0

Circuit Design

- Is it possible to design a circuit with logic gates such that given a truth table that specifies the input-output relationship?

Input			Output
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

One way to solve this problem is to examine the cases when the output is 1 to identify the pattern.

In this example, the highlighted (yellow and blue) cells are the cases when the output is 1.

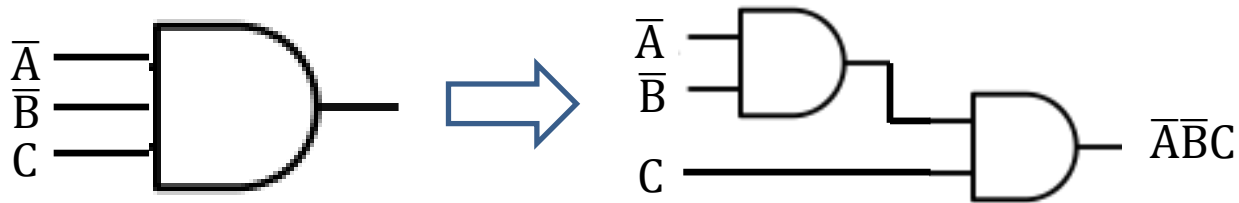
- By inspection, you can find out the pattern that the yellow highlighted cells correspond to the case when $C=0$, which can be expressed as \bar{C} .
- For the remaining case highlighted in blue, it corresponds to the case $A=0$ AND $B=0$ AND $C=1$, which can be expressed by $\bar{A}\bar{B}C$.

Combining these cases, the output can be expressed by \bar{C} OR $\bar{A}\bar{B}C$, i.e., $F = \bar{C} + \bar{A}\bar{B}C$.

Note that although the above expression is different from the one shown in previous slide, it is also a valid design. This means that different circuit design can achieve the same input-output relationship

AND/OR Gate with Multiple Input

- In the previous slide, the circuit is designed as $F = \bar{C} + \bar{A}\bar{B}C$. The 2nd term requires an AND gate with 3 input. This can be constructed from 2 2-input AND gates, i.e., $(\bar{A}\bar{B})C$



- In general, an N-input AND gate can be obtained by concatenating (N-1) 2-input AND gates
- Similarly, an N-input OR gate can be obtained by concatenating (N-1) 2-input OR gates

Circuit Design Revisited

- Let's say that you are not able to derive a pattern as we did, is there any way to derive the expression for the output?

Input			Output
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Now all highlighted cells that correspond to the cases for $F=1$. We see that these 5 highlighted cells correspond to the following cases

1. $\overline{A}\overline{B}\overline{C}$: $A=0$ AND $B=0$ AND $C=0$
2. $\overline{A}\overline{B}C$: $A=0$ AND $B=0$ AND $C=1$
3. $\overline{A}B\overline{C}$: $A=0$ AND $B=1$ AND $C=0$
4. $A\overline{B}\overline{C}$: $A=1$ AND $B=0$ AND $C=0$
5. $AB\overline{C}$: $A=1$ AND $B=1$ AND $C=0$

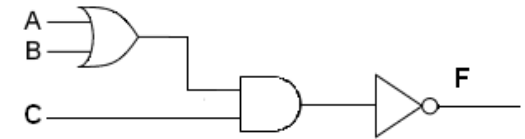
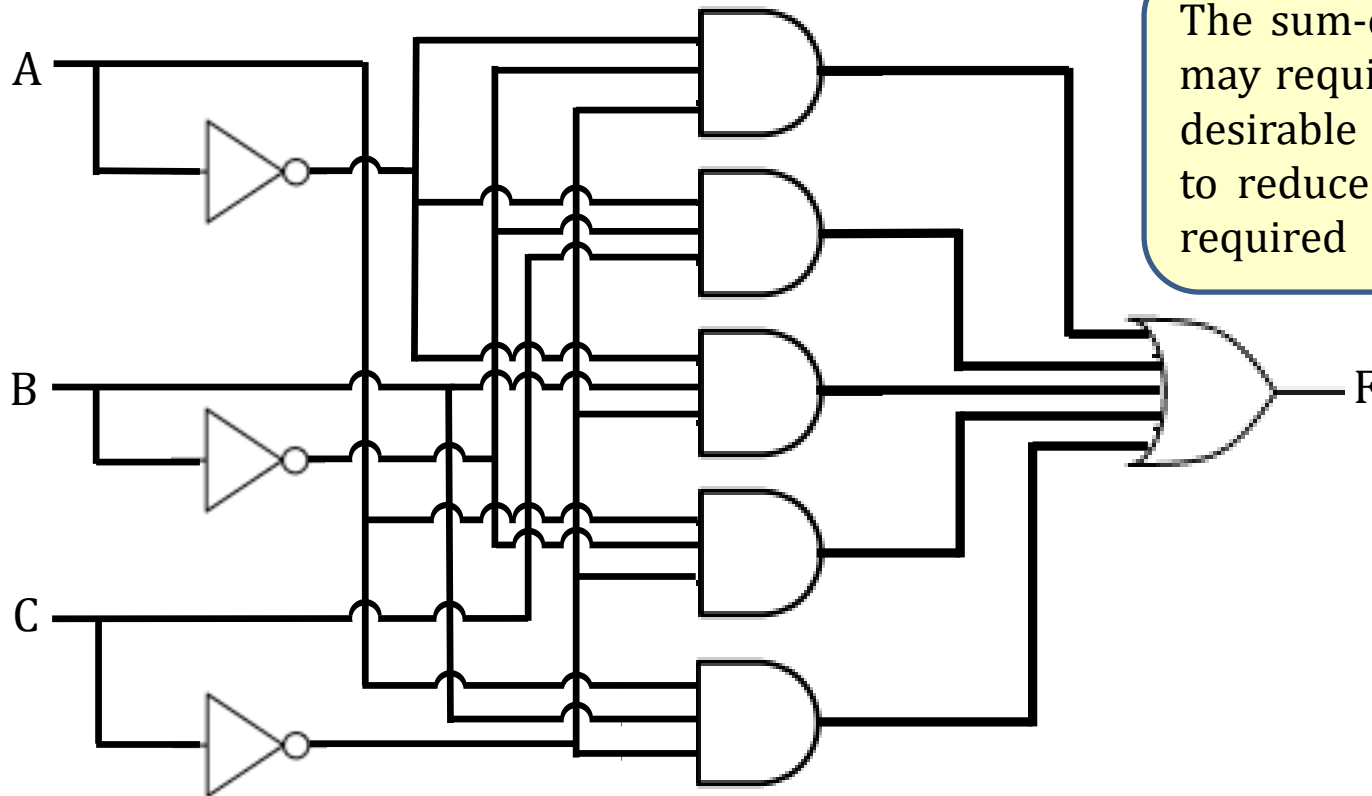
Combining these cases, the output can be expressed by the above 5 terms joined by the OR operators, i.e.,

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C}$$

This is known as sum-of-product expression in digital circuit.

Sum-of-Product Expression

- $F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$



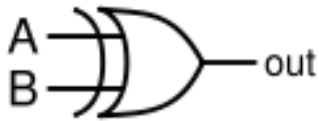
The sum-of-product expression may require many gates so it is desirable to simplify the circuit to reduce the number of gates required

1st level: AND gates

2nd level: OR gate

XOR Gate

XOR gate



The output is True if either, but not both, of the inputs are True; otherwise, the output is False.

Truth Table

Input		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

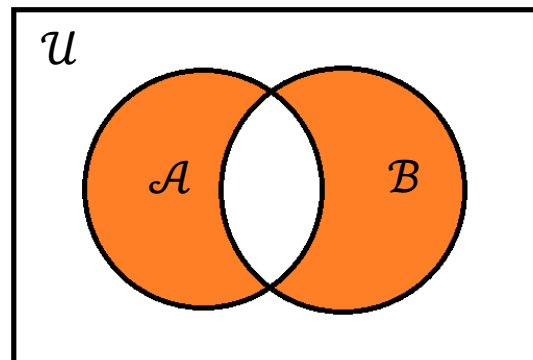
False XOR False is False

False XOR True is True

True XOR False is True

True XOR True is False

- If the inputs are the same, then the output of an XOR gate is false
- If the inputs are different, then the output of an XOR gate is true



\mathcal{U} : lunch special

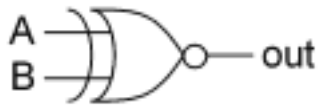
\mathcal{A} : coffee

\mathcal{B} : tea

Color region - $\mathcal{A} \text{ XOR } \mathcal{B}$:
lunch special with coffee or tea
(but not both)

XNOR Gate

XNOR gate



The output is True if both of the inputs are True or both of the inputs are False ;
otherwise, the output is False.

Truth Table

Input		Output
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

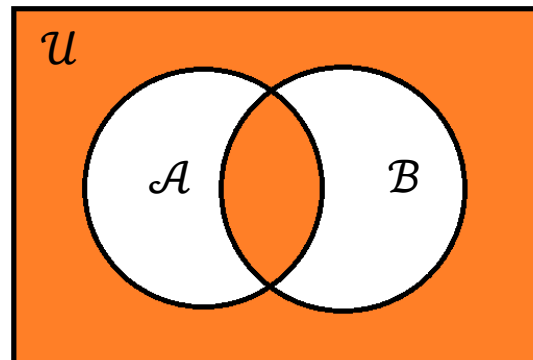
False XNOR False is True

False XNOR True is False

True XNOR False is False

True XNOR True is True

- If the inputs are the same, then the output of an XNOR gate is true
- If the inputs are different, then the output of an XNOR gate is false
- XNOR gate = XOR gate + NOT gate



\mathcal{U} : fast food combo

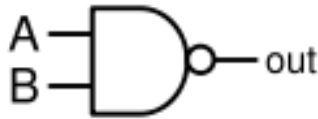
\mathcal{A} : extra large fries

\mathcal{B} : extra large drink

Color region - $\mathcal{A} \text{ XNOR } \mathcal{B}$:
fast food combo with extra
large fries and drink or
regular size fries and drink 20

NAND Gate

NAND gate



The output is True if either input is False; otherwise, the output is False.

Truth Table

Input		Output
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

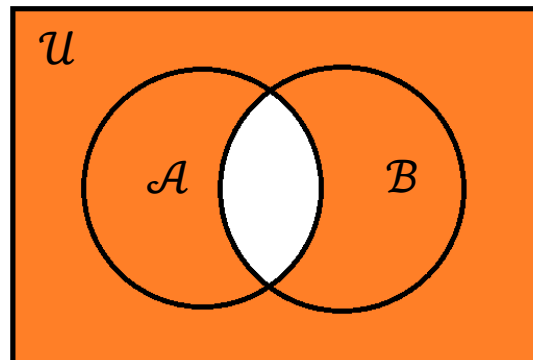
False NAND False is True

False NAND True is True

True NAND False is True

True NAND True is False

- If *any* input is false, then the output of an NAND gate is true
- If ALL inputs are true, then the output of an NAND gate is false
- NAND gate = AND gate + NOT gate



\mathcal{U} : steak

\mathcal{A} : gravy sauce

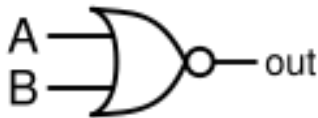
\mathcal{B} : garlic sauce

Color region - $\mathcal{A} \text{ NAND } \mathcal{B}$:

You can have gravy sauce or garlic sauce or no sauce with your steak, but cannot have both sauce

NOR Gate

NOR gate



The output is True if both inputs are False; otherwise, the output is False.

Truth Table

Input		Output
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

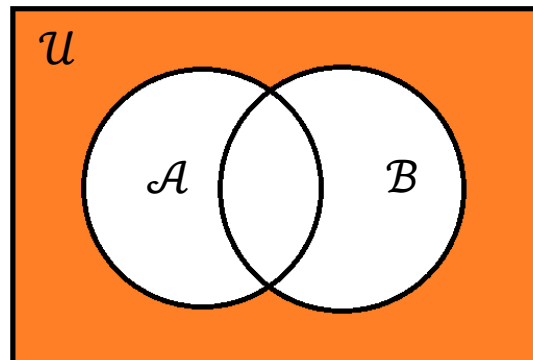
False NOR False is True

False NOR True is False

True NOR False is False

True NOR True is False

- If *any* input is true, then the output of an NOR gate is false
- If ALL inputs are false, then the output of an NOR gate is true
- NOR gate = OR gate + NOT gate



U : green diet

A : meat

B : leftover

Color region - $A \text{ NOR } B$:
you have a green diet when
you do not eat meat and you
finish your dishes so there
is no leftover

Boolean Expression in Programming (1)

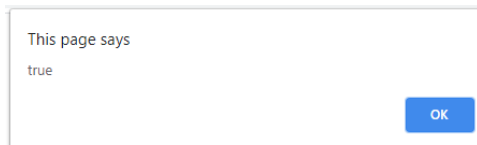
- In programming we often use Boolean expression to determine if a certain condition is true/false
- The comparison operators such as “>” or “<” are commonly used to form a Boolean expression
- A comparison operator compares 2 items (known as operands) and decides whether it is true/false
e.g. $4 > 3$ is true but $1 > 5$ is false

Scratch

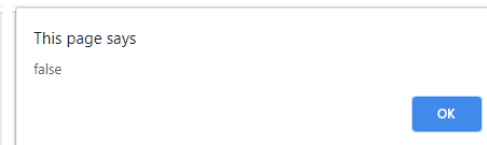


Javascript

```
alert(4>3);
```



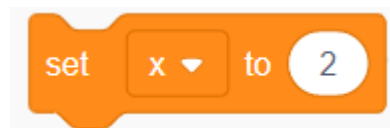
```
alert(5<1);
```



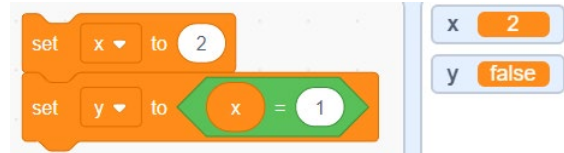
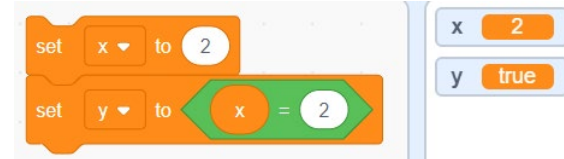
Boolean Expression in Programming (2)

- A comparison operator may also check if 2 operands are equal, e.g., true if $x=2$ and false if $x \neq 2$
 - Note that the comparison operator should not be confused with the variable assignment, i.e., there is a difference between *checking if* $x=2$ and *setting* $x=2$

Scratch



Example:



Javascript

`x == 2`

`x = 2`

```
x = 2;  
y = x == 2;  
alert(x+", "+y);
```

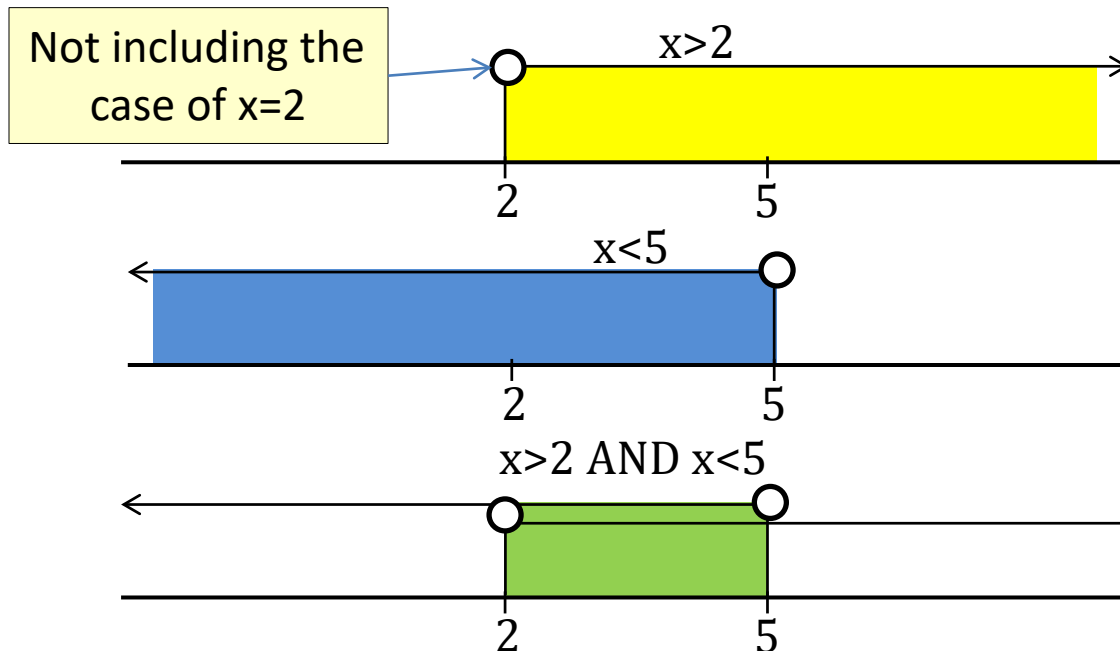
```
x = 2;  
y = x == 1;  
alert(x+", "+y);
```

This page says
2, true

This page says
2, false

Logical Operator in Programming

- The logical operators AND, OR, NOT can be used to form compound Boolean expressions
- E.g., let us represent x graphically by the shaded region corresponding to a given condition



Scratch

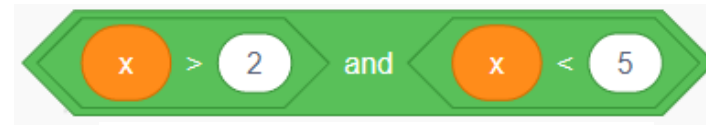


Javascript

$x > 2$



$x < 5$



$(x > 2) \&\& (x < 5)$

More Logical Operator Examples

Not including the case of $x=2$

$x > 2$



$x > 2$

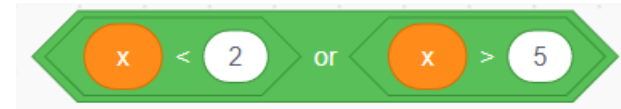
$\text{NOT}(x > 2)$

Including the case of $x=2$



$x \leq 2$

$x < 2 \text{ OR } x > 5$



$(x < 2) \text{ || } (x > 5)$

$\text{NOT}(x > 2) \text{ OR } (x > 5 \text{ AND } x < 7)$



$(x \leq 2) \text{ || } ((x > 5) \&\& (x < 7))$

Conditionals

- In real-life you are making decisions based on certain conditions, e.g.,
 - Before going out, you check the outside weather and will bring an umbrella **if it is raining**
 - **If it is sunny**, you go out to play basketball. **Otherwise**, you stay home to watch TV
- In the above examples, the conditions are highlighted in red and these conditions affect the actions you take
- In programming we can also give instructions to the computer to carry out certain action if a condition is satisfied; or do something else if it is not satisfied
 - This can be achieved by the use of conditional constructs, i.e., if-then or if-then-else

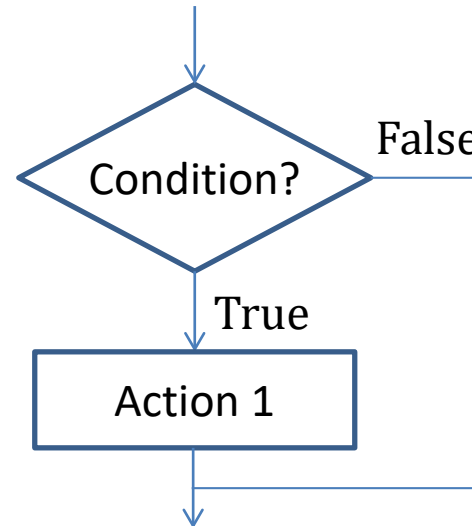
Conditional Constructs

- It is common among programming languages to use if-then or if-then-else as conditional constructs

A flow chart helps to visualize the program flow.

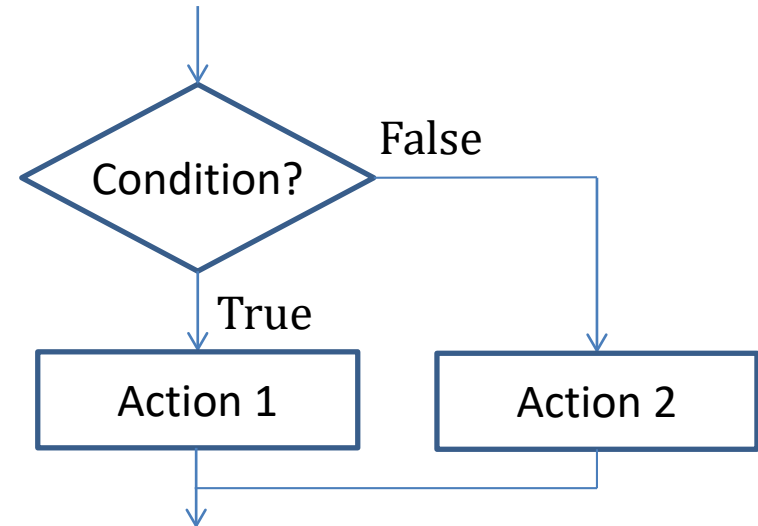
The diamond shape is used to denote a decision block in a flow chart.

There should be 2 arrows labeled True (or Y) and False (or N) coming out of this block, denoting how the program should flow according to the outcome of the condition



if <condition> then
<Action 1> if true

if-then
(one-way conditional)

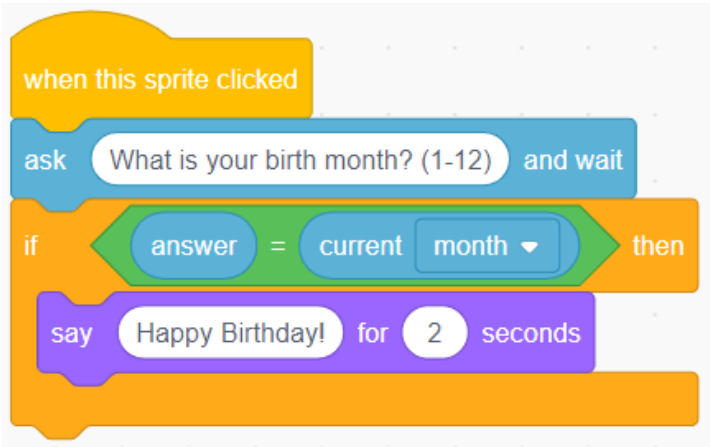


if <condition> then
 <Action 1> if true
else
 <Action 2> if false

if-then-else
(two-way conditional)

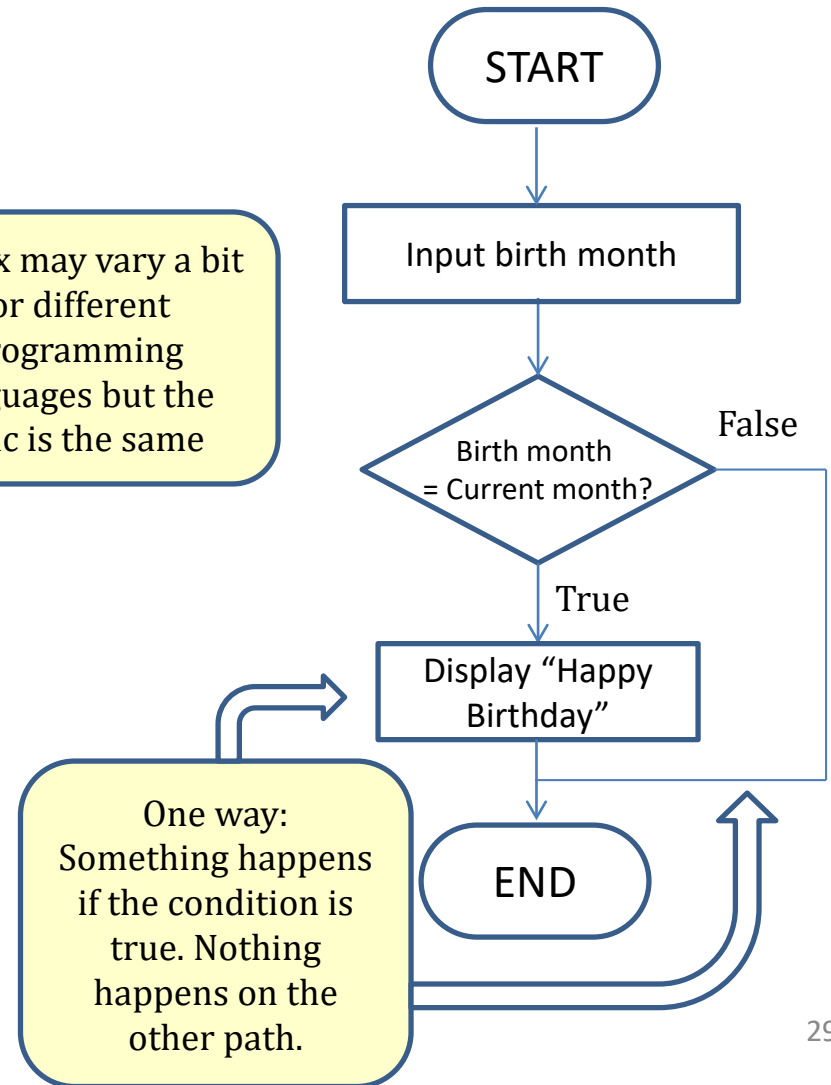
One-way Conditional Example

Scratch (if-then block)



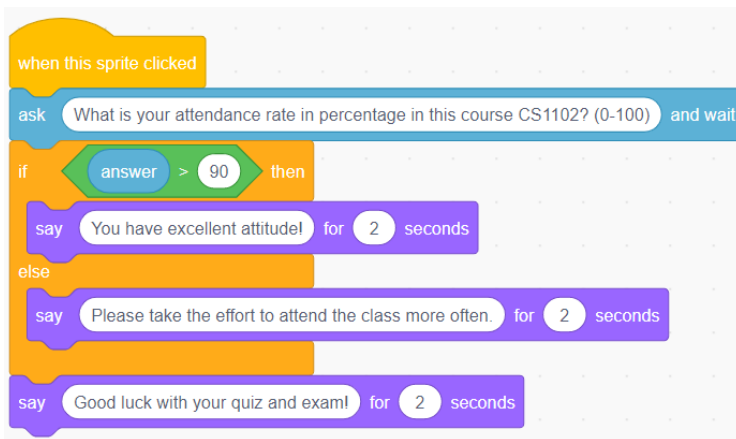
<https://scratch.mit.edu/projects/120774489/>

Syntax may vary a bit for different programming languages but the logic is the same

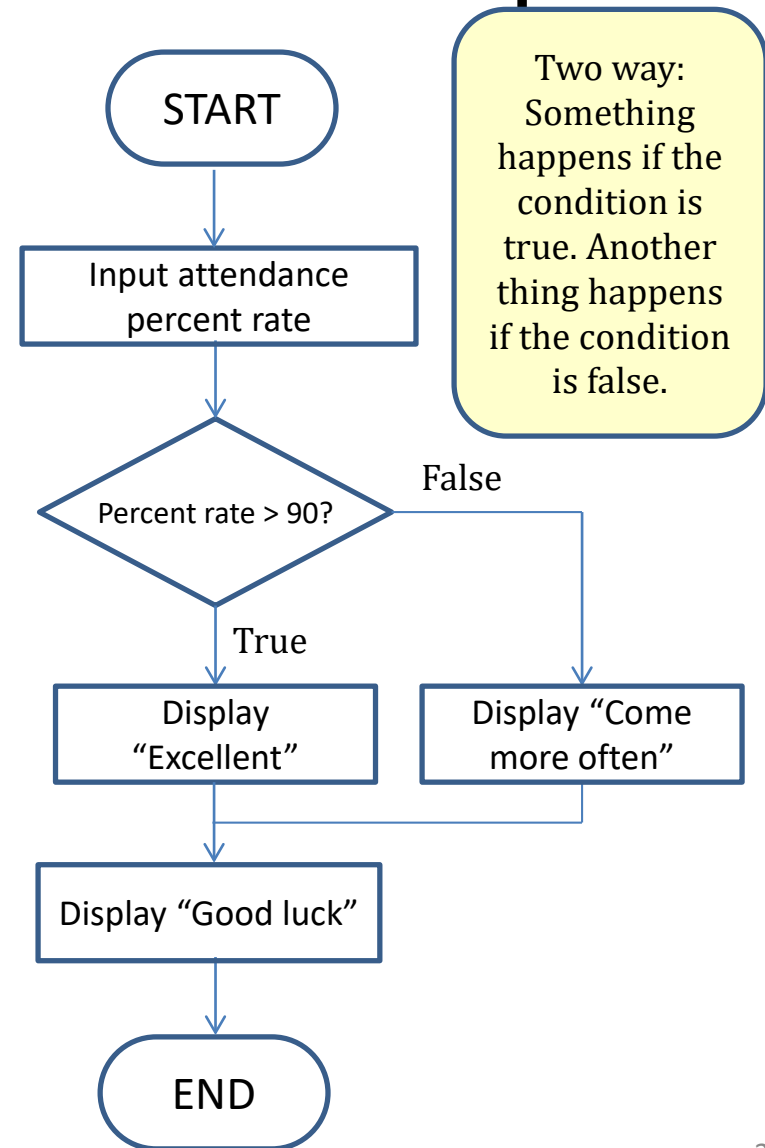


Two-way Conditional Example

Scratch (if-then-else block)

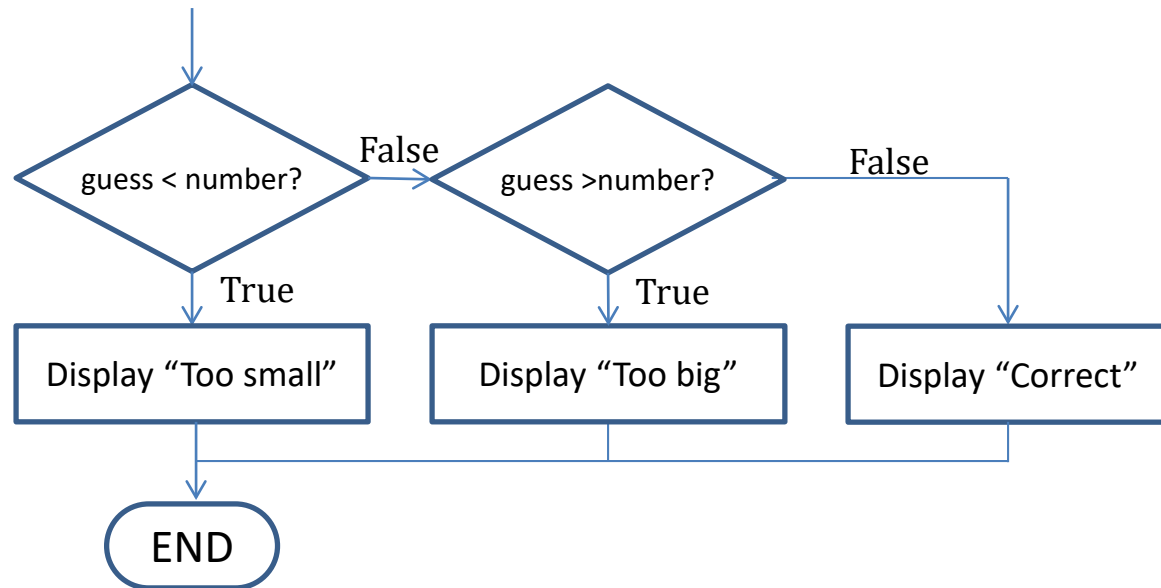
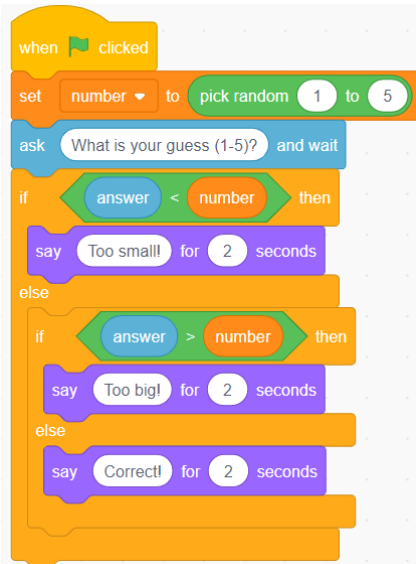


<https://scratch.mit.edu/projects/120774489/>



3-way Conditional

- What if there are more than 2 ways for consideration?
 - Example: you provide your guess for a number. You are notified if your guess is 1) too big; 2) too small; or 3) correct. So there are 3 possible outcomes (3-way).
 - This can be done by putting a two-way conditional inside another two-way conditional



<https://scratch.mit.edu/projects/120783393/>

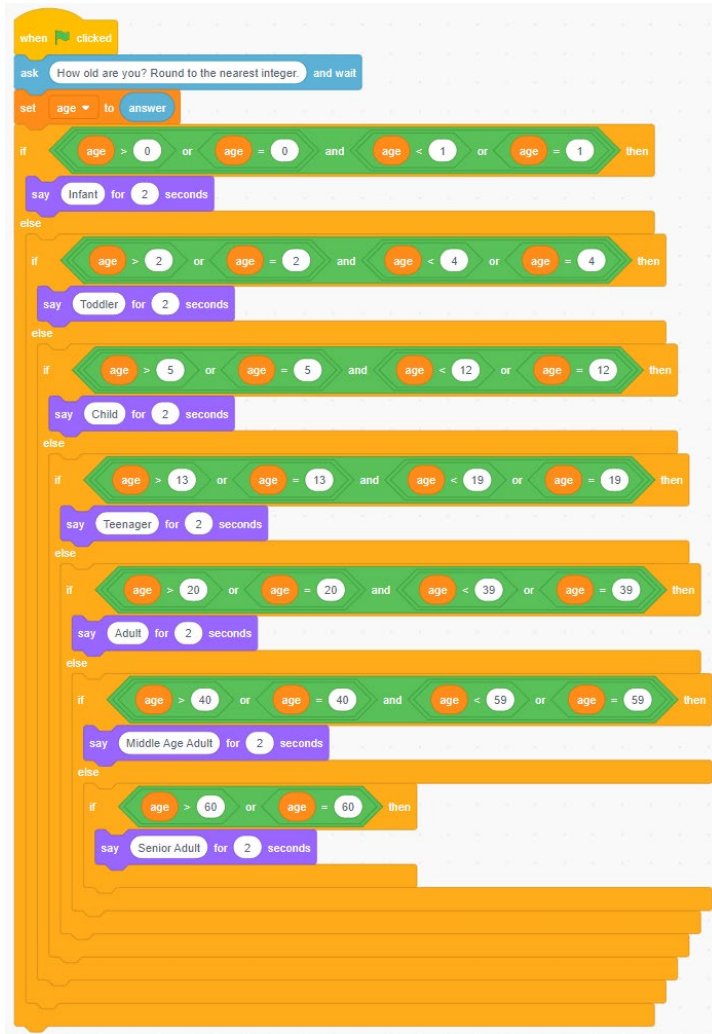
N-way Conditional

- In general, an N-way conditional can be constructed in a similar way by keep placing if-then-else block inside another one in a *nested* manner
 - E.g., how to make the program to output the stage of life given the age of a person round to the nearest integer?

Age	Stage of Life
0-1	Infant
2-4	Toddler
5-12	Child
13-19	Teenager
20-39	Adult
40-59	Middle Age Adult
60+	Senior Adult

<https://integrisok.com/resources/on-your-health/2015/october/stages-of-life-health-for-every-age>

N-way Conditional Example: Nested If



<https://scratch.mit.edu/projects/423273621/>

Age	Boolean Expression	Stage of Life
0-1	age>=0 AND age<=1	Infant
2-4	age>=2 AND age<=4	Toddler
5-12	age>=5 AND age<=12	Child
13-19	age>=13 AND age<=19	Teenager
20-39	age>=20 AND age<=39	Adult
40-59	age>=40 AND age<=59	Middle Age Adult
60+	age>=60	Senior Adult

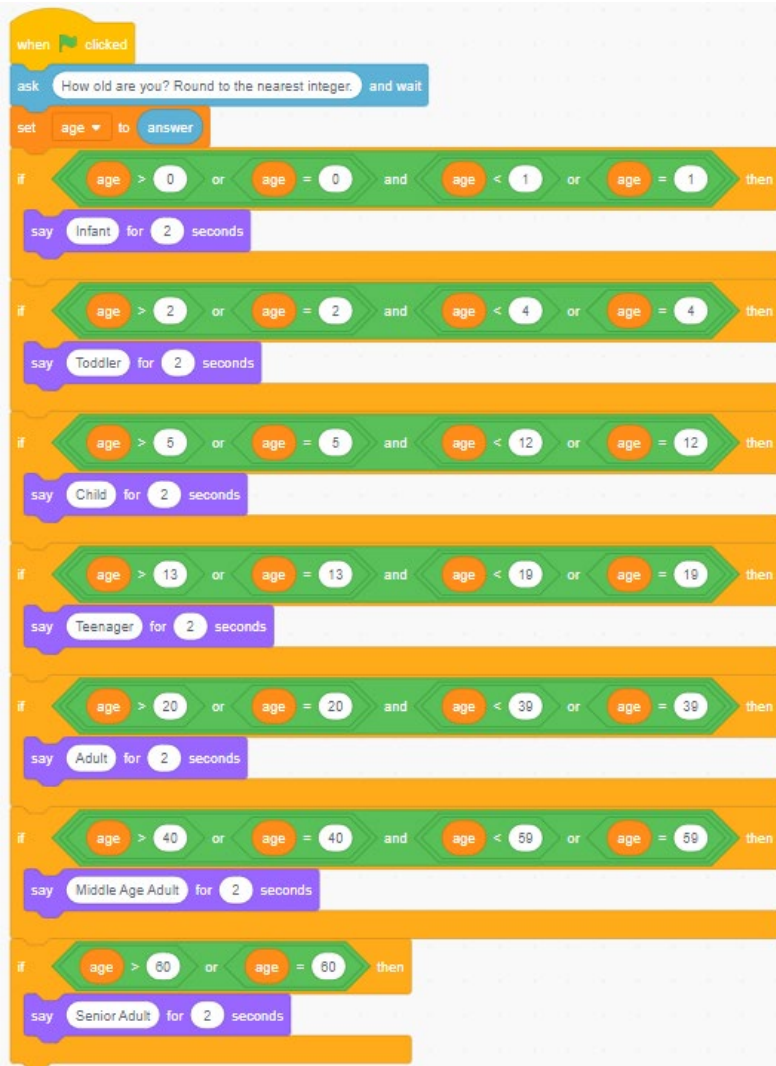
In Scratch, there is no block that directly corresponds to the comparison operator \geq (greater than or equal to). In the example, it is implemented as two Boolean expressions $>$ and $=$ separately, followed by the OR logical operator.



Alternatively, the same function can be achieved by the following block:



N-way Conditional Example: Multiple If?

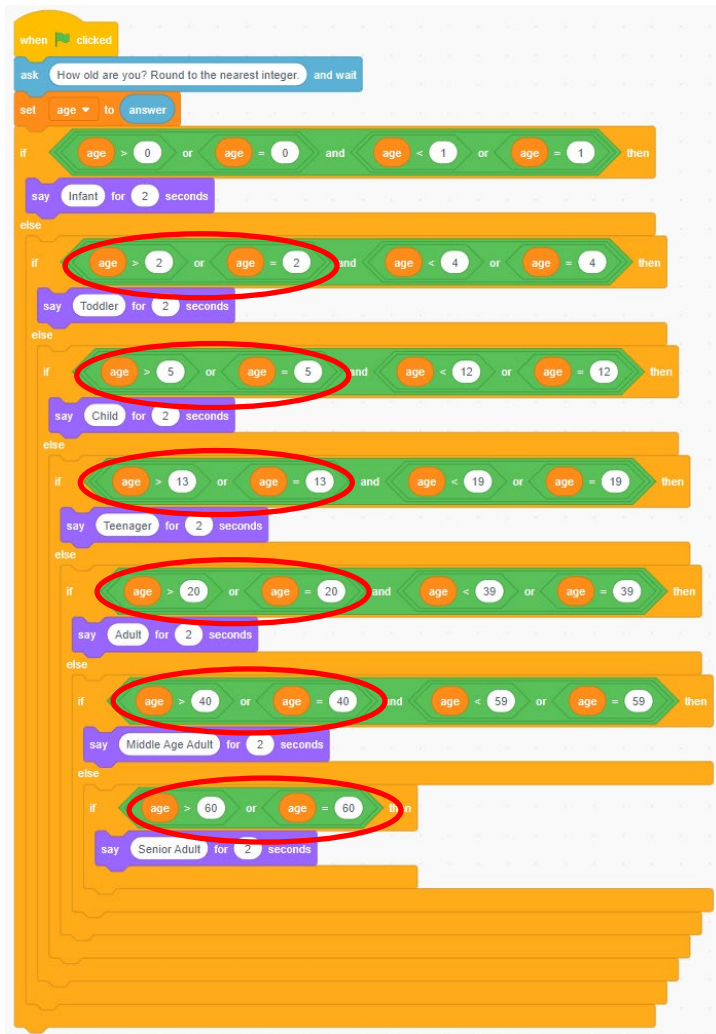


- Novice programmers tend to write program with multiple if-then blocks instead of nested if-then-else blocks
- Do both approaches, multiple if-then blocks and nested if-then-else blocks, work in the same way?

In this example, the program with multiple if-then blocks achieves the same function as the previous program with nested if-then-else blocks, meaning that given the same input, the same output will result from both programs.

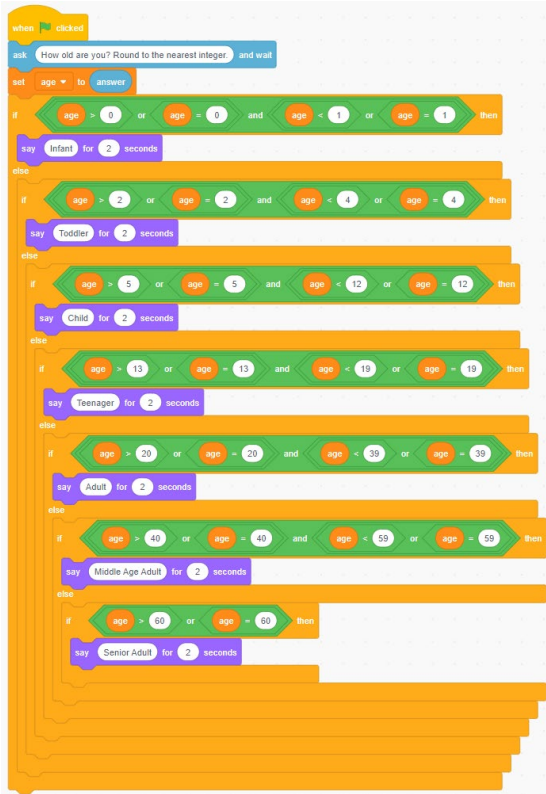
However, this program with multiple if-then blocks is less efficient than the previous program with nested if-then-else blocks. This is because in this program, ALL the conditions are ALWAYS checked. In the previous program, as long as a condition is checked to be true, all other subsequent conditions will not be checked.

N-way Conditional Example: Simplified Conditions in Nested If



The circled conditions are redundant meaning that they can be removed without affecting the correctness of the program. This is because each of them represents the negation of the previous conditions, e.g., $age \geq 2$ is equivalent to $\text{NOT}(age \leq 1)$, assuming that age is an integer. If such a condition is being checked, then it means that the previous condition is not true, implying that the condition being checked must be true so there is no need to explicitly check it.

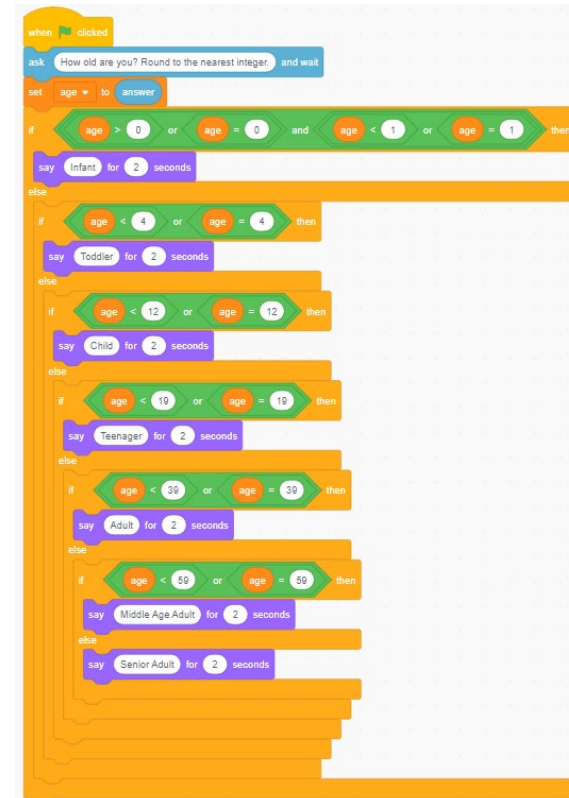
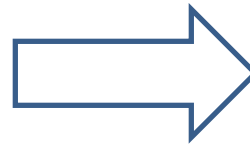
N-way Conditional Example: Simplified Conditions in Nested If (2)



<https://scratch.mit.edu/projects/423273621/>

You can switch the order of the if-then-else block nesting and this program would work in the same way

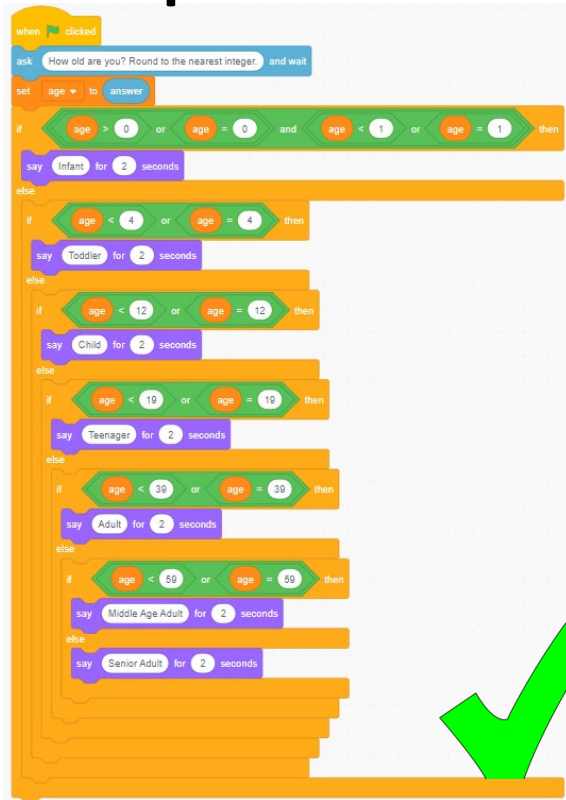
Simplified
Conditions



<https://scratch.mit.edu/projects/423277360/>

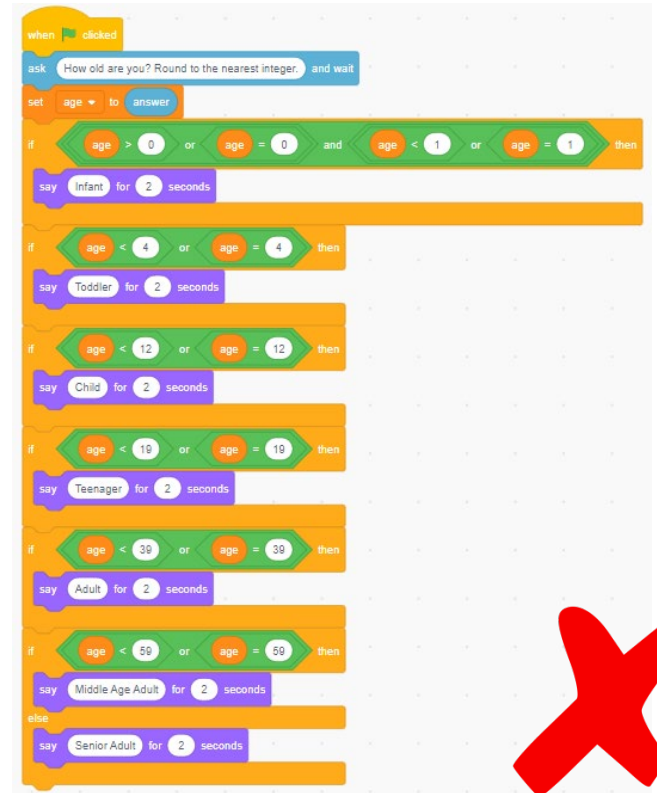
You CANNOT switch the order of the if-then-else block nesting in this program because the simplification is achieved by making assumptions about the conditions that have been checked before

N-way Conditional Example: Simplified Conditions in Nested If (2)



<https://scratch.mit.edu/projects/423277360/>

≠



<https://scratch.mit.edu/projects/423278843/>

If you turn the nested if-then-else blocks with simplified conditions (left) into multiple if-then blocks (right), the resulting program does not work in the same way any more, so it is incorrect.

Condition Ordering

- Let's say we run a simulation to generate a random integer number in the range 1-100 many times and we would like to count how many times the number is larger than 10 as well as how many times the number is less than or equal to 5.
- The following code differs only by the order of the conditions, and performs exactly the same function
- Which program is more efficient? In other words, which program do you expect to run faster given the same number of trials?

Time required for
100,000 trials:
0.20s



Time required for
100,000 trials:
0.26s



<https://scratch.mit.edu/projects/120692359/>

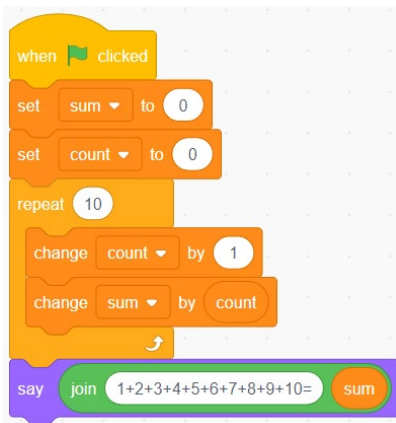
It can be concluded from this experiment that the left program is more efficient because it finishes running faster. This is because the first condition that the left program checks in the nested if-then-else block, $\text{number} > 10$, is much more probable than the second condition, $\text{number} < 6$ (since $1 \leq \text{number} \leq 100$).

Loop

- A loop in a program corresponds to the instructions that perform some tasks repeatedly
- In real-life, you may repeat a task for a given number of times, or you may keep doing something until certain condition is satisfied, e.g.,
 - You attend both the lecture and the lab for this course every week and you repeat this process for a duration of 13 weeks
 - You study for this course repeatedly until you master all the related concepts in order to prepare for the final exam
- Boolean expressions are often used in a loop to check whether some conditions are true/false to determine whether the loop should be stopped/repeated

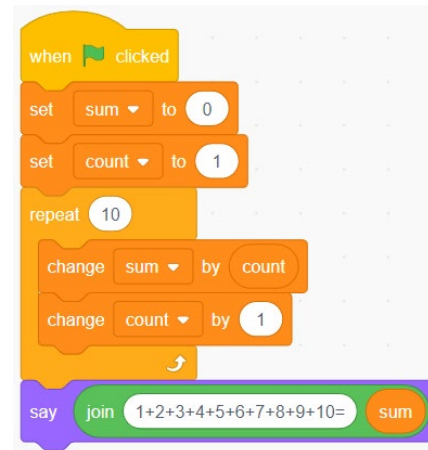
Loop that Runs for a Finite Number of Times

- In Scratch, there is a repeat block that allows you to specify how many times a loop should be run
- In other programming language such as Javascript, often a for-loop is used to define a loop that is to be run for a fixed number of times.
- E.g., compute the sum $1+2+3+4+5+6+7+8+9+10$



<https://scratch.mit.edu/projects/120794114/>

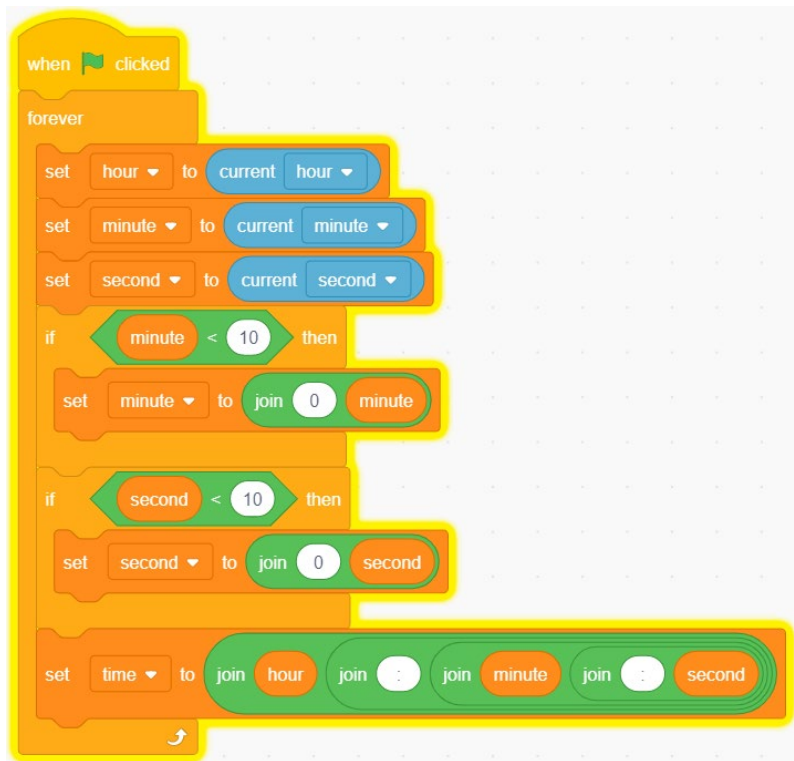
or



Of course you can also type out the entire expression in the code, but it is easy to have typo especially for a long expression and it does not generalize (what if you are asked to add from 1 to 100 instead and you do not know the formula?)

Infinite Loop

- An infinite loop is a loop that will run forever
- An infinite loop can be stopped if the program is stopped



<https://scratch.mit.edu/projects/120794818/>

Loop with Termination Condition

- A programmer may define a loop with **termination** condition such that the loop would continue running unless the termination condition is satisfied, e.g., repeat-until block in Scratch
- Example, a number guessing game in which the player repeatedly guesses the number until it is the correct number



<https://scratch.mit.edu/projects/120612968/>

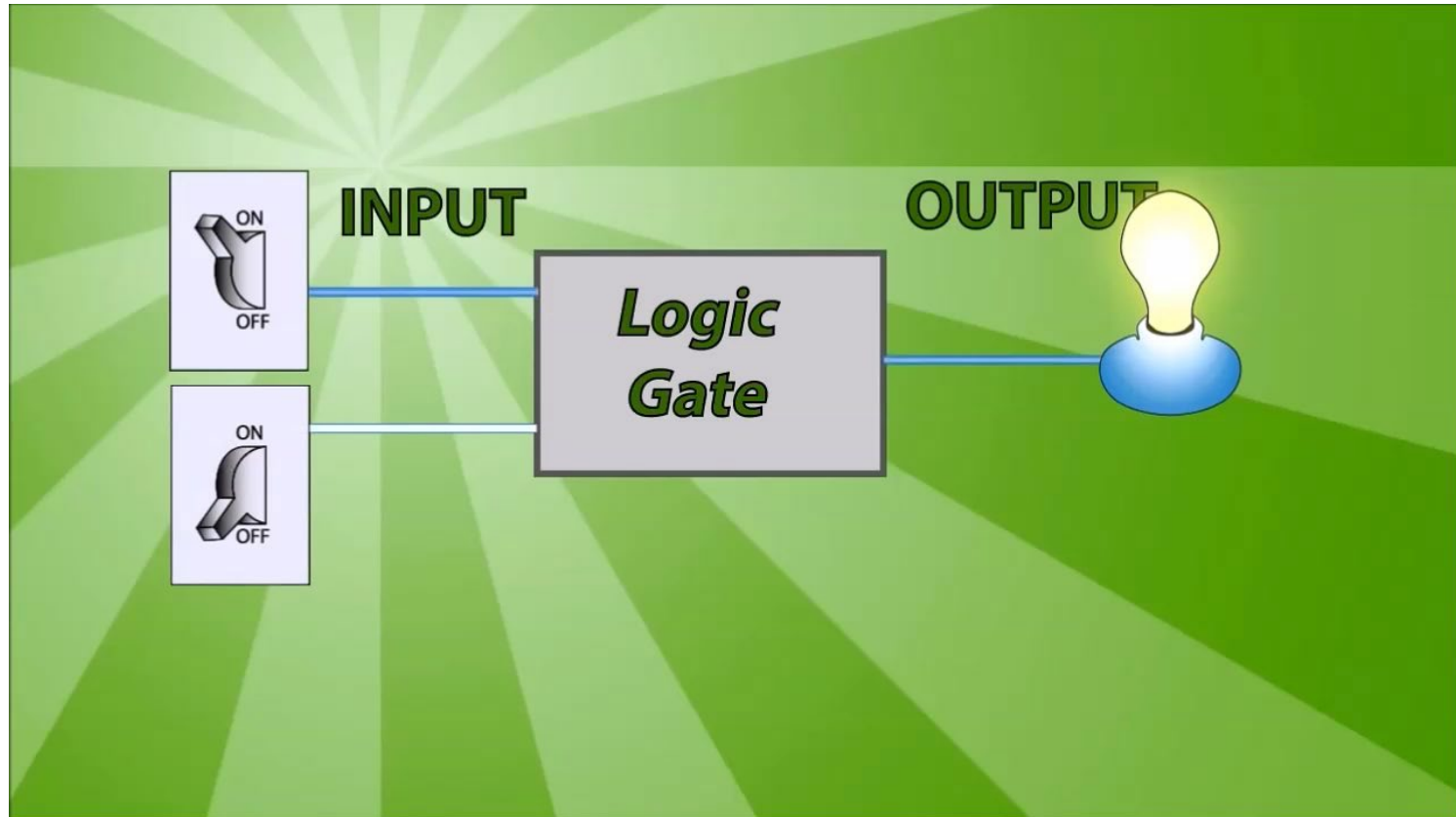
Lesson Summary

- Boolean logic is a form of algebra with operations to work with values that are either True or False
- Compound Boolean expression can be formed by Boolean operators AND, OR, NOT
- Computers are built on a set of strict logic (Boolean logic); complex circuits that perform particular functions are constructed using the basic logic gates
- Conditionals let the program decide which code will be run depending on a condition corresponding to the value of a Boolean expression
- A loop allows the program to carry out a task repeatedly for a fixed number of times, endlessly, or while a continuation condition is satisfied, or until a termination condition is met

Reference

- [1] Howstuffworks.com - How Boolean Logic Works
 - <http://computer.howstuffworks.com/boolean.htm>
- [2] Virginia Tech - Machine Architecture - Gates
 - <http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/Gates/index.html>
- [3] Logic Gate Puzzler
 - <https://www.khanacademy.org/computer-programming/logic-gate-puzzler/1522357785>
- [4] Bee: Loops – Simple game to apply the concept of loops
 - <https://studio.code.org/s/course2/stage/8/puzzle/1>
- [5] Bee: Conditionals – Simple game to apply the concept of conditionals with loops
 - <https://studio.code.org/s/course2/stage/13/puzzle/1>

Video



Logic Gates Basics

<https://www.youtube.com/watch?v=Xi18hl1LqAA>