CSci 127: Introduction to Computer Science

hunter.cuny.edu/csci

Frequently Asked Questions

- What is pseudocode? Why do we use it?
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.

- What was that % symbol?
  It's the symbol for remainder (or modulus). Ex: `11%5` *is* `1`.

- What are types of variables?
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*

- How can I tell strings from variables?
  Strings are surrounded by quotes (either single or double).
  Variables names (identifiers) for memory locations are not. Ex: '`num`' *vs.* `num`.

Today's Topics

- Recap: Indexing, Slicing, & Decisions
- Logical Expressions
- Circuits

Lecture Slip: In Pairs or Triples...

Some review:

1
```
motto = "Mihi cura futuri"
print(motto[2:4])
print(motto[2:4].upper())
```

2
```
ER = "The future belongs to those who believe in the beauty of their dreams."
print(ER.upper()[2], ER[13], ER[2], "a", ER[15], ER[14], "r  R.")
```

Recap: Indexing & Slicing

```
motto = "Mihi cura futuri"
print(motto[2:4])
print(motto[2:4].upper())
```

Recap: Indexing & Slicing

```
motto = "Mihi cura futuri"
print(motto[2:4])
print(motto[2:4].upper())
```

M	i	**h**	**i**		c	u	r	a		f	u	t	u	r	i
0	1	**2**	**3**	4	5	6	7	8	9	10	11	12	13	14	15

Output:

```
hi
HI
```

Recap: Indexing & Slicing

```
ER = "The future belongs to those who believe in the beauty of their dreams."
print(ER.upper()[2], ER[13], ER[2], "a", ER[15], ER[14], "r  R.")
```

T	h	**e**		f	u	t	u	r	e		b	e	**l**	**o**	**n**	g	s
0	1	**2**	3	4	5	6	7	8	9	10	11	12	**13**	**14**	**15**	16	17

Output:

E l e a n o r R.

In Pairs or Triples...

Some challenges with types & decisions:

```
#What are the types:
y1 = 2017
y2 = "2018"
print(type(y1))
print(type("y1"))
print(type(2017))
print(type("2017"))
print(type(y2))
print(type(y1/4.0))

x = int(y2) - y1
if x < 0:
    print(y2)
else:
    print(y1)
```

```
cents = 432
dollars = cents // 100
change = cents % 100
if dollars > 0:
    print('$'+str(dollars))
if change > 0:
    quarters = change // 25
    pennies = change % 25
    print(quarters, "quarters")
    print("and", pennies, "pennies")
```

Python Tutor

```
#What are the types:
y1 = 2017
y2 = "2018"
print(type(y1))
print(type("y1"))
print(type(2017))
print(type("2017"))
print(type(y2))
print(type(y1/4.0))

x = int(y2) - y1
if x < 0:
    print(y2)
else:
    print(y1)
```

(Demo with pythonTutor)

Decisions

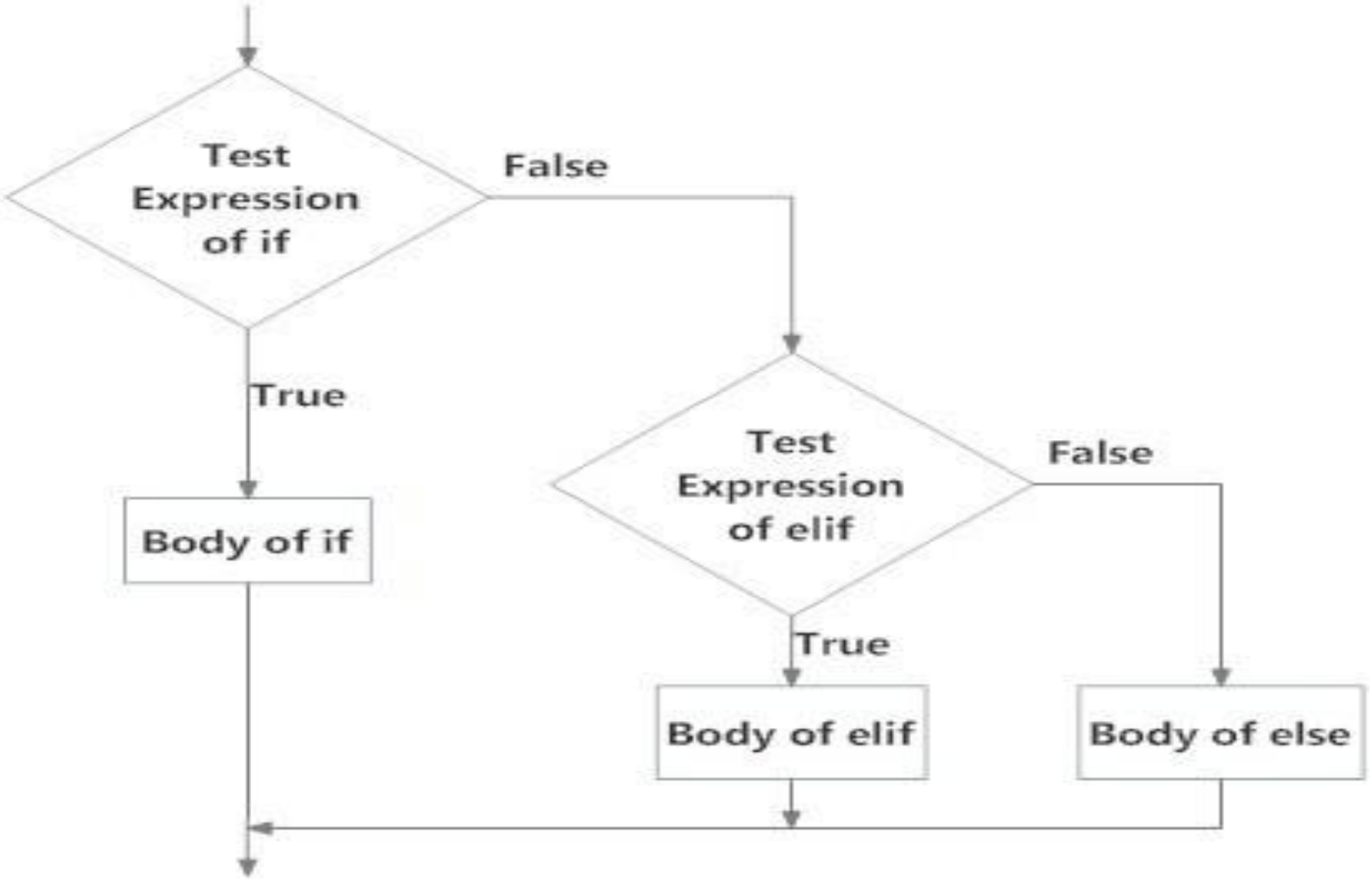
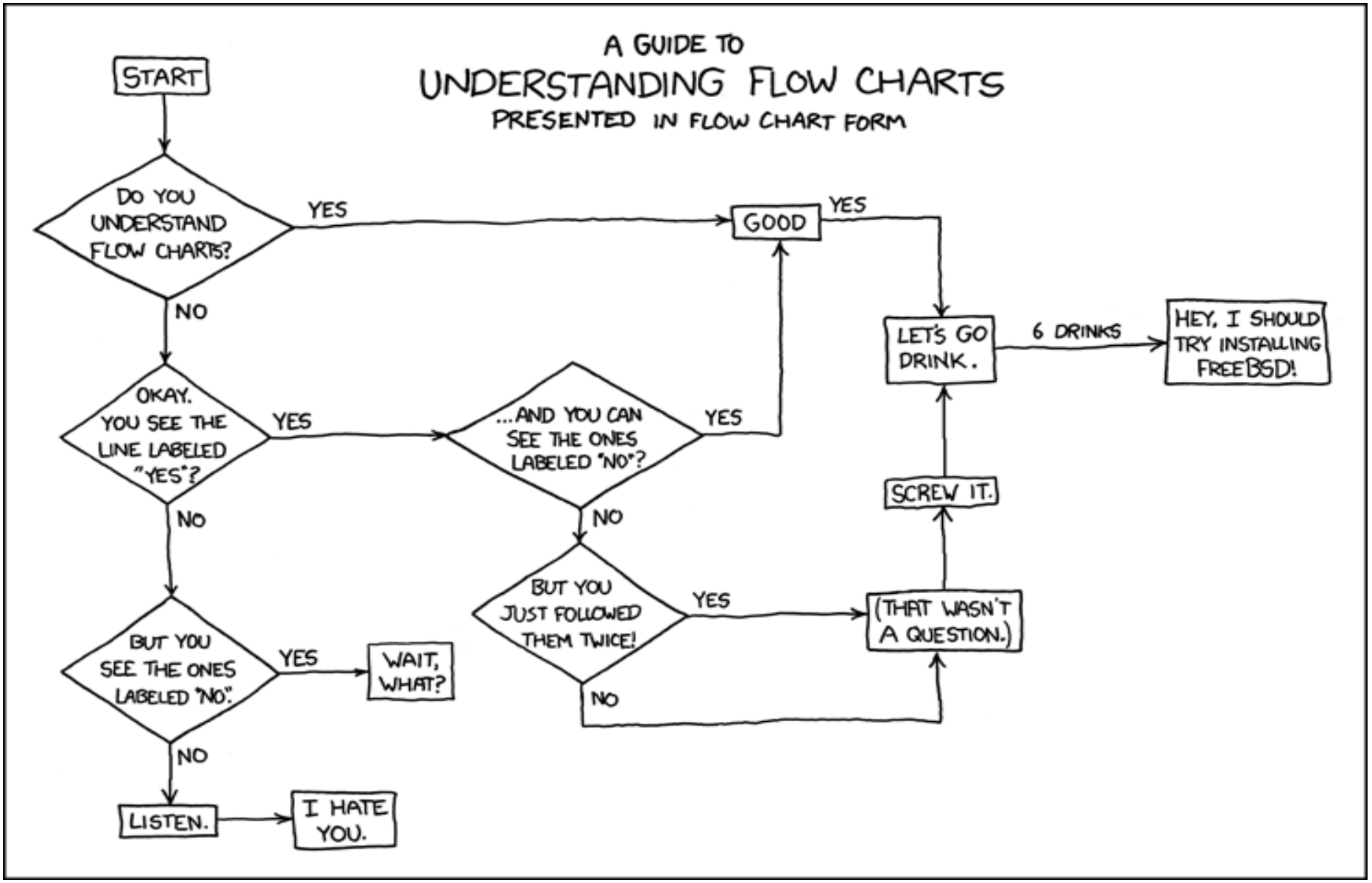


Fig: Operation of if...elif...else statement

Side Note: Reading Flow Charts



In Pairs or Triples

Predict what the code will do:

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
      (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Python Tutor

```python
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
      (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

(Demo with `pythonTutor`)

Logic gates

- **Boolean Operation:** An operation that manipulates one or more true/false values

- Specific operations

AND

OR

NOT

Logic gates

Figure 1.1 The possible input and output values of Boolean operations AND, OR, and XOR (exclusive or)

The AND operation

0	0	1	1
AND 0	AND 1	AND 0	AND 1
0	0	0	1

The OR operation

0	0	1	1
OR 0	OR 1	OR 0	OR 1
0	1	1	1

The XOR operation

0	0	1	1
XOR 0	XOR 1	XOR 0	XOR 1
0	1	1	0

Logic gates

- **Gate:** A device that computes a Boolean operation

  Often implemented as (small) electronic circuits
  Provide the building blocks from which computers are constructed

Logic gates

Types of Gates:

NOT, AND, OR ; NAND, NOR

XOR(Exclusive OR) , XNOR (Exclusive-NOR)

Logic gates

AND Gate

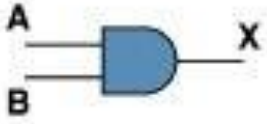
Has 2 inputs, single output

AND gate

Boolean Expression

$X = A \cdot B$

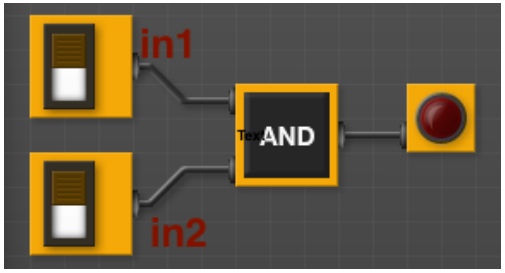
Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Circuit Demo



(Demo https://logic.ly/demo)

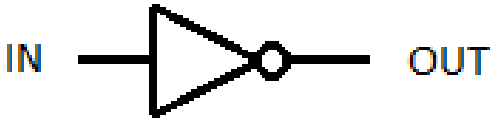
Logic gates

NOT Gate – Inverter

Has a single input, single output

NOT Gate

Symbol



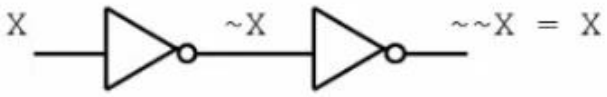
Logic function:

IN = A; OUT = Q;

Q = NOT A

Table of truth

IN	OUT
0	1
1	0

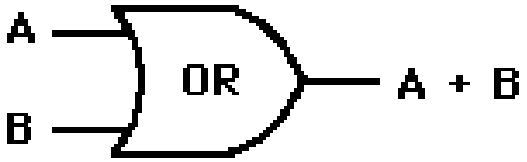


X	~X	~~X
0	1	0
1	0	1

Logic gates

OR Gate

Curved input (2 inputs , 1 output)



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

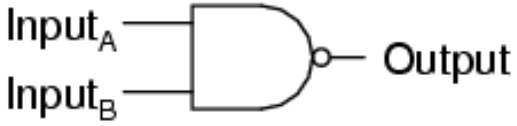
The OR operation will be signified by A + B. Other common mathematical notations for it are $A \vee B$ and $A \cup B$, called the union of A and B.

Logic gates

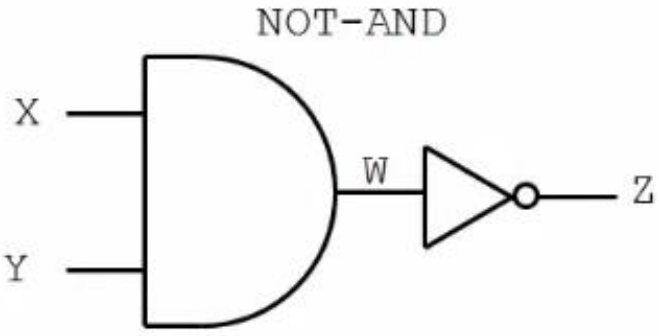
NAND Gate

NAND = AND + NOT

NAND gate



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

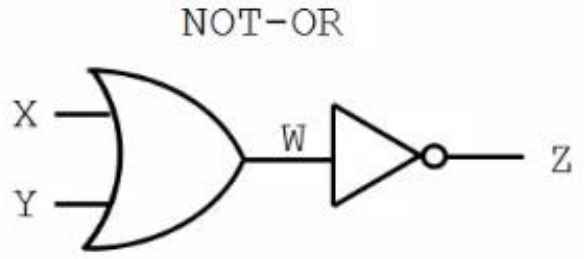


X	Y	W	Z
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Logic gates

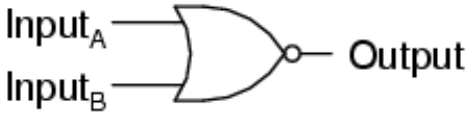
NOR Gate

NOR = OR + NOT



X	Y	W	Z
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR gate



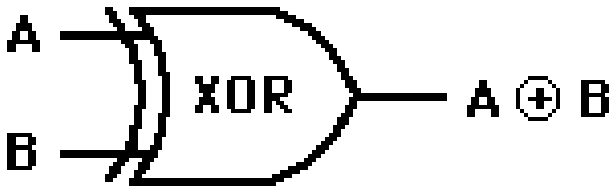
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

Logic gates

Exclusive-OR Gate (XOR)

OUT-PUT = 1 (If either input is one, other = 0)

‘0’ if Both are ‘1’ or both are ’0’

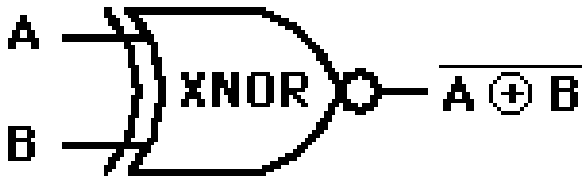


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Logic gates

Exclusive-NOR Gate

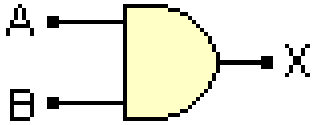
XNOR = XOR + NOT



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Logic gates

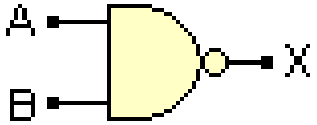
AND



$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

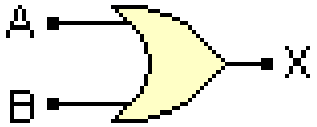
NAND



$X = \overline{A \cdot B}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

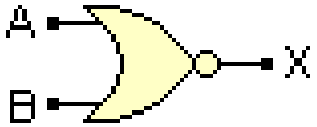
OR



$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

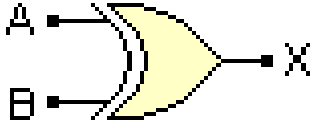
NOR



$X = \overline{A + B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

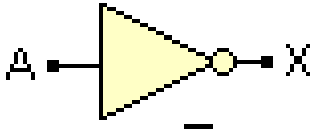
XOR



$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Inverter (NOT)



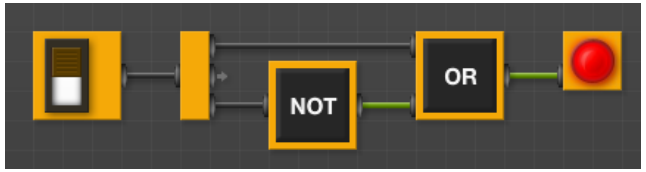
$X = \overline{A}$

A	X
0	1
1	0

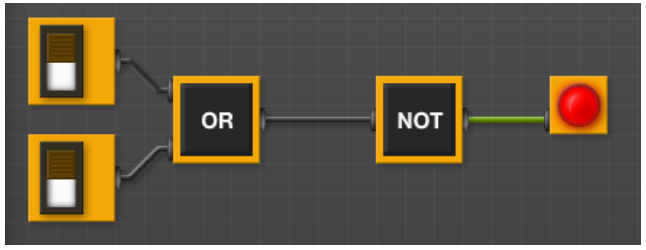
In Pairs or Triples

Predict when these expressions are true:

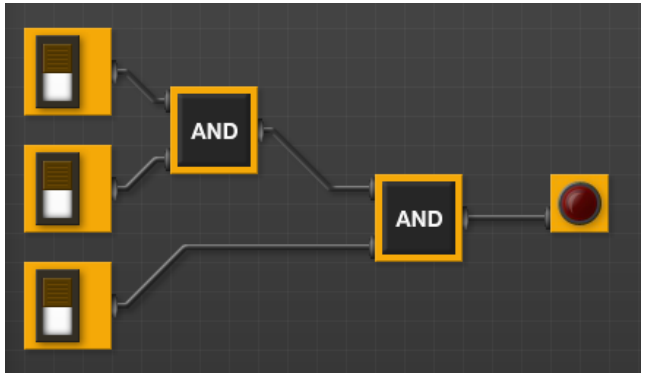
- `in1 or not in1`:



- `not(in1 or in2)`:



- `(in1 and in2) and in3`:



Logical Operators

and

in1		in2	*returns:*
False	and	False	False
False	and	True	False
True	and	False	False
True	and	True	True

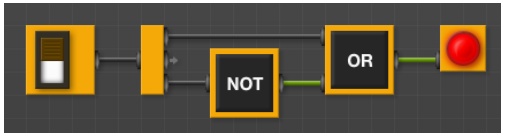
or

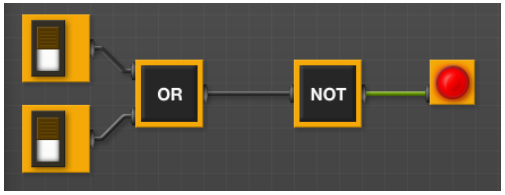
in1		in2	*returns:*
False	or	False	False
False	or	True	True
True	or	False	True
True	or	True	True

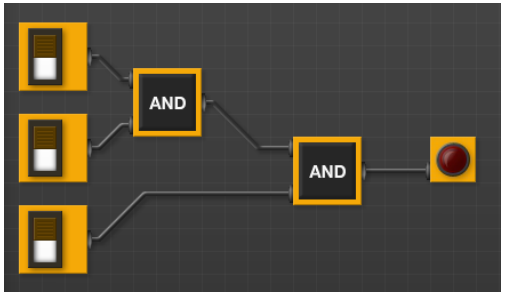
not

	in1	*returns:*
not	False	True
not	True	False

Circuit Demo







(Demo https://logic.ly/demo)

Sample exam question

From Final Exam, Fall 2017, Version 3:

Name: **EmpID:** CSci 127 Final, V3, F17

1. (a) What will the following Python code print:

```
flist = "speech,worship,want,fear,fdr"
freedoms = flist.split(",")
pres = freedoms[-1]
print(pres.upper())
num = flist.count(",")
print(num, "Freedoms")
for i in range(0,4):
     if i < 2:
          print("\tof", end=" ")
     else:
          print("\tfrom", end=" ")
     print(freedoms[i])
```

Output: