Binary : 2 values

- True / False
- 1 / 0

bools

1 : True

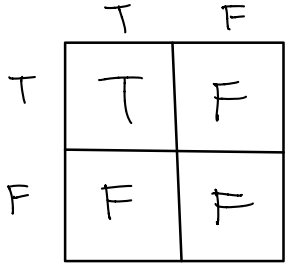
0 : False

Boolean operators / expressions : produce boolean value

↳ examples : and, or, xor, not

2 inputs 1 input

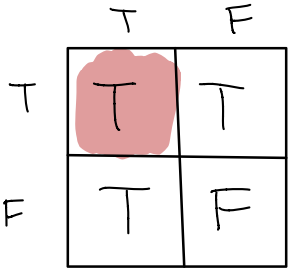
and



x and y

both x, y have to be true for expression to be true

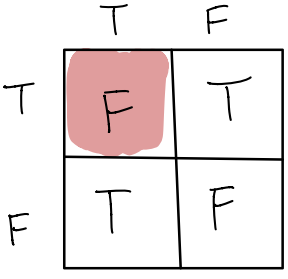
or



x or y

either x or y have to be true for expression to be true

xor



x xor y

stricter definition on or.

both x and y cannot be true at same time

not

input:	T	F
output:	F	T

* only has 1 input

not x

flips boolean value of x

true becomes false,

false becomes true

print (not (3 == 3))

⇒ False

Truth tables: can help us construct more complex boolean operators

a xor b CAN'T write in python

Thought question: create an xor function using not, and, or and a truth table

Hint: think deeply about what xor means

Define P xor Q

P	Q	P and Q	not(P and Q)	P or Q	P or Q AND not(P and Q)	P xor Q
T	T	T	F	T	F	F
T	F	F	T	T	T	T
F	T	F	T	T	T	T
F	F	F	T	F	F	F

Other functions which output boolean value: ≥, >, ≤, <, ==, !=

True and False

print (5 ≥ 3) True

print (3 == 3) True

print (3 != 4 and 3 == 3) True

print (3 != 3 ~~or~~ and 3 == 3) True ⇒ False

↳ These typically take in floats / ints

a = 17 $\quad$ 3 == 3 $\quad$ ~~3 == 5~~ $\quad$ 3 == 4 ✗

name = "Swetha" ↳ == vs. = $\quad$ comparison vs. assignment

↳ ! vs. not

a = 3

print (a = 3) # not a real thing

print (a == 3) # depends on a

mean the same thing:

- print (not (~~a == 3~~ True)) ⟹ False
- print (a != 3) ⟹ False

bool = boolean $\quad$ bool (1) ⟹ True # any value other than 0

print(bool (0)) ⟹ False

Short circuiting: