

Control Flow

Booleans: True False

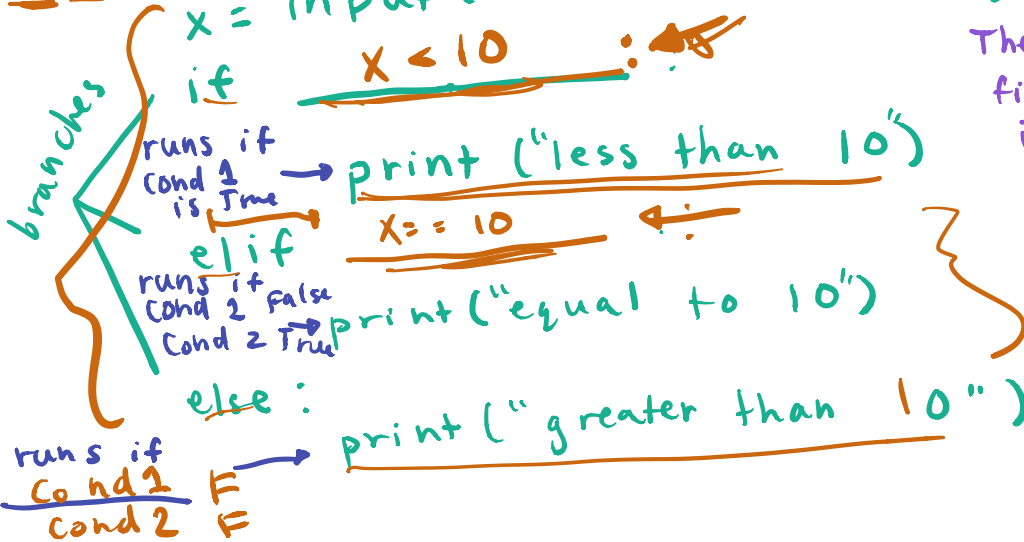
Comparison Operators:

Evaluate to booleans

Fill in

	Ex.	Evaluates To
<code>==</code> equal to	<code>6 == 7</code>	False
<code>!=</code> not equal to	<code>6 != 7</code>	True
<code>></code> greater than	<code>6 > 7</code>	False
<code><</code> less than	<code>6 < 7</code>	True
<code>>=</code> greater than or equal to	<code>6 >= 7</code>	False
<code><=</code> less than or equal to	<code>6 <= 7</code>	True

True or False



The computer first looks at if and in the end only the commands of one of the branches is executed.

What would Python print if :

x = 5 ? less than 10
x = 20 ? greater than 10
x = 10 ? equal to 10

Logical Operators: and, or, not

negation

both either

True and True : True

false and False : False

True and False: False

True or False: True

True or True: True

False or False: False

not True: False

Write an if statement that prints
"Let's go to the beach" if it's hot
and it's summer.

```
hot = input("Is it hot? Yes or No: ")
```

```
summer = input("Is it summer? Yes or No: ")
```

Hint: You can compare strings to see if they're
equal using ==

Short circuiting:

ex. False and True

(not True) and False

False and False

Useful for avoiding errors

ex. $5/x > 1$ vs $x \neq 0$ and $5/x > 1$

Loops:

for loop:
for i in range(num):
line to execute

while loop:
while (x < 10):
line to execute

Ex. Using a for loop, print the numbers 1 to 5

→ for i in range(5): ^{start 0} end: n-1
 print(i+1) _n i = 0, 1, 2, 3, 4
1 2 3 4 5 i = n-1

Ex. Using a while loop, print the numbers 1 to 5

x = 1
→ while (x < 6)
 print(x)
 x += 1
x 1 2 3 4 5 6

Generally use while loop when you don't know how many times you want your code to run.

Use a for loop to print all even numbers from 0 to x where x is a user inputted number.

[1, 2, 5, 7, 9]

```
i = 0  
while (x[i] != 5)  
    print(x[i])  
    i += 1
```