Intro to Coding with Python– Conditionals

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

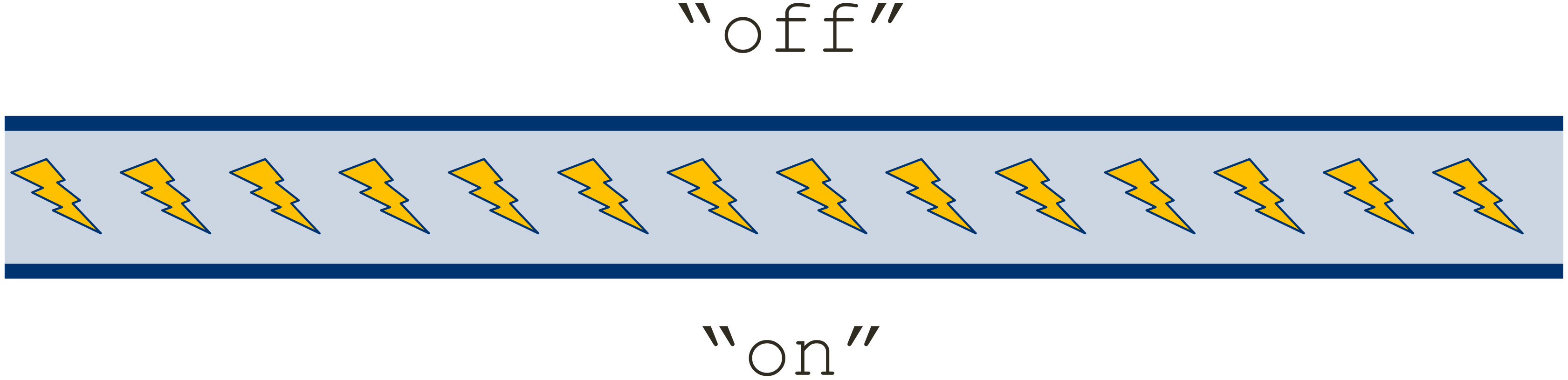
Plan for Today

- Intro to conditionals

RECAP

How is information represented using **electricity**?

One wire: a "bit"



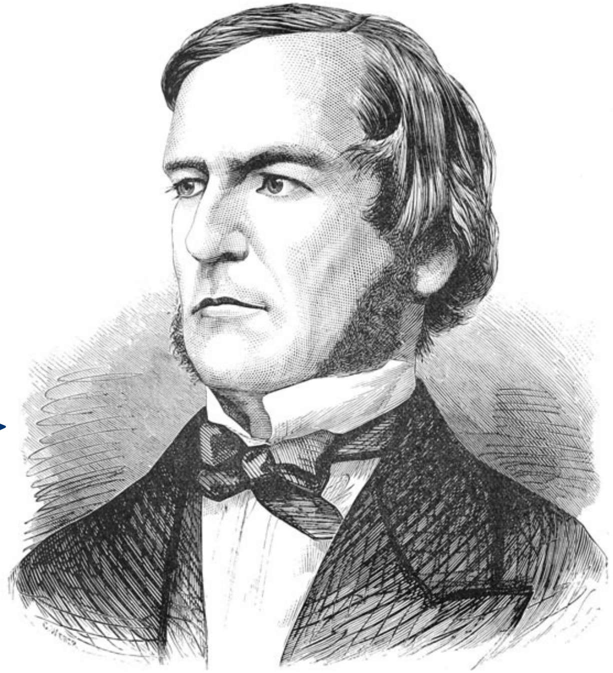
Bits and booleans

- **Bits**: 0 and 1
- **Boolean values**: `True` and `False`
- **Boolean switches**: imagine a world where every decision has a binary choice:

Go out or **stay in**?

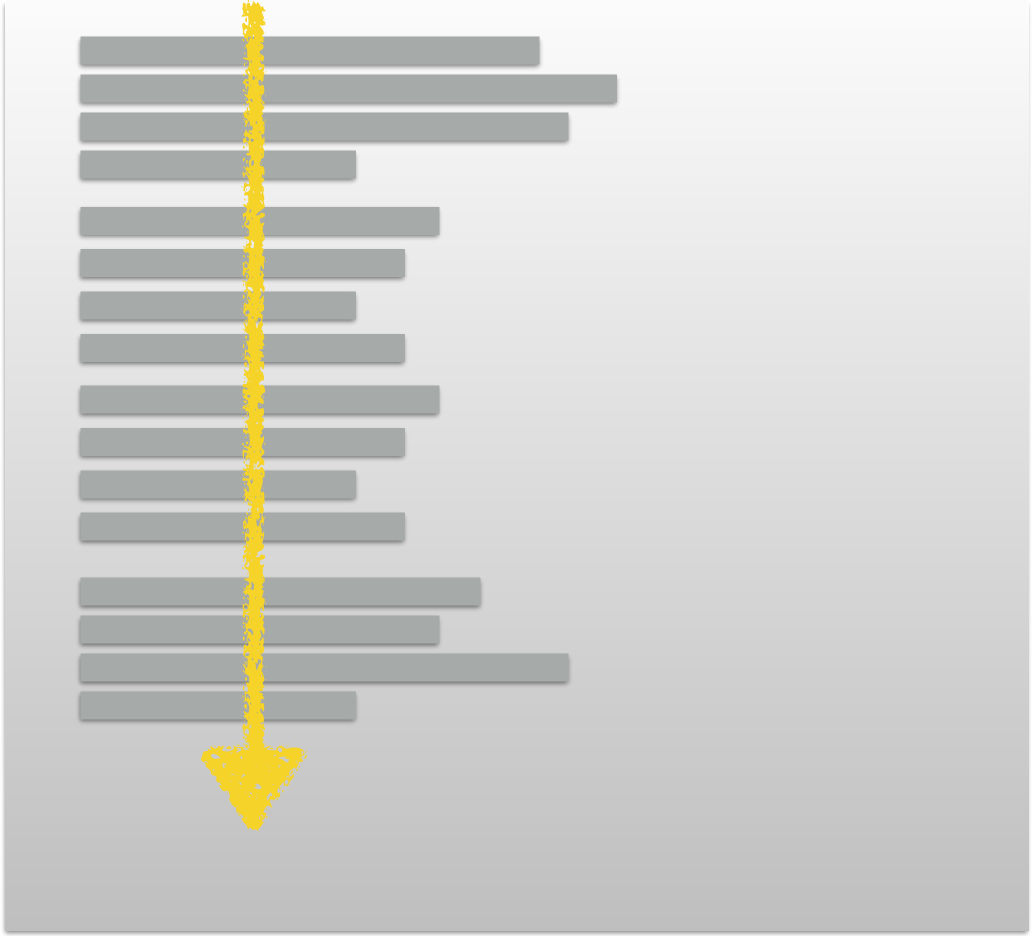
Walk or **take the car**?

Cats or **Dogs**?

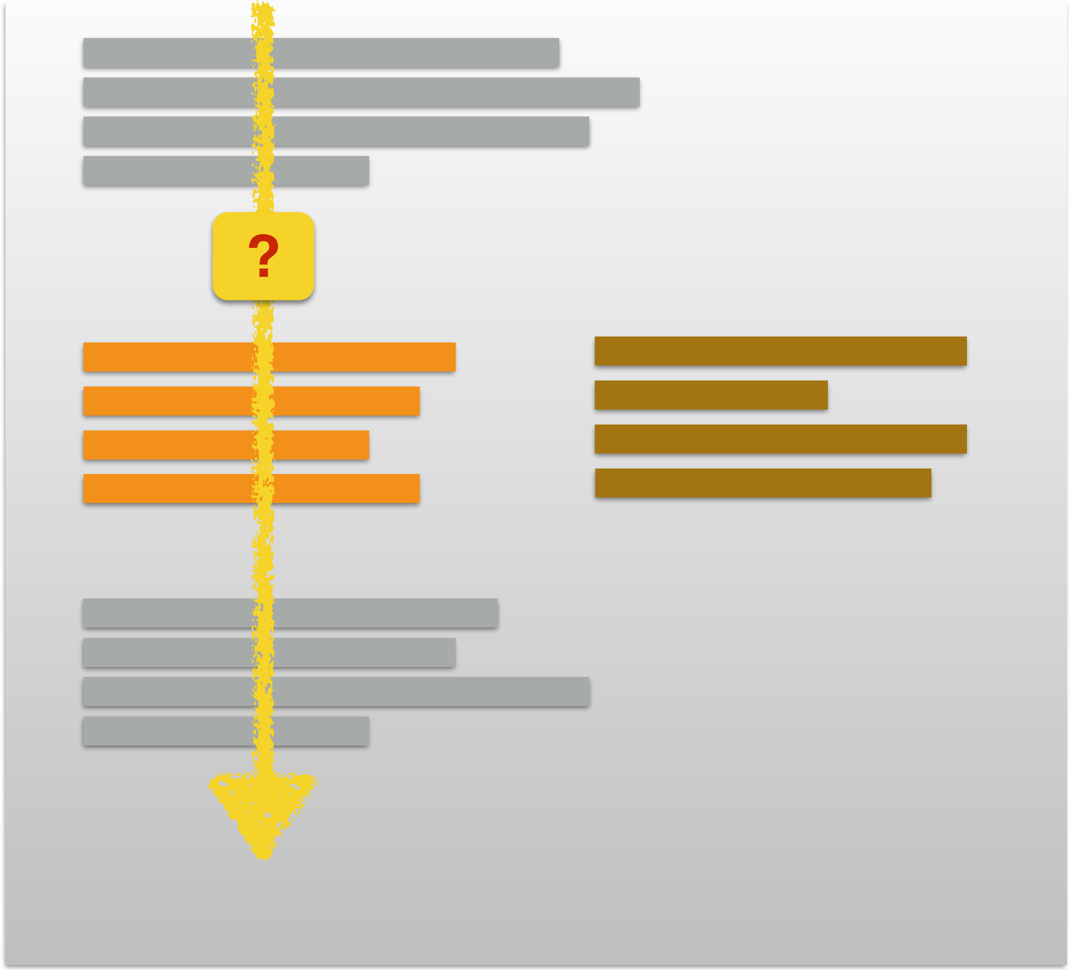


George Boole
1815 - 1864

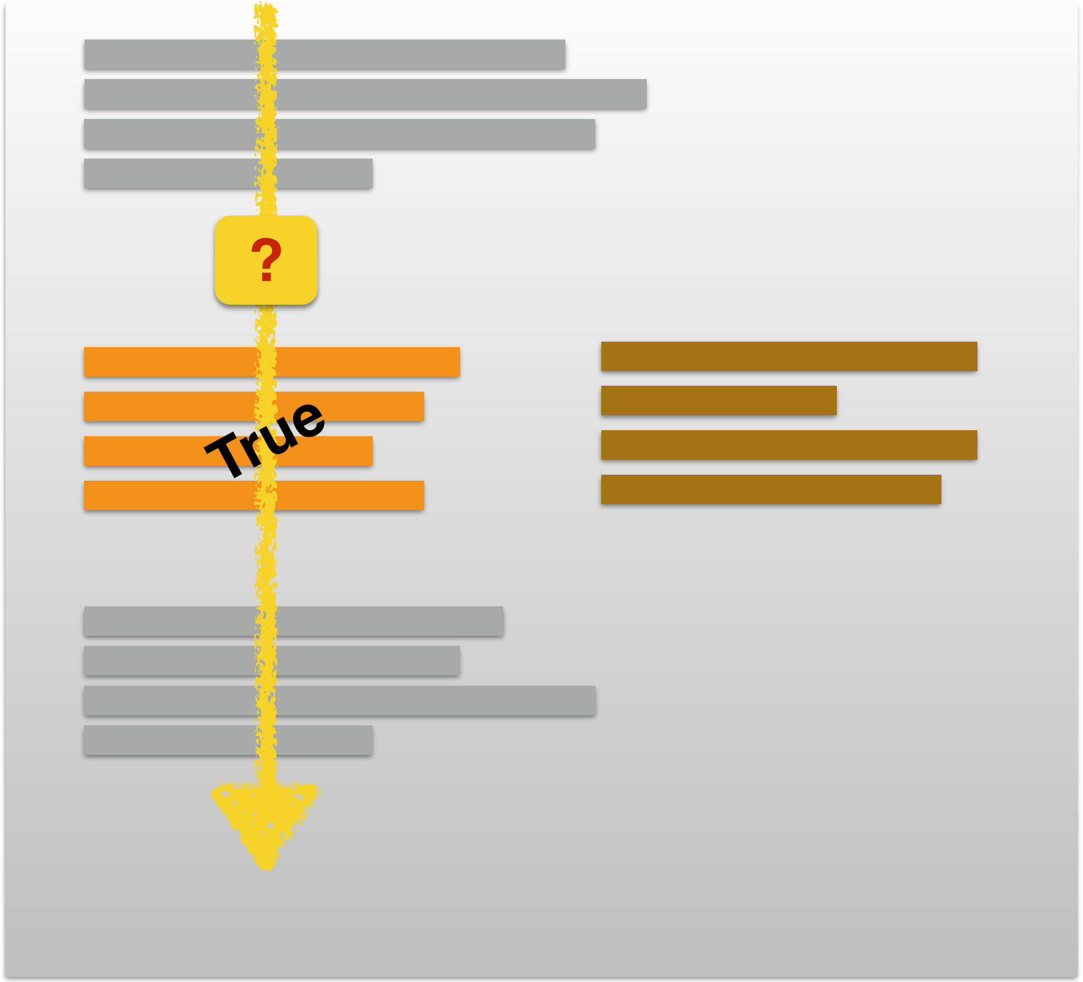
So far: linear programs



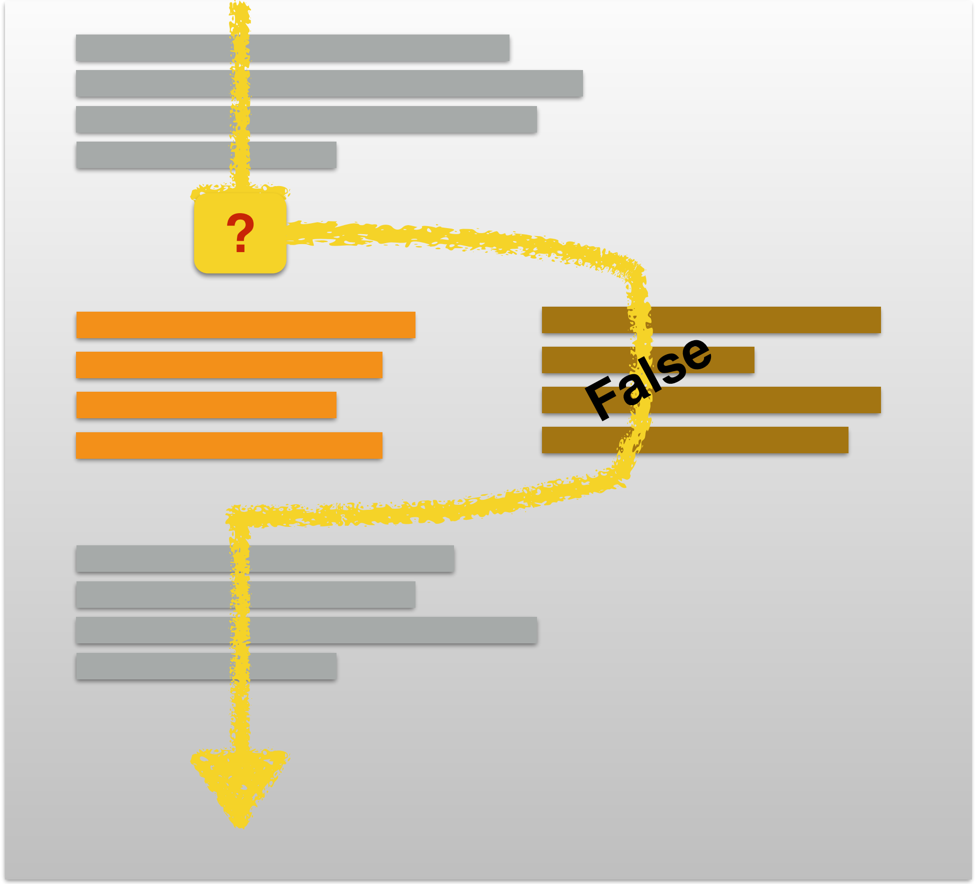
What if we need to make a **choice**?



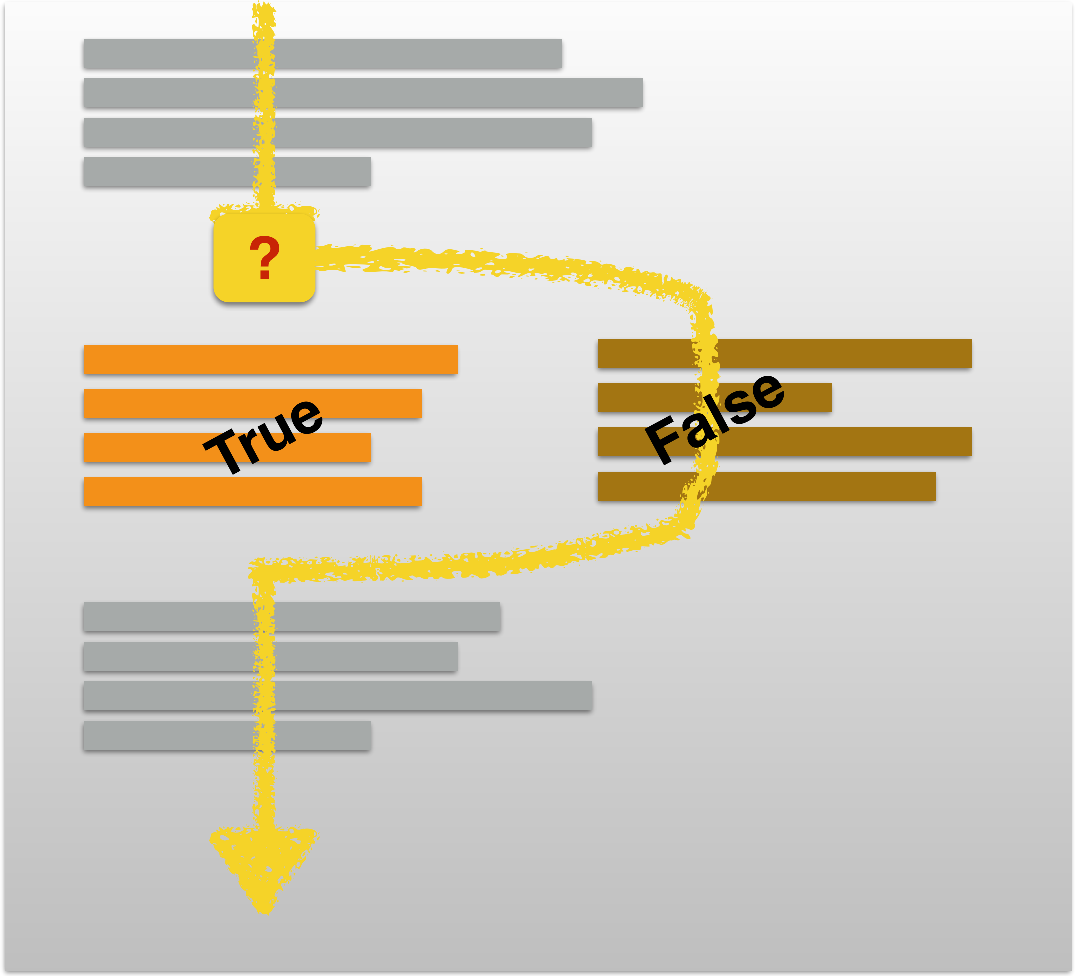
Booleans to the rescue!



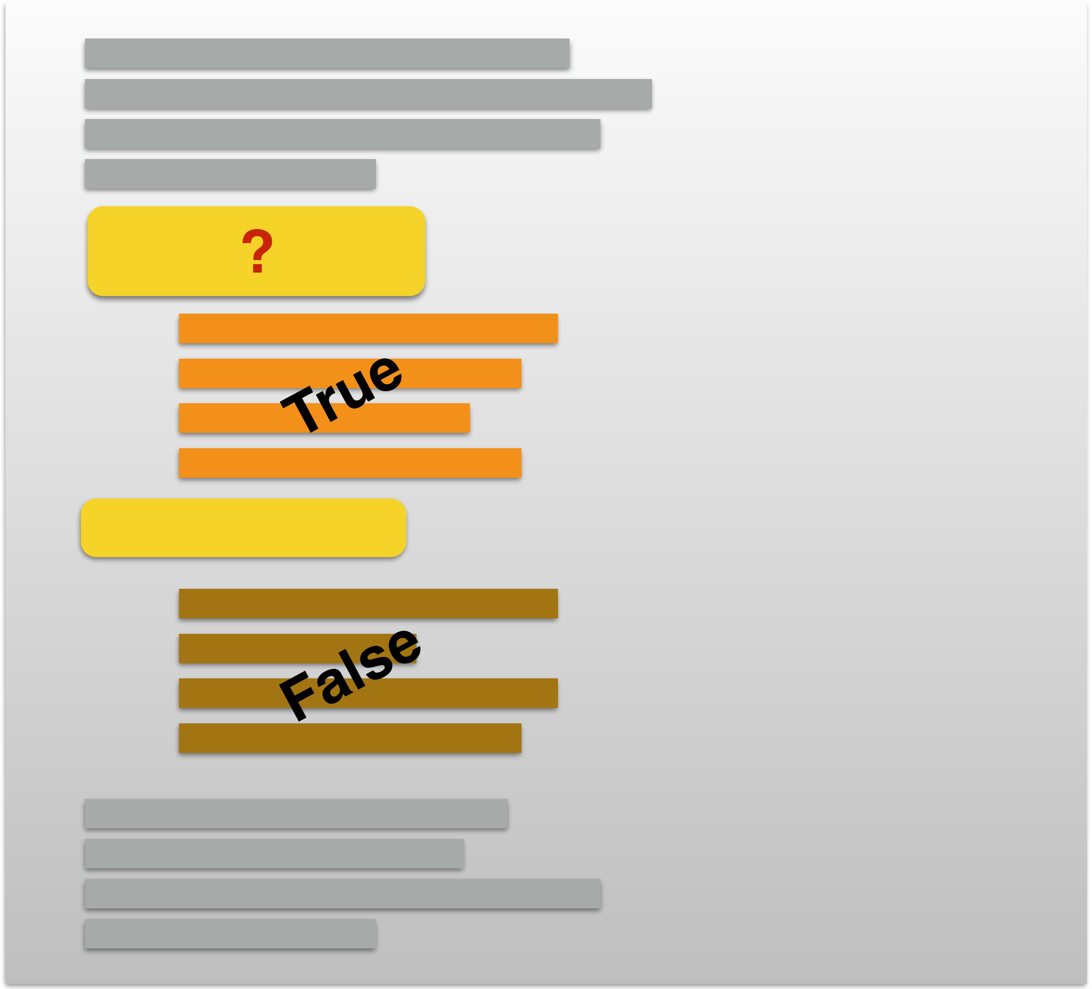
Booleans to the rescue!



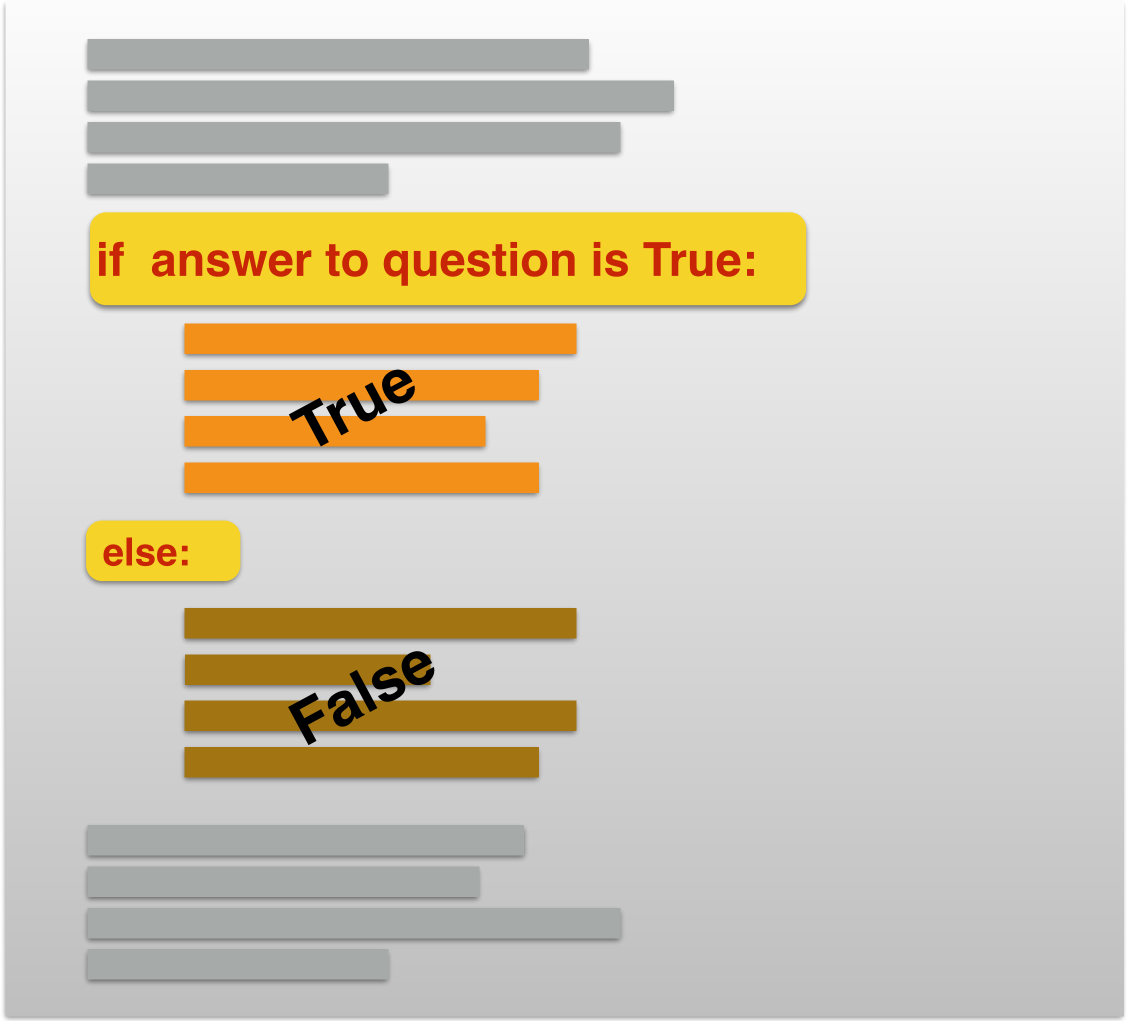
Just one **problem**: how do we write it?



We can only type **one line** at a time…



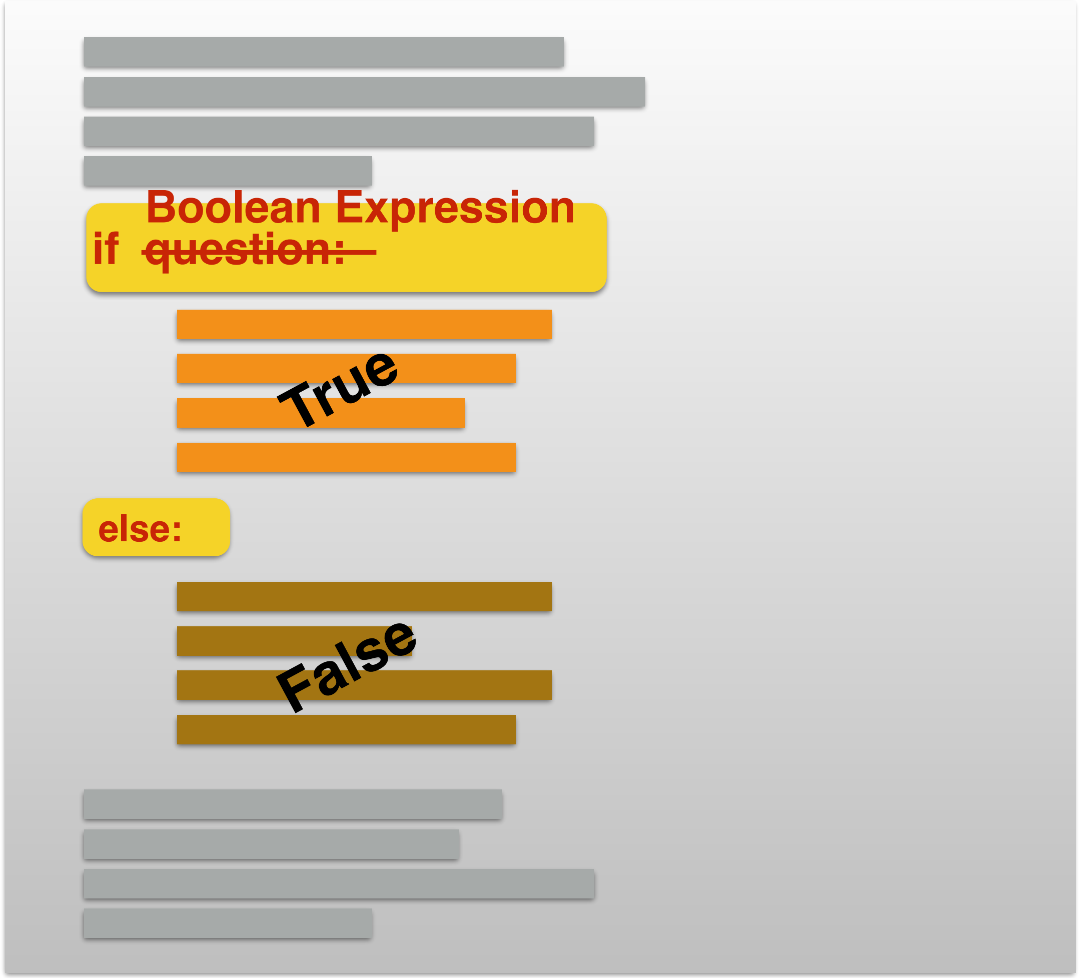
What we **want** to say



if answer to question is True:

else:

What we have to **work with**



Boolean Expression

if ~~question:~~

True

else:

False

Real life examples (pseudocode)

```
if (today is a weekday):
      go to class
else: # (today is a weekend)
      sleep in
```

Real life examples (pseudocode)

```
if (today is a weekday):
      go to class
else: # (today is a weekend)
      sleep in

if (food at dining hall looks good):
      eat at dining hall
else: # food at dining hall doesn’t look good
      order Domino’s
```

Real life example (change machine)

How many 20s to get to total amount of dollars?

```
Ex. User inputs $71
    Output should be 3 $20-bills
```

print the ”s” only if necessary

```
Ex. User inputs $21
    Output should be 1 $20-bill
```

print the ”s” only if necessary

```
Ex. User inputs $5
    Output should be 0 $20-bills
```

print the ”s” only if necessary

Real life example (change machine)

```
How many 20s to get to total amount of
dollars?


Ex. User inputs $71
    Output should be 3 $20-bills
```

print the ”s” only if necessary

```
Ex. User inputs $21
    Output should be 1 $20-bill
```

print the ”s” only if necessary

```
Ex. User inputs $5
    Output should be 0 $20-bills
```

print the ”s” only if necessary

What is the if-else statement for this in pseudo code?

Real life example (change machine)

How many 20s to get to total amount of dollars?

```
Ex. User inputs $71
    Output should be 3 $20-bills
Ex. User inputs $21
    Output should be 1 $20-bill
Ex. User inputs $5
    Output should be 0 $20-bills
```

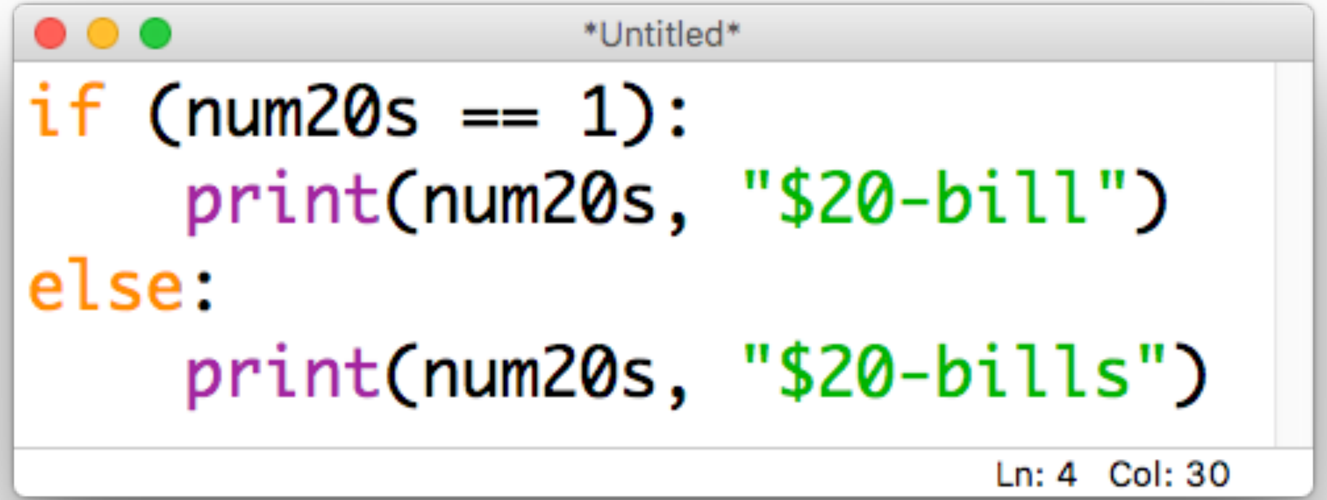
print the ”s” only if necessary

print the ”s” only if necessary

print the ”s” only if necessary

```
if (only one 20):
   output message has no s
else: #multiple or no 20s’
   output message has s
```

Real life example (change machine)



```
if (num20s == 1):
    print(num20s, "$20-bill")
else:
    print(num20s, "$20-bills")
```

Relational operators

Operator	Meaning
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

these come in handy when constructing **boolean statements**

Demo!

Operator	Meaning
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Multiple conditions

```
if (it is sunny):
      go to the beach
if (it is snowy):
      go skiing
else:
      stay home
```

Sequential `if` statements are **independent**

```
if (it is sunny):
      go to the beach
if (it is snowy):
      go skiing
else:
      stay home
```

Sequential `if` statements are **independent**

```
if (it is sunny):
        go to the beach
if (it is snowy):
        go skiing
else:
        stay home
```

this line will **always** run

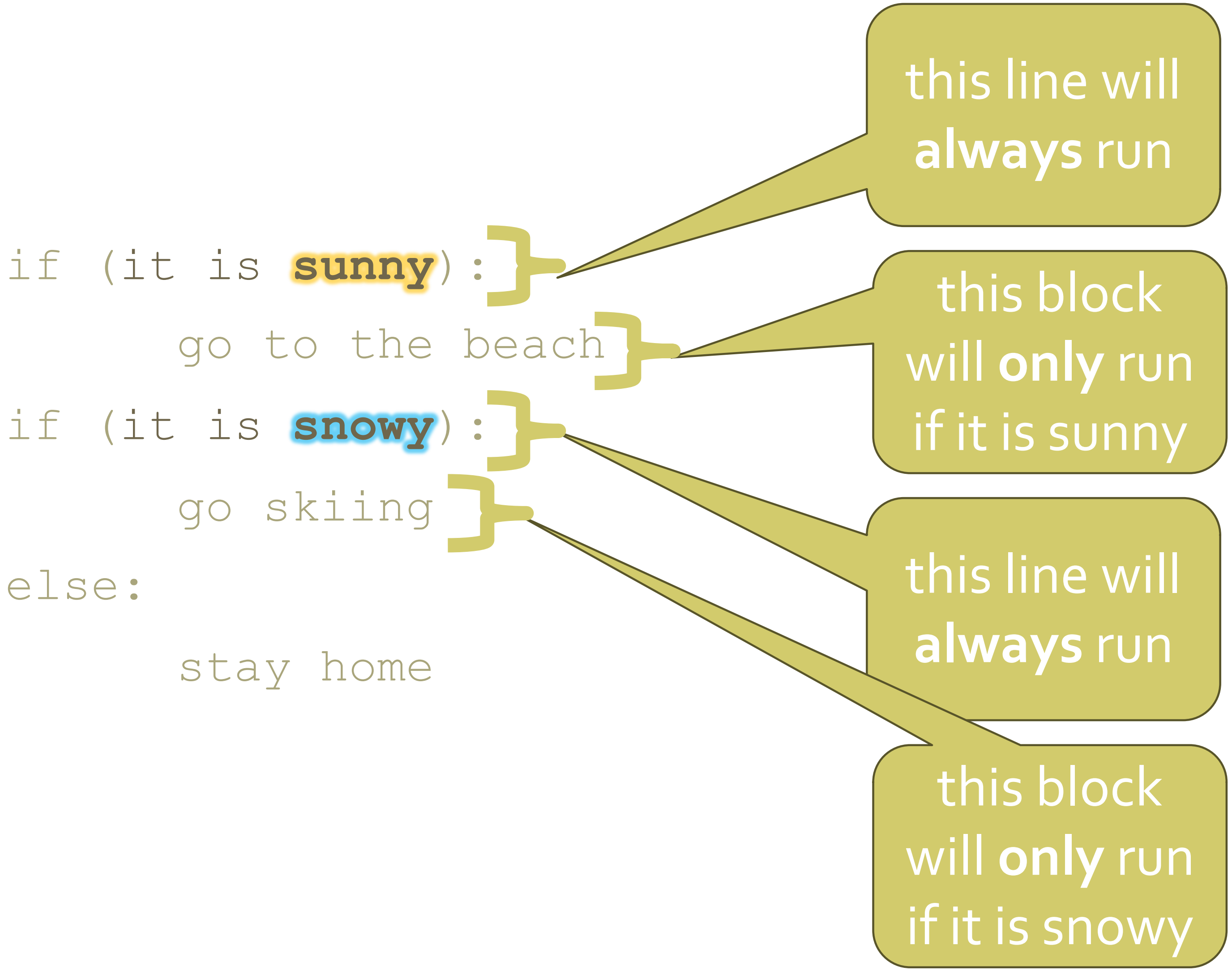
Sequential **`if`** statements are **independent**

```
if (it is sunny):
      go to the beach
if (it is snowy):
      go skiing
else:
      stay home
```

this line will **always** run

this block will **only** run if it is sunny

Sequential `if` statements are **independent**



```
if (it is sunny):
      go to the beach
if (it is snowy):
      go skiing
else:
      stay home
```

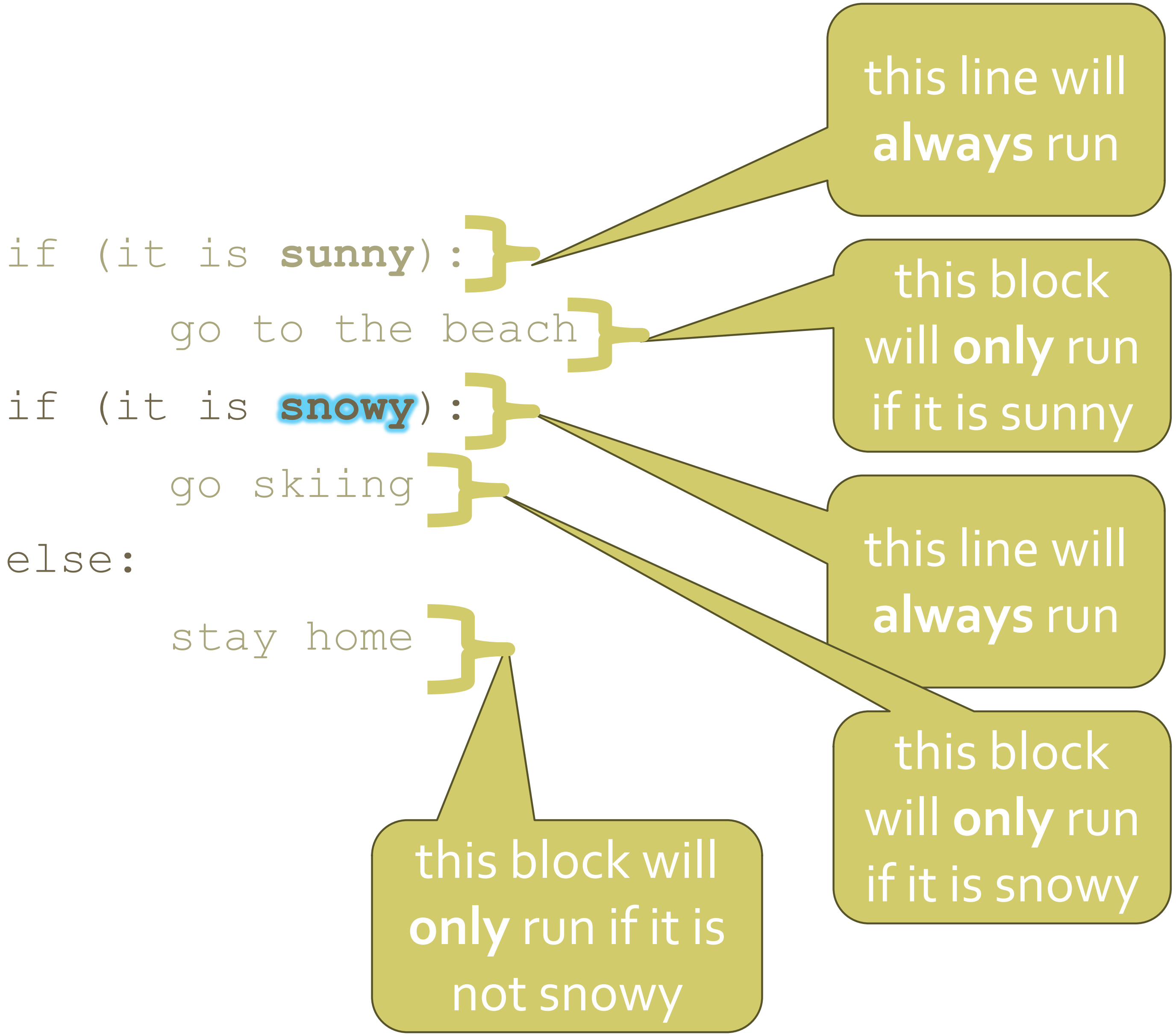
this line will **always** run

this block will **only** run if it is sunny

this line will **always** run

this block will **only** run if it is snowy

The `else` refers only to the **nearest** `if`



```
if (it is sunny):
        go to the beach
if (it is snowy):
        go skiing
else:
        stay home
```

this line will **always** run

this block will **only** run if it is sunny

this line will **always** run

this block will **only** run if it is snowy

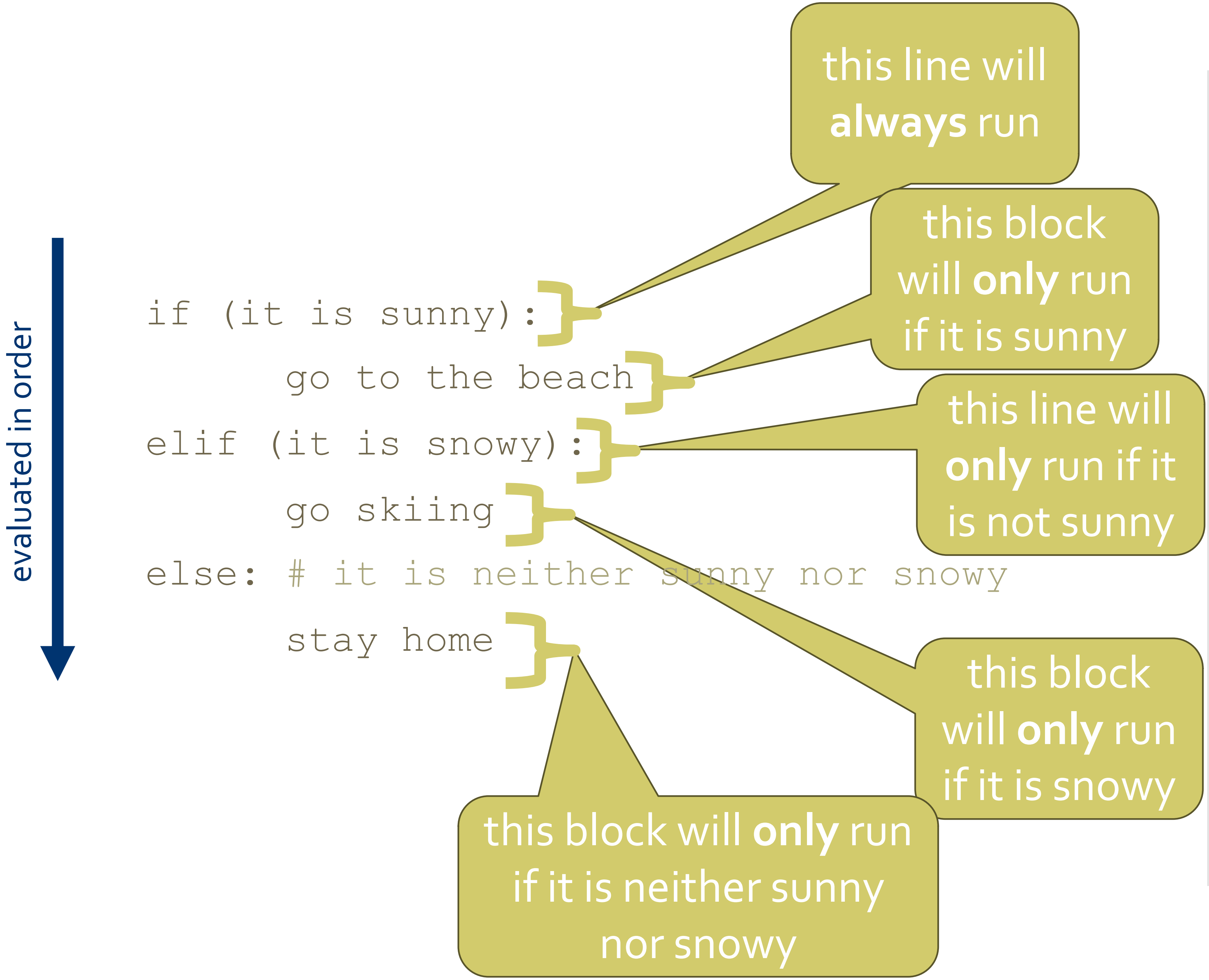
this block will **only** run if it is not snowy

To chain multiple “checks” together: **`elif`**

evaluated in order

```
if (it is sunny):
        go to the beach
elif (it is snowy):
        go skiing
else: # it is neither sunny nor snowy
        stay home
```

To chain multiple “checks” together: `elif`



evaluated in order

```
if (it is sunny):
      go to the beach
elif (it is snowy):
      go skiing
else: # it is neither sunny nor snowy
      stay home
```

this line will **always** run

this block will **only** run if it is sunny

this line will **only** run if it is not sunny

this block will **only** run if it is snowy

this block will **only** run if it is neither sunny nor snowy

Remember: order matters!

evaluated in order

```
if (it is sunny): # regardless of snow
      go to the beach
elif (it is snowy): # but not sunny
      go skiing
else: # it is neither sunny nor snowy
      stay home
```

Remember: order matters!

evaluated in order

```
if (it is snowy): # regardless of sun
    go skiing
elif (it is sunny): # but not snowy
    go to the beach
else: # it is neither sunny nor snowy
    stay home
```

Nested conditions

```
if (class is cancelled):
    if (you have homework):
        work on homework
    else: # class cancelled, no HW
        binge-watch Netflix
```

Simultaneous conditions

```
if (it’s Friday and it’s 4pm):
      go to tea


if (you’re hungry or you’re bored):
      go to the CC
```

Work with someone near you to write a program that:

1. Asks the user for a number between -10 and 10
2. Prints “even” if the number is even and “odd” if the number is odd
3. Prints “positive” if the number is positive, “negative” if the number is negative, and nothing (“”) if the number is 0
4. **Challenge**: Have your program print “the number is even and positive” if the number is even and positive, “the number is odd and positive” if the number is odd and positive, etc.