

Booleans, Logic, & Conditional “if” Statements.

Week 2 | Lecture 1 (2.1.1)

While waiting for class to start:

Download and open the Jupyter Notebook (.ipynb) for Lecture 2.1.1

You may also use this lecture’s JupyterHub link instead (although opening it locally is encouraged).

Upcoming:

- Lab 2 deadline this Friday @ 11 PM
- Lab 3 released Friday @ 1 PM
- Reflection 2 released Friday
- PRA (Lab) on Friday @ 2PM this week

if nothing else, write `#cleancode`

Today's Content

- **Lecture 2.1.1**
 - **Booleans, Logic, & Conditional if Statements**
- **Lecture 2.1.2**
 - String Comparisons and More on if Statements

Coffee Break with Ben/Craig/Behrang!

- Extra help hours!
 - **TIP FOR SUCCESS:** Put in calendar, treat as a scheduled class
- Schedule and Link is on Quercus



Coffee Break (Office Hours)

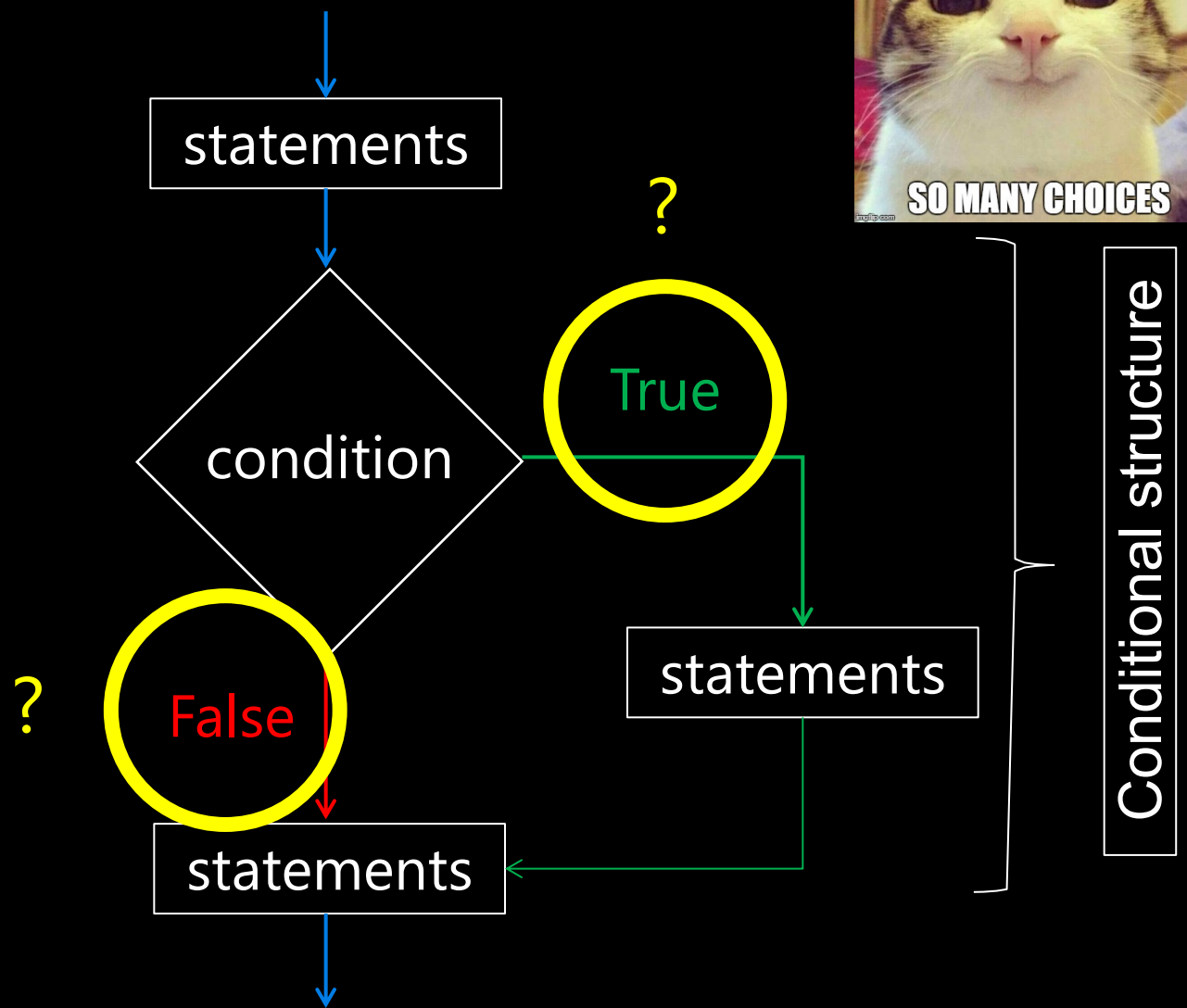
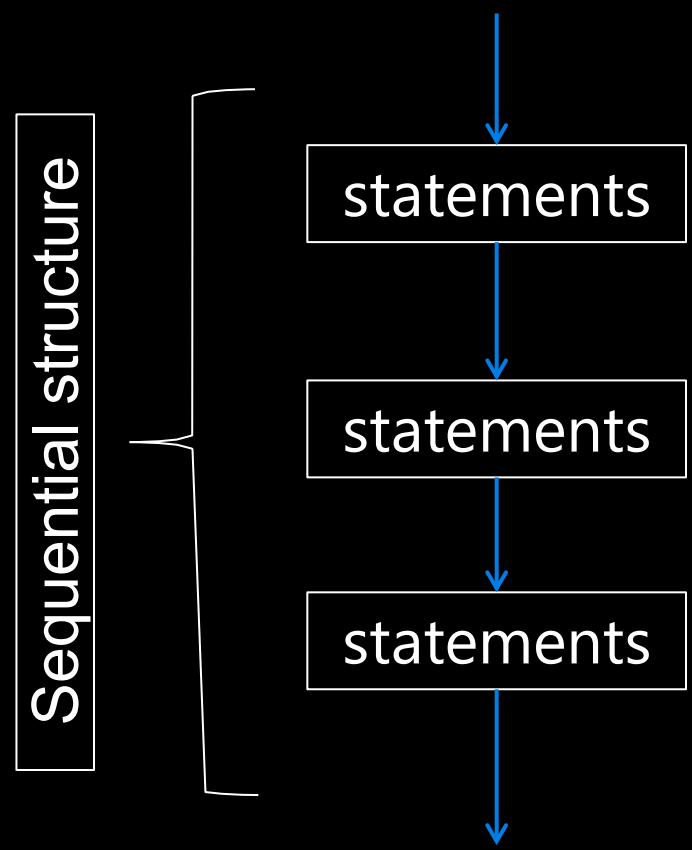
Instructor	Time	Location/Link
Ben	Wednesday @ 1-2 PM	Zoom Link
Craig	Tuesday @ 1-2 PM	(SF 3202?)
Behrang	Friday @ 1 - 2 PM	Zoom Link



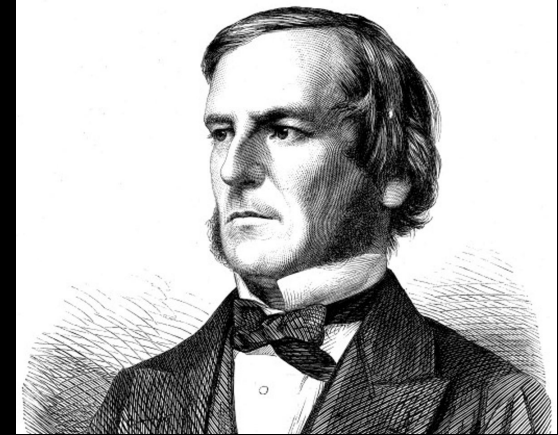
**Office
Hours**

**Coffee
Break**

Making Choices



Boolean Type (bool)

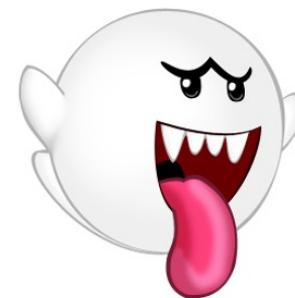


- Named after George Boole (mid-1800s)
 - Boolean algebra and Boolean logic
 - Laid the foundation for information age and computer science
- Python type bool has only two possible values: **True** and **False**

```
>>> state = True
>>> state
True
```

```
>>> state = False
>>> state
False
```

"bool" is a sub-type of "int", where `True == 1`, `False == 0`

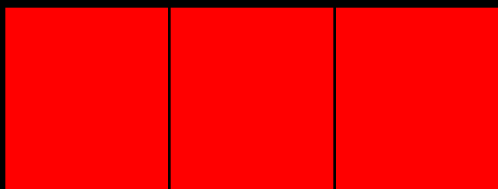
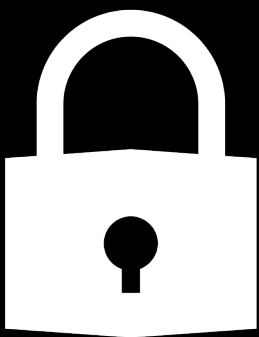


Boo



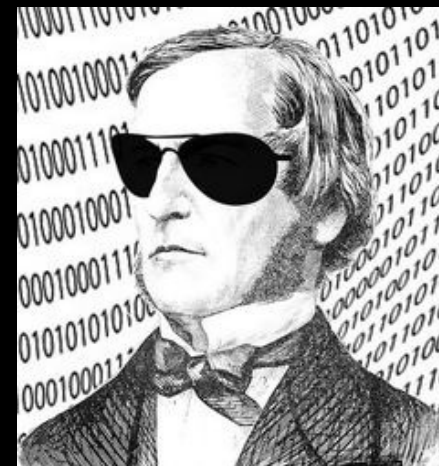
Boolean

Boolean Example / "Wordle" Break



CODE

Crack the code!



Cool Boole

6	8	2
6	1	4
2	0	6
7	3	8
8	7	0

One number is correct and well placed

One number is correct but wrong place

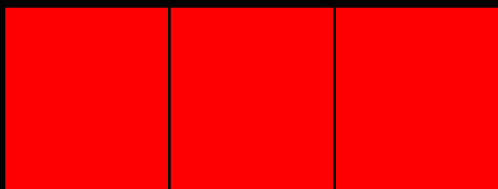
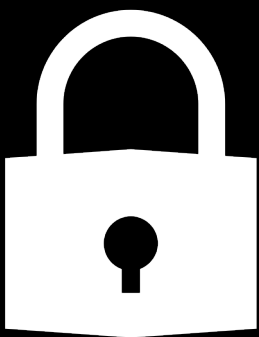
Two numbers are correct but wrong places

Nothing is correct

One number is correct but wrong place

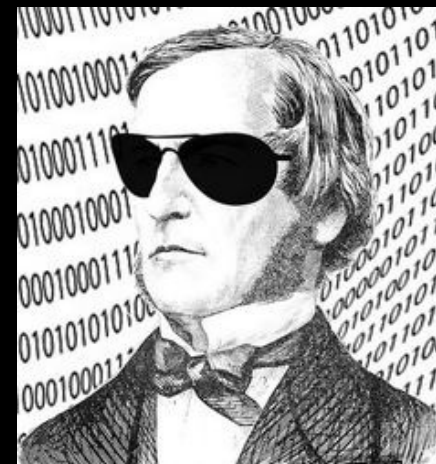
Boolean Example / "Wordle" Break

02:00



CODE

Crack the code!



6	8	2	One number is correct and well placed
6	1	4	One number is correct but wrong place
2	0	6	Two numbers are correct but wrong places
7	3	8	Nothing is correct
8	7	0	One number is correct but wrong place

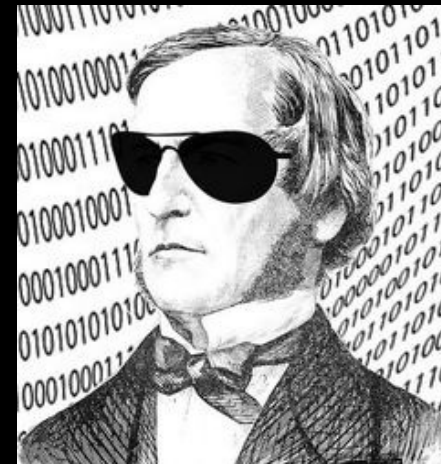
Boolean Example / "Wordle" Break



0 4 2

CODE

Crack the code!



6 8 2

One number is correct and well placed

6 1 4

One number is correct but wrong place

2 0 6

Two numbers are correct but wrong places

7 3 8

Nothing is correct

8 7 0

One number is correct but wrong place

Relational Operators

- Relational (or comparison) operators take two values (examples: `int`, `float`, `str`) and produce a `bool` value (True or False)

Description	Operator	Example	Result
Less than	<	3<4	True
Greater than	>	3>4	False
Equal to	==	3==4	False
Less than or equal to	<=	3<=4	True
Greater than or equal to	>=	3>=4	False
Not equal to	!=	3!=4	True

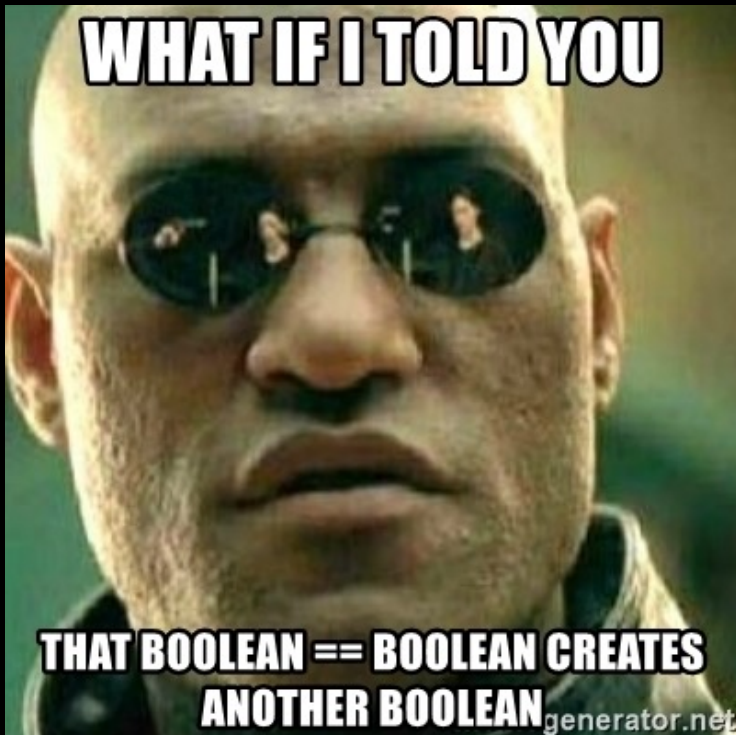
Boolean
Expressions

Boolean
Values

Python uses `==` for equality,
because `=` is used for assignment

Using Python as your (bool) Calculator

- Let's take a look at how this works in Python!
 - Boolean type
 - Relational (or comparison) operators



**Open your
notebook**

Click Link:

**1. Introducing
Booleans**

Logical Operators

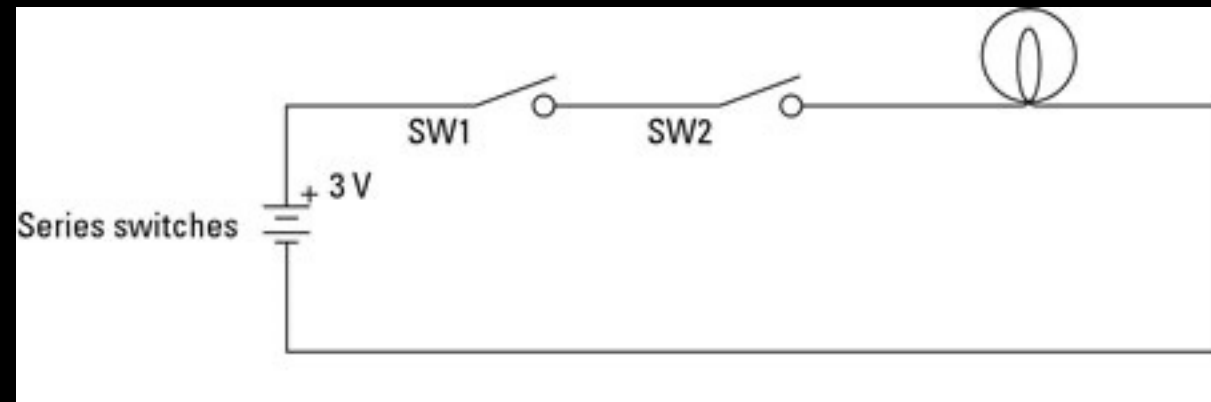
- Take Boolean operands and evaluate to Boolean values

expr1	expr2	expr1 and expr2	expr1 or expr2	not expr1
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

The **and** Operator

- Binary operator
- The expression **left and right** produces:
 - **True** if **both** **left** **and** **right** are **True**
 - **False** otherwise

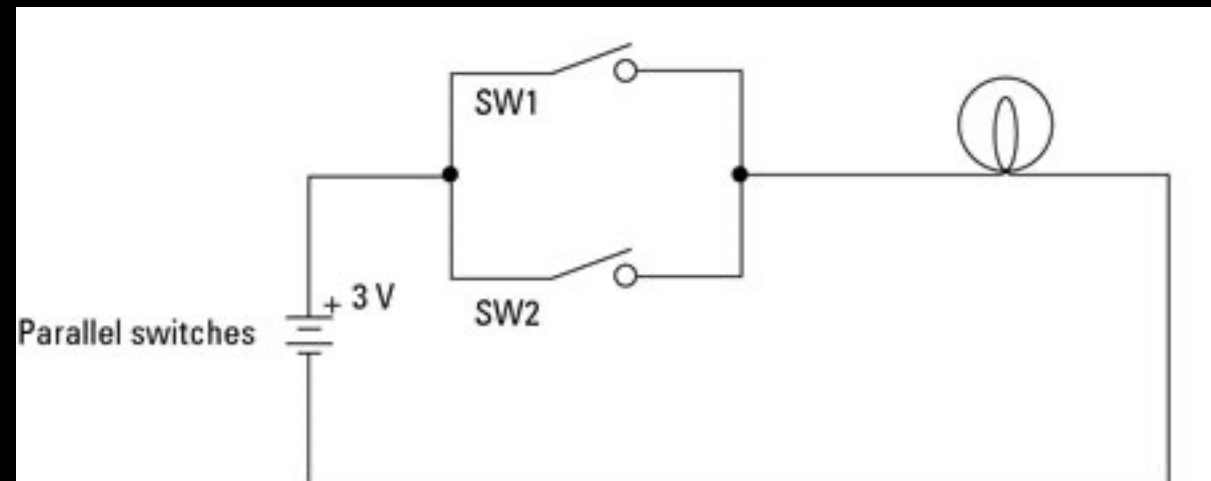
Switch 1 AND Switch 2 must be on for light to turn on



The **or** Operator

- Binary operator
- The expression **left or right** produces:
 - **True** if **either** left **or** right are **True**
 - **False** only if both are **False**

Switch 1 OR Switch 2 must be on for light to turn on



The **not** Operator

- **not** Binary operator (see what I did there?)
- Results in a Boolean value which is the opposite of the operand value
- An expression involving **not** produces:
 - **True** if the original value is **False**
 - **False** if the original value is **True**

BOO-lean operators

Ghost
NOT
Scream



Scream
NOT
Ghost



Ghost
AND
Scream



Scream
OR
Ghost



Order of Precedence

- We can override precedence with brackets
- In general, brackets should be added to make things easier to read and understand

Operator	Precedence
not	highest
and	
or	lowest

All the Operators!

1. Arithmetic (+, -, /, etc.)
2. Relational (<, ==, etc.)
3. Logical/Boolean (not, and, or)

- Precedence when combining

- Arithmetic operators have higher precedence than relational operators
- Relational operators have higher precedence than Logical/Boolean operators
- All relational operators have the same precedence (i.e. read left to right)



Coding Time!

- Let's go experiment with some of what we just saw
 - Logical operators (and, or, not)
 - Order of precedence

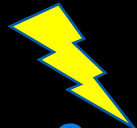
**Open your
notebook**

Click Link:
2. Logical Operators

Binary Operators

- Rules for evaluation

1. Evaluate the left operand (i.e. expression) to a value and replace that operand expression with that value

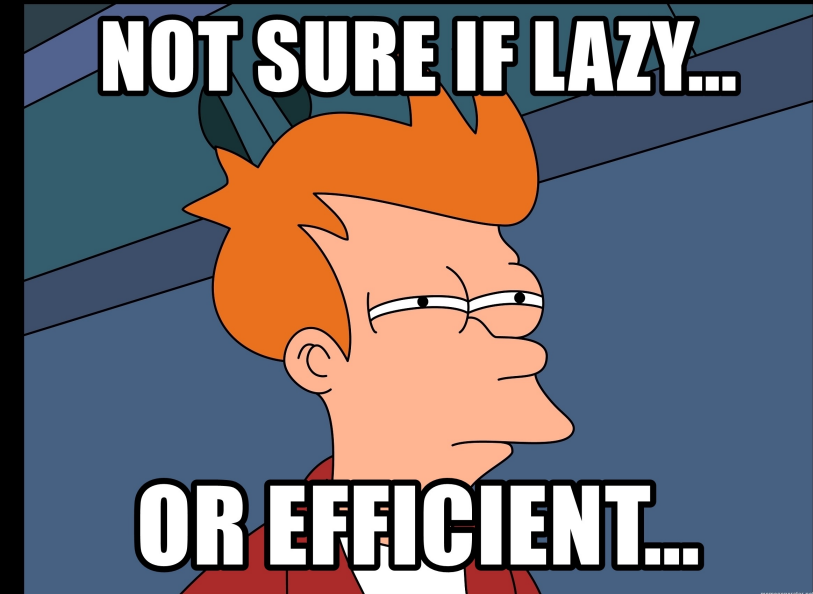


POSSIBLE SHORT CIRCUIT

2. Evaluate the right operand (i.e. expression) to a value and replace that operand expression with that value
3. Apply the operator to the two resultant values

Short-Circuit (Lazy) Evaluation

- **or** evaluates to **True** iff at least one operand is **True**
 - **operand_1 or operand_2**
 - If operand_1 is **True**, operand_2 will not be checked!
- **and** evaluates to **False** iff at least one operator is **False**
 - **operand_1 and operand_2**
 - If operand_1 is **False**, operand_2 will not be checked!
- Like how in a Multiple-Choice Question, if you for sure know the answer is A, you can save time not reading B, C, and D!



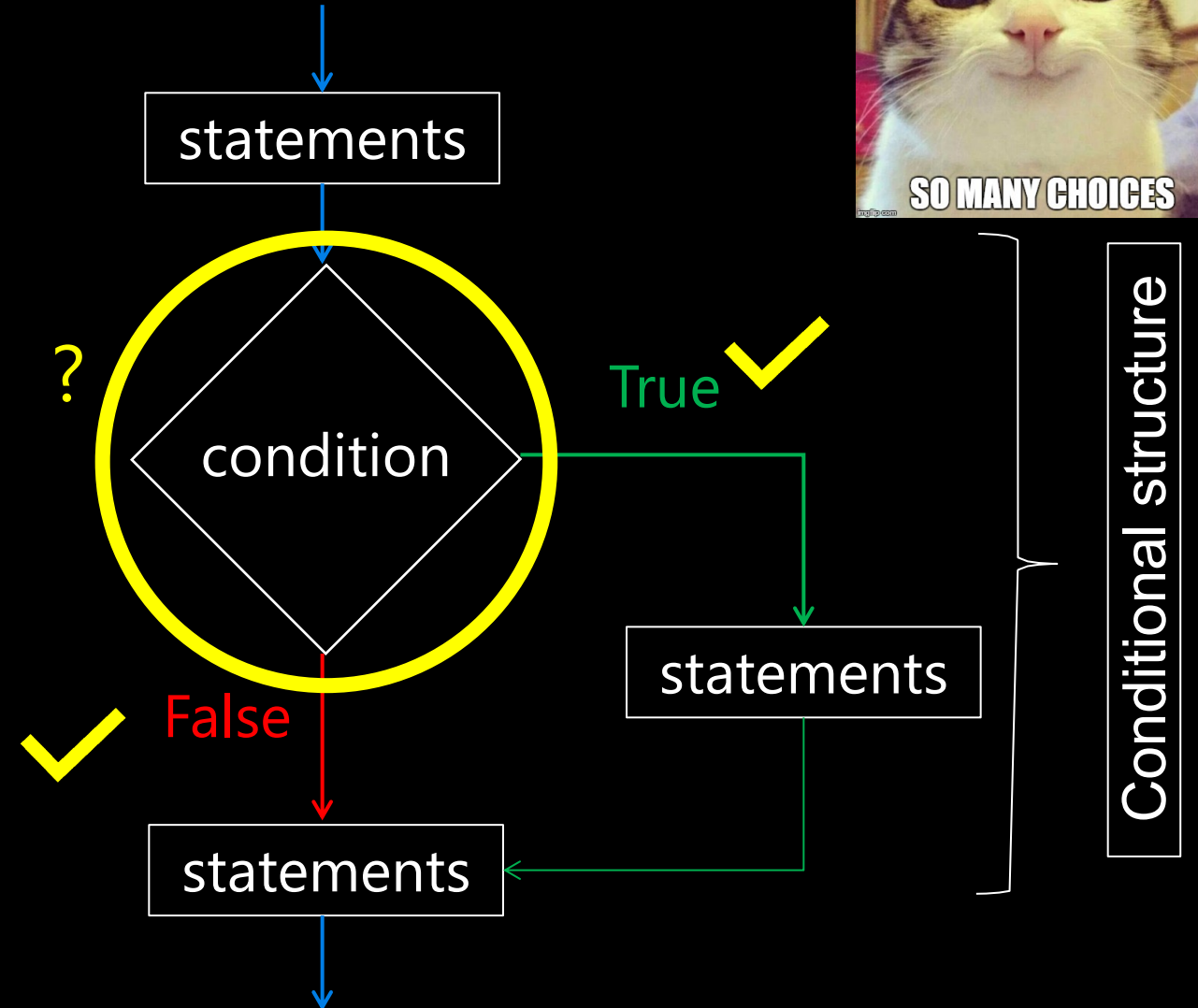
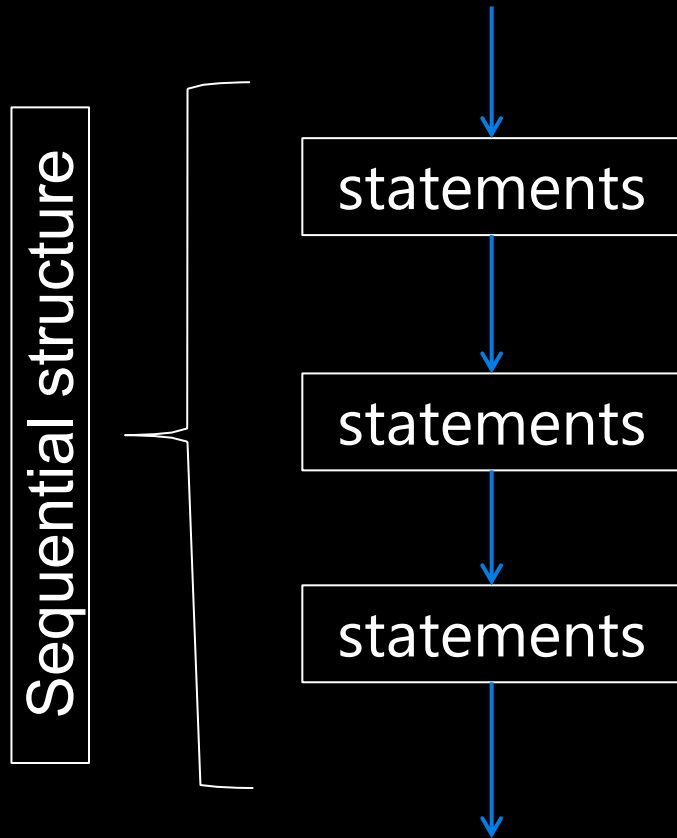
Coding Time!

- Let's go experiment with some of what we just saw
 - Short-circuit (lazy) evaluation

**Open your
notebook**

Click Link:
3. Lazy Evaluation

Making Choices



The if statement

- A general form of an if statement is as follows:

if expression:
→ body



- The “body” only executes if the if statement is True
- if statements are always followed by a colon (:)
 - This is how Python knows you are creating a new block of code
 - Indenting four spaces tells Python what lines of code are in that block

if Statement Example

```
grade = 51
if grade < 50:
    print("You failed APS106...")
if grade >= 50:
    print("Hooray you passed!")
```

```
Hooray you passed!
```

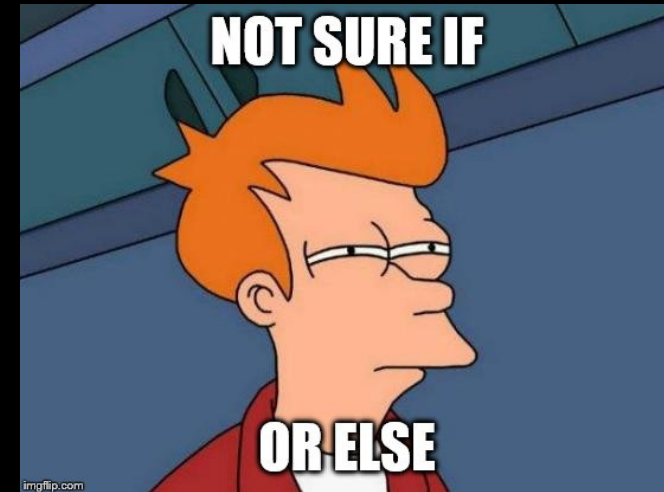
There must be an easier way to
account for all the other cases...

Adding the else statement

- A more general form of the if conditional statement is:

```
if expression:  
    → body1  
else:  
    → body2
```

- ONLY 1 of body1 or body2 will be executed.
 - if statement is True, executes body1
 - if statement is False, executes body2



if-else Statement Example

```
grade = 51
if grade <= 50:
    print("You failed APS106...")
else:
    print("Hooray you passed!")
```

Hooray you passed!

Let's Code!

- Let's go experiment with some of what we just saw
 - if statements
 - Using the else keyword

**Open your
notebook**

Click Link:
**4. Introducing if
statements**

Booleans, Logic, & Conditional “if” Statements.

Week 2 | Lecture 1 (2.1.1)

if nothing else, write `#cleancode`