If Statements

Adapted from “Python Crash Course” By Eric Matthes

If Statements

Goals for today:

- Write conditional tests
- Write simple if statements
- Write more complex ones

Starting simple

```
cities = ["laramie","casper","jackson","cheyenne"]

for city in cities:
    if city == "laramie":
        print(city.upper())
    else:
        print(city.title())
```

[1] ✓ 0.0s

```
LARAMIE
Casper
Jackson
Cheyenne
```

Conditional tests

The basis of every `if` statement

Seeing if something is `True` or `False`

Checking for equality

Done with `==`

A single `=` is a statement to set something

A double `==` compares two things

```
#Explained in the next cell
city = "Laramie"
print(city=="Laramie")
```

[5]

```
True
```

```
city = "Laramie"
print(city=="Cheyenne")
print(city=="laramie")
```

```
False
False
```

Ignoring case

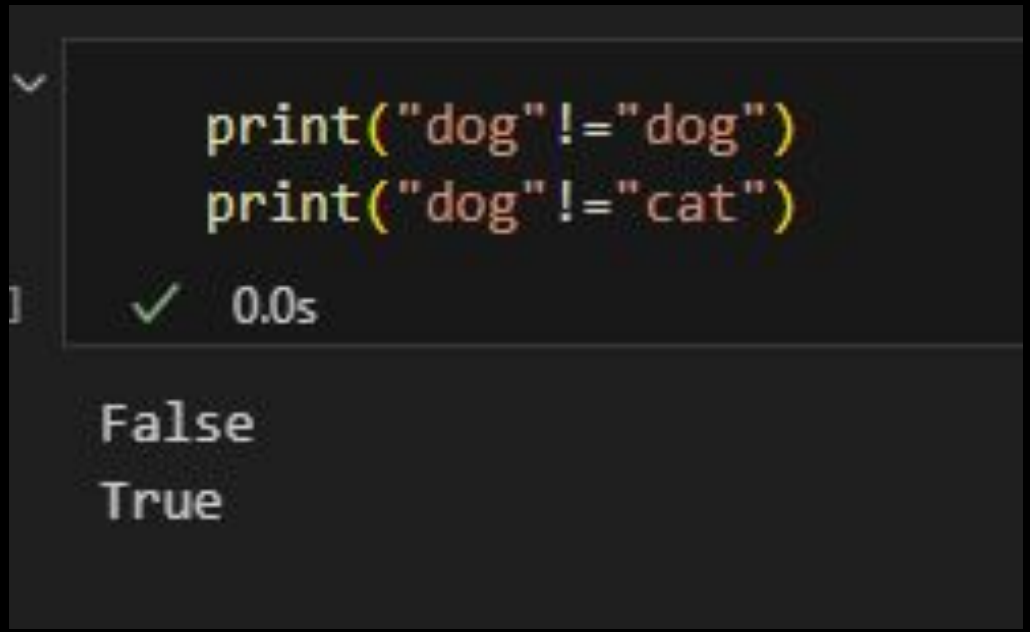
```
city = 'Laramie'
print(city.upper()=='laramie'.upper())
```

[7]

```
True
```

Checking for inequality

Done with `!=`



Numerical comparisons

```
age_to_drive = 16

print(age_to_drive == 16)
print(age_to_drive == 15)
```

```
True
False
```

```
age_to_drive = 16

print(age_to_drive != 16)
print(age_to_drive != 15)
```

```
False
True
```

Numerical comparisons

```
age_to_drive = 16
print(age_to_drive < 16)
print(age_to_drive > 16)
print(age_to_drive <= 16)
print(age_to_drive >= 16)
```

```
False
False
True
True
```

Checking multiple conditions

Multiple conditions can be checked at once

Done with the following keywords

- and
- or

Using `and`

Used to check if two statements are true

Both sides need to evaluate to true

```
age_zero = 22
age_one = 20
print(age_zero >= 21)
print(age_one >= 21)
print(age_zero >= 21 and age_one >= 21)
print((age_zero >= 21) and (age_one >= 21))
```

[4] ✓ 0.0s

```
True
False
False
False
```

Using `or`

Again checks multiple conditions

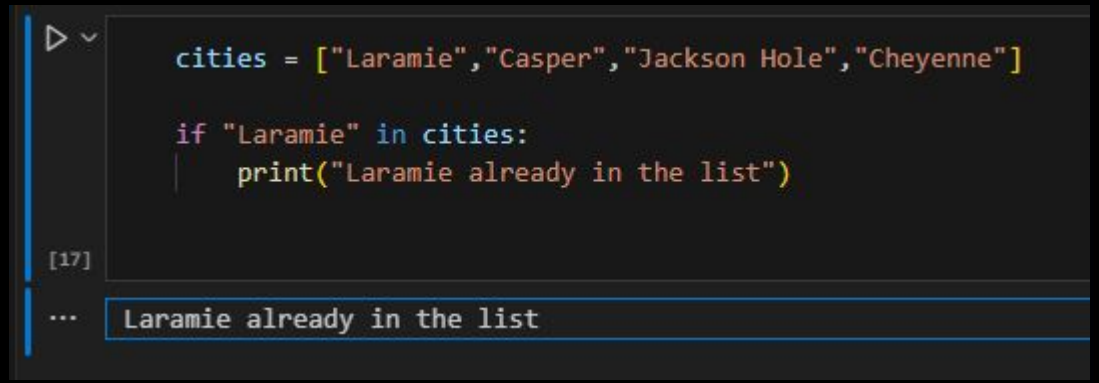
Only one side needs to be true

```
age_zero = 22
age_one = 20
print(age_zero >= 21)
print(age_one >= 21)
print(age_zero >= 21 or age_one >= 21)
```

```
True
False
True
```

Seeing if a value is in a list

Done using the `in` keyword



Seeing if a value is in a string

```
cities = ["Laramie","Casper","Jackson Hole","Cheyenne"]

if " " in cities[0]:
    print(f"There is a space in {cities[0]}")

if " " in cities[2]:
    print(f"There is a space in {cities[2]}")
```

[18]

```
There is a space in Jackson Hole
```

Seeing if something isn’t in a list

Can utilize the `not` keyword

```
banned_phrases = ["Go Rams","Boise is a state"]

if "Go Pokes" not in banned_phrases:
    print("Go Pokes")
```

[19]

```
Go Pokes
```

Boolean expressions

Conditional tests AKA Boolean expressions

Booleans are only one of two values:

- True
- False

Simple `if` statements

```
[20]
if True:
    print('This will print')

if False:
    print('This not will print')

if not True:
    print('This not will print')

if not False:
    print('This will print')
```

```
This will print
This will print
```

```
[21]
age = 19

if age >= 18:
    print("you can vote!")
```

```
you can vote!
```

`if-else` statements

Allows you to make a branched decision

Couples the `else` keyword with `if`

```
age = 17

if age >= 18:
    print("You are old enough to vote!")
else:
    print("You can't yet vote")
```

[22]

```
You can't yet vote
```

The `if-elif-else` chain

Adds in the `elif` keyword

Allows you to have multiple possible outcomes

```
[24]
age = 12
if age < 4:
    price = 2
elif age < 65:
    price = 12
else:
    price = 7

print(f"Your cost of admission is ${price}.00")
```

```
Your cost of admission is $12.00
```

Multiple `elif` blocks

Can have as many `elif` statements as are needed

```
[25]
age = 12
if age < 4:
    price = 2
elif age < 18:
    price = 10
elif age < 65:
    price = 12
else:
    price = 7

print(f"Your cost of admission is ${price}.00")
```

```
Your cost of admission is $10.00
```

Omitting the `else` block

```
age = 12
if age < 4:
    price = 2
elif age < 18:
    price = 10
elif age < 65:
    price = 12
elif age >= 65:
    price = 7

print(f"Your cost of admission is ${price}.00")
```

Testing multiple conditions

```
cities = ["laramie","casper","jackson","cheyenne"]

if 'laramie' in cities:
    print("Laramie is in Wyoming")
if 'cheyenne' in cities:
    print("Cheyenne is in Wyoming")
```

✓ 0.0s

```
Laramie is in Wyoming
Cheyenne is in Wyoming
```

Checking for special items

```
cities = ["Laramie","Casper","Jackson","Cheyenne"]

for city in cities:
    if 'a' in city.lower():
        print(f"{city} has an 'a' in it!")
```

[6] ✓ 0.0s

```
Laramie has an 'a' in it!
Casper has an 'a' in it!
Jackson has an 'a' in it!
```

Checking for special items

```
cities = ["Laramie","Casper","Jackson","Cheyenne"]

for city in cities:
    if city.lower() == 'laramie':
        print(f"{city} is the home of the University of Wyoming")
```

✓ 0.0s

```
Laramie is the home of the University of Wyoming
```

Ensuring a list has items

```
user_home_cities = []

if user_home_cities:
    for city in cities:
        print(city)
else:
    print("No home cities")
```

```
No home cities
```

```
user_home_cities = []

if len(user_home_cities) == 0:
    print("No cities stored")
else:
    print(f"There are {len(user_home_cities)} stored")
```

```
No cities stored
```

Using multiple lists

```
avail_toppings = ['pepperoni', 'extra cheese', 'green peppers', 'bacon']

requested_toppings = ['extra cheese', 'green onions', 'pepperoni']

for req_top in requested_toppings:
    if req_top in avail_toppings:
        print(f"Adding {req_top}")
    else:
        print(f"{req_top}  not available")
```

[6]

```
Adding extra cheese
green onions  not available
Adding pepperoni
```