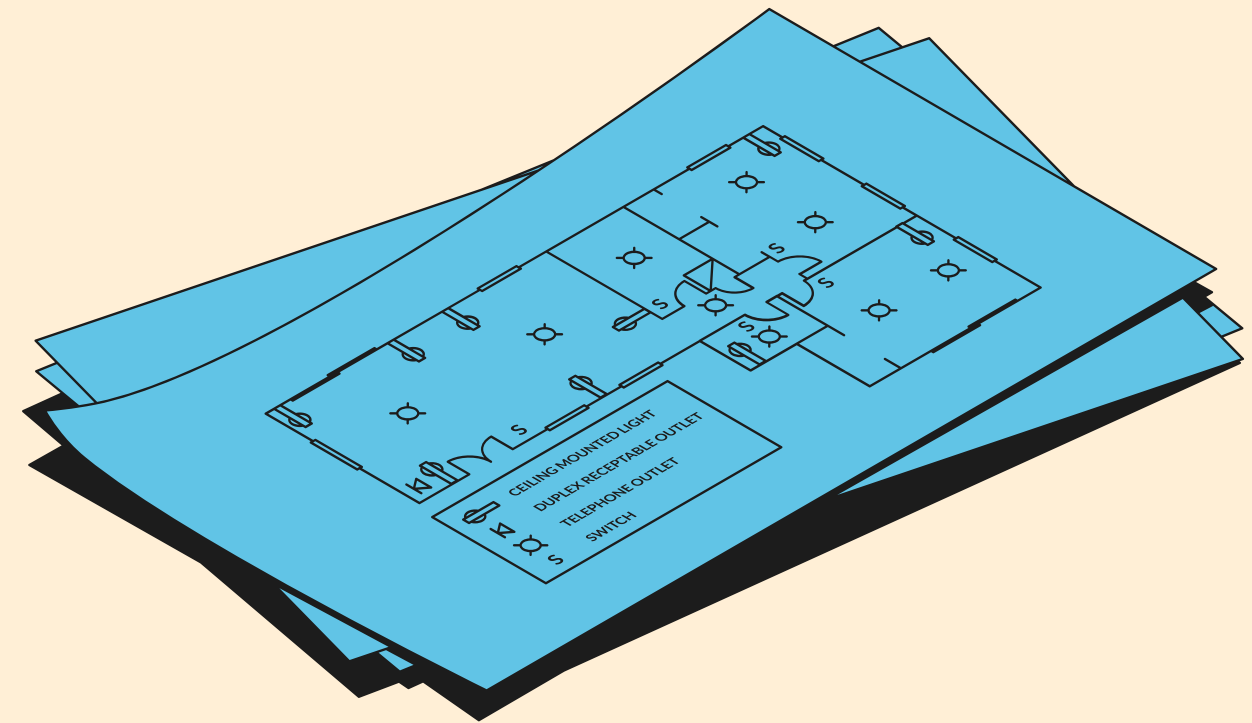# Defining our own Classes

Think of a Class as a Blueprint

A Class:

Is a Blueprint/Template for creating objects

Class

Contains a general overview this type of object

Object

The final Object created from the Class using the description

An Object (instance) of a Class:

instance = <u>C</u>lass()

Object

An Object is a ***Variable equal to a Class***

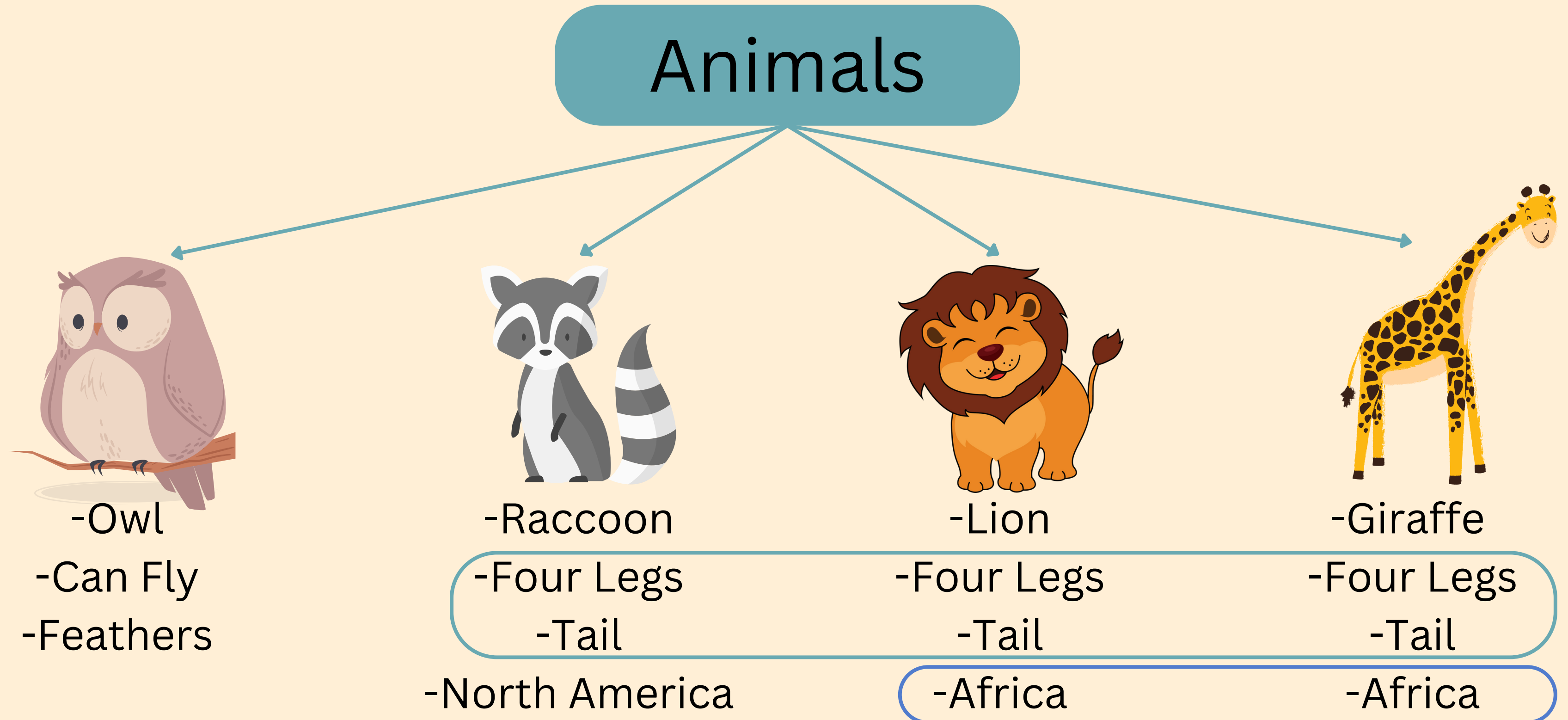A class is **always** Capitalized and followed by parentheses

Remember -> **t1 = Turtle()**

Object

Class

Examples of Objects:

They are **All Animals** sharing some of the same properties. **They all have additional properties and actions of their own**

Animals

-Owl

-Can Fly

-Feathers

-Raccoon

-Four Legs

-Tail

-North America

-Lion

-Four Legs

-Tail

-Africa

-Giraffe

-Four Legs

-Tail

-Africa

Examples of Objects:

They are **all Phones** sharing some of the same properties. **They all have additional properties and actions of their own**

Telephone

-Phone

-Black

-Phone

-Black

-Wireless

-Phone
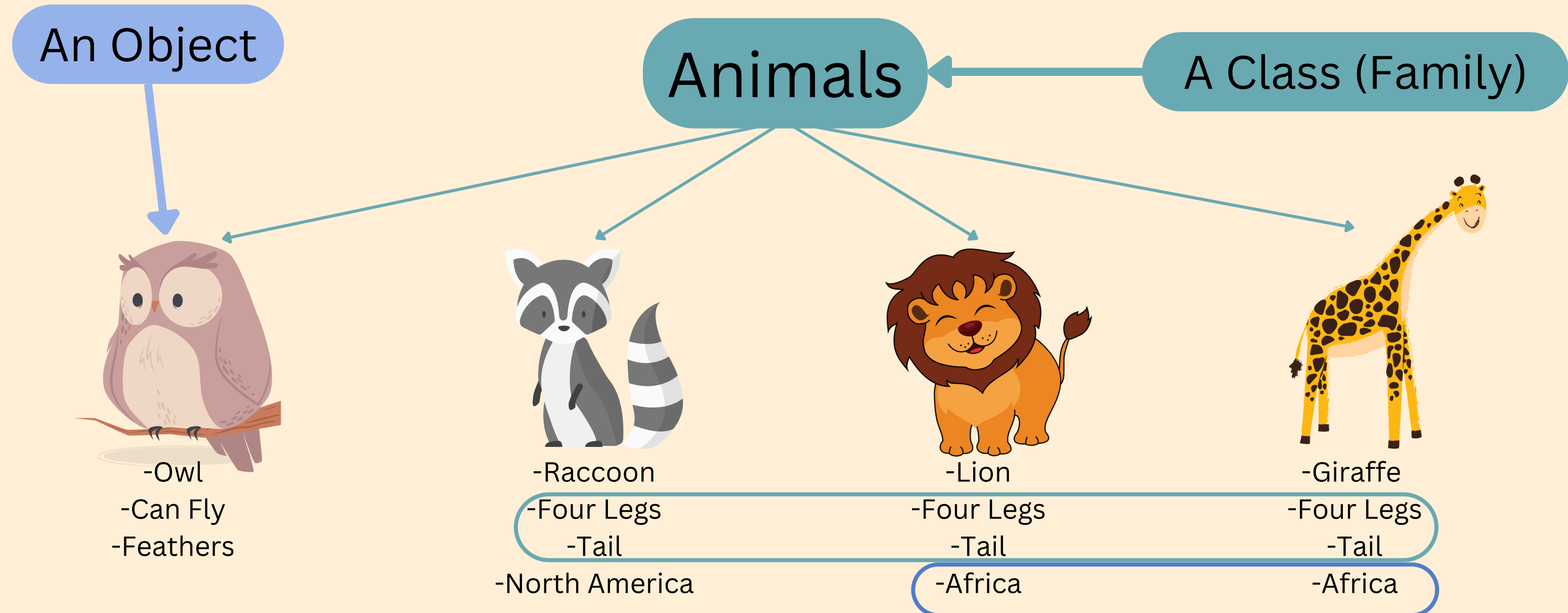
-Black

-Mobile

-Texting

-Phone

-Black

-Mobile

-Smartphone

Creating a Class:

Before creating any Objects, we must first **build our class**.
A Class is where we will create any **Properties (variables)**
and **Methods (functions)** our objects will need

An Object

Animals

A Class (Family)

-Owl
-Can Fly
-Feathers

-Raccoon
-Four Legs
-Tail
-North America

-Lion
-Four Legs
-Tail
-Africa

-Giraffe
-Four Legs
-Tail
-Africa

The Constructor Function:

It's a **method** (function) that is **automatically ran** when an object is created. It **creates an instance** of a specific class

```
Constructor
    │
    ▼
Construction
    │
    ▼
Used to Build
```

The Constructor Function:

It's a **method** (function) that is **automatically ran** when an object is created. It **creates an instance** of a specific class

__init__()

This is the name **we must** use for our constructor function

__init__ -> Initialize -> Start

Constructor

Construction

Used to Build

Creating our Classes:

```python
class My_class():

    def __init__(self, parameter1, parameter2):

        self.property1 = parameter1

        self.property2 = parameter2

    def drive(self):

        x = self.property1 + self.property2

        print("The speed is:" , x)


car = My_class(50 , 60)

car.drive()
```

self -> Is basically **a "key"**. It **unlocks the class** so we can use the properties and methods throughout

Creating a basic class named, My_class.

Here we **create our Constructor method** (**__init__**). This is where we define the Properties our class will need

Here is a **Method named, drive**. This method uses the values from our two properties and gets the sum.

Here we **create an object** named, car. The value of an object is a Class()

Your **Arguments** must match your **parameters**

Calling/Using the drive method we built in the class

Creating our Classes:

```python
class My_class():

    def __init__(self, parameter1, parameter2):
        self.property1 = parameter1
        self.property2 = parameter2

    def drive(self):
        x = self.property1 + self.property2
        print("The speed is:" , x)


car = My_class(50 , 60)
car.drive()
```

Creating a basic class named, My_class.

Here we **create our Constructor method** (_init_). This is where we <u>define the Properties our class will need</u>

Here is a **Method named, drive**. This method <u>uses the values from our two properties and gets the sum</u>.

Here we **create an object** named, car. The <u>value of an object is a Class</u>()

Your **Arguments** <u>must match</u> your **parameters**

Calling/Using the drive method we built in the class

Creating our Classes:

```python
class My_class():

    def __init__(self, parameter1, parameter2):

        self.property1 = parameter1

        self.property2 = parameter2

    def drive(self):

        x = self.property1 + self.property2

        print("The speed is:" , str(x), "km/h")


car = My_class(50 , 60)

car.drive()
```

Output in Terminal
The speed is: 110 km/h

Looking at another Class:

```python
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username
    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage
```

```python
product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

What do we think will have with product_one and product_two above?

What is the name of my class and object(s)

Looking at another Class:

Class

```
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage
```

Objects

```
product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

What do we think will have with product_one and product_two above?

What are the value of my properties?

Looking at another Class:

```python
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage
```

```python
product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

What do we think will have with product_one and product_two above?

Where are the Arguments that'll be given to my Class?

Looking at another Class:

```
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage
```

These are our arguments
(users, storage and
username)

```
product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

<u>What do we think will have with
product_one and product_two
above?</u>

Where are the Arguments that'll be given to my Class?

ClassBreakdown:

```python
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage

product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

Creating a Class called App

A condition within the login function which **checks the properties against a set value**

A **method which increases the current value** of self.storage

Creating an object called **product_one** and calling both methods

Class Breakdown:

```python
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

    def increase_capacity(self, number):

        self.storage += number

        return self.storage


product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)


product_two = App(35, 128, "josh")
product_two.login()
```

Creating a Class called App

Setting up the constructor method with the properties we'll use

A condition within the login function which **checks the properties against a set value**

A **method which increases the current value** of self.storage

Creating an object called **product_one** and calling both methods

Creating an object called product_two and calling the login method

Class Breakdown:

```python
class App():

    def __init__(self, users, storage, username):

        self.users = users

        self.storage = storage

        self.username = username

    def login(self):

        if self.username == "owner" and self.users >= 1:

            print("Welcome,", self.username)

            print("Your storage is:", self.storage)

        else:

            print("Login is denied!")

    def increase_capacity(self, number):

        self.storage += number

        print("Updated storage:", self.storage)

product_one = App(35, 256, "owner")
product_one.login()
product_one.increase_capacity(50)

product_two = App(35, 128, "josh")
product_two.login()
```

product_one Terminal Output

Welcome, owner
Your storage is: 256
Updated storage: 306

product_two Terminal Output

Login is denied!

Flashback pre-object:

A class is a big group of methods that all **contain various elements** of python, such as, **Conditions, Loops, Lists, Dictionaries**, Etc...

Conditional Statements	While Loops	For Loops
if something is true, do this. Other wise do something else.	**While something is true**, keep repeating otherwise stop	**for every item inside something** do something with the current iteration item

Conditional Statements
```python
if name != "done" or name != "quit":
    print("Hello,", name)
elif name == "josh":
    print("It's me Josh")
else:
    print("Goodbye")
```

While Loops
```python
count = int(input("Enter count: "))

while count != 0:
    if count >= 10:
        count -= 2
    else:
        count -= 1
    print("Current count:", count)
    count = int(input("Enter count: "))
```

For Loops
```python
ages = [33, 25, 62, 15, 19, 33]

for age in ages:
    if age <= 17:
        ages.remove(age)
        print("Removed, under 18")
    else:
        print("Person Age:", age)
ages.sort()
print("Ages list:", ages)
```

Challenge #1 in VS code:

DO NOT GO BACK!

Try to recreate the code from previous slide!

Use it as an example to build your own

One Base Class - (App)

Three methods(init, login, increase_capacity)

login -> check if username == "owner"

increase_capacity -> add a number to current storage

Create two objects, **both objects call both methods**