MAKE SCHOOL

Encapsulation & Instantiation

What we're going to learn

MAKE SCHOOL

- What encapsulation is and how it applies to OOP
- The difference between a class and an object
- How to define a class
- How to instantiate an object
- How to apply OOP concepts to model real world objects

MAKE SCHOOL

Encapsulation: what does this word mean to you?

Students, write your response!

What is an Object?

MAKE SCHOOL

An **OBJECT** is a way to organize a collection of *variables* (called **properties**) and *functions* (called **methods**) that act on those variables.

What Do Objects Help Us Do?

MAKE SCHOOL

Encapsulation

- Bundling related data and functionality together
- Providing an interface to access this information while hiding the details

Encapsulation helps us use data and functionality easily without needing to know the inner workings and details

Encapsulation has been helping you this whole time! Here is an example of an object that uses encapsulation principles: **a list!**

Encapsulation

MAKE SCHOOL

Lists have helpful functionality in Python like **`.append()`** that a nice team of developers already wrote for us.

We can use **`.append()`** but we don't need to know the details of the code that makes it work – that's **encapsulation** in action!

We've already been *users* of **objects** and **encapsulation**, and today we are going to learn how to *make objects ourselves!*

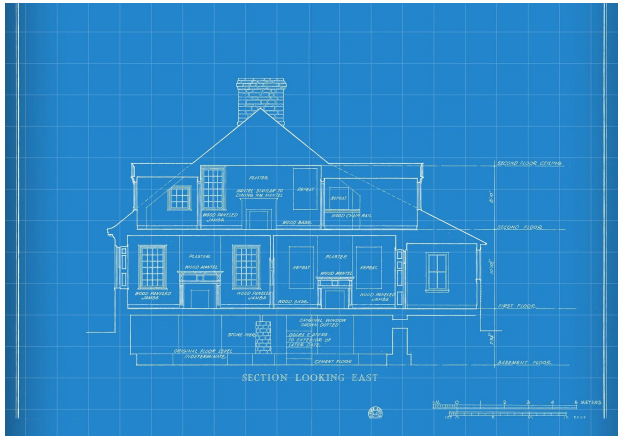
MAKE SCHOOL

What are some other examples of Python objects you have used before?

Students, write your response!

Objects vs Classes

- A **CLASS** is a blueprint to make *objects* (aka instances)
- Classes define the *attributes* (called **properties**) and *behaviors* (called **methods**) that characterize objects.
  - A **property** is a variable connected and unique to its class.
  - A **method** is a function inside a class that can act on an object's data.



Examples of Objects and Classes

MAKE SCHOOL

- Example class: Video (on YouTube)
  - **Properties**: title, author, length, resolution, etc ...
  - **Methods**: play, stop, rewind, double_speed, etc ...
- Example object: RickRoll Video
  - **Properties**: title = “Never Gonna Give You Up - Rick Astley”, length = 312 seconds, etc ...

Draw two more examples of classes and their properties and methods

MAKE SCHOOL

Students, draw anywhere on this slide!

Object Instantiation

MAKE SCHOOL

Classes can be utilized by **instantiating** them into computer memory.

Instantiation of a class creates it as an object in memory and allows users to explicitly manipulate the object to...

- run methods
- alter properties
- perform complex tasks

This is similar to how we assign values to variables!

Pac-Man as a Class

Consider the famous arcade game **Pac-Man**.

Pac-Man can be used to illustrate class hierarchies and conventions.

Let's consider two classes: **`PacMan`** and **`Pellet`**.

Pac-Man must eat **`Pellet`** instances.

`Pellet` instances have a property called **`.is_eaten`**.

In this activity, you will create a **`Person` class** and instantiate an **object** that represents ***you***.

- Create a **`Person`** class
- Define relevant methods
- Instantiate an object
- Set relevant properties

Students, draw anywhere on this slide!

MAKE SCHOOL

Time to write some code!

MAKE SCHOOL

Take a look at this code snippet, what do you think it does?

```
class Person(object):
  def __init__(self):
    pass

  def say_hello():
    # TODO: Save your name as variable `person_name`
    person_name = ""
    print("Hi everyone! My name is {}!".format(person_name))
```

Students, write your response!

MAKE SCHOOL

Create a *Person* Class Code Along!

Students browse: repl.it/@JessDahmen/personclass?lite=true

MAKE SCHOOL

Take a look at this code snippet, what do you think it is doing?

```
John = Person()

# TODO: Call this method to say a basic hello!
John.say_hello()
```

Students, write your response!

MAKE SCHOOL

Create a *Person* Class Code Along!

Students browse: repl.it/@JessDahmen/personclass?lite=true

MAKE SCHOOL

How could we improve the Person class we just wrote?

Students, write your response!

Create Relevant Properties

MAKE SCHOOL

Let's create a **property** to save our name to the class so that we can instantiate different classes with different names!

```
class Person(object):
  def __init__(self, person_name):
    self.name = person_name

  def say_hello():
    print("Hi everyone! My name is {}!".format(self.name))
```

Create Relevant Properties

MAKE SCHOOL

Properties are accessible anywhere in the class, the way we wrote our code before person_name was a local variable inside the scope of the say_hello method and we couldn't use it anywhere else!

```
class Person(object):
  def __init__(self, person_name):
    self.name = person_name

  def say_hello():
    print("Hi everyone! My name is {}!".format(self.name))
```

Instantiating the Class

MAKE SCHOOL

Let's instantiate our class again and this time, we'll send in our name as an argument to the object itself!

Update your code to the following:

```
# TODO: Change the class name and argument to your name
John = Person("John")

John.say_hello()
```

MAKE SCHOOL

Create a *Person* Class Code Along!

Students browse: repl.it/@JessDahmen/personclass?lite=true

MAKE SCHOOL

What method is called when an object is instantiated?

Students choose an option

Red: a method, orange: a property, yellow: a local variable

```
class Pokemon:


    def __init__(self, pokemon_name, pokemon_hp):
        self.name = pokemon_name
        self.hp = pokemon_hp


    def say_name(self):
        greeting = "hello! My name is: "
        print(self.name)
```

MAKE SCHOOL

Why do we define properties?

Students, write your response!

Let’s practice!

Complete the #TODO’s in the given code!

(You can do something other than a sheep, be creative, we will have show and tell!)

MAKE SCHOOL

Show and Tell!

MAKE SCHOOL

Shout Outs

Students, write your response!