Video 16 : Classes & Objects

In this tutorial I'll teach you how object oriented programming works with Python. Object Oriented Programming is the act of modeling real world objects in code.

Real world objects have attributes and capabilities. A person has the attributes of height, weight, name, etc. They also have the capability of being able to talk, walk, eat, etc.

With OOP we store the attributes in variables called fields. We then model capabilities using functions which are called methods.

A class is a blueprint we use to define an objects attributes (fields) and capabilities (methods). Here we will model a Dog object.

CODE

```
# Start by defining the objects name with
class Dog:
    # The init method is called to create an object
    # We give default values for the fields if none
    # are provided
    def __init__(self, name="", height=0, weight=0):
        # self allows an object to refer to itself
        # It is like how you refer to yourself with my
        # We will take the values passed in and assign
        # them to the new Dog objects fields (attributes)
        self.name = name
        self.height = height
        self.weight = weight

    # Define what happens when the Dog is asked to
    # demonstrate its capabilities
    def run(self):
        print("{} the dog runs".format(self.name))
    def eat(self):
        print("{} the dog eats".format(self.name))
    def bark(self):
        print("{} the dog barks".format(self.name))


def main():
    # Create a new Dog object
    spot = Dog("Spot", 66, 26)
    spot.bark()

main()
```

Getters & Setters

Getters and Setters are used to protect our objects from assigning bad fields or for providing improved output. If someone tries to assign "A" to weight block it. Maybe you want to provide lbs or kgs along with weight. With a getter you can do that.

Here is an example where we use both getters and setters while creating a Square object.

CODE

```
class Square:
    def __init__(self, height="0", width="0"):
        self.height = height
        self.width = width

    # This is the getter
    # @property defines that any call to height runs the code in the height method below it
    @property
    def height(self):
        print("Retrieving the height")

        # Put a __ before this private field
        return self.__height

    # This is the setter
    @height.setter
    def height(self, value):

        # We protect the height from receiving a bad value
        if value.isdigit():

            # Put a __ before this private field
            self.__height = value
        else:
            print("Please only enter numbers for height")

    # This is the getter
    @property
    def width(self):
        print("Retrieving the width")
        return self.__width

    # This is the setter
    @width.setter
    def width(self, value):
        if value.isdigit():
            self.__width = value
        else:
            print("Please only enter numbers for width")

    def get_area(self):
        return int(self.__width) * int(self.__height)


def main():
    square = Square()

    height = input("Enter height : ")
```

```
    width = input("Enter width : ")

    square.height = height
  square.width = width

    print("Height :", square.height)
    print("Width :", square.width)

    print("The Area is :", square.get_area())
```

That’s all for now. In the next video I’ll create 2 warrior objects and have them fight to the death!