SECTION 8: OBJECT ORIENTED PROGRAMMING, 1 hour 21 minutes, 9 parts

- **2/9 Object Oriented Programming - Attributes and Class Keyword**
  - **-> using OOP to create objects -> class and attributes**

  - **-> in the .ipynb file**
    - **-> what an object is**
      - she's created a list
      - **mylist. <- then shift tab shows the options**
      - -> type(myset) is a set
      - -> using the class keyword to create a class to create a user defined object
        - **a class is a blueprint which defines the nature of a future object**
        - **-> we have instances of classes**
        - -> class SampleWord <- capitalised names for classes

    - **-> example creating a class**
      - class Sample():
        - pass
      - my_sample = Sample()
      - type(my_sample) <- this is a class

    - **-> another example**
      - class Sample():
        - def __init__(self, breed): <- self connects the method to the instance of the class
          - **self.breed = breed <- __init is to initialise the class**
      - -> then she's created an instance of the class, via Dog(breed = 'lab') e.g
      - **-> __init__(self, breed) <- init is to initialise the class, self represents the instance of the object**
        - **-> self refers to that instance of the class**
        - -> breed is an argument which gets passed in
      - -> setting attributes under __init__ -> defining these is like defining methods
      - -> self.my_attribute = breed, e.g
        - -> pass in the parameter or argument then it gets assigned

    - ***-> so the class is defined***
      - ***-> indented under that is an instance of the class defined***
        - ***-> def __init__(self, breed,...,...):***
          - ***-> then indented under this is self.breed, self.name, self.spots etc***

    - **-> then she is making an example of the dog class**
      - you can check the values of the parts of that instance of the class -> via the methods which were defined
        - -> **when you define classes, you need to make sure that they have certain attributes**
        - -> define the name of the class, then use the __init__ method which acts as a constructor
          - **self acts as a reference to the instance of the class**