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

- **3/9 Object Oriented Programming - Class Object Attributes and Methods**
  - -> OOP
  - **-> class object attributes (the same for any instance of the class)**
  - -> then methods -> self.breed = breed

  - **-> we can define attributes at class object level**
    - -> defining attributes abuse __init__ and below class
    - -> these are the same for any instance of a class
    - -> self.breed e.g <- these are for that instance of the class
    - **-> and then abuse the __init__ you can use a more global approach (species = 'mammal' e.g) -> these apply to all instances of the class**
    - **-> shift tab -> this is for the options**
    - -> my_dog.species -> mammal

  - **-> you can see the documentation for your class when you enter shift tab when calling the function**
  - -> breed / name / spots
  - -> the init method

  - **methods**
    - -> functions defined in the body of the class
    - -> these can use attributes defined in the class definition
    - **-> they look like functions defined in a class -> <u>a method is a function which is in a class</u>**
    - -> she defines an example method (bark, which prints woof) -> **the argument of the method when it is defined in this case is self**
  - -> she then creates an example

  - **-> attributes vs methods**
    - **attributes are my_dog.Species <- e.g -> an attribute is a piece of information about an instance of the class**
    - **-> then a method (aka a function for a class) has brackets**
      - **-> you "execute" a method**

  - **-> methods use information about the attribute**
    - **-> in the definition of the method in this case, she's written self.name (rather than using 'name' as a variable, she's used self.name in the definition of the function**
    - -> slef.name e.g and passing it into 'bark'

  - **-> methods can take arguments which aren't defined in the class**
    - e.g 'number'
    - self.number -> passing in a number call
    - -> running everything to make the changes
    - -> my_dog.species
    - -> self.number
    - **-> passing in the self keyword as the argument of the method in its definition, and slef.method**

  - **-> new class example**
    - class Circle():

- #class object attribute (something which is true in all instances of the class, constants)
- pi = 3.14
- def __init__(self, radius=1): <- initialise a circle at radius 1
    - self.radius = radius <- the attribute radius is the same as in the __init__(self argument
- then she defines different methods (functions) for the circle class
- def get_circumference(self):
    - return self.radius*self.pi*2 <- when defining functions for the class (methods), she's used self.radius e.g

- my_circle = Circle()

- **my_circle. <- then enters shift tab, which shows the different methods for the circle class**
    - -> my_circle.pi <- this returns the class object attribute pi (3.14)
- -> my_circle.radius -> this overwrites the default value
- -> my_circle.get_circumference() <- this returns the circumference
    - -> methods

- **-> when working with OOP**
    - an attribute doesn't have to be defined from a particular parameter call
    - -> e.g self.area = (radius**2)* self.pi
    - -> then my_circle.area returns the area of the circle
    - **-> you can use self.pi or self.area = radius*radius*Circle.pi <- in other words, Circle.pi is a class object attribute -> this is referenced at the top of the definition of the class, and can be used repeatedly in the code**