In [25]:

```
class Dog():

    #Class Object Attribute
    #Same for any instance or object of a class
    #So self is not required
    species = 'mammal'

    def __init__(self, breed, name):

        self.breed = breed
        self.name = name

    # Operations/Actions ---> Methods

    def bark(self,number):
#         print("WOOF! My name is {} and the number is {}"
#                         .format(self.name,number))

        self.number = number

    def display(self):
         print("WOOF! My name is {} and the number is {}"
                    .format(self.name,self.number))

    def sample(self,number):
        self.number=number
```

In [26]:

```
my_dog = Dog('Lab', 'Frankie')
```

In [27]:

```
my_dog.bark(20)
```

In [28]:

```
my_dog.display()
```

```
WOOF! My name is Frankie and the number is 20
```

Creating Userdefined Method ::

In [29]:

```
class Circle():

    #class object attribute
    pi = 3.14

    def __init__(self,radius=1): #radius = 1 is a default variable

        self.radius = radius
        self.area = radius*radius*self.pi #we can write Circle.pi as its
                                          #class variable

    #Method
    def get_circumference(self):
        return self.radius * self.pi*2 # we can write "Circle.pie*2"
```

In [30]:

```
my_circle = Circle()
```

In [31]:

```
my_circle.pi
```

Out[31]:

```
3.14
```

In [32]:

```
my_circle.radius #default redius value
```

Out[32]:

```
1
```

In [33]:

```
my_circle = Circle(30) # overide the radius value
```

In [34]:

```
my_circle.radius
```

Out[34]:

```
30
```

In [35]:

```
my_circle.get_circumference()
```

Out[35]:

```
188.4
```

In [36]:

```
my_circle.area
```

Out[36]:

```
2826.0
```