In [5]:

```
#Prog 1
#Class Attribute
class Dog:
    species = 'mammal'
    #Initializer / Instance Attributes
    def __init__(self,name,age):
        self.name = name
        self.age = age
    #instance method
    def description(self):
        return "{} is {} years old".format(self.name,self.age)
    #instance method
    def speak(self,sound):
        return "{} says {}".format(self.name,sound)
#Instantiat the Dog object
pike = Dog("Pike", 6)

#Call our instance methods
print(pike.description())
print(pike.speak("Woof Woof!"))
```

```
Pike is 6 years old
Pike says Woof Woof!
```

In [6]:

```
#Prog 2
#Parent class
class Dog:
    #class attribute
    species='mammal'

    #Initializer/Instance attributes
    def __init__(self,name,age):
        self.name = name
        self.age = age

    #instance method
    def description(self):
        return "{} is {} years old".format(self.name,self.age)

    #instance method
    def speak(self,sound):
        return "{} says {}".format(self.name,sound)

#Child class (inherits from Dog class)
class RussellTerrier(Dog):
    def run(self,speed):
        return "{} runs {}".format(self.name,speed)
#Child class(inherits from Dog class)
class Bulldog(Dog):
    def run(self,speed):
        return "{} runs {}".format(self.name,speed)

#Child classes inherit attributes and behaviors from the parent class
thunder = Bulldog("Thunder",9)
print(thunder.description())

#Child classes have specific attributes and behaviors as well
print(thunder.run("Slowly"))

spinter = Bulldog("Spinter",12)
print(spinter.description())
print(spinter.run("Fast"))

roger = RussellTerrier("Roger",5)
print(roger.description())
print(roger.run("quickly"))
```

```
Thunder is 9 years old
Thunder runs Slowly
Spinter is 12 years old
Spinter runs Fast
Roger is 5 years old
Roger runs quickly
```

In [4]:
```
# Prog 3(with error)
# Parent class
class Dog:
    # Class attribute
    species = 'mammal'
    # Initializer / Instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
    # instance method
    def description(self):
        return "{} is {} years old".format(self.name, self.age)
    # instance method
    def speak(self, sound):
        return "{} says {}".format(self.name, sound)
# Child class (inherits from Dog() class)
class RussellTerrier(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)
# Child class (inherits from Dog() class)
class Bulldog(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)
# Child classes inherit attributes and behaviors from the parent class
thunder = Bulldog("Thunder", 9)
print(thunder.description())
# Child classes have specific attributes and behaviors as well
print(thunder.run("slowly"))
# Is thunder an instance of Dog()?
print(isinstance(thunder, Dog))
# Is thunder_kid an instance of Dog()?
thunder_kid = Dog("ThunderKid", 2)
print(isinstance(thunder, Dog))
# Is Kate an instance of Bulldog()
Kate = RussellTerrier("Kate", 4)
print(isinstance(Kate, Dog))
# Is thunder_kid and instance of kate?
print(isinstance(thunder_kid, Kate))
print("Thanks for understanding the concept of OOPs")
```

```
Thunder is 9 years old
Thunder runs slowly
True
True
True

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-9d04014cdcd2> in <module>
     36 print(isinstance(Kate, Dog))
     37 # Is thunder_kid and instance of kate?
---> 38 print(isinstance(thunder_kid, Kate))
     39 print("Thanks for understanding the concept of OOPs")

TypeError: isinstance() arg 2 must be a type or tuple of types
```

In [1]:

```python
# Prog 3(without error)
# Parent class
class Dog:
    # Class attribute
    species = 'mammal'
    # Initializer / Instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
    # instance method
    def description(self):
        return "{} is {} years old".format(self.name, self.age)
    # instance method
    def speak(self, sound):
        return "{} says {}".format(self.name, sound)

# Child class (inherits from Dog() class)
class RussellTerrier(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)
# Child class (inherits from Dog() class)
class Bulldog(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)

# Child classes inherit attributes and behaviors from the parent class
thunder = Bulldog("Thunder", 9)
print(thunder.description())

# Child classes have specific attributes and behaviors as well
print(thunder.run("slowly"))

# Is thunder an instance of Dog()?
print(isinstance(thunder, Dog))

# Is thunder_kid an instance of Dog()?
thunder_kid = Dog("ThunderKid", 2)
print(isinstance(thunder, Dog))

# Is Kate an instance of Bulldog()
Kate = RussellTerrier("Kate", 4)
print(isinstance(Kate, Dog))

print("Thanks for understanding the concept of OOPs")
```

```
Thunder is 9 years old
Thunder runs slowly
True
True
True
Thanks for understanding the concept of OOPs
```

In [ ]:

```python

```