Polymorphism

We've learned that while functions can take in different arguments, methods belong to the objects they act on. In Python, *polymorphism* refers to the way in which different object classes can share the same method name, and those methods can be called from the same place even though a variety of different objects might be passed in. The best way to explain this is by example:

In [1]:

```
class Dog:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return self.name+' says Woof!'

class Cat:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return self.name+' says Meow!'

#instances of dog and cat class
Gruto = Dog('Gruto')
snowbell = Cat('snowbell')

#now call the speak method for both instances
print(Gruto.speak())
print(snowbell.speak())
```

```
Gruto says Woof!
snowbell says Meow!
```

In the above example we have a Dog class and a Cat class, and each has a `.speak()` method. When called, each object's `.speak()` method returns a result unique to the object.

There a few different ways to demonstrate polymorphism. First, with a for loop:

In [2]:

```
for pet in [Gruto,snowbell]:
    print(pet.speak())
```

```
Gruto says Woof!
snowbell says Meow!
```

Another is with functions:

In [3]:

```
def pet_speak(pet):
    print(pet.speak())

pet_speak(Gruto)
pet_speak(snowbell)
```

```
Gruto says Woof!
snowbell says Meow!
```

In [ ]:

```

```