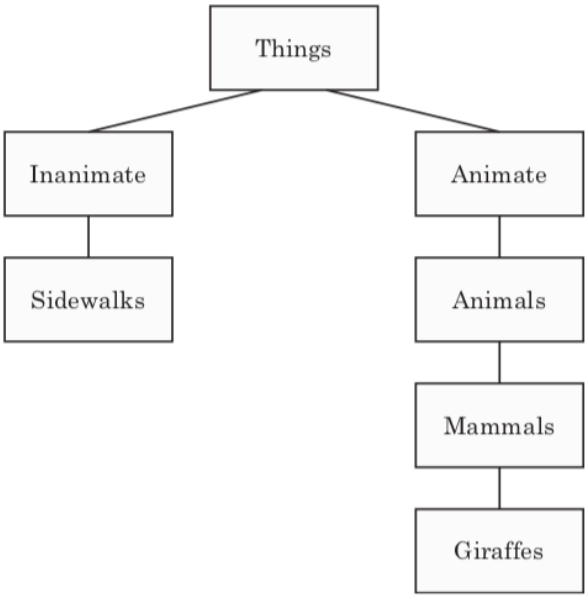
# Chapter 8: How to Use Classes and Objects

Wednesday, March 7, 2018 11:35 AM

1. Breaking Things into Classes

   ***Objects*** are a way of organizing code in a program and breaking things down to make it easier to thin about complex ideas.

   ***Classes*** are a way to classify objects into groups




```
>>> class Things:
    pass
```

   a. Children and parents
      I. A **child** class is a class that is part of another class
      II. A **parent** class is not contained as part of another class

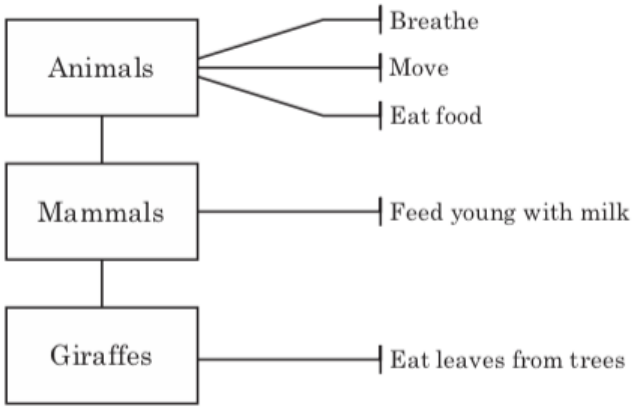
```
>>> class Inanimate(Things):
    pass
>>> class Animate(Things):
    pass
>>> class Sidewalks(Inanimate):
    pass
>>> class Animals(Animate):
    pass
```

```
>>> class Mammals(Animals):
        pass
>>> class Giraffes(Mammals):
        pass
```

b. Adding objects to classes
   I. An **object** is instance of a class
   ```
   >>> reginald = Giraffes()
   ```
   II. We call Reginald an ***object*** of the class Giraffes (you may also see the term ***instance*** of the class). T

c. Defining functions of classes
   I. A **method** is a function within a class
   ```
   >>> class ThisIsMySillyClass:
         def this_is_a_class_function():
            print('I am a class function')
         def this_is_also_a_class_function():
            print('I am also a class function. See?')
   ```

d. Adding Class Characteristics as functions
   I. A **characteristic** is a trait that all of the members of the class(and its children) share
   II. The characteristics can be thought of as **actions**, or **functions** that an object of that class can do




```
>>> class Animals(Animate):
      def breathe(self):
         pass
      def move(self):
         pass
      def eat_food(self):
```

```
    def eat_food(self):
        pass
```

e. Why use classes and objects?
   I. You can create multiple instances of the same class
   II. These instances can perform the same function (methods) provided by its class and its parent classes.

```
>>> reginald = Giraffes()
>>> reginald.move()
>>> reginald.eat_leaves_from_trees()
```

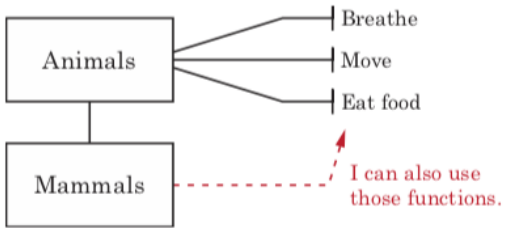
f. Objects and Classes in Pictures
   I. The turtle module allows us to create objects out of a class (ex: pen).

```
>>> import turtle
>>> avery = turtle.Pen()
>>> kate = turtle.Pen()
```

   II. We can create multiple turtle objects

```
>>> import turtle
>>> avery = turtle.Pen()
>>> kate = turtle.Pen()
```

2. Other Useful features of objects and classes
   a. Inherited functions
      I. The Giraffes class ***inherits*** functions from the Mammals class, which, in turn, inherits from the Animals class.




      II. Functions defined in any parent class are available to its child classes

```
>>> reginald = Giraffes()
>>> reginald.breathe()
breathing
```

```
breathing
>>> reginald.eat_food()
eating food
>>> reginald.feed_young_with_milk()
feeding young
```

b. Functions calling other functions
   I. The self parameter is a way for one function in the class to call another function.

```
>>> class Giraffes(Mammals):
        def find_food(self):
            self.move()
            print("I've found food!")
            self.eat_food()
        def eat_leaves_from_trees(self):
            self.eat_food()
        def dance_a_jig(self):
            self.move()
            self.move()
            self.move()
            self.move()
```

c. Initializing an object
   I. When we initialize an object, we are getting it ready to be used.
   II. To do this, we create an **__Init__** function (notice that there are two underscore characters on each side, for a total of four)
   III. The ***init*** function is a way to set the properties for an object when the object is first created, and Python will automatically call this function when we create a new object.

```
>>> class Giraffes:
        def __init__(self, spots):
            self.giraffe_spots = spots
```