**Python for Everyone: Notes**

**Chapter 14: Objects**

* Input -> process -> output
* Object Oriented
  + A program is made of many cooperating objects
  + Instead of being the “whole program”, each object is a little “island” within the program and cooperatively working with other objects
  + A program is made up of one or more objects working together – objects make use of each others capabilities
* Object
  + An object is a bit of self contained code and data
  + A key aspect of the object approach is to break the problem into smaller understandable parts (divide and conquer)
  + Objects have boundaries that allow us to ignore unneeded detail
  + We have been using objects all along: string objects, interger objects, dictionary objecs, list objects
* Definitions
  + Class – a template
  + Method or message – a defined capability of a class
  + Field or attribute – a bit of data in a class
  + Object or instance – a particular instance of a class
* Terminology: class
  + Defines the abstract characteristics of a thing (object), including the thing’s characteristics (its attributes, fields, properties) and the thing’s behaviors (the things it can do, or methods, operations or features). One might say that a class is a blueprint or factory that describes the nature of something. For example, the class Dog would consist of traits shared by all dogs, such as breed and fur color (characteristics), and the ability to bark and sit (behaviors)
* Terminology: instance
  + One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the lassie object is an instance of the dog class. The set of values of the attributes of a particular object is called its state. The object consists of state and the behavior that’s deined in the object’s class
* Terminology: method
  + An object’s abilities. In language, methods are verbs. Lassie, being a dog, has the ability to bark. So bark() is one of lassie’s methods. She may have other methods as well, for example sit() or eat() or walk() or save\_timmy(). Within the program, using a method usually affects only one particular objective: all dogs can bark, but you need only one particular dog to do the barking
* Class example
  + class PartyAnimal: <- template for making the partyanimal object
  + x = 0
  + def party(self) :
  + self.x = self.x + 1
  + print("So far",self.x)
  + an = PartyAnimal() <- construct a partyanimal object and store in an
  + an.party() <- partanimal.party(an)
  + an.party()
  + an.party()
  + Class is a reserved word
  + Each party animal object has a bit of data
* A nerdy way to find capabilities
  + The dir() command lists capabilities
  + Ignoring the ones with underscores
    - These are used by python itself
  + The rest are real operations that the object can perform
  + It is like type() – it tells us something about a variable
* Object lifecycle
  + Objects are created, used, and discarded
  + We have special blocks of code (methods) that get called
    - At the moment of creation (constructor)
    - At the moment of destruction (destructor)
  + Constructors are used a lot
  + Destructors are seldom used
* Constructor
  + The primary purpose of the constructor is to set up some instance variables to have the proper initial values when the object is created
  + In object oriented programming, a constructor in a class is a special block of statements called when an object is created
* Many instances
  + We can create lots of objects – the class is the template of the object
  + We can store each distinct object in the own variable
  + We call this having multiple instances of the same class
  + Each instance has its own copy of the instance variables
* Constructors can have additional parameters. These can be used to set up instance variables for the particular instance of the class
* Inheritance
  + When we make a new class – we can reuse an exisiting class and inherit all the capabilities of an exisiting class and then add our own little bit to make our new class
  + Another form of store and reuse
  + Write once – reuse many times
  + The new class (child) has all the capabilities of the old class (paren) – and then some more
* Terminology: inheritance
  + Subclasses are more specialized versions of a class, which inherit attributes and behaviors from their parent classes, and can introduce their own
* Definitions
  + Class – a template
  + Attribute – a variable within a class
  + Method – a function within a class
  + Object – a particular instance of a class
  + Constructor – code that runs when an object is created
  + Inheritance – the ability to extend a class to make a new class