Object-Oriented Programming Terms

- Class
  - A blueprint for the computer to build a structure that holds related information and can manipulate that information
  - Example – The `Animal` class defines a blueprint for the computer to construct an `Animal` object that has information about the sound an animal makes when it makes noise (instance variable `call`) and a way for it to to make noise (instance method `make_noise()`)

```
class Animal:
    def __init__(self,call):
        self.call = call

    def make_noise(self):
        print self.call
```

- Object instance
  - A single, independent collection of data that has a structure specified by its class. All object instances exist independently of each other. When working with two objects, a change to one will not necessarily affect the other
  - Example:

```
>>> dog = Animal("woof! ")
>>> cat = Animal("meow! ")
>>> print dog.make_noise()
woof!
>>> print cat.make_noise()
meow
```

- Instance variable
  - A piece of data in an object instance that may change. A change to one object's instance variable does not affect a another object's instance variables (unless we tell it to specifically)
  - Example:

```
>>> dog = Animal("woof! ")
>>> cat = Animal("meow ")
>>> print dog.make_noise()
woof!
>>> print cat.make_noise()
meow
>>>
>>> cat.call = "rawr!"
>>> print dog.make_noise()
woof!
```

```
>>> print cat.make_noise()
rawr!
```

- Instance method
  - A short algorithm or function that is called from an instance object and can return information about that object, changes the state of the object, prints information, or any combination of the three.
  - Example – The instance method `make_noise()` prints `self.call`.
- Public [variable|method]
  - An instance variable or instance method that may be changed or called from anywhere in a program (this is what you're most familiar with already)
- Private [variable|method]
  - An instance variable or instance method that may only be changed or called from within the object itself
  - **Note**: to define an instance variable or an instance method as private, add two underscores "_" to the beginning of the name (see `__set_wage()` in the example below; `__set_wage()` may only be called from within one of the other instance methods of `Employee`)
  - Example:

```
class Employee:
    def __init__(self):
        self.wage = 0.0

    def give_raise(self,wage_increase):
        self.__set_wage(wage + wage_increase)

    def __set_wage(self,wage):
        if wage > 0.0:
            self.wage = wage
```