Exercise 4.4 – Designing Your Own Inheritance

For this exercise, we want you to describe a generic superclass and at least three subclasses of that superclass, listing at least two attributes that each class would have. It’s easiest to simply describe a real-world object in this manner. An example of what we’re looking for would be to describe a generic Shoe class and some specific subclasses with attributes that they might have, as shown here:

```python
class Shoe:
    Attributes: self.color, self.brand

class Converse(Shoe):  # Inherits from Shoe
    Attributes: self.lowOrHighTop, self.tongueColor, self.brand = "Converse"

class CombatBoot(Shoe):  # Inherits from Shoe
    Attributes: self.militaryBranch, self.DesertOrJungle

class Sandal(Shoe):  # Inherits from Shoe
    Attributes: self.openOrClosedToe, self.waterproof
```

You can use any real-world object except a shoe for this problem :)}
Exercise 4.5 – More Inheritance

Consider the following code:

class Spell:
    def __init__(self, incantation, name):
        self.name = name
        self.incantation = incantation

    def __str__(self):
        return self.name + ' ' + self.incantation + '
' + self.get_description()

    def get_description(self):
        return 'No description'

    def execute(self):
        print self.incantation

class Accio(Spell):
    def __init__(self):
        Spell.__init__(self, 'Accio', 'Summoning Charm')

class Confundo(Spell):
    def __init__(self):
        Spell.__init__(self, 'Confundo', 'Confundus Charm')

    def get_description(self):
        return 'Causes the victim to become confused and befuddled.'

def study_spell(spell):
    print spell

spell = Accio()
spell.execute()
study_spell(spell)
study_spell(Confundo())
1. What are the parent and child classes here?

2. What does the code print out? (Try figuring it out without running it in Python)

3. Which get_description method is called when `study_spell(Confundo())` is executed? Why?

4. What do we need to do so that `print Accio()` will print the appropriate description (`This charm summons an object to the caster, potentially over a significant distance`)? Write down the code that we need to add and/or change.
Exercise 4.6 – Overriding

Alyssa P. Hacker made the following Python class:

class Address:
    def __init__(self, street, num):
        self.street_name = street
        self.number = num

She now wants to make a subclass of the class Address called CampusAddress that has a new attribute, office_number, that can vary. This subclass will always have the street attribute set to Massachusetts Ave and the num attribute set to 77. She wants to use the class as follows:

```python
>>> Sarina_addr = CampusAddress("32-G904")
>>> Sarina_addr.office_number
'32G-904'
>>> Sarina_addr.street_name
'Massachusetts Ave'
>>> Sarina_addr.number
77
```

Alyssa is stuck and needs your help. Please help her implement the CampusAddress class; look at exercise 4.5, particularly the implementations of the two subclasses Accio and Confundo, if you’re stuck.