**Assignment 4 – Python**

**Exercise – Intro to Object Oriented Programming**

1. What is the difference between a local variable and an object’s attribute?

Local variables are used within a method. An objects attribute are variables that can be used throughout the whole object regardless of the method which it is first defined.

1. What method is called when the object is created?

\_\_init\_\_(self, …)

1. If you have an object instance, obj, and you want to call its do something() method (assuming it has one), how would you do this? (write the line of code you would use)

obj.do\_something()

**Exercise – Working with Objects**

1. Write a class called Address that has two attributes: number and street name. Make sure you have an init method that initializes the object appropriately.

class Address:

def \_\_init\_\_(self, street, num):

self.street\_name = street

self.number = num

2. Consider the following code:

class Clock:

def \_\_init\_\_(self, time):

self.time = time

def print\_time(self):

time = ’6:30’

print self.time

clock = Clock(’5:30’)

clock.print\_time()

1. What does the code print out? If you aren’t sure, create a Python file and run it.

5:30

1. Is that what you expected? Why?

Yes, because what was printed out was the attribute self.time, not the local variable time.

3. Consider the following code:

class Clock:

def \_\_init\_\_(self, time):

self.time = time

def print\_time(self, time):

print time

clock = Clock(’5:30’)

clock.print\_time(’10:30’)

1. What does the code print out? If you aren’t sure, create a Python file and run it.

10:30

1. What does this tell you about giving parameters the same name as object attributes?

It is better if you give parameters, local variables, and attributes different names.

4. Consider the following code:

class Clock:

def \_\_init\_\_(self, time):

self.time = time

def print\_time(self):

print self.time

boston\_clock = Clock(’5:30’)

paris\_clock = boston\_clock

paris\_clock.time = ’10:30’

boston\_clock.print\_time()

1. What does the code print out? If you aren’t sure, create a Python file and run it.

10:30

1. Why does it print what it does? (Are boston clock and paris clock different objects? Why or why not?)

boston\_clock and paris\_clock are two names for the same object. This is called "aliasing."

**Exercise – 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:

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

class Vehicle:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

class van(Vehicle):

def \_\_init\_\_(self, make, model, mileage):

Vehicle.\_\_init\_\_(make, model)

self.mileage = mileage

class airplane(Vehicle):

def \_\_init\_\_(self, make, model, capacity):

Vehicle.\_\_init\_\_(make, model)

self.capacity = capacity

class boat(Vehicle):

def \_\_init\_\_(self, make, model, size):

Vehicle.\_\_init\_\_(make, model)

self.size = size

**Exercise – 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 + ’\n’ + 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?

The parent class is Spell. The child classes are Accio and Confundo.

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

Accio

Summoning Charm Accio

No description

Confundus Charm Confundo

Causes the victim to become confused and befuddled.

1. Which get description method is called when ‘study spell(Confundo())’ is executed? Why?

The one inside the Confundo class because methods in the child class override the parent class if they exist in both classes.

1. 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.

Add this method to the Accio class:

def get\_description(self):

return 'This charm summons an object to the caster, potentially over a significant distance’

And change the Spell.\_\_str\_\_ method to reverse the order of self.name and self.incantation.

**Exercise – Overriding**

class Address:

def \_\_init\_\_(self, street, num):

self.street\_name = street

self.number = num

Now 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. Use the class as follows:

>>> Sarina\_addr = CampusAddress("32-G904")

>>> Sarina\_addr.office\_number

’32G-904’

>>> Sarina\_addr.street\_name

’Massachusetts Ave’

>>> Sarina\_addr.number

77

class Address:

def \_\_init\_\_(self, street, num):

self.street\_name = street

self.number = num

class CampusAddress(Address):

def \_\_init\_\_(self, off\_num):

Address.\_\_init\_\_(self, "Massachusetts Ave", 77)

self.office\_number = off\_num

Sarina\_addr = CampusAddress("32-G904")

print(Sarina\_addr.office\_number)

print(Sarina\_addr.street\_name)

print(Sarina\_addr.number)



