Inheritance and Polymorphism

Ch. 13 in Zybooks

In some of the later examples in the last chapter, sometimes we have multiple classes that share a lot of the same features. If you look at those codes, you'll see that there is a lot of duplication of code. That's inefficient, at best. Maybe there is a better way ...

Example:

Business Class

- name (str)
- address (str)
- hours of operation (str)

Restaurant class

→ all the business stuff
+
→ cuisine type (str)
→ rating (int)

Car Dealership class

→ all the business stuff
→ Car Makes Sold (list)
→ Average Price of Make (dict)

Example 2:

Base class

Items at Grocery Store class

→ name
→ quantity

Produce

→ expiration date (string)

Cereal

→ sugar content % (int)

Meats

→ price per pound (float)
→ weight (float)

Derived Classes

Let's see how this works in Python:

```
class Item:
    def __init__(self):
        self.name = ""
        self.quantity = 0
    def set_name(self, name):
        self.name = name
    def set_quantity(self, quantity):
        self.quantity = quantity
    def display(self):
        print(self.name, self.quantity)
```

derived from Item !!

```
class Produce(Item):
    def __init__(self):
        Item.__init__(self)
        self.expiration = ""
    def set_expiration(self, expiration):
        self.expiration = expiration
```

explicit call to Base constructor class

```
def get_expiration(self)
    return self.expiration

if "__name__" == "__main__":

    item1 = Item()
    item1.set_name('Cereal')
    item1.set_quantity(9)
    item1.display()

    item2 = Produce()
    item2.set_name('Apples')
    item2.set_quantity(40)
    item2.set_expiration("5/5/2024")
    item2.display
    print(f"Expires on {item2.get_expiration()}")
```

I want to really emphasize how the necessity to explicitly call the init method of the base class (or super

class) in the init method of the derived class. This is different than in other languages (C++, Java) where this happens automatically.

There are tons of discussions on Reddit and Stack Exchange about this choice that Python has made. I love Python, but I have a difficult time understanding why it is constructed this way.

In my opinion, it violates the spirit of the inheritance principle.

Note also that we must do this for all derived classes, all the way down a chain:

```
class Base:
    def __init__(self):
```

```
    def --
        name = " "

class  Derived One (Base):
    def --init_(self):
        Base.--init--(self):
        address = " "

class Derived Two (Derived One):
    def --init_ (self):
        Derived One.--init--(self):
        zipCode = ϕ
```
