Object Inheritance

In object-oriented programming, the concept of inheritance allows you to build relationships between objects, grouping together similar concepts and reducing code duplication. Let's create a custom Fruit class with color and flavor attributes:

```
>>> class Fruit:
...     def __init__(self, color, flavor):
...         self.color = color
...         self.flavor = flavor
...
```

We defined a Fruit class with a constructor for color and flavor attributes. Next, we'll define an Apple class along with a new Grape class, both of which we want to inherit properties and behaviors from the Fruit class:

```
>>> class Apple(Fruit):
...     pass
...
>>> class Grape(Fruit):
...     pass
...
```

In Python, we use parentheses in the class declaration to have the class inherit from the Fruit class. So in this example, we're instructing our computer that both the Apple class and Grape class inherit from the Fruit class. This means that they both have the same constructor method which sets the color and flavor attributes. We can now create instances of our Apple and Grape classes:

```
>>> granny_smith = Apple("green", "tart")
>>> carnelian = Grape("purple", "sweet")
>>> print(granny_smith.flavor)
tart
>>> print(carnelian.color)
purple
```

Inheritance allows us to define attributes or methods that are shared by all types of fruit without having to define them in each fruit class individually. We can then also define specific attributes or methods that are only relevant for a specific type of fruit. Let's look at another example, this time with animals:

```
>>> class Animal:
...     sound = ""
...     def __init__(self, name):
...         self.name = name
...     def speak(self):
...         print("{sound} I'm {name}! {sound}".format(
...             name=self.name, sound=self.sound))
...
>>> class Piglet(Animal):
...     sound = "Oink!"
...
>>> class Cow(Animal):
...     sound = "Moooo"
...
```

We defined a parent class, Animal, with two animal types inheriting from that class: Piglet and Cow. The parent Animal class has an attribute to store the sound the animal makes, and the constructor class takes the name that will be assigned to the instance when it's created. There is also the speak method, which will print the name of the animal along with the sound it makes. We defined the Piglet and Cow classes, which inherit from the Animal class, and we set the sound attributes for each animal type. Now, we can create instances of our Piglet and Cow classes and have them speak:

```
>>> hamlet = Piglet("Hamlet")
>>> hamlet.speak()
Oink! I'm Hamlet! Oink!
...
>>> class Cow(Animal):
...     sound = "Moooo"
...
>>> milky = Cow("Milky White")
>>> milky.speak()
Moooo I'm Milky White! Moooo
```

We create instances of both the Piglet and Cow class, and set the names for our instances. Then we call the speak method of each instance, which results in the formatted string being printed; it includes the sound the animal type makes, along with the instance name we assigned.

Mark as completed

Like **Dislike** **Report an issue**