Python, Day 7: Inheritance

Andrew Bydlon

January 13, 2019

Review: Classes and objects

Recall that last time, we found out how to create are own classes and objects within python.

Syntax (Classes)

```
class ClassName:   # adjoining (ParentClass) adds a parent class
    method1
    method2
    etc
```

Syntax (Objects)

```
MyObject = MyClass(InitVar1, InitVar2, . . . , InitVarN)
```

Today we will talk about what this "parent class" means.

Inheritance and Polymorphism

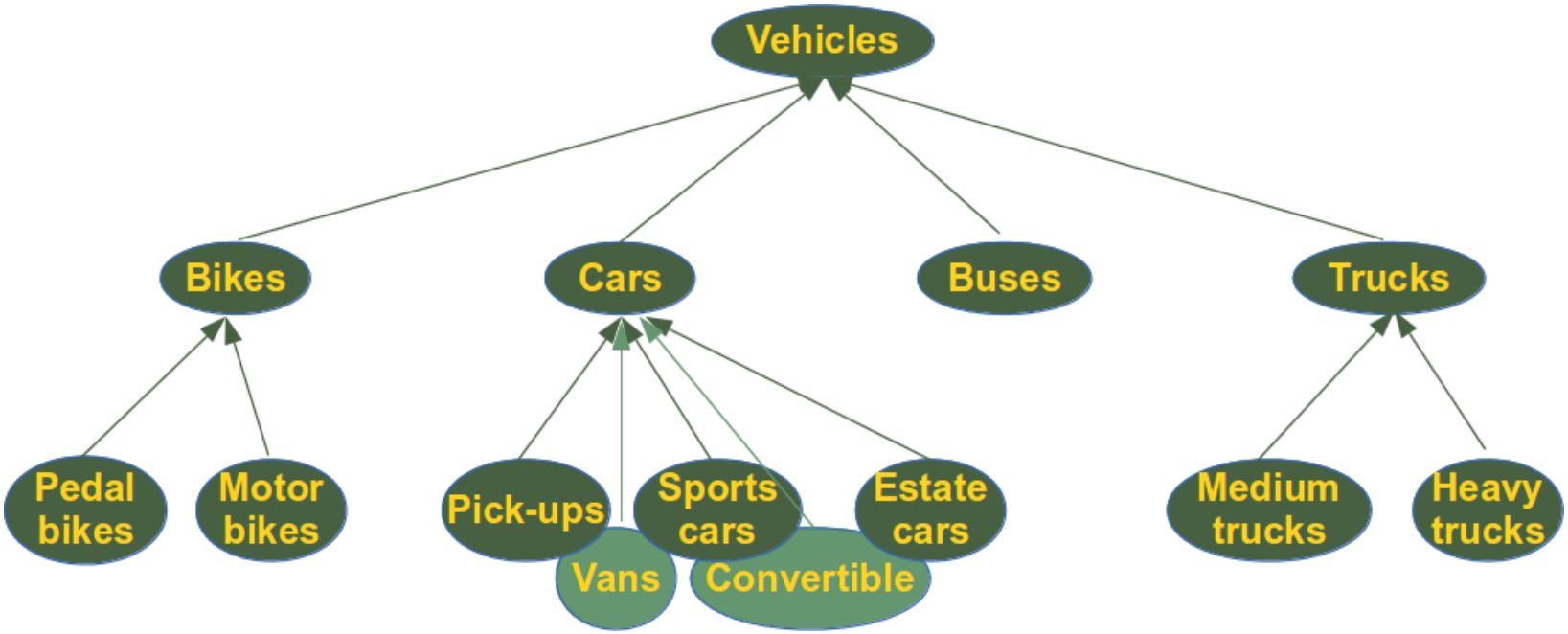
We can now define our own classes and use them to define data. What if two distinct classes share a lot of information?

As an example, what if I am a contractor and want to define classes for buildings?Some examples are as follows:

- House
- Duplex
- Apartment Building
- Office Building
- *The list goes on*

Many properties, such as square-footage, or height, or address, are shared between all of these building types.Therefore, instead of reissuing all of our commands, we can reuse code via **inheritance**.

A quick picture



Note: The picture was taken from `https://www.python-course.eu/python3_inheritance.php`.

Parent Classes

Example (Buildings)

```
class Building:# Define the properties of a Building
    def __init__ (self, height=0.0, squarefootage=0.0, address="")
        self.height = height
        self.sqft = squarefootage
        self.address = address
    def gasheat(self, indoor, outdoor):# Computes the heating cost
        if(indoor<=outdoor):
            return 0
        else:
            return round(self.sqft*(indoor-outdoor)/430,2)
        # A very made up approximation.
    def heightmeters(self, amount):# Height in Meters
        return self.height*.3048
```

Child Classes

Example (Buildings)

```
class House(Building):
    def __init__ (self, occupants, adults):
        self.occupants = occupants
        self.adults = adults
class Apartment(Building):
    def __init__ (self, rooms, available, repairs):
        self.rooms = rooms
        self.available = available
        self.repairs = repairs
class Office(Building):
    def __init__ (self, offices, employees, presidential):
        self.offices = offices
        self.employees = employees
        self.presidential = presidential
```

Use of our child & parent classes

Example (Using the Building hierarchy)

```
MyApartment = Apartment(10,4,2)

print("My apartment building has", MyApartment.available -
MyApartment.repairs, "available currently.")
print("There are", MyApartment.rooms, "rooms in total, of
which",MyApartment.rooms - MyApartment.available, "are occupied.")

MyApartment.height = 30
MyApartment.sqft = 8000
MyApartment.address = "100 New York St, NYC"

print("My apartment building is", MyApartment.height, "feet tall and
has", MyApartment.sqft, "square feet!")
print("It will cost me $" + str(MyApartment.gasheat(68,32)) + " to heat
my building.")
```

And the results!

Example (Running code from previous slide)

My apartment building has 2 available currently.
There are 10 rooms in total, of which 6 are currently occupied.
My apartment building is 30 feet tall and has 8000 square feet!
It will cost me $669.76 to heat my building.

Process finished with exit code 0

The story doesn't stop here

By default, every object has as it's parent class "object". You can for example declare parent class after parent class, after . . .

Example (shapes)

```
class Polygon:
    some def statements

class FourSided(Polygon):
    some def statements

class Rectangle(FourSided):
    some def statements

class Square(Rectangle):
    some def statements
```

Polymorphism

In a literal sense, **Polymorphism** means to take many forms. A function definition is inherited from parent classes, but CAN be changed to accommodate new information.

Example

```
class Rectangle:
    def __init__ (self, side1=1, side2=1):
        self.side1 = side1
        self.side2 = side2

    def Area (self):
        return self.side1*self.side2

    def Perimeter (self):
        return self.side1*2 + self.side2*2
```

Example continued

Example (Example Continued)

```
class Square(Rectangle):
    def __init__ (self, side):
        self.side = side
# Just in case someone wants to view it as a rectangle.
        self.side1 = side
        self.side2 = side

    def Area (self):
        return self.side**2

    def Perimeter (self):
        return self.side*4
```

Assignment 12

Create a class called pets, with parameters name, age, weight, etc (feel free to add more). Then make at least 2 inheritance subclasses, be they dog, cat, rock, etc. Add in new parameters and functions as you see fit, such as time devoted to walking, or cost to feed, or other such things.
If you are feeling ambitious, add a 3rd layer of inheritance, such as Beagle, Persian, or Granite (as above), with its own parameters and functions.
Upload the .py when you finish.