Pratical Learning #7
Inheritance
Working with Multiple Objects

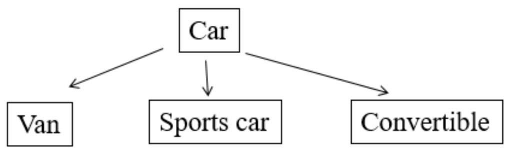
- Object-oriented programming is a programming language model organized around "objects"
- An object is a software bundle of related attributes and behavior (methods)
- A class is a blueprint or prototype from which objects are created

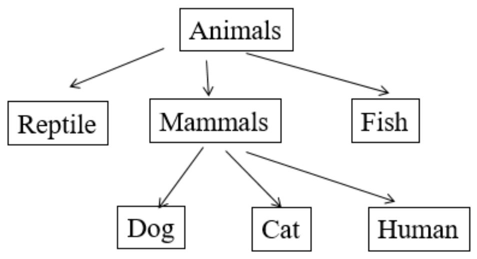
1. Create a new Python file named PL7_LastnameFirstname.
2. Type and study the following code:

```python
class Player(object):
  def __init__(self, name = "Enterprise", fuel = 0):
     self.name = name
     self.fuel = fuel
  def status(self):
     print(self.name,self.fuel)
myship = Player("Appolo")
myship.status()
```

- Object encapsulation & respect privacy
- Inheritance makes objects (classes) special
  - Derive new classes from existing ones
  - Extend the definition of existing classes
  - Override method definitions of existing classes
- Create objects of different classes in the same program
- Allow objects to communicate with each other
- Create more complex objects by combining simpler ones

Inheritance Models "is a" Relationship





Using Inheritance to Create New Classes

- **Inheritance:** An element of OOP that allows a new class to be based on an existing one where the new automatically gets (or inherits) all of the methods and attributes of the existing class
- *The **children** classes get all the capabilities (methods) and properties (attributes) the **parent** class has*; the children classes are also called **derived** classes

- Get the code for free! (code-reuse) – inheritance allows a new class to re-use code which already existed in another class (the parent class)

Derived Classes are New Classes

- To **create specializations** of existing classes or objects by adding new attributes and methods!
  - often called subtyping when applied to classes. In specialization, the new class or object has data or behavior aspects that are not part of the inherited class.
- **Over-ridding** (e.g., over-ridding of the + operator, so + has different meaning, addition of two numbers, concatenation of two strings, etc) – the same method that does something different

Inheritance Example: Animal Class

3. Comment previous code.
4. Type and study the following code:

```
class Animal(object):
    def __init__(self, name):     # Constructor
        self.name = name
    def get_name(self):
        return self.name

class Cat(Animal):
    def talk(self):
        return 'Meow!'

class Dog(Animal):
    def talk(self):
        return 'Woof! Woof!'

animals = [Cat('Missy'), Cat('Mr. Bojangles'),
           Dog('Lassie')]

for animal in animals:
    print (animal.talk() + ' I am ' + animal.get_name())
```

Base class: A class upon which another is based; it is inherited from by a derived class
Derived class: A class that is based upon another class; it inherits from a base class

Altering the Behavior of Inherited Methods: Overriding

- **Override:** To redefine how inherited method of base class works in derived class
- Two choices when overriding
  - Completely new functionality vs. overridden method
  - Incorporate functionality of overridden method, add more

Overriding to Create a New Version

5. Comment previous code.
6. Type and study the following:

```
class Animal(object):
    def __init__(self, name):
        self.name = name

    def talk(self):
        return 'Hello!'

class Cat(Animal):
    def talk(self):
        return 'Meow!'

class fish(Animal):
    pass

yourPet = Animal('Zoro')
print(yourPet.talk())

myPet = Cat('Mingming')
print(myPet.talk())

hisPet = fish('Ariel')
print(hisPet.talk())
```

If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.

Use the pass keyword when you do not want to add any other properties or methods to the class.

Understanding Polymorphism

- **Polymorphism:** Aspect of object-oriented programming that allows you to send same message to objects of different classes, related by inheritance, and achieve different but appropriate results for each object
- When you invoke talk() method of Cat object, you get different result than when you invoke the same method of an Animal (or fish) object