```python
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 10 23:33:01 2015
@author: hina
Reference: https://docs.python.org/3/tutorial/index.html
"""

print ()

# Classes have two attributes - data (class variable) and method

# Notes:
#   - Data attributes may be referenced by methods as well as by ordinary users
#     of an object.
#     In other words, classes are not usable in Python to implement pure abstract
#     data types.
#     In fact, nothing in Python makes it possible to enforce data hiding –
#     it is all based upon convention.
#   - Often, the first argument of a method is called self.
#     This is nothing more than a convention:
#     the name self has absolutely no special meaning to Python.
#     Note, however, that by not following the convention your code may be
#     less readable to other Python programmers.

class Dog:                              # class definition

    """This class defines Dogs"""      # class summary

    code = 123                     # class variable - for data shared by all instances
    kind = 'Canine'                # class variable - for data shared by all instances

    def __init__ (self, name, age=-1): # class instantiation method
        print ("in the instantiation method")
        self.name = name           # instance variable - for data unique to each instance
        self.age = age             # instance variable - for data unique to each instance
        self.tricks = []           # instance variable - for data unique to each instance

    def bark (self):                    # method
        print ("in the bark method")
        if 'bark' in self.tricks:
            print ('...bow-wow!...')
        else:
            print ('...silence...')

    def add_trick (self, trick):         # method
        print ("in the add_trick method")
        self.tricks.append(trick)

fido = Dog('Fido', 5)           # instantiating obj of class Dog named 'Fido', aged 5
fido.add_trick('roll over')
fido.add_trick('play dead')
print(fido.name, fido.code, fido.kind, fido.age, fido.tricks)
fido.bark()
print ()

buddy = Dog('Buddy', 10)        # instantiating obj of class Dog named 'Buddy', aged 10
```

```python
buddy.add_trick('bark')
print(buddy.name, buddy.code, buddy.kind, buddy.age, buddy.tricks)
buddy.bark()
print ()

# Inheritance
#   - Inheritance is used to indicate that one class will get most or all of its
#     features from a parent class.
#   - A class can be drived from a base class within the same module or a base class
#     in a different module
#   - Derived classes may override methods of their base classes.

class Parent (): # base class

    def __init__ (self):
        print ("instantiating base class")

    def Height (self):
        print ("i am tall")

    def HairColor (self):
        print ("i have black hair")

class Child (Parent): # derived class

    def __init__ (self):
        print ("instantiating derived class")

    def HairColor (self): # overriding based class function
        print ("i have blonde hair")

mom = Parent()
mom.Height()
mom.HairColor()

daughter = Child()
daughter.Height()
daughter.HairColor()
print ()

# test

class Cat ():
    pass
tiger = Cat()

class Flower:
    petals = 0
    def __init__(self, n):
        petals = n
rose = Flower(10)
print (rose.petals)

class Phone:
    rings = 0
    def __init__(self, n):
        self.rings = n
myphone = Phone(5)
print (myphone.rings)

print ()
```