class Critter(object):
    name = "Anonymous"

c = Critter()
print c.name
c.name = "George"
print c.name
class Critter(object):
    name = "Anonymous"
    def sayHi(self):
        print "Hi, my name is %s" % self.name

c = Critter()
c.name = "Martha"
c.sayHi()
class Critter(object):
    # name is an instance variable
    name = "Anonymous"

    # constructor is called when new critter is created
    def __init__(self):
        # begin by initializing parent class
        object.__init__(self)

        # constructors usually initialize instance variables
        self.name = "Anonymous"

def main():
    c = Critter()
    print c.name
    c.name = "George"
    print c.name

if __name__ == "__main__":
    main()
class Critter(object):
    def __init__(self, name = "Anonymous"):
        object.__init__(self)
        self.name = name

    def sayHi(self):
        print "Hi, may name is %s!" % self.name

    c = Critter()
    c.sayHi()
    d = Critter("George")
    d.sayHi()
print "In critter.py, namespace is %s" % __name__

class Critter(object):
    def __init__(self):
        object.__init__(self)
        self.name = "Anonymous"

    def sayHi(self):
        print "Hi, my name is %s!" % self.name

def main():
    c = Critter()
    c.name = "George"
    c.sayHi()

main()
""" useCritter
    illustrates default namespace
    requires 'critter.py' to be in same directory
"""

    from critter import *
c = Critter()
#there's only one critter defined here, with a default name
print c.name
#however, you'll see two print statements!

#importing critter.py causes its main() function to run