类和实例

In [1]:
```
class Animal(object):
    pass
```

In [2]:
```
class Animal(object):
    def __init__(self, name):
        self.name = name
```

In [3]:
```
animal = Animal('dog')
animal2 = Animal('dog2')
```

In [4]:
```
print(animal2.name)
```

```
dog2
```

In [5]:
```
class Animal(object):
    def __init__(self, name):
        self.name1 = name
    def greet(self):
        print ('Hello, I am %s.' % self.name1)
```

In [6]:
```
dog1 = Animal('dog1')
```

In [7]:
```
dog1.name1
```

Out[7]:
```
'dog1'
```

In [8]:
```
dog1.greet()
```

```
Hello, I am dog1.
```

访问限制

In [9]:
```
class Animal(object):
    def __init__(self, name):
        self.__name = name
    def greet(self):
        print ('Hello, I am %s.' % self.__name)
```

In [10]:
```
dog1 = Animal('dog1')
```

In [11]:
```
dog1.__name
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/7_/dw3jfv_s2vqby4klxk9qy39w0000gq/T/ipykernel_12024/1402845143.py
in <module>
----> 1 dog1.__name
```

```
AttributeError: 'Animal' object has no attribute '__name'
```

In [12]:
```
dog1.greet()
```

```
Hello, I am dog1.
```

继承

In [13]:
```
class Animal(object):
    def __init__(self, name):
        self.name = name
    def greet(self):
        print ('Hello, I am %s.' % self.name)
```

In [14]:
```
class Dog(object):
    def __init__(self, name):
        self.name = name
    def greet(self):
        print('WangWang.., I am %s.' % self.name)
```

In [15]:
```
dog2 = Dog('dog2')
#print(dog2.name)
dog2.greet()
```

```
WangWang.., I am dog2.
```

In [16]:
```
class Animal(object):
    def __init__(self, name):
        self.name = name
    def greet(self):
        print ('Hello, I am %s.' % self.name)

class Dog(Animal):
    def greet(self):
        print ('WangWang.., I am %s. ' % self.name)
```

In [17]:
```
dog3 = Dog('dog3')
# print(dog3.name)
dog3.greet()
```

```
WangWang.., I am dog3.
```

In [18]:
```
class Dog(Animal):
    def greet(self):
        print ('WangWang.., I am %s. ' % self.name)
    def run(self):
        print ('I am running.I am running')
```

In [19]:
```
dog3 = Dog('dog3')
print(dog3.name)
dog3.greet()
```

```
dog3
WangWang.., I am dog3.
```

In [20]:
```
dog3.run()
```

```
I am running.I am running
```

super 用法

In [21]:
```
class Animal(object):
    def __init__(self, name):
        self.name = name
    def greet(self):
        print ('Hello, I am %s.' % self.name)

class Dog(Animal):
    def greet(self):
        super(Dog, self).greet()
        print ('WangWang...')
```

In [22]:
```
dog = Dog('dog')
```

In [23]:
```
dog.greet()
```

```
Hello, I am dog.
WangWang...
```

In [24]:
```
class Base(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def show(self):
        print('(a,b) is (%d, %d)' % (self.a, self.b))

class A(Base):
    def __init__(self, a, b, c):
        super(A, self).__init__(a, b)  # Python3 可使用 super().__init__(a, b)
#         self.a = a
#         self.b = b
        self.c = c
    def show(self):
        print('(a,b,c) is (%d, %d, %d)' % (self.a, self.b, self.c))
```

In [25]:
```
x = Base(2,3)
```

In [26]:
```
x.show()
```

```
(a,b) is (2, 3)
```

In [27]:
```
y = A(1,3,4)
y.show()
```

```
(a,b,c) is (1, 3, 4)
```

In [28]:
```
import numpy as np
```

In [29]:
```
a = np.random.randn(4,4)
```

In [30]:
```
a
```

Out[30]:
```
array([[-0.55977009, -1.4460655 , -0.8864708 ,  0.21625709],
       [-0.61931999,  2.71434401,  0.51851105,  1.05906195],
       [-0.05626471, -0.19531414, -1.23913716, -3.14423142],
       [-0.55265434, -0.37944761,  0.26505988, -0.05297263]])
```

In [31]:
```
b = a.reshape(2,2,2,2)
```

In [32]:
```
c = b.transpose(0,2,1,3)
```

In [33]:
```
c
```

Out[33]:
```
array([[[[-0.55977009, -1.4460655 ],
         [-0.61931999,  2.71434401]],

        [[-0.8864708 ,  0.21625709],
         [ 0.51851105,  1.05906195]]],


       [[[-0.05626471, -0.19531414],
         [-0.55265434, -0.37944761]],

        [[-1.23913716, -3.14423142],
         [ 0.26505988, -0.05297263]]]])
```

In [34]:
```
d = c.reshape(4,2,2)
```

In [35]:
```
d[0]
```

Out[35]:
```
array([[-0.55977009, -1.4460655 ],
       [-0.61931999,  2.71434401]])
```

In [36]:
```
d[1]
```

Out[36]:
```
array([[-0.8864708 ,  0.21625709],
       [ 0.51851105,  1.05906195]])
```

In [37]:
```
a = np.random.rand(2,2)
```

In [38]:
```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

In [39]:
```
a.shape
```

Out[39]:
```
(3, 3)
```

In [40]:
```
a.squeeze().shape
```

Out[40]:
```
(3, 3)
```

In [41]:
```
b = np.pad(a,(3,3),'symmetric')
```

In [42]:
```
b.shape
```

Out[42]:
```
(9, 9)
```

In [43]:
```
b
```

Out[43]:
```
array([[9, 8, 7, 7, 8, 9, 9, 8, 7],
       [6, 5, 4, 4, 5, 6, 6, 5, 4],
       [3, 2, 1, 1, 2, 3, 3, 2, 1],
       [3, 2, 1, 1, 2, 3, 3, 2, 1],
       [6, 5, 4, 4, 5, 6, 6, 5, 4],
       [9, 8, 7, 7, 8, 9, 9, 8, 7],
       [9, 8, 7, 7, 8, 9, 9, 8, 7],
       [6, 5, 4, 4, 5, 6, 6, 5, 4],
       [3, 2, 1, 1, 2, 3, 3, 2, 1]])
```

In [44]:
```
# !?
np.random.randn(*a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/7_/dw3jfv_s2vqby4klxk9qy39w0000gq/T/ipykernel_12024/522988716.py
 in <module>
      1 # !?
----> 2 np.random.randn(*a)

mtrand.pyx in numpy.random.mtrand.RandomState.randn()

mtrand.pyx in numpy.random.mtrand.RandomState.standard_normal()

_common.pyx in numpy.random._common.cont()

TypeError: only integer scalar arrays can be converted to a scalar index
```

In [45]:
```
a = np.expand_dims(a, 0)
```

In [46]:
```
a.shape
```

Out[46]:
```
(1, 3, 3)
```

In [47]:
```
a
```

Out[47]:
```
array([[[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]])
```

In [48]:
```
a
```

Out[48]:
```
array([[[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]])
```

In [49]:
```
a = np.zeros((1,2,3))
```

In [50]:
```
np.random.randint(5)
```

Out[50]:
```
1
```

In [51]:
```
import torch
```

In [52]:
```
a = torch.randn(3,1)
```

In [53]:
```
a
```

Out[53]:
```
tensor([[ 1.0167],
        [ 0.8786],
        [-0.3223]])
```

In [54]:
```
b = torch.randn(4,1)
```

In [55]:
```
b
```

Out[55]:
```
tensor([[-1.0866],
        [ 1.6895],
        [ 0.7484],
        [-0.3539]])
```

In [56]:
```
X = torch.randn(3,4,requires_grad=True)
```

In [57]:
```
X
```

Out[57]:
```
tensor([[ 0.0394,  0.1012, -0.6656, -0.6785],
        [-0.1844,  1.5155, -0.4016,  0.6330],
        [-1.0402, -0.2858,  0.3793,  0.3413]], requires_grad=True)
```

In [58]:
```
y = torch.mm(torch.mm(a.transpose(1,0), X), b)
```

In [59]:
```
y
```

Out[59]:
```
tensor([[1.5715]], grad_fn=<MmBackward>)
```

In [60]:
```
y = torch.mm(torch.mm(a.t(), X), b)
```

In [61]:
```
y
```

Out[61]:
```
tensor([[1.5715]], grad_fn=<MmBackward>)
```

In [62]:
```
from torch import autograd
grads = autograd.grad(y, X)
```

In [63]:
```
print(grads)
```

```
(tensor([[-1.1047,  1.7177,  0.7609, -0.3598],
        [-0.9547,  1.4844,  0.6576, -0.3109],
```

```
        [ 0.3502, -0.5445, -0.2412,  0.1140]]),)
```

In [64]:
```
print(torch.mm(a, b.t()))
```

```
tensor([[-1.1047,  1.7177,  0.7609, -0.3598],
        [-0.9547,  1.4844,  0.6576, -0.3109],
        [ 0.3502, -0.5445, -0.2412,  0.1140]])
```

In [65]:
```
grads = autograd.grad(y, a)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
/var/folders/7_/dw3jfv_s2vqby4klxk9qy39w0000gq/T/ipykernel_12024/1565098405.py
in <module>
----> 1 grads = autograd.grad(y, a)

~/opt/anaconda3/envs/pytorch-init/lib/python3.8/site-packages/torch/autograd/_
_init__.py in grad(outputs, inputs, grad_outputs, retain_graph, create_graph,
 only_inputs, allow_unused)
    224         retain_graph = create_graph
    225
--> 226     return Variable._execution_engine.run_backward(
    227         outputs, grad_outputs_, retain_graph, create_graph,
    228         inputs, allow_unused, accumulate_grad=False)

RuntimeError: One of the differentiated Tensors does not require grad
```

In [66]:
```
print(a.requires_grad)
```

```
False
```

In [67]:
```
print(X.requires_grad)
```

```
True
```

In [68]:
```
a.dtype
```

Out[68]:
```
torch.float32
```

In [ ]:
```

```