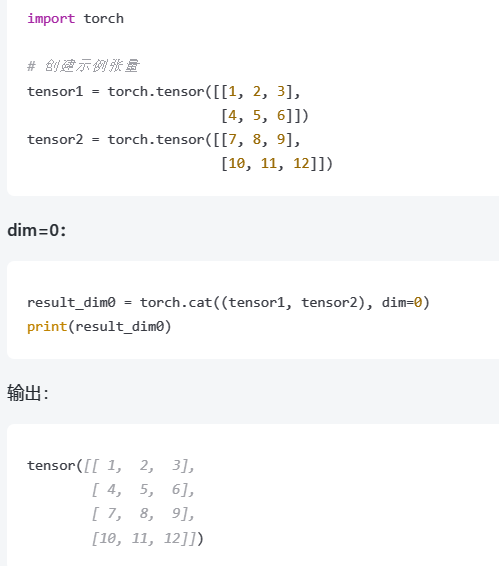
1. 类
2. 调用类首先要用（）进行实例化，调用方式：实例.属性or实例.方法
   1. Self参数：代表类的实例本身
   2. 在调用类属性时要先实例化，类方法中调用方法与属性时可以用self代替类名（就当是个替身，用来简化书写）
   3. 类方法里的第一个默认参数self，传参的时候可以忽略
3. \_\_call\_\_函数将类变成可直接调用的函数
4. 实例化时传入的参数用于初始化类，和调用\_\_call\_\_方法时传入的参数不同！（qwq卡了好久qwq）
5. 继承：在类后面括号中写入另外一个类名，表示当前类继承另外一个类
6. Super（）：即使不写super().\_\_init\_\_()，子类也是自动继承了除父类的\_\_init\_\_()之外的所有方法。
7. 一些乱七八糟的东西
   1. if \_\_name\_\_ == '\_\_main\_\_'：当于是 Python 模拟的程序入口，\_\_name\_\_是内置变量，用于表示当前模块的名字。
   2. Super（）：即使不写super().\_\_init\_\_()，子类也是自动继承了除父类的\_\_init\_\_()之外的所有方法。
   3. 多重赋值语法：允许同时将一个元组（或列表等可迭代对象）中的多个值分配给多个变量。

Eg：for data in test\_dataloader:

imgs, targets = data

* 1. torch.cat((tensor1, tensor2), dim)



* 1. 打开tensorboard后呈现三角形而非直线：多次加载导致重复，删除logs文件夹里的文件再运行即可
  2. tensorboard --logdir logs 要在terminal中运行
  3. 不写writer.close会报错，切记别忘了
  4. torch.randn函数是PyTorch中用于生成具有正态分布（均值为0，标准差为1）的随机数的函数。它可以用于创建具有指定形状的张量，并且张量中的每个元素都是独立的随机数，遵循标准正态分布（均值为0，标准差为1）