1. 实现基类 Filter，至少包括两个数据属性，一个属性是待处理的图片实例，即 PIL 库的 Image 实例，另一个是参数列表，用以存储可能使用参数的滤波器的参数；至少包括一个方法属性，即 filter()方法，能够对 Image 实例的特定处理。但在该类中并不需要进行实现，其实现细节应该交给子类。

2. 实现 Filter 类的多个子类，分别实现对图片的一些滤波处理，至少应进行边缘提取，锐化，模糊及大小调整四类操作，也即应实现至少 4 个子类，分别对基类中的 filter()方法进行实现。注意，并不需要真正实现对应的操作，可简单地通过 PIL 中的 Image 和 ImageFilter 模块来实现。具体可参见 https://pillow.readthedocs.io/en/stable/reference/ImageFilter.html。

3. 实现类 ImageShop，其至少包含四个数据属性，分别是图片格式，图片文件（应该支持目录），存储图片实例(Image 实例)的列表以及存储处理过的图片（如果需要的话）。至少包含如下方法属性，分别是从路径加载特定格式的图片（load_images()，应加载文件或目录中的所有特定格式图片）；处理图片的内部方法__batch_ps(Filter),利有某个过滤器对所有图片进行处理；批量处理图片的对外公开方法（batch_ps()），注意该方法要至少有一个操作参数，且该参数可以不定长，即可以同时进行若干操作（如调整大小并模糊等），其参数可定义成一种特定格式的 tuple 输入，比如（操作名，参数），根据操作名生成对应的 Filter 子类并调用 __batch_ps 来完成批处理；处理效果显示（display（）），注意该方法应该有参数，如考虑多图片呈现可能需要行，列以及每张图片的大小，以及最多显示多少张等，可通过 matplotlib 中的 subplot 等来实现；处理结果输出（save()），该方法应该有输出路径或输出格式等参数。

4. 实现测试类 TestImageShop，对该类进行测试，指定图片路径，指定要进行的操作（如有参数也可应提供），并对执行结果进行呈现和存储。

5. 附加：观察一些经过过滤后图片的变化，思考这些处理对图片本身的一些相关的计算，如图片的相似性等有无影响，并进行简单的实验验证。另外，这些预处理本身对下游的机器学习模型会带来哪些好处？

6. 附加：进一步了解 torchvision.transforms 中对图片的一些基础操作，以及这些基础操作在数据增强中的常见应用，如作业中的资料：实践教程 | 13 个 Pytorch 图像增强方法总结.html。

7. 附加：进一步了解一下深度卷积网络以及其在计算机视觉中的应用现状。