1. 基类 Filter

Image 为 Image 的实例，parmlist 为可能的列表

```
class Filter:
    def __init__(self,image) -> None:
        self.parmlist=[]
        self.im=image

    def filter(self):
        pass
```

2. Filter 的子类

（1） Edgefilter

用于对图片进行边缘提取

```
class Edgefilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self):
        self.im=self.im.filter(ImageFilter.FIND_EDGES)
        #self.im.show()
        return self.im
```

（2） Sharpenfilter

对图片进行锐化

```
class Sharpenfilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self):
        self.im=self.im.filter(ImageFilter.SHARPEN)
        #self.im.show()
        return self.im
```

（3） Blurfilter

对图片进行模糊处理

```
class Blurfilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self):
        self.im=self.im.filter(ImageFilter.BLUR)
        #self.im.show()
        return self.im
```

（4） Resizefilter

调整图片的尺寸

```
class resizefilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self,m,n):
        self.im=self.im.resize((m,n))
        #self.im.show()
        return self.im
```

（5） Contourfilter

提取图片的浮雕

```
class Contourfilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self):
        self.im=self.im.filter(ImageFilter.CONTOUR)
        #self.im.show()
        return self.im
```

（6） Edge_enhancefilter

增强图片的边缘

```
class Edge_enhancefilter(Filter):
    def __init__(self, image) -> None:
        self.im=image

    def filter(self):
        self.im=self.im.filter(ImageFilter.EDGE_ENHANCE_MORE)
        #self.im.show()
        return self.im
```

3. ImageShop

（1） 初始化

利用字典来储存属性，分别为图片文件（目录）的路径，Image 实例的列表，处理后的图片列表和图片的类型

```
def __init__(self,path):
    self.info={}
    self.info["path"]=path
    self.info["images_list"]=[]
    self.info["images_dealt"]=[]
```

（2） load_images

允许输入一个或者多个要加载的图片类型；同时可以判断是图片文件还是目录，如果是文件那么图片类型是单个字符串，如果是目录将会对该目录下的所有符合格式要求的图片进行加载，用列表储存实例和图片的类型。

```
def load_images(self,*ptype):
    type_list=list(ptype)
    if os.path.isfile(self.info["path"]):
        a,b=os.path.split(self.info["path"])
        m,n=os.path.splitext(b)
        if n in type_list:
            self.info["images_list"].append(Image.open(self.info["path"]))
            self.info["image_type"]=b

    elif os.path.isdir(self.info["path"]):
        self.info["images_type"]=[]
        for dirpath,dirnames,files in os.walk(self.info["path"]):
            for file in files:
                f,e = os.path.splitext(file)
                if e in type_list:
                    self.info["images_list"].append(Image.open(os.path.join(self.info["path"],file)))
                    self.info["images_type"].append(file)
```

（3） 内部方法__batch_ps

首先给出对应 Filter 子类的过滤器的操作列表，根据用户的输入执行对应的操作，对实例列表中的图片进行批量处理；考虑到可能存在同时有多个操作，此时要判断图片是否已经经过操作（如果处理后的图片列表为空则图片没有进行过操作，反之则经过），如果经过操作那么要对储存处理后的图片列表进行修改，反之对原始列表进行修改

```
def __batch_ps(self,filterway,*arg):
    ways_list=["BLUR","RESIZE","SHARPEN","EDGE","CONTOUR","EDGE-ENHANCE"]
    if filterway not in ways_list:
        print("filter not exist")
    else:
        if filterway == ways_list[0]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Blurfilter(j).filter())
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = resizefilter(self.info["images_dealt"][i]).filter()
        elif filterway == ways_list[1]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(resizefilter(j).filter(arg[0],arg[1]))
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = resizefilter(self.info["images_dealt"][i]).filter((arg[0],arg[1]))
        elif filterway == ways_list[2]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Sharpenfilter(j).filter())
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = Sharpenfilter(self.info["images_dealt"][i]).filter()
        elif filterway == ways_list[3]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Edgefilter(j).filter())
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = Edgefilter(self.info["images_dealt"][i]).filter()
        elif filterway == ways_list[4]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Contourfilter(j).filter())
```

```
        elif filterway == ways_list[4]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Contourfilter(j).filter())
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = Contourfilter(self.info["images_dealt"][i]).filter()
        elif filterway == ways_list[5]:
            if self.info["images_dealt"] == []:
                for j in self.info["images_list"]:
                    self.info["images_dealt"].append(Edge_enhancefilter(j).filter())
            else:
                length=len(self.info["images_dealt"])
                for i in range(length):
                    self.info["images_dealt"][i] = Edge_enhancefilter(self.info["images_dealt"][i]).filter()
```

（4） 外部方法 batch_ps

用户可以输入一个或者多个操作指令，如果操作指令有参数，要按照（指令，参数）的形式输入

```
def batch_ps(self,*arg):
    for i in arg:
        if isinstance(i,tuple) and i[0] == "RESIZE":
            ImageShop.__batch_ps(self,i[0],int(i[1]),int(i[2]))
        else:
            ImageShop.__batch_ps(self,i)
```

（5） Display

对处理结果进行展示，如果为多个图片，用户可以输入想要的行数和列数，同时能同时显示的最大数量为图片总数和行列数的乘积两者的最小值；

```
def display(self,*arg):
    if os.path.isfile(self.info["path"]):
        self.info["images_list"][0].show()
    else:
        max=len(self.info["images_dealt"])
        m=arg[0]
        n=arg[1]
        if max > m*n:
            for i in range(1,m*n+1):
                plt.subplot(m,n,i)
                plt.imshow(self.info["images_dealt"][i-1])
                plt.xticks([])
                plt.yticks([])
                plt.axis('off')
            plt.show()
        else:
            for i in range(1,max+1):
                plt.subplot(m,n,i)
                plt.imshow(self.info["images_dealt"][i-1])
                plt.xticks([])
                plt.yticks([])
                plt.axis('off')
            plt.show()
```

（6） Save

可以根据用户要求对图片进行保存，mode 参数控制保存的格式，默认为 0，即保留原来的格式；同时也可以输入想要保存的格式，如“png”,“gif”等

```
def save(self,path,mode = 0):
    if mode == 0:
        if len(self.info["images_dealt"]) == 1:
            self.info["images_dealt"][0].save(os.path.join(path,self.info["image_type"]))
        else:
            for i in range(len(self.info["images_dealt"])):
                self.info["images_dealt"][i].save(os.path.join(path,self.info["images_type"][i]))
    else:
        if len(self.info["images_dealt"]) == 1:
            self.info["images_dealt"][0].save(os.path.join(path,os.path.splitext(self.info["image_type"])[0]),mode)
        else:
            for i in range(len(self.info["images_dealt"])):
                self.info["images_dealt"][i].save(os.path.join(path,os.path.splitext(self.info["images_type"][i])[0])+"."+mode)
```

4. 测试类 TestImageShop

分别对图片的加载功能（选择加载 jpg 格式的图片），批量处理功能（同时对所有图片进行模糊并且调整大小为 300*500），处理结果展示（以 3*3 的形式展示），保存处理结果（将图片保存为 png 格式）进行测试

```
class TestImageShop:
    def __init__(self) -> None:
        self.t=ImageShop(picpath)

    def loadtest(self):
        self.t.load_images(".jpg")
        print(self.t.info)

    def displaytest(self):
        self.t.batch_ps("BLUR",("RESIZE",300,500))
        self.t.display(3,3)

    def savetest(self):
        self.t.save(resultpath,"png")
```

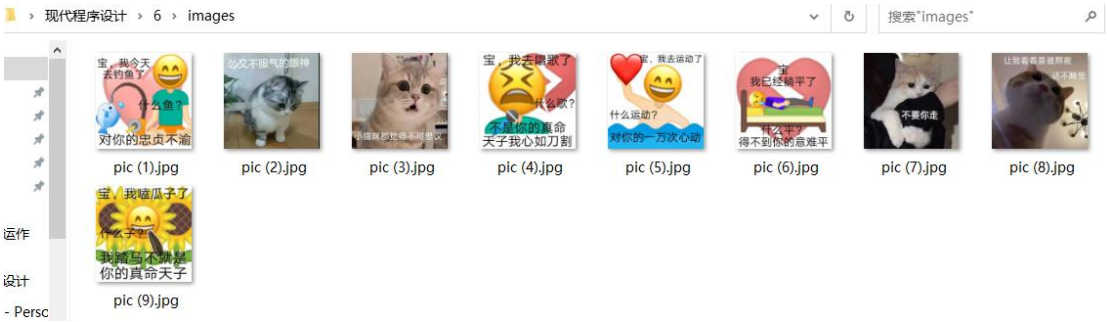
```
from PIL import Image,ImageFilter
from matplotlib import pyplot as plt
import os

picpath=r"C:\Users\86186\Desktop\现代程序设计\6\images"
resultpath=r"C:\Users\86186\Desktop\现代程序设计\6\result"
```

```

s=TestImageShop()
s.loadtest()
s.displaytest()
s.savetest()
```

原始图片：

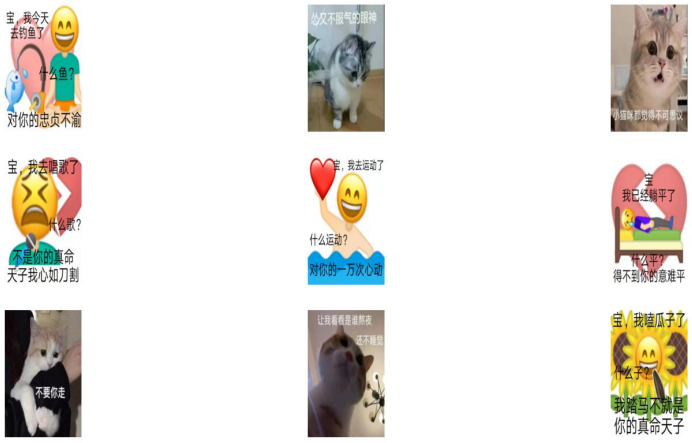


运行结果：

（1） 图片加载的字典


```
{'path': 'C:\\Users\\86186\\Desktop\\现代程序设计\\6\\images', 'images_list': [<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1080x1080 at 0x25CAF044DA0>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=690x690 at 0x25CAF10C588>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=632x630 at 0x25CAF10C5F8>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1080x1080 at 0x25CAF10C668>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1080x1080 at 0x25CAF10C6D8>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1080x1080 at 0x25CAF10C748>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=960x966 at 0x25CAF10C7B8>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=690x690 at 0x25CAF10C828>, <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1080x1080 at 0x25CAF10C898>], 'images_dealt': [], 'images_type': ['pic (1).jpg', 'pic (2).jpg', 'pic (3).jpg', 'pic (4).jpg', 'pic (5).jpg', 'pic (6).jpg', 'pic (7).jpg', 'pic (8).jpg', 'pic (9).jpg']}
```

（2） 处理后的效果展示，可以发现比列发生了变化



（3） 保存的图片结果