**目 录**

1. **用Python来分割图片**
2. **用Python来合并图片**
3. **使用Python的第三方库itchat获取全部微信好友头像，并拼接成大图**
4. **过滤掉朋友圈头像中不含人脸的图片**
5. **最后任务---生成照片墙**

**01.用PYTHON来分割图片**

**目的**

将给定的图片分解成小图片，为下一节学习图片合并原理，准备图片资源。

* 列表推导式
* PIL crop方法

**安装pillow**

**Python install pillow**

from PIL import Image  
  
# 区域由一个4元组定义，表示为坐标是 (left, upper, right, lower)。 Python Imaging Library 使用左上角为 (0, 0)的坐标系统。  
# 同时要注意，这些坐标指向像素之间的位置，因此上述例子中描述的区域的大小为300x300像素。  
# 后两个数字需要比前两个大

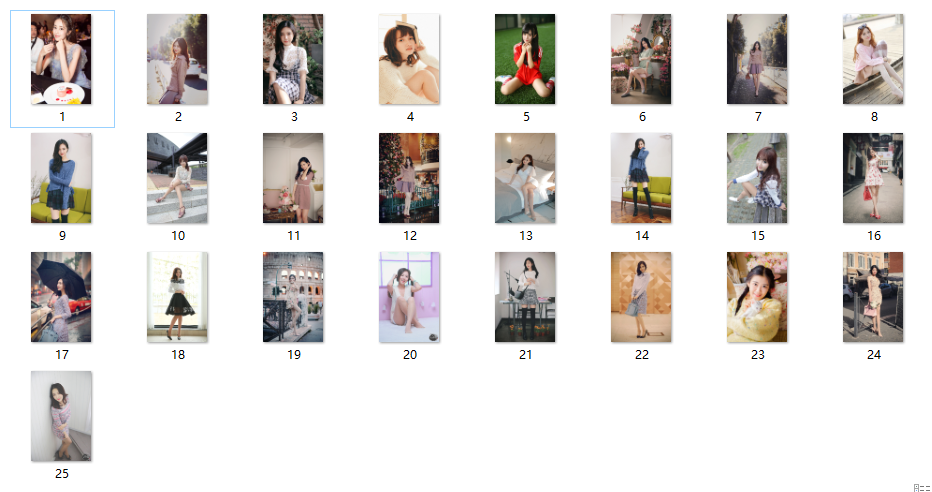
def cut\_image(image,count1,count2):  
 width, height = image.size  
 item\_width = int(width / count1)  
 item\_height = int(height / count2)  
 box\_list = [] #定义一个空列表，容器  
 # (left, upper, right, lower)

for i in range(0,count1):  
 for j in range(0,count2):  
 box = (j\*item\_width,i\*item\_height,(j+1)\*item\_width,(i+1)\*item\_height)  
 box\_list.append(box) #append向后面添加元素，参数可以是任何东西，将作为元素添加到列表

尾部。

image\_list = [image.crop(box) for box in box\_list]#截取子图片，列表推导式  
 return image\_list

#保存  
def save\_images(image\_list):  
 index = 1  
 for image in image\_list:  
 image.save(str(index) + '.png', 'PNG')  
 index += 1  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 file\_path = "total1.png" #图片保存的地址  
 image = Image.open(file\_path)  
  
 image\_list = cut\_image(image,5,5)  
 save\_images(image\_list)

效果展示：

把一整张图片切割成数张小的图片。

**02.用Python来合并图片**

**目的**

为第三步制作照片墙功能热身。（照片墙功能中用到了 PIL paste方法）

* PIL paste方法

将第一个任务分解出来的图片放到项目的images目录中。

import os  
from PIL import Image  
#把目录下的图片拼接成一张大图片  
  
# 图片压缩后的大小  
width\_i = 200  
height\_i = 300  
  
# 每行每列显示图片数量  
line\_max = 5  
row\_max = 5  
  
# 参数初始化  
all\_path = []  
num = 0  
pic\_max = line\_max \* row\_max #列\*行,图片最大数量  
  
for filename in os.listdir('./images'): #方法用于返回指定的文件夹包含的文件或文件夹的名字的列表。  
 if filename.endswith('jpg') or filename.endswith('png'):  
 #函数判断一个文本是否以某个或几个字符结束，结果以True或者False返回。  
 all\_path.append(os.path.join('./images', filename)) #os.path.join()函数用于路径拼接文件路径。  
  
toImage = Image.new('RGBA', (width\_i \* line\_max, height\_i \* row\_max)) #Image.new(mode,size,color)  
  
for i in range(0, row\_max):  
 for j in range(0, line\_max):  
 # 图片比计划的少  
 if num >= len(all\_path):  
 print("break")  
 break  
 pic\_fole\_head = Image.open(all\_path[num])  
 width, height = pic\_fole\_head.size #元组拆包  
 tmppic = pic\_fole\_head.resize((width\_i, height\_i)) #重设宽高，最终图片  
 loc = (int(i % line\_max \* width\_i), int(j % line\_max \* height\_i))  
 print("第" + str(num) + "存放位置" + str(loc))  
 toImage.paste(tmppic, loc)  
 #paste函数的参数为(需要修改的图片，粘贴的起始点的横坐标，粘贴的起始点的纵坐标）  
 num = num + 1  
 # 图片比计划的多  
 if num >= pic\_max:  
 break  
  
print(toImage.size)  
toImage.save('merged.png')

效果展示：

**03.使用python的第三方库itchat获取全部微信好友头像，并拼接成大图**

**目的**

为第四步过滤非人脸图片做准备。

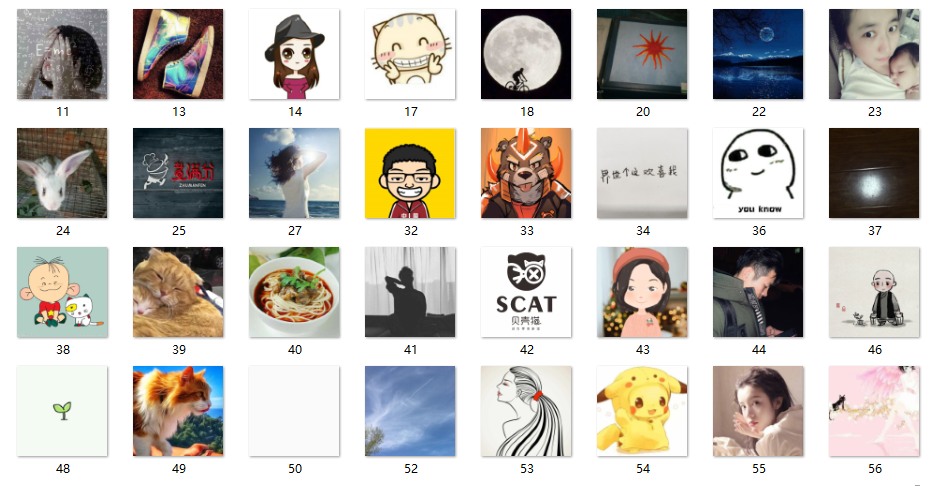
**安装第三方库itchat**

**pip install itchat**

注意，首先项目目录下要创建一个user目录，用来保存获取的微信好友头像。

import itchat  
import PIL.Image as Image  
from os import listdir  
  
def get\_imgs():#完成主要的下载头像的任务  
 #每次都需要扫描二维码，方便切换微信账号测试，调用即可完成登录。  
 itchat.auto\_login(hotReload=False)  
 #获取朋友列表（只获取256个），返回字典类型的数据集，获取好友的索引数  
 friends = itchat.get\_friends(update=True)[0:256]  
 #为图片命名的变量  
 num = 0  
 #遍历好友列表  
 for friend in friends:  
 #获取好友的头像  
 img = itchat.get\_head\_img(userName=friend["UserName"])  
 #在项目文件的主创建一个user文件用于放头像，并写入对应的图片名，空白的  
 fileImage = open("./user/" + str(num) + ".jpg",'wb')  
 #将获取到的头像文件写到创建的图片文件中  
 fileImage.write(img)  
 #关闭资源  
 fileImage.close()  
 num += 1  
#制作大的大头像  
def get\_big\_img():  
 #获取usr文件夹所有文件的名称  
 pics = listdir("user") #os.listdir() 方法用于返回指定的文件夹包含的文件或文件夹的名字的列表。这个列表以字母顺序。  
 #创建图片大小  
 toImage = Image.new("RGB", (800, 800))  
 #用于图片的位置  
 x = 0  
 y = 0  
 #遍历user文件夹的图片  
 for pic in pics:  
 #依次打开图片  
 img = Image.open("user/{}".format(pic)) #一个循环,相当于从userList里面取出每个元素user;在{}里可以直接使用user,即每次取出的值。  
 #重新设置图片的大小  
 img = img.resize((50, 50), Image.ANTIALIAS) #重新设定大小,设定ANTIALIAS,即抗锯齿  
 #将图片粘贴到最后的大图片上，需要注意对应的位置  
 toImage.paste(img, (x \* 50, y \* 50))  
 #设置每一行排16个图像  
 x += 1  
 if x == 16:  
 x = 0  
 y += 1 #一行16个排满切换到下一行  
  
 #保存图片为bigPhoto.jpg  
 toImage.save("user/" +"bigPhoto.jpg")  
 #将做好图片发送东自己的手机上  
 itchat.send\_image("user/" +"bigPhoto.jpg", 'filehelper')  
#定义执行的主函数  
def main():  
 get\_imgs()  
 get\_big\_img()  
#运行  
if \_\_name\_\_=="\_\_main\_\_":  
 main()

效果展示：



**04.过滤掉朋友圈头像中不含人脸的图片**

**目的**

为第五步合成照片墙做准备，过滤掉朋友圈头像中不含人脸的图片。

**安装第三方库opencv**

**pip install opencv**

**项目准备**

* 新建一个目录(user\_no\_face)，不含人脸的图片会被转移到该目录
* 把人脸识别模型文件（haarcascade\_frontalface\_alt.xml），拷贝到项目目录下。

备注：

1. detecMultiScale()函数

参数介绍：

参数1：image--待检测图片，一般为灰度图像加快检测速度；

参数2：objects--被检测物体的矩形框向量组；  
参数3：scaleFactor--表示在前后两次相继的扫描中，搜索窗口的比例系数。默认为1.1即每次搜索窗口依次扩大10%;  
参数4：minNeighbors--表示构成检测目标的相邻矩形的最小个数(默认为3个)。  
如果组成检测目标的小矩形的个数和小于 min\_neighbors - 1 都会被排除。  
如果min\_neighbors 为 0, 则函数不做任何操作就返回所有的被检候选矩形框，  
这种设定值一般用在用户自定义对检测结果的组合程序上；  
参数5：flags--要么使用默认值，要么使用CV\_HAAR\_DO\_CANNY\_PRUNING，如果设置为  
CV\_HAAR\_DO\_CANNY\_PRUNING，那么函数将会使用Canny边缘检测来排除边缘过多或过少的区域，  
因此这些区域通常不会是人脸所在区域；

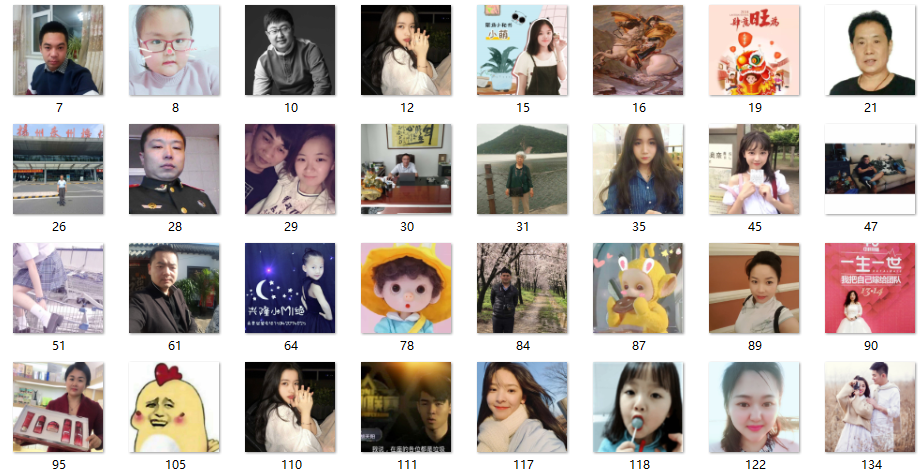
参数6、7：minSize和maxSize用来限制得到的目标区域的范围。

2.人脸检测器

Opencv自带训练好的人脸检测模型，存储在sources/data/haarcascades文件夹和sources/data/lbpcascades文件夹下。其中几个.xml文件如下：  
人脸检测器（默认）：haarcascade\_frontalface\_default.xml  
人脸检测器（快速Harr）：haarcascade\_frontalface\_alt2.xml  
人脸检测器（侧视）：haarcascade\_profileface.xml  
眼部检测器（左眼）：haarcascade\_lefteye\_2splits.xml  
眼部检测器（右眼）：haarcascade\_righteye\_2splits.xml  
嘴部检测器：haarcascade\_mcs\_mouth.xml  
鼻子检测器：haarcascade\_mcs\_nose.xml  
身体检测器：haarcascade\_fullbody.xml  
人脸检测器（快速LBP）：lbpcascade\_frontalface.xml

import os  
import cv2  
import shutil  
  
def getAllPath(dirpath):  
 PathArray = [] #定义一个空列表，存文件名  
 for filename in os.listdir(dirpath): #返回指定的文件夹包含的文件或文件夹的名字的列表。  
 if filename.endswith('jpg') or filename.endswith('png'):  
 PathArray.append(os.path.join(dirpath, filename)) #连接两个或更多的路径名组件  
 return PathArray  
  
# 从源路径中读取所有图片放入一个list，然后逐一进行检查，把其中的脸扣下来，存储到目标路径中  
def readPicSaveFace(sourcePath,invalidPath): #无效的  
 try:  
 ImagePaths = getAllPath(sourcePath)  
 # 对list中图片逐一进行检查,找出其中的人脸然后写到目标文件夹下  
 # haarcascade\_frontalface\_alt.xml为库训练好的分类器文件，下载opencv，安装目录中可找到  
 face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_alt.xml') #级联分类器  
 for imagePath in ImagePaths:  
 img = cv2.imread(imagePath)  
 if type(img) != str:  
 faces = face\_cascade.detectMultiScale(img, 1.1, 5)  
 #它可以检测出图片中所有的人脸，并将人脸用vector保存各个人脸的坐标、大小（用矩形表示），函数由分类器对象调用  
 # 参数1：image - -待检测图片，一般为灰度图像加快检测速度；  
 # 参数3：scaleFactor - -表示在前后两次相继的扫描中，搜索窗口的比例系数。默认为1.1  
 # 即每次搜索窗口依次扩大10 %;  
 # 参数4：minNeighbors - -表示构成检测目标的相邻矩形的最小个数(默认为3个)。  
 # 如果组成检测目标的小矩形的个数和小于min\_neighbors - 1都会被排除。  
 # 如果min\_neighbors为0, 则函数不做任何操作就返回所有的被检候选矩形框，  
 # 这种设定值一般用在用户自定义对检测结果的组合程序上；  
 if len(faces):  
 print(imagePath + " have face")  
 else:  
 shutil.move(imagePath, invalidPath) #移动文件(目录)  
 except IOError:  
 print("Error")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 invalidPath = './user\_no\_face'  
 sourcePath = 'user'  
 readPicSaveFace(sourcePath,invalidPath)

效果展示：



**05.最后任务---生成照片墙**

**学习一个知识python匿名函数（lambda表达式）**

**了解一个函数point**

from PIL import Image

im = Image.open('35.jpg').convert('RGBA')

\_, \_, \_, alpha = im.split()

alpha = alpha.point(lambda i: 150)

im.putalpha(alpha)

im.save('luozi.png')

**准备**

1.字体文件demo.ttf  
链接：[https://pan.baidu.com/s/1wLKMd0VoO7C5jCAAfpjBEw](https://links.jianshu.com/go?to=https%3A%2F%2Fpan.baidu.com%2Fs%2F1wLKMd0VoO7C5jCAAfpjBEw" \t "_blank)  
提取码：ub9l  
2.项目目录下新建文件夹out，存放最终生成的图片

# 匿名函数,Python的lambda表达式基本语法是在冒号（：）左边放原函数的参数，可以有多个参数，用逗号（，）隔开即可；冒号右边是返回值。  
#写一些执行脚本时，使用lambda就可以省下定义函数过程，简化代码的可读性，由于普通的函数阅读经常要跳到开头def定义部分，  
# 使用lambda函数可以省去这样的步骤。  
from PIL import Image, ImageDraw, ImageFont  
import os  
#gen\_text\_img函数根据设定的文字以及字体大小生成一张图片（包含设定的文字），  
# 先用给定的字符串生成一张图片，然后将该图片的每个像素的宽扩张edge\_len倍，高也扩张edge\_len倍，假设edge\_len=60，  
# 那么原文字图片（如上图）的每个像素就变成了60\*60像素的一个图片（我们会将每个朋友圈头像放进去）；原文字图片的每个像素的  
# 透明度不同，比如显示芒种这2个字的地方，透明度低（不透明），这2个字周边的地方，透明度高（透明），我们根据原文字图片每个  
# 像素的透明度，来设定放到这个像素（其实宽高已经扩大了60倍）位置的微信好友头像的透明度（trans\_alpha方法实现）。  
def gen\_text\_img(text, font\_size=20, font\_path=None):  
 # 从文字生成图像，输入：文字内容，文字字体大小，字体路径  
 font = ImageFont.truetype(font\_path, font\_size) if font\_path is not None else None  
 #加载一个TrueType字体文件,并且创建一个字体对象。  
 (width, length) = font.getsize(text) # 获取文字大小,元组拆包  
 text\_img = Image.new('RGBA', (width, length)) #文字图片  
 draw = ImageDraw.Draw(text\_img) #创建一个可以在给定图像上绘图的对象。  
 # 第一个tuple表示未知(left,up)，之后是文字，然后颜色，最后设置字体  
 draw.text((0, 0), text, fill=(0, 0, 0), font=font) #参数：位置，文本，颜色，字体  
 text\_img.save('testtext.png')  
 return text\_img  
  
def trans\_alpha(img, pixel):  
 *'''根据pixel的rgba调节img的透明度  
 这里传进来的pixel是一个四元组（r,g,b,alpha）  
 '''* \_, \_, \_, alpha = img.split() #分离通道  
 alpha = alpha.point(lambda i: pixel[-1]\*10) #point()改变像素点(函数),匿名函数，提高图片透明度，使观察方便  
 img.putalpha(alpha) #对图像添加 alpha 层，即将给定通道拷贝到图像 alpha 层.  
 return img  
  
def picture\_wall\_mask(text\_img, edge\_len, pic\_dir="./user"):  
 # 根据文字图像生成对应的照片墙，输入：文字图像，各个照片边长，照片所在路径  
 new\_img = Image.new('RGBA', (text\_img.size[0] \* edge\_len, text\_img.size[1] \* edge\_len))  
 file\_list = os.listdir(pic\_dir) #返回指定的文件夹包含的文件或文件夹的名字的列表。  
 img\_index = 0  
 for x in range(0, text\_img.size[0]):  
 for y in range(0, text\_img.size[1]):  
 pixel = text\_img.getpixel((x, y)) #获取某个像素位置的值  
 file\_name = file\_list[img\_index % len(file\_list)]  
 try:  
 img = Image.open(os.path.join(pic\_dir, file\_name)).convert('RGBA') #把目录和文件名合成一个路径  
 img = img.resize((edge\_len, edge\_len))  
 img = trans\_alpha(img, pixel)  
  
 new\_img.paste(img, (x \* edge\_len, y \* edge\_len))  
 img\_index += 1  
 except Exception as e:  
 print(f"open file {file\_name} failed! {e}")  
 return new\_img  
  
def main(text='', font\_size = 20, edge\_len = 60,pic\_dir = "./user", out\_dir = "./out/", font\_path = './demo.ttf'):  
  
 if len(text) >= 1:  
 text\_ = ' '.join(text)#将字符串用空格分隔开  
 print(f"generate text wall for '{text\_}' with picture path:{pic\_dir}")  
 #加f后可以在字符串里面使用用花括号括起来的变量和表达式,  
 text\_img = gen\_text\_img(text\_, font\_size, font\_path)  
 # text\_img.show()  
 img\_ascii = picture\_wall\_mask(text\_img, edge\_len, pic\_dir)  
 # img\_ascii.show()  
 img\_ascii.save(out\_dir + os.path.sep + '\_'.join(text) + '.png') #os.path.sep:路径分隔符  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main(text='芒种')

代码原理：

1.gen\_text\_img函数根据设定的文字以及字体大小生成一张图片（包含设定的文字），如下图所示：

先用给定的字符串生成一张图片（如上图），然后将该图片的每个像素的宽扩张edge\_len倍，高也扩张edge\_len倍，假设edge\_len=60，那么原文字图片（如上图）的每个像素就变成了60\*60像素的一个图片（我们会将每个朋友圈头像放进去）；原文字图片的每个像素的透明度不同，比如显示东软这2个字的地方，透明度低（不透明），这2个字周边的地方，透明度高（透明），我们根据原文字图片每个像素的透明度，来设定放到这个像素（其实宽高已经扩大了60倍）位置的微信好友头像的透明度（trans\_alpha方法实现）。

最终效果：

