python高阶教程-使用imap接收邮箱的附件(中文字符编码与MIME)

本篇内容来自原创小册子《*python*高阶教程》，点击查看目录。

背景

我们常常要使用邮件来接收报名表、作业等内容，然后统计出已交和未交的人数，通知没有发邮件的注意时间。

这是一个很繁琐的流程，而如果能够用程序自动化地完成该过程，无疑会大大加快工作进度。

初始代码

在网上找了一份使用imap的代码，如下：

NOTE 这份代码仅供示例，可能并不工作

```
import imaplib, string, email
M = imaplib.IMAP4_SSL("imap.gmail.com")
print(M)
try:
    try:
        M.login('chemboking@gmail.com','12345678')
    except Exception,e:
        print('login error: %s' % e)
        M.close()
    M.select()
    result, message = M.select()
    typ, data = M.search(None, 'ALL')
    for num in string.split(data[0]):
        try:
            typ, data = M.fetch(num, '(RFC822)')
            msg = email.message_from_string(data[0][1].decode('utf8'))
            print msg["Subject"]
        except Exception,e:
            print('got msg error: %s' % e )
    M.logout()
    M.close()
except Exception, e:
    print('imap error: %s' % e)
    M.close()
```

问题一：主题不能显示中文

在`print msg["Subject"]`输出是`=?gb2312?B?1dDJzNL4ONDQxdPDv6i159fT1cu1pTIwMTjE6jEw1MI=?=`,

并不是中文，推测前面的gb2312是编码信息。并且除了gb2312还有utf8。

考虑到自己写一个解析器太麻烦，搜索到可以使用email.header.decode_header()函数，输入包含编码信息

的base64字符串，解析出解码后的字节串和charset，解析器的返回是list套一个tuple，即(bytes, charset).

如果charset为None，说明bytes里面不是字节而是str.

问题二：从fetch到email总是解码出错

很快，在解析了几个较简单的邮件之后程序就报错，报错行是

msg = email.message_from_string(data[0][1].decode('utf8'))，单独调试发现可能是gbk编码，

于是采用data[0][1].decode('utf8'),又欢快地跑了一阵，再次报错，尝试了gb2132,gb18030等

等编码都没有效果。

查看fetch后的原始信息，发现里面有MIME字样，并且明确提醒某些邮箱客户端可能不支持。这就不是更换编码可以解决的了。

在email

和imaplib搜索了很久也没有有用信息，偶尔看到博客上有人的代码是这样写的：

```
from email.parser import Parser
messages = [server.retr(i) for i in range(1, len(server.list()[1]) + 1)]
messages = [b'¥r¥n'.join(mssg[1]).decode() for mssg in messages]
messages = [Parser().parsestr(mssg) for mssg in messages]
```

这份代码是使用pop接收163邮件的，在我的126上无法使用，原因未知，所以我采用imap。这里采用的方法是Parser().parsestr。但是有点鸡肋，因为依旧是使用utf8对fetch后的数据解码，然后用Parser()解析。碰到这种MIME的邮件估计还是要出问题。

好在很快就在bytes parser找到了一个可以使用的类email.parser.BytesParser，该类自动解码，测试可以对MIME邮件进行解析。

问题三： 出现unknown-8bit编码

在采用BytesParser之后，依旧使用msg.get('Subject')获取邮件主题，使用email.header.decode_header()对base64进行解码，却在解码后出现了unknown-8bit的charset。经测试，将其强制解码为utf8是可行的。

search 邮件

搜索中文

```
typ, msg_ids = c.search('GB2312','SUBJECT "消费提醒"'.encode('gb2312'))
```

搜索英文不再赘述。

注意使用163邮箱可能无法搜索；将gb2312换为utf8也无法搜索。

猜测这里的搜索是基于云端的，即imap服务器进行实际搜索，所以与编码相关，即使用utf8的字符串无法匹配gb2312.

获取附件

附件的获取按照网络上的代码没有出现问题.

```
for part in message.walk():
    fileName = part.get_filename()
    fileName = decode_str(fileName)
    # 保存附件

    if fileName:
        with open(fileName, 'wb') as fEx:
            data = part.get_payload(decode=True)
            fEx.write(data)
            print("附件%s已保存" % fileName)
```

其中,decode_str是自定义函数，在文末的代码中可以找到。

整体可以work的代码

```
import imaplib
import email
from email.parser import BytesParser
from email.utils import parseaddr
host = 'imap.126.com'
user = ''
passwd = ''
mail_directory = 'INBOX'
conn = imaplib.IMAP4(host)
conn.login(user,passwd)
conn.select(mail_directory)
status, data = conn.search(None, 'ALL')
email_list = list(reversed(data[0].split()))
def decode_str(s):
    try:
        subject = email.header.decode_header(s)
    except:
        # print('Header decode error')
        return None
    sub_bytes = subject[0][0]
    sub_charset = subject[0][1]
    if None == sub_charset:
        subject = sub_bytes
    elif 'unknown-8bit' == sub_charset:
        subject = str(sub_bytes, 'utf8')
    else:
        subject = str(sub_bytes, sub_charset)
    return subject
def get_email(num, conn):
    typ, content = conn.fetch(num, '(RFC822)')
    msg = BytesParser().parsebytes(content[0][1])
```

```
            sub = msg.get('Subject')
            for part in msg.walk():
                fileName = part.get_filename()
                fileName = decode_str(fileName)
                if None != fileName:
                    print('++++++++++++++++++')
                    print(fileName)

            print(num, decode_str(sub))
        for num in email_list:
            get_email(num, conn)
        conn.close()
        conn.logout()
```