机器人流程自动化RPA

最佳实践

最佳实践

1.Email

```
from email.mime.text import MIMEText
msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

暂无

Remarks/备注

```
#注意s不能简单地传入‘name <addr@example.com>’ ，因为如果包含中文，需要通过Header对象进行编码。
name, addr = parseaddr(s)
#接收的是字符串而不是list，如果有多个邮件地址，用,分隔即可。
msg['To']
#加密SMTP 使用标准的25端口连接SMTP服务器时，使用的是明文传输，发送邮件的整个过程可能会被窃听。要更安全地发送邮件，可以加密SMTP会话，实际上就是先创建SSL安全连接，然后再使用SMTP协议发送邮件。
smtp_server = 'smtp.gmail.com'
smtp_port = 587
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()
# 剩下的代码和前面的一模一样:
server.set_debuglevel(1)
```

Related/相关

常用邮箱端口说明：

邮箱	SMTP服务器	SSL协议端口	非SSL协议端口
163	smtp.163.com	465或者994	25
qq	smtp.qq.com	465或587	25

邮件发送

1.构造MIMEText对象时，第一个参数就是邮件正文，第二个参数是MIME的subtype，传入'plain'表示纯文本，最终的MIME就是'text/plain'，最后一定要用utf-8编码保证多语言兼容性。
2.用set_debuglevel(1)就可以打印出和SMTP服务器交互的所有信息。SMTP协议就是简单的文本命令和响应。login()方法用来登录SMTP服务器，sendmail()方法就是发邮件，由于可以一次发给多个人，所以传入一个list，邮件正文是一个str，as_string()把MIMEText对象变成str。

Example/实例

Example 邮件发送

```
def _format_addr(s):
name, addr = parseaddr(s)
return formataddr((Header(name, 'utf-8').encode(), addr))
pass
def EmailSend():
from_addr = '2771403***@qq.com'
password ='emi**********ega'
to_addr ='136*****384@163.com'
smtp_server = 'smtp.qq.com'
msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
msg['From'] = _format_addr('Python爱好者 <%s>' % from_addr)
msg['To'] = _format_addr('管理员 <%s>' % to_addr)
msg['Subject'] = Header('来自SMTP的问候……', 'utf-8').encode()
server = smtplib.SMTP(smtp_server, 25)
server.set_debuglevel(1)
server.login(from_addr, password)
server.sendmail(from_addr, [to_addr], msg.as_string())
server.quit()
pass
```

Example 加密邮件发送

```
#加密邮件
```

```
def _format_addr(s):
name, addr = parseaddr(s)
return formataddr((Header(name, 'utf-8').encode(), addr))
pass
def EmailSend():
from_addr = '2771403***@qq.com'
password ='emi**********ega'
to_addr ='136*****384@163.com'
#smtp_server = 'smtp.qq.com'
msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
msg['From'] = _format_addr('Python爱好者 <%s>' % from_addr)
msg['To'] = _format_addr('管理员 <%s>' % to_addr)
msg['Subject'] = Header('来自SMTP的问候……', 'utf-8').encode()
smtp_server = 'smtp.gmail.com'
smtp_port = 587
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()
# 剩下的代码和前面的一模一样:
server.set_debuglevel(1)
server.login(from_addr, password)
server.sendmail(from_addr, [to_addr], msg.as_string())
server.quit()
pass
```

Example 添加附件发送

```
#增加附件
def _format_addr(s):
name, addr = parseaddr(s)
return formataddr((Header(name, 'utf-8').encode(), addr))
pass
def EmailSend():
from_addr = '2771403***@qq.com'
password ='emi**********ega'
to_addr ='136*****384@163.com'
smtp_server = 'smtp.qq.com'
msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
msg['From'] = _format_addr('Python爱好者 <%s>' % from_addr)
msg['To'] = _format_addr('管理员 <%s>' % to_addr)
msg['Subject'] = Header('来自SMTP的问候……', 'utf-8').encode()
with open('test/test.png', 'rb') as f:
# 设置附件的MIME和文件名，这里是png类型:
mime = MIMEBase('image', 'png', filename='test.png')
# 加上必要的头信息:
mime.add_header('Content-Disposition', 'attachment', filename='test.png')
mime.add_header('Content-ID', '<0>')
mime.add_header('X-Attachment-Id', '0')
# 把附件的内容读进来:
mime.set_payload(f.read())
# 用Base64编码:
encoders.encode_base64(mime)
# 添加到MIMEMultipart:
msg.attach(mime)
server = smtplib.SMTP(smtp_server, 25)
```

```
server.set_debuglevel(1)
server.login(from_addr, password)
server.sendmail(from_addr, [to_addr], msg.as_string())
server.quit()
pass
```

接收Email

```
print(myemail.recive('xxxxxxxx@qq.com','oqkgbaghkakrbiah','pop.qq.com'))
>>>[{'From': ' <xxxxxxxx@qq.com>', 'To': ' <xxxxxxxx@qq.com>', 'Subject': '使用python发送邮件的内容', 'Text': 'python发送邮件'}]
```

Parameters/参数说明

参数名称	参数类型	参数说明
email	string	email地址
password	string	登录密码
pop3_server	string	服务地址
length	int	读取几个邮件，从时间新的开始读，默认读取最新的

Return Value/返回信息

返回一个字典数组，每个字典代表一个邮件的内容

Remarks/备注

POP3和IMAP

POP是指邮局协议，目的是让用户可以访问邮箱服务器中的邮件，允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上,同时删除保存在邮件服务器上的邮件，而POP3服务器则是遵循POP3协议的接收邮件服务器，用来接收电子邮件的。后来又出现了IMAP协议(Interactive Mail Access Protocol),即交互式邮件访问协议，与POP3的不同在于：开启了IMAP后，在电子邮件客户端收取的邮件仍然保留在服务器上，同时在客户端上的操作都会反馈到服务器上，如：删除邮件，标记已读等，服务器上的邮件也会做相应的动作。

Related/相关

1.构造MIMEText对象时，第一个参数就是邮件正文，第二个参数是MIME的subtype，传入'plain'表示纯文本，最终的MIME就是'text/plain'，最后一定要用utf-8编码保证多语言兼容性。
2.用set_debuglevel(1)就可以打印出和SMTP服务器交互的所有信息。SMTP协议就是简单的文本命令和响应。login()方法用来登录SMTP服务器，sendmail()方法就是发邮件，由于可以一次发给多个人，所以传入一个list，邮件正文是一个str，as_string()把MIMEText对象变成str。

Example/实例

poplib的常用方法:

方法	描述
POP3(server)	实例化POP3对象，server是pop服务器地址
user(username)	发送用户名到服务器，等待服务器返回信息
pass_(password)	密码
stat()	返回邮箱的状态,返回2元祖(消息的数量,消息的总字节)
list([msgnum])	stat()的扩展，返回一个3元祖(返回信息, 消息列表, 消息的大小)，如果指定msgnum，就只返回指定消息的数据
retr(msgnum)	获取详细msgnum，设置为已读，返回3元组(返回信息, 消息msgnum的所以内容, 消息的字节数)，如果指定msgnum，就只返回指定消息的数据
dele(msgnum)	将指定消息标记为删除
quit()	登出，保存修改，解锁邮箱，结束连接，退出

Example POP3接收

```
from poplib import POP3
p = POP3('pop.163.com')
p.user('xxxxxxx@163.com')
p.pass_('xxxxxxxx')
p.stat()
...
p.quit()
```

python中的imaplib包支持IMAP4常用方法:

方法	描述
IMAP4(server)	与IMAP服务器建立连接
login(user, pass)	用户密码登录
list()	查看所有的文件夹(IMAP可以支持创建文件夹)
select()	选择文件夹默认是”INBOX”

search() 三个参数，第一的是CHARSET,通常为None(ASCII),第二个参数不知到是干什么官方没解释	

Example IMAOP4接收

```
import getpass, imaplib
M = imaplib.IMAP4()
M.login(getpass.getuser(), getpass.getpass())
M.select()
typ, data = M.search(None, 'ALL')
for num in data[0].split():
typ, data = M.fetch(num, '(RFC822)')
print 'Message %s\n%s\n' % (num, data[0][1])
M.close()
M.logout()
```

2.数据库

2.1MY SQL

```
conn = mysql.connector.connect(user='用户名', password='密码', database='数据库名')
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

```
暂无
```

Remarks/备注

1.执行INSERT等操作后要调用commit()提交事务；

2.MySQL的SQL占位符是%s。

Related/相关

暂无

Example/实例

Example MySQL 连接

```
# 导入MySQL驱动:
import mysql.connector

conn = mysql.connector.connect(user='用户名', password='用户密码', database='数据库名')
cursor = conn.cursor()
# 创建user表:
cursor.execute('create table user (id varchar(20) primary key, name varchar(20))')
# 插入一行记录，注意MySQL的占位符是%s:
cursor.execute('insert into user (id, name) values (%s, %s)', ['1', 'Michael'])
cursor.rowcount
# 提交事务:
conn.commit()
cursor.close()
# 运行查询:
cursor = conn.cursor()
cursor.execute('select * from user where id = %s', ('1',))
values = cursor.fetchall()
values
[('1', 'Michael')]
# 关闭Cursor和Connection:
cursor.close()
conn.close()
```

2.2SQL Server

```
conn = pymssql.connect(server, user, password, "连接默认数据库名称")  #获取连接
```

Parameters/参数说明

参数名称	参数类型	参数说明
server	string	IP地址
user	string	用户名称
password	string	密码
DbName	string	连接默认数据库名称

Return Value/返回信息

```
暂无
```

Remarks/备注

```
暂无
```

Related/相关

```
暂无
```

Example/实例

Example SqlServer 连接

```
import pymssql
```

```
server = "187.32.43.13" # 连接服务器地址
user = "root" # 连接帐号
password = "1234" # 连接密码

conn = pymssql.connect(server, user, password, "连接默认数据库名称") #获取连接

cursor = conn.cursor() # 获取光标

# 创建表
cursor.execute("""
IF OBJECT_ID('persons', 'U') IS NOT NULL
DROP TABLE persons
CREATE TABLE persons (
id INT NOT NULL,
name VARCHAR(100),
salesrep VARCHAR(100),
PRIMARY KEY(id)
)
""")

# 插入多行数据
cursor.executemany(
"INSERT INTO persons VALUES (%d, %s, %s)",
[(1, 'John Smith', 'John Doe'),
(2, 'Jane Doe', 'Joe Dog'),
(3, 'Mike T.', 'Sarah H.')])
# 你必须调用 commit() 来保持你数据的提交如果你没有将自动提交设置为true
conn.commit()

# 查询数据
cursor.execute('SELECT * FROM persons WHERE salesrep=%s', 'John Doe')

# 遍历数据（存放到元组中）方式1
row = cursor.fetchone()
while row:
print("ID=%d, Name=%s" % (row[0], row[1]))
row = cursor.fetchone()
# 遍历数据（存放到元组中）方式2
for row in cursor:
print('row = %r' % (row,))

# 关闭连接
conn.close()
```

Example 存储过程

```
import pymssql

server = "187.32.43.13" # 连接服务器地址
user = "root" # 连接帐号
password = "1234" # 连接密码

with pymssql.connect(server, user, password, "你的连接默认数据库名称") as conn:
with conn.cursor(as_dict=True) as cursor: # 数据存放到字典中
```

```
cursor.execute('SELECT * FROM persons WHERE salesrep=%s', 'John Doe')
for row in cursor:
print("ID=%d, Name=%s" % (row['id'], row['name']))
```

2.3Oracle

```
#方法一：用户名、密码和监听分开写
import cx_Oracle
db=cx_Oracle.connect('username/password@host/orcl')
db.close()

#方法二：用户名、密码和监听写在一起
import cx_Oracle
db=cx_Oracle.connect('username','password','host/orcl')
db.close()

#方法三：配置监听并连接
import cx_Oracle
tns=cx_Oracle.makedsn('host',1521,'orcl')
db=cx_Oracle.connect('username','password',tns)
db.close()
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

暂无

Remarks/备注

暂无

Related/相关

1.引用模块cx_Oracle
2.连接数据库
3.获取cursor
4.使用cursor进行各种操作
5.关闭cursor
6.关闭连接

Example/实例

Example Oracle 连接

```
import cx_Oracle                                    #引用模块cx_Oracle
conn=cx_Oracle.connect('load/123456@localhost/ora11g') #连接数据库
c=conn.cursor() #获取cursor
x=c.execute('select * from Rpa_Info') #使用cursor进行操作
x.fetchone()
c.close() #关闭cursor
conn.close() #关闭连接
```

Example Oracle 连接操作

```
import cx_Oracle
def Oracle_Connet():
db=cx_Oracle.connect('username/password@host/orcl')
cr=db.cursor()
Strsql='select * from Rpa_info'
rs=execute(Strsql)
#返回所有结果集
print('all:(%s)'%rs)
for x in rs
print (x)

#返回行
while(1):
rs=cr.fetchone()
if rs==None:break
db.close()

#参数
#字典
pr={'ID':'123'}
Strsql=Strsql + ' where ID=:ID'
cr.execute(Strsql,pr)
```

```
#传参
Strsql=Strsql + ' where ID=456'
cr.execute(Strsql,pr)
cr.close()
db.close()
pass
```

3.打印文档

3.1 打印图片

```
for y in range(0, height, scale):  #根据缩放长度 遍历高度
for x in range(0, width, scale): #根据缩放长度 遍历宽度
choice =gray_img.getpixel((x, y)) * char_len // 255 #获取每个点的灰度 根据不同的灰度填写相应的 替换字符
if choice==char_len:
choice=char_len-1
sys.stdout.write(char_lst[choice]) #写入控制台
sys.stdout.write('\n')
sys.stdout.flush()
```

Parameters/参数说明

参数名称	参数类型	参数说明
StrFile	string	文件路径

Return Value/返回信息

```
暂无
```

Remarks/备注

拿到图片对象，并转换图片模式（ ‘’ L ‘’ ）,L模式可以去到图片的个像素的灰度参数；
定义图片缩放长度、替换字符的串（根据灰度值排序）；

遍历缩放后每一个点位置，并获取该点位置的灰度值，根据灰度值替换为相应的替换字符;
打印在控制台;

Related/相关

利用 python 的 image模块和简单用符号，再控制台打印出图片的图片轮廓

PIL 安装 pip install Pillow

Example/实例

Example 打印图片

```
from PIL import Image
import sys
import os

def PrintImg(StrFile):
try:
pic = os.path.abspath(StrFile) #获取图片路径参数
except:
print(StrFile)
img = Image.open(pic) #获取图片对象
width = img.width #获取图片宽度
height = img.height #获取图片高度

gray_img = img.convert('L') #图片转换为'L'模式 模式“L”为灰色图像，它的每个像素用8个bit表示，0表示黑，255表示白，其他数字表示不同的灰度

scale = width // 100 #图片缩放100长度
char_lst = ' .:-=+*#%@' #要替换的字符
char_len = len(char_lst) #替换字符的长度

for y in range(0, height, scale): #根据缩放长度 遍历高度
for x in range(0, width, scale): #根据缩放长度 遍历宽度
choice =gray_img.getpixel((x, y)) * char_len // 255 #获取每个点的灰度 根据不同的灰度填写相应的 替换字符
if choice==char_len:
choice=char_len-1
sys.stdout.write(char_lst[choice]) #写入控制台
sys.stdout.write('\n')
sys.stdout.flush()
pass
```

3.2 打印ExcelWord文档

```
#读取docx中的文本代码示例
import docx
#获取文档对象
file=docx.Document("E:\\test.docx")
print("段落数:"+str(len(file.paragraphs)))#段落数为13，每个回车隔离一段

#输出每一段的内容
for para in file.paragraphs:
print(para.text)

#输出段落编号及段落内容
for i in range(len(file.paragraphs)):
print("第"+str(i)+"段的内容是："+file.paragraphs[i].text)
```

Parameters/参数说明

参数名称	参数类型	参数说明
StrFile	string	docx 文件路径

Return Value/返回信息

暂无

Remarks/备注

python无法读取.doc文件

问题分析：python利用python-docx (0.8.6)库可以读取.docx文件或.txt文件，且一路畅通无阻，而对.doc文件本身python是无法操作.doc文件是他的先天不足，将.doc文档利用手动的方式“另存为”.docx文档，就能够成功打开转化后的.docx文档，于是我就尝试利用代码方式完成这个手动的“另存为”功能，问题得以解决。

利用python将大批.doc文件转化为.docx文件，再读写.docx文件

Related/相关

暂无

Example/实例

Example 打印word

```
import sys
import pickle
import re
import codecs
import string
import shutil
from win32com import client as wc
import docx

def start():
# 在此处开始编写您的应用
pass

def PrintWord(StrFile):
#获取文档对象
file=docx.Document(StrFile)
print("段落数:"+str(len(file.paragraphs)))#每个回车隔离一段

#输出每一段的内容
for para in file.paragraphs:
print(para.text)

#输出段落编号及段落内容
for i in range(len(file.paragraphs)):
print("第"+str(i)+"段的内容是：+file.paragraphs[i].text)
pass

def doSaveAas():
word = wc.Dispatch('Word.Application')
doc = word.Documents.Open(u'E:\\code\\test.doc') # 目标路径下的文件

doc.SaveAs(u'E:\\test.docx', 12, False, "", True, "", False, False, False, False) # 转化后路径下的文件
doc.Close()
word.Quit()
pass
```

打印Excel文档

```
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
```

Parameters/参数说明

参数名称	参数类型	参数说明
file	string	docx 文件路径
colnameindex	int	行号
by_index	int	索引

Return Value/返回信息

暂无

Remarks/备注

暂无

Related/相关

确保安装好excle的处理包xlrd

安装过程如下：http://blog.csdn.net/weixin_40449300/article/details/79138741

Example/实例

Example 打印Excel

```
import  xdrlib ,sys
import xlrd
def open_excel(file= 'file.xls'):
try:
data = xlrd.open_workbook(file)
return data
except Exception as e:
print(str(e))
pass
#根据索引获取Excel表格中的数据 参数:file：Excel文件路径 colnameindex：表头列名所在行的所以 ，by_index：表的索引
def excel_table_byindex(file= 'E:\\test.xls',colnameindex=0,by_index=0):
data = open_excel(file)
table = data.sheets()[by_index]
nrows = table.nrows #行数
ncols = table.ncols #列数
colnames = table.row_values(colnameindex) #某一行数据
list =[]
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
pass

#根据名称获取Excel表格中的数据 参数:file：Excel文件路径 colnameindex：表头列名所在行的所以 ，by_name：Sheet1名
称
def excel_table_byname(file= 'E:\\test.xls',colnameindex=0,by_name=u'电度'):
data = open_excel(file)
table = data.sheet_by_name(by_name)
nrows = table.nrows #行数
colnames = table.row_values(colnameindex) #某一行数据
list =[]
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
pass

def main():
tables = excel_table_byindex()
for row in tables:
print(row)

tables = excel_table_byname()
for row in tables:
print(row)
```

```
pass
```

4.网络

4.1 HTTP请求

HTTP是在网络上传输HTML的协议，用于浏览器和服务器的通信。

```
import urllib2
response = urllib2.urlopen('http://localhost:8080/jenkins/api/json?pretty=true')
print response.read()
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

响应代码：
200表示成功，
300表示重定向，
400表示客户端发送的请求有错误，
500表示服务器端处理时发生了错误；

响应类型：由Content-Type指定；

以及其他相关的Header；

通常服务器的HTTP响应会携带内容，也就是有一个Body，包含响应的内容，网页的HTML源码就在Body中。

Remarks/备注

暂无

Related/相关

暂无

Example/实例

Example urllib2 Get请求

```
import urllib2
import urllib
post_data = urllib.urlencode({})
response = urllib2.urlopen('http://localhost:8080/, post_data)
print response.read()
print response.getheaders()
```

Example httplib Get请求

```
import httplib
conn = httplib.HTTPConnection("www.python.org")
conn.request("GET", "/index.html")
r1 = conn.getresponse()
print r1.status, r1.reason
data1 = r1.read()
conn.request("GET", "/parrot.spam")
r2 = conn.getresponse()
data2 = r2.read()
conn.close()
```

Example httplib Post请求

```
import httplib, urllib
params = urllib.urlencode({'@number': 12524, '@type': 'issue', '@action': 'show'})
headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/plain"}

conn = httplib.HTTPConnection("bugs.python.org")
conn.request("POST", "", params, headers)
response = conn.getresponse()
data = response.read()
print data
```

```
conn.close()
```

Example requests Get请求

```
print requests.get('http://localhost:8080).text
```

Example requests Post请求

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.post("http://httpbin.org/post", data=payload)
print r.text
```

4.2 创建HTTP服务

对于Python 2，简单搭建Web服务器，只需在需要搭建Web服务器的目录（如C:/ 或 /home/klchang/）下，输入如下命令：

```
#使用 8080端口的创建Web服务器，可以使用浏览器 http://127.0.0.1:8080 或 将 127.0.0.1 替换为本机的外部IP地址，进行访问。
python -m SimpleHTTPServer 8080
```

对于Python 3，其创建Web服务器的等价命令，如下：

```
python3 -m http.server 8080
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

```
暂无
```

Remarks/备注

暂无

Related/相关

建立服务

url : HTTP服务 url，可以是ip地址也可以是域名等等，如果不传则默认本机ip地址

port : HTTP服务端口号，注意不要填写已被占用端口号，默认8901

recvNum : HTTP服务最大连接数，超过此连接数则无法请求成功 ,默认20

logLevel : HTTP服务后台日志级别 ，1:debug,2:info,3:warning,4:error

Example/实例

Example 创建HTTP服务

```
python3 -m http.server 8080
```

4.3 WebSocket

WebSocket 是一种标准协议，用于在客户端和服务端之间进行双向数据传输。但它跟 HTTP 没什么关系，它是基于 TCP 的一种独立实现。

```
#启动Socket并监听连接
sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

暂无

Remarks/备注

暂无

Related/相关内容

暂无

Example/实例

Example WebSocket

```
def Create_Socket():
#启动Socket并监听连接
sock=socket.socket(socket.AF_INFT,socket.SOCK_STREAM)
try:
sock.bind('127.0.0.1',8001)
#操作系统会在服务器Socket 被关闭或服务器进程终止后马上释放该服务器端口
sock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
sock.listen(5)
except Exception as e:
logging.error(e)
return
else:
loging.info('Server running...')

#等待访问
while True:
conn , addr = sock.accept() #进入waiting 状态
data=str(conn.recv(1024))
logging.debug(data)
header_dict={}
header, _ =data.split(r'\r\n\r\n',1)
for line in header.split(r'\r\n')[1:]:
key , val = line.split(':',1)
header_dict[key]=val
```

```
if 'Sec-WebSocket-Key' not in header_dict:
logging.error('This socket is not websocket,client close.')
conn.close()
return

magic_ket=''
sec_key=header_dict['Sec-WebSocket-Key']+magic_ket
key=base64.b64encode(hashlib.shal(bytes(sec_key,encodeing='utf-8')).digest())
logging.debug(key_str)

response=''
conn.send(bytes(response,encoding='utf-8'))
WebSocketThread(conn).start()
pass

def read_msg(data):
mes_len=data[1]&127 # 数据载荷长度
if msg_len==126:
mask =data[4:8] #mask掩码
content =data[8:] #消息内容
elif msg_len==127:
mask =data[10:14]
content =data[14:]
else:
mask=data[2:6]
content=data[6:]
raw_str= '' #解码后的内容
for i,d in enumerate(content):
raw_str+=chr(d ^ mask[i%4])
return raw_str
pass

def write_msg(message):
data = struct.pack('B',129)

# 写入包长度
msg_len=len(message)
if msg_len<=125:
data+=stryct.pack('B',msg_len)
elif msg_len <= (2 ** 16-1):
data += struct.pack('!BH',126,meg_len)
elif msg_len <= (2 ** 64-1):
data += struct.pack('!BQ',126,meg_len)
else:
return
data += bytes(message,encoding='utf-8')
return data
pass
```

5.切换输入法

```
# 设置键盘布局为英文
result = win32api.SendMessage(
hwnd,
WM_INPUTLANGCHANGEREQUEST,
0,
0x0409)
```

Example/实例

Example 切换输入法

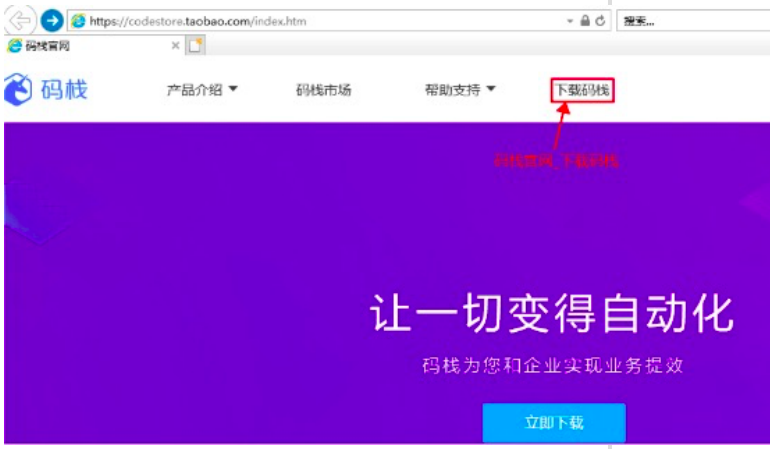
```
from rpa.core import *
from rpa.utils import *
import rpa
from win32con import WM_INPUTLANGCHANGEREQUEST
import win32gui
import win32api

def start():
# 在此处开始编写您的应用
# 语言代码
LID = {0x0804: "Chinese (Simplified) (People's Republic of China)",
0x0409: 'English (United States)'}

# 获取前景窗口句柄
hwnd = win32gui.GetForegroundWindow()

# 获取前景窗口标题
title = win32gui.GetWindowText(hwnd)
print('当前窗口：' + title)

# 获取键盘布局列表
im_list = win32api.GetKeyboardLayoutList()
im_list = list(map(hex, im_list))
print(im_list)

# 设置键盘布局为英文
result = win32api.SendMessage(
hwnd,
WM_INPUTLANGCHANGEREQUEST,
0,
0x0409)
if result == 0:
print('设置英文键盘成功！')

pass
```

6.IE上传下载

6.1 IE download

Example IE9 IE10 IE11

版本	网页控件录制图示
IE9	
IE10	

	 码栈 产品介绍 码栈市场 帮助支持 下载码栈 码栈官网_下载码栈 多个版本，量体裁衣 节省您的每一分投资 下载码栈客户端 码栈官网_下载码栈客户端 码栈官网_保存下拉 Do you want to open or save CodeStoreSetup (75.9 MB) from juwuba-version.oss-cn-hangzhou.aliyuncs.com? Open Save Save and open
IE11	 https://codestore.taobao.com/index.htm 码栈官网 码栈 产品介绍 码栈市场 帮助支持 下载码栈 让一切变得自动化 码栈为您和企业实现业务提效 立即下载

import rpa
import os
def upload*download():*
page =
rpa.ie.create('https://codestore.taobao.com/index.htm ')
*page.click("码栈官网*下载码栈
"*,*simulate=True)
page.wait('码栈官网*下载码栈客户端') #等待控件加载*
*page.click("码栈官网*下载码栈客户端") #通过点击代替直接使用download
rpa.win32.wait("码栈官网*保存下拉",
timeout=20) #wait方法暂时只支持桌面控件录制(图像)录制的控件*
*rpa.win32.click("码栈官网*保存下拉")
rpa.win32.click("码栈官网_另存为") #自定义存储路径时需要录制保存下拉框内的另存为控件
rpa.win32.save_file_dialog(r' C:\test\codestore.exe' ,title = '另存为')
sleep(3)
page.close()	

Example IE7 IE8

版本	网页控件录制图示	
IE7(不支持)		

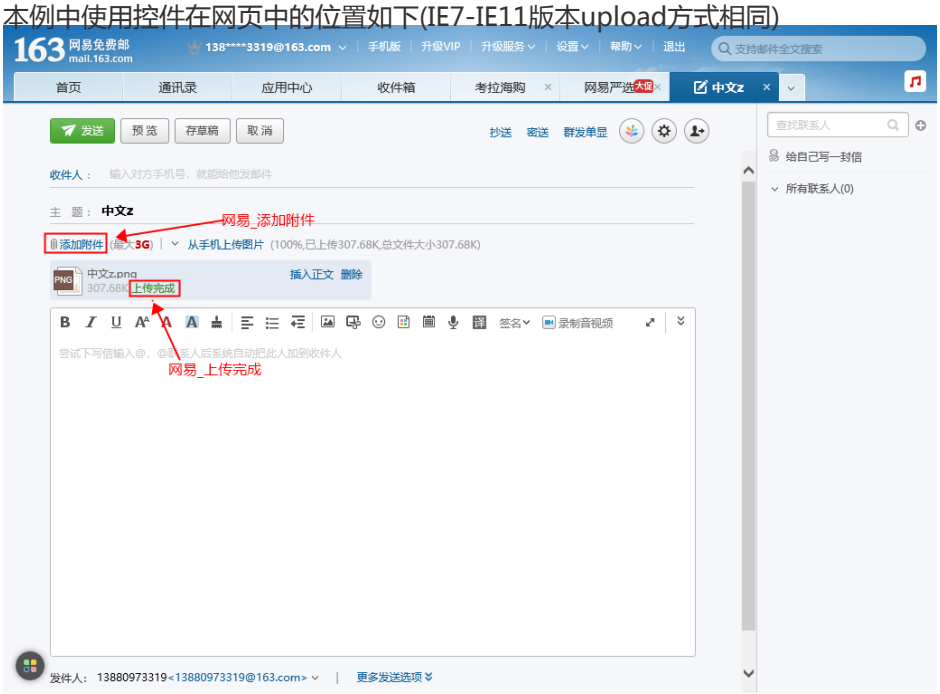
		
IE8		
import rpa import os def upload*download():* *page =* *rpa.ie.create(‘https://codestore.taobao.com/index.htm ‘)* *page.click(“码栈官网*下载码栈”,simulate=True) page.wait(‘码栈官网*下载码栈客户端’) #等待控件加载* *page.click(“码栈官网*下载码栈客户端”) #通过点击代替直接使用download rpa.win32.wait(“码栈官网*保存”, timeout=20) #wait方法暂时只支持桌面控件录制(图像)录制的控件* *rpa.win32.click(“码栈官网*保存”) rpa.win32.save_file_dialog(r’C:\test\codestore.exe’,title = ‘另存为’) sleep(3) page.close()		

#

6.2 IE upload

Example

本例中使用控件在网页中的位置如下(IE7-IE11版本upload方式相同)



请先手动登录163官网,并进入写信界面后再使用本例代码操作

```
import rpa
def ie11upload():
page = rpa.ie.catch('网易邮箱6.0版',mode = 'title') #catch用户手动打开的写信界面
page.upload("网易_添加附件", r"C:\test\中文z.png") #上传图片到附件
sleep(3) #上传过程等待
page.wait("网易_上传完成",timeout=30) #等待上传操作后,上传完成控件加载
text = page.value("网易_上传完成") #获取上传结果,显示是否上传成功
print(text)
if text == str("上传完成"):
print('upload上传成功！')
else:
print('upload上传失败！')
sleep(5)

page.close()
```

7.使用Python激活窗口

方法如下，可以通过修改match方法来改变匹配方式

```
from rpa.core import *
from rpa.utils import *
import rpa
import win32gui, win32con

def start():
# 在此处开始编写您的应用
win_active('记事本')

def win_active(winTitle):
def match(winText):
"""匹配方式：模糊匹配"""
return winText.find(winTitle) > -1
hwnds = match_windows(match)
if hwnds:
win32gui.ShowWindow(hwnds[0], win32con.SW_SHOWMAXIMIZED) # SW_SHOWDEFAULT 默认大小
，SW_SHOWMAXIMIZED 最大化显示
win32gui.SetForegroundWindow(hwnds[0])
win32gui.SetActiveWindow(hwnds[0])

def match_windows(match_func):
def callback(hwnd, hwnds):
if win32gui.IsWindowVisible(hwnd) and win32gui.IsWindowEnabled(hwnd):
winText = win32gui.GetWindowText(hwnd)
if match_func(winText):
hwnds.append(hwnd)
return True
hwnds = []
win32gui.EnumWindows(callback, hwnds)
return hwnds
```

9. 透视表相关操作

注: 客户端版本需要在3.3.26及以上

创建数据视图

```
xls = rpa.excel.open(r'C:\data\案例数据.xlsx')
sheet = xls.get_sheet('口碑汇总数据')
pivot_settings = rpa.excel.PivotTableSettings('MyPivotTable')
pivot_settings.columns['税额'] = {} # 添加"列标签"
pivot_settings.filters['不含税金额'] = {} # 添加"筛选字段"
pivot_settings.rows['税率'] = {} # 添加"行标签"
pivot_settings.values['税额2'] = {"Function": "xlAverage"} # 添加"数值"
pivot_settings.values['购方纳税人识别号'] = {"Function": "xlSum"} # 添加"数值"
sheet.create_pivottable('口碑汇总数据', 'G1:J4952', 'Sanco', 'A1', pivot_settings)
xls.save()
```

Function取值范围参考https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.office.interop.excel.xlconsolidationfunction?redirectedfrom=MSDN&view=excel-pia

刷新数据视图

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
sheet = xls.get_sheet('Sanco')
sheet.refresh_pivottable(index=1)
xls.save()
```

获取透视表筛选列的所有项

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
sheet = xls.get_sheet('Sanco')
items = sheet.get_all_pivot_field_items('不含税金额')
print(items)
```

选择/取消选择一组透视表筛选项

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
sheet = xls.get_sheet('Sanco')
sheet.select_pivot_field_items('不含税金额', [0.02, 0.03, 0.07], select=False)
xls.save()
```

9. 持久化应用日志

本功能要求客户端版本必须为3.3.31及以上
rpa.logger提供了use_logger方法，用来设置自定义的日志记录程序，结合python的*logging*库即可实现日志持久化的功能。
示例代码:

```
from rpa.core import *
from rpa.utils import *
import rpa
import os
import logging
from logging.handlers import TimedRotatingFileHandler

def use_my_logger():
""" 设置自定义的日志记录程序
使用TimedRotatingFileHandler按日期循环记录日志，可替换为任意handler
"""
logger = logging.getLogger('mylogger')
logger.setLevel(logging.DEBUG) # 设置根日志级别
log_file = os.path.join(os.getenv('APPDATA'), r'AliRPA\log\myapp.log') # 设置日志文件存放位置
(%appdata%\AliRPA\log\app\myapp.log)
formatter = logging.Formatter('%(asctime)s[%(name)s][%(levelname)s] %(message)s')
handler = TimedRotatingFileHandler(log_file, when='D', interval=1)
handler.setFormatter(formatter)
handler.suffix = '%Y%m%d'
handler.setLevel(logging.DEBUG) # 设置当前handler日志级别
logger.addHandler(handler)
rpa.logger.use_logger(logger) # 替换为自定义的logger

def start():
# 在此处开始编写您的应用
use_my_logger()
print('message 1')
rpa.logger.info('info-message')
rpa.logger.warn('warn-message')
rpa.logger.error('error-message')
```