安装Python

安装Sublime Text

Python文件的新建和打开方式

安装完Sublime Text，把代码编译器发送至桌面，方便使用

注意：打开代码编译器直接写代码，写完保存再运行

- 修改文件的默认打开方式后再打开
- 新建文本文档，修改后缀名
- 直接拖动

命令提示符的使用

模块的介绍

基本介绍

我们可以把**模块**简单理解为**Python文件。**

不同的模块具有不同的功能，所以当我们用Python写代码的时候可能会需要导入不同的模块，也就是在当前的Python文件中导入另一个Python文件。

导入模块

格式：import 模块名

```
import time
```

使用模块的具体方法

格式：**模块名.方法名** 或 **子模块名.方法名**

> 模块就是Python中常说的库，方法就是Python中常说的函数，只是叫法不同而已。

```
import time

time.sleep(10)
print(123456789)
```

给导入的模块设置别名

格式：import 模块名 as 别名

```
import time as tm

tm.sleep(10)
print(123456789)
```

导入模块下的子模块

格式：from 模块名 import 子模块名

```
from time import sleep

sleep(10)
print(123456789)
```

模块种类

模块分为：

- 内置模块：Python自带的模块，不需要额外下载安装，直接通过import导入即可。
- 第三方模块：不属于Python自带的模块，需要额外下载安装后，才能通过import导入。

> 是由第三方使用Python开发，需要下载安装才能使用的工具包。
>
> 这里的第三方是指来自各行各业使用Python的开发人员，他们为不同行业的不同业务提供了解决方案。

如何下载安装第三方模块?

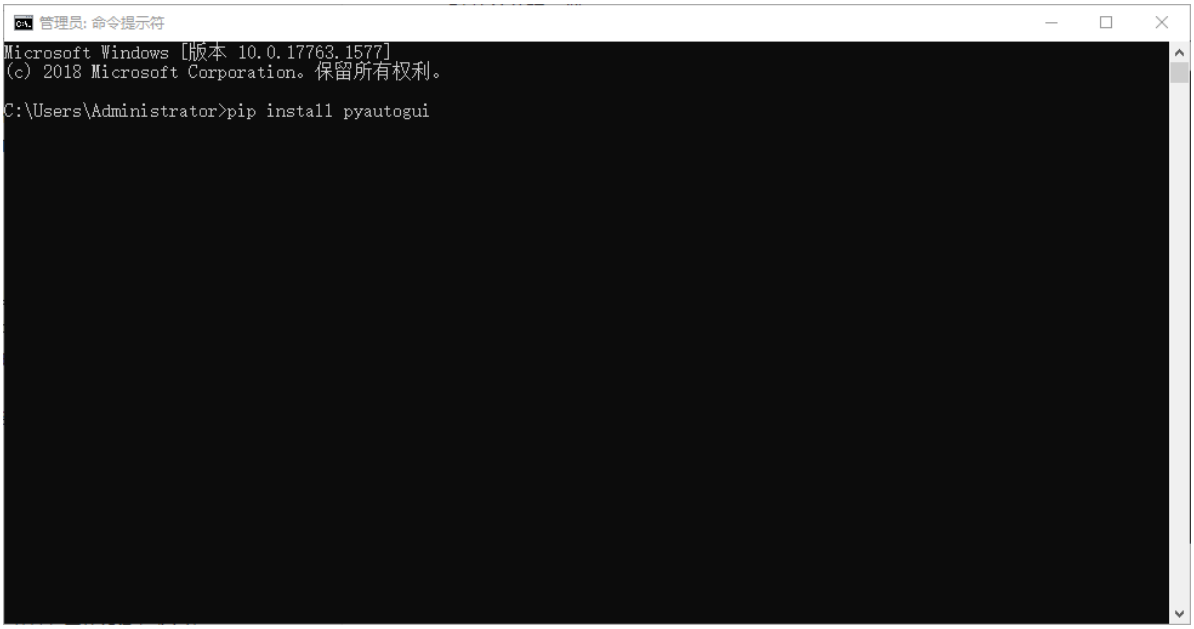
假如我现在想通过Python实现自动化控制鼠标和键盘的操作，我们就可以下载pyautogui这个第三方模块库。

下载安装

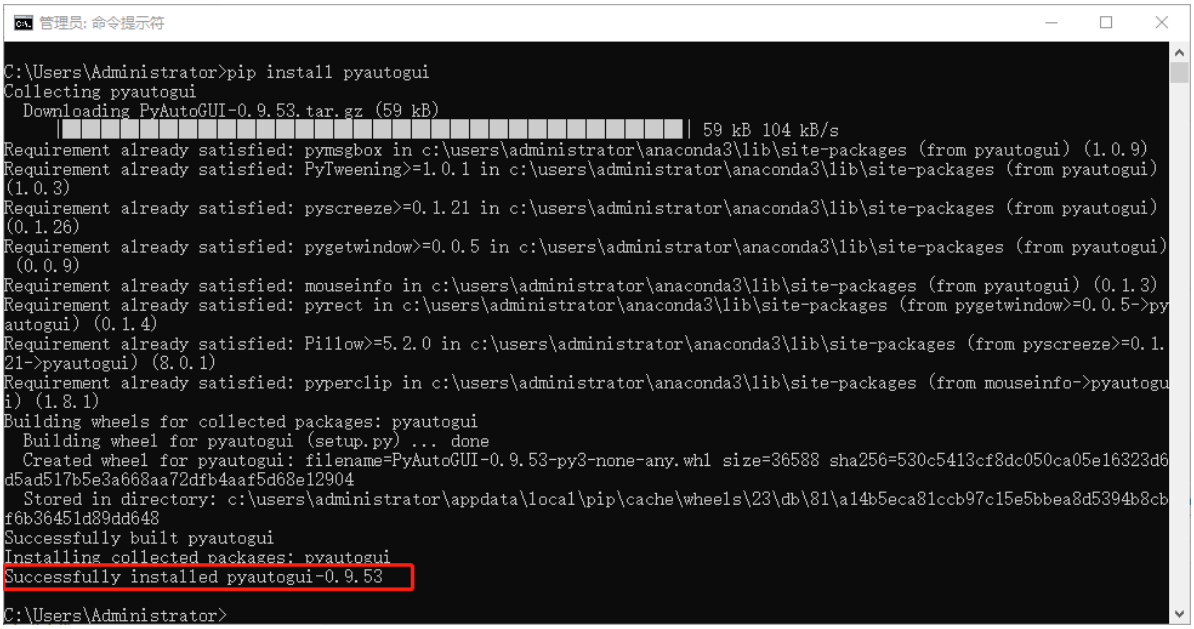
打开命令提示符

输入： `pip install` +第三方模块的名称，然后点击回车键即可进行下载安装。

在这里我们输入：`pip install pyautogui`



点击回车键进行下载安装，当出现如下红框内的**Successfully installed ……**就说明已经下载安装成功了。



下面我们尝试使用这个模块执行自动移动鼠标的操作：

```
import pyautogui

pyautogui.moveTo(1255, 480)
```

第三方模块下载安装失败的解决办法

情况1

我们在我们安装某个第三方模块的时候，会遇到**下载速度很慢**、或者**下载失败**的情况，这是因为pip命令默认使用的是国外的pypi镜像（pypi.python.org），安装慢不说，有时甚至会导致出现超时等网络问题，造成安装失败。所以，建议使用国内的pypi镜像加速安装第三方模块。国内的第三方镜像源有很多，如下所示：

- **阿里云镜像源**：http://mirrors.aliyun.com/pypi/simple/
- **中国科技大学镜像源**： https://pypi.mirrors.ustc.edu.cn/simple/
- **豆瓣镜像源**： http://pypi.douban.com/simple/
- **清华大学镜像源**：https://pypi.tuna.tsinghua.edu.cn/simple/
- **中国科学技术大学镜像源**：http://pypi.mirrors.ustc.edu.cn/simple/

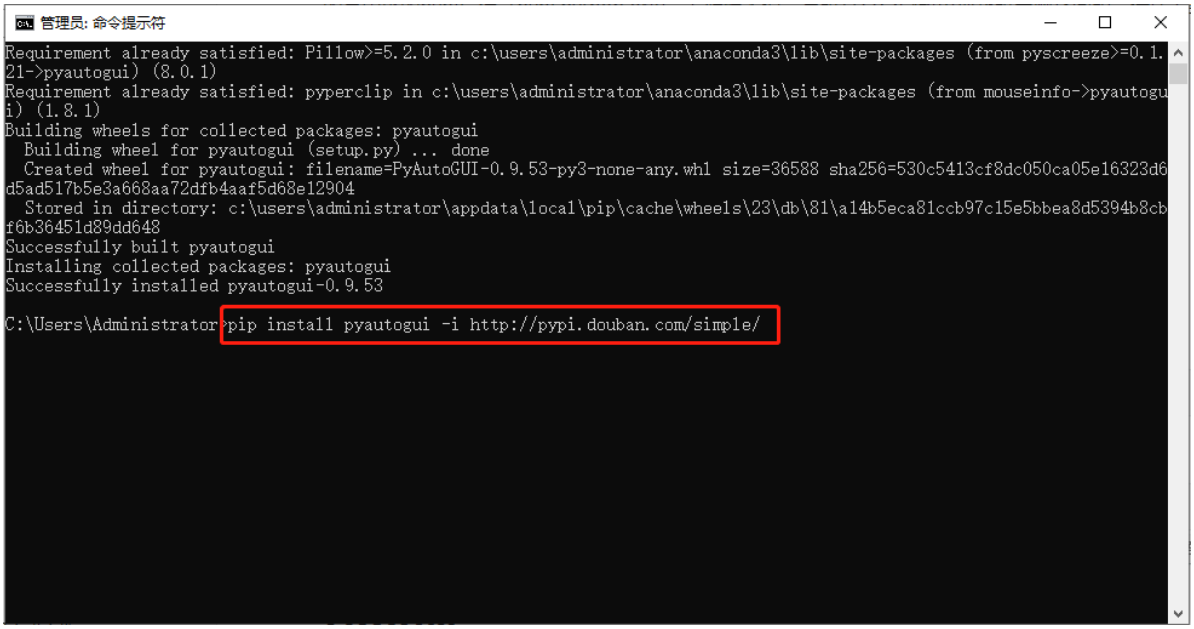
怎么使用国内的第三方镜像源呢?

只需把国内的第三方镜像源复制粘贴在最后即可：

```
pip install 第三方模块的名称 -i 国内的第三方镜像源
```

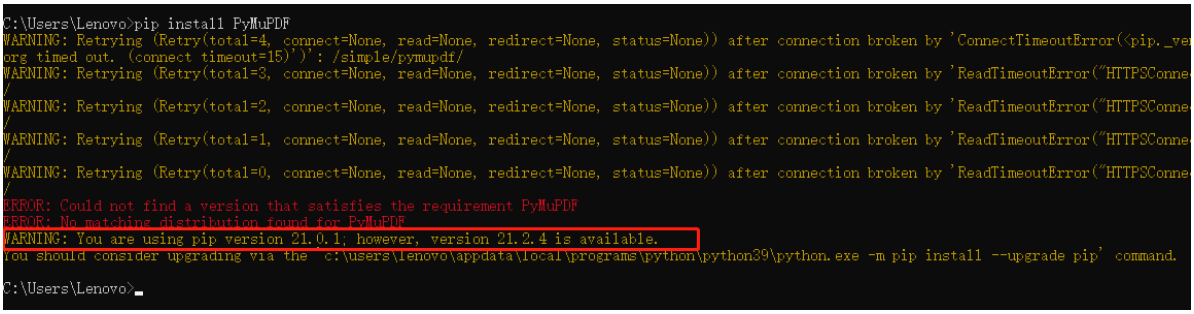
例如，用**豆瓣镜像源**下载安装pyautogui库：

```
pip install pyautogui -i http://pypi.douban.com/simple/
```



情况2

我们安装某个第三方模块的时候，如果出现如下红框内的提示，就需要**先对pip进行升级**，pip升级完成后再重新进行下载安装就不会报错了。



打开**命令提示符**运行以下代码，进行pip下载安装升级：

```
python -m pip install --upgrade pip
```

注释

- 注释是对代码的解释和说明
- 注释不是代码的一部分，不会被执行
- 让代码更具有可读性

单行注释

在文字或者代码的最左段加上`#`键，就可以完成注释。

注释的快捷键：`Ctrl`+`?`键

我们可以选中需要注释的文字或者代码，通过快捷键进行快速注释。

实例

```
#导入模块
import pyautogui
import time

#延迟10秒执行
time.sleep(10)

#移动鼠标光标
pyautogui.moveTo(1255, 480)
time.sleep(2)
pyautogui.moveTo(200, 200)
time.sleep(2)
pyautogui.moveTo(600, 480)
```

多行注释

在多行内容的前后两行分别加上三个单引号`'''`，即可注释多行内容。

```
#导入模块
import pyautogui
import time

#延迟10秒执行
time.sleep(10)

#移动鼠标光标
pyautogui.moveTo(1255, 480)
time.sleep(2)
'''
pyautogui.moveTo(200, 200)
time.sleep(2)
pyautogui.moveTo(600, 480)
'''
```

常量与变量

变量使用前需要声明

格式：变量名=变量值

变量的三要素：**变量名，变量值，变量的数据类型**

```
#常量    '张三'  18
print('张三')
print(18)

#声明变量
name='张三'
age=18

#使用变量，通过变量名获取变量值
print(name)
print(age)
print(name,age)
print('我的好朋友是',name)
```

数据类型

基本介绍

数据类型：**整型、浮点型、字符串、布尔值、空**

- 整型（int）：表示整数，例如`-35、0、8、66`等等
- 浮点型（float）:表示小数，例如`3.14、5.867`等等
- 字符串（str）：表示一串字符，定义时需要用单引号`' '`或`" "`双引号括起来，比如`'abc'、'张三'`等等
- 布尔值（bool）：表示真假的判断，一个布尔值只能`True`或`False`
- 空（None）：表示什么也没有，None不能理解为0，因为0是有意义的，而None是一个特殊的空值

```
#整型、浮点型属于数值型，可以参与运算
#整型（int）
a=1
b=2
print(a+b)

#浮点型（float）
c=5.5
d=2.1
print(c+d)

#字符串（str）
s1='财务'
s2='Python'
s3='62.75' #数值形式的字符串
s4='' #空字符串
print(s1,s2)
```

```
print('我们要学习',s1,s2)

#布尔值（bool）
e=True
f=False
print(10>1)

#查看数据类型type()
print(type(s1))
```

字符串的拼接

```
#用+拼接  需保证每一个字符串都必须是字符串类型
a='财务'
b='Python'
c=a+b
print(c)
print(a+b) #把字符串拼接在一起进行打印
print(a,b) #分别打印多个字符

#用format拼接
d='财务{}'.format('Python')
d='{}财务{}'.format('你好','Python')
print(d)
```

转义字符

字符串中具有特殊意义的字符

```
#转义字符
a='财务Python'
a='财务\nPython'
print(a)

#取消转义  文件路径  网址
b='D:\Program Files\Wind\Wind.NET.Client\WindNET'
b=r'D:\Program Files\Wind\Wind.NET.Client\WindNET'
b='D:\\Program Files\\Wind\\Wind.NET.Client\\WindNET'
print(b)
```

数据类型的转换

```
#数值型的字符串转换为整型
a='500'
b=int(a)+100
print(b)

#数值型的字符串转换为浮点型
a='5.67'
b=int(a)+100
print(b)

#浮点型数值转换为整型
a='5.67'
```

```
b=int(a)
print(b)

#整型数值转换为浮点型
a='500'
b=int(a)
print(b)

#整型、浮点型数值转换为字符串
a=500
b=str(a)
print(b.type)
print('我的成绩是'+b)
```

运算符

运算符是Python语言中各种表达式中的符号。

- 算术运算符
- 赋值运算符
- 比较（关系）运算符
- 逻辑运算符

算术运算符：数学运算中的符号

以下假设变量： **a=10，b=20**：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分（**向下取整**）	`>>> 9//2 4 >>> -9//2 -5`

```
#算术运算符
a=10
b=3

print(a/b) #除法
print(a//b) #除法，只保留整数部分
print(a%b) #余数
print(a**b) #幂次方
```

赋值运算符：给变量赋值

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

```
#赋值运算符
a=10
a+=5
print(a)
```

比较（关系）运算符：比较两个变量之间的关系

以下假设变量a为10，变量b为20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等。**python3 已废弃。**	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。	(a < b) 返回 true。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 true。

```
#比较（关系）运算符
a=10
b=5
print(a>b)
print(a<b)
print(a!=b)
print(a==b) #=表示赋值，==表示判断
```

逻辑运算符：判断表达式之间的逻辑关系

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的计算值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

```
#逻辑运算符
print(10>1 and 5>2)
print(10>1 or 5>2)
print(not 5>2)
```

列表(list)

- 列表是一个有序的序列结构，列表中的每一个元素都有一个索引
- 列表可以存放不同数据类型的数据
- 列表中的元素可以重复
- 列表可以进行元素的添加、修改、删除

```
#定义一个列表
name=['资产',100,'负债',50,'收入',100,'利润',50]
#       0    1    2    3    4    5    6    7
#      -8   -7   -6   -5   -4   -3   -2   -1

#通过索引取值
print(name[2])
print(name[-6])

#切片
print(name[2:3])
print(name[2:7:2])
print(name[2:])
print(name[:5])

#增加列表元素
name.append('净利润')
```

```
name.insert(2,'净资产')
print(name)

#删除列表元素
name.remove('资产')
name.pop(2)
print(name)

#修改列表元素
name=['资产',100,'负债',50,'收入',100,'利润',50]
name[2]='净资产'
print(name)

#判断一个元素是否存在
name=['资产',100,'负债',50,'收入',100,'利润',50]
print('资产' in name)
```

字典(dict)

字典是一种键值对结构的序列结构，其中的键(key)和值(value)是一一对应的。

- 无序，没有索引
- 键值对形式，通过键获取值
- 键不可以重复
- 一般使用字符串组为键

```
name={'资产':100,'负债':50,'收入':100,'利润':50}

#使用键获取值
print(name['负债'])
print(name['利润'])

#增加数据
name['净利润']=50
print(name)

#删除数据
name.pop('负债')
print(name)

#判断一个键是否存在
print('资产' in name)
```

len()函数

len()函数：获取序列长度

```
#定义一个列表
name=['资产',100,'负债',50,'收入',100,'利润',50]
#获取name序列长度
print(len(name))

word='你好！我是注册会计师'
print(len(word))
print(word[-2])
```

If判断语句

```
#单个条件的判断
a=60
if a>=60:
    print('恭喜你！及格了！')
else: #else可以省略
    print('继续努力！')

#多个条件的判断
a=60
if a>=90:
    print('优异！')
elif a>=80:
    print('良好！')
elif a>=60:
    print('及格！')
else:
    print('继续努力！')
```

for循环语句

```
'''
for 变量名 in 序列:
    循环语句
'''
#执行次数等于序列中元素的个数
for i in range(0,10):
    print('第',i,'次打印')

#设置步长
for i in range(0,10,3):
    print('第',i,'次打印')

#循环遍历列表
name=['资产','负债','收入','利润']
for i in name:
    print(i)

#根据索引循环遍历列表
name=['资产','负债','收入','利润']
for i in range(0,len(name)):
    print(name[i])

#循环遍历列表求和
```

```
name=[10,20,30,40]
sum=0
for i in name:
    print(i)
    sum=sum+i
print(sum)

#循环遍历字典
name={'资产':100,'负债':50,'收入':100,'利润':50}
for i in name:
    print(i)
    print(name[i])
```

捕捉异常语句

什么是异常?

异常即是一个事件，该事件会在程序执行过程中发生，影响了程序的正常执行。

一般情况下，在Python无法正常处理程序时就会发生一个异常。

异常是Python对象，表示一个错误。

当Python脚本发生异常时我们需要捕捉处理它，否则程序会终止执行。

异常处理

捕捉异常可以使用try/except语句。

try/except语句用来检测try语句块中的错误，从而让except语句捕捉异常信息并处理。

如果你不想在异常发生时结束你的程序，只需在try里捕捉它。

```
#检测语句一是否存在错误
try:
    语句一

#如果语句一存在错误，可捕捉错误，然后执行语句二
except 异常名称:
    语句二

#无论是否存在错误，都会执行语句三
finally:
    语句三
```

```
#检测语句一是否存在错误
try:
    age=18
    name='我今年'+age
    print(name)

#如果语句一存在错误，可捕捉错误
#Exception代表常规错误的基类
except Exception as e:
    print('代码异常！')
    print(e)
```

```
#无论是否存在错误，都会执行语句三
finally:
    print('运行结束！')
```

代码运行报错的解决办法

常见报错：

- 注意缩进
- 注意字母大小写
- 注意中英文输入法切换（括号，引号，冒号）
- 其他……等等

提示：首先要明白报错的原因，然后再分析报错的原因

```
age=18
name='我今年'+str(age)
print(name)

name=[10,20,30,40]
for i in name:
    print(i)
```

爬虫基本原理

用Python代码**请求网页地址**，根据**请求网页地址后返回的内容**，进行**内容（数据或者文本）的提取。**

可以分解为如下三个步骤：

- **请求网页地址**
- **请求网页地址后返回的内容**
- **内容（数据或者文本）的提取**

浏览器开发者工具

当我们用Python爬取不同的网页时，需要对不同的网页进行分析，分析网页时就需要用到浏览器的开发者工具。在这里，我以谷歌（Chrome ）浏览器为例讲解一下Chrome 浏览器开发者工具。

浏览器开发者工具的打开方式

- 在网页页面右击鼠标，选择“检查”，可打开Chrome开发者工具。
- 在网页页面直接按“F12”，可打开Chrome开发者工具。

元素面板（Element）

通过元素（Element）面板，我们能查看到想抓取页面渲染内容所在的标签等内容。例如我想要爬取天眼查网页中的工商信息，可以打开Chrome开发者工具，进入元素面板。

通过这种方法，我们能快速定位到页面中的某个元素，然后可以提取出相关的解析语句。鼠标移动到该元素，然后右击鼠标，选择 “Copy”，能快速复制出Xpath等内容解析库的解析语句。

网络面板（Network）

网络（Network）面板记录页面上每个网络操作的相关信息，包括请求头信息、请求返回的结果等。

展示窗口会列出检索的每一个 HTTP 请求。 默认情况下，此窗口按时间顺序排序，最早的资源在顶部。点击资源的名称可以显示更多信息。

- Headers：显示 HTTP 请求的 Headers，我们通过这个能看到请求的方式，以及携带的请求参数等。
- Preview ： 请求结果的预览。一般用来查看请求到的图片，对于抓取图片网站比较给力。
- Response：请求返回的结果。一般结果是整个网站的源代码。如果该请求是异步请求，返回的结果内容一般是 Json 文本数据。
- Cookies：能看到请求携带的 Cookies 以及服务器返回的 Cookies。

静态网页与动态网页

网页的表现形式分为两种：静态网页、动态网页

静态网页：传统网页，不使用Ajax，如果需要更新内容，必须重新加载整个网页页面；传输数据格式方面，使用的是xml语法。

动态网页：使用Ajax，可以使网页实现异步更新，这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新；数据交互基本上都是使用json。

get请求与post请求

对网页的请求方式分为两种：get请求与post请求。

get请求

- get请求是从服务器上获取数据。

哪些网页属于get请求?

- 中商官网：https://s.askci.com/stock/a/0-0?reportTime=2021-06-30&pageNum=1#QueryCondition

  > 静态网页+get请求

- 中注协官网：https://www.cicpa.org.cn/ztzl1/Professional_standards/xxzztx/swzy/

  > 静态网页+get请求

post请求

- post请求是向服务器传送数据。
- post请求和get请求相比，多了一个data参数，data是一个字典，里面是一些键值对。

哪些网页属于post请求?

- 证监会官网：http://www.csrc.gov.cn/wcm/websearch/zjh_simp_list.jsp

  > 静态网页+post请求

- 巨潮资讯网：http://www.cninfo.com.cn/new/commonUrl/pageOfSearch?url=disclosure/list/search

  > 动态网页+post请求

requests库

requests是用python语言基于urllib编写的，与urllib相比，requests更加方便，可以节约我们大量的工作，建议爬虫使用requests库。

安装

```
pip install requests
```

我们前面讲了如何判断当前网页**是静态网页还是动态网页**，接着又找出了**当前网页请求的真实网址**，根据请求的真实网址确定了**当前网页的请求方式是get还是post。**这些内容是**进行Python网络爬虫的首要分析步骤。**

简单总结一下：

1. 判断**网页形式**：

   > 静态网页还是动态网页

2. 寻找**真实网址**：

   打开浏览器开发工具，点击network，刷新当前网页，network窗口会加载出很多的请求，可以通过点击每一个请求后出现的**Preview**（返回结果的预览）、**Response**（返回结果的源代码）来判断**是否为当前网页的真实请求。**

   > 静态网页：一般存在于network里面的第一个请求的headers。
   >
   > 动态网页:一般不会存在于network里面的第一个请求，需要在加载出来的所有请求中一个一个的去试，　　返回的结果一般为json格式。

3. 查看**请求方式**

   > 真实网址的下方就是请求方式。

走完这三步之后，我们就需要通过Python代码来实现请求网址、返回请求内容等操作。

这时候，就可以通过requests来实现。

get请求

核心代码：`requests.get(url)`

实例：

```
import requests

url = 'https://www.baidu.com'

response = requests.get(url)

print(response)
```

打印出来的结果是：`<Response [200]>`。`<>`表示这是一个对象，也就是我们这里获取的是一个response的对象，**200表示状态码， 代表服务器已成功处理了请求。**

post请求

核心代码：`requests.post(url,data={请求参数})`

实例：

```
import requests

url = 'https://fanyi.baidu.com'

data = {'from': 'zh',
        'to': 'en',
        'query': '注册会计师'
        }

response = requests.post(url, data=data)

print(response)
```

data部分的参数，取自于页面**NetWork→Headers→Form** Data。打印出来的结果是：`<Response [200]>`。

response方法

获取网页的解码字符串

通过上述例子我们可以看到，不管是get请求还是post请求，我们得到的返回结果都是一个Response[200]的对象，但是我们想要得到的，应该是与网页response下一样的字符串对象，这时就需要用到response的方法了。

- response.text。获取网页的HTML字符串，该方式往往会出现乱码。

```
import requests

url = 'http://www.baidu.com'

response = requests.get(url)

print(response.text)
```

- response.content.decode()。把相应的二进制字节流转化为str类型。

```
import requests

url = 'http://www.baidu.com'

response = requests.get(url)

print(response.content.decode())
```

在这里我总结了获取网页源码的三种方式，通过这三种方式，一定可以获取到网页正确解码之后的字符串：

- response.text
- response.content.decode()
- response.content.decode('编码形式')　　注：编码形式一般为`utf-8`或者`gbk`

伪装浏览器原理

反爬虫机制之一：判断用户是否用浏览器访问，如果不是，请求就会被拦截

- 构造请求头：`header={}`

- 伪装浏览器：`header={'User-Agent' : '浏览器名称'}`
- 把请求头装进请求对象：`requests.get(url,headers=header)`

```
import requests

url = 'http://www.baidu.com'

header={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
                      AppleWebKit/537.36 (KHTML, like Gecko) \
                      Chrome/92.0.4515.131 Safari/537.36'}

response = requests.get(url,headers=header)

print(response.content.decode())
```

xpath表达式

首先，我们需要安装lxml库来支持xpath的操作。

```
pip install lxml
```

然后通过lxml库对html进行转换，对转换后的html对象进行xpath的定位操作。

xpath的定位操作分为：

- 绝对定位
- 相对定位

xpath绝对定位

格式：`/html/body/div[1]/div[2]/div[2]/a[1]`

> 优点：直接打开浏览器开发者工具，复制xpath绝对路径，快捷方便，定位准确
>
> 缺点：绝对路径发生变化，定位就会失效，不方便后期维护

实例

```
import requests
from lxml import etree

url = 'http://www.baidu.com'

header={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
                      AppleWebKit/537.36 (KHTML, like Gecko) \
                      Chrome/92.0.4515.131 Safari/537.36'}

response = requests.get(url,headers=header)

#对html字符串进行转换
html = etree.HTML(response.content.decode())

#获取'贴吧'所在的标签
#xpath绝对定位
result=html.xpath('//*[@id="s-top-left"]/a[6]')
```

```
print(result)
# print(result[0].text)
```

xpath相对定位

格式：`//input[@id="s-top-left"]`

> 优点：相对路径不易发生变化
>
> 缺点：需要查找标签元素

由于网页源码比较多，不方便讲解，所以我们以**解析本地html文件**为例对**xpath的操作原理**进行详细讲解。

本地html文件如下：

```
<!DOCTYPE html>
<html>
<head>
    <title>这是需要解析的页面</title>
</head>
<body>
<div>
    <ul>

         <li class="item-88">这是一个li标签</li>

         <li class="item-0"><a href="baidu.com">张三</a></li>

         <li class="item-1"><a href="link2.html">李四</a></li>

         <li class="item-inactive">
            <a href="link3.html">
                <span class="bold">王五<br/>666</span>
            </a>
         </li>

         <li class="item-1"><a href="link4.html">赵六</a></li>

         <li class="item-0"><a href="link5.html">老七</a></li>

     </ul>
 </div>
</body>
</html>
```

获取某一类标签

```
#获取一类标签

from lxml import etree

#对本地的html文件进行转换
html=etree.parse(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python基础课程\新建文本文档.html")

result=html.xpath("//a") #获取所有a标签的信息

print(result)

# for i in result:
#   print(i.text)
```

获取指定属性的标签

```
#获取指定属性的标签

from lxml import etree

#对本地的html文件进行转换
html = etree.parse(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python基础课程\新建文本文档.html")

result=html.xpath("//li/a[@href='link2.html']")

print(result)

# for i in result:
#   print(i)
```

获取标签的属性

```
#获取标签的属性

from lxml import etree

#对本地的html文件进行转换
html = etree.parse(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python基础课程\新建文本文档.html")

result=html.xpath("//li/a/@href")

print(result)

# for i in result:
#   print(i)
```

获取子标签

```
#获取子标签

from lxml import etree
```

```
#对本地的html文件进行转换
html = etree.parse(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python基础课程\新建文本文档.html")

result1=html.xpath("//li/a") #获取下一级子标签
result2=html.xpath("//li//span") #获取所有符合条件子标签

#print(result1[0].text)

#获取li标签下a标签里所有的class

result3=html.xpath("//li/a//@class")

print(result3)
```

一行代码爬取网页表格

- 安装第三方库`pandas`：

```
pip install pandas
```

- `pandas`的read_html()方法：

```
#导入模块
import pandas as pd

#用pandas读取本地Excel，如果不存在就新建一个Excel
write = pd.ExcelWriter("资产负债表.xlsx")

#一行代码爬取网页表格，适用于大多数网页
table=pd.read_html('https://s.askci.com/stock/financialreport/000001/')
print(table)
print(table[0])

#写入Excel
table[0].to_excel(write, sheet_name='资产负债表')

#保存
write.save()
```

爬取中商官网A股上市公司三大报表（静态网页+get请求）

- re是Python的内置模块，正则表达式功能。

```
#导入模块
import pandas as pd
import requests
from lxml import etree
import re


#构造请求头，伪装浏览器
```

```
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)\
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0\
    .2743.116 Safari/537.36'
    }

#用pandas读取本地Excel，如果不存在就新建一个Excel
write = pd.ExcelWriter("资产负债表.xlsx")
write2 = pd.ExcelWriter("利润表.xlsx")
write3 = pd.ExcelWriter("现金流量表.xlsx")

#循环每一页
for i in range(1,11):

    #拼接网址
    url = 'http://s.askci.com/stock/a/?0-0?reportTime=2021-06-30&pageNum=%s' %
(str(i))
    response=requests.get(url,headers=headers).content.decode('utf-8')
    html=etree.HTML(response)

    #定位当前页面所有公司的标签元素
    res=html.xpath('//*[@id="myTable04"]/tbody/tr')
    print(len(res))

    for r in res:

        #股票代码
        r1=r.xpath('./td/a/text()')[0]

        #公司名称
        r2=r.xpath('./td/a/text()')[1]
        print(r2)

        #去除*ST中的*
        r2=re.sub(r"\*","",r2)
        print(r2)

        #链接
        r3=r.xpath('./td/a/@href')[2]

        print(r1)
        print(r2)
        print(r3)

        #获取资产负债表
        tb = pd.read_html(r3)[0]
        print(tb)
        #写入Excel
        tb.to_excel(write, sheet_name=r1+r2)

        #获取利润表
        tb2 = pd.read_html(r3+'/profit/')[0]
        print(tb2)
        #写入Excel
        tb2.to_excel(write2, sheet_name=r1+r2)

        #获取现金流量表
        tb3 = pd.read_html(r3+'/cashflow/')[0]
```

```
        print(tb3)
        #写入Excel
        tb3.to_excel(write3, sheet_name=r1+r2)

    print('-------------------第'+str(i)+'页抓取完成-------------------')

#保存
write.save()
write2.save()
write3.save()
```

爬取中注协PDF版审计准则（静态网页+get请求）

- os是Python的内置模块，用来处理文件和目录。

```
#导入模块
import requests
from lxml import etree
import os


#构造请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)\
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0\
    .2743.116 Safari/537.36'}

#指定网址
url='https://www.cicpa.org.cn/ztzl1/Professional_standards/xxzztx/zyzz/sjzz/'

#用当前路径和字符串“审计准则”拼接一个文件路径
path=os.path.join(os.path.dirname(__file__),"审计准则")

#判断当前路径下是否存在名为“审计准则”的文件夹，如没有，就新建一个
if not os.path.exists(path):
    os.mkdir(path)

#循环每一页
for i in range(0,3):
    if i==0:
        urls=url
    else:
        urls=url+'index_%s.html' % i  #拼接网址
    i=i+1

    print('第',i,'页')

    #请求网址
    response=requests.get(urls,headers=headers).content.decode('utf-8')
    #解析返回的网页源码
    html=etree.HTML(response)

    #提取链接
    result1=html.xpath('//*[@id="sub"]//li/a/@href')
```

```
    #提取标题
    result2=html.xpath('//*[@id="sub"]//li/a/@title')

    print(len(result1),'个文件**********************************')

    #循环每一个标题和链接
    for r in range(len(result1)):
        print(result2[r])
        print(result1[r][1:])
        #拼接PDF下载地址
        rs=url+result1[r][1:]
        print(rs)

        #请求PDF下载地址，获取返回的字节类型内容
        data=requests.get(rs).content

        #在当前路径下新建一个PDF文件并写入内容,wb代表以二进制的格式写入
        with open(path+"//{}.pdf".format(result2[r]),"wb") as f:
            f.write(data)
```

爬取证监会官网反馈意见（静态网页+post请求）

- **打开证监会官网，在搜索框内直接搜索“反馈意见”，我们对搜索出来的内容进行批量爬取。**
- **这个案例包括各种处理情况，可以练习我们爬虫时的综合处理能力。**
- 安装第三方库 `python-docx`：

  ```
  pip install python-docx
  ```

- 捕捉异常语句 `try...except...`

```
#导入模块
import requests
from lxml import etree
from docx import Document


#构造请求头
headers={
    "user-agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36"
    }

#指定网址
login_url = 'http://www.csrc.gov.cn/wcm/websearch/zjh_simp_list.jsp'

#循环每一页
for j in range(1,3):

    print("第"+str(j)+"页\n-------------------------------------------------")

    #构造请求参数
    name_pwd = {'page':j,'schword':'反馈意见','searchword':'pub/newsite/'}
    #请求网址
```

```
    response = requests.post(url=login_url, data=name_pwd,headers=headers).text
    html=etree.HTML(response)

    #定位当前页面所有标题文件的标签元素
    resu=html.xpath('//div[@class="h1"]/a')

    #循环每一个标签元素
    for i in range(0,len(resu)):

        #提取该标签下的所有文本，拼接成标题
        title=''.join(resu[i].xpath('.//text()'))
        print(title)

        #提取该标签下的链接
        href=resu[i].xpath('./@href')[0]
        # print(href)

        #拼接网址
        url='http://www.csrc.gov.cn'+href
        print(url)

        #判断该网址中是否包含pdf，如果包含，就下载该PDF文件
        if '.pdf' in url:

            #请求PDF下载地址，获取返回的字节类型内容
            data=requests.get(url).content

            #在当前路径下新建一个PDF文件并写入内容,wb代表以二进制的格式写入
            with open("{}.pdf".format(title),"wb") as f:
                f.write(data)

        elif '.doc' in url:

            #请求DOC下载地址，获取返回的字节类型内容
            data=requests.get(url).content

            #在当前路径下新建一个DOC文件并写入内容,wb代表以二进制的格式写入
            with open("{}.doc".format(title),"wb") as f:
                f.write(data)

        else:

            #捕捉异常语句 try...except...
            try:

response=requests.get(url=url,headers=headers).content.decode('utf-8')
                html=etree.HTML(response)

                #新建Word文档
                doc=Document()

                #定位当前页面标题的标签元素
                title=html.xpath('//div[@class="title"]//text()')[0]
                print(title)
                #写入word
                doc.add_heading(level=1).add_run(title)

                #定位当前页面时间的标签元素
```

```
                time=html.xpath('//div[@class="time"]//text()')
                time=''.join(time)
                print(time)
                #写入word
                doc.add_paragraph().add_run(time)

                #定位当前页面正文的标签元素
                Custom=html.xpath('//div[@class="Custom_UnionStyle"]/p')
                for Cus in Custom:
                    text=''.join(Cus.xpath('.//text()'))
                    print(text)
                    #写入word
                    doc.add_paragraph().add_run(text)

                #保存word文档
                doc.save("{}.docx".format(title))

            except Exception as e:
                print("下载失败……")
```

留两个小作业:

- 爬取：证监会首页 > 上市公司监管部 > 行政许可公开 > **反馈意见**

  网址：http://www.csrc.gov.cn/pub/newsite/ssgsjgb/ssbbgczxzxkhz/bgczfkyj/index.htm

  形式：静态网页+get请求

- 爬取：证监会首页 > 信息公开 >按仲裁文种查看 > **行政处罚决定**

  网址：http://www.csrc.gov.cn/pub/zjhpublic/

  形式：动态网页+get请求

爬取证监会官网行政处罚（动态网页+get请求）

```
#导入模块
import requests
from lxml import etree
from docx import Document
# from docx.oxml.ns import qn
# from docx.shared import Pt
# from docx.shared import RGBColor


#构造请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)\
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0\
    .2743.116 Safari/537.36'}

#循环每一页
for x in range(0,74):

    #构造url
    if x==0:
```

```
        url = 'http://www.csrc.gov.cn/pub/zjhpublic/3300/3313/index_7401.htm'
    else:

url="http://www.csrc.gov.cn/pub/zjhpublic/3300/3313/index_7401_"+str(x)+".htm"

    x=x+1
    print('第',x,'页')
    print('----------------------------------------------------')

    #请求url
    response=requests.get(url,headers=headers).content.decode('utf-8')
    html=etree.HTML(response)
    # print(response)

    #获取当前页面文件的所有标签元素
    adr=html.xpath('//li[@class="mc"]/div/a')
    for a in adr:

        #新建Word文档
        doc=Document()
        adr2=a.xpath('./text()')[0]
        adr3='http://www.csrc.gov.cn/pub/zjhpublic'+a.xpath('./@href')[0][5:]
        # print(adr2)
        # print(adr3)

        #标题
        print(adr2)

        #直接写入word
        doc.add_heading(level=1).add_run(adr2)

        # #修改格式后写入word
        # head=doc.add_heading(level=1).add_run(adr2)
        # head.font.name = u'宋体'
        # head._element.rPr.rFonts.set(qn('w:eastAsia'), u'宋体')

        #内容
        response=requests.get(adr3,headers=headers).content.decode('utf-8')
        html=etree.HTML(response)
        con=html.xpath('//*[@id="ContentRegion"]//p')
        for co in con:
            c="".join(co.xpath('.//text()'))
            print(c)

            #直接写入word
            doc.add_paragraph().add_run(c)

            # #修改格式后写入word
            # p=doc.add_paragraph()
            # p.paragraph_format.space_before=Pt(0)
            # p.paragraph_format.space_after=Pt(0)
            # p.paragraph_format.line_spacing = Pt(18)
            # pa=p.add_run(c)
            # pa.font.name = u'宋体'
            # pa.font.size = Pt(10.5)
            # pa._element.rPr.rFonts.set(qn('w:eastAsia'), u'宋体')

        #保存Word文档
```

```
        doc.save("{}.docx".format(adr2))

    print('爬取完毕！')
    print('-----------------------------------------------------')
```

爬取巨潮资讯网PDF版公告（动态网页+post请求）

- json是Python的内置模块，是一种轻量级的数据交换格式，易于阅读和编写。

```
#导入模块
import requests
from lxml import etree
import json


# url='http://www.cninfo.com.cn/new/commonUrl/pageOfSearch?
url=disclosure/list/search'

#指定网址
url='http://www.cninfo.com.cn/new/hisAnnouncement/query' #真实URL

#构造请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64)\
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0\
    .2743.116 Safari/537.36'}

for k in range(1,3):

    #构造请求参数
    query={'pageNum':k,
        'pageSize':30,
        'column':'szse',
        'tabName':'fulltext',
        'stock':'000001,gssz0000001',

 'category':'category_ndbg_szsh;category_bndbg_szsh;category_yjdbg_szsh;category
_sjdbg_szsh;category_yjygjxz_szsh;category_dshgg_szsh;category_jshgg_szsh',
        'seDate':'2015-12-31~2021-12-31',
        'isHLtitle':'true'}

    #请求网址
    response=requests.post(url,query,headers).text
    # print(response)

    #对返回的json格式数据进行解析
    data = json.loads(response)
    # print(data)

    count=data['announcements']

    #循环当前页面的所有文件
```

```
    for i in range(len(count)):

        #获取标题
        title=count[i]["announcementTitle"]
        print(title)

        #拼接PDF下载地址
        url2='http://static.cninfo.com.cn/'+count[i]["adjunctUrl"]
        print(url2)

        #请求PDF下载地址，获取返回的字节类型内容
        data=requests.get(url2).content

        #在当前路径下新建一个PDF文件并写入内容,wb代表以二进制的格式写入
        with open("{}.pdf".format(title),"wb") as f:
            f.write(data)

        print("-------------下载完成!-------------")

    print("第",k,"页****************下载完成!****************")
```

selenium库

基本介绍

Selenium是一个Web的自动化测试工具，最初是为网站自动化测试而开发的, Selenium可以直接运行在浏览器上，它支持所有主流的浏览器（包括谷歌，火狐等），可以接收指令，让浏览器自动加载页面，获取需要的数据。

总而言之，Selenium是可以模拟人类行为操纵浏览器的一个工具。

为什么要用Selenium?

网页数据分为2种：

- 静态页面：数据都包含在网页的HTML中，可以直接在网页的HTML中提取数据。
  处理方式：正则表达式，**xpath**， beautifulsoup4。
- 动态页面：数据是在网页打开后的时候用动态手段（js等）加载到页面上来的,并非一开始就在打开的网页HTML中，所以静态处理方式拿不到数据。
  处理方式：**抓包**，逆解js，**自动化工具（Selenium等）**。

Selenium怎么用?

1. 安装selenium模块：`pip install selenium`
2. 下载和浏览器版本对应的driver驱动。
3. 代码里正常导入，并指定驱动名称(路径)就可以使用了。

基本步骤如下：

```
#导入selenium模块
from selenium import webdriver

#加在浏览器驱动，指定chromedriver驱动的路径
driver = webdriver.Chrome(executable_path=r'E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe')

#打开网页
driver.get('https://www.baidu.com/')
```

如何下载和浏览器版本对应的driver驱动?

> 淘宝npm镜像 地址： http://npm.taobao.org/
>
> chromedriver驱动的下载地址如下：
> http://chromedriver.storage.googleapis.com/index.html

元素定位

定位元素，selenium提供了8中元素定位方法：

> （1）**find_element_by_id()**：html规定，id在html中必须是唯一的，有点类似于身份证号
>
> （2）**find_element_by_name()**：html规定，name用来指定元素的名称，有点类似于人名
>
> （3）find_element_by_tag_name()：通过元素的签名来定位
>
> （4）find_element_by_class_name()：html规定，class指定元素的类名
>
> （5）**find_element_by_link_text()**：专门用来定位文本链接
>
> （6）**find_element_by_partial_link_text()**：是对link_text的一种补充，有些文字链接比较长，可以取一部分链接文字进行定位，只要这部分文字是唯一标志这个链接的
>
> （7）find_element_by_css_selector()
>
> （8）**find_element_by_xpath()**

id与name定位

```
#导入selenium模块
from selenium import webdriver

#加在浏览器驱动，指定chromedriver驱动的路径
driver = webdriver.Chrome(executable_path=r'E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe')

#打开网页
driver.get('https://www.baidu.com/')

#输入内容
driver.find_element_by_id('kw').send_keys('注册会计师')
# driver.find_element_by_name('wd').send_keys('注册会计师')

#点击搜索
driver.find_element_by_id('su').click()
```

link_text与partial_link_text定位

```
#导入selenium模块
from selenium import webdriver

#加在浏览器驱动，指定chromedriver驱动的路径
driver = webdriver.Chrome(executable_path=r'E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe')

#打开网页
driver.get('https://www.baidu.com/')

#点击
driver.find_element_by_link_text('新闻').click()
# driver.find_element_by_partial_link_text('123').click()
```

xpath定位

```
#导入selenium模块
from selenium import webdriver

#加在浏览器驱动，指定chromedriver驱动的路径
driver = webdriver.Chrome(executable_path=r'E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe')

#打开网页
driver.get('https://www.baidu.com/')

#点击贴吧
driver.find_element_by_xpath('//*[@id="s-top-left"]/a[6]').click()
```

模拟操作

比较常见的几种操作元素的方式：

- 点击按钮

  ```
  driver.find_element_by_xpath('//*[@id="s-top-left"]/a[6]').click()
  ```

- 清除输入框的内容

  ```
  driver.find_element_by_xpath('//*[@id="s-top-left"]/a[6]').clear()
  ```

- 在输入框内输入内容

  ```
  driver.find_element_by_xpath('//*[@id="s-top-left"]/a[6]').send_keys('内容')
  ```

- 浏览器窗口最大化

  ```
  driver.maximize_window()
  ```

- 跳转浏览器窗口

```
driver.switch_to_window(driver.window_handles[0])
```

- 滑动浏览器滚动条

```
js="var action=document.documentElement.scrollTop=770"
driver.execute_script(js)
```

- 网页截图

```
driver.save_screenshot('123.png')
```

- 退出浏览器

```
driver.quit()
```

解析网页

```
#导入模块
from selenium import webdriver
from lxml import etree

#加在浏览器驱动，指定chromedriver驱动的路径
driver = webdriver.Chrome(executable_path=r'E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe')

#打开网页
driver.get('https://www.baidu.com/')

#浏览器窗口最大化
driver.maximize_window()

#点击贴吧
driver.find_element_by_xpath('//*[@id="s-top-left"]/a[6]').click()

#跳转浏览器窗口
driver.switch_to_window(driver.window_handles[1])

#滑动浏览器滚动条
js="var action=document.documentElement.scrollTop=1000"
driver.execute_script(js)

#网页截图
driver.save_screenshot('123.png')

#解析网页
source=driver.page_source
html=etree.HTML(source)

#定位标签元素，提取文本“全吧搜索”
name=html.xpath('//*[@id="tb_header_search_form"]/span[2]/a')

#打印标签元素
print(name)
```

```
#打印标签内的文本
print(name[0].text)

#退出浏览器
driver.quit()
```

爬取天眼查企业工商信息（selenium爬虫实战）

- 安装第三方库`openpyxl`

```
pip install openpyxl
```

```
#导入模块
from selenium import webdriver
from time import sleep
from lxml import etree
from openpyxl import load_workbook


#实例化一个浏览器      executable_path="chromedriver的本地路径"
driver = webdriver.Chrome(executable_path=r"E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe")

#窗口最大化
driver.maximize_window()

#打开天眼查官网，记得写完整的url，包括http和https
driver.get('https://www.tianyancha.com/search?')
sleep(2)

#登陆
driver.find_element_by_xpath('//a[@class="link-nav"]').click()    #进入登录界面
driver.find_element_by_xpath('//div[@active-tab="1"]').click()    #切换至密码登陆
driver.find_element_by_xpath('//*[@id="mobile"]').send_keys('账号')    #输入账号
driver.find_element_by_xpath('//*[@id="password"]').send_keys('密码')     #输入密码
driver.find_element_by_xpath('//div[@class="btn -xl btn-primary -
block"]').click()    #确认登录
sleep(10)

#打开excel
wb=load_workbook('数据.xlsx')

#把工作表Sheet1赋值给ws
ws=wb['Sheet1']

#进入for循环，根据公司全称依次查询企业工商信息，max_row代表工作表的下限
for i in range(2,ws.max_row+1):

    sleep(1)
    driver.find_element_by_id("header-company-search").clear()     #清除搜索框内容
    driver.find_element_by_id("header-company-
search").send_keys(ws.cell(i,1).value)     #把公司全称输入到搜索框
    sleep(1)
```

```
driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div/div[2]/div[1]/div
').click()    #点击搜索
    sleep(2)
    driver.find_element_by_xpath('//div[@class="header"]/a[@class="name
"]').click()    #点击搜索到的第一家企业，即搜索相似度最高的企业
    sleep(1)

    driver.switch_to_window(driver.window_handles[1])    #切换窗口

    source=driver.page_source    #获取网页源码

    #解析网页源码，根据需要查询的信息获取对应的网页元素
    html=etree.HTML(source)

    name=html.xpath('//*
[@id="company_web_top"]/div[3]/div[3]/div[1]/span/span/h1')
    jyzt=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[1]/td[4]')
    clrq=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[2]/td[2]')
    zczb=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[3]/td[2]/div')
    sjzb=html.xpath('//td[@width=""]')
    hzrq=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[6]/td[6]')
    tyshxydm=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[5]/td[2]/span/span')
    zzjgdm=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[5]/td[6]/span/span')
    gszch=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[4]/td[4]')
    nsrsbh=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[5]/td[4]/span/span')
    hy=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[7]/td[4]')
    gslx=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[7]/td[2]')
    yyqx=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[6]/td[2]/span')
    djjg=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[8]/td[4]')
    rygm=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[7]/td[6]')
    cbrs=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[8]/td[2]')
    jyzt=html.xpath('//*[@id="_container_baseInfo"]/table/tbody/tr[1]/td[4]')
    zcdz=html.xpath('//div[@style="max-height:16px;"]/div')
    jyfw=html.xpath('//*
[@id="_container_baseInfo"]/table/tbody/tr[11]/td[2]/span')

    #打印信息
    print("企业名称："+name[0].text)
    print("经营状态："+jyzt[0].text)
    print("成立日期："+clrq[0].text)
    print("注册资本："+zczb[0].text)
    print("实缴资本："+sjzb[-1].text)
    print("核准日期："+hzrq[0].text)
    print("统一社会信用代码："+tyshxydm[0].text)
    print("组织机构代码："+zzjgdm[0].text)
    print("工商注册号："+gszch[0].text)
    print("纳税人识别号："+nsrsbh[0].text)
    print("行业："+hy[0].text)
    print("公司类型："+gslx[0].text)
    print("营业期限："+yyqx[0].text)
    print("登记机关："+djjg[0].text)
    print("人员规模："+rygm[0].text)
```

```
    print("参保人数："+cbrs[0].text)
    print("经营状态："+jyzt[0].text)
    print("注册地址："+zcdz[0].text)
    print("经营范围："+jyfw[0].text)

    #把信息写入excel
    ws.cell(i,2).value=jyzt[0].text
    ws.cell(i,3).value=clrq[0].text
    ws.cell(i,4).value=zczb[0].text
    ws.cell(i,5).value=sjzb[-1].text
    ws.cell(i,6).value=hzrq[0].text
    ws.cell(i,7).value=tyshxydm[0].text
    ws.cell(i,8).value=zzjgdm[0].text
    ws.cell(i,9).value=gszch[0].text
    ws.cell(i,10).value=nsrsbh[0].text
    ws.cell(i,11).value=hy[0].text
    ws.cell(i,12).value=gslx[0].text
    ws.cell(i,13).value=yyqx[0].text
    ws.cell(i,14).value=djjg[0].text
    ws.cell(i,15).value=rygm[0].text
    ws.cell(i,16).value=cbrs[0].text
    ws.cell(i,17).value=jyzt[0].text
    ws.cell(i,18).value=zcdz[0].text
    ws.cell(i,19).value=jyfw[0].text

    #关闭当前窗口
    driver.close()

    #跳转至初始窗口
    driver.switch_to_window(driver.window_handles[0])
    sleep(1)

#保存excel
wb.save('数据.xlsx')

#退出浏览器
driver.quit()

#提示运行结束
print("运行结束！")
```

爬取东方财富网财务数据（selenium爬虫实战）

- 在当前浏览器打开一个新窗口：

```
#网址是一个字符串
driver.execute_script('window.open("https://www.baidu.com/")')

#网址是一个变量url
driver.execute_script('window.open("'+url+'")')
```

- get_attribute的常用方法：

```
#获取元素标签的内容(文本信息)
table=driver.find_element_by_id('table_ls').get_attribute('textContent')
table=driver.find_element_by_id('table_ls').text

#获取元素内的全部HTML
table=driver.find_element_by_id('table_ls').get_attribute('innerHTML')
```

```
#导入模块
from selenium import webdriver
from time import sleep
from lxml import etree
import pandas as pd

#实例化一个浏览器
driver = webdriver.Chrome(executable_path=r"E:\常用\B\Python库\辅助
\chromedriver_win32\chromedriver.exe")

#窗口最大化
driver.maximize_window()

#记得写完整的url 包括http和https
driver.get('http://quote.eastmoney.com/center/boardlist.html#boards-BK07311')
sleep(3)

#用pandas读取本地Excel，如果不存在就新建一个Excel
write = pd.ExcelWriter("result.xlsx")

#死循环
while True:

    #解析网页源码
    source=driver.page_source
    html=etree.HTML(source)

    #提取所有的资金流url
    urls=html.xpath('//*[@id="table_wrapper-table"]/tbody/tr/td[4]/a[2]/@href')
    button=html.xpath('//*[@id="main-table_paginate"]/a[2]/@class')[0]
    print(urls)

    #循环每一个资金流url
    for url in urls:

        print(url)
        #在当前浏览器打开一个新窗口
        driver.execute_script('window.open("'+url+'")')
        sleep(2)

        #切换窗口
        driver.switch_to_window(driver.window_handles[1])

        #获取资金流表格所在标签元素内的全部HTML
        table=driver.find_element_by_id('table_ls').get_attribute('innerHTML')

        #一行代码爬取网页表格，适用于大多数网页
        table=pd.read_html(table)
        print(table)
```

```
        print(table[0])

        #获取公司名称

 title=driver.find_element_by_xpath('/html/body/div[1]/div[8]/div[2]/div[11]/div
[1]/div[1]').text

        #过滤文本中的*
        if '*' in title:
            title=title.replace('*','')
        print(title)

        #表格写入Excel
        table[0].to_excel(write,title)

        #关闭当前窗口
        driver.close()
        #跳转至初始窗口
        driver.switch_to_window(driver.window_handles[0])

    #判断元素属性
    if button=="next paginate_button":
        driver.find_element_by_xpath('//*[@id="main-
table_paginate"]/a[2]').click()
        sleep(1)
        continue  #继续循环
    else:
        break    #退出循环

#保存excel
write.save()
#退出浏览器
driver.quit()
print("运行结束！")
```

网络爬虫总结

方法对比

- requests模块爬取速度快，效率高，但是遇到一些反爬虫手段较强的动态网页，就会很难爬取
- selenium模块爬取速度慢，效率低，但是基本上所有的网页都可以进行爬取，容易上手

注意事项

- 尽量用sleep设置延迟时间

  > 1、防止爬取速度太快，被识别为机器人，导致代码运行中断
  >
  > 2、针对selenium，可以防止网页元素还未加载出来，导致元素获取失败、代码运行中断

- 尽量加上异常语句try...except...finally...

  > 可以在代码运行中断的时候，保存已经爬取的数据

- 运行出现异常，首先查看错误出现在第几行，然后再找出原因、解决问题

进阶学习

- 自动识别总页数，进行翻页（掌握）
- 根据爬取的不同内容，创建本地文件夹，分类存放（掌握）
- 模拟登录（了解）
- 破解验证码（了解）
- 多线程（了解）
- 协程（了解）
- JS逆向解析（了解）

OCR文字识别

利用Python编程技术对图像中的文字进行自动化识别，并导出为电子数据。

想要使用python实现文字识别的功能，有以下两种方法：

1. 使用Python的pytesseract库
2. 调用百度AI平台文字识别接口

使用Python的pytesseract库进行文字识别，直接使用 **pip install** 安装pytesseract库即可，比较简单，但效果并不是很理想。

所以在本课程中，我们详细讲解第二种方法——调用百度AI平台文字识别接口。

百度AI开放平台：如果已经有百度账号的，可以使用百度账号登录。

用Python实现文字识别的原理

关于API和SDK

API是什么?

> 电脑需要调用手机里面的信息，这时候你会拿一根数据线将电脑手机连接起来，电脑和手机上连接数据线的接口就相当于“API接口”。

SDK是什么?

> SDK 就是 Software Development Kit 的缩写，翻译过来——软件开发工具包。这是一个覆盖面相当广泛的名词，可以这么说：辅助开发某一类软件的相关文档、范例和工具的集合都可以叫做SDK。
>
> SDK被开发出来是为了减少程序员工作量的。
>
> 你可以把SDK想象成一个虚拟的程序包，在这个程序包中有一份做好的软件功能，这份程序包几乎是全封闭的，只有一个小小接口可以联通外界，这个接口就是API。

总的来说，两者没有值得比较的区别，因为是具有关联性的两种东西。

直接调用API进行文字识别，代码量较多；直接调用SDK进行文字识别，代码量较少。

通用文字识别

百度AI官网技术文档：https://ai.baidu.com/ai-doc/OCR/7kibizyfm#%E9%80%9A%E7%94%A8%E6%96%87%E5%AD%97%E8%AF%86%E5%88%AB

多场景、多语种、高精度的整图文字检测和识别服务

> 通用文字识别，顾名思义，它是通用的、也是最基础的文字识别功能，相信听完本套课程后，大家会有比较深刻的认识。
>
> 通用文字识别，和其他文字识别（增值税发票识别、火车票识别.....等等）的原理都是相同的；在上一小节中，我们初步了解了API和SDK，后面我们会分别通过API和SDK来实现通用文字识别功能，百度AI官网的技术文档中已经为我们提供了所有文字识别板块的API和SDK操作文档，也就是说文字识别的Python核心代码已经给我们写好了，我们在使用的时候，可以根据需求直接复制使用。

通用文字识别，有四个版本可供使用，详见如下功能介绍：

☐ **标准版**

对图片中的文字进行检测和识别，支持中、英、法、俄、西、葡、德、意、日、韩、中英混合等10种语言，并支持中、英、日、韩四语种的类型检测。

用户向服务请求识别某张图中的所有文字。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

'''
通用文字识别
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/general_basic"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用通用文字识别, 图片参数为本地图片 """
client.basicGeneral(image);
```

```
""" 如果有可选参数 """
options = {}
options["language_type"] = "CHN_ENG"
options["detect_direction"] = "true"
options["detect_language"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别, 图片参数为本地图片 """
client.basicGeneral(image, options)

url = "https//www.x.com/sample.jpg"

""" 调用通用文字识别, 图片参数为远程url图片 """
client.basicGeneralUrl(url);

""" 如果有可选参数 """
options = {}
options["language_type"] = "CHN_ENG"
options["detect_direction"] = "true"
options["detect_language"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别, 图片参数为远程url图片 """
client.basicGeneralUrl(url, options)
```

☐ **标准含位置版**

在通用文字识别（标准版）的基础上，返回文字在图片中的位置信息，方便进行版式的二次处理。

用户向服务请求识别某张图中的所有文字，并返回文字在图中的位置信息。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/general"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用通用文字识别（含位置信息版）, 图片参数为本地图片 """
client.general(image);

""" 如果有可选参数 """
options = {}
options["recognize_granularity"] = "big"
options["language_type"] = "CHN_ENG"
options["detect_direction"] = "true"
options["detect_language"] = "true"
options["vertexes_location"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别（含位置信息版）, 图片参数为本地图片 """
client.general(image, options)

url = "https//www.x.com/sample.jpg"

""" 调用通用文字识别（含位置信息版）, 图片参数为远程url图片 """
client.generalUrl(url);

""" 如果有可选参数 """
options = {}
options["recognize_granularity"] = "big"
options["language_type"] = "CHN_ENG"
options["detect_direction"] = "true"
options["detect_language"] = "true"
options["vertexes_location"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别（含位置信息版）, 图片参数为远程url图片 """
client.generalUrl(url, options)
```

☐ **高精度版**

在通用文字识别（标准版）的基础上，提供更高精度的识别服务，支持更多语种识别（丹麦语、荷兰语、马来语、瑞典语、印尼语、波兰语、罗马尼亚语、土耳其语、希腊语、匈牙利语），并将字库从1w+扩展到2w+，能识别所有常用字和大部分生僻字。

用户向服务请求识别某张图中的所有文字，相对于通用文字识别该产品精度更高，但是识别耗时会稍长。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64
```

```
'''
通用文字识别（高精度版）
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/accurate_basic"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用通用文字识别（高精度版） """
client.basicAccurate(image);

""" 如果有可选参数 """
options = {}
options["detect_direction"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别（高精度版） """
client.basicAccurate(image, options)
```

☐ **高精度含位置版**

在通用文字识别（高精度版）的基础上，返回文字在图片中的位置信息，方便进行版式的二次处理。

用户向服务请求识别某张图中的所有文字，并返回文字在图片中的坐标信息，相对于通用文字识别（含位置信息版）该产品精度更高，但是识别耗时会稍长。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

'''
通用文字识别（高精度含位置版）
'''
```

```
request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/accurate"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用通用文字识别（含位置高精度版） """
client.accurate(image);

""" 如果有可选参数 """
options = {}
options["recognize_granularity"] = "big"
options["detect_direction"] = "true"
options["vertexes_location"] = "true"
options["probability"] = "true"

""" 带参数调用通用文字识别（含位置高精度版） """
client.accurate(image, options)
```

提示：使用示例代码前，请记得替换其中的示例Token、图片地址或Base64信息。

基于API实现通用文字识别

```
# encoding:utf-8

import requests
import base64
import json


#填写官网注册的API信息
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

#获取access_token
url = 'https://aip.baidubce.com/oauth/2.0/token?
grant_type=client_credentials&client_id=' + API_Key + '&client_secret=' +
Secret_Key
```

```
#请求网址
response = requests.get(url).text
#将json格式数据转换为字典
token_info = json.loads(response)
#获取access_token
access_token = token_info['access_token']


'''
通用文字识别（高精度版）
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/accurate_basic"

# 二进制方式打开图片文件
f = open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\代码
\11.png", 'rb')
img = base64.b64encode(f.read())

#构造请求参数
params = {"image":img}
#构造url
request_url = request_url + "?access_token=" + access_token
#构造请求头
headers = {'content-type': 'application/x-www-form-urlencoded'}
#请求网址
response = requests.post(request_url, data=params, headers=headers)

# print (response.text)
# print (type(response.text))

#将json格式数据转换为字典
result = json.loads(response.text)
#提取列表
words_result = result['words_result']

#循环出识别出的每一个字符串
for word in words_result:
    print(word['words'])
```

基于SDK实现通用文字识别

- 安装使用Python SDK

```
pip install baidu-aip
```

```
from aip import AipOcr

#填写官网注册的API信息
APP_ID="16756034"
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

client=AipOcr(APP_ID,API_Key,Secret_Key)

#读取图片
```

```
with open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\代码\11.png","rb") as f:
    image=f.read()

#调用通用文字识别（高精度版）
response=client.basicAccurate(image)

#调用通用文字识别（标准版）
# response=client.basicGeneral(image)

# print (response)
# print (type(response))

#提取列表
words_result = response['words_result']

# #循环出识别出的每一个字符串
# for word in words_result:
#     print(word['words'])

#把识别结果写入txt文件
with open("识别结果.txt","a") as f:

    #循环出识别出的每一个字符串
    for word in words_result:

        print(word['words'])
        f.write(word['words']+'\n')
```

手写文字识别

请大家思考一下：如何用Python通过百度AI文字识别接口实现手写文字识别功能?

自己尝试用完整的Python代码写出这么一个程序。

温馨提示：


```
from aip import AipOcr

#填写官网注册的API信息
APP_ID="16756034"
```

```
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

client=AipOcr(APP_ID,API_Key,Secret_Key)

#读取图片
with open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\手写文字.jpg","rb") as f:
    image=f.read()

#调用手写文字识别
response=client.handwriting(image)

# print (response)
# print (type(response))

#提取列表
words_result = response['words_result']

# #循环出识别出的每一个字符串
# for word in words_result:
#     print(word['words'])

#把识别结果写入txt文件
with open("识别结果.txt","a") as f:

    #循环出识别出的每一个字符串
    for word in words_result:

        print(word['words'])
        f.write(word['words']+'\n')
```

表格文字识别

对图片中的表格文字内容进行提取和识别，支持识别完整框线表格、含合并单元格表格或无框线表格，并可选择以JSON或Excel形式进行返回

- [ ] **简单表格文字识别**

支持识别具备完整框线的常规简单表格，结构化输出表头、表尾及每个单元格的文字内容

- [ ] **复杂表格文字识别**

可识别无表格框线，但行、列位置明确的表格，支持含合并单元格的复杂表格文字识别

表格文字识别(同步接口)

支持识别表格线齐全的常规表格和无框线表格的单元格内容，结构化输出表头、表尾及每个单元格的文字内容。 本接口为同步接口，相比于异步接口，本接口在请求后会实时返回请求结果。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64
```

```
'''
表格文字识别(同步接口)
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/form"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用表格文字识别同步接口 """
client.form(image);
```

表格文字识别(异步接口)

对图片中的表格文字内容进行提取和识别，结构化输出表头、表尾及每个单元格的文字内容。支持识别常规表格及含合并单元格表格，并可选择以JSON或Excel形式进行返回。 本接口为异步接口，分为两个API：提交请求接口、获取结果接口。下面分别描述两个接口的使用方法。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

'''
表格文字识别(异步接口)
'''

request_url = "https://aip.baidubce.com/rest/2.0/solution/v1/form_ocr/request"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
```

```
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

image = get_file_content('example.jpg')

""" 调用表格文字识别 """
client.tableRecognitionAsync(image);
```

```
requestId = "23454320-23255"

""" 调用表格识别结果 """
client.getTableRecognitionResult(requestId);

""" 如果有可选参数 """
options = {}
options["result_type"] = "json"

""" 带参数调用表格识别结果 """
client.getTableRecognitionResult(requestId, options)
```

```
""" 同步获取表格识别 返回识别结果 """
client.tableRecognition(
    get_file_content('table.jpg'),
    {
        'result_type': 'json',
    },
)
```

```
from aip import AipOcr
import requests
import time


#填写官网注册的API信息
APP_ID = '16756034'
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"
client=AipOcr(APP_ID,API_Key,Secret_Key)

#图片地址
filePath=r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\代码
\22.jpg"

#读取图片
with open(filePath, 'rb') as fp:
```

```
    image = fp.read()

#调用表格文字识别
res = client.tableRecognitionAsync(image)

#获取识别ID号
req_id = res['result'][0]['request_id']

#OCR识别也需要一定时间，设定每隔2秒查询一次
for count in range(1,10):

    res = client.getTableRecognitionResult(req_id)     #通过ID获取识别状态信息
    print(res['result']['ret_msg'])

    if res['result']['ret_msg'] == '已完成':
        break    #云端处理完毕，成功获取表格文件下载地址，跳出循环
    else:
        time.sleep(2)

#返回excel的文件下载地址
url = res['result']['result_data']

#用图片名称给excel重命名
xls_name = filePath.split('.')[0] + '.xls'

#excel文件下载
r = requests.get(url)
with open(xls_name, 'wb') as f:
    f.write(r.content)
```

财务票据文字识别

增值税发票识别

支持对增值税普票、专票、卷票、电子发票、区块链发票的所有字段进行结构化识别，包括发票基本信息、销售方及购买方信息、商品信息、价税信息等，其中五要素识别准确率超过 99.9%； 同时，支持对增值税卷票的 21 个关键字段进行识别，包括发票类型、发票代码、发票号码、机打号码、机器编号、收款人、销售方名称、销售方纳税人识别号、开票日期、购买方名称、购买方纳税人识别号、项目、单价、数量、金额、税额、合计金额(小写)、合计金额(大写)、校验码、省、市，四要素平均识别准确率可达95%以上。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

'''
增值税发票识别
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/vat_invoice"
# 二进制方式打开图片文件
```

```
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

```
# encoding:utf-8

import requests
import base64
import json
import pandas as pd


#填写官网注册的API信息
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

#获取access_token
url = 'https://aip.baidubce.com/oauth/2.0/token?
grant_type=client_credentials&client_id=' + API_Key + '&client_secret=' +
Secret_Key
#请求网址
response = requests.get(url).text
#将json格式数据转换为字典
token_info = json.loads(response)
#获取access_token
access_token = token_info['access_token']


'''
增值税发票识别
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/vat_invoice"

# 二进制方式打开图片文件
f = open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\发票
_1.jpg", 'rb')
img = base64.b64encode(f.read())

#构造请求参数
params = {"image":img}
#构造url
request_url = request_url + "?access_token=" + access_token
#构造请求头
headers = {'content-type': 'application/x-www-form-urlencoded'}
#请求网址
response = requests.post(request_url, data=params, headers=headers)

# print (response.text)
# print (type(response.text))
```

```
#将json格式数据转换为字典
result = json.loads(response.text)
#提取列表
words_result = result['words_result']
# print(words_result)

#把票面信息写入列表list
list=
[[words_result['InvoiceTypeOrg'],words_result['InvoiceCode'],words_result['Invoi
ceNum'],words_result['InvoiceDate'],words_result['PurchaserName'],words_result['
PurchaserRegisterNum'],words_result['PurchaserAddress'],words_result['PurchaserB
ank'],words_result['SellerName'],words_result['SellerRegisterNum'],words_result[
'SellerAddress'],words_result['SellerBank'],words_result['CommodityName'],words_
result['CommodityType'],words_result['CommodityUnit'],words_result['CommodityNum
'],words_result['CommodityPrice'],words_result['CommodityAmount']
,words_result['CommodityTaxRate'],words_result['CommodityTax'],words_result['Tot
alAmount'],words_result['TotalTax'],words_result['AmountInFiguers'],words_result
['Remarks'],words_result['Payee'],words_result['Checker'],words_result['NoteDraw
er']]]

#读取本地Excel
write = pd.ExcelWriter("发票批量识别.xlsx")

#把list转换为DataFrame格式，传给df
df = pd.DataFrame(list)

#给df添加列标题
df.columns = ['发票名称','发票代码','发票号码','开票日期','购方名称','购方纳税人识别
号','购方地址及电话','购方开户行及账号','销售方名称','销售方纳税人识别号','销售方地址及电
话','销售方开户行及账号','货物名称','规格型号','单位','数量','单价','金额','税率','税
额','合计金额','合计税额','价税合计','备注','收款人','复核','开票人']

#把df导入本地Excel
df.to_excel(write,'Sheet1')

#保存Excel
write.save()


#打印票面信息
print ('发票名称:',words_result['InvoiceTypeOrg'])
print ('发票代码:',words_result['InvoiceCode'])
print ('发票号码:',words_result['InvoiceNum'])
print ('开票日期:',words_result['InvoiceDate'])

print ('购方名称:',words_result['PurchaserName'])
print ('购方纳税人识别号:',words_result['PurchaserRegisterNum'])
print ('购方地址及电话:',words_result['PurchaserAddress'])
print ('购方开户行及账号:',words_result['PurchaserBank'])

print ('销售方名称:',words_result['SellerName'])
print ('销售方纳税人识别号:',words_result['SellerRegisterNum'])
print ('销售方地址及电话:',words_result['SellerAddress'])
print ('销售方开户行及账号:',words_result['SellerBank'])

print ('货物名称:',words_result['CommodityName'])
print ('规格型号:',words_result['CommodityType'])
```

```
print ('单位:',words_result['CommodityUnit'])
print ('数量:',words_result['CommodityNum'])
print ('单价:',words_result['CommodityPrice'])
print ('金额:',words_result['CommodityAmount'])
print ('税率:',words_result['CommodityTaxRate'])
print ('税额:',words_result['CommodityTax'])

print ('合计金额:',words_result['TotalAmount'])
print ('合计税额:',words_result['TotalTax'])
print ('价税合计:',words_result['AmountInFiguers'])

print ('备注:',words_result['Remarks'])
print ('收款人:',words_result['Payee'])
print ('复核:',words_result['Checker'])
print ('开票人:',words_result['NoteDrawer'])
```

调用SDK

```
client = AipOcr(conf.pw['appId'], conf.pw['ak'], conf.pw['sk'])
//可选参数
options={}
options['type']='roll'
//本地文件识别
client.vatInvoice(open('resources/vat.jpeg', 'rb').read(),options=options)

//url识别
client.vatInvoiceUrl("http://test.jpg",options=options)

//本地pdf文件识别
client.vatInvoicePdf(open('test.pdf', 'rb').read(),options=options)
})
```

```
from aip import AipOcr
import pandas as pd

#填写官网注册的API信息
APP_ID="16756034"
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

client=AipOcr(APP_ID,API_Key,Secret_Key)

#本地文件识别
response=client.vatInvoice(open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\发票_1.jpg", 'rb').read())

# print(response)

#提取列表
words_result = response['words_result']
# print(words_result)

#把票面信息写入列表list
```

```
list=
[[words_result['InvoiceTypeOrg'],words_result['InvoiceCode'],words_result['Invoi
ceNum'],words_result['InvoiceDate'],words_result['PurchaserName'],words_result['
PurchaserRegisterNum'],words_result['PurchaserAddress'],words_result['PurchaserB
ank'],words_result['SellerName'],words_result['SellerRegisterNum'],words_result[
'SellerAddress'],words_result['SellerBank'],words_result['CommodityName'],words_
result['CommodityType'],words_result['CommodityUnit'],words_result['CommodityNum
'],words_result['CommodityPrice'],words_result['CommodityAmount']
,words_result['CommodityTaxRate'],words_result['CommodityTax'],words_result['Tot
alAmount'],words_result['TotalTax'],words_result['AmountInFiguers'],words_result
['Remarks'],words_result['Payee'],words_result['Checker'],words_result['NoteDraw
er']]]

#读取本地Excel
write = pd.ExcelWriter("发票批量识别.xlsx")

#把list转换为DataFrame格式，传给df
df = pd.DataFrame(list)

#给df添加列标题
df.columns = ['发票名称','发票代码','发票号码','开票日期','购方名称','购方纳税人识别
号','购方地址及电话','购方开户行及账号','销售方名称','销售方纳税人识别号','销售方地址及电
话','销售方开户行及账号','货物名称','规格型号','单位','数量','单价','金额','税率','税
额','合计金额','合计税额','价税合计','备注','收款人','复核','开票人']

#把df导入本地Excel
df.to_excel(write,'Sheet1')

#保存Excel
write.save()


#打印票面信息
print ('发票名称:',words_result['InvoiceTypeOrg'])
print ('发票代码:',words_result['InvoiceCode'])
print ('发票号码:',words_result['InvoiceNum'])
print ('开票日期:',words_result['InvoiceDate'])

print ('购方名称:',words_result['PurchaserName'])
print ('购方纳税人识别号:',words_result['PurchaserRegisterNum'])
print ('购方地址及电话:',words_result['PurchaserAddress'])
print ('购方开户行及账号:',words_result['PurchaserBank'])

print ('销售方名称:',words_result['SellerName'])
print ('销售方纳税人识别号:',words_result['SellerRegisterNum'])
print ('销售方地址及电话:',words_result['SellerAddress'])
print ('销售方开户行及账号:',words_result['SellerBank'])

print ('货物名称:',words_result['CommodityName'])
print ('规格型号:',words_result['CommodityType'])
print ('单位:',words_result['CommodityUnit'])
print ('数量:',words_result['CommodityNum'])
print ('单价:',words_result['CommodityPrice'])
print ('金额:',words_result['CommodityAmount'])
print ('税率:',words_result['CommodityTaxRate'])
print ('税额:',words_result['CommodityTax'])

print ('合计金额:',words_result['TotalAmount'])
```

```
print ('合计税额:',words_result['TotalTax'])
print ('价税合计:',words_result['AmountInFiguers'])

print ('备注:',words_result['Remarks'])
print ('收款人:',words_result['Payee'])
print ('复核:',words_result['Checker'])
print ('开票人:',words_result['NoteDrawer'])
```

火车票识别

支持对红、蓝火车票的13个关键字段进行结构化识别，包括车票号码、始发站、目的站、车次、日期、票价、席别、姓名、座位号、身份证号、售站、序列号、时间。

核心代码：

调用API

```
# encoding:utf-8

import requests
import base64

'''
火车票识别
'''

request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/train_ticket"
# 二进制方式打开图片文件
f = open('[本地文件]', 'rb')
img = base64.b64encode(f.read())

params = {"image":img}
access_token = '[调用鉴权接口获取的token]'
request_url = request_url + "?access_token=" + access_token
headers = {'content-type': 'application/x-www-form-urlencoded'}
response = requests.post(request_url, data=params, headers=headers)
if response:
    print (response.json())
```

调用SDK

```
client = AipOcr(conf.pw['appId'], conf.pw['ak'], conf.pw['sk'])
//可选参数
options={}
options['type']='roll'
//本地文件识别
client.trainTicket (open('resources/vat.jpeg', 'rb').read(),options=options)

//url识别
client.trainTicketUrl("http:/test.jpg",options=options)
})
```

```
from aip import AipOcr
import pandas as pd

#填写官网注册的API信息
```

```
APP_ID="16756034"
API_Key ="eAIbqQ1oN14Rr521v5xH"
Secret_Key ="dflMjOV7RvgqnlyPZDkIVip2PF5G"

client=AipOcr(APP_ID,API_Key,Secret_Key)

#本地文件识别
response=client.trainTicket(open(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\文字识别\火车票_1.jpg", 'rb').read())

# print(response)

#提取列表
words_result = response['words_result']
# print(words_result)

#把票面信息写入列表list
list=
[[words_result['serial_number'],words_result['starting_station'],words_result['destination_station'],words_result['train_num'],words_result['date'],words_result['time'],words_result['ticket_rates'],words_result['seat_category'],words_result['name'],words_result['seat_num'],words_result['id_num'],words_result['sales_station'],words_result['serial_number']]]

#读取本地Excel
write = pd.ExcelWriter("火车票批量识别.xlsx")

#把list转换为DataFrame格式，传给df
df = pd.DataFrame(list)

#给df添加列标题
df.columns = ['票号码','始发站','目的站','车次','日期','时间','票价','席别','姓名','座位号','身份证号','售站','序列号']

#把df导入本地Excel
df.to_excel(write,'Sheet1')

#保存Excel
write.save()
```

iOCR自定义模板文字识别

iOCR通用版

iOCR 通用版是 iOCR 自定义模板文字识别针对通用场景下固定版式的卡证票据、文件资料提供的一款 OCR 定制化产品，您仅需上传一张模板图片，即可通过框选参照字段及识别区快速制作结构化识别模型；同时，还可针对制作的多个模板训练自定义分类器，一步完成图片的自动分类和结构化识别。

iOCR财会版

iOCR 财会版是 iOCR 自定义模板文字识别针对财会报销场景提出的专项解决方案，预置多种财务场景常用识别模板及财务票据分类器，无需制作或训练即可直接使用；并提供混贴票据识别功能，可对粘贴在一张报销单上的多张不同种类发票进行切分识别；同时支持对未预置的固定版式票据可定制结构化识别模板和分类器。

OCR文字识别总结

- 通过Python调用百度AI平台文字识别接口，可以很容易的实现文字识别的各种功能，很强大
- 百度AI平台文字识别接口有免费调用次数，个人使用的话，完全够用
- 问题思考：

  1、增值税发票识别后的列表、字典格式数据再处理，正则表达式、列表、字典等方法

  2、如何批量识别增值税发票?

  3、银行流水/出库单等纸质单据的OCR文字识别，内容格式化输出

  4、输出格式：Excel/CSV/Word/TXT

软件操作流程自动化

- 财务人员：报税、操作财务软件
- 审计人员：操作审计软件
- 其他.....等等

pyautogui库

pyautogui是一个第三方模块库，也是一个纯Python的GUI自动化工具，其目的是可以用程序自动控制鼠标和键盘操作。

安装

```
pip install pyautogui
```

鼠标操作

```
import pyautogui

#2秒钟鼠标移动坐标为100,100位置   绝对移动
pyautogui.moveTo(100, 100,2)

#鼠标单击左键一次
pyautogui.click()

#鼠标双击左键一次
pyautogui.doubleClick()

#鼠标单击右键一次
pyautogui.rightClick()

#鼠标单击中键一次
pyautogui.middleClick()

#鼠标当前位置滚轮滚动
pyautogui.scroll()
```

实时获取位置坐标

- **微信截图：** `Alt+A`

- **QQ截图：** `Ctrl`+`Alt`+`A`
- **通过position()方法实时获取位置坐标**

```
import os,time
import pyautogui as pag
while True:
      print ("Press Ctrl-C to end")
      x,y = pag.position()
      pos="Position:"+str(x).rjust(4)+','+str(y).rjust(4)
      print (pos)
      time.sleep(0.3)
      os.system('cls')
print ('ending....')
```

键盘操作

```
import pyautogui

#模拟输入信息
pyautogui.typewrite(message='Hello world!',interval=0.5)

#点击回车键
pyautogui.press('enter')

#模拟组合热键
pyautogui.hotkey('ctrl', 'v')
```

截图操作

```
import pyautogui

#整个屏幕截图
im1 = pyautogui.screenshot()
#保存截图
im1.save('my_screenshot.png')

#整个屏幕截图并保存
im2 = pyautogui.screenshot('my_screenshot2.png')

#局部截图
img = pyautogui.screenshot(region=[522,175,938,726]) # x,y,w,h
#保存截图
img.save('{}.jpg'.format(y))
```

输入中文

pyautogui.typewrite不能输入汉字，只能输入功能键，数字，字母。

可以用pyperclip模块实现自动输入中文。

pyperclip模块中有两个函数，分别是copy()和paste()，copy()用于向计算机的剪贴板发送文本，paste()用于从计算机剪贴板接收文本。

安装pyperclip模块：

```
pip install pyperclip
```

操作举例：

```
import pyautogui
import pyperclip
import time

#复制文字
pyperclip.copy('中国')

#中间最好停顿2秒，以防粘贴失败
time.sleep(2)

#粘贴文字
pyautogui.hotkey('ctrl','v')
```

通过pywin32快速打开外部程序或文件

pywin32是一个第三方模块库，主要的作用是方便python开发者快速调用windows API。

安装

```
pip install pywin32
```

如果用pyautogui打开外部程序或文件，我们需要先定位坐标、再进行双击操作，整个过程实现起来并不难，但是缺点在于，如果程序或文件的位置发生稍微的变化，就会导致无法打开指定程序或文件。

为了解决这一问题，我们可以导入win32api模块，通过ShellExecute函数快速打开外部程序或文件，不再局限于位置坐标。

```
import win32api

#打开某文件
win32api.ShellExecute(1, 'open', r'某文件完整路径', '', '', 1)
```

自动化操作审计软件（案例实战）

用Python自动化操作审计大师软件，批量新建审计项目。

```
#导入模块
import openpyxl
import pyautogui
import time
import pyperclip

#双击打开审计软件
pyautogui.moveTo(487, 435)
pyautogui.doubleClick()
time.sleep(5)

#点击确定
pyautogui.moveTo(580, 516)
```

```
pyautogui.click()
time.sleep(1)

wb=openpyxl.load_workbook(r"E:\我的文件夹\我的开发\培训开发\财务编程系列课程\财务编程之Python\机器人流程自动化\新建工作表.xlsx")
sheet=wb.worksheets[1]

for i in range(2,7):

    #点击审计项目管理
    pyautogui.moveTo(98, 79)
    pyautogui.click()
    time.sleep(1)

    #点击新建审计项目
    pyautogui.moveTo(442, 120)
    pyautogui.click()
    time.sleep(1)

    #点击单位名称输入框
    pyautogui.moveTo(607, 275)
    pyautogui.click()
    time.sleep(1)

    #输入单位名称
    pyperclip.copy(sheet.cell(i,1).value)
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(1)

    #填写报表会计期间
    #起始日期
    pyautogui.moveTo(578, 360)
    pyautogui.dragTo(510, 360,0.5,button='left') # 按住鼠标左键，用0.5秒钟把鼠标拖拽到(510, 360)位置
    pyautogui.press('delete')
    time.sleep(1)
    pyperclip.copy('2020-01-01')
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(1)

    #截止日期
    pyautogui.moveTo(765, 360)
    pyautogui.click()
    pyperclip.copy('2020-12-31')
    pyautogui.hotkey('ctrl', 'v')

    #点击下一步，点击下一步
    pyautogui.moveTo(690, 512)
    pyautogui.click()
    time.sleep(1)
    pyautogui.moveTo(690, 512)
    pyautogui.click()

    #选择审计模板
    pyautogui.moveTo(911, 347)
    pyautogui.click()
    time.sleep(1)
    pyautogui.moveTo(652, 460)
```

```
    pyautogui.click()

    #点击下一步
    pyautogui.moveTo(690, 512)
    pyautogui.click()
    time.sleep(1)

    #点击确定
    pyautogui.moveTo(844, 451)
    pyautogui.click()
```

自动化申报个税（案例实战）

```
#导入模块
import pyautogui
import time
import pyperclip

n=0

while True:

    #综合所得申报
    pyautogui.moveTo(1497, 346)
    pyautogui.click()
    time.sleep(1)

    #综合所得申报
    pyautogui.moveTo(1756, 356)
    pyautogui.click()
    time.sleep(1)

    #确定
    pyautogui.moveTo(957, 644)
    pyautogui.click()
    time.sleep(1)

    #确定
    pyautogui.moveTo(912, 609)
    pyautogui.click()
    time.sleep(1)

    #确定
    pyautogui.moveTo(662, 469)
    pyautogui.click()
    time.sleep(1)

    #立即复制数据
    pyautogui.moveTo(913, 660)
    pyautogui.click()
    time.sleep(1)

    #关闭
    pyautogui.moveTo(1222, 703)
```

```
    pyautogui.click()
    time.sleep(1)

    #综合所得申报
    pyautogui.moveTo(389, 129)
    pyautogui.click()
    time.sleep(1)

    #税款计算
    pyautogui.moveTo(844, 196)
    pyautogui.click()
    time.sleep(20)

    #附表填写
    pyautogui.moveTo(1282, 193)
    pyautogui.click()
    time.sleep(1)

    #申报表报送
    pyautogui.moveTo(1689, 194)
    pyautogui.click()
    time.sleep(2)

    #发送申报
    pyautogui.moveTo(1818, 315)
    pyautogui.click()
    time.sleep(20)

    #立即获取
    pyautogui.moveTo(897, 587)
    pyautogui.click()
    time.sleep(1)

    #确定
    pyautogui.moveTo(959, 609)
    pyautogui.click()
    time.sleep(1)

    n=n+1

    if n<3:

        #单位管理
        pyautogui.moveTo(1732, 64)
        pyautogui.click()
        time.sleep(1)

        pyautogui.moveTo(1495, 671)
        #点击滑块
        for i in range(0,n):
            pyautogui.click()
            time.sleep(0.5)
            pyautogui.click()
            time.sleep(0.5)

        #进入
        pyautogui.moveTo(1438, 510)
        pyautogui.click()
```

```
        time.sleep(2)

        #点击密码输入框
        pyautogui.moveTo(885, 570)
        pyautogui.click()
        time.sleep(1)

        #输入密码
        pyperclip.copy('密码')
        pyautogui.hotkey('ctrl','v')
        time.sleep(1)

        #关闭
        pyautogui.moveTo(1216, 804)
        pyautogui.click()
        time.sleep(1)

        #关闭
        pyautogui.moveTo(1013, 772)
        pyautogui.click()
        time.sleep(8)

        continue

    else:

        break
```

自动化操作用友软件（案例实战）

```
#导入模块
import pyautogui
import time
import pyperclip
import openpyxl


wb=openpyxl.load_workbook(r"U9采购订单.xlsx")
sheet=wb.worksheets[0]

for i in range(4,7):

    #单据类型
    time.sleep(1)
    pyautogui.moveTo(254,282)
    pyautogui.click()
    time.sleep(5)

    #标准订单
    pyautogui.moveTo(686,331)
    pyautogui.doubleClick()
    time.sleep(2)

    #供应商
    pyautogui.moveTo(254,333)
    pyautogui.click()
```

```
    time.sleep(3)

    #点击第一个名称
    pyautogui.moveTo(746,243)
    pyautogui.doubleClick()
    time.sleep(2)

    #点击料号放大镜
    pyautogui.moveTo(481,494)
    pyautogui.click()
    time.sleep(1)
    pyautogui.click()
    time.sleep(2)
    pyautogui.moveTo(540,494)
    pyautogui.click()
    time.sleep(2)
    pyautogui.click()
    time.sleep(5)

    #双击第一个品名
    pyautogui.moveTo(727,324)
    pyautogui.doubleClick()
    time.sleep(3)

    #需求数量
    pyautogui.moveTo(720,492)
    pyautogui.doubleClick()
    time.sleep(1)
    pyperclip.copy(sheet.cell(i,15).value)
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(1)
    pyautogui.moveTo(790,492)
    pyautogui.click()
    time.sleep(1)

    #向右滑动
    pyautogui.moveTo(1324,644)
    pyautogui.click()
    time.sleep(1)

    #最终价
    pyautogui.moveTo(563,493)
    pyautogui.click()
    time.sleep(1)
    pyperclip.copy(sheet.cell(i,17).value)
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(1)
    pyautogui.moveTo(490,586)
    pyautogui.click()
    time.sleep(1)

    #保存
    pyautogui.moveTo(20,220)
    pyautogui.click()
    time.sleep(2)

    #提交
    pyautogui.moveTo(201,220)
```

```
    pyautogui.click()
    time.sleep(2)

    #审核
    pyautogui.moveTo(274,220)
    pyautogui.click()
    time.sleep(2)

    #新增
    pyautogui.moveTo(89,220)
    pyautogui.click()
    time.sleep(3)
```

软件操作流程自动化总结

- 根据人工操作，设计代码的整体思路
- 找准坐标
- 写完一步，测试一步
- 每次点击操作之后，最后停顿1-2s，根据情况设置
- 测试发生异常时，可以按组合键`Ctrl`+`Alt`+`Delete`终止程序运行
- 会操作一种软件的某些流程，就会操作其他软件的所有流程，原理一样，要学会触类旁通
- 屏幕分辨率改变，程序会运行失效