# 内建函数

内建函数参考http://www.runoob.com/python/python-built-in-functions.html

print('Hello world!') ------------------------ 括号内的字符串显示在屏幕上

print('Hello', end='')  
print('World') 输出就会像这样：HelloWorld ------------------ 把多行变成一行。

# 传递一个空字符串，这样print函数不会在字符串末尾添加一个换行符，而是添加一个空字符串，其实这也是一个语法要求，表示这个语句没结束。

print('cats', 'dogs', 'mice', sep=',')  
输出结果 cats,dogs,mice ------------------ #sep 关键字参数， 替换掉默认的分隔字符串

my = input(‘>’) -------------------------- 等待用户在键盘上输入一些文本，并按下回车键

len() ------------ 计算字符串长度

str() --------------------- 字符串

int() ---------------------- 数字整数

float() ----------------- 浮点数

break ------------------- 终止整个循环

continue ---------------- 结束本次循环

range() ------------------- 可迭代对象

min(5,10) ------------ 返回参数里的的最小值（就是返回5）

zip() ----------- 将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表,在 Python 3.x 中为了减少内存，zip() 返回的是一个对象。如需展示列表，需手动 list() 转换。

转义字符包含一个倒斜杠（\）， 紧跟着是想要添加到字符串中的字符

字符串开始的引号之前加上 r， 使它成为原始字符串。“原始字符串” 完全忽略所有的转义字符， 打印出字符串中所有的倒斜杠。如下：print(r'That is Carol\'s cat.')

**字符串方法**：

upper() ------------ 返回的字符串全部大写

lower() ------------ 返回的字符串全部小写

isupper() ----------- 检测是不是全部大写，如果是返回真否则返回假

islower() ----------- 检测是不是全部小写，如果是返回真否则返回假

isalpha() --------- 如果字符串只包含字母，并且非空返回 True

isalnum() ---------- 如果字符串只包含字母和数字，并且非空返回 True

isdecimal() -------- 如果字符串只包含数字字符，并且非空返回 True

isspace() --------- 如果字符串只包含空格、制表符和换行，并且非空返回 True

istitle() ---------- 如果字符串仅包含以大写字母开头、后面都是小写字母的单词返回 True

startswith() ------ 检测是不是以什么开头，是返回真，否则返回假

endswith() -------- 检测以什么结尾，是返回真，否则返回假

join() ------------ 将字符串列表连接起来，成为一个单独的字符串

split() ------------ 对一个字符串调用，返回一个字符串列表

rjust() ------------ 通过插入空格来对齐文本右对齐,如：'Hello'.rjust(20, '\*')返回'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Hello'

ljust() ----------- 通过插入空格来对齐文本左对齐,'Hello'.ljust(20, '-')返回'Hello---------------'

center() -------- 居中对齐'Hello'.center(20, '=')返回'=======Hello========'

strip() ------------- 删除字符串两边的空白符

rstrip() -------- 删除右边的空白符

lstrip() --------- 删除左边的空白符

list() -------------------- **列表 [ ]**，如：print(list((1,2,3,4,5))) 打印出来的是列表，一般用来转换字符串

列表的操作方法：del fan[1]从列表中删除下标1的值

apam.remove(‘fan’) ----------------- 从列表删除fan这个值

spam.index(‘fan’) -------------- 传入一个值，如果该值存在于列表中，就返回它的下标。如果该值不在列表中，Python 就报 ValueError

spam.append(‘fan’) --------------- 将值插入列表末尾

spam.insert(1,’fan’) --------------- 在什么位置后面插入，表示在下标1后面插入新值

spam.sort() --------------------- 排序

tuple() --------------- **元组**（），如print(tuple((1,2,3,4,5)))打印出来是元组,一般用来转换字符串

dict() ------------- 字典{ }

spam.keys() --------------- 字典的键

spam.values() -------------- 字典的值

spam.items() ---------- 字典的键和值

spam = {'color': 'red', 'age': 42}

for k, v in spam.items():

print('Key: ' + k + ' Value: ' + str(v)) #输出结果如下：

Key: age Value: 42

Key: color Value: red

spam.get('cups', 0) ------------------它有两个参数：要取得其值的键，以及如果该键不存在时，返回的备用值

spam.setdefault('color', 'black') ------------传递给该方法的第一个参数，是要检查的键。第二个参数，是如果该键不存在时要设置的值。如果该键确实存在，方法就会返回键的值

.replace() --------- 把字符串中的 old[旧字符串]替换成new[新字符串]，如果指定第三个参数，则表示替换不超过多少次

.setdefault() ----- Python 字典的方法, ,返回指定键的值，如果键不在字典中，将会添加键并将值设置为一个指定值，默认为None

raise --------------------- 在代码中抛出自定义的异常内容,如下提示

*raise* Exception('This is the error message.')

Traceback (most recent call last):

File "E:/System/PyCharm/自动化练习/010/1.调试.py", line 24, in <module>

raise Exception('This is the error message.')

Exception: This is the error message.

使用 try 和 except 语句，你可以更优雅地处理错误，而不是让整个程序崩溃

*def* boxPrint(*symbol*, *width*, *height*):

*if* len(*symbol*) != 1:  
 *raise* Exception('Symbol must be a single character string.')  
 *if width* <= 2:  
 *raise* Exception('Width must be greater than 2.')  
 *if height* <= 2:  
 *raise* Exception('Height must be greater than 2.')  
 print(*symbol* \* *width*)  
 *for* i *in* range(*height* - 2):  
 print(*symbol* + (' ' \* (*width* - 2)) + *symbol*)  
 print(*symbol* \* *width*)  
*for* sym, w, h *in* (('\*', 4, 4), ('O', 20, 5), ('x', 1, 3), ('ZZ', 3, 3)):  
 *try*:  
 boxPrint(sym, w, h)  
 *except* Exception *as* err:  
 print('An exception happened: ' + str(err))

## 读写文件

读写文件有 3 个步骤：  
1．调用 open()函数， 返回一个 File 对象。  
2．调用 File 对象的 read()或 write()方法。  
3．调用 File 对象的 close()方法，关闭该文件

*with* open("E:\\System\\asweigart\\fangx.txt","r+",encoding="utf-8") *as* file:

print(file.readline())  
 print(file.read())

readline() --------- 读取一行,readlines是读取成列表。列表中的每个字符串就是文本中的每一行

write() ------- 写入文件,不会换行，必须手动写换行符(\n)

round(2.13455, 2) ------ 指定的精度四舍五入到一个浮点数,第一个数带有小数点的数字，第二个数是需要保留几位小数点

# math模块:数学运算相关函数

# shelve模块:保存变量

# random模块：随机数相关函数

random.randint(1,10) ------------- 随机返回1到10之间的数字

# sys模块

sys.exit() ---------------------- 让程序终止或者退出

sys.argv --------------- 获取用户输入的参数,然后把参数变成列表,位置从0开始(本身名，输入的参数1…)

*if* len(sys.argv) < 2: #排断参数有几个

print('使用方法: cmd命令窗口输入python .\password.py 账号')  
 sys.exit()  
account = sys.argv[1] #定义变量account 等于第一个输入的参数

# copy模块

copy.copy() -------------------- 可以用来复制列表或字典这样的可变值

copy.deepcopy() ------------------- 如果列表里包含了列表，复制的时候需要用这个函数

# pprint 模块:漂亮打印

pprint.pprint() --------------- 输出看起来更干净，通过键排序

pprint.pformat() ---------------- 得到漂亮打印的文本作为字符串， 而不是显示在屏幕上

# time模块：

time.sleep(3) ------- 等待多长时间

# pyperclip模块：拷贝粘贴字符串

pyperclip.copy() ------------------- 复制文本到剪贴板

pyperclip.paste() ----------------------- 粘贴剪贴板文体

pprint.pformat() ----------------- 函数保存变量

# re模块:所有正则表达式的函数

常用的字符分类缩写代码：加上+号多少匹配多个

\d 0到9的任何数字

\D 除了0到9的数字之外的任何字符

\w \w 任何字母、数字或下划线字符（可以认为是匹配“单词”字符）  
\W 除字母、数字和下划线以外的任何字符  
\s 空格、制表符或换行符（可以认为是匹配“空白”字符）  
\S 除空格、制表符和换行符以外的任何字符

^ 开始处插入,表示最开头匹配

$ 结尾处插入,表示已什么结尾

. 通配符,他匹配除换行之外的所有字符,只能匹配一个

.\* 匹配所有的字符串

re.compile() ------------------ 传入字符串值，表示正则表达式，它将返回一个正则表达式对象

.search() --------------------- 正则表达式对象的查找方法,查找传入的字符串,返回相匹配对象,

re.DOTALL ---------- 作为 re.compile()的第二个参数， 可以让句点字符匹配所有字符， 包括换行字符

re.VERBOSE ---------- 在search里传入的第二个参数,把正则表达式写成多行，并且自动忽略空格

re.IGNORECASE 或 re.I --------- 在search里传入的第二个参数,忽略大小写

以上3个如果想一起使用，需要用|管道符如：a= re.compile('foo', re.IGNORECASE | re.DOTALL | re.VERBOSE)

.findall() ----------- 可以找到所有匹配的地方,返回字符串列表

.group() ------------------- 返回被查找字符串中实际匹配的文本,如下例子：

phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d') #正则表达式

mo = phoneNumRegex.search('My number is 415-555-4242.') #查找传入的数字  
print('Phone number found: ' + mo.group()) #返回实际配置的内容

返回如下：Phone number found: 415-555-4242

#-------------findall

phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d') # has no groups  
mo7=phoneNumRegex.findall('Cell: 415-555-9999 Work: 212-555-0000')  
print(mo7)

groups() -------------- 一次获取所有的分组,在compile里用（）分组，第一对是第一组，依次类推

sub() --------------- 返回回替换完成后的字符串,需要传入两个参数。第一个参数是一个字符串用于取代发现的匹配。第二个参数是一个字符串，即正则表达式。

**正则表达式符号特别介绍：**

• ?匹配零次或一次前面的分组。  
• \*匹配零次或多次前面的分组。  
• +匹配一次或多次前面的分组。  
• {n}匹配 n 次前面的分组。  
• {n,}匹配 n 次或更多前面的分组。  
• {,m}匹配零次到 m 次前面的分组。  
• {n,m}匹配至少 n 次、至多 m 次前面的分组。  
• {n,m}?或\*?或+?对前面的分组进行非贪心匹配。  
• ^spam 意味着字符串必须以 spam 开始。  
• spam$意味着字符串必须以 spam 结束。  
• .匹配所有字符，换行符除外。  
• \d、 \w 和\s 分别匹配数字、单词和空格。  
• \D、 \W 和\S 分别匹配出数字、单词和空格外的所有字符。  
• [abc]匹配方括号内的任意字符（诸如 a、 b 或 c）。  
• [^abc]匹配不在方括号内的任意字符。

# os模块：作系统的功能接口的模块

参考：http://www.runoob.com/python/os-file-methods.html

os.path.abspath( path ) ----------------- 返回参数的绝对路径，可以将相对路径转换为绝对路径

os.path.isabs( path ) -------------如果参数是绝对路径就返回True,如果是相对路径就返回False

os.path.relpath( path start )-------将返回从start路径到path的相对路径的字符串。如果没有提供start，就使用当前工作目录作为开始路径。

os.path.dirname(path) --------- 将返回一个字符串，它包含path参数中最后一个斜杠之前的所有内容。

os.path.basename(path) -------- 将返回一个字符串，它包含path参数中最后一个斜杠之后的所有内容。

os.path.split(path) ------------------- 返回2个字符串元组,路径的目录名称和基本名称(对照上面2个参数

os.getcwd()-----------------------------取得当前工作路径

os.chdir(c:\\windows) -----------------------------改变当前工作路径，变成c:\windows

os.path.join('fan','fan')---------------------------------拼接文件路径,返回 fan\fan

os.makedirs('E:\\System\\PyCharm\\fan')---------------------------创建多层目录(文件夹fan

os.mkdir(path) ------------------------------------- 创建文件夹

os.path.getsize(path)----------------------------将返回 path 参数中文件的大小

os.listdir(path)-------------------------------------将返回文件名字符串的列表

os.path.exists(path)------------path 参数所指的文件或文件夹存在将返回 True，否则返回 False。

os.path.isfile(path)---------------如果 path 参数存在并且是一个文件,将返回 True， 否则返回 False。

os.path.isdir(path)--------------如果 path 参数存在并且是一个文件夹，将返回 True，否则返回 False。

os.unlink(path)-----------------------------将删除 path 处的文件

os.rmdir(path)-------------------------将删除 path 处的文件夹。该文件夹必须为空，其中没有任何文件和文件夹。

os.walk()----------------遍历目录树

*import* os

*for* folderName, subfolders, filenames *in* os.walk('E:\\System\\PyCharm\\自动化练习\\其他\\时尚'):  
 print(folderName)  
 *for* sub *in* subfolders:  
 print(folderName + '\\'+ sub)  
 *for* file *in* filenames:  
 print(folderName + '\\' + file)  
 print('')

# shutil模块: Python 程序中复制\移动\改名和删除文件

shutil.copy(source, destination)------------------------将路径 source 处的文件复制到路径 destination处的文件夹

shutil.copytree(source, destination) --------- 复制整个文件夹，以及它包含的文件夹和文件

shutil.move('C:\\bacon.txt', 'C:\\eggs') ------------------------将 C:\bacon.txt 移动到文件夹 C:\eggs 中

shutil.rmtree(path)------------------将删除 path 处的文件夹，它包含的所有文件和文件夹都会被删除。

# zipfile模块:压缩文件

zip = zipfile.ZipFile('gx.zip')

print(zip.namelist())-------------------读取ZIP压缩包的内容

info = zip.getinfo(’spam.txt’) ----------------------file\_size表示原来文件大小和compress\_size表示压缩后文件大小

print(info. file\_size) print(info. compress\_size)

zip.extractall()------------------------------解压缩

newZip = zipfile.ZipFile('new.zip', 'w')

newZip.write('22', compress\_type=zipfile.ZIP\_DEFLATED)------------------新加压缩包，添加的话吧w换成a，

Zip.close() ------------------- 关闭压缩包

# traceback模块：异常的获取与处理

只要抛出的异常没有被处理， Python 就会显示反向跟踪

traceback.print\_exc() ----------- 是对异常栈输出

traceback.format\_exc() -------------- 是把异常栈以字符串的形式返回，print(traceback.format\_exc()) 就相当于traceback.print\_exc()

不是让程序在异常发生时就崩溃，可以将反向跟踪信息写入一个日志文件，并让程序继续运行。稍后，在准备调试程序时，可以检查该日志文件。

*import* traceback

*try*:  
 *raise* Exception('This is the error message.')  
*except*:  
 errorFile = open('errorInfo.txt', 'w')  
 errorFile.write(traceback.format\_exc())  
 errorFile.close()  
 print('The traceback info was written to errorInfo.txt.')

# logging模块：自定义日志记录

logging.basicConfig --------------- 指定日志格式

**logging.basicConfig函数各参数：**

filename：指定日志文件存放位置和保存名称；

filemode：和file函数意义相同，指定日志文件的打开模式，'w'或者'a'；

format：指定输出的格式和内容，format可以输出很多有用的信息，

%(levelno)s：打印日志级别的数值

%(levelname)s：打印日志级别的名称

%(pathname)s：打印当前执行程序的路径，其实就是sys.argv[0]

%(filename)s：打印当前执行程序名

%(funcName)s：打印日志的当前函数

%(lineno)d：打印日志的当前行号

%(asctime)s：打印日志的时间

%(thread)d：打印线程ID

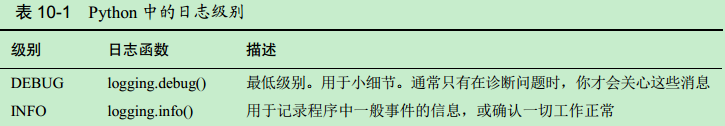
%(threadName)s：打印线程名称

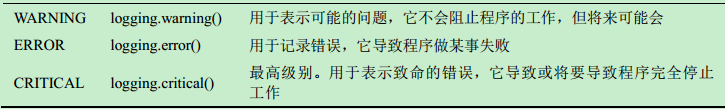
%(process)d：打印进程ID

%(message)s：打印日志信息

datefmt：指定时间格式，同time.strftime()；

level：设置日志级别，默认为logging.WARNNING；级别有：debug、info、warning、error、critical





stream：指定将日志的输出流，可以指定输出到sys.stderr，sys.stdout或者文件，默认输出到sys.stderr，当stream和filename同时指定时，stream被忽略；

logging.debug() ------------- 打印日志信息

logging.disable(logging.CRITICAL) ----------- 禁用日志消息，传一个日志级别，它就会禁止该级别和更低级别的所有日志消息, logging.CRITICAL禁用所有级别的日志,放在最上面禁用所有

*import* logging

# logging.disable(logging.CRITICAL)  
logging.basicConfig(filename=r'E:\System\fanx\log.log',level=logging.DEBUG, format=' %(asctime)s - %(levelname)s - %(message)s') #把日志放在E:\System\fanx\log.log里  
*def* factorial(*n*):  
 logging.debug('Start of factorial(%s%%)' % (*n*))  
 total = 1  
 *for* i *in* range(1,*n* + 1):  
 total = total \* i  
 logging.debug('i is ' + str(i) + ', total is ' + str(total))  
 logging.debug('End of factorial(%s%%)' % (*n*))  
 *return* total  
print(factorial(4))  
#-----------------------------  
logging.info("Start print log")  
logging.debug("Do something")  
logging.warning("Something maybe fail.")

# webbrowser：打开浏览器获取指定页面。

webbrowser.open(r'https://www.cnblogs.com/') -------- 启动默认浏览器,打开指定的网站(博客园)

# requests：从因特网上下载文件和网页。

requests.get('https://api.github.com/events') ---------下载一个网页内容

res = requests.get('https://www.cnblogs.com/')

res.raise\_for\_status() #下载文件出错，会报错,如果不加默认会继续执行

print(res.text[:1000]) #返回html源代码

print(res.encoding) #查看返回文本编码,一般都会自适应

res.encoding = 'GBK' #可修改返回的编码格式

将文件保存下来的方式

.iter\_content() ------------- 可用于for循环,每次迭代返回一段内容

res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt') #下载的文件

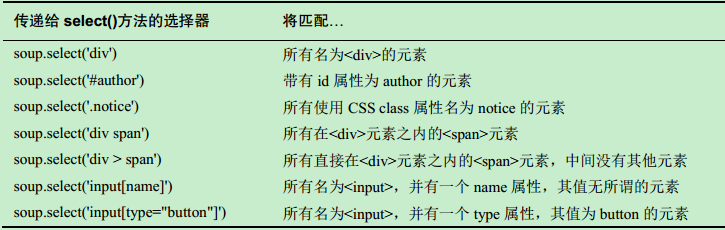
res.raise\_for\_status() #如果访问失败报错  
file = open(r"E:\System\fanx\file.txt","wb") #打开file.txt文本，用二进制方式写入  
*for* i *in* res.iter\_content(100000): #循环写入,如果文本特别大，最好用循环，里面的100000代表一次写入多少个字节  
 file.write(i) #开始写入  
file.close() #关闭文本

# Beautiful Soup：解析 HTML,即网页编写的格式。

用于这个比正则表达式好很多,BeautifulSoup 模块的名称是 bs4（表示 Beautiful Soup，第 4 版）,虽然安装时使用的名字是 beautifulsoup4，但要导入它，就使用 import bs4

bs4.BeautifulSoup() -------------- 传入将要解析的HTML

.select() --------- 方法寻找元素



res = requests.get(r"https://www.cnblogs.com/") #下载的网页

res.raise\_for\_status() #如果访问不通就报错  
soup = bs4.BeautifulSoup(res.text,features="lxml") #将响应的结果传给bs4.BeautifulSoup()  
elem = soup.select("#cnts") #查找id属性为cnts的元素  
print(type(elem)) #返回的是一个列表  
print(len(elem)) #看到长度是1,表示只有一个值，只匹配到1次  
print(elem[0].getText()) #返回该元素的文本  
# print(elem[0].attrs) #attrs返回给一个字典，包含该元素的属性'id'，以及id属性的值'cnts'  
*for* i *in* elem:  
 print(i.getText())

# selenium：能够填写表单，并模拟鼠标在这个浏览器中点击。

导入 selenium 的模块不是 import selenium，而是要运行 from selenium import webdriver

*from* selenium *import* webdriver

chrome=webdriver.Chrome() #谷歌浏览器  
ie = webdriver.Ie() #IE浏览器  
firefox = webdriver.Firefox() #火狐浏览器  
edge = webdriver.Edge() #edge浏览器  
chrome.get("http://www.baidu.com/") #用谷歌浏览器打开百度

以上操作需要安装驱动，驱动下载地址如下,如谷歌下载驱动(chromedriver.exe),放在python安装目录下

chrome: <http://chromedriver.storage.googleapis.com/index.html> 和 <https://sites.google.com/a/chromium.org/chromedriver/downloads>

Edge: <https://github.com/mozilla/geckodriver/releases>

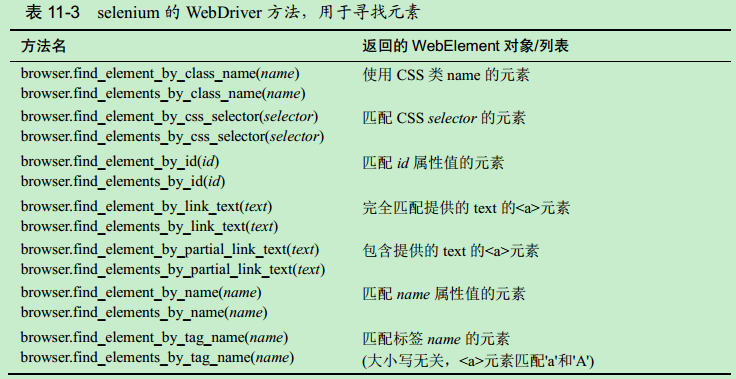
firefox: <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

IE: <https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver>

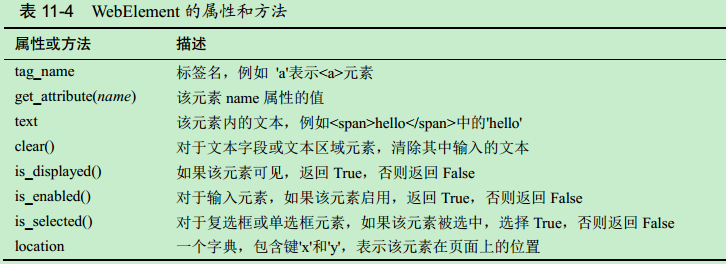
或官网下载：https://www.seleniumhq.org/download/

.find\_element\_\* -------- 返回web元素对象

.find\_elements\_\* ---------- 返回web元素对象的列表,包含页面所有元素



除了\*\_by\_tag\_name()方法，所有方法的参数都是区分大小写的,一旦有了 WebElement 对象,可以读取如下图中的属性，或调用其中的方法



注意：动态id就是第一次点击显示的id与二次点击显示的不一样，一般是元素属性中包含一段数字的这种情况。

类似这种：

1 <input type="button" value="选择" id="btncussel1452">

这种情况，个人觉得比较简单的定位方法是用XPath来根据部分元素属性定位即模糊定位：

有三种情况：

 1  driver.find\_element\_by\_xpath("//input[starts-with(@id,'btncusse')]").click()

 2  driver.find\_element\_by\_xpath("//input[contains(@id,'btncusse')]").click()

 3  driver.find\_element\_by\_xpath("//input[ends-with(@id,'btncusse')]").click()

解释一下：

btncusse:这个是id中不变的部分字符串；

starts-with：是id中以不变的部分字符串开头；contains：是id中包含不变的部分字符串；ends-with：是id中以不变的部分字符串结尾；

[]中括号前面，前面的input，是根据本条代码的Html标签决定的 。

browser.find\_element\_by\_xpath("//\*[text()='花呗套现']").click() == 通过文本定位元素

browser.find\_element\_by\_xpath("//\*[contains(text(),'花呗')]").click() == 通过部分文字定位

.switch\_to\_frame() ---------- 用来定位frame，可以传入id、name、index以及selenium的WebElement对象

.click() ---------- 模拟鼠标在元素上点击,个方法可以用于链接跳转，选择单选按钮，点击提交按钮

browser = webdriver.Chrome()

browser.get('http://www.baidu.com')  
link = browser.find\_element\_by\_link\_text('地图')  
link.click()

.send\_keys() ---- 向web提交表单内容

. submit() ----- 把表单中的信息，提交到后台

selenium.webdriver.common.keys ------ 针对不能用字符串值输入的键盘击键.由于这个模块名非常长，所以在程序顶部运行 from selenium.webdriver. common.keys import Keys 就比较容易。



更多的键盘输入方法可参考：https://www.cnblogs.com/mengyu/p/6942584.html

*from* selenium *import* webdriver

*from* selenium.webdriver.common.keys *import* Keys #针对不能用字符串值输入的键盘  
browser = webdriver.Chrome()  
browser.get('http://www.163.com')  
htmlElem = browser.find\_element\_by\_tag\_name('html')  
htmlElem.send\_keys(Keys.END) # 回到页面下面  
htmlElem.send\_keys(Keys.HOME) #回到首页

模拟点击浏览器的各种按钮

browser.back() --------- 点击“返回”按钮。

browser.forward() ----------- 点击“前进”按钮。

browser.refresh() ---------- 点击“刷新”按钮。

browser.quit() ---------------- 点击“关闭窗口”按钮

# openpyxl模块：读取和修改 Excel

*import* openpyxl

wb = openpyxl.load\_workbook(r"E:\System\fanx\example.xlsx") #打开excel表格  
print(wb.sheetnames) #查看所有工作表名称，返回列表  
print(wb.worksheets) #查看所有workbook对象名,返回列表  
print(wb['Sheet1']) #获取某个workbook工作表对象  
print(wb['Sheet1'].title) #获取工作表名,就是标题名

print(wb.active) # 获取当前活跃的workbook  
print(wb.read\_only) #排断是否以只读模式打开  
print(wb.encoding) #查看文件的编码格式  
fan = wb['Sheet1']  
print(fan["A1"].value) #获取工作表sheet1的A列第一行的数据  
print(fan["A1"].row) #行  
print(fan["A1"].column) #列  
print(fan["A1"].coordinate) #坐标

.remove：删除一个表格

.create\_sheet：创建一个空的表格

.copy\_worksheet：在Workbook内拷贝表格

.dimensions：表格的大小，这里的大小是指含有数据的表格的大小，即：左上角的坐标:右下角的坐标

.max\_row：表格的最大行

.min\_row：表格的最小行

.max\_column：表格的最大列

.min\_column：表格的最小列

.iter\_rows：按行获取所有单元格，内置属性有(min\_row,max\_row,min\_col,max\_col)

.iter\_columns：按列获取所有的单元格

.append：在表格末尾添加数据

.merged\_cells：合并多个单元格

.unmerged\_cells：移除合并的单元格

#修改工作表格名

wb = openpyxl.load\_workbook(r"E:\System\fanx\example.xlsx")  
wb.active.title = "spam"  
wb.save(r"E:\System\fanx\example\_copy.xlsx") #重新保存为新的

创建execl表格

wb = openpyxl.Workbook() #打开workbook

wb.create\_sheet('hmm') #插入工作表,默认插入到最后  
wb.create\_sheet('fangx',0) #插入到最开始的位置  
print(wb.sheetnames) #返回['fangx', 'Sheet', 'hmm'] Sheet是自动创建的,按照sheet，sheet1，sheet2自动增长  
wb["Sheet"].title = 'fzy' # 修改工作表名

wb.active['A1'] = 'Hello world!' #将值写入A1单元格

wb.active.append([1,2,3]) #也可以用列表方式写入  
print(wb.active['A1'].value) #查看单元格A1位置的值  
wb.save(r"E:\System\fanx\fangx.xlsx") #保存新的execl表格

wb = openpyxl.load\_workbook(r"E:\System\fanx\fangx.xlsx") #打开execl表格  
wb.remove(wb["Sheet"]) #删除Sheet工作表 或者这样写 del wb['fangx']  
wb.save(r"E:\System\fanx\fangx.xlsx")

设置单元格的字体

Font() ----------- 设置字体的风格属性

*from* openpyxl.styles *import* Font #导入这个模块

wb = openpyxl.Workbook()  
sheet = wb['Sheet']  
italic = Font(u'宋体',size=11,bold=*True*,italic=*True*,strike=*True*,color='000000')  
#设置字体为宋体,大小为11，bold为加粗,italic为斜体,strike为删除线,颜色为黑色  
sheet["A1"].font = italic #A1单元格  
sheet["A1"] = "Hello" #A1单元格的值为Hello  
wb.save(r"E:\System\fanx\italic.xlsx")

PatternFill() ----------- 填充图案和渐变色

可参考：https://www.jianshu.com/p/7af9a7c5b27d

Border() --------- 单元格的边框

Alignment() ---------- 单元格的对齐方式等

protection() --------- 写保护

execl的公式计算

wb = openpyxl.load\_workbook(r"E:\System\fanx\fanaaa.xlsx")

# wb["hmm"]["B20"] = '=SUM(B2:B19)' #直接在单元格上填写公式就可以计算

print(wb["hmm"]["B20"].value) #不加data\_only参数只能看到是公式  
wc = openpyxl.load\_workbook(r"E:\System\fanx\fanaaa.xlsx",data\_only=*True*)  
print(wc["hmm"]["B20"].value) #加上可以看到结果

wb = openpyxl.Workbook()

sheet = wb["Sheet"]

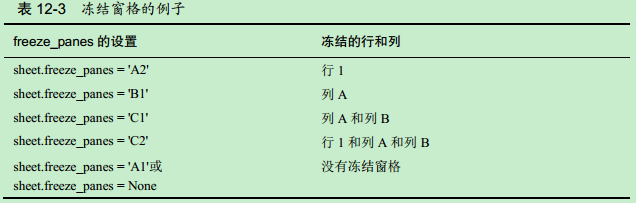
sheet.row\_dimensions[1].height = 70 #设置单元格的行高,1代表第一行

sheet.column\_dimensions['B'].width = 20 #设置单元格的列宽，B代表B列

sheet.merge\_cells('B2:C4') #合并单元格

sheet.unmerge\_cells('B2:C4') #拆分单元格

sheet.freeze\_panes ="A2" #冻结单元格



**制作表格**

openpyxl.chart.BarChart() -------- 创建条行图

openpyxl.chart.LineChart() --------- 建折线图

openpyxl.chart.ScatterChart() ---------- 散点图

openpyxl.chart.PieChart() -------- 饼图

# PyPDF2模块:处理PDF

# time 模块：时间模块

time.time --------------- Unix 纪元时间戳

time.sleep() --------- 让程序暂停,括号里面的数字是秒

# datetime模块：

datetime.datetime(year, month, day, hour, minute, second)函数 -- 返回参数指定的时刻的 datetime 对象。如果没有提供 hour、minute 或 second 参数，它们默认为 0。

datetime.datetime.now() ------ 当前计算机时间

dt = datetime.datetime.now () #获取当前计算机时间

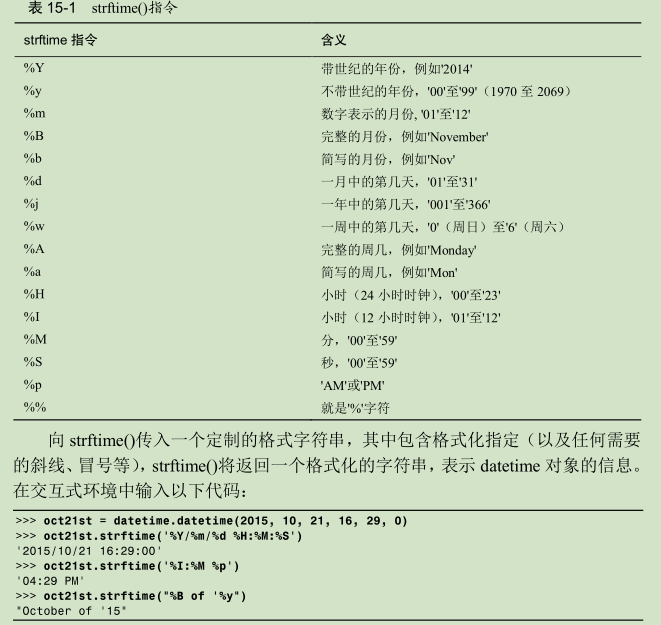
print(dt.year , dt.month , dt.day , dt.hour , dt.minute , dt.second) # 打印 年月日 时分秒

delta = datetime.timedelta(days=11, hours=10, minutes=9, seconds=8) #接受关键字参数 weeks、days、hours、minutes、print(delta) #seconds、milliseconds 和 microseconds  
#可以通过当前时间加上这个推算之后的天数

datetime.datetime.fromtimestamp(epoch)函数 -- 返回 epoch 时间戳参数表示的时刻的 datetime 对象。

strftime() -------- 将datetime 对象显示为字符串

total\_seconds()方法用于 timedelta 对象，返回 timedelta 对象表示的秒数



strptime() ----- datetime.datetime.strftime()与strftime()方法相反

# threading模块：多线程

*import* threading, time

print('Start of program.')  
*def* takeANap():  
 time.sleep(1)  
 print('Wake up!')  
threadObj = threading.Thread(target=takeANap) # 多线程调用  
threadObj.start()  
print('End of program.')  
time.sleep(10)  
print("done")

# subprocess 模块：

subprocess.Popen("D:\TIM\Bin\QQScLauncher.exe") #打开QQ

# smtplib模块：SMTP发送邮件的协议，负责发送邮件

smtplib.SMTP(smtp\_server) -------- SMTP的地址

.set\_debuglevel(1) -------------- 打印出SMTP服务器的交互地址

.login ------------------ 登录SMTP邮箱

.sendmail(from\_addr, to\_addr, msg.as\_string()) ------------------- 发邮件（发送人,收件人，as\_string()把MIMEText对象变成str）

# mail模块：负责构造邮件

# pyautogui 模块：自动化控制键盘和鼠标

pyautogui 模块可以向 Windows、OS X 和 Linux 发送虚拟按键和鼠标点击。根据你使用的操作系统，在安装 pyautogui 之前，可能需要安装一些其他模块（称为依赖关系）

1、在 Windows 上，安装PyGetWindow==0.0.1。

2、在 OS X 上，运行 sudo pip3 install pyobjc-framework-Quartz，sudo pip3 install pyobjc-core，然后 sudo pip3 install pyobjc。

3、在 Linux 上，运行 sudo pip3 install python3-xlib，sudo apt-get install scrot，sudo apt-get install python3-tk，以及 sudo apt-get install python3-dev（Scrot 是PyAutoGUI 使用的屏幕快照程序）。在这些依赖安装后，运行 pip install pyautogui安装 pyautogui。

pyautogui.PAUSE = 1 == 让程序在执行动作之后暂停的秒数

pyautogui.FAILSAFE = True == 自动防故障功能 =False 是禁止这项功能

print(pyautogui.size()) == 显示电脑显示屏的分辨率

pyautogui.moveTo(x,y) == 将鼠标移动到指定位置(坐标)

pyautogui.moveRel(0,0) == 在当前的位置移动鼠标

print(pyautogui.position()) == 获取鼠标的位置

for i in range(10): == 循环10次移动鼠标，duration 表示鼠标停留的秒数,

pyautogui.moveTo(100, 100, duration=0.25) #.moveRel()的参数和这个一样

pyautogui.moveTo(200, 100, duration=0.25)

pyautogui.moveTo(200, 200, duration=0.25)

pyautogui.moveTo(100, 200, duration=0.25)

pyautogui.click() == 使用鼠标点击，默认是左键点击

pyautogui.click(100,150,button='left') == 在坐标（100，150）处单击鼠标左键

pyautogui.click(1000,250,button = 'right') == 在坐标(200，250)处点击右键

pyautogui. mouseDown(x,y, button) == 按下鼠标按键，再调用

pyautogui.mouseUp(x,y, button) == 释放鼠标按键

pyautogui.doubleClick() == 双击鼠标左键

pyautogui.rightClick() == 点击右键

pyautogui.middleClick() == 点击中键

pyautogui.dragTo(x,y) == 按住鼠标左键移动鼠标

pyautogui.dragRel(x,y) == 按住鼠标,在当前的位置拖动

这2个方法的参数和moveTo() 和 moveRel() 相同：x 坐标/水平移动，y 坐标/垂直移动，以及可选的时间间隔

*import* pyautogui, time

time.sleep(5)  
pyautogui.click() # click to put drawing program in focus  
distance = 200  
*while* distance > 0:  
 pyautogui.dragRel(distance, 0, duration=0.2) # move right  
 distance = distance - 5  
 pyautogui.dragRel(0, distance, duration=0.2) # move down  
 pyautogui.dragRel(-distance, 0, duration=0.2) # move left  
 distance = distance - 5  
 pyautogui.dragRel(0, -distance, duration=0.2) # move up

pyautogui.scroll(2) == 滚动鼠标(传递正整数表示向上滚动，传递负整数表示向下滚动)

im=pyautogui.screenshot() == 获取屏幕快照的 Image 对象。然后使用

im.getpixel(x,y) 返回坐标位置的RGB元组，包含3 个整数，表示像素(红绿蓝值)

pyautogui.pixelMatchesColor(50, 200, (130, 135, 144)) == 指定x、y坐标的颜色,如果匹配返回True不匹配返回False

pyautogui.locateOnScreen('submit.png') == 图像识别,返回４个整数的元组，表示图像所在的坐标及宽度和高度,将这个元组传递给center()函数

pyautogui.center((643, 745, 70, 29)) == 将返回该区域中心的x、y坐标，就可以使用click()进行点击了

a = pyautogui.locateOnScreen("ypb.png") #识别到返回4个整数元组 识别不到返回None

e,f = pyautogui.center((a))  
pyautogui.click(e,f)

pyautogui.typewrite("hello word",0.25) == 在鼠标当前位置以0.25秒的速度输入,

print(pyautogui.KEYBOARD\_KEYS) == 查看所有能输入的键

pyautogui.keyDown() == 向计算发送虚拟的按键

pyautogui.keyUp() == 释放按键

pyautogui.press() == 调用上面两个函数，模拟完整的击键(按住并释放)

pyautogui.keyDown('shift'); pyautogui.press('4'); pyautogui.keyUp('shift') == 按住shift和4然后释放, 输出了$

pyautogui.keyDown('alt'); pyautogui.press('a'); pyautogui.keyUp('alt') == 执行按键alt+a

pyautogui.hotkey('alt', 'a') == 接受多个键字符串参数,按顺序按下,再按相反的顺序释放