安装环境

驱动安装

```
【1】定义
    phantomjs为无界面浏览器(又称无头浏览器)，在内存中进行页面加载,高效

【2】下载地址
    2.1) chromedriver : 下载对应版本
       http://npm.taobao.org/mirrors/chromedriver/

    2.2) geckodriver
       https://github.com/mozilla/geckodriver/releases

    2.3) phantomjs
       https://phantomjs.org/download.html

【3】Ubuntu安装
    3.1) 下载后解压 : tar -zxvf geckodriver.tar.gz

    3.2) 拷贝解压后文件到 /usr/bin/  (添加环境变量)
         sudo cp geckodriver /usr/bin/

    3.3) 添加可执行权限
         sudo chmod 777 /usr/bin/geckodriver

【4】Windows安装
    4.1) 下载对应版本的phantomjs、chromedriver、geckodriver
    4.2) 把chromedriver.exe拷贝到python安装目录的Scripts目录下(添加到系统环境变量)
         # 查看python安装路径: where python
    4.3) 验证
         cmd命令行: chromedriver

**************************总结***************************************
【1】解压 - 放到用户主目录(chromedriver、geckodriver、phantomjs)
【2】拷贝 - sudo cp /home/tarena/chromedriver /usr/bin/
【3】权限 - sudo chmod 777 /usr/bin/chromedriver

# 验证
【Ubuntu | Windows】
ipython3
from selenium import webdriver
webdriver.Chrome()
或者
webdriver.Firefox()

【mac】
ipython3

from selenium import webdriver
```

```
webdriver.Chrome(executable_path='/Users/xxx/chromedriver')
或者
webdriver.Firefox(executable_path='/User/xxx/geckodriver')
```

第三库selenium安装

```
安装: pip install selenium==XXXX
卸载 pip uninstall selenium
查看版本号: pip show selenium
```

示例代码

```
"""示例代码一: 使用 selenium+浏览器 打开百度"""

# 导入seleinum的webdriver接口
from selenium import webdriver
import time

# 创建浏览器对象
browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
# 5秒钟后关闭浏览器
time.sleep(5)
browser.quit()
```

```
"""示例代码二: 打开百度, 搜索赵丽颖, 点击搜索, 查看"""

from selenium import webdriver
import time

# 1.创建浏览器对象 - 已经打开了浏览器
browser = webdriver.Chrome()
# 2.输入: http://www.baidu.com/
browser.get('http://www.baidu.com/')
# 3.找到搜索框,向这个节点发送文字: 赵丽颖
browser.find_element_by_xpath('//*[@id="kw"]').send_keys('赵丽颖')
# 4.找到 百度一下 按钮,点击一下
browser.find_element_by_xpath('//*[@id="su"]').click()
```

```
案例:
□启动火狐浏览器,
□首先打开我要自学网页面, 打印网页标题, 等待3秒
□打开百度首页, 打印网页标题, 再等待2秒
□关闭浏览器。
from selenium import webdriver
from time import sleep

#加载浏览器驱动

driver=webdriver.Firefox()
```

```
11
12  #打开自学网页面
13  driver.get("http://www.51zxw.net")
14  print(driver.title)
15  sleep(3)
16
17  #打开百度首页
18  driver.get("http://www.baidu.com")
19  print(driver.title)
20  sleep(3)
21
22  #关闭浏览器
23  driver.quit()
```

浏览器对象方法

```
1   【1】browser.get(url=url)    - 地址栏输入url地址并确认
2   【2】browser.quit()          - 关闭浏览器
3   【3】browser.close()         - 关闭当前页
4   【4】browser.page_source     - HTML结构源码
5   【5】browser.page_source.find('字符串')
6       从html源码中搜索指定字符串,没有找到返回: -1,经常用于判断是否为最后一页
7   【6】browser.maximize_window() - 浏览器窗口最大化
8   【7】browser执行JS脚本
9       browser.execute_script('window.scrollTo(0,document.body.scrollHeight)')
10
11  from selenium import webdriver
12  driver = webdriver.Firefox()
13  driver.get('https://www.baidu.com')
14  driver.get('https://news.baidu.com')
15  # 后退
16  driver.back()
17  # 前进
18  driver.forward()
19  driver.quit()
```

```
1   浏览器操作
2   □浏览器窗口大小设置
3   □页面前进后退
4   □页面刷新
5   from  selenium import webdriver
6   from time import  sleep
7
8   driver=webdriver.Firefox()
9
10  driver.get("http://www.51zxw.net")
11  driver.maximize_window()
12  sleep(2)

13
```

```
driver.get("http://www.51zxw.net/list.aspx?cid=615")
driver.set_window_size(400,800)
driver.refresh()
sleep(2)

driver.back()
sleep(2)

driver.forward()

sleep(2)
driver.quit()
```

定位节点八种方法

```
【1】单元素查找('结果为1个节点对象')
    1.1) 【最常用】browser.find_element_by_id('id属性值')
    1.2) 【最常用】browser.find_element_by_name('name属性值')
    1.3) 【最常用】browser.find_element_by_class_name('class属性值')
    1.4) 【最万能】browser.find_element_by_xpath('xpath表达式')
    1.5) 【匹配a节点时常用】browser.find_element_by_link_text('链接文本')
    1.6) 【匹配a节点时常用】browser.find_element_by_partical_link_text('部分链接文本')
    1.7) 【最没用】browser.find_element_by_tag_name('标记名称')
    1.8) 【较常用】browser.find_element_by_css_selector('css表达式')

【2】多元素查找('结果为[节点对象列表]')
    2.1) browser.find_elements_by_id('id属性值')
    2.2) browser.find_elements_by_name('name属性值')
    2.3) browser.find_elements_by_class_name('class属性值')
    2.4) browser.find_elements_by_xpath('xpath表达式')
    2.5) browser.find_elements_by_link_text('链接文本')
    2.6) browser.find_elements_by_partical_link_text('部分链接文本')
    2.7) browser.find_elements_by_tag_name('标记名称')
    2.8) browser.find_elements_by_css_selector('css表达式')

元素定位不到:
元素没有加载完成
iframe
元素不可用，不可读，不可见
动态属性
```

id与name 定位

```python
from  selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.baidu.com")

driver.find_element_by_id("kw").send_keys("Selenium我要自学网")
driver.find_element_by_name("wd").send_keys("Selenium我要自学网")

sleep(2)
driver.find_element_by_id("su").click()
driver.close()
```

tag_name定位

```python
# 案例：打开我要自学网页面，在用户名输入框输入用户名“selenium”
from  selenium import webdriver
from  time import sleep

driver=webdriver.Firefox()

driver.get("http://www.51zxw.com")

# 定位标签名为input的元素
driver.find_element_by_tag_name("input").send_keys("selenium")

# 获取页面所有标签名称为“input”的标签。
driver.find_elements_by_tag_name("input")[0].send_keys("selenium")

sleep(3)

driver.quit()
```

class_name定位

```python
# 根据标签中属性class来进行定位的一种方法
from  selenium import webdriver
from  time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")

driver.find_element_by_class_name("s_ipt").send_keys("Selenium 我要自学网")
sleep(2)

driver.find_element_by_id("su").click()
sleep(3)

driver.quit()
```

link_text与partial_link_text定位

```
# link_text定位就是根据超链接文字进行定位。
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.51zxw.net/")

driver.find_element_by_link_text('程序设计').click()
sleep(3)
driver.find_element_by_partial_link_text('神秘面纱').click()
sleep(3)
driver.quit()
```

XPath定位

xpath解析

- **定义**

```
XPath即为XML路径语言，它是一种用来确定XML文档中某部分位置的语言，同样适用于HTML文档的检索
```

- **匹配演示 - 猫眼电影top100**

```
【1】查找所有的dd节点
    //dd
【2】获取所有电影的名称的a节点：所有class属性值为name的a节点
    //p[@class="name"]/a
【3】获取dl节点下第2个dd节点的电影节点
    //dl[@class="board-wrapper"]/dd[2]
【4】获取所有电影详情页链接：获取每个电影的a节点的href的属性值
    //p[@class="name"]/a/@href

【注意】
    1> 只要涉及到条件，加 [] ：//dl[@class="xxx"]   //dl/dd[2]
    2> 只要获取属性值，加 @  ：//dl[@class="xxx"]   //p/a/@href
```

- **选取节点**

```
【1】// ：从所有节点中查找（包括子节点和后代节点）
【2】@  ：获取属性值
  2.1> 使用场景1（属性值作为条件）
       //div[@class="movie-item-info"]
  2.2> 使用场景2（直接获取属性值）
       //div[@class="movie-item-info"]/a/img/@src

【3】练习 - 猫眼电影top100

  3.1> 匹配电影名称
```

```
        //div[@class="movie-item-info"]/p[1]/a/@title
    3.2> 匹配电影主演
        //div[@class="movie-item-info"]/p[2]/text()
    3.3> 匹配上映时间
        //div[@class="movie-item-info"]/p[3]/text()
    3.4> 匹配电影链接
        //div[@class="movie-item-info"]/p[1]/a/@href
```

- **匹配多路径（或）**

```
xpath表达式1 | xpath表达式2 | xpath表达式3
```

- **常用函数**

```
【1】text() ： 获取节点的文本内容
    xpath表达式末尾不加 /text() :则得到的结果为节点对象
    xpath表达式末尾加 /text() 或者 /@href ： 则得到结果为字符串

【2】contains() ： 匹配属性值中包含某些字符串节点
    匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
     //div[contains(@class,"movie-item")]
```

- **终极总结**

```
【1】字符串：xpath表达式的末尾为： /text() 、/@href  得到的列表中为'字符串'

【2】节点对象：其他剩余所有情况得到的列表中均为'节点对象'
    [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
    [<element div at xxxa>,<element div at xxxb>]
    [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]
```

- **课堂练习**

```
【1】匹配汽车之家-二手车,所有汽车的链接 ：
    //li[@class="cards-li list-photo-li"]/a[1]/@href
    //a[@class="carinfo"]/@href
【2】匹配汽车之家-汽车详情页中,汽车的
     2.1)名称：  //div[@class="car-box"]/h3/text()
     2.2)里程：  //ul/li[1]/h4/text()
     2.3)时间：  //ul/li[2]/h4/text()
     2.4)挡位+排量：//ul/li[3]/h4/text()
     2.5)所在地：//ul/li[4]/h4/text()
     2.6)价格：   //div[@class="brand-price-item"]/span[@class="price"]/text()
```

```
xpath绝对与相对定位
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

```

```
driver.get("http://www.baidu.com")

# 绝对路径定位
driver.find_element_by_xpath("/html/body/div[2]/div[1]/div/div[1]/div/form/span[1]/input").send_keys("51zxw")

# 利用元素熟悉定位--定位到input标签中为kw的元素
driver.find_element_by_xpath("//input[@id='kw']").send_keys("Selenium")

# 定位input标签中name属性为wd的元素
driver.find_element_by_xpath("//input[@name='wd']").send_keys("Selenium")

# 定位所有标签元素中，class属性为s_ipt的元素
driver.find_element_by_xpath("//*[@class='s_ipt']").send_keys("Python3")

driver.find_element_by_id('su').click()

sleep(3)
driver.quit()


Xpath层级与逻辑定位
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.51zxw.net/")
#层级和属性结合定位--自学网首页输入用户和名密码
driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[1]").send_keys("51zxw")
driver.find_element_by_xpath("//form[@id='loginForm']/ul/input[2]").send_keys("66666")

#逻辑运算组合定位
driver.find_element_by_xpath("//input[@class='loinp' and @name='username']").send_keys("51zxw")

sleep(3)
driver.quit()
```

css定位

冻结窗口

setTimeout(function(){debugger}, 5000)

Selenium极力推荐使用CSS 定位，而不是XPath来定位元素，原因是CSS 定位比XPath 定速度快，语法也更加简洁。

CSS常用定位方法

1. find_element_by_css_selector（）
2. #id id选择器根据id属性来定位元素
3. .class class选择器，根据class属性值来定位元素
4. [attribute='value'] 根据属性来定位元素
5. element>element 根据元素层级来定位 父元素>子元素

```
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")

#根据id来定位
driver.find_element_by_css_selector('#kw').send_keys("Selenium 我要自学网")

#根据class定位
driver.find_element_by_css_selector('.s_ipt').send_keys('python')

#通过属性来定位
driver.find_element_by_css_selector("[autocomplete='off']").send_keys("selenium")

sleep(2)
driver.find_element_by_id('su').click()

driver.get("http://www.51zxw.net")

#通过元素层级来定位
driver.find_element_by_css_selector("form#loginForm>ul>input").send_keys("51zxw")

sleep(2)
driver.quit()
```

节点对象操作

```
【1】文本框操作
    1.1) node.send_keys('')   - 向文本框发送内容
    1.2) node.clear()         - 清空文本
    1.3) node.get_attribute('value') - 获取文本内容

【2】按钮操作
    1.1) node.click()       - 点击
    1.2) node.is_enabled()  - 判断按钮是否可用
    1.3) node.get_attribute('value') - 获取按钮文本
```

设置无界面模式

```python
from selenium import webdriver

options = webdriver.ChromeOptions()
# 添加无界面参数
options.add_argument('--headless')
browser = webdriver.Chrome(options=options)

```

鼠标操作

```
鼠标操作：现在页面中随处可以看到需要右击、双击、鼠标悬停、甚至是鼠标拖动等操作的功能设计。在webdriver中这些关于鼠标操作的方法由ActionChains类提供。
ActionChains类提供的鼠标操作的常用方法：
perform()执行所有ActionChains中存储的行为
context_click() 右击
double_click() 双击
drag_and_drop() 拖动
move_to_element() 鼠标悬停

from selenium import webdriver
# 导入鼠标事件类
from selenium.webdriver import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.baidu.com/')

# 移动到 设置，perform()是真正执行操作，必须有
element = driver.find_element_by_xpath('//*[@id="u1"]/a[8]')
ActionChains(driver).move_to_element(element).perform()

# 单击，弹出的Ajax元素，根据链接节点的文本内容查找
driver.find_element_by_link_text('高级搜索').click()
```

```python
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")
driver.maximize_window()

driver.find_element_by_css_selector("#kw").send_keys("Python")

# 获取搜索框元素对象
element=driver.find_element_by_css_selector("#kw")

sleep(3)
#双击操作
```

```
ActionChains(driver).double_click(element).perform()

sleep(2)

#右击操作
ActionChains(driver).context_click(element).perform()

sleep(3)

#鼠标悬停
above=driver.find_element_by_css_selector(".pf")
ActionChains(driver).move_to_element(above).perform()

sleep(4)
driver.quit()
```

键盘操作

```
node.send_keys(Keys.SPACE)
node.send_keys(Keys.CONTROL, 'a')
node.send_keys(Keys.CONTROL, 'c')
node.send_keys(Keys.CONTROL, 'v')
node.send_keys(Keys.ENTER)

# 案例： 在百度搜索关键词“Python” 然后将关键词复制或剪切到搜狗搜索框进行搜索
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")
driver.find_element_by_css_selector("#kw").send_keys("Python")

sleep(2)
# 键盘全选操作 Ctrl+A
driver.find_element_by_css_selector("#kw").send_keys(Keys.CONTROL,'a')

# 键盘选择复制或剪切操作 Ctrl+C
driver.find_element_by_css_selector("#kw").send_keys(Keys.CONTROL,'c')
driver.find_element_by_css_selector("#kw").send_keys(Keys.CONTROL,'x')

# 打开搜狗页面
driver.get("http://www.sogou.com/")
sleep(2)

# 粘贴复制内容
driver.find_element_by_css_selector(".sec-input").send_keys(Keys.CONTROL,'v')
sleep(2)

# 点击搜索按钮
# driver.find_element_by_xpath("//input[@id='stb']").click()
```

```
driver.find_element_by_css_selector("#stb").click()

sleep(3)
driver.quit()
```

切换页面

- **适用网站+应对方案**

```
【1】适用网站类型
    页面中点开链接出现新的窗口，但是浏览器对象browser还是之前页面的对象，需要切换到不同的窗口进行操作

【2】应对方案 - browser.switch_to.window()

    # 获取当前所有句柄（窗口）- [handle1,handle2]
    all_handles = browser.window_handles
    # 切换browser到新的窗口，获取新窗口的对象
    browser.switch_to.window(all_handles[1])
```

iframe

- **特点+方法**

```
【1】特点
    网页中嵌套了网页，先切换到iframe，然后再执行其他操作

【2】处理步骤
    2.1) 切换到要处理的Frame
    2.2) 在Frame中定位页面元素并进行操作
    2.3) 返回当前处理的Frame的上一级页面或主页面

【3】常用方法
    3.1) 切换到frame  -  browser.switch_to.frame(frame节点对象)
    3.2) 返回上一级   -  browser.switch_to.parent_frame()
    3.3) 返回主页面   -  browser.switch_to.default_content()

【4】使用说明
    4.1) 方法一：默认支持id和name属性值 ： switch_to.frame(id属性值|name属性值)
    4.2) 方法二：
        a> 先找到frame节点 ： frame_node = browser.find_element_by_xpath('xxxx')
        b> 在切换到frame   ： browser.switch_to.frame(frame_node)

【5】iframe子框架
    browser.switch_to.frame(iframe_element)
    # 写法1 - 任何场景都可以：
    iframe_node = browser.find_element_by_xpath('')
    browser.switch_to.frame(iframe_node)

```

```
    # 写法2 - 默认支持 id 和 name 两个属性值:
    browser.switch_to.frame('id属性值|name属性值')
```

- **示例1 - 登录豆瓣网**

```
"""
登录豆瓣网
"""
from selenium import webdriver
import time

# 打开豆瓣官网
browser = webdriver.Chrome()
browser.get('https://www.douban.com/')

# 切换到iframe子页面
login_frame = browser.find_element_by_xpath('//*[@id="anony-reg-new"]/div/div[1]/iframe')
browser.switch_to.frame(login_frame)

# 密码登录 + 用户名 + 密码 + 登录豆瓣
browser.find_element_by_xpath('/html/body/div[1]/div[1]/ul[1]/li[2]').click()
browser.find_element_by_xpath('//*[@id="username"]').send_keys('自己的用户名')
browser.find_element_by_xpath('//*[@id="password"]').send_keys('自己的密码')
browser.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div[5]/a').click()
time.sleep(3)

# 点击我的豆瓣
browser.find_element_by_xpath('//*[@id="db-nav-sns"]/div/div/div[3]/ul/li[2]/a').click()
```

元素等待

```
元素等待

概念

· 显示等待是针对某一个元素进行相关等待判定;

· 隐式等待不针对某一个元素进行等待，全局元素等待。
a.相关模块
□WebDriverWait 显示等待针对元素必用
□expected_conditions 预期条件类（里面包含方法可以调用，用于显示等待）
□NoSuchElementException 用于隐式等待抛出异常
□By 用于元素定位
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait          #注意字母大写
```

```
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException
显示等待
案例：检测百度页面搜索按钮是否存在，存在就输入关键词"自学网 Selenium" 然后点击搜索
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

from time import sleep

driver=webdriver.Firefox()

driver.get("http://www.baidu.com")


driver.find_element_by_css_selector("#kw").send_keys("自学网 Selenium")

sleep(2)

#显示等待--判断搜索按钮是否存在
element=WebDriverWait(driver,5,0.5).until(EC.presence_of_element_located((By.ID,"su")
))
element.click()
sleep(3)

driver.quit()
隐式等待
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from time import sleep,ctime

driver=webdriver.Firefox()
driver.get("http://www.baidu.com")

sleep(2)

driver.implicitly_wait(5) #隐式等待时间设定 5秒

#检测搜索框是否存在
try:
    print(ctime())
    driver.find_element_by_css_selector("#kw").send_keys("Python")
    driver. find_element_by_css_selector("#su").click
except NoSuchElementException as msg:
    print(msg)
finally:
    print(ctime())

sleep(3)
driver.quit()

```

滚动条控制操作

```
# 案例：打开我要自学网页面，然后将滚动条拖到最底部，然后再拖到顶部
from  selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.51zxw.net/")
sleep(2)

#将滚动调拖到最底部
js="var action=document.documentElement.scrollTop=10000"
driver.execute_script(js)
sleep(2)

#将滚动条拖到最顶部
js="var action=document.documentElement.scrollTop=0"
driver.execute_script(js)
sleep(3)

driver.quit()
```

网页截图操作

```
# 案例：分别打开我要自学网页面和百度页面，然后进行截图
from selenium import webdriver
from time import sleep

#加载浏览器驱动
driver=webdriver.Firefox()

#打开自学网页面并截图
driver.get("http://www.51zxw.net")
driver.get_screenshot_as_file(r"E:\Python_script\51zxw.jpg")

#打开百度页面并截图
driver.get("http://www.baidu.com")
driver.get_screenshot_as_file(r"E:\Python_script\baidu.png")


sleep(2)
driver.quit()
```

上传文件

```
案例：在百度搜索上传本地图片进行搜索。
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.baidu.com")

driver.find_element_by_css_selector(".soutu-btn").click()
sleep(3)
driver.find_element_by_css_selector(".upload-pic").send_keys(r"E:\Python_script\Webdriver\shuiyin.png")

sleep(3)
driver.quit()
```

Cookie处理

- 什么是Cookie

Cookie是储存在用户本地终端上的数据，实际上是一小段的文本信息。

- Cookie作用

帮助 Web 站点保存有关访问者的信息，方便用户的访问。如记住用户名密码实现自动登录。

```
# 案例：查看访问我要自学网时的Cookie内容

from selenium import webdriver

driver=webdriver.Firefox()
driver.get("http://www.51zxw.net/")

#获取cookie全部内容
cookie=driver.get_cookies()
#打印全部cookile信息
print(cookie)
#打印cookie第一组信息
print(cookie[0])

#添加cookie
driver.add_cookie({'name':'51zxw','value':'www.51zxw.net'})
for cookie in driver.get_cookies():
    print("%s --- %s" %(cookie['name'],cookie['value']))

driver.quit()
```

自动化测试验证码问题

验证码作用

不少网站在用户登录、用户提交信息等登录和输入的页面上使用了验证码技术。验证码技术可以有效防止恶意用户对网站的滥用，使得网站可以有效避免用户信息失窃、保证网站稳定安全性。

但是验证码给自动化测试带来一些不便，使脚本无法正常运行覆盖功能模块。

如何解决

1.去掉验证码

这是最简单的方法，对于开发人员来说，只是把验证码的相关代码注释掉即可，如果是在测试环境，这样做可省去了测试人员不少麻烦，如果自动化脚本是要在正式环境跑，这样就给系统带来了一定的风险。

2.设置万能码

去掉验证码的主要是安全问题，为了应对在线系统的安全性威胁，可以在修改程序时不取消验证码，而是程序中留一个“后门”---设置一个“万能验证码”，只要用户输入这个“万能验证码”，程序就认为验证通过，否则按照原先的验证方式进行验证。

3.验证码识别技术（OCR）

例如可以通过Python-tesseract 来识别图片验证码，Python-tesseract是光学字符识别Tesseract OCR引擎的Python封装类。能够读取任何常规的图片文件(JPG, GIF ,PNG , TIFF等)。不过，目前市面上的验证码形式繁多，目前任何一种验证码识别技术，识别率都不是100% 。

4.记录cookie

通过向浏览器中添加cookie 可以绕过登录的验证码。

基于Cookie绕过验证码自动登录

```
# 案例：使用Cookie绕过百度验证码自动登录账户。
from selenium import webdriver
from time import sleep

driver=webdriver.Firefox()
driver.get("http://www.baidu.com/")

#手动添加cookie
driver.add_cookie({'name':'BAIDUID','value':'9E4BF1D44014…(根据实际获取值填写)'})
driver.add_cookie({'name':'BDUSS','value':'根据实际抓包获取值填写'})

sleep(2)
driver.refresh()
sleep(3)
driver.quit()
```

自动化测试模型

概念

自动化测试模型可以看作自动化测试框架与工具设计的思想。自动化测试不仅仅是单纯写写脚本运行就可以了，还需要考虑到如何使脚本运行效率提高，代码复用、参数化等问题。自动化测试模型分为四大类：线性模型，模块化驱动测试、数据驱动、关键词驱动。

```
线性模型
线性脚本中每个脚本都相互独立，且不会产生其他依赖与调用，其实就是简单模拟用户某个操作流程的脚本。

模块化驱动测试
线性模型虽然每个用例都可以拿出来独立运行，但是用例之间重复代码很多，开发、维护成本高。其实把重复的操作代码封装为独立的公共模块，当用例执行时需要用到这部分，直接调用即可，这就是模块驱动的方式。比如登录系统、退出登录、截图函数等等。

数据驱动测试
模块驱动的模型虽然解决了脚本的重复问题，但是需要测试不同数据的用例时，模块驱动的方式就不很适合了。 数据驱动就是数据的改变从而驱动自动化测试的执行，最终引起测试结果的改变。 装载数据的方式可以是列表、字典或是外部文件（txt、csv、xml、excel），目的就是实现数据和脚本的分离。

关键字驱动测试
通过关键字的改变引起测试结果的改变叫关键字驱动测试。 selenium IDE也是一种传统的关键字驱动的自动化工具，Robot Framework 是一个功能更强大的关键字驱动测试框架
```

Fiddler抓包工具

- **配置Fiddler**

```
【1】Tools -> Options -> HTTPS
    1.1) 添加证书信任：  勾选 Decrypt Https Traffic 后弹出窗口，一路确认
    1.2) 设置之抓浏览器的包：  ...from browsers only

【2】Tools -> Options -> Connections
    2.1) 设置监听端口（默认为8888）

【3】配置完成后重启Fiddler ('重要')
    3.1) 关闭Fiddler,再打开Fiddler
```

- **配置浏览器代理**

```
【1】安装Proxy SwitchyOmega谷歌浏览器插件

【2】配置代理
    2.1) 点击浏览器右上角插件SwitchyOmega -> 选项 -> 新建情景模式 -> myproxy(名字) -> 创建
    2.2) 输入  HTTP://  127.0.0.1  8888
    2.3) 点击 : 应用选项

【3】点击右上角SwitchyOmega可切换代理

【注意】: 一旦切换了自己创建的代理,则必须要打开Fiddler才可以上网
```

- **Fiddler常用菜单**

```
【1】Inspector : 查看数据包详细内容
    1.1) 整体分为请求和响应两部分

【2】Inspector常用菜单
    2.1) Headers : 请求头信息
    2.2) WebForms: POST请求Form表单数据 : <body>
                   GET请求查询参数: <QueryString>
    2.3) Raw : 将整个请求显示为纯文本
```

selenium

```
【1】定义
    1.1) 开源的Web自动化测试工具

【2】用途
    2.1) 对Web系统进行功能性测试,版本迭代时避免重复劳动
    2.2) 兼容性测试(测试web程序在不同操作系统和不同浏览器中是否运行正常)
    2.3) 对web系统进行大数量测试

【3】特点
    3.1) 可根据指令操控浏览器
    3.2) 只是工具，必须与第三方浏览器结合使用

【4】安装
    4.1) Linux: sudo pip3 install selenium
    4.2) Windows: python -m pip install selenium
```

- **PhantomJS浏览器**

```
【1】定义
    phantomjs为无界面浏览器(又称无头浏览器)，在内存中进行页面加载,高效

【2】下载地址
    2.1) chromedriver : 下载对应版本
        http://npm.taobao.org/mirrors/chromedriver/

```

```
    2.2) geckodriver
       https://github.com/mozilla/geckodriver/releases

    2.3) phantomjs
       https://phantomjs.org/download.html

【3】Ubuntu安装
    3.1) 下载后解压 : tar -zxvf geckodriver.tar.gz

    3.2) 拷贝解压后文件到 /usr/bin/  (添加环境变量)
         sudo cp geckodriver /usr/bin/

    3.3) 添加可执行权限
         sudo chmod 777 /usr/bin/geckodriver

【4】Windows安装
    4.1) 下载对应版本的phantomjs、chromedriver、geckodriver
    4.2) 把chromedriver.exe拷贝到python安装目录的Scripts目录下(添加到系统环境变量)
         # 查看python安装路径: where python
    4.3) 验证
         cmd命令行: chromedriver

**************************总结***************************************
【1】解压 - 放到用户主目录(chromedriver、geckodriver、phantomjs)
【2】拷贝 - sudo cp /home/tarena/chromedriver /usr/bin/
【3】权限 - sudo chmod 777 /usr/bin/chromedriver

# 验证
【Ubuntu | Windows】
ipython3
from selenium import webdriver
webdriver.Chrome()
或者
webdriver.Firefox()

【mac】
ipython3
from selenium import webdriver
webdriver.Chrome(executable_path='/Users/xxx/chromedriver')
或者
webdriver.Firefox(executable_path='/User/xxx/geckodriver')
```

```
1
```

sudo apt-get update 1.开机先卸载python2.7 sudo apt-get remove python2.7 2.卸载python2.7及其依赖 sudo apt-get remove --auto-remove python2.7 3.消除python2.7 sudo apt-get purge python2.7 or sudo apt-get purge --auto-remove python2.7 4.国内源 -i https://pypi.tuna.tsinghua.edu.cn/simple

4.安装系统环境 python3.6

mysql sudo apt install mysql-server 如果安装过程中没有提示设置账户密码，那么 查看账号密码：sudo cat /etc/mysql/debian.cnf 然后用此账号登陆mysql 换到mysql数据库，查看user表中root用户的权限及密码：mysql> use mysql mysql> select Host,user,authentication_string,plugin from user; 查看用户的权限，是否是mysql_native_password，如果不是，则将auth_sock改为mysql_native_password。 mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '你的密码'; 允许MYSQL远程连接： sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf 将bind-address = 127.0.0.1注释掉 设置某个用户可以远程访问 update user set host='%' where user='root' and host='localhost'; flush privileges; 退出数据库 重启数据库 sudo service mysql restart

nginx sudo apt install nginx mysqlclient sudo apt-get install python3-dev default-libmysqlclient-dev 查看是否已安装 apt list | grep 'default-libmysqlclient-dev' 进行安装 sudo pip3 install mysqlclient 再次查看是否安装完成 pip3 list | grep 'mysqlclient' redis sudo apt-get install redis-server redis-sentinel sudo apt install redis-sentinel mongodb sudo apt install mongodb tesseract-ocr sudo apt-get install tesseract-ocr # 图像识别库，提取文字 nmap sudo apt install nmap # 扫描局域网内所有IP nmap -sP 176.198.105.0/24

5.安装pip3 pip3 sudo apt install python3-pip

6.安装python库 mycli pip3 install mycli #MySQL的命令行工具 sudo apt install mycli redis sudo pip3 install redis # redis django 2.2.12 sudo pip3 install django==2.2.12 uwsgi 2.0.18 sudo pip3 install uwsgi==2.0.18 django-cors-headers sudo pip3 install django-cors-headers # 跨域 Celery sudo pip3 install -U Celery # 异步 django-redis sudo pip3 install django-redis jieba sudo pip3 install jieba # 中文分词器 django-crontab sudo pip3 install django-crontab # 定时器 python-alipay-sdk sudo pip3 install python-alipay-sdk # 阿里支付 pymongo sudo pip3 install pymongo # 爬虫存数据用的 fake_useragent sudo pip3 install fake_useragent # 爬虫生成useragent的插件 lxml sudo pip3 install lxml # 爬虫xpath解析模块 selenium sudo pip3 install selenium # web测试工具 服务器没装 scrapy_redis sudo pip3 install scrapy_redis # 分布式爬虫 pytesseract sudo pip3 install pytesseract # 图像识别成文字 pycryptodome sudo pip3 install pycryptodome # 加解密模块

数据分析库

numpy sudo pip3 install numpy # 基础数值算法 scipy sudo pip3 install scipy # 科学计算 matplotlib sudo pip3 install matplotlib # 数据可视化 pandas sudo pip3 install pandas # 序列高级函数

7.爬虫框架 Scrapy sudo pip3 install Scrapy 如果依赖缺失： 1.1) 安装依赖包 a> sudo apt-get install libffi-dev b> sudo apt-get install libssl-dev c> sudo apt-get install libxml2-dev d> sudo apt-get install python3-dev e> sudo apt-get install libxslt1-dev f> sudo apt-get install zlib1g-dev g> sudo pip3 install -I -U service_identity

```
1          1.2) 安装scrapy框架
2              a> sudo pip3 install Scrapy
```

8.机器学习，数据分析库 （服务器没装） jupyter sodu pip3 install jupyter # 网页在线编程

```
1
2
3
```