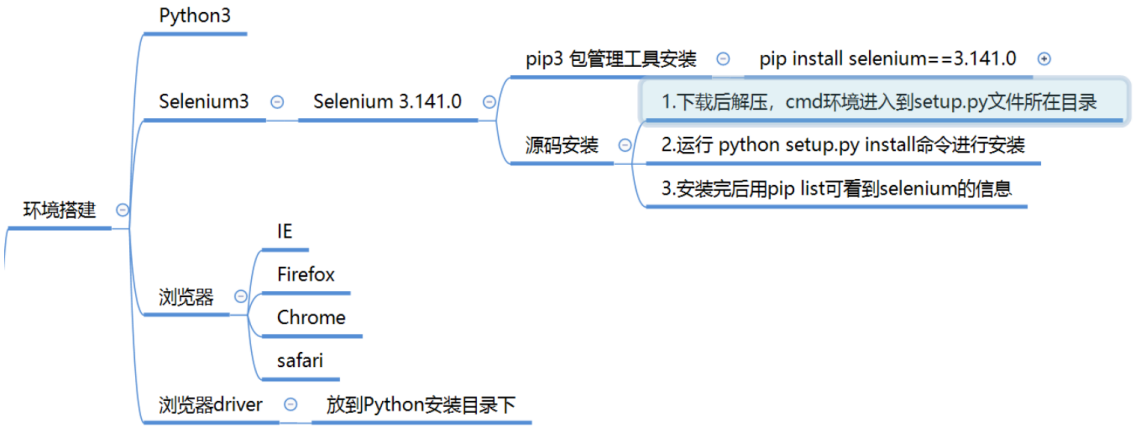
# Selenium-API操作

## 1 环境搭建




### 1.1 基于Python环境搭建

- Python 开发环境
- 安装Selenium包
- 安装浏览器
- 安装对应版本浏览器驱动

### 1.2 安装Selenium包

前置条件：Python环境已成功安装并运行

#### 1.2.1 pip工具

Python包管理工具，对Python提供包的查找、下载、安装、 卸载等功能

- 安装

```
pip install selenium  默认命令安装selenium最新版本
```

指定版本安装

```
pip install selenium==3.141.0
```

- 卸载

```
pip uninstall selenium
```

- 查询

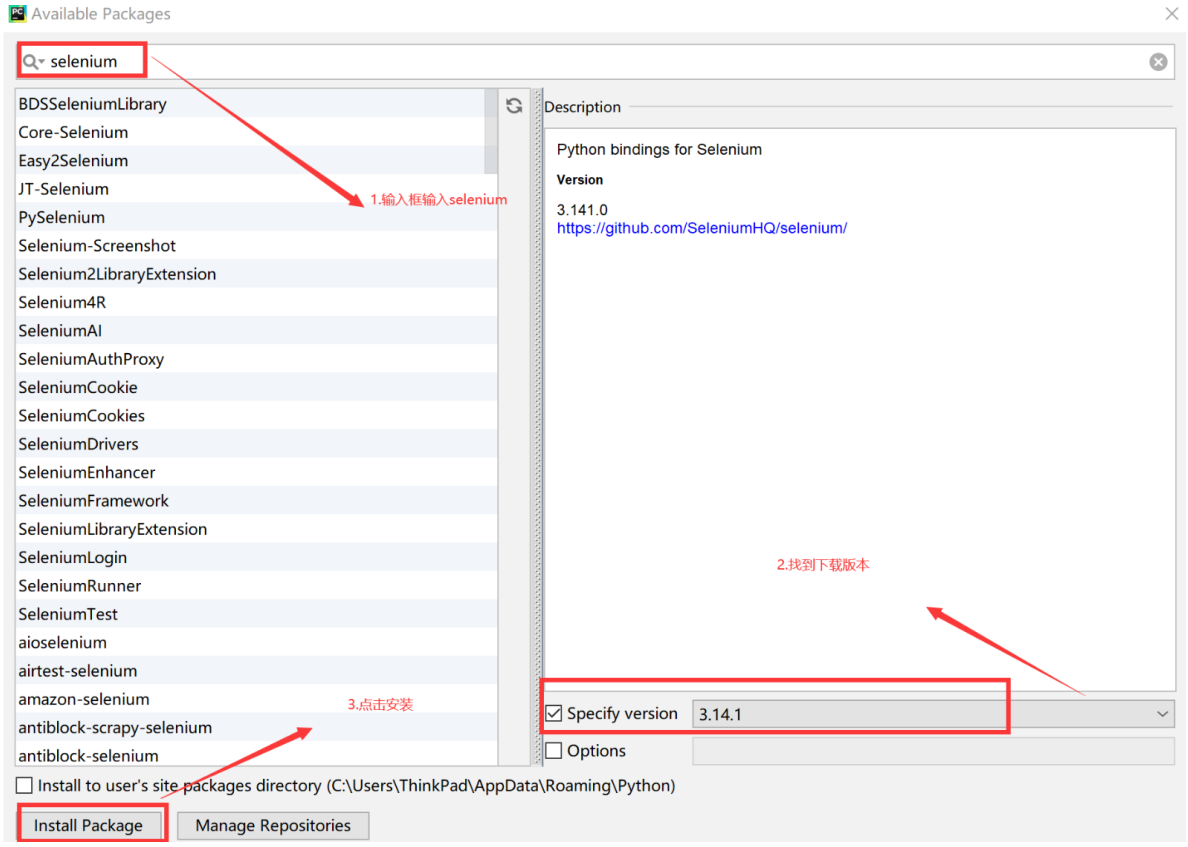
```
pip list
```

### 1.2.2 源码安装

若pip不能安装selenium，可通过源码包安装

- 下载后解压，cmd环境进入到setup.py文件所在目录
- 运行 python setup.py install命令进行安装
- 安装完后用pip show selenium 可看到selenium的信息

### 1.2.3 Pycharm安装




## 1.3 安装浏览器驱动

### 1.3.1 谷歌浏览器

- Selenium3.x + Chrome驱动（chromedriver）
- 驱动下载地址：http://npm.taobao.org/mirrors/chromedriver/
- 地址栏输入：chrome://version/ 查询chrome版本


**注意* 浏览器版本和驱动版本需要匹配**

**配置Chromedriver的环境变量**

windows：直接对应的driver放在python根目录下

**Mac**

vim ~/.bash_profile 修改环境变量的配置

export PATH=$PATH:[chromedriver所在路径]

### 1.3.2 Firefox浏览器

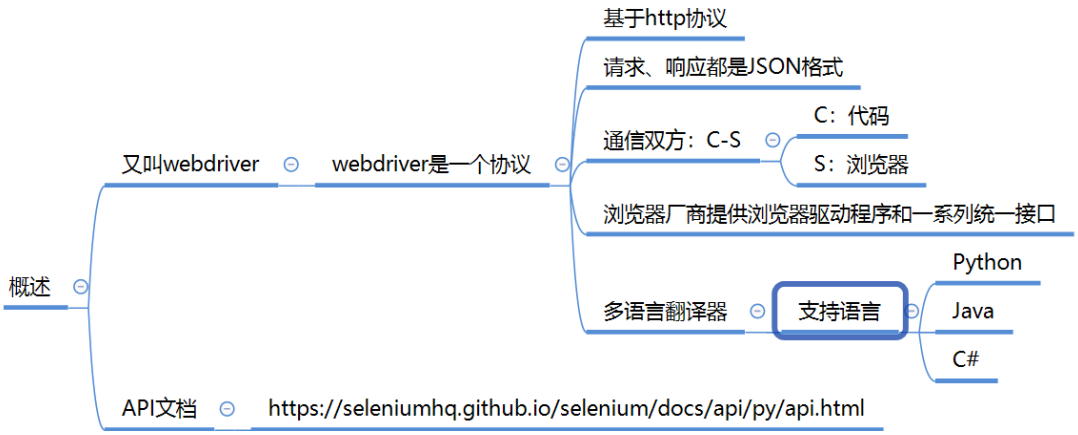
- Firefox 48 以上版本
- selenium 3.x + Firefox驱动(geckodriver)
- 驱动版本匹配地址https://firefox-source-docs.mozilla.org/testing/geckodriver/Support.html
- 驱动下载地址：https://github.com/mozilla/geckodriver/releases

# 2 Selenium介绍

## 2.1 概念

Selenium是一个Web应用程序测试工具 ,化学元素 硒

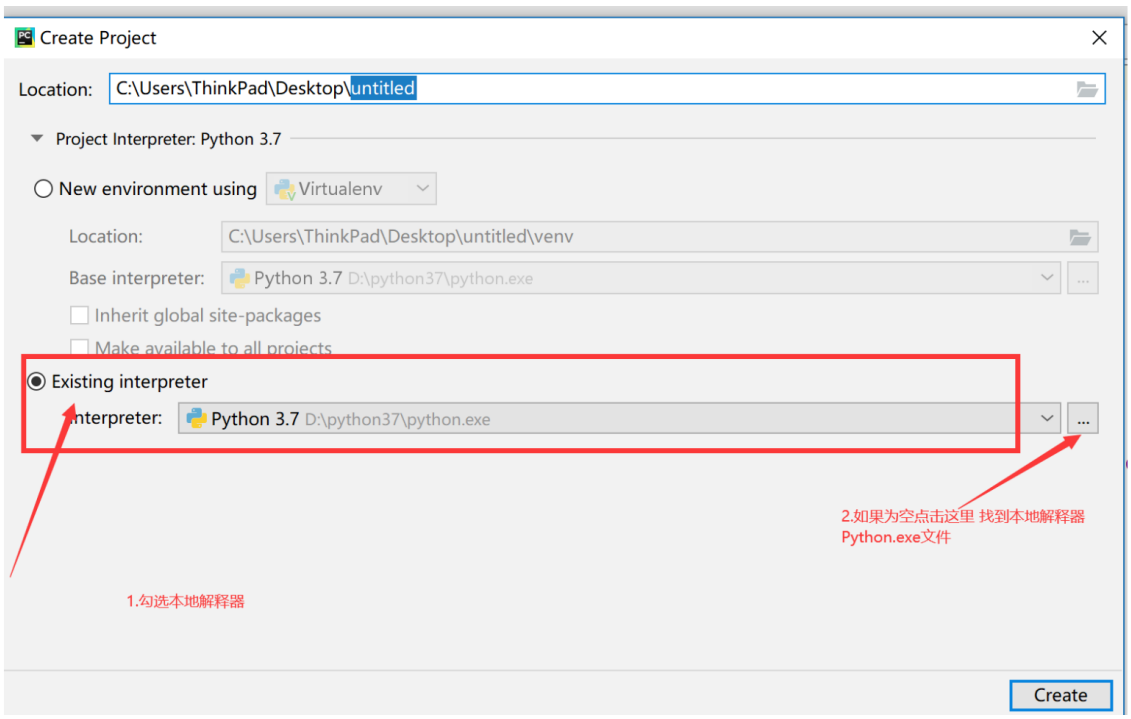
## 2.2 原理




# 3 浏览器的实例管理

## 3.1 通过程序驱动浏览器

- 项目创建
  - 项目名称不要与第三方的模块名同名
  - 文件名称也不要与第三方的模块名或者是类名同名
  - 项目创建时不要使用虚拟环境




代码实例：

```
导包

from selenium import webdriver

创建浏览器驱动对象

Firefox浏览器：driver = webdriver.Firefox()

Chrome浏览器：driver = webdriver.Chrome()
```

## 3.2 四个导航

```
get()

back()

forward()

refresh()
```

## 3.3 三个页面属性

```
title

current_url

page_source
```

## 3.4 两个关闭方法

```
close：关闭当前操作的窗口(并非关闭超链接新打开的窗口)

quit：关闭client与远端浏览器的会话--关闭该浏览器启动的所有窗口
```

## 3.5 一组管理窗口的方法

```
maximize_window

get_window_size

set_window_size
```

## 3.6 截图方法

为什么窗口截图：

自动化脚本程序执行，单纯看打印信息不是很准确，如果出粗对窗口截图，图片是可以很直观的看到错误

```
get_screenshot_as_file(imgpath)
```

截图动态获取

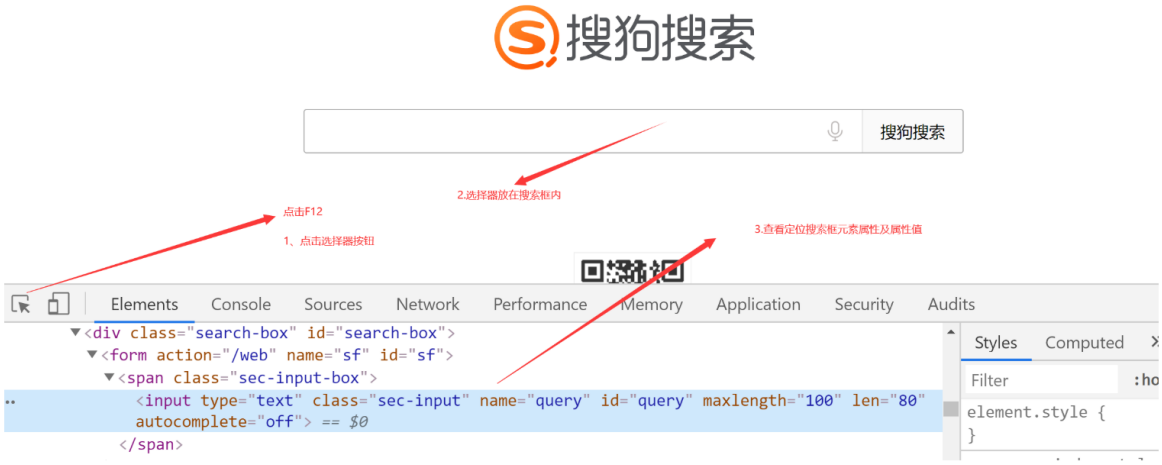
driver.get_screenshot_as_file('**{}.png**'.format(time.strftime('**%Y%m%d-%H%M%S**')))

# 4 Selenium 元素定位

## 4.1 浏览器开发者工具

浏览器开发者工具就是给专业的web应用和网站开发人员使用的工具。 包含了对HTML查看和编辑 Javascript控制台 网络状况监视等功能，是开发JavaScript CSS HTML和Ajax的得力助手。

作用：快速定位元素，查看元素信息




- 快捷键：一般在windows系统上打开浏览器开发者工具　--　按F12
- 火狐浏览器：在页面上点击右键选择'查看元素'
- 谷歌浏览器：在页面上点击右键选择'检查'

如何进行元素定位

元素定位：通过元素的信息或元素层级结构来定位元素的

- 元素的信息：指元素的标签名以及元素的属性　id　class　name
- 元素的层级结构：指元素之间相互嵌套的层级结构

## 4.2 Selenium八种定位元素方法

Selenium提供了八种定位元素方式

```
id

name

class_name

tag_name

link_text

partial_link_text

XPath

CSS
```

### 4.2.1 ID定位

id定位：通过元素的id属性来定位元素，HTML规定id属性在整个HTML文档中必须是唯一的

需求：驱动谷歌浏览器 打开搜狗浏览器　搜索框输入'拉勾教育'

- id定位方法

```
find_element_by_id(id) # id参数表示的是id的属性值
```

- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 定位搜索框,输入拉勾教育

driver.find_element_by_id('query').send_keys('拉勾教育')
```

### 4.2.2 name定位

name定位：通过name属性进行定位，name属性可以重复

- 定位方法

  ```
  find_element_by_name(name) # name 参数表示的是name的属性值
  ```

- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 定位搜索框,输入拉勾教育
```

```
# driver.find_element_by_id('query').send_keys('拉勾教育')

# 使用name属性定位

driver.find_element_by_name('query').send_keys('拉勾教育')
```

### 4.2.3 className定位

通过元素的class属性值进行元素定位 class属性值是可重复的

- 定位方法

```
find_element_by_class_name(class_name) # class_name参数表示的是class的其中一个属性值
```

- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 定位搜索框,输入拉勾教育

# driver.find_element_by_id('query').send_keys('拉勾教育')

# 使用name属性定位

# driver.find_element_by_name('query').send_keys('拉勾教育')

# 使用classname定位

driver.find_element_by_class_name('sec-input').send_keys('拉勾教育')
```

### 4.2.4 tag_name定位

通过标签名称进行定位， 在同一个html页面当中，相同标签会有很多

如果有重复的标签，定位到的元素默认都是第一个标签

需求:

- 定位方法

```
find_element_by_tag_name(tag_name) #tag_name表示的是元素的标签名称。
```

- 案例实现 通过tag-name方法 打印网页超链接文本信息

```
# 导包
from selenium import webdriver
```

```
# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')


# 通过tag-name 定位

print(driver.find_element_by_tag_name('li').text)
```

### 4.2.5 link_text定位

通过超链接的全部文本信息进行元素定位 ,主要用来定位a标签

- 定位方法

```
find_element_by_link_text(link_text) # link_text 参数代表的是a标签的全部文本内容
```

需求：搜狗浏览器通过link-text方法点击微信超链接

- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 通过link_text 定位

driver.find_element_by_link_text('微信').click()
```

### 4.2.6 partial_link_text定位

通过超链接的局部文本信息进行元素定位，主要用来定位a标签

- 定位方法

```
find_element_by_partial_link_text(partial_link_text)
```

- partial_link_text表示的是a标签 的局部文本内容
- 需求（同上）
- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 通过link_text 定位  输入'信'

driver.find_element_by_partial_link_text('信').click()
```

### 4.2.7 定位一组元素

- 定位一组元素的方法

  ```
  find_elements_by_tag_name(tag_name)
  ```

- 定位一组元素返回的值是一个列表
- 可以通过下标来使用列表中的元素，下标是从0开始
- 需求 通过li标签定位一组元素，通过索引方法对知乎链接点击
- 案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开搜狗浏览器

driver.get('https://sogou.com')

# 定位一组元素,通过索引点击知乎

el = driver.find_elements_by_tag_name('li')
el[2].click()
```

### 4.2.8 Xpath定位

- XPath即为XML Path的简称，它是一门在 XML 文档中查找元素信息的语言
- HTML可以看做是XML的一种实现，所以Selenium用户可以使用这种强大的语言在Web应用中定位元素

- 定位思路
  - 路径-定位
  - 利用元素属性-定位
  - 层级与属性结合-定位

    举例：https://www.w3school.com.cn/example/xmle/books.xml

- 定位方法

```
driver.find_element_by_xpath(xpath)
```

- 路径定位

  - 绝对路径：从最外层元素到指定元素之间所有经过元素层级的路径

    1). 绝对路径以/html根节点开始，使用/来分隔元素层级；

    如：/html/body/div/fieldset/p[1]/input

  案例实现 商城首页使用绝对路径方式定位搜索框-输入框输入‘手机’

导包

```
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开'拉勾商城'

driver.get('http://localhost:8081/')

# 定位搜索框，输入手机，使用xpath方法

driver.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/form/input').send_
keys('手机')
```

- 相对路径：匹配任意层级的元素，不限制元素的位置

  1). 相对路径以//开始

  2). 格式：//input

- 案例实现 拉勾商城首页使用相对路径方式定位搜索框-输入框输入‘手机’

```
  # 导包
```

```
  from selenium import webdriver

  # 实例化浏览器

  driver = webdriver.Chrome()

  # 打开'拉勾商城'

  driver.get('http://localhost:8081/')

  # 定位搜索框，输入手机，使用xpath方法
  # 绝对路径

  #
driver.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/form/input').send_
keys('手机')
  # 相对路径

  driver.find_element_by_xpath('//input').send_keys('手机')
```

- 利用元素属性定位 拉勾商城首页使用属性值定位搜索框-输入框输入‘手机’

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开'拉勾商城'

driver.get('http://localhost:8081/')

# 通过属性定位  id

driver.find_element_by_xpath('//input[@id="q"]').send_keys('手机')
```

- 利用层级与属性结合 商城首页定位搜索框-输入框输入‘手机’

  同级节点 或者 兄弟节点  父级节点

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开'拉勾商城'

driver.get('http://localhost:8081/')

# 通过层级和属性结合
```

```
driver.find_element_by_xpath('//form/input[@id="q"]').send_keys('手机')
```

- **xpath扩展-了解**

  一个/是绝对路径，从根元素开始

  两个//是相对路径，递归查找所有子孙

  一个. 是当前层，两个.是上一层

  @表示取属性

  []叫谓语，里面跟的是查询条件

  条件支持算数运算+-*/><，条件支持逻辑运算 and or

  取文本值用text(),不加@因为它是个函数

  常用函数：contains(属性名或者节点名，文本值)

  text()是取文本值，也可以当做一个查询条件

  last()取末尾，倒数第二last()-1

  starts-with(),表示以XX开头，写法是括号加两个入参

  not(),表示否定，把内容全包进去

  count(),取节点或属性个数

  **xpath练习题**

  1.选取 第一本书的定价

  2.选取 最后一本书的作者

  3.选取 定价在30到40之间的书的标题

  4.选取 作者多于一个的书的标题

  5.选取 分类不是web 且价格低于40的书的作者

  6.选取 标题名称包含X的 所有书的定价值

  7.选取 单个作者 且 分类是web的书的出版年份值

  8.选取 多个作者 且 标题是以X开头的 书的定价

  9.选取 分类是web 且作者不包括James的所有书的标题

### 4.2.9 CSS定位

- CSS是一种语言，它用来描述HTML元素的显示样式；
- 在CSS中，选择器是一种模式，用于选择需要添加样式的元素；
- 在Selenium中也可以使用这种选择器来定位元素。
- 提示：

  CSS定位比XPath定位速度要快

- 定位思路

    - id选择器
    - class选择器
    - 属性选择器
    - 层级选择器

需求：拉勾商城首页定位搜索框，输入‘手机’

**ID定位** #

案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器

driver = webdriver.Chrome()

# 打开'拉勾商城'

driver.get('http://localhost:8081/')

css定位 id

driver.find_element_by_css_selector('#q').send_keys('手机')
```

**Class定位** .

案例实现

```
# 导包
from selenium import webdriver

# 实例化浏览器
driver = webdriver.Chrome()

# 打开
'拉勾商城' driver.get('http://localhost:8081/')

# css定位 id
driver.find_element_by_css_selector('#q').send_keys('手机')

#css定位 class
driver.find_element_by_css_selector('.ecsc-search-input').send_keys('手机')
```

属性定位实现 []

```
打开'拉勾商城'

  driver.get('http://localhost:8081/')

css定位 id

driver.find_element_by_css_selector('#q').send_keys('手机')

css定位 class

driver.find_element_by_css_selector('.ecsc-search-input').send_keys('手机')

属性定位 []

driver.find_element_by_css_selector('[name="q"]').send_keys('手机')
```

层级选择器

```
driver.find_element_by_css_selector('#q').send_keys('手机')

css定位 class

driver.find_element_by_css_selector('.ecsc-search-input').send_keys('手机')

属性定位 []

driver.find_element_by_css_selector('[name="q"]').send_keys('手机')

层级定位

driver.find_element_by_css_selector('form[id=searchForm] > input').send_keys('手机')
```

# 5 页面元素处理

## 5.1 找到元素后操作

点击操作:

```
element.click() element表示的是元素对象
```

输入操作:

```
element.send_keys("value") element表示的是元素对象， value表示的是要输入的内容
```

清除操作:

```
element.clear() element表示的是元素对象. 将输入框里面的内容全部清除
```

## 5.2 获取常用元素方法

### 5.2.1 text

获取元素的文本内容

案例 -- 获取登录文本信息

```
导包

from selenium import webdriver

实例化浏览器

driver = webdriver.Chrome()

打开'拉勾商城'

driver.get('http://localhost:8081/')

text 打印登录信息

print(driver.find_element_by_link_text('登录').text)
```

### 5.2.2 get_attribute("attribute")

获取元素对应属性名称的属性值， attribute表示的是属性名

案例 -- 获取登陆超链接的

```
导包

from selenium import webdriver

实例化浏览器

driver = webdriver.Chrome()

打开'拉勾商城'

driver.get('http://localhost:8081/')

text 打印登录信息

print(driver.find_element_by_link_text('登录').text)

获取属性值

print(driver.find_element_by_link_text('登录').get_attribute('href'))
```

### 5.2.3 is_enabled()

判断元素是否可用，返回值为true或者false

案例 -- 点击注册，定位协议复选框判断

```
导包

from selenium import webdriver

实例化浏览器

driver = webdriver.Chrome()

打开'拉勾商城'

driver.get('http://localhost:8081/')

点击注册

driver.find_element_by_link_text('注册').click()

判断元素是否可用  注册按钮

print(driver.find_element_by_css_selector('[class="regbtn
J_btn_agree"]').is_enabled())
```

案例 -- 点击注册，定位协议复选框判断

### 5.2.4 is_selected()

判断复选框或者单选框是否被选中，返回值为true或者false

```
导包

from selenium import webdriver

实例化浏览器

driver = webdriver.Chrome()

打开'拉勾商城'

driver.get('http://localhost:8081/')

点击注册

driver.find_element_by_link_text('注册').click()

判断单选框是否选中
```

```
print(driver.find_element_by_css_selector("[class='iyes fn-fl
J_protocal']").is_selected())
```