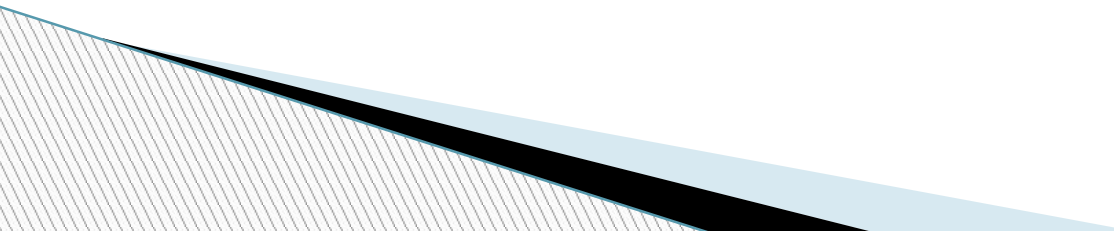


# Python基础

凡猫学院

# 知识点

- ▶ 1.python环境搭建（mac、Windows）
  - ▶ 2.第一个Python程序
  - ▶ 3.Python输入输出
  - ▶ 5.常用数据类型
  - ▶ 6.循环和条件判断
  - ▶ 7.函数式和面向对象编程
- 

# Python开发环境搭建

▶ 安装python

▶ 安装编辑器Pycharm

Ps:所有安装包以及依赖工具和安装文档

<https://pan.baidu.com/s/1tFBXOSIZjPyK35dXcaVNxw>

<http://npm.taobao.org/mirrors/chromedriver/>

密码: fdug

# 安装时注意事项:

- ▶ Python2和Python3可以共存
- ▶ Mac:
  - 自带python2.版本
  - 查看python安装路径: 打开终端→输入 `which python`
- ▶ Windows:
  - 勾选 `add python to path`
  - 查看python安装路径: 打开终端→输入 `where python`
- ▶ 安装成功:
  - `Cmd`→输入 `python`

# 第一个Python程序

## ▶ 输入与输出

# Python输出

- ▶ `print()`在括号中加上字符串，就可以向屏幕上输出指定的文字。

例如：`print('hello, world')`

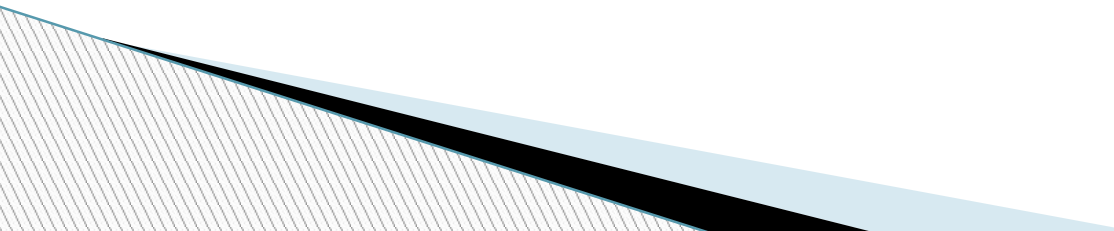
- ▶ `Print()`函数也可以接受多个字符串，用逗号“,”隔开，就可以连成一串输出：

例如：`print('python', 'java', 'js')`

## python输入：input()

- ▶ Python提供了input可以让用户输入字符串，
- ▶ 并存放到一个变量里。
  - 例如：
  - 输入用户的名字： `name = input()`
  - `Print (name)`

# Python数据类型

- 1、字符串
  - 2、布尔类型
  - 3、整数
  - 4、浮点数
  - 5、数字
  - 6、列表
  - 7、元组
  - 8、字典
- 



# 字符串

- ▶ 字符串是以单引号‘或双引号’括起来的任意文本。
- ▶ 例如：'abc'，"xyz", "123"等等。
- ▶ 注意："或'"本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有a，b，c这3个字符

如果'本身也是一个字符，那就可以用"括起来，  
比如"I'm OK"包含的字符是I，'，m，空格，OK这6个字符

# 转义字符\

▶ 如果字符串内部既包含'又包含"怎么办? 可以用转义字符\来标识

- 例如: 某个字符串包含” I’m “ok” ” 中间的’ “ 需要加上转义字符
- `Print('I\'m \"OK\"!')`

# 整数

- ▶ Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样

例如：1，100，-1000

a=1,

print(type(a)) 查看变量a的数据类型

# 浮点数

- ▶ 浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的。
- ▶ 对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代
- ▶ 例如：  
12300000 可以写成 $1.23e7$   
0.000012 可以写成 $1.2e-5$

# 列表 (list)

- ▶ list是一种有序的集合，可以随时添加和删除其中的元素，用[]表示

- 例如：

```
names = ['Michael', 'Bob', 'Tracy']
```

```
names = ['Michael', 'Bob', 'Tracy']
```

- 变量names就是一个list。用len()函数可以获得list元素的个数： len(names )
- 用索引来访问list中每一个位置的元素，记得索引是从0开始的： names [0]
- 注意： 数组越界，经常导致bug产生

# 字典(dictionary)

- ▶ 字典是除列表之外python中最灵活的内置数据结构类型。
- ▶ 列表是有序的对象结合, 字典是无序的对象集合。
- ▶ 两者之间的区别在于: 字典当中的元素是通过键-值(key-value)存储和取值。列表通过偏移存取。

# 一个简单的字典实例：

- ▶ `dict = {'Name': 'JH', 'Age': 7, 'Class': 'First'}`
- ▶ 访问字典里的值

```
print "dict['Name']: ", dict['Name'];  
print "dict['Age']: ", dict['Age'];
```

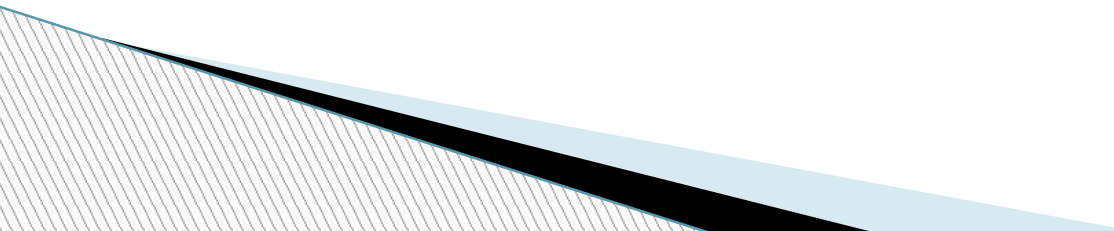


# 字典和列表的特点

## ▶ dict :

- ▶ key必须是不可变对象（哈希算法Hash）
- ▶ 查找和插入的速度极快，不会随着key的增加而变慢
- ▶ 需要占用大量的内存，内存浪费多

## ▶ List :

- ▶ 查找和插入的时间随着元素的增加而增加
  - ▶ 占用空间小，浪费内存很少
- 

# 切片

## ▶ 参数:

通常一个切片操作要提供三个参数 [start\_index: stop\_index: step]

start\_index: 切片的起始位置

stop\_index: 切片的结束位置

step: 步长值, 不能为0, 可以不提供, 默认值是1

## ▶ 特点:

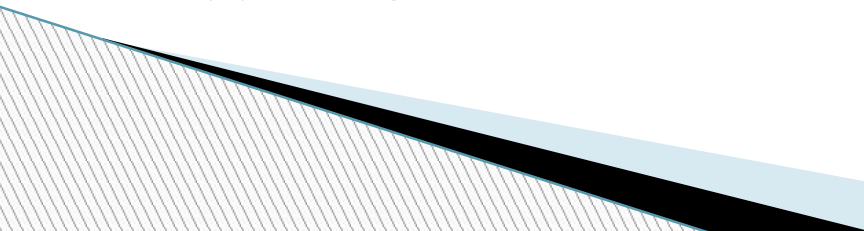
- ▶ 始终包含开始, 始终排除结尾

- ▶ 省略的第一个索引默认为0, 省略的第二个索引默认为它的长度

# 练习1:

- ▶ 将world中间字母大写:
- ▶ 切片、upper、len()

# 循环：

- ▶ 为了让计算机能计算成千上万次的重复运算，我们就需要循环语句
  - ▶ Python的循环有两种：
    - ▶ for (for ... in... ;for ... in range...)
    - ▶ While
- 

# while:

- ▶ While 条件为真:
  - ▶ 循环体:
- ▶ 真: 1.任何非零整数为True,零为False
- ▶ 2.任何长度不为0的东西为真,空序列为错的

## 练习2：用while计算【1， 100】的和

- ▶ `n=0;sum=0`
- ▶ `while n<101:`
- ▶ 用while:求x的n次方 例如：  $(2^3)$

# for...in

- ▶ 例如：依次把list中的每个元素迭代出来，

```
names = ['Michael', 'Bob', 'Tracy']
```

```
for name in names:
```

```
    print(name)
```

执行这段代码，会依次打印names的每一个元素：

Michael

Bob

Tracy

再比如我们想计算1-10的整数之和，可以用一个sum变量做累加：

```
sum = 0
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    sum = sum + x
print(sum)
```

如果要计算1-100的整数之和，从1写到100有点困难，好Python提供一个range()函数



# For X in range()

- ▶ `sum = 0`
- ▶ `for x in range(101):`
- ▶     `sum = sum + x`
- ▶ `print(sum)`

# 条件判断：if

- ▶ 计算机之所以能做很多自动化的任务，因为它可以自己做条件判断
- ▶ if 满足条件判断1:
  - ▶     Action1
- ▶ else:
  - ▶     Action2
- ▶ 如果条件为True,执行action1，反正执行action2，
- ▶ else是可选的

# 列如： `abs()`函数

- ▶ `x=input()`
- ▶ If `x>0`:
- ▶     Return `x`
- ▶ Else :
- ▶     Return `x`

# elif:

- ▶ 可以做更精致的判断；避免过多的缩进
- ▶ If 条件判断1:
  - ▶ Action1
- ▶ Elif 条件判断2:
  - ▶ Action2:
- ▶ Elif 条件判断3:
  - ▶ Action3
- ▶ Else:
  - ▶ Action4

# 函数

- ▶ 内置函数：列如
  - ▶ 求绝对值的函数abs()
  - ▶ 最大值max()
  - ▶ 获取list的长度len()
  - ▶ 随机数random.randint()
  - ▶ 数据类型转换int(),str()

# 自定义函数

- ▶ 定义一个函数使用def语句
- ▶ 依次是函数名、括号、括号中的参数和冒号；然后在缩进块中编写函数体
- ▶ 函数的返回值用return语句返回,一旦执行到return时，函数就执行完毕

# 练习：

- ▶ 学完Python的输入输出、数据类型、for循环，我们就可以写出一个简单算法来看看效果
- ▶ 练习：
- ▶ `list=['javac','java','pythonc','python','php','nodec']`
- ▶ 如果list中的元素的长度为奇数，将最中间的字母大写：
- ▶ 输出结果：  
`['jaVac','java','pytHonc','python','pHp','noDec']`

# 第一个自动化脚本

## ▶ 安装Selenium

- Windows: `Pip install selenium`

- Mac: `sudo pip install selenium`

## ▶ 查看selenium版本信息：

`Pip show selenium`



# 三行代码自动用浏览器打开百度首页

- ▶ Python+selenium:
- ▶ `from selenium import webdriver`
- ▶ `driver=webdriver.Chrome()`  
`driver.get("www.baidu.com")`

# 面象对象技术

- ▶ 类(Class)

- ▶ 类是抽象的：用来描述具有相同属性和功能方法的对象的集合

- ▶ 例如：人类；学生类；员工类；

- ▶ 相同的属性：眼睛，成绩，编号

- ▶ 相同的功能：吃饭，上课，打开



# 实例

- ▶ 1.根据类创建出来的具体的对象
- ▶ 2.每个对象有相同的属性和方法，但是属性的值都不一样，相互独立
- ▶ 例如： 员工类
  - ▶ 属性值： 名称、编号，工作年限

# 实例化

- ▶ 创建一个类的实例，类的具体对象
- ▶ 在其它编程语言中一般用new
- ▶ 在python中类似函数的调用

# 例如：创建类和实例

- ▶ 创建一个员工类：
- ▶ `class Employee:`
  - ▶ `pass #类体`
- ▶ 创建类的实例对象
- ▶ `emp1=Employee()`
- ▶ `emp2=Employee()`

# 给实例绑定一个属性

- ▶ `emp1=Employee()`
  - ▶ `emp1.name='zhangsan'`
  - ▶ `emp1.age='18'`
- 
- ▶ `emp2=Employee()`
  - ▶ `emp2.name='zhangsan'`
  - ▶ `emp2.age='18'`

# 类方法

- ▶ 类中定义的函数
- ▶ 第一个参数永远是self，其它和普通函数一样
- ▶ 调用：实例化对象.方法名称

# Self:

- ▶ 代表的是实例本身
- ▶ 例如:
- ▶ Class Test:
  - ▶ Def test(self):
    - ▶ print('self代表的是%s'%self)
- ▶ a=Test()
- ▶ print(a)
- ▶ print(a.test())



# \_\_init\_\_方法:

- ▶ 背景：由于类可以起到模板作用，我们在创建类的时候，可以将每个对象具有的相同的或必要的属性绑定到类中，通过特殊的\_\_init\_\_方法
- ▶ 再创建每个类的实例对象时，该对象就拥有这些属性了

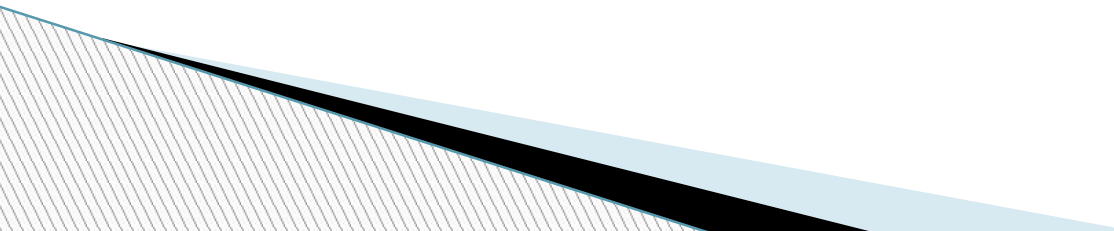
# \_\_init\_\_方法:

- ▶ 类中间特殊的构造方法:
- ▶ 1. 第一个参数永远是self
- ▶ 2. 每当创建一个类的实例化对象，就会自动去调用这个初始化的方法
- ▶ 3. 在init方法内部，可以将各种属性绑定到self上（因为self所指向的就是所创建的实例本身）

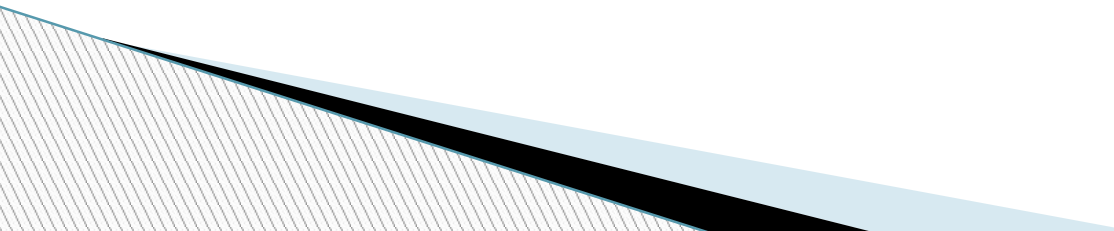
# 理解\_\_init\_\_

- ▶ 例如：
  - ▶ 员工类绑定属性name, age, years
- ▶ Class Employee:
  - ▶ def \_\_init\_\_(self,name,age,years)
    - ▶ Self.name=name
    - ▶ Self.age=age
    - ▶ Self.years=years
- ▶ emp3=Employee('zhangsan','18',3)
- ▶ #有了init方法后，创建实例的时候需要传入和init方法同样的参数，self不需要传，python解释器自动会把实例变量传进去

# 总结：

- ▶ 函数式编程：各函数相互独立，无共用数据
  - ▶ 面向对象编程：
    - ▶ 1.将数据和逻辑封装起来（增强安全性和简化编程）
    - ▶ 2.方便调用，不需要知道具体的实现细节
    - ▶ 3.代码重用，通过继承机制
- 

# 自动化测试： selenium

- ▶ 1.它可以让浏览器自动执行各种web应用，主要用于做自动化测试
  - ▶ 2.自动化管理基于web的无聊费事的重复工作
  - ▶ 3.无数测试工具、核心框架、api的技术支撑
  - ▶ 4.扩展（可用于爬虫）
- 

# 模拟手机浏览器：

- ▶ 用iphonex模式打开百度网址
- ▶ `mobile_emulation={"deviceName":"iPhone X"}`
- ▶ `options=webdriver.ChromeOptions()`
- ▶ `option.add_experimental_option("mobileEmulation",'mobile_emulation')`
- ▶ `driver=webdriver.Chrom()`
- ▶ `driver.get("https://www.baidu.com")`

# selenium基本操作

- ▶ 刷新: `refresh`
- ▶ 获取当前窗口大小: `get_window_size()`
- ▶ 设置窗口大小: `set_window_size('width', height)`
- ▶ 最大化窗口: `maximize_window()`
- ▶ 最小化窗口: `minimize_window()`
- ▶ 获取窗口个数: `get_handles`
- ▶ 切换窗口: `switch_to.handles(handle)`
- ▶ 前进: `forward`
- ▶ 后退: `back`
- ▶ 关闭浏览器: `close()`
- ▶ 退出浏览器: `quit()`

# 键盘操作： Tab、Enter

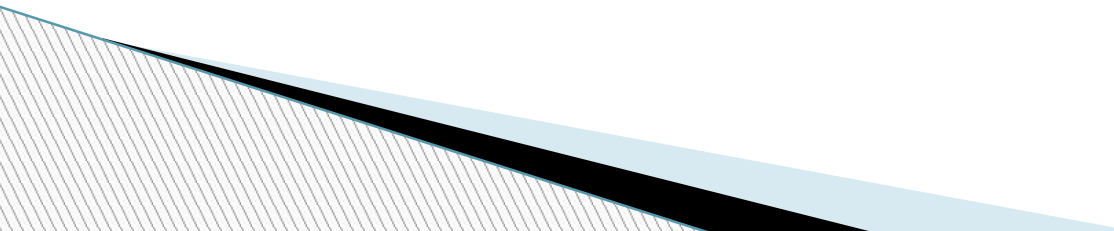
- ▶ 1.先导入selenium中的class:Key
- ▶ `from selenium.webdriver.common.keys import Keys`
- ▶ Tab: `driver.find_element_by_xpath("").send_keys(Keys.TAB)`
- ▶ Enter: `driver.find_element_by_xpath("").send_keys(Keys.ENTER)`



# 鼠标操作：

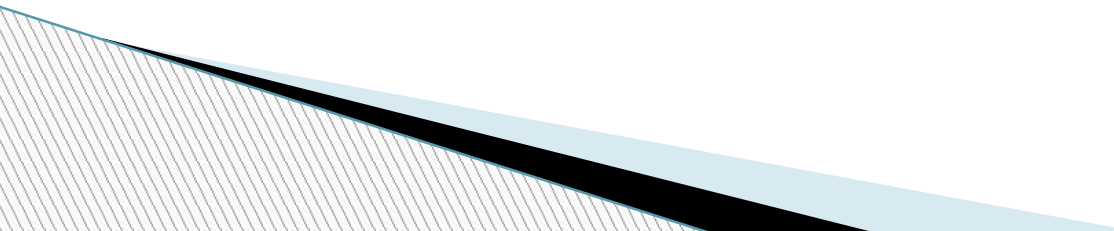
- ▶ 1.先导入selenium中的class:ActionChains
- ▶ `from selenium.webdriver.common.action_chains import ActionChains`
- ▶ ActionChains原理：
  - ▶ 调用该类里面的方法是，不会立即执行，将需要执行的动作按顺序存放在一个列队里，
  - ▶ 调用 `perform()`方法时，依次执行列队里的动作

# 双击、拖拽、鼠标移动

- ▶ 双击: `double_click`
  - ▶ 拖拽: `drag_and_drop`
  - ▶ 点击鼠标左键不松开: `click_and_hold`
  - ▶ 松开鼠标左键: `release`
  - ▶ 鼠标移至某个元素: `move_to_element`
- 

# 元素定位：

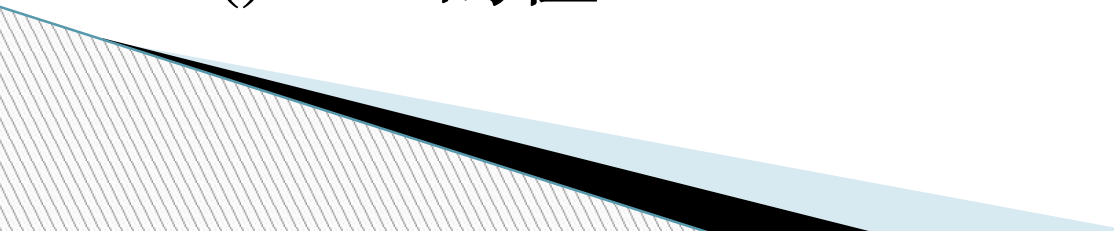
对于对于 Web 自动化测试来说，就是操作页面上的各种元素，在操作元素之间需要先找到元素，换句话说就是定位元素  
selenium常见定位元素8中方法：

- ▶ Id
  - ▶ Name
  - ▶ Classname
  - ▶ Tagname
  - ▶ linkText
  - ▶ partialLinkText
  - ▶ Xpath
  - ▶ cssSelector
- 

# xpath:path就是路径，类似文件夹Desktop/ classnotes/

- ▶ 绝对路径定位：copy xpath
  - ▶ 缺点：当页面元素父级元素发生改变，页面元素的位置发生改变时，都需要修改
- ▶ 相对路径定位：以//开头+标签名
  - ▶ 优点：长度和标签开始的位置并不受限制，稳定且万能

# 定义：

- ▶ // :表示相对路径
  - ▶ \* : 标签名任意
  - ▶ ../:表示找上级
  - ▶ / :表示找下级
  - ▶ @:标签属性定位
  - ▶ () :text的值
- 

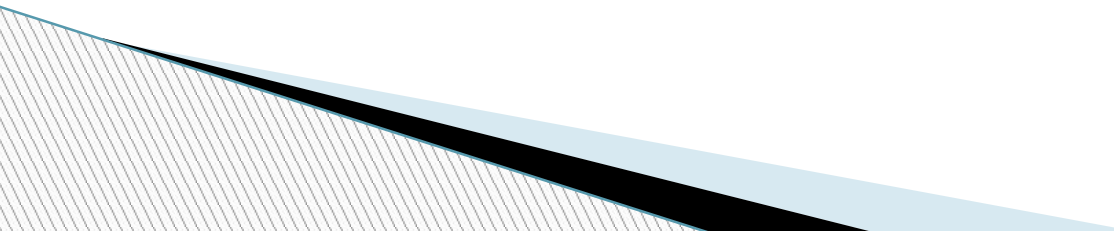
# SVG:可伸缩矢量图形 (Scalable Vector Graphics)

- ▶ 定位svg元素，需要用到xpath里面的name()方法：
  - ▶ 例如： `*[name()='svg']`

# frame标签：

- ▶ frame标签有三种:frameset、frame、iframe
- ▶ frameset跟其他普通标签没有区别，不会影响到正常的定位，而frame与iframe需要切换进去才能定位到其中的元素"
- ▶ 通过switch\_to\_frame(id,name,frame的xpath)
- ▶ 跳出frame: switch\_to\_default\_content()

# unittest:

- ▶ unittest是Python自带的单元测试框架，我们可以用其来作为我们自动化测试框架的用例组织执行框架。
  - ▶ 1.组织用例
  - ▶ 2.测试用例灵活的执行
  - ▶ 3.验证测试结果
  - ▶ 4.集成测试报告
- 



# 为什么要学unittest:

- ▶ 1.方便用例运行
- ▶ 2.如果有fail的用例不影响后面的用例运行

# unittest:

- ▶ 1.一个class继承unittest.TestCase(所有测试用例类继承的基本类)
- ▶ 2.测试方法均以 test 开头，否则是不被unittest识别
- ▶ 3.每一个用例执行的结果的标识，成功是 .，失败是 F，出错是 E
- ▶ 4.verbosity参数可以控制执行结果的输出，0 是简单报告、1 是一般报告、2 是详细报告。
- ▶ 5.用 setUp()、tearDown()、setUpClass()以及 tearDownClass()可以在用例执行前布置环境，以及在用例执行后清理环境
- ▶ 6.参数中加stream，可以将报告输出到文件：可以用TextTestRunner输出txt报告，以及可以用HTMLTestRunner输出html报告
- ▶ 7.unittest.main():可以方便的将一个测试模块，变为可直接运行的测试脚本

# unittest流程

- ▶ a.写好TestCase (#创建一个TestSuite实例)
- ▶ b.可以通过addTest和addTests向TestSuite中添加case, (#组织测试用例)
- ▶ c.通过TextTestRunner来来运行TestSuite, 运行的结果保存在TextTestResult中,
  - ▶ 或者通过unittest.main()执行时, main会调用TextTestRunner中的run来执行

# 等待方式：强制等待、显性等待、隐形等待

- ▶ Sleep:强制等待几秒，执行下一步操作
- ▶ 隐式等待implicitly\_wait(x):在X时间内，页面加载完成，进行下一步操作
- ▶ 显式等待WebDriverWait:程序每隔X秒看一眼，如果条件成立了，则执行下一步，否则继续等待，直到超过设置的最长时间，然后抛出TimeoutException异常
  - ▶ WebDriverWait(driver, 超时时间, 调用频率).until(要执行的方法, 超时时返回的错误信息)

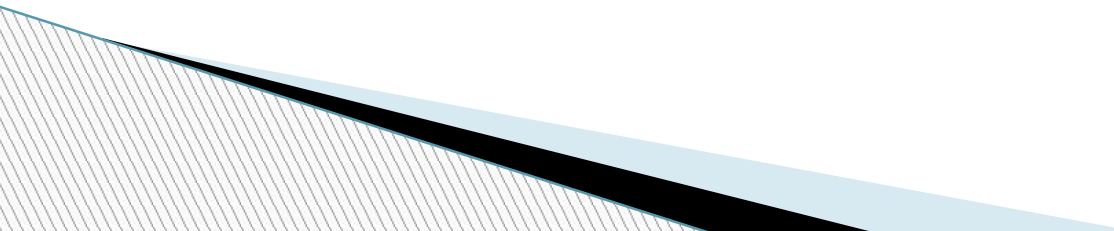
# 常用js方法：

- ▶ 首先selenium常用方法：
  - ▶ 1.判断元素是否存在： `is_displayed`
  - ▶ 2.获取元素的文本： `webelement.text`
  - ▶ 3.获取标题 `driver.title`
  - ▶ 4.获取元素的属性： `webelement.get_attribute('属性值')`

# javascript实现方法：

- ▶ JavaScript可以获取浏览器提供的很多对象，并进行操作。
- ▶ window就是一个对象：表示浏览器窗口
  - ▶ 1.打开新的浏览器窗口：window.open(url)
  - ▶ 浏览器内部宽度window.innerWidth
  - ▶ 浏览器内部高度window.innerHeight
  - ▶ 浏览器整体高度window.outerWidth
  - ▶ 浏览器整体高度window.outerHeight
  - ▶ 浏览器滚动条：window.scrollTo(0,1000)
    - ▶ window.By(0, 1000)
  - ▶ 非浏览器类型的滚动条：document.getElementById(id)[0].scrollTop='1000'

# location当前页面的URL对象

- ▶ 1.获取当前url:location.href获取
  - ▶ 2.更改当前url:location.assign()
  - ▶ 3.刷新: location.reload()
- 

# document:表示当前页面对象

- ▶ HTML在浏览器中以DOM形式表示为树形结构document对象就是整个DOM树的根节点,然后去操作子节点
- ▶ 1.获取当前标题: `document.title`
- ▶ 2.输入文本值: `document.getElementById(id)[0].value="`
- ▶ 3.操作标签`document.getElementById(id)[0].click()`
- ▶ 4.更改属性:  
`document.getElementById('vip').style.visibility='visible'`



# SMTP邮件发送：

## 发送邮件的协议，Python内置对SMTP的支持

- ▶ 模块email：构建邮件
  - ▶ `from email.mime.text import MIMEText`
    - ▶ MIMEText:邮件的信息:
      - ▶ `plain`表示纯文本
      - ▶ `as_string()` 把MIMEText 对象变成str
  - ▶ `from email.mime.multipart import MIMEMultipart`
    - ▶ MIMEMultipart:构造附件
    - ▶ 服务端向客户端浏览器发送文件时，如果需要提示用户保存，就要利用Content-Disposition进行一下处理
  - ▶ `from email.header import Header`
    - ▶ Header:设置头文件信息的格式编码
- ▶ 模块smtplib：发送邮件
  - ▶ `Import smtplib`

# 增加附件：

- ▶ 导入：`from email.mime.multipart import MIMEMultipart`
- ▶ MIMEMultipart：代表邮件本身
  - ▶ 里面加上一个MIMEText作为邮件正文，
  - ▶ 再通过attach继续往里面加上表示附件的对象即可

# python3安装robotframework

- ▶ Pip install robotframework==3.0.2
- ▶ Pip install robotframework-selenium2library3.0.0
- ▶ Pip install seleniumlibrary==3.0.1
- ▶ Pip install robotframework-pabot==0.44
- ▶ Pip install robotframework-ride==2.0a1
  - ▶ Pip安装包所在路径: Lib-site-packages
- ▶ 演示ride工具

# Pycharm运行RF

- ▶ 一：安装插件intellibot
  - ▶ file-setting-plugins-Browse repositories
  - ▶ 安装IntelliBot
- ▶ 执行脚本：
  - ▶ File-setting-Tools-External tools
    - ▶ Parameters:-d results \$FileName\$
      - ▶ (-d results -t "\$SelectedText\$" ./)
    - ▶ Working directory:\$FileDir\$

# RF基本语法和关键字

- ▶ 注释: #
- ▶ 等待时间: sleep
- ▶ Log:等于print
- ▶ 导入库: Library
- ▶ Open browser:
- ▶ Click element:
- ▶ Input text:
- ▶ 定义变量: scalar
- ▶ Mouse over:
- ▶ Wait until element is visible (not)
- ▶ element should be visible (not)
- ▶ Element text should be
- ▶ Xpath should match x times
- ▶ Execute javascript
- ▶ Get element attribute      xpath=?@?
- ▶ <http://robotframework.org/Selenium2Library/Selenium2Library.html#Click%20Element%20At%20Coordinates>

# Evaluate

- ▶ 通过Evaluate来使用python语言中所提供的方法

- ▶ 例如：

- ▶ `${a}`      set variable      10
- ▶ `${b}`      set variable      8
- ▶ `${c}`      evaluate      `${a}+${b}`
- ▶ log      `${c}`

# 关键字驱动（用户关键字）

- ▶ \*\*\*Keywords\*\*\*
- ▶ \*\*\*variables\*\*\*
- ▶ 再resource导入
- ▶ 练习：郑州大学

# 支持python关键字开发

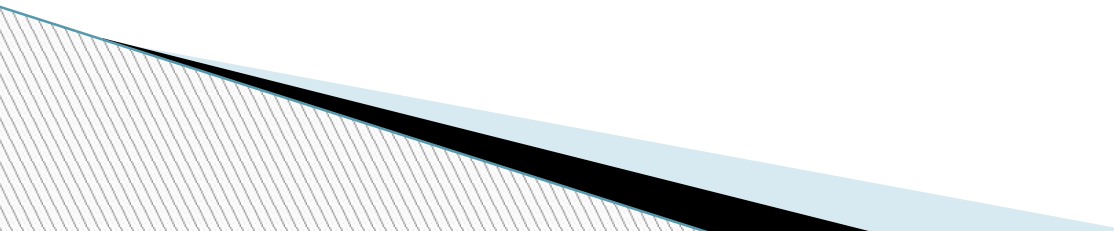
- ▶ Library    python文件路径
- ▶ 直接调用python函数



# RF常用关键字

- ▶ get text:参数: locator;
  - ▶ Returns the text value
- ▶ get matching xpath count (Get WebElements)
  - ▶ 参数: xpath
  - ▶ Returns number of elements matching `xpath`
- ▶ get element attribute
  - ▶ 参数: locator
  - ▶ Return value of element attribute
  - ▶ for example "element\_id@class"
- ▶ Execute javascript
  - ▶ 参数: \*jscode

# RF常用判断关键字

- ▶ 1.element should (not) be visible
  - ▶ 2.element text should be
  - ▶ 3.xpath should match x times
- 

# IF FOR

- ▶ IF用法:

- ▶ Run keyword if 条件 runkeyword
- ▶ ... ELSE IF runkeyword
- ▶ ... ELSE runkeyword

- ▶ 高阶用法: runkeywords

- ▶ AND

# FOR循环

- ▶ for循环用法
- ▶ `:FOR  $\{i\}$  IN RANGE 6`
- ▶ `\ Log  $\{i\}$`
- ▶ `:FOR  $\{i\}$  IN list`
- ▶ `\ log  $\{i\}$`

# Homework

'''

selenium: 将测试报告已邮件的形式发送: unittest htmltestrunner

RF: 将三个报告用邮件发送给我

作业1:

查询快递单号, 如果已经查询过; 删除该记录;

作业2:

定义一个数组: 里面有正确的单号, 错误的单号; 循环去查询数组里面的快递单号

如果是错误的单号, 继续输入;

如果是正确的, 进入查询记录页面

'''

