软件测试

# 1、测试对象：软件

1.1 需求分析阶段：各种需求规格说明书

1.2 软件架构设计：api接口文档（接口测试基础）

1.3 编码实现阶段：源代码（白盒测试、单元测试）

1.4 系统功能使用：软件功能主题

# 2、测试级别

2.1 单元测试（UT）：代码级别的测试，主要是类，函数，组件。

2.2 集成测试（IT）：接口测试，多个单元模块组合看是否能正常工作。

2.3 系统测试（ST）：最多的一种测试，测试员充当用户，对软件功能主题测试。

2.3.1 功能测试：当前软件主体共体功能是否可用

2.3.2 兼容性测试：软件在不同环境下是否能正常使用

2.3.3 安全测试：是否只对软件授权用户体现功能

2.3.4 性能测试：软件消耗的资源和产出比是否划算

2.4 验证测试：使用户为软件买单。

2.4.1 阿尔法a测试--内侧；

2.4.2 贝塔测试--公测；

2.4.3 UAT测试--用户组织人员来测试；

# 3、常见测试方法

3.1 从测试对象分

3.1.1 白盒测试：软件代码的测试 ，不在意界面，只求底层 功能实线，逻辑正确。

3.1.2 黑盒测试：外在功能测试。

3.1.2.1 功能测试：逻辑功能测试 、界面测试、易用性测试、安装测试、兼容性测试

3.1.2.2 性能测试：时间性能、空间性能、一般性能、稳定性、负载测试、压力测试

3.1.3 灰盒测试：代码和外在功能都测。

3.2 按测试对象是否执行分类

3.2.1 静态测试：不执行测试，文档，网页效果等。

3.2.2 动态测试：运行在真实环境测试。

3.3 按测试手段分

3.3.1 手工测试：优点是灵活 。

3.3.2 自动化测试：一种是自己写脚本，另一种是第三方工具。

3.4 其他

3.4.1 回归测试

3.4.2 冒烟测试

3.4.3 随机测试：对重要部分复测或新增部分重点测试，测试没有覆盖到的用例，特殊点，特殊环境，并发性等测试，尤其是以前发现过重大 bug的地方再次测试 。

3.4.4 验证测试

# 4、软件质量特性

4.1 功能性：满足用户显示或隐式功能

4.2 易用性：好用，易学

4.3 可靠性：必须实线软件需求当中的具体功能

4.4 效率性：高效

4.5 可维护性：修复后继续使用的功能

4.6 可移植性：从一个平台移植到另一个平台

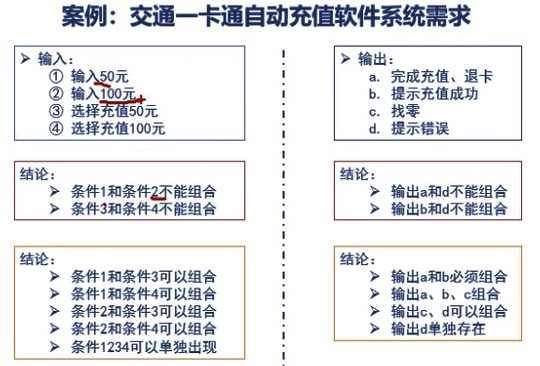
# 5、软件测试流程

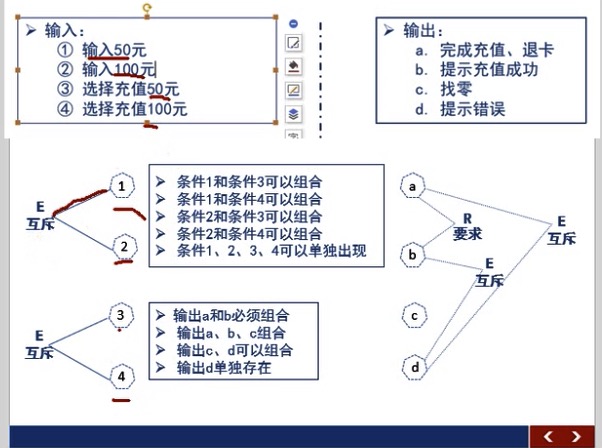
5.1 需求分析：梳理需要设计的点，来源于规格说明书、api文档、竟品分析、个人经验；

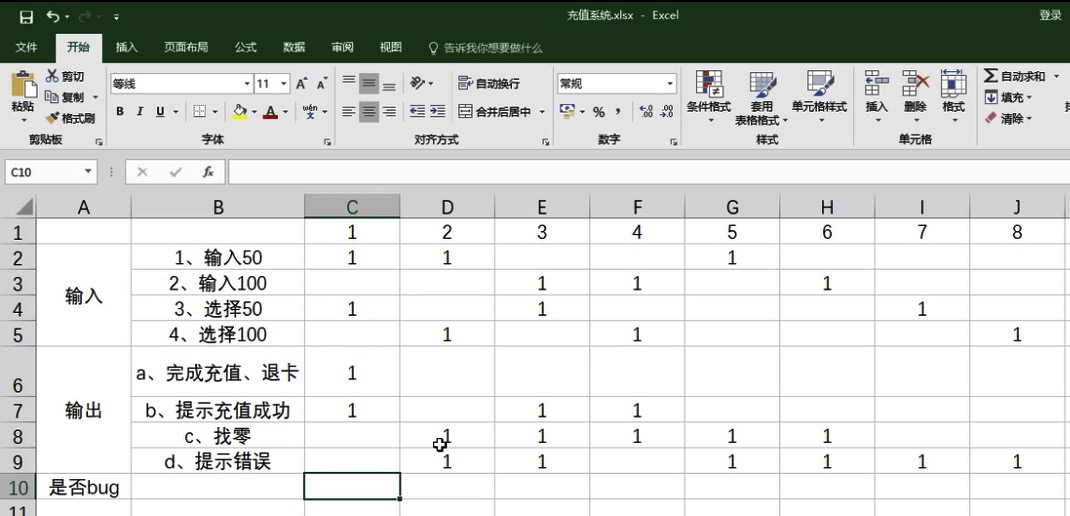
5.2 设计用例

5.2.1 等价类划分法：黑盒测试法，数据分类，每类挑选代表性数据来测试，分有效等价类（按条件选取，边界值一定要测） ，无效等价类（与条件相反、特殊符号、空）。

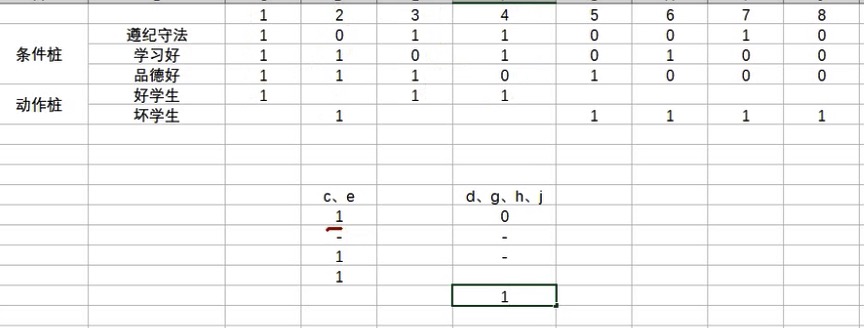
5.2.2 因果图法：条件的组合，防止重复测试或漏侧，有恒等关系、取反关系、或关系、与关系四种，约束条件有互斥、包含、屏蔽、唯一、要求五种，使用步骤（找所有因果，找因果制约条件及组合关系，因果图转化为判定表）。





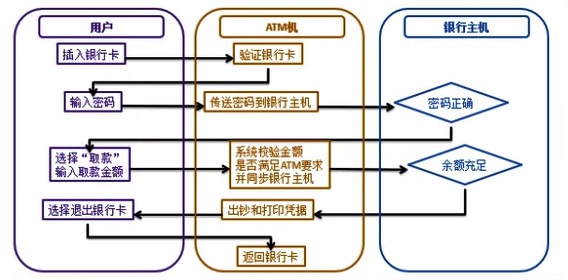


5.2.3 判定表：因果图的表格。可以合并用例条数，相同的合并。



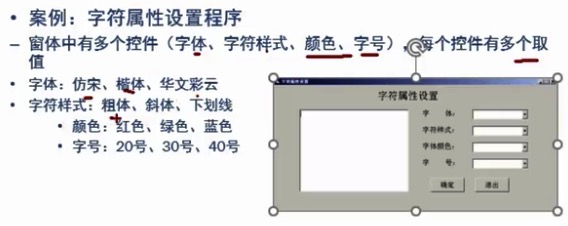
5.2.4 场景法：测试业务流程，模拟用户操作。基本流是正常操作，备选流是把错误场景操作一遍。

5.2.5 流程分析法：了解需求，找各页面及流转关系，画业务流程（pm做），写测试用例。正常流程，异常流程。



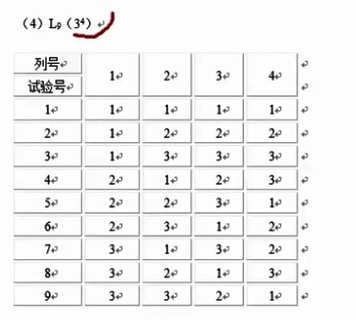
5.2.6 错误推断法：利用直觉和经验猜测错误，列出清单来测试。

5.2.7 正交表：使用最小的测试用例获得最大的测试覆盖率，时间紧迫可以用，但是为了全面性还是要全部用例都测的。可用软件allpairs自动生成表格。



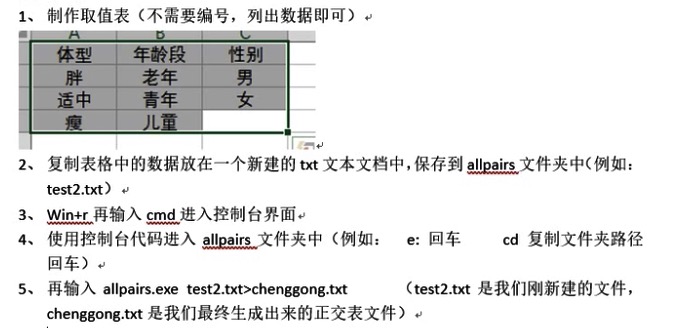
有81中，在其中找代表性的，正交表有表格可以去网上找，如

L9（3的4次方）等表格样式。





5.2.8 混合正交表：不是规则的正交表，用allpairs自动生成。



5.3 评审用例：对当前用例进行添加或删除。同行审核、用户审核

5.4 配置环境：操作系统+服务器+数据库+软件底层代码的执行环境。

5.5 执行用例：执行用例前会先做冒烟测试（过一遍主体执行流程或核心功能，有问题就返回rd），冒烟测试通过后才开展全面测试。

5.6 回归测试及缺陷跟踪：缺陷修复完成后需要再测试，缺陷发现后需要一直跟踪到结束缺陷。

5.7 输出测试报告：将当前测试过程产生的数据输出，方便别人查看。

5.8 测试结束：文件归档方便以后要用。

# 6、软件架构:指导软件开发的思想

6.1 b/s：浏览器到服务端，产品成熟所以更标准，浏览器负责展示，服务器负责处理，刷新就升级了。

6.2 c/s：客户端到服务端，客户端可以存储一些数据，效率高些，更安全些，要手动更新。

# 7、软件缺陷（没有满足用户需求）

7.1 类型：没达到说明书功能，出现说明书指明不出的错误，出现了说明书没要求的功能，出现说明书没说但是应该达到的功能，软件难理解难用速度慢或其他不好的。

7.2 表现形式：功能、特性没有实线或部分实线；产品和期望不一样；没达到期望性能指标；运行出错；硬件或软件上存在的其他问题；其他。

7.3 缺陷状态

7.3.1 提交

7.3.2 打开：确认提交的缺陷等待处理

7.3.3 拒绝：不需修复，重复缺陷，不是缺陷，无法重现等。

7.3.4 修复：缺陷被修复

7.3.5 关闭：确认修复的缺陷，将其关闭

7.3.6 推迟：以后解决但要确定修复日期和版本。

7.4 缺陷严重程度

7.4.1 low：表面性错误

7.4.2 medium：不是非常关键

7.4.3 high：功能点没实现或不符合需求，数据丢失

7.4.4 very high：死机，系统大部分不可用

7.4.5 critical：系统瘫痪，异常退出，死循环，计算错误等

7.5 软件缺陷信息：缺陷id，缺陷状态，缺陷标题，缺陷严重程度，缺陷优先级，缺陷所属模块，缺陷记录者，缺陷 提交时间，缺陷处理人，缺陷结果描述，缺陷处理时间，缺陷验证人，缺陷结果描述。

7.6 缺陷分类

7.6.1 系统缺陷：程序引起死机或异常退出，程序死循环，程序错误不能执行正常的工作或重要功能，使系统崩溃或资源不足。

7.6.2数据缺陷：数据计算、约束、输入输出错误。

7.6.3 数据库缺陷：数据库锁死，数据库的表、缺省值未加约束条件，数据库链接错误，数据库的表有过多空字段。

7.6.4 接口缺陷：程序接口错误或数据通信错误

7.6.5 功能缺陷：功能无法实线或实线错误

7.6.6 安全性缺陷：权限无法实现，超时限制错误，访问控制错误，加密错误

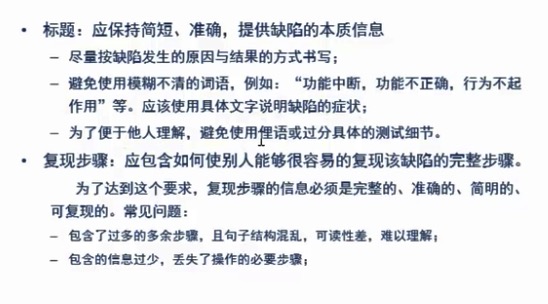
7.6.7 兼容性错误：与需求规定配置兼容性不符合

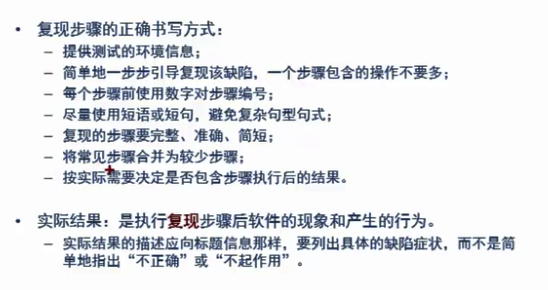
7.6.8 性能缺陷：未达到预期性能目标，性能测试中出错导致无法继续性能测试。

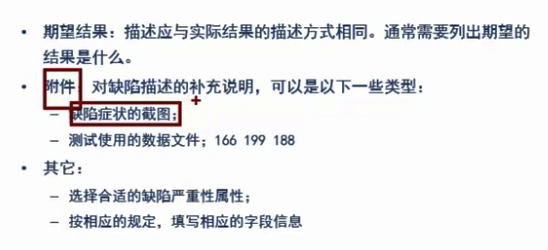
7.6.9 界面缺陷：界面操作错误，内容格式错误，删除操作美给提示，长时间操作没给提示，界面不规范。

7.6.10 建议：功能建议，操作建议

7.7 缺陷报告（简单，准确，专业的描述，缺陷尽量可重现，一个缺陷一个报告，发现错误要自己先确认是否是错误）

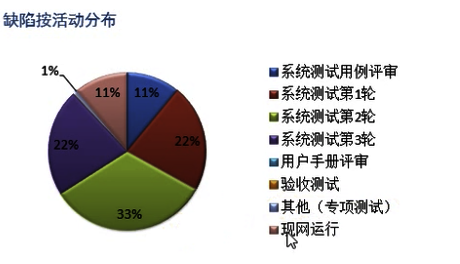


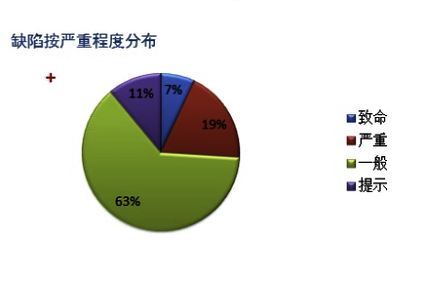


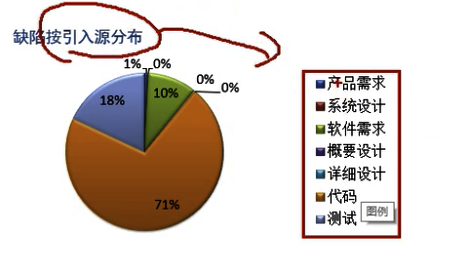




7.8 缺陷统计（领导统计）：缺陷密度是每千行代码缺陷个数。







7.9 常用找缺陷的方法

7.9.1 UI界面：色彩，功能布局，图片，页面大小，字体，窗体大小 ，文字

7.9.2 容错处理（功能缺陷，比如错误提示）

7.9.3 数据转换

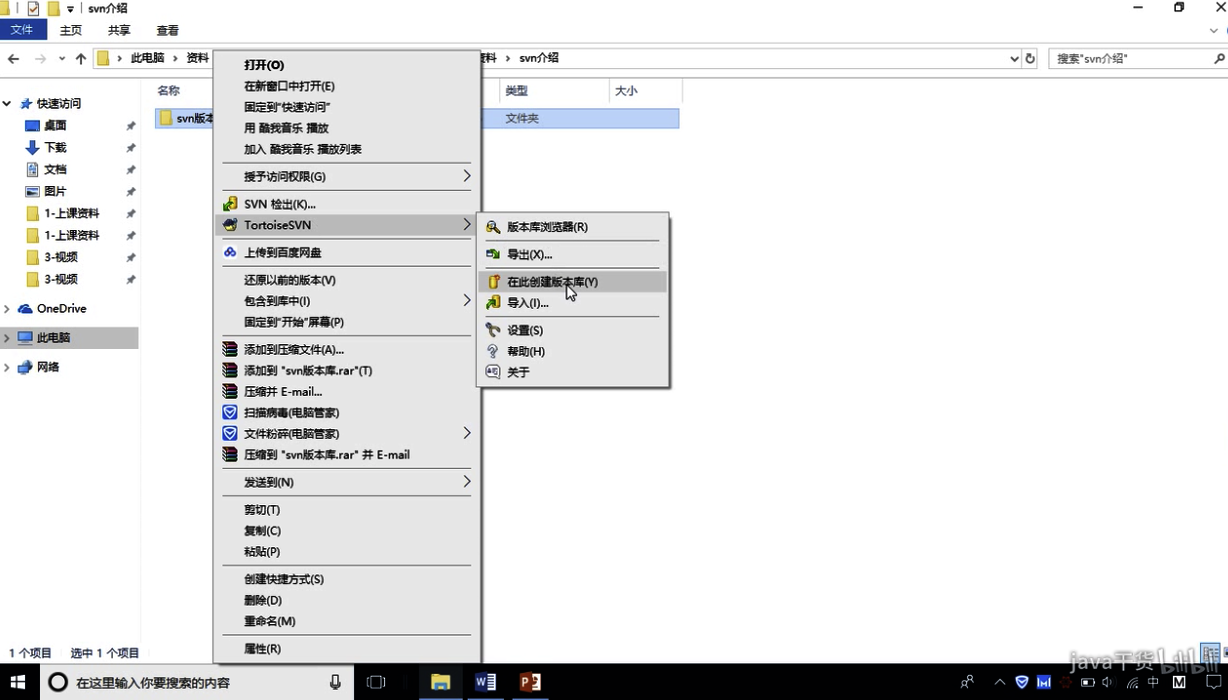
7.9.4 性能缺陷（打开慢，占内存大，压力测试不合格等）

# 8、SVN：版本管理软件

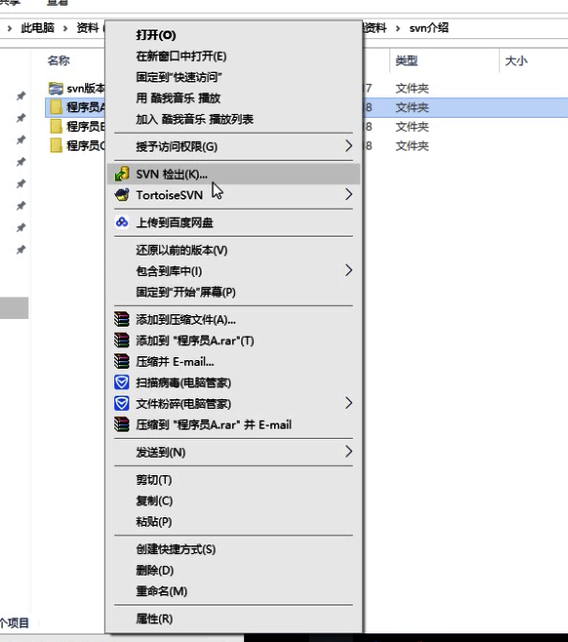
8.1 安装：下载安装包，安装好后重启电脑，选取文件夹作为访问文件，右键选tortoiseSVN 再选择设置再选择中文。

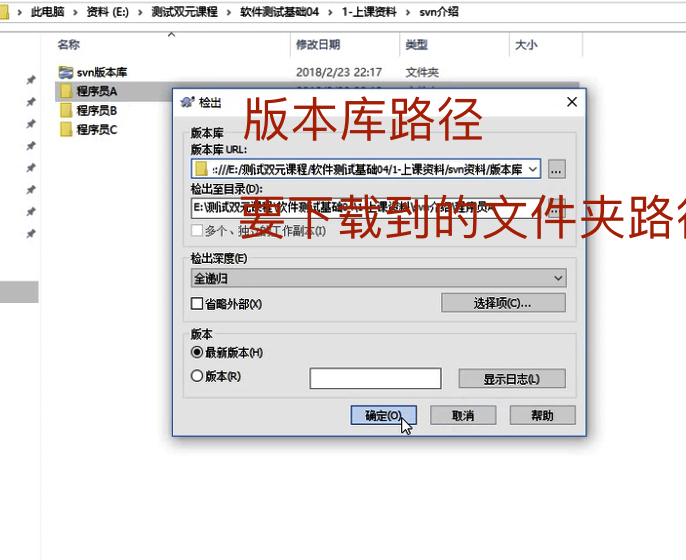
8.2 操作：

8.2.1 新建：创建文件夹，右键，生成版本库

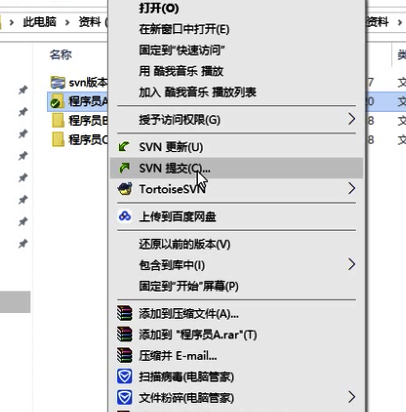


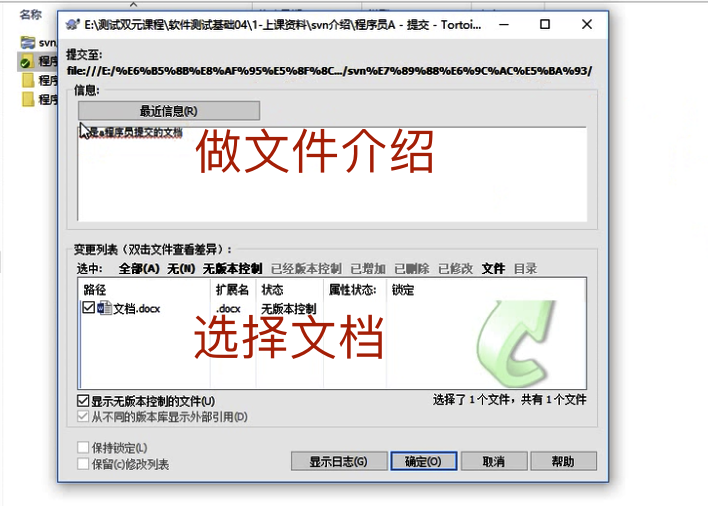
8.2.2 下载版本库内容到自己文件夹里





8.2.3 提交

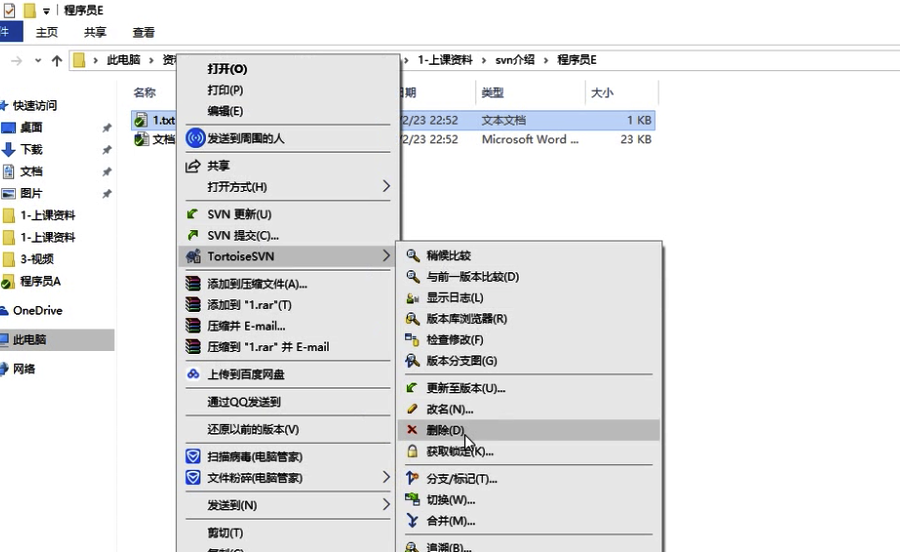


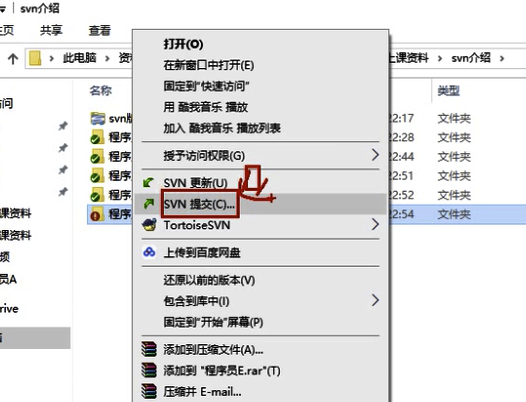


8.2.4 更新（下载了的文档改了后要再提交才会同步到版本库），已经下过的人要同步可右键点击跟新。

8.2.5 添加到版本库：要提交的文件右键提交

8.2.6 删除版本库内容：





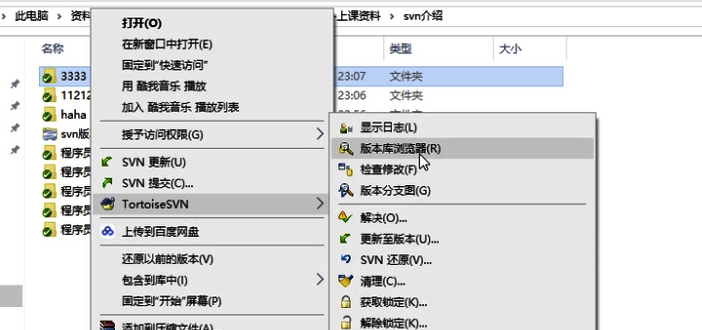
8.2.7 文件改名字



改完名字再提交到版本库

8.2.8 文件移动

随便一个文件夹，右键选版本库浏览器，移动文件，然后再提交一次



8.2.9 更新历史版本

下载历史版本，首先先检出，这样文件夹就受控制，然后右键选择想要的版本。



# 自动化测试

9.1 web自动化测试（需求变动不频繁，项目周期长，项目需要回归测试的可用自动化）

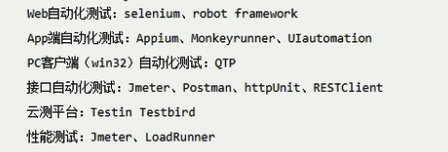
9.1.1 主流测试工具：QTP（收费）更多用于桌面自动化测试；

selenium（开源）web测试，主要用于功能测试；

Jmeter（开源、web、接口、性能）使用java平台开发；

Loadrunner（收费，web，性能）；

Robot framework是基于python的自动化测试框架;



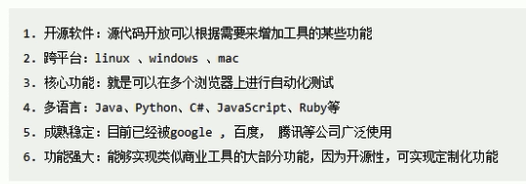
9.2 接口自动化测试

9.3 app自动化测试

9.4 单元自动化测试

# 10、selenium

10.1 selenium特点



10.2 selenium家族

Selenium 1.0 : selenium ide火狐插件，测试用例可转换成其他语言的自动化脚本，使用selenium IDE录制的脚本转换成代码语言 ；selenium girl；selenium rc；

Selenium2.0=selenium1.0+webdriver（有一套很好的API，是一套类库，不依赖任何测试框架，除了必要的浏览器驱动，支持java、python、php、js、perl、ruby、c#，用于自动化测试的定位和操作元素 ）；

Selenium 3.0

\*firebug插件是火狐浏览器一款插件，能 调试所有网站语言和快速定位html页面中的元素。

10.3 selenium-ide脚本编辑操作及常用命令

10.3.1 编辑命令：由commend、target、value组成。

10.3.2 插入命令：在某一条命令上右击选择insert new command命令就可插入一个空白，对空白进行编辑；

10.3.3 插入注释：在某一条命令上右击选择insert new comment命令就可插入一个空白，对空白进行编辑；

10.3.4 移动命令，选中拖拽；

10.3.5 删除命令：选中要删的一个或多个命令，右键delete；

10.3.6 打开路径：open（路径），绝对路径和相对路径都行；

10.3.7 暂停脚本运行：pause（暂停时间，单位毫秒）

10.3.8 后退：goBack（）没有参数是回退到上一步；

10.3.9 刷新页面：refresh（）没有参数；

10.3.10 单击：click（locator）单击一个链接、按钮、复选框、单选按钮等；

10.3.11 输入：type（locator，value）locator是哪里要输值，value是要输的值；

10.3.12 关闭操作：close（）

10.4 webdriver（有python3和pip后再安装，注意火狐版本，firebug、selenium-ide、驱动等安装）

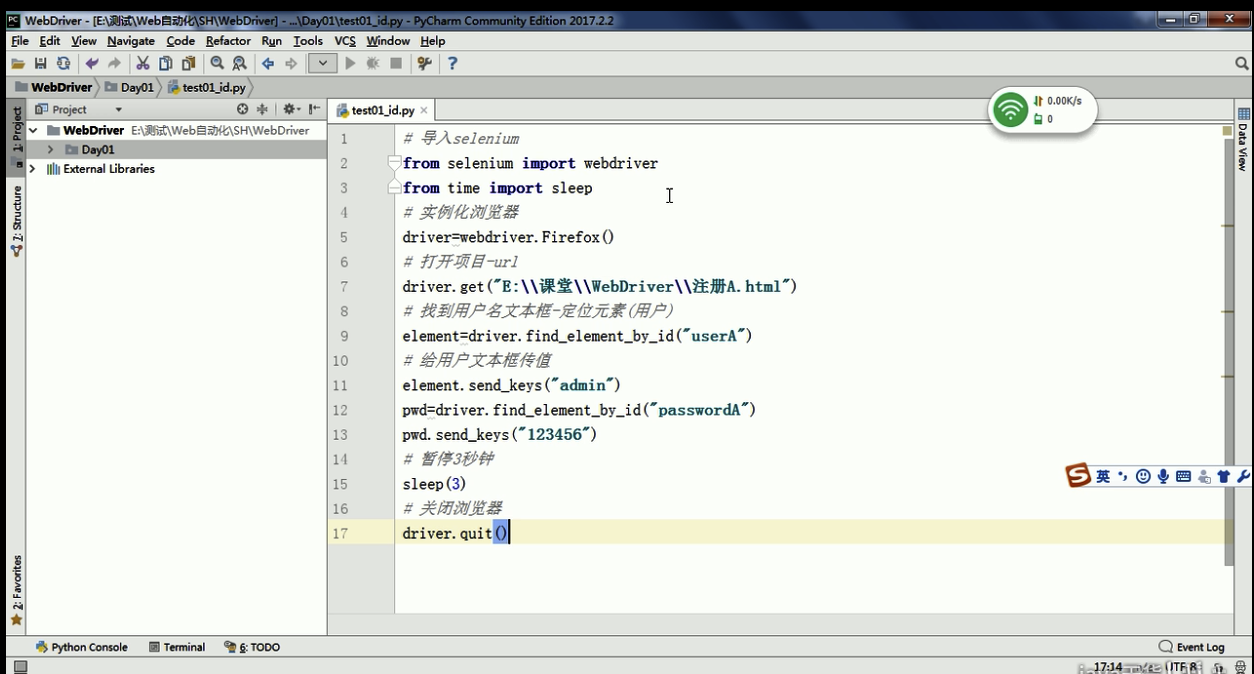
10.4.1 安装selenium：pip install selenium==2.48.0（版本号，不写是最新版本）

10.4.2 卸载selenium ：pip uninstall selenium

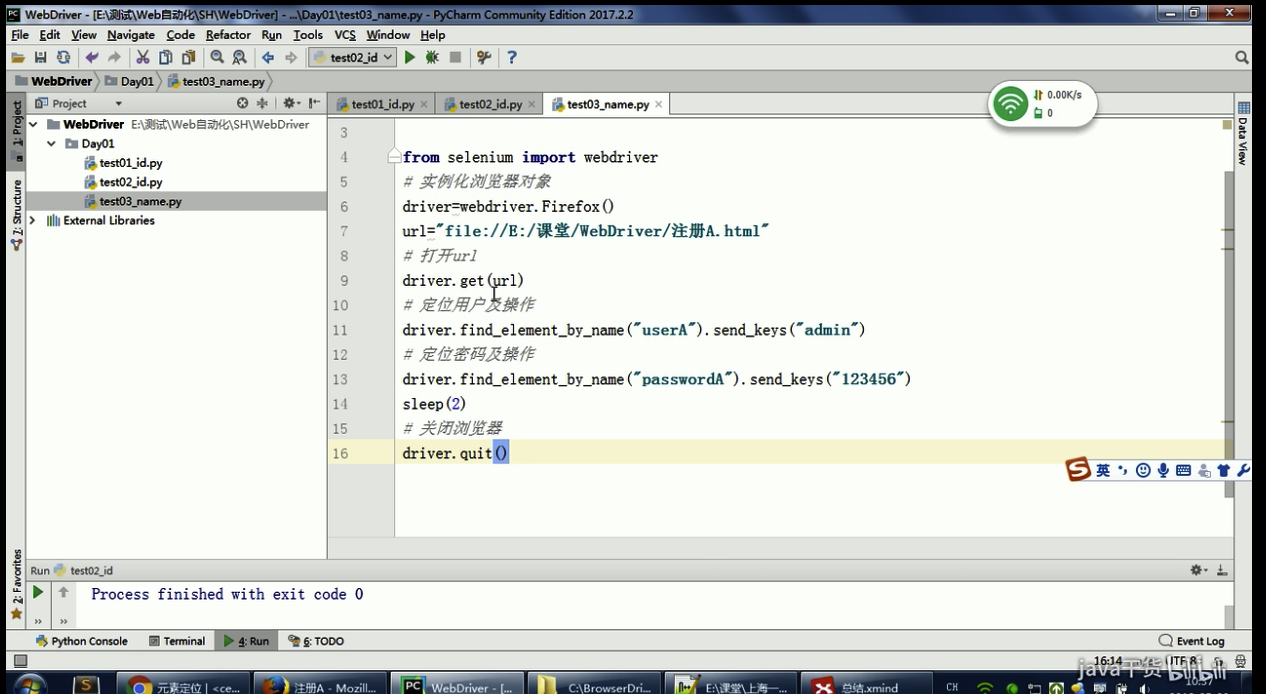
10.4.3 查看：pip show selenium

10.4.4 元素定位

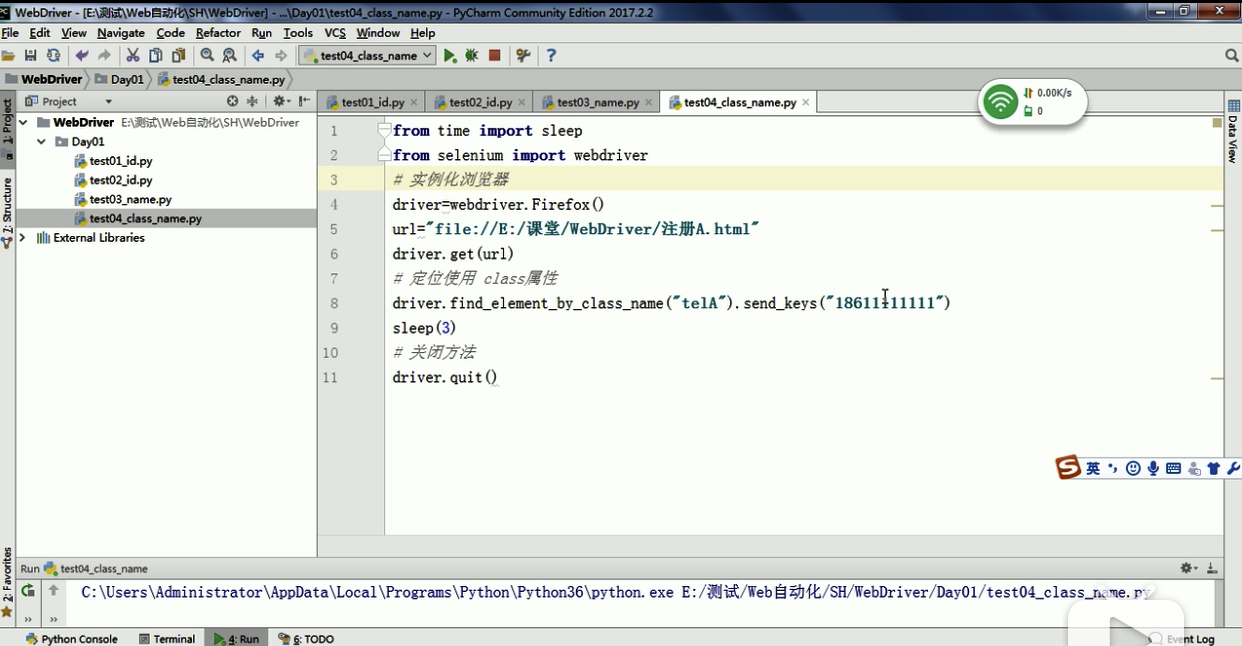
10.4.4.1 id（元素属性）：导包----url中的转义（1、r修饰：url=r“e：\...”；2、转义：url=“e：\\..\\..”;复制浏览器上地址，乱码没关系能识别）----id定位----发送内容----暂停方法（sleep（））----关闭浏览器（quit（））



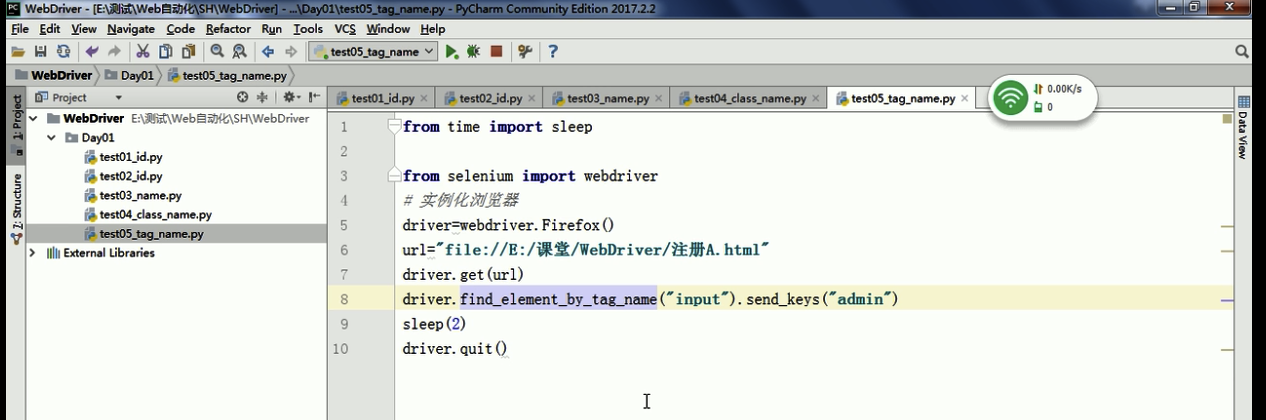
10.4.4.2 name（元素属性）：定位方法是find\_element\_by\_name(),其他 与id定位 步骤一致。



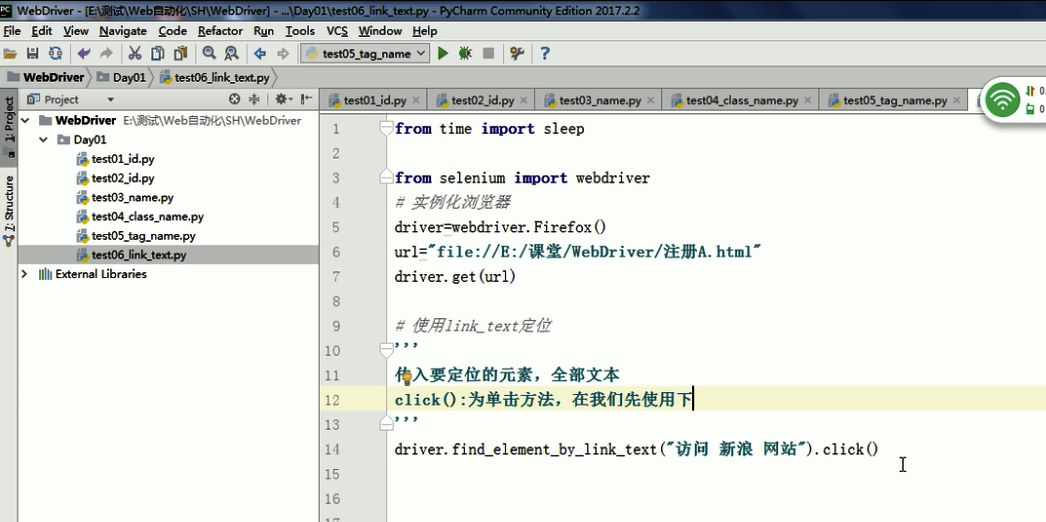
10.4.4.3 class\_name（元素属性）：定位方法是find\_element\_by\_class\_name(),其他 与id定位 步骤一致。

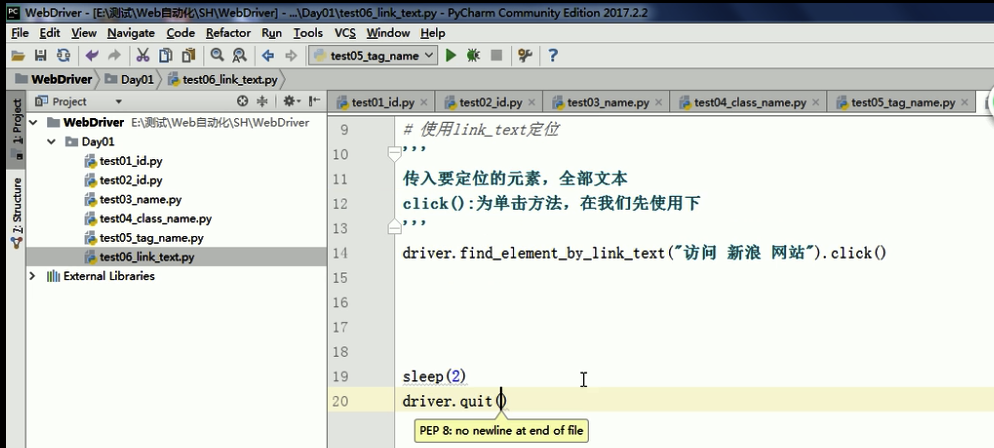


10.4.4.4 tag\_name（标签属性）:定位方法是find\_element\_by\_tag\_name(),返回符合条件的第一个元素，其他与id定位 步骤一致。

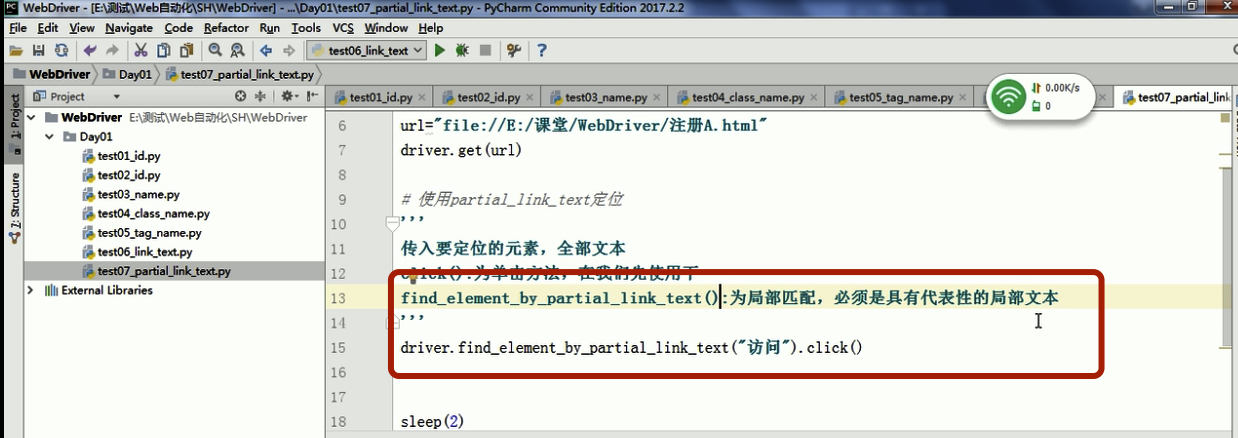


10.4.4.5 link\_text（链接属性）：定位超链接文本，find\_element\_by\_link\_name(传入全部文档，包括空格)





10.4.4.6 partial\_link\_text（链接属性）：为模糊匹配，link-text为全部匹配，传入链接标签局部文本但是要能唯一表达，find\_element\_by\_partial\_link\_name（）。



\*find\_elements\_by\_XXX()【下标，从0开始】,加了S，要加下标选中选择哪一个，要不会返回所有符合要求的元素。

10.4.4.7 Xpath（路径定位XML+path）：find\_element\_by\_Xpath()

1. 路径定位：绝对路径（可用firebug快速定位，以单斜杠开头，中间不能跳跃元素）、相对路径（可用firebug的firepath快速获得，双斜杠开头，后边必须跟标签名称或\*）
2. 元素定位：例如／／\*【@属性如id等】、／／\*【text（）=“XXX”】文本内容是XXX 的、／／\*【starts-with（@attribute，’XXX‘）】以XXX开头的属性、／／\*【contains（@attribute，‘Sxxx’）】属性中包含XXX的元素
3. 层级和属性定位：例如／／\*【@🆔id=“aa”】／input
4. 属性和逻辑结合：例如／／\*【@🆔id=“a” and @🆔id=“b”】

10.4.4.8 css（css选择器），速度快，在css状态搜，find\_element\_by\_css\_selector()

1. ID选择器：#id
2. class选择器
3. 元素选择器
4. 属性选择器
5. 层级选择器

10.4.4.9 by：find\_element(By.ID,”loginName”)、find\_element(By.NAME,”SubjectName”)、find\_element(By.CLASS\_NAME,”u-btn-levred”)、find\_element(By.TAG\_NAME,”input”)、

find\_element(By.LINK\_TEXT,”退出”)、

find\_element(By.PARTIAL\_LINK\_TEXT,”退”)、

find\_element(By.XPATH,”.//\*[@id=’Title”)、

find\_element(By.CSS\_SELECTOR,”[type=submit]”)

# 11、移动端测试

11.1 移动端测试分类

11.1.1 app功能测试：

11.1.1.1 业务逻辑正确性测试：产品文档设计测试用例和隐形需求测试；

11.1.1.2 兼容性测试：系统版本、分辨率、网络情况；

11.1.1.3 异常测试：热启动应用（前后台切换）、网络切换和中断恢复、电话和信息中断恢复；

11.1.1.4 升级与安装测试；

11.1.1.5 健壮性测试：手机资源消耗（cpu、内存等）、流量消耗、崩溃恢复等测试；

11.1.2 app自动化测试

11.1.3 app安全测试

11.2 appium(支持java、c#、python、PHP、perl、ruby、node.js，支持安卓和iOS系统，支持跨应用)

11.2.1 安装appium

11.2.1.1 客户端安装：官网下载安装包，打开软件，填写本机ip至host，点击进入qppium。安装appium-python，再安装命令行,下载node.js，安装完后检查有没有安装好，中断输入npm，再安装cnpm，在安装appium。

11.2.1.2 安装apk到手机：

driver.install\_app（app.path）

11.2.1.3 卸载app：driver.remove\_app(app.path)

11.2.1.4 判断app是否存在手机中 ：driver.is\_app\_install(app包名)

11.2.1.5 发送文件到手机：

import base64

Deta=str(base64.b64encode(data.encode(‘utf-8’)),’utf-8’)

Driver.push\_file(手机上设备的路径 ，文件内数据)

11.2.1.6 从手机中拉取文件

import base64

Data=driver.pull\_file(设备手机上路径)

Print(str(base64.b64decode(data),’utf-8’))

11.2.1.7 获取 当前屏幕内元素结构

Driver.page\_source

# 12、接口测试

12.1 接口：系统与外部其他系统之间的接口，以及系统内部各个子模块之间的接口。

12.2 接口测试：测试的重点是要检查数据的交换，传递和控制管理过程，以及系统间的相互逻辑依赖关系等。测后段实线是否符合接口规范。测程序是否安全是否效率，测法是模拟客户想服务器发数据查看响应，首先定位服务器资源，再提交数据，再查看响应。安装并运行程序，分析接口文档，逐一测试接口。

12.3 接口测试分类

12.3.1 web接口测试（bs架构）

服务器接口测试：测试自己公司写的程序 ；

第三方接口测试：测试别的公司实现的接口；

12.3.2 模块接口测试（cs架构）

12.4 环境搭建

12.4.1 运维或开发实现程序部署正常运行

12.4.2 自己搭建

12.4.2.1 安装python

12.4.2.2 安装依赖模块：pip install -rrequirements.txt -s <https://pypi>.douban.com/simple

12.4.2.3 启动项目

12.4.2.4 测试

12.5 插件：浏览器的接口测试工具，如火狐的RESTVClient，谷歌的postman，java实现的jmeter，提高测试效率。

12.6 RESTful风格：规范了url，提交数据、响应数据的格式。程序易读和易维护。是约束增删改查和上面三要素的。

12.7 JOSN：是一种数据载体，和html、xml类似，不过要简单，但因为太简单所以分辨度差。

josn的格式是{“title”:”hello”,”font”:”hello world”}标题和内容。数组【值1，值2，值3......】

# 13、jmeter（用java开发的测试工具）

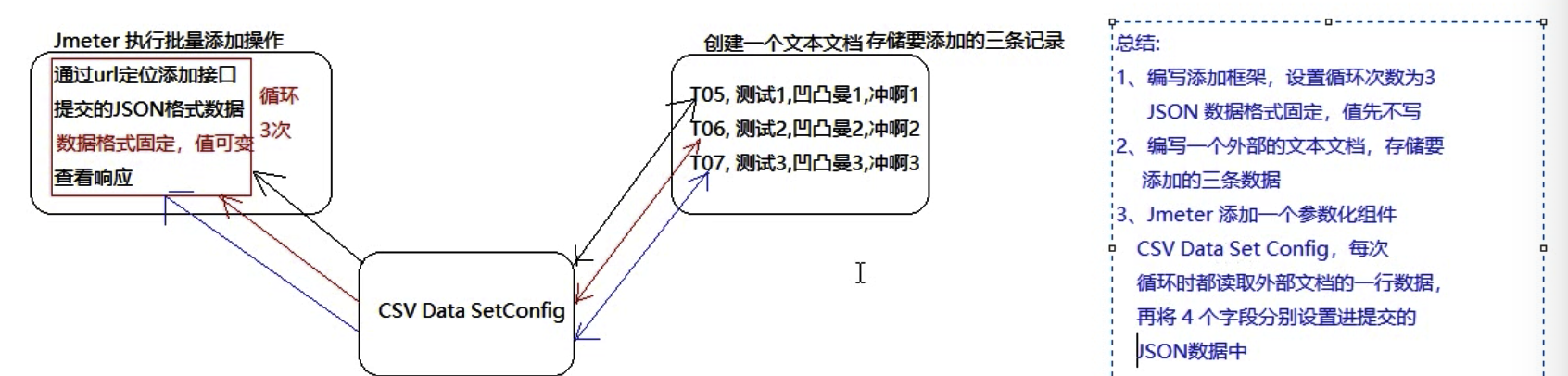
13.1 优点：安全，高效，功能强大，可模拟多次并发和循环操作，开源免费，支持多协议。可实现接口，性能，压力，web自动化，数据库，java程序的测试。

缺点：不支持ip欺骗，不支持前端测试。

13.2 用法：下载,先下java环境，再下jmeter，了解jmeter目录结构，启动，编写测试案例。

13.3 jmeter线程组：按线程性质对线程分组，方便管理，默认并发执行，可通过选择实现顺序执行。

13.4 jmeter参数化：动态获取并设置数据，执行批量操作时，之前数据都是一个个手写，效率太低，参数化可以自动批量添加。用CSV Date SetConfig；



13.5 高级线程组：setup最先执行，tear dowm最后执行。

# 14、性能测试

14.1 性能测试：基于协议模拟用户发出请求 ，对服务器形成一定负载，来测试服务器的性能指标，主要是时间性能和空间性能，与界面无关。

14.2 性能测试分类：

14.2.1 狭义性能测试（验证性能是否能满足宣称的性能要求）；

14.2.2 负载测试（不断加压，指导找到极限点）；

14.2.3 压力测试（饱和状态下，系统的处理能力）；

14.2.4 并发测试（同时并发一个应用看是否死锁或有其他问题）；

14.2.5 配置测试（对软硬件环境不断调整看系统状态）；

14.2.6 可靠性／稳定性测试（系统持续运行会不会出错，一般是72小时的测试）；

14.3 性能测试的指标：响应时间（网络传输+服务器执行时间，不包括前端页面渲染时间），tps（每秒处理的事物数），资源利用（cpu（不允许超过7、8成，队列长度等）、mem利用率（内存，8成以下，页交换律）、带宽利用率），用户数（并发用户数，并发请求数）

14.4 性能测试流程：

14.4.1 需求分析（测试对象（常用的，核心的）、性能指标（吞吐量、响应时间、用户数、资源利用率）、测试场景（单一场景（登陆、注册、搜索等）、混合场景（用户使用场景、系统使用场景）））

14.4.2 测试计划和方案：计划有测试目标，人员，进度安排，压力机，风险；方案有测试工具（roadrunner、jmeter）、测试环境、测试策略（单一场景、混合场景）、监控工具Linux（nmon、rpc、jvisuaIVm、spotlight）、windows（spotlight、perfmon.exe）；

14.4.3 用例设计：测试脚本、场景测试

14.4.4 性能测试执行：脚本编写--场景监控设计--运行场景--监控场景--测试报告--定位分析问题（后段（代码、软件、硬件）、前端、网络）

14.4.5 roadrunner：支持Ie9以下版本和一些windows版本。要钱，可用破解版。三大组件（VuGen（生成脚本 ）、controller、analysis）

# 15、QC

15.1 QC介绍：一种缺陷管理工具