# 基础篇

## 环境搭建

### 安装Python

由于Robot Framework 框架是基于Python2 开发，所以这里我们选择安装Python2。；对于Windows 用户来说，如果你的系统是32 位的请选择x86 版本，如果是64 位系统

请选择64 版本进行下载。下载完成会得到一个以.msi 为后缀名的文件，

### 安装steuptools 与pip

setuptools是一组Python 的distutilsde工具的增强工具，可以让程序员更方便的创建和发布Python 包，特别是那些对其它包具有依赖性的状况。用于支持安装各种测试库的工具，它会把安装好的测试库的信息放到一个公共的文件里（easy-install.pth），可以看做是一个测试库的引导清单。

pip 是一个安装和管理Python 包的工具，pip 的安装依赖于setuptools ，所以在安装pip 之间需要先安装setuptools。

Python2.7.10版本自带安装pip和setuptools，如果是低于Python2.7.10的版本，需单独安装pip和steuptools：

通过解压工具进行解压将得到两个文件夹，在Windows 命令提示符进入到文件解压目录，通过Python执行安装文件setup.py 进行安装。

C:\package\setuptools-7.0>python setup.py install

C:\package\pip-1.5.6>python setup.py install

### 安装robot framework

下载地址:https://pypi.python.org/pypi/robotframework/2.8.7

可以通过下载exe 程序进行安装，Robot Framework 分别提供了，win-amd64.exe 和win32.exe 两个

windows 版本，你可以根据自己的环境下载相应的版本，双击进行安装。

如果像安装普通的Python 程序，可以下载tar.gz 文件，解压并运行setup.py 文件进行安装。

C:\robot\robotframework-2.8.7>python setup.py install

因为在上一小节中我们已经安装了pip，所以通过pip 命令安装更为方便和快捷：

C:\Python27\Lib\site-packages>pip install robotframwork

### 安装wxPython

下载地址:http://www.wxpython.org/download.php

wxPython 是Python 非常有名的一个GUI 库，因为RIDE 是基于这个库开发的，所以这个必须安装。在官网上找到相应的版本下载，为exe 可执行文件

必须安装wxPython2.8,因为RIDE 是基于这个版本编译的.

http://sourceforge.net/projects/wxpython/files/wxPython/2.8.12.1/

### 安装ride

下载地址:https://pypi.python.org/pypi/robotframework-ride

RIDE 是Robot Framework 测试数据的编辑器。它使测试用例的创建、运行、测试项目的组织可以在图形界面下完成。

RIDE 同样提供了win-amd64.exe 和win32.exe 两个windows 版本和一个tar.gz 文件，前者下载双击进行安装。后者解压并执行setup.py 文件。

C:\robot\robotframework-ride-1.3>python setup.py install

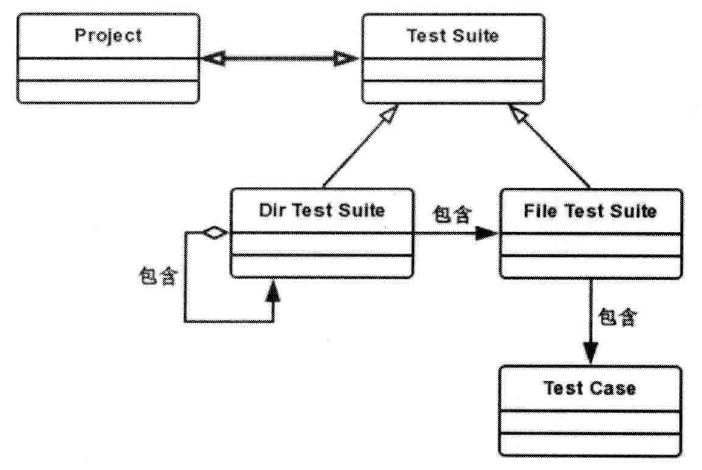
安装完成后在C:\Python27\Scripts目录下会生成ride.py文件，使用Python执行该文件既可启动ride界面。

# ride工具

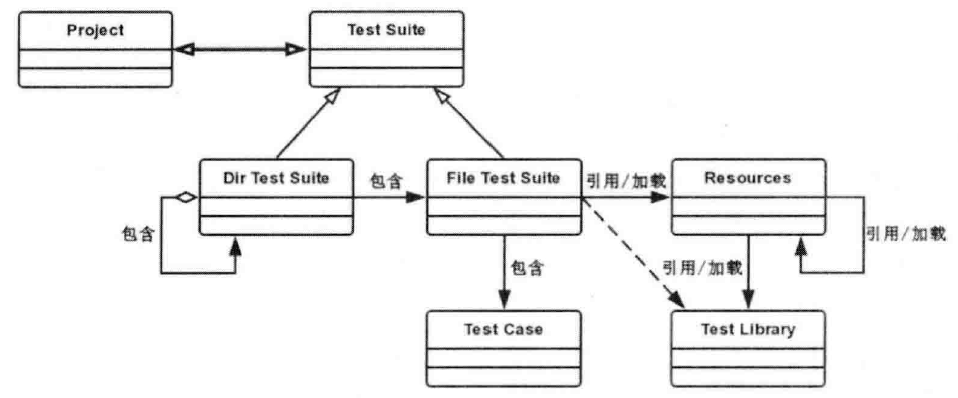
## 工程、套件和用例

### 三者之间的关系

一般情况下，我们会选择将测试项目定义成一个目录，这样它就可以分成多个套件，套件可定义为不同的业务，不同的业务下再分用例，结构会更加清晰。也可以把“测试套件”创建成一个目录，这样就不能直接在其下面创建用例了，还需要再创建的“File”类型的“子测试套件”。说白了就是用例只能创建在file 类型的套件中。



完整的工程结构关系图



注：在Test Suite去引用或加载Test Library也可以，但是不推荐，所以使用的虚线。

### 新建工程

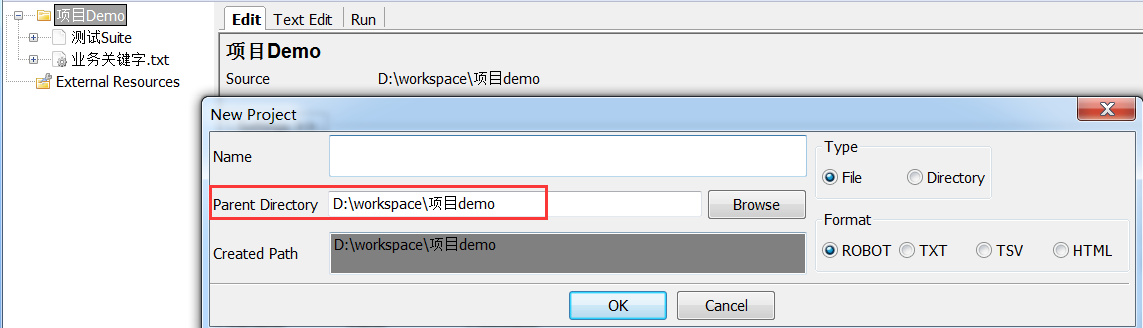
1.Parent Dictionary：上级目录

注：创建工程时，需看清楚Parent Dictionary上级目录，默认是上一次工程的目录。

2.Create Dictionary：创建工程后的完整路径

3.Type：工程类型，分为文件类型和目录类型，如果项目内容比较简单，选择file类型；如果项目内容比较复杂，为了便于管理，可以选择目录类型。

4.Format：文件格式，默认是ROBOT格式，也可以考虑TXT格式，便于维护和版本管理。（目前默认ROBOT格式）



### 工程右键菜单

Change Format：之前设定了工程或测试套件的格式，这里可以进行修改。

### 新建测试套件

同新建工程

### 测试套件右键菜单

Sort Keywords会对当前的Suite下面的关键字进行排序

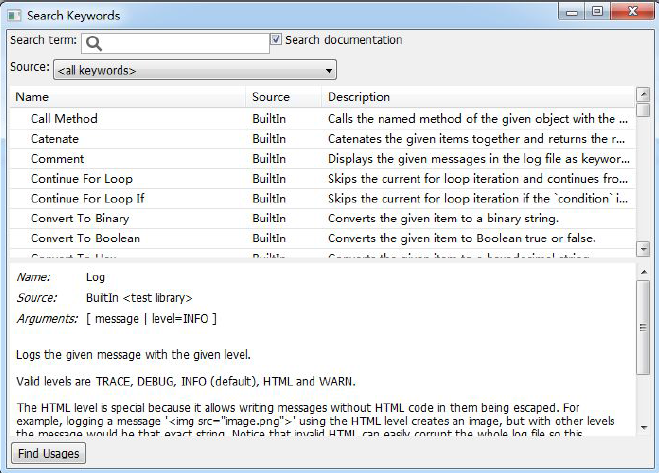
Search Keywords查询关键字，快捷键F5

### 新建用例

### 用例右键菜单

### 查询关键字

RIDE 提供了F5 快捷键来打开帮助文档



search term: 用于搜索关键字。

source：用于选择相关库，默认在所有库下搜索关键字。

创建分上下两部分，上部分显关键字列表，下半部分显示某一关键字的详细说明。

### 常用快捷键

重命名——》F2

搜索关键字——》F5

执行用例——》F8

创建新工程——》ctrl+n

创建新测试套——》ctrl+shift+f

创建新用例——》ctrl+shift+t

创建新关键字——》ctrl+shift+k

向上移动用例——》ctrl+↑

向下移动用例——》ctrl+↓

显示关键字信息——》 ctrl+鼠标悬浮（鼠标悬浮于关键字上）

自动补全关键字——》ctrl+shift+空格

删除行——》ctrl+d

删除单元格——》ctrl+shift+d

插入单元格——》ctrl+shift+i

插入行——》ctrl+i

屏蔽代码——》ctrl+#

取消屏蔽——》ctrl+$

保存整个工程——》ctrl+shit+s

局部保存，保存鼠标点击的部分——》ctrl+s

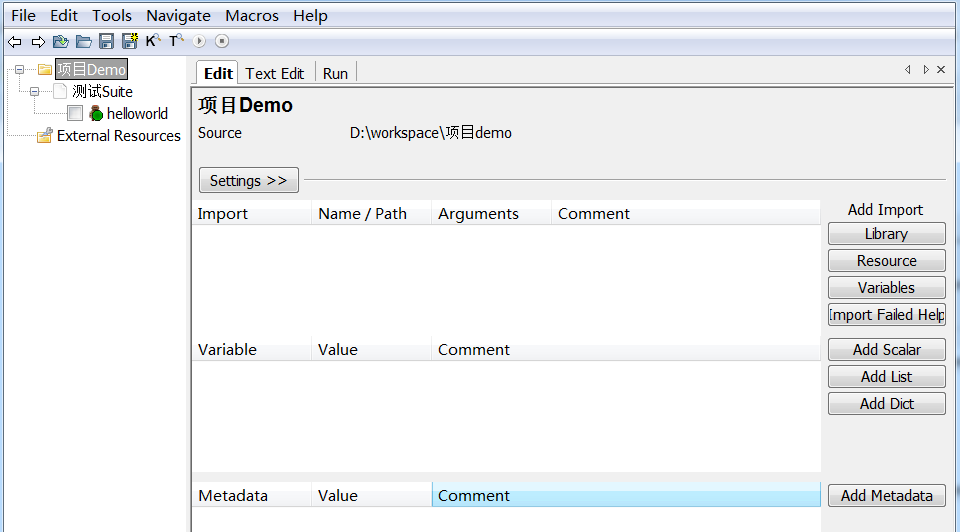
查看log——》ctrl+L

查看report——》ctrl+r

注意：如果快捷键不能使用，先看看是否有其他软件已占用相应的快捷键

## Edit标签

从“测试项目”和“测试套件”所提供的Edit 标签的功能也可看出两者是一样的。



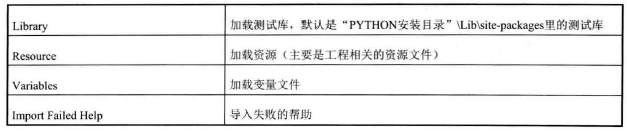
在Edit 标签页中主要分：加载外部文件、定义内部变量、定义元数据等三个部分。

（1）：加载外部文件

Add Library：加载测试库，主要是[PYTHON 目录]\Lib\site-packages 里的测试库

Add Resource：加载资源，主要是你工程相关的资源文件

Add Variables：加载变量文件。



（2）：定义内部变量

Add Scalar：定义变量

Add List：定义列表

Add Dict：定义字典

（3）：定义元数据

Add Metadata：定义元数据。大概看了一下作用是在report和log 里显示定义好的内容，格式和document一样。

### 导入库

点击Edit 标签页右侧的“Library”按钮，来添加库。如果是添加扩展库，在添加之前，需进行手动安装，例如在添加Selenium2Library库之前，需要在命令行使用pip命令进行安装：

pip install robotframework-selenium2library

注意：

1.如果添加的库不存在或库名错误，将会红色显示，黑库正常表示正常。

2.如果在“测试套件”中添加库，那么这个库中所提供的关键字可以被当前测试套件下的用例使用；如果在“测试项目”中添加库，当前项目下的测试用例不能使用库中的关键字，需要在用例所在的“测试套件”中再次添加库。

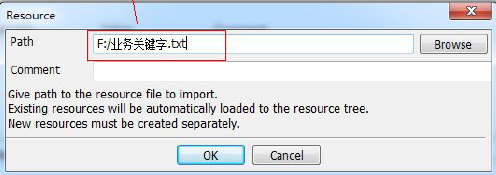
3.按F5，可查看已添加库中所提供的关键字。

4. 可以导入robot framework内置的库目录中库，例如： C:\Python27\Lib\site-packages\robotframework-3.0-py2.7.egg\robot\libraries\ Collections.py

### 导入资源

点击Edit 标签页右侧的“Resource”按钮来添加资源。这个资源一般为项目关的文件。比如，项目的自定义关键字文件。

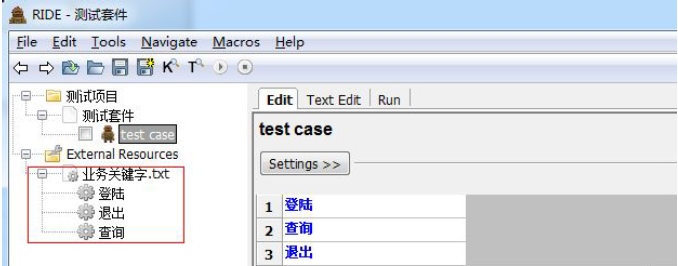
添加一个“业务关键字.txt”文件：



注：Path里的路径分隔符"\"需要写成“\\”，因为"\"会被认为是转义字符，如果嫌麻烦，可以用“/”来分隔。

关于添加资源的作用域与库一样。我这里是添加到的测试套件中，那么它的作用域就是当前测试套件下的所有用例。

查看RIDE 左侧项目列表，会发现“External Resource”下多了一下“业务关键字.txt”的资源。展开关键字会看到文件中定义的登录、退出和查询三个关键字。现在就可以在用例中使用这些业关键字了。



### 定义变量

点击Edit 标签页右侧的“Add Scalar”按钮来创建变量。这里创建的变量可以被整个测试套件中的用例所使用。也可以认为是一个“公共变量”。

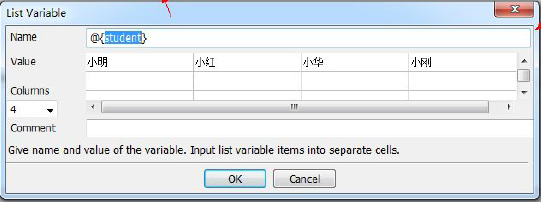


Name 用于定义变量名：${hi}

Value 用于给变量赋值

### 定义列表变量

点击Edit 标签页右侧的“Add List”按钮来创建列表变量，列表变量可以用来定义一维或二维数组。



Name 定义变量名为：${student}

Value 填写列表变量的值

在测试用例中可以对这个一维数组进行遍历。



### 测试用例的Edit

1）Settings设置：

参见ride工具中的Settings选项章节

2）步骤表格：

调用有返回值的无参方法：表格列依次是：变量、方法名

调用有返回值的有参方法：表格列依次是：变量、方法名、参数列表

调用无返回值的无参方法：表格列依次是：方法名

调用无返回值的有参方法：表格列依次是：方法名、参数列表

变量定义格式：${变量名}，以绿色字体显示

系统关键字和用户关键字：以蓝色字体显示

不合法数据：以红色字体显示

必填参数：表格以红色背景显示

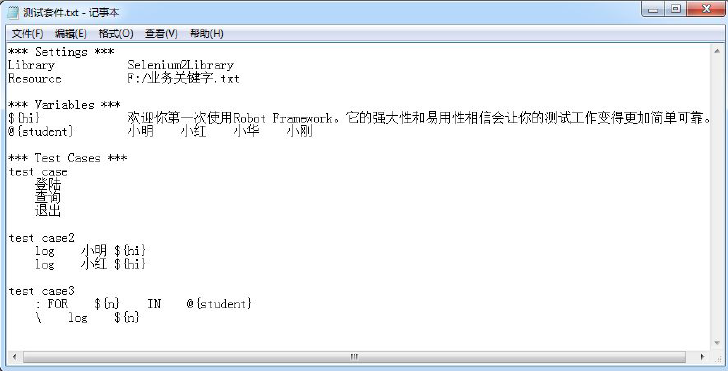
注：

表格中如果需要使用其他类型数据，可以使用Evaluate关键字，调用Python中相应方法，例如调用int()方法。

## Text Edit标签

在Edit 标签页完成的工作，都可以在Text Edit 标签页上完成。它们之间是对应关系，Edit 可视化的提供的按钮输入框，对于用户来说更容易知道我要怎么做。

切换到“...\robot\测试项目”目录下，通过记事本打开“测试套件.txt”，这和在Text Edit中看到的信息一致。测试用例的本来面目也是这个样子的，只是在RIDE 中对它进行的“美化”。

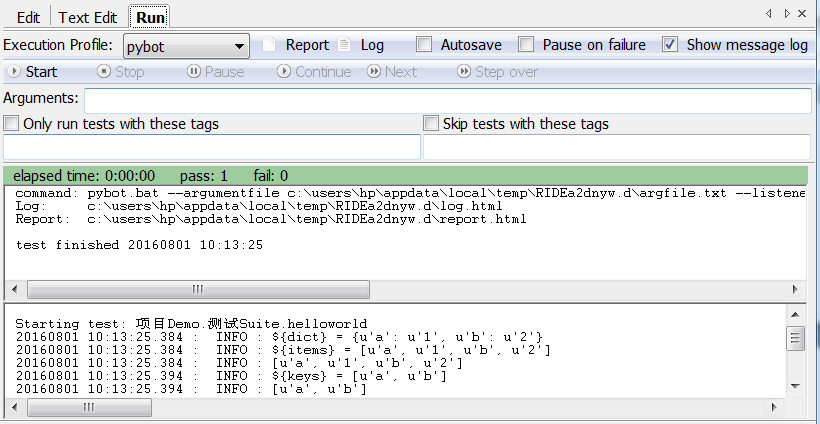


其实在这个在Text Edit 下或第三方编辑器下编写Robot Framework 测试的效率要远远高于Edit 标签中的“填表格”式编写。读者可以在两种标签页之间切换来提高用例的开发效率。

## Run标签

在RIDE 中运行测试用例，勾选想要运行的用例，然后点击按钮即可。想想我们在做单元测试的时候可不会这么方便，调用addTest()方法将一个个想要运行的测试方法添加到测试套件中，或者一行行的注释掉不添加到测试套件的测试用例的addTest()方法，这是个极其痛苦的过程。

### Run界面介绍



1)Execution Profile：选择运行方式，里面有pybot、jybot 和custom script。其中我们默认是用pybot来运行用例，pybot 的运行Python 编译器完成。jybot 需要安装Jython 的支持。custom script 是选择自定义的脚本来运行。

2) Start 和Stop：用例的运行和停止。

3) Report 和Log：报告和日志，要运行之后才能点击。

4) Autosave：自动保存，如果不勾选，在修改了用例之后如果没有保存的话，运行用例时会提示是否保存。勾选则在运行时自动保存了。

5) Arguments：pybot 的参数（或者jybot 等），可以在这里输入pybot 的命令完成相应的操作。

6) Only Run Tests with these Tags：只运行这些标记的测试用例。

7) Skip Tests with these Tags： 跳过这些标记的测试用例。

中间区域记录用例的执行过程，底部的区域输出用例的执行结果。

### 报告与日志

当用例运行结束，Robot Framework 生成三个文件：output.xml、log.html 和report.html。

output.xml 记录的测试结果是xml 文件，这个文件不够直观。根据特定的需要可以编写脚本读取xml文件并生成特定的测试报告。

相比较而言log.html 和report.html 报告要直观得多，因为是html 格式。

log.html 和report.html：日志更多是过程的记录，在测试用例调试的过程中更多使用日志来查看执行错误。报告更多是结果上的展示，当只想知道测试用例的最终执行情况时用报告。

## 筛选执行用例

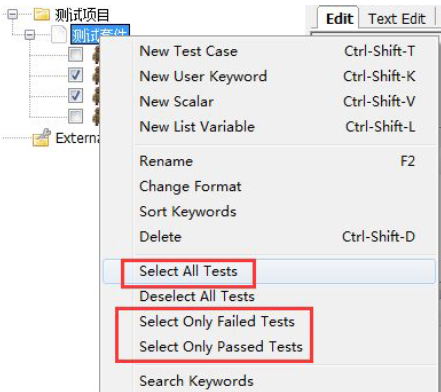
### 勾选用例

要运行哪条用例就勾选哪一条。如果全部不勾选，点击“运行”按钮会运行

所有用例。



也可以在“测试套件”上右键选择：



Select All Test：选择当前套件的所有用例。

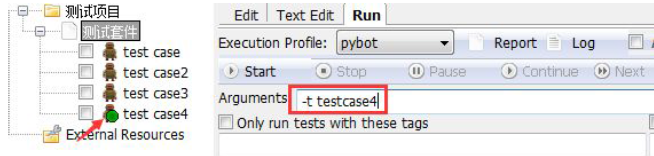
Select Only Failed Test：选择当前套件下运行失败的用例。

Select Only Passed Test：选择当前套件下运行成功的用例。

### 使用命令

使用Run 标签中的Arguments 功能。

在Arguments 的输入框内输入“-t testcase4”。点击“Start”按钮，只执行了test case4 这一条用例。



Arguments 能做的事情可不止于此，想了解更多多命令，可以在cmd.exe 下执行“pybot.bat --help”。

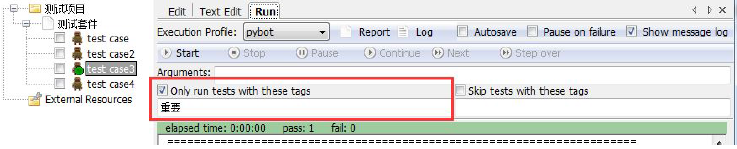
### 筛选标记

先在test case的Settings选项中，在Tags一栏输入标签，例如“重要”，这条用例就打上了“重要”的标记。

然后切换到Run 标签：

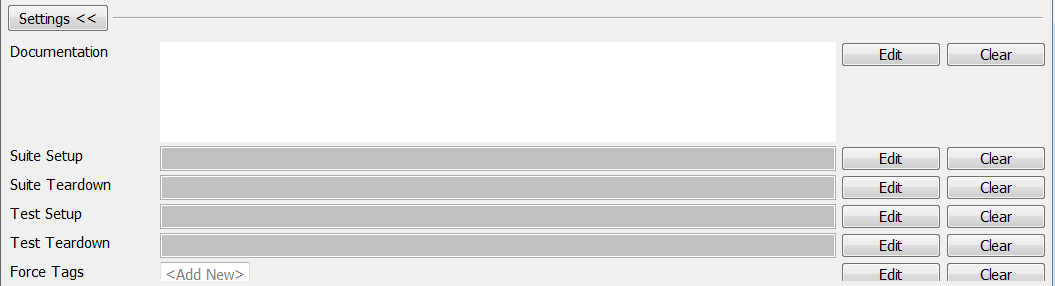
Only Run Tests with these Tags：只运行这些标记的测试用例。

Skip Tests with these Tags： 跳过这些标记的测试用例。



## Settings选项

### 工程的Settings设置



1）Documentation：用于描述当前工程的说明

可以加粗：\*描述内容\*

可以斜体：\_描述内容\_

可以把URL 地址转换为可点击的链接。



2）Suite setUp：在每一个测试套件前先被执行，用于设置初始化工作。

3）Suite tearDown：在每一个测试套件执行后调用，用于完成清理工作。

4）Test setUp：在每一个测试用例前先被执行，用于设置初始化工作。

5）Test tearDown：在每一个测试用例执行后调用，用于完成清理工作。

执行顺序：

套件1初始化

用例1初始化

用例1执行

用例1清理

用例2初始化

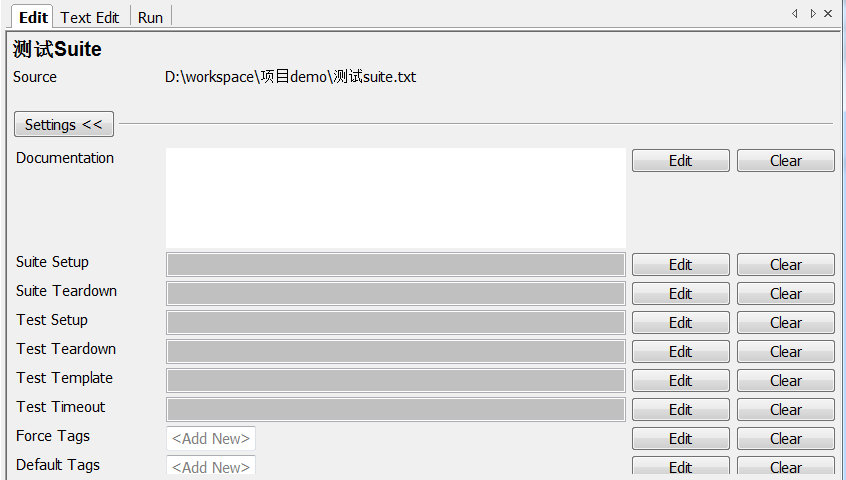
用例2执行

用例2清理

套件1清理

6）Force Tags：强制标记，给当前测试套件下的每个测试用例都打上Tags，而且不能在用例上删除这个Tags，只能在设置的地方删除。

### 测试套件的Settings设置



Force Tags：表示当前测试套件下测试用例强制的标记。

Default Tags：表示当前测试套件下测试用例默认的标记。

Test Template：指定模板使用的关键字。（不推荐使用）

TimeOut：用于设置用例的超时时间，可以输入如“1 min 10s”、“ 2 hours”等，也可以直接输入秒数，系统会自动转换为时分秒。

Default Tags ：用于给用例添加标记。

### 用例的Settings设置

## 使用用户关键字

在Robot Framework 中关键字的创建分两种：系统关键字和用户关键字。系统关键字需要通过脚本开发相应的类和方法。用户关键字的创建就要简单得多，它主要利用现有的系统关键字，根据不同的业务，把多个重复的步骤集合在一起组成用户关键字（相当于自定义函数）。

例如将for循环封装成一个用户关键字：

### 创建用户关键字

右键点击“测试项目”或“测试套件”都可以看到“New User Keyword”的选项。我们可以根据自己的需求选择在项目或套件下创建用户关键字。

在弹出的菜单中输入用户关键字的名称。查看左侧项目列表，用户关键字已经创建完成。

注：如果用户关键字和测试用例在同级目录，那么在测试用例中可以直接使用该用户关键字。

### 实现用户关键字

1）设置参数

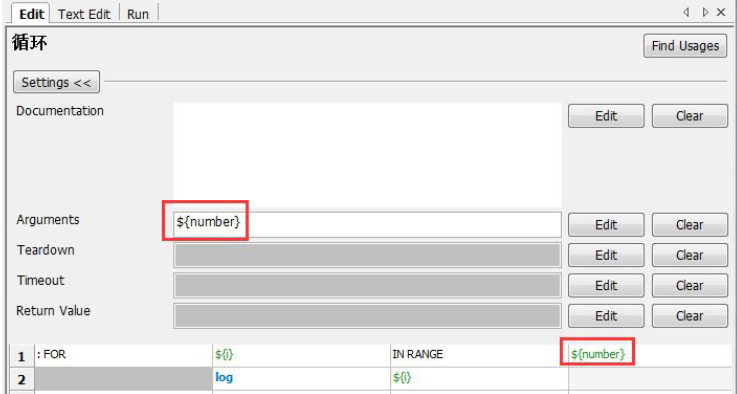
点击该关键字，点击“setting>>”按钮，Arguments 参数为：${number} ，这就像定函数的输入参。可以设置多个变量，之间用“|”分隔。

2）实现步骤

添加循环的用例，循环的次数为：${number}



添加完成后如下显示：



### 使用用户关键字

在测试用例中输入用户关键字，然后跟上参数，既可调用该用户关键字。



## 使用资源

### 系统资源

在实际的项目测试过程中，我们往往需要创建许多这样的具有通用性的用户关键字。不希望这些关键字依附于某个测试套件，甚至是某个项目。那么我们可以创建资源文件用于存放这些关键字。

1）创建资源

右键“测试项目”选择“New Resource”创建资源，输入资源名称。

注：在资源的Edit界面，有一个Find Usages按钮，是用来查询关键字都用在什么地方。

2）创建用户关键字

右键该资源名称，选择“New User Keyword” 创建用户关键字，输入关键字名称。

3）实现该用户关键字功能

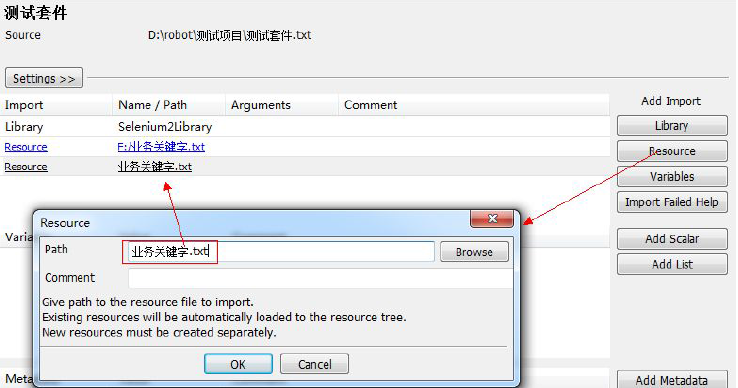
创建完成的项目结构如下：



4）导入资源

因为“业务关键字.txt”和“测试套件”属于并列关系。测试套件要想使用业务关键字下的“循环”关键字，需要导入资源。

注：在Path文本框中需输入资源的完整文件名（带扩展名）



现在就可以在测试用例中使用“循环”关键字了。



### 外部资源（External Resource）

外部资源是指不在Project管辖范围内的资源文件，如果是目录类型的Project，只要不在Project目录内的资源文件都算外部资源。如果是文件类型的Project，它自己无法创建资源文件，其他的资源文件都算它的外部资源，即使和Project文件是平级的。

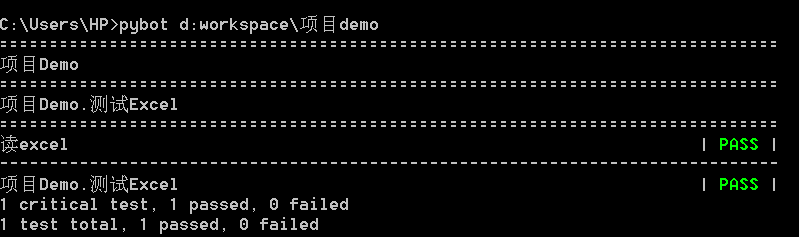
# 持续集成

## 命令行运行

使用pybot命令，命令参数可参见pybot - -help

### 执行整个项目

pybot+项目路径



### 执行测试套件

pybot+测试集的路径



### 执行测试用例

pybot - - test [测试用例](javascript:;)名称 测试用例所在的测试集的路径



## 使用Jenkins

# 基础语法

## 常量

包括环境变量、数值常量、特殊字符常量和系统保留常量。

其中环境变量的标识符为“%”，其他的标识符与Scalar变量相同，都为“$”，所以单值变量要根据它的变量名和变量值，才能确定是什么变量。

### 环境变量

对于环境变量，只能去定义它的地方去修改它的值，在robot framework中只是使用，不能赋值，虽然叫变量，实际上属于常量。

例如：打印Windows中设置的环境变量



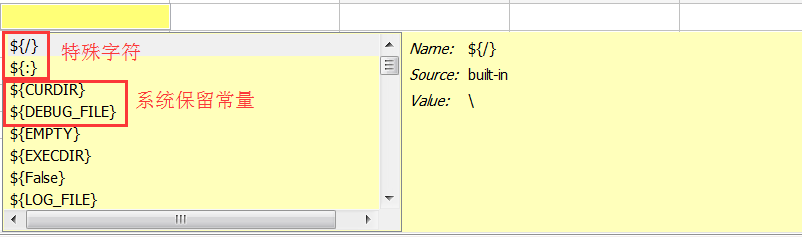
### 数值常量

在ride中默认将所有字符都当做字符串处理，即使变量的值是数值，默认也是字符串的形式存在。如果想要让它直接以数值的方式存在，就可以使用数值常量。

例如数值2.6，可以写成${2.6}；如果直接写2.6，打印出来就是u‘2.6’（Unicode字符串）

### 特殊字符和系统保留常量

在Case的单元格，使用快捷键Ctrl+Alt+空格：



特殊字符常量主要有${/}、${：}、${EMPTY}、${False}、${None}、${null}、${SPACE}、${Ture}，其他的都是系统保留常量

注：如果右侧的value值为空（除了${ EMPTY }），都是在运行时才有它的值。

## 变量

### 变量标识符

变量标识符{}：每个变量都可用变量标识符{变量名}表示

变量主要有两类：

Scalar变量，标量变量，使用$标识

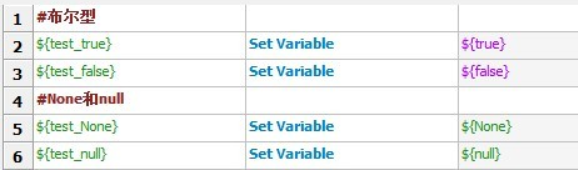
List变量，列表变量，使用@标识

Scalar变量可理解为单值变量；List变量可以理解为多值变量。

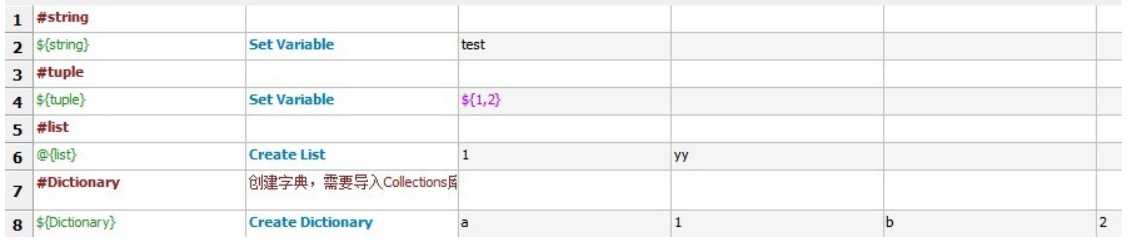
### 变量类型

1）数字变量

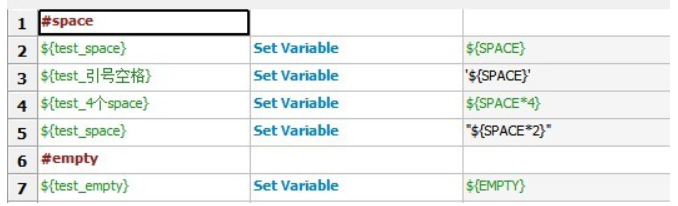
2）布尔变量和None/null



3）字符串、元组、list和字典

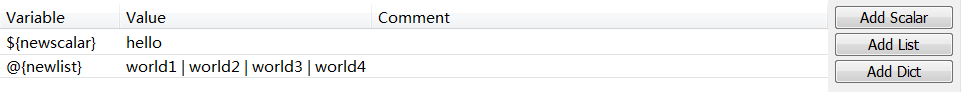


4）space和empty



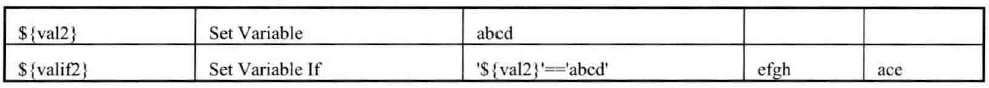
### 变量赋值

方法一：通过右键菜单Add Scalar或Add List，或者在Edit界面Add Scalar或Add List



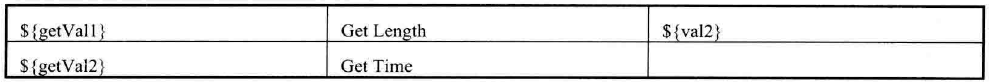
方法二：使用Set Variable关键字添加Scalar变量和List变量。





方法三：使用Get赋值

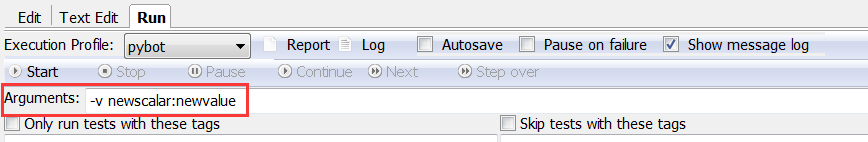
使用返回值进行赋值，包括系统关键字的返回值和用户关键字的返回值。



方法四：命令行赋值

在运行时使用“-v”参数来给变量赋值





可以看到，newScalar的值变成了newvalue

### 变量的作用域

每个变量的作用域默认都是局部变量，默认情况下：

1)在case中添加的变量，作用域在该case内部；

2)在文件类型的suite中添加的变量，作用域在该suite内部，该suite下面的case也都可以使用；

3)在目录类型的suite中添加的变量，作用域在该目录内，但也只是目录内，它下面的文件类型的suite是无法使用的，所以一般在目录下新增变量没有太大意义。

4)在userkeyword中添加的变量，作用域在该userkeyword内部；

改变变量作用域的方法（建议少用）：

Set Global Variable：设置为全局变量，在所有测试套件及测试用例都有效；

Set Suite Variable：设置为File Suite级变量，在当前的File Suite内都有效，包括下面的case；

Set Test Variable：设置为case级变量，在当前的case内有效；

### 变量的使用

1）在判断中使用



条件成立，执行log关键字；条件不成立，不做任何操作。

2）在字符串中使用

字符串连接：

如果想把变量作为一个字符串的一部分，可以直接这么写：

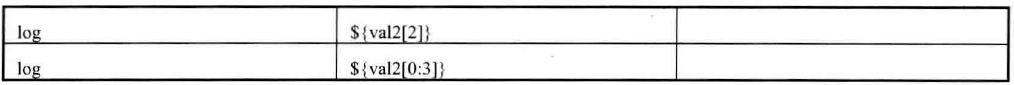


在默认情况下，robot framework的变量是字符型的，并且连接两个字符串不需要加什么符号，直接连起来就行了。如果写成abc+123，最终结果就是abc+123，它会把你写的任何内容都当做字符串。

字符串截取：

${变量名[a]}：截取第a个字符，序号是从0开始。

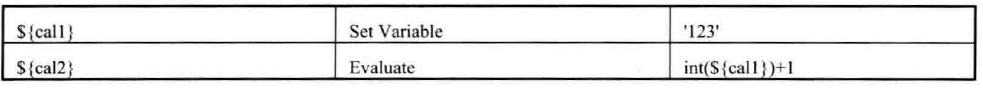
${变量名[a：b]}：实际上是[a:b-1]区间，序号是从0开始。



3）参与运算

使用Evaluate关键字可以像Python一样进行各种运算。包括调用Python中的库函数等。

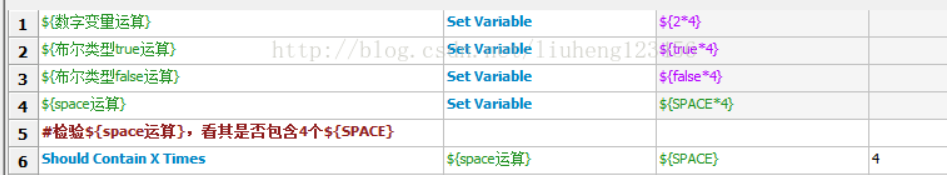




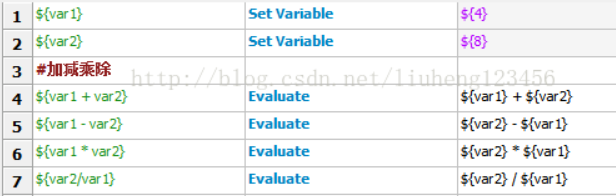
### List变量（88页-149页）

## 常用运算

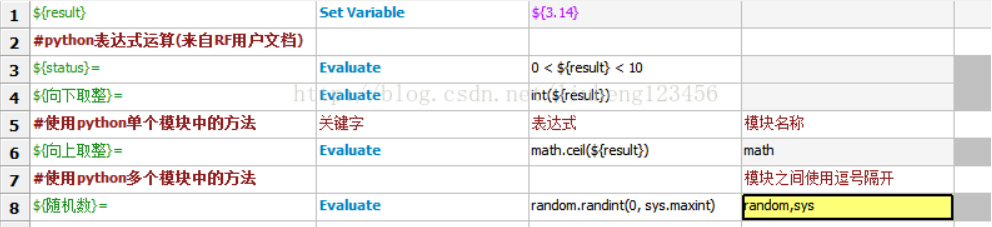
### 特殊变量运算



### 使用Evaluate运算



### Python表达式运算



## 控制语句

### if语句

使用Run Keyword If关键字

函数释义：如果给出的判断条件满足，就执行给出的关键字。

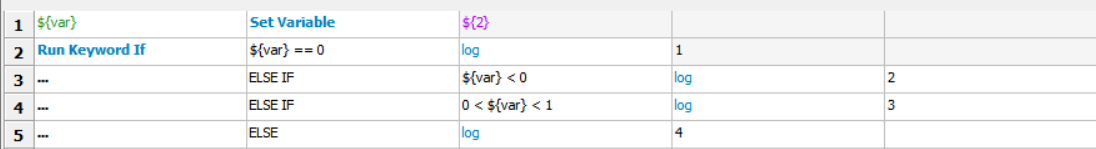
函数结构范例：

Run Keyword If  判断条件  其他关键字

...    ELSE IF   判断条件  其他关键字

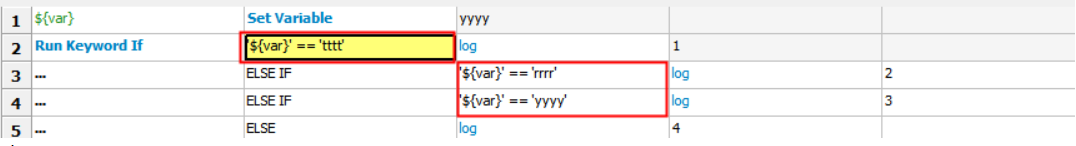
...     ELSE  判断条件  其他关键字

1）判断数值



注：if语句可以写在同一行，如果需要换行，则需在第二行的第一列加三个点（...）

2）判断字符

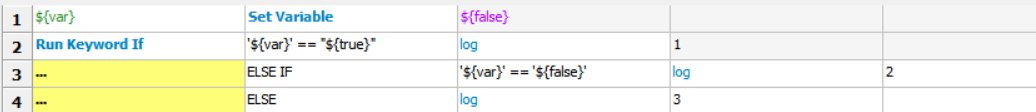


注意：

1.变量${var}值为字符时，在判断条件中其需要加单引号或双引号（如上图红色框中所示），为’${var}’或者”${var}”。

2.在判断条件中，与变量${var}对比的字符常量也需要加单引号或双引号，为’tttt’或者”tttt”。

3）判断布尔类型

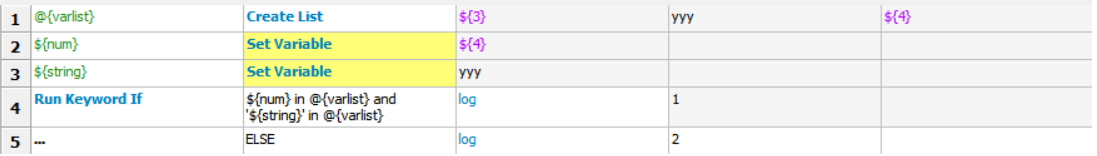


注意：

1.变量${var}值为true或false时，在判断条件中其需要加单引号或双引号，为’${var}’或者”${var}”。

2.在判断条件中，与变量${var}对比的布尔值也需要加单引号或双引号，为’${true}’或者”${true}”。

4）判断列表是否包含元素



注意：

1.变量${string}值为字符时，在判断条件中其需要加单引号或双引号，为’${string}’或者”${string}”。

5）判断列表是否相等

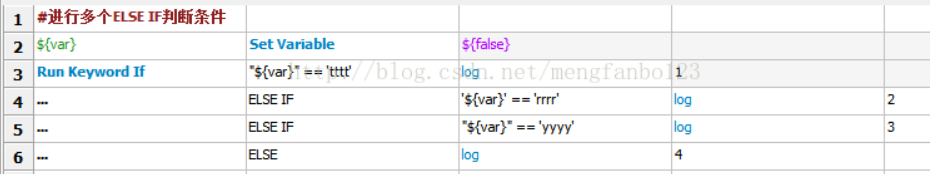


6）判断列表元素是否相等



1.列表元素值(如上图中@{varlist2}[1])为字符时，在判断条件中其需要加单引号或双引号，如下图中@{varlist2}[1]，变为’@{varlist2}[1]’或者”@{varlist2}[1]”。

7）多个判断条件



8）综合使用



### For循环

格式1：

:For 变量  IN  序列(or 列表)

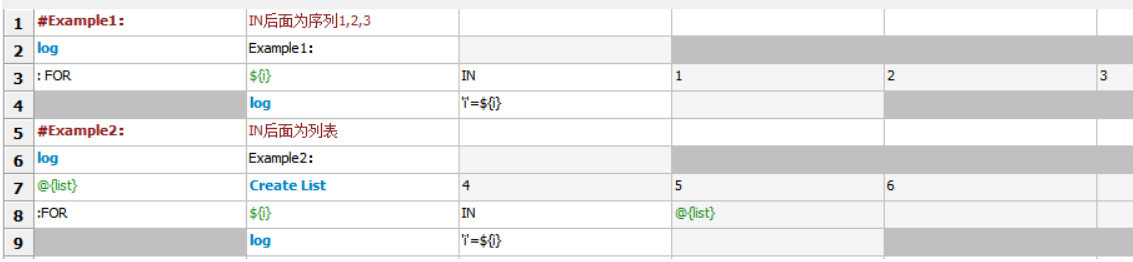
     关键字 参数值

 格式2：

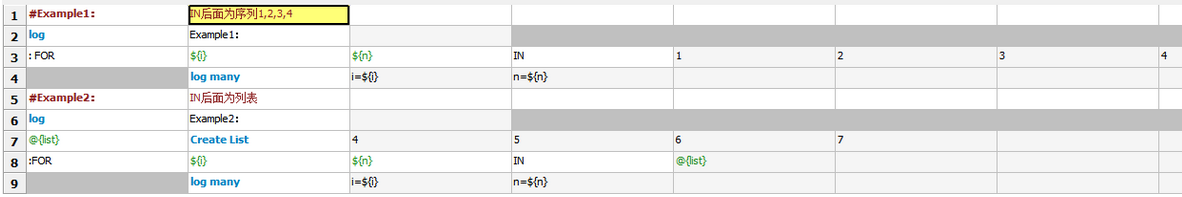
:For 变量  IN RANGE 循环限量

     关键字  参数值

1）单个循环变量

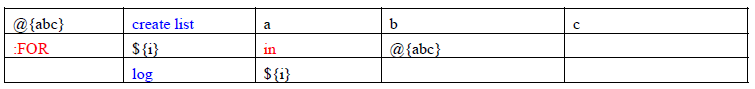


2）多个循环变量



3）遍历列表

通过过“:for”循环来来遍历@{abc}列表中的字符。



4）使用for ... in range

通过“：for”定义for 循环；in range 用于指定循环的范围。



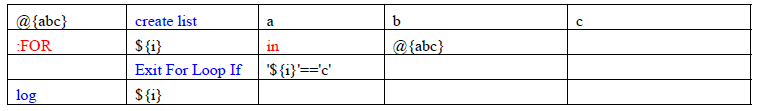
注意，in range 定义为10，它的范围是0~9。





5）循环中的判断

通过“Exit For Loop If”关键字时行for 循环内的判断，当满足Exit For Loop If 条件后，循环结束。



使用Exit For Loop

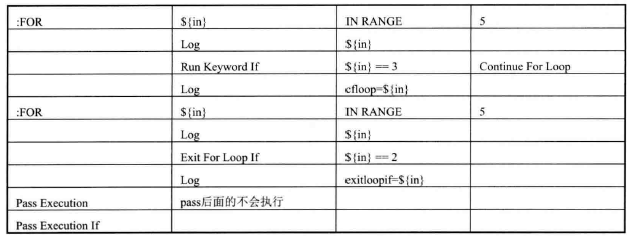


6）综合使用

Continue For Loop：继续执行下一次循环，当前循环内后面的脚本不在执行；

Exit For Loop If：满足条件时推出循环；

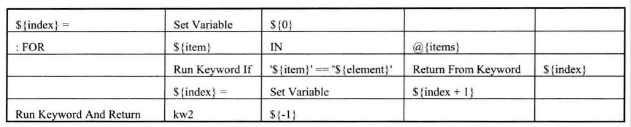
Pass Execution：直接给一个Pass的结果，当前案例后面的脚本都不再执行



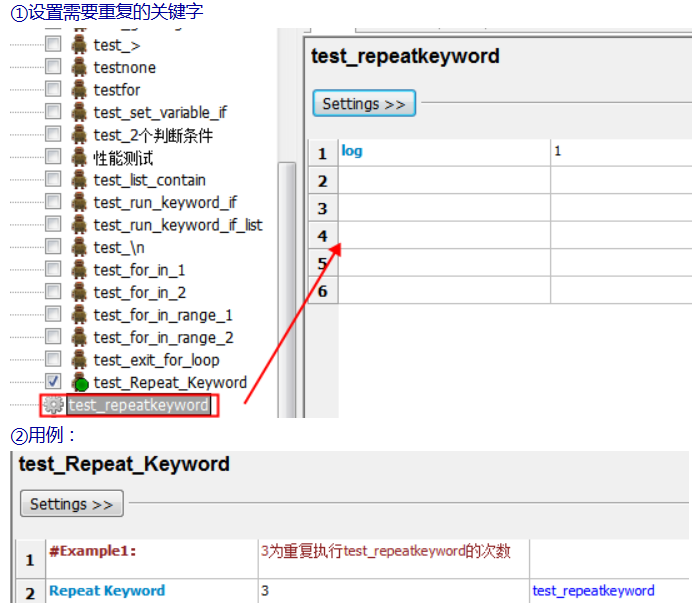
Return From Keyword：在程序中间进行返回

Run Keyword And Return：实际上实现了一个闭包的关键字处理，调用另一个关键字，然后把那个关键字的返回值作为当前关键字的返回值返回。

以下是用来判断一个元素是否在list中存在，如果存在则返回它的${index}，如果不存在，则返回-1。

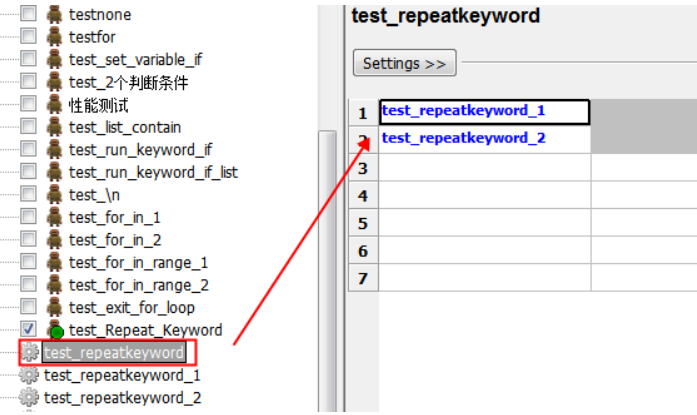


7）重复执行单个关键字



8）重复执行多个关键字

如果需要重复执行多个关键字，可以把多个关键字封装在一个关键字里面，然后在用例里重复执行。



## 编码与解码

### 常见编码场景

情况1：网址路径中包含汉字

输入网址“http://zh.wikipedia.org/wiki/春节 ，查看HTTP请求的头信息，会发现IE实际查询的网址是“http://zh.wikipedia.org/wiki/%E6%98%A5%E8%8A%82 ”。

结论1：网址路径的编码，用的是utf-8编码。

情况2：查询字符串包含汉字

在IE中输入网址“http://www.baidu.com/s?wd=春节 ”。注意，“春节”这两个字此时 属于查询字符串，不属于网址路径

查看HTTP请求的头信息，会发现IE将“春节”转化成了一个乱码。切换到十六进制方式，才能清楚地看到，“春节”被转成了“B4 BA BD DA”。

结论2：查询字符串的编码，用的是操作系统的默认编码。

情况3：Get方法生成的URL包含汉字

前面说的是直接输入网址的情况，但是更常见的情况是，在已打开的网页上，直接用Get或Post方法发出HTTP请求。

根据台湾中兴大学[吕瑞麟老师的试验](http://web.nchu.edu.tw/%7Ejlu/classes/xml/ajax/urlencoding.shtml) ，这时的编码方法由网页的编码决定，也就是由HTML源码中字符集的设定决定。

　　<meta http-equiv="Content-Type" content="text/html;charset=xxxx">

如果上面这一行最后的charset是UTF-8，则URL就以UTF-8编码；如果是GB2312，URL就以GB2312编码。

结论3：GET和POST方法的编码，用的是网页的编码。

情况4：Ajax调用的URL包含汉字

前面三种情况都是由浏览器发出HTTP请求，最后一种情况则是由Javascript生成HTTP请求，也就是Ajax调用

结论4：在Ajax调用中，IE总是采用GB2312编码（操作系统的默认编码），而Firefox总是 采用utf-8编码。

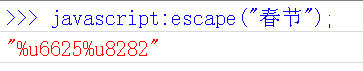
### 统一编码：

原理：使用Javascript先对URL编码，然后再向服务器提交，不要给浏览器插手的机会。因为Javascript的输出总是一致 的，所以就保证了服务器得到的数据是格式统一的。

Javascript语言用于编码的函数：

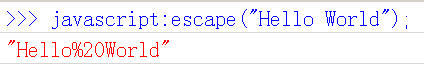
1. escape()

实际上，escape()不能直接用于URL编码，它的真正作用是返回一个字符的Unicode编码值。比如“春节”的返回结果 是%u6625%u8282，也就是说在Unicode字符集中，“春”是第6625个（十六进制）字符，“节”是第8282个（十六进制）字符。



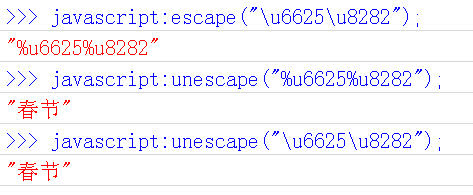
它的具体规则是，除了ASCII字母、数字、标点符号“@ \* \_ + - . /”以外，对其他所有字符进行编码。在/u0000到/u00ff之间的符号被转成%xx的形式，其余符号被转成%uxxxx的形式。对应的解码函数是 unescape()。

所以，“Hello World”的escape()编码就是“Hello%20World”。因为空格的Unicode值是20（十六进制）。



还有两个地方需要注意。

首先，无论网页的原始编码是什么，一旦被Javascript编码，就都变为unicode字符。也就是说，Javascipt函数的输入和输出， 默认都是Unicode字符。这一点对下面两个函数也适用。

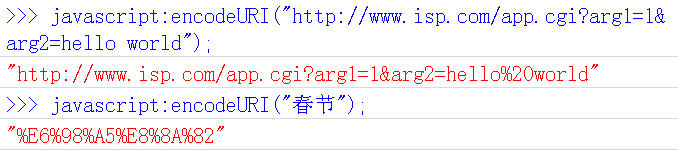


其次，escape()不对“+”编码。但是我们知道，网页在提交表单的时候，如果有空格，则会被转化为+字符。服务器处理数据的时候，会把+号处 理成空格。所以，使用的时候要小心。

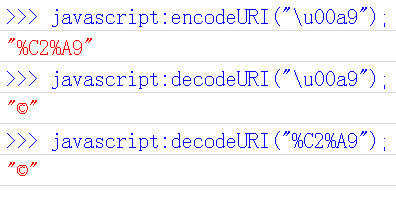
2.encodeURI()

encodeURI()是Javascript中真正用来对URL编码的函数。

它着眼于对整个URL进行编码，因此除了常见的符号以外，对其他一些在网址中有特殊含义的符号“; / ? : @ & = + $ , #”，也不进行编码。编码后，它输出符号的utf-8形式，并且在每个字节前加上%。



它对应的解码函数是decodeURI()。

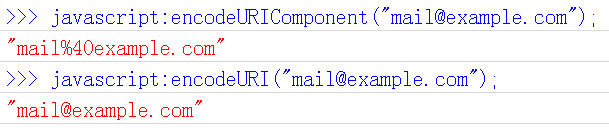


需要注意的是，它不对单引号'编码。

3.encodeURIComponent()

最后一个Javascript编码函数是encodeURIComponent()。与encodeURI()的区别是，它用于对URL的组成部分 进行个别编码，而不用于对整个URL进行编码。

因此，“; / ? : @ & = + $ , #”，这些在encodeURI()中不被编码的符号，在encodeURIComponent()中统统会被编码。至于具体的编码方法，两者是一样。



它对应的解码函数是decodeURIComponent()。

注：网页里的form编码其实不完全取决于网页编码，form标记中有一个accept-charset属性，在非ie浏览器种，如果将其赋值(比如 accept-charset="UTF-8")，则表单会按照这个值表示的编码方式进行提交。

# 常用库及关键字

## Builtin库

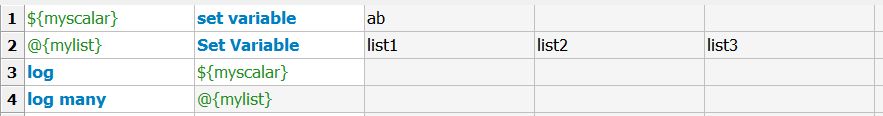
Builtin 为Robot Framework 标准类库，该库会默认加载。

### log（打印）

功能：输出Scalar变量的值，与编程语言里的“print”功能类似

### log many

功能：输出List变量的值



### set log level

设置log级别。

可设置的值为：TRACE, DEBUG, INFO (默认), WARN, ERROR and NONE (no logging).

### log to console

将指定的log信息输出到命令行

### Set variable

功能：用于定义变量



### Catenate（拼接字符串）

功能：用于连接对象

例如：使用“Catenate”关键字连接多个信息。

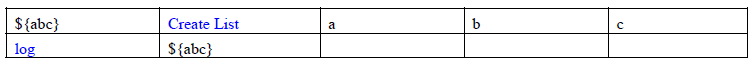


加上“SEPARATOR=”可以对多个连接的信息进行分割。



### Create List

功能：用于定义列表



执行结果：

Starting test: 测试项目.测试套件.test case

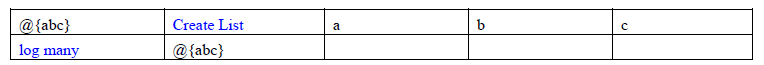
20150303 10:23:20.760 : INFO : ${a} = [u'a', u'b', u'c']

20150303 10:23:20.762 : INFO : [u'a', u'b', u'c']

Ending test: 测试项目.测试套件.test case

每个字符串前面加u，是为了统一编码问题，将字符串转为Unicode 编码。

如果通过“@{}”去定义列表的话，可以通过“log many”关键字进行打印



执行结果：

Starting test: 测试项目.测试套件.test case

20150303 17:04:30.631 : INFO : @{abc} = [ a | b | c ]

20150303 17:04:30.632 : INFO : a

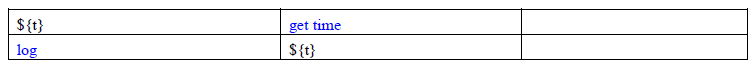
20150303 17:04:30.633 : INFO : b

20150303 17:04:30.633 : INFO : c

Ending test: 测试项目.测试套件.test case

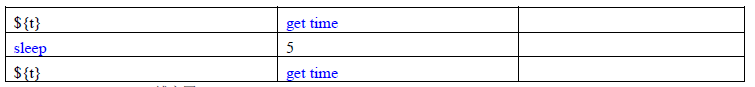
### get time

功能：获取当前时间



### sleep

功能：设置休眠指定的时间，默认单位为秒。



### Import Library

1.调用框架

功能：导入标准和第三方模块或框架



假如我们通过“Import Library”导入一个不存在的模块，Robot Framework 会遍历Python 安装目录下的相关目录查找该模块。

2.调用Python文件

test.py

#coding=utf-8

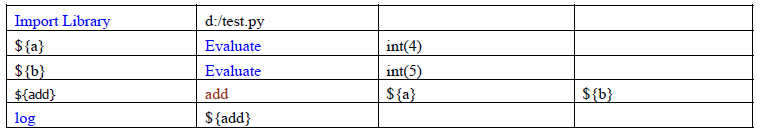
def add(a,b):

return a+b

if \_\_name\_\_ == "\_\_main\_\_":

a = add(4,5)

print a



注： Robot Framework 默认会把表格的数据当做字符串进行处理，如果需要使用整型，需调用Python中的int()方法。

### Comment（添加注释）

功能：用于设置脚本中的注释。

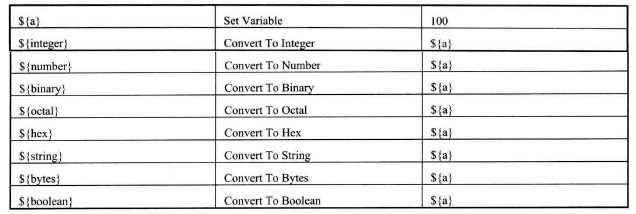


也可以像Python 一样使用“#”号进行注释。



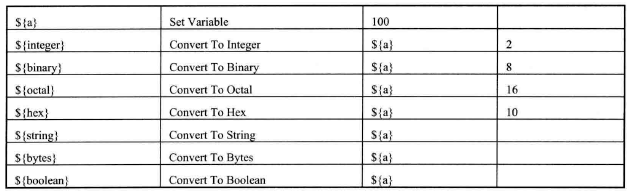
### Convert to Integer

1.不带参数：将$(a)（默认为十进制）转化为指定类型



注：其中Number是Python的float类型。

2.带参数：将$(a)（参数指定的进制）转化为指定类型



### Fail和Fatal Error？？？

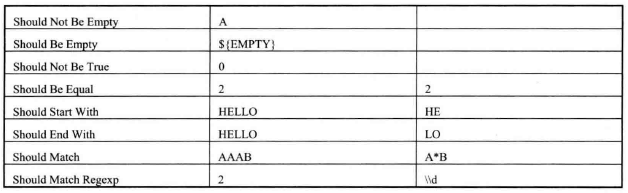
可以自己增加某些条件的判断来执行Fail，当Fail执行后，当前的案例会停止，后续的案例会继续执行。



Fatal和Fail的差别就是它会停止所有的案例，包括当前案例之后没有运行的案例



### Should（not）be





### Get Count

获取第二个参数在第一个参数中有几个。



### Get Length

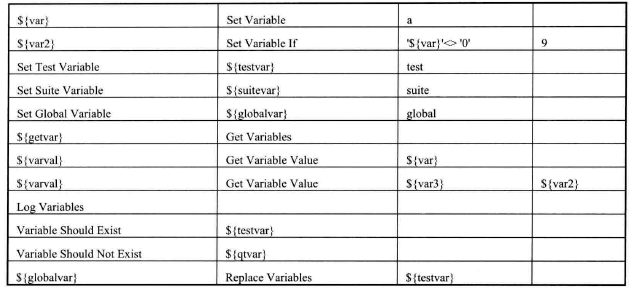
获取list变量中的元素个数，如果后面的参数是Scalar变量，就是获取字符的个数。



### Set Test（Suite、Global Variable）

设置变量作用域

### Get Variable



获取变量的值，它有2个参数，如果参数1的变量没有找到，则获取参数2的变量的值。默认参数2的值是None。

### No Operation

通常用在还没设计好的用户关键字里，因为关键字不写任何脚本执行案例会报错，所以加上该关键字执行案例就不会报错，也不会做任何操作。

## Evaluate（调用Python方法）

该关键字属于Builtin库，通过它可以调用Python中所提供的方法。

### 随即数

一般用evaluate都是前面放变量接收值，第三列是具体的运算表达式，第四列是要用到的Python的module。

python中使用random 模块的randint()方法用于获取随机数:

>>> import random

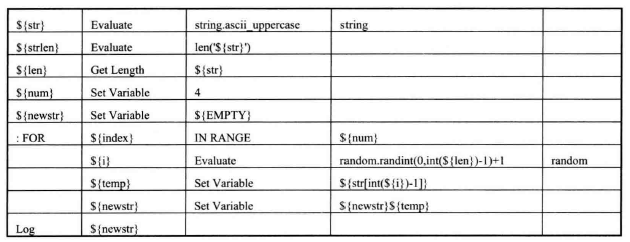
>>> random.randint(1000, 9999)

在Robot Framework 中使用“Evauate”也可以调用Python 所提供的random 模块下的randint()方法

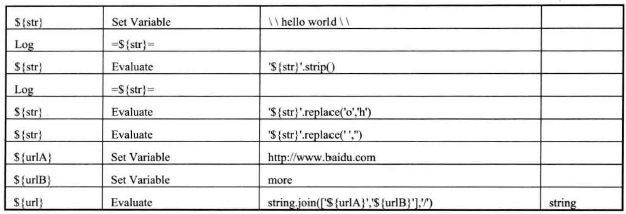


### 随机字符串

首先通过string来获取一个全是大写的ascii因为字母全集，然后通过len（或者get length）获取字符个数。通过循环每次随机从${str}里取出一个字母，然后拼接起来并放到${newstr}里，循环结束也就完成了随机字符串的生成。



### 处理字符串

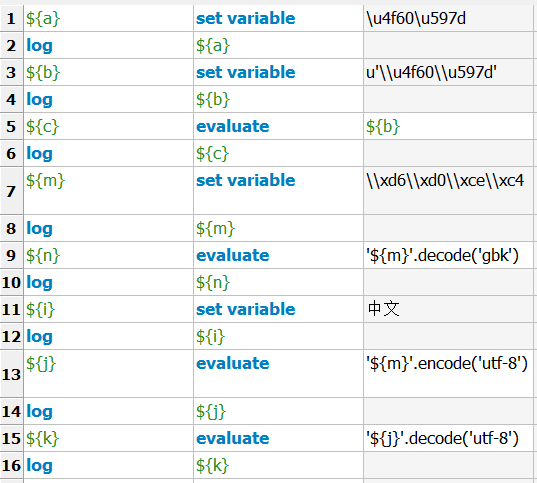


### 中文处理

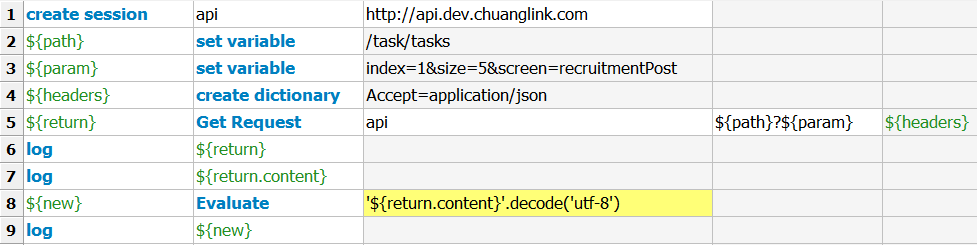
在测试中最常见的就是类似"u'\\u4e2d\\u6587'"或者"\\xd6\\xd0\\xce\\xc4"这两种，前者是Unicode字符，后者是gbk字符。

ride默认是Unicode，所以如果遇到这两种情况，基本上直接用就可以了。

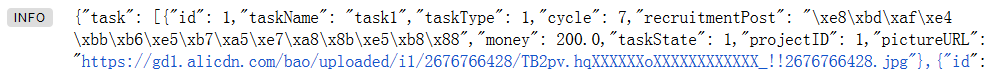
如果一定要看到中文，就需要使用Evaluate关键字了：



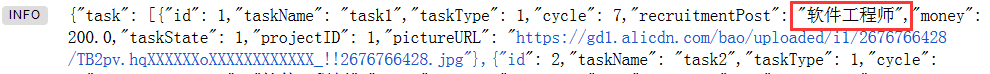
在接口测试中，发送请求后返回的内容中包含汉字，如果需要显示汉字，则需要使用evaluate进行转换。



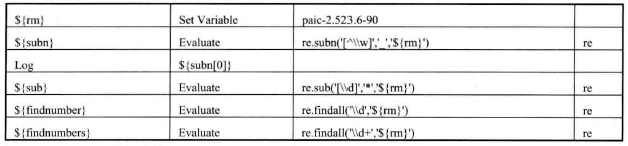
转换前：



转换后：



### 正则表达式



### 日期处理



### 执行本地程序

通过调用Python 所提供的os模块中的system()方法可以执行本地Python 文件。至于在.py 文件中Python 可以做任何想做的事。



注：对于system()方法来说，它也不单单可执行Python 文件，任何在cmd 命令提示符下可运行文件和命令，它都可以执行。

不过，一般情况下不建议通过system()方法去执行外部程序。这样做其实就脱离了Robot Framework，也就是说不管Robot Framework 什么事了。我们尽量把要做的事情通过Python 封装成关键字给Robot Framework 使用。



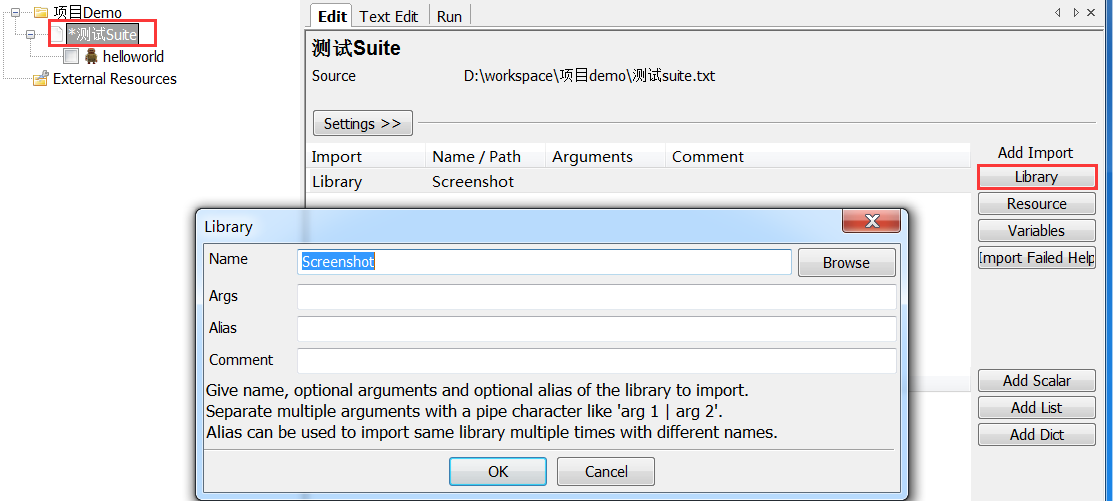
## Screenshot 库

Scrennshot同样为Robot Framework 标准类库，该库需要手动加载。

### Take Screenshot（截屏）

功能：截取当前屏幕。

1.加载Scrennshot库



2.使用



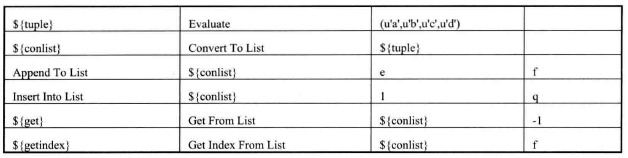
## DateTime

## Collections 库

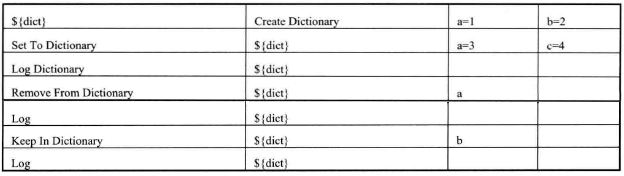
Collections 库同样为Robot Framework 标准类库，该库需要手动加载。

Collections 库提供的关键字主要用于列表、索引、字典的处理。

### List



### Dictionary



### Create Dictionary

功能：用于创建字典。



### Get Dictionary Items

功能：获取字典中的key 和value。

### Get Dictionary Keys

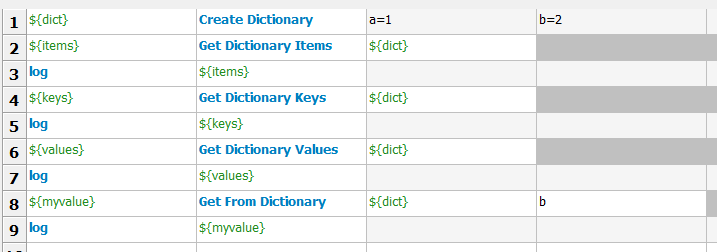
功能：获取字典中的key。

### Get Dictionary Values

功能：获取字典中的value。

### Get From Dictionary

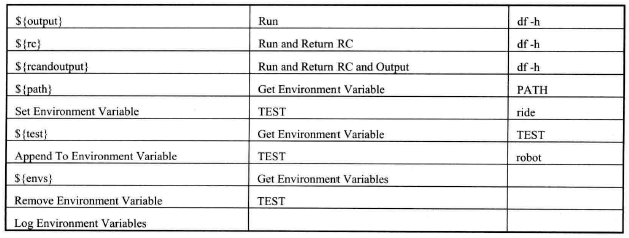
功能：获取字典中的key 对应的value



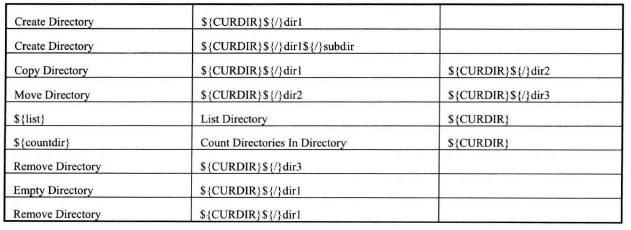
## OperatingSystem

用于操作系统的操作，比如环境变量、文件、文件夹、路径等。

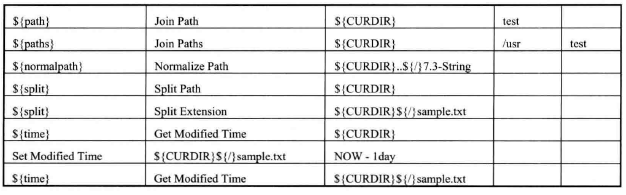
### Environment



### Directory



### Path



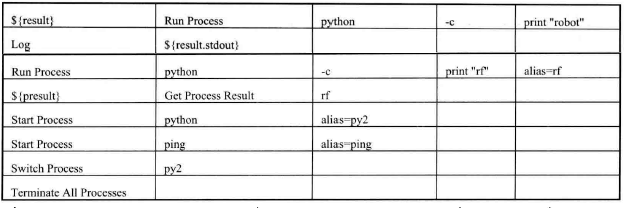
## Process

用于操作进程。

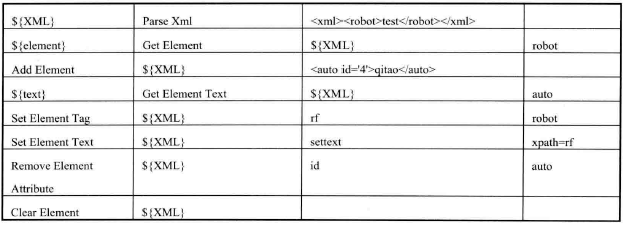
Run Process的两种用法：

1.直接返回结果对象

2.通过get Process result获得结果对象，但返回的对象用法是一样的。



## XML



## DatabaseLibrary 库

DatabaseLibrary 库用于数据库的操作, 它兼容任何数据库API 规范2.0 模块。通过查询数据库来验证测试结果。

### 安装DatabaseLibrary 库

DatabaseLibrary 下载地址：

https://pypi.python.org/pypi/robotframework-databaselibrary/0.6

在线文档：

http://franz-see.github.io/Robotframework-Database-Library/

如果像安装普通的Python 程序，可以下载tar.gz 文件，解压并运行setup.py 文件进行安装。

C:\robot\robotframework-databaselibrary-0.6>python setup.py install

如果已经安装了pip，可以通过pip 命令安装更为方便和快捷：

C:\Python27\Lib\site-packages>pip install robotframework-databaselibrary

### 安装数据库驱动

现在只安装DatabaseLibrary 库，如果要操作不同的数据库，还需要安装相应的数据库驱动。

Oracle 数据库驱动：cx\_Oracle

https://pypi.python.org/pypi/cx\_Oracle

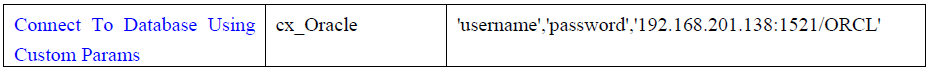
cx\_Oracle 是一个用来连接并操作Oracle 数据库的Python 扩展模块， 支持包括Oracle 9.2 10.2以及11.1 等版本。

MySQL 数据库驱动：PyMySQL

https://pypi.python.org/pypi/PyMySQL/

这个包包含一个纯python MySQL 客户端库。

### 连接Oracle数据库



Connect To Database Using Custom Params ：连接Oracle 数据库关键字。

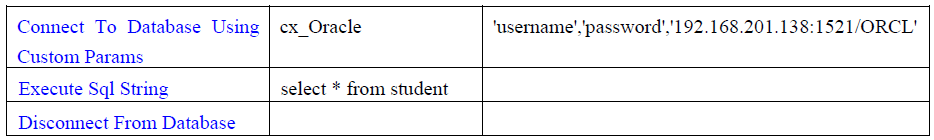
cx\_Oracle：连接oracle 驱动。

'username','password','192.168.201.138:1521/ORCL' ：

连接数据库配置信息，用户名，密码，IP 地址，端口号，数据库名。

### 执行SQL 语句

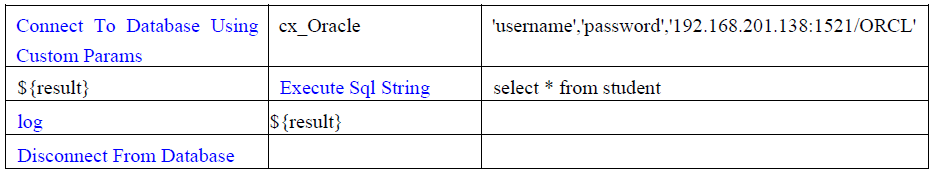
执行sql 语句：



Execute Sql String 关键字用于执行sql 语句。注意sql 语句结尾不要有分号“；”。

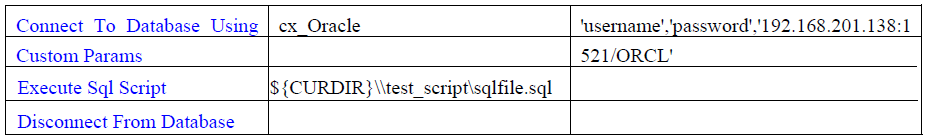
Disconnect From Database 关键字用于断开与数据的连接。

sql 语句结果的输出到测试报告：



### 执行SQL 文件

执行sql 文件：



Execute Sql Script 关键字用于执行SQL 文件。

${CURDIR} 表示当前项目路径。

### 添加系统关键字

在Database Library 库中所提供的Execute Sql Script 不支持sql 脚本文件中包含begin,end 函数，就需要我们自己添加关键字了。

找到....\Python27\Lib\site-packages\DatabaseLibrary\ 目录下的query.py 文件，

创建execute\_sql\_funcfile 函数（关键字）

## String库

String库主要用来操作字符串，需要手动加载该库。

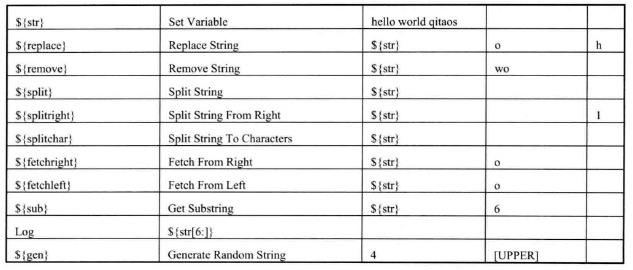
使用evaluate：



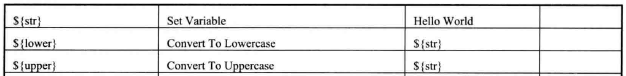
使用String库：



### String

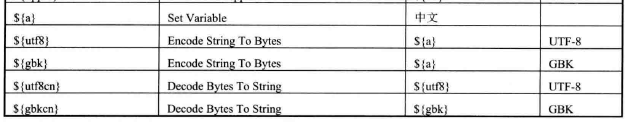


### Convert To

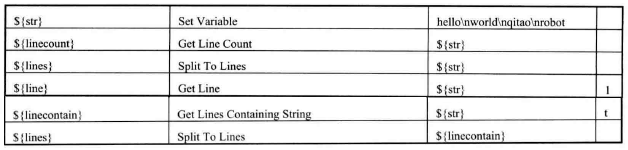


### Encode String to

### Decode Bytes to



### Line（多行文本处理）



注：第一行使用\n换行

Get Line返回指定的行

Get Lines Containing String返回包含某个字符串的行。

## Dialogs

## Telnet

## Remote

# [Selenium2Library](https://github.com/robotframework/Selenium2Library) 库

## 基础使用

[Selenium2Library](https://github.com/robotframework/Selenium2Library)库用于web自动化测试，是robot framework的扩展库，需要手动安装。

安装成功后，需要在ride 的测试套件中导入该库，才可以使用。

注：对Robot Framework 框架的Selenium 库有两个： SeleniumLibrary（基于Selenium1.0） 和Selenium2Library （基于Selenium1.0）。如果没有历史遗留问题，我们直接使用Selenium2Library。

### 安装Selenium2Library库

下载地址:https://pypi.python.org/pypi/robotframework-selenium2library

方法1：下载exe文件安装

可以通过下载exe 程序进行安装，Robot framework-selenium2library 分别提供了win-amd64.exe和win32.exe 两个Windows 版本，根据自己的环境下载相应的版本并安装。

方法2：下载tar.gz 文件安装

也可以下载tar.gz 文件，解压并并使用Python命令运行setup.py 文件进行安装。

C:\robot\Robot framework-selenium2library-1.5.0>python setup.py install

方法3：使用pip工具安装

如果已经安装了pip工具，可以通过pip 命令安装更为方便和快捷：

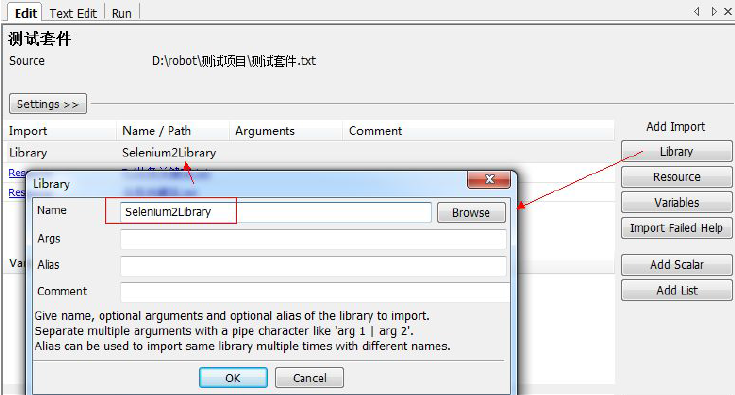
使用Python的pip命令：pip install robotframework-selenium2library

### 使用Selenium2Library库

1）导入Selenium2Library库

Selenium2Library安装成功后，需要在ride 的测试套件中导入该库，才可以使用。

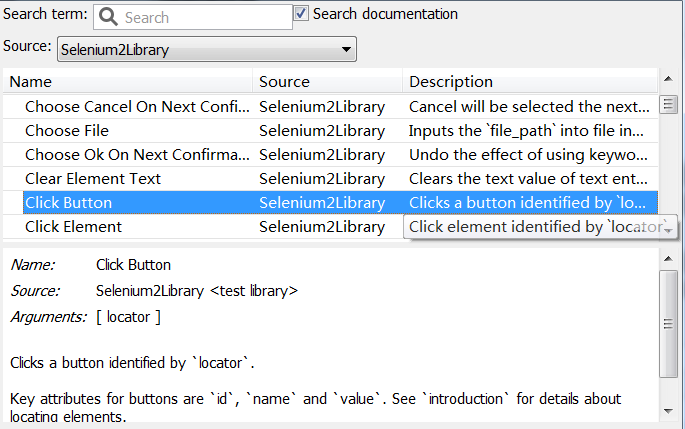
将“Selenium2Library”库添加到相应的测试套件中。

添加完成，黑色示添加的库正常，红色表示库不存。如果为红色，请检查

C:\Python27\Lib\site-packages 目录下是否有Selenium2Library 目录

2）查询Selenium2Library库关键字

通过按F5 快捷键来查询Selenium2Library库所提供的关键字。

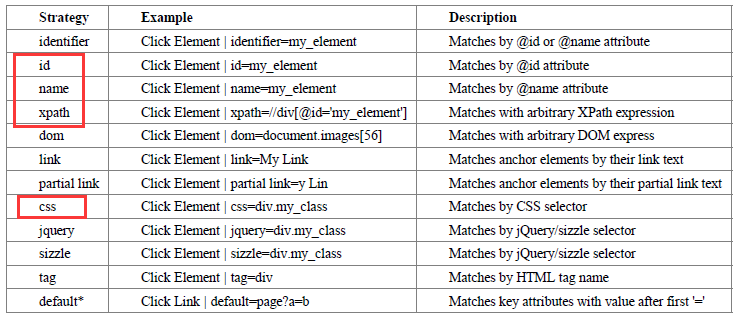


### 简单实例



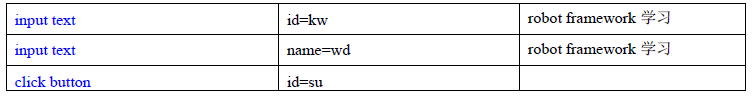
## 元素定位

### Selenium2Library 的定位器



只介绍4 种定位方式，id、name、xpath 和css。介绍id 和name，是因为这两种定位方式非常简单且实用，介绍xpath 和css，是因为这两种定位方式足够强大，可以满足几乎所有定位需求。

### id 和name 定位



Input text 用于输入框的关键字，Click Button 是按钮点击的关键字。

### css 定位

CSS 可以比较灵活选择控件的任意属性，一般情况下定位速度要比XPath 快。

CSS 选择器的常见语法



通过**class** 属性定位：

css=.s\_ipt

css=.bg s\_btn

点号（.）表示通过class 属性来定位元素。

通过**id** 属性定位：

css=#kw

css=#su

井号（#）表示通过id 属性来定位元素。

通过标签名定位：

css=input

在CSS 语言中用标签名定位元素不需要任何符号标识，直接使用标签名即可，但我们前面已经了解到标签名重复的概率非常大，所以通过这种方式很难唯一的标识一个元素。

通过父子关系定位：

css=span>input

上面的写法表示有父亲元素，它的标签名叫span，查找它的所有标签名叫input 的子元素。

通过属性定位：

css=input[autocomplete='off']

css=input[maxlength='100']

css=input[type='submit']

在CSS 当中也可以使用元素的任意属性，只要这些属性可以唯一的标识这个元素。

组合定位：

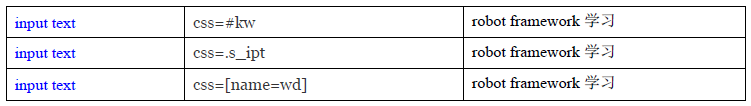
我们当然可以把上面的定位策略组合起来使用，这样就大大加强了元素的唯一性。

css=span.bg s\_ipt\_wr>input.s\_ipt

css=span.bg s\_btn\_wr>input#su

有一个父元素，它的标签名叫span，它有一个class 属性值叫bg s\_ipt\_wr，它有一个子元素，标签名叫input，并且这个子元素的class 属性值叫s\_ipt。

Robot framework 中的写法：



### xpath 定位

**使用xpath 的相对路径**：

1.元素本身：

Xpath 同样可以利用元素自身的属性：

Xpath = //\*[@id=’kw1’]

//表示某个层级下，\*表示某个标签名。@id=kw1 表示这个元素有个id 等于kw1 。

当然，一般也可以指定标签名：

Xpath = //input[@id=’kw1’]

元素本身，可以利用的属性就不只局限为于id 和name ，如：

Xpath = //input[@type=’text’]

Xpath = //input[@autocomplete=’off’]

但要保证这些元素可以唯一的识别一个元素。

2.找上级：

元素的上级属性为：

<form id="form1" class="fm" action="/s" name="f1">

<span class="bg s\_ipt\_wr">

<input id="kw1" class="s\_ipt" type="text" maxlength="100" name="wd" utocomplete="off">

找父节点：

xpath = //span[@class=’bg s\_ipt\_w’]/input

如果父节点没有唯一的属性，可以再往上级找：

xpath = //form[@id=’form1’]/span/input

这样一级一级找上去，直到html ，那么就是一个绝对路径了。

3.布尔值写法：

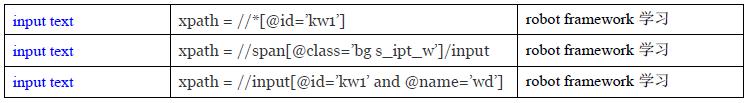
如果一个人的姓名不是唯一的，身份证号也不是唯一的，但是同时叫张三并且身份证号为123 的人却可以唯一的确定一个人。那么可以这样写：

Xpath = //input[@id=’kw1’ and @name=’wd’]

可以and ，当然也可以or（但or 的实际意义不太）：

Xpath = //input[@id=’kw1’ or @name=’wd’]

使用举例：



**使用xpath 的绝对路径**：

xpath = /html/body/div[1]/div[4]/div[2]/div/form/span[1]/input

我们可以从最外层开始找，html 下面的body 下面的div 下面的第4 个div 下面的....input 标签。通过一级一级的锁定就找到了想要的元素。

注：绝对路径的用法往往是在我们迫不得已的时候才用的，大多时候用相对路径更简便。

## Selenium2Library 关键字

关于Selenium2Library 的关键字，可以参考：

http://rtomac.github.io/robotframework-selenium2library/doc/Selenium2Library.html#Unselect%20Checkbox

或者通过F5查找Selenium2Library 关键字库

### Open Browser

Open Browser:打开浏览器



要想通过不同的浏览打开URL 地址，一定要安装浏览器相对应的驱动。

chrome 的驱动为：chromedriver.exe 。

IE 的驱动为：IEDriverServer.exe

浏览器默认为空时启动FireFox。

浏览器对应的关键字：



open browser 同样也可以打开本地html 页面



### Close Browsers

关闭当前浏览器。

### Close All Browser

关闭所有打开的浏览器和缓存重置。

### Maximize Browser Window

使当前打开的浏览器全屏。

### Set Windows Size

用于设置打开浏览器的宽度和高度。以像素为单位



### Get Windows Size

用于获取当前浏览器的宽度和高度。获得浏览浏览器窗口宽、高，将显示在log.html 的日志中。

### Input Text

用于向文本框内输入内容。

### Click Button

用于点击页面上的按钮。

### Click Element

用于点击页面上的元素，单击任何可以点击按钮、文字/图片连接、复选框、单选框、甚至是下拉框等。

### Wait Until Page Contains Element

用于等待页面上的元素显示出来。



42 ： 表示最长等待时间，单位是秒。

Error ： 表示错误提示，自定义错误提示，如：“元素不能正常显示”

### get title

用于获得当前浏览器窗口的title 信息。

通常会将获取的title 传递给一个变量，然后与预期结果进行比较，从而判断当前脚本执行成功。

### get text

用于获取元素的文本信息。

### Get Element Attribute

用于获取元素的属性值



id=kw@name：id=kw 表示定位的元素；@name 获取这个元素的name 属性值。

### get cookies

获得当前浏览器的所有cookie 。



### get cookie value

获得cookie 值。key\_name 表示一对cookie 中key 的name 。



### add cookie

添加cookie。添加一对cooke （key：value）



### delete cookie

删除cookie。删除key 为name 的cookie 信息。



### delete all cookies

删除当前浏览器的所有cookies。



### Should Contain

判断参数1是否包含参数2





### Select Frame

进入表单

有时候在页面中会出现表单嵌套，这个时候需要进入到表单才能操作相关元素。



### Unselect Frame

退出表单（与Select Frame配套使用）



### Select From List by value

选择下拉列表中指定的项



Xpath=//\* [@] 定位下拉框

Vlaue 选择下拉框里的属性值

### Unselect From List by value

不选择下拉列表中指定的项



### Execute Javascript

用于使用JavaScript 代码



# 接口自动化测试

## 安装requests和requestsLibrary

### 安装requests

方法1：

地址：https://pypi.python.org/pypi/requests

python setup.py install

方法2：

pip install requests

### 安装requestsLibrary

requestsLibrary是robot framework的测试库，它的底层是基于requests这个工具的，所以要先安装requests，再安装requestsLibrary

方法1：

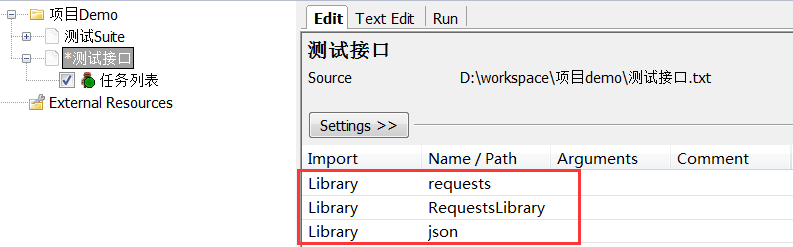
地址：https://pypi.python.org/pypi/robotframework-requests

python setup.py install

方法2：

pip install robotframework-requests

### 添加库到ride中



## requestsLibrary关键字

### Create Session

创建一个session，连接某个服务器

### Create Ntlm Session

也是创建一个session，但加上了域、用户名和密码，用于NTLM认证

### Get Request

通过Get方式发送请求

### Post Request

通过Post方式发送请求

### Head Request

发送一个Head请求

### TO Json

将文本转换为json对象

## 测试接口

### 测试get请求

1.接口文档（http://api.dev.chuanglink.com）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 路径 | 请求方式 | 参数 | 返回值 |
| 任务列表 | /task/tasks | Get | Int32:Index;  Int32:Size;  String:Screen; |  |

# 移动端自动化测试

# 数据库自动化测试

# Robot Framework 分层设计

## 通过创建关键字

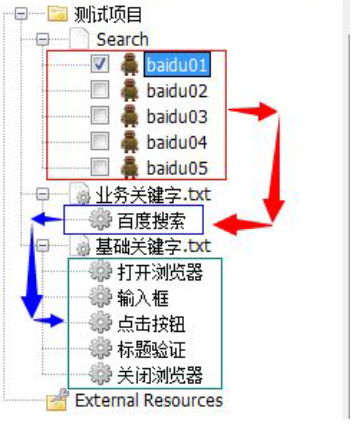
### 思路

将重复的步骤抽象成关键字（函数或方法）的定义，不同的数据抽象成参数

### 创建关键字步骤

参照【ride工具-使用资源】章节

### 继续分层



# 系统关键字开发

## 自定义Python文件

test.py

#coding=utf-8

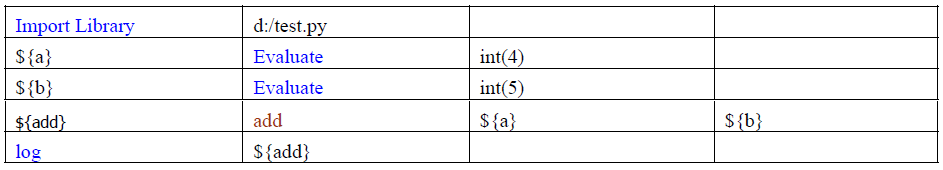
def add(a,b):

return a+b

if \_\_name\_\_ == "\_\_main\_\_":

a = add(4,5)

print a



## 自定义库

### 创建库目录

首先在..\Python27\Lib\site-packages 目录下创建CustomLibrary 目录，用于放自定义的library库。

### 创建库文件

1）在" CustomLibrary "文件夹中新建一个"version.py"文件,用于描述自定义测试库的版本信息，代码如下：

VERSION='1.0'

2）在" CustomLibrary "文件夹中创建runbat.py 文件，代码如下：

#-\*- coding:utf-8 -\*-

'''

created by bugmaster 2015-01-29

'''

from robot.api import logger

from version import VERSION

import os

\_\_version\_\_ = VERSION

class Runbat(object):

def run\_all\_bat(self,path):

u'''接收一个目录的路径，并执行目录下的所有bat 文件.例

| run all bat | filepath |

'''

for root,dirs,files in os.walk(path):

for f in files:

if os.path.splitext(f)[1] == '.bat':

os.chdir(root)

#print root,f

os.system(f)

def \_\_execute\_sql(self, path):

logger.debug("Executing : %s" % path)

print path

def decode(self,customerstr):

return customerstr.decode('utf-8')

if \_\_name\_\_ == "\_\_main\_\_":

path = u'D:\\test\_boject'

run = Runbat()

run.run\_all\_bat(path)

注意在run\_all\_bat()方法下面加上清晰的注释，最好给个实例。这样在robot framework 的帮助中能看到这些信息，便于使用者理解这个关键字的使用。

3）在CustomLibrary 目录下创建\_\_init\_\_.py 文件，用于在robot framework 启动后加载这个关键字，代码如下：

#-\*- coding:utf-8 -\*-

from runbat import Runbat

from version import VERSION

\_\_version\_\_ = VERSION

class CustomLibrary(Runbat):

"""

这里也可以装x 的写上我们创建的CustomLibrary如何如何。

"""

ROBOT\_LIBRARY\_SCOPE = 'GLOBAL'

这个文件中其实有用的信息就三行，但必不可少。robot framwork 在启动时会加载这个文件，因为在这个文件里指明了有个runbat 文件下面有个Runbat 类。从而加载类里的方法（run\_all\_bat()）。

4）检查以上Python文件是否运行正常（不报错），没问题后在ride中导入CustomLibrary库，如果显示黑色则导入成功。