VBS 简易教程
www.itestware.com

VBS 简介

QTP 的测试脚本语言采用的是 VBScript，而 QTP 的专家视图其实就是一个 VBScript 脚本编辑器，因此，测试人员可以利用简单易用而不失强大和灵活特性的 VBScript 脚本语言来增强自动化测试。

什么是 VBS?

VBScript，简称 VBS，是 VB 的一个安全子集。VBScript 是脚本语言，它与其它脚本语言有很多共同的特点，例如变量无关、解释执行等。

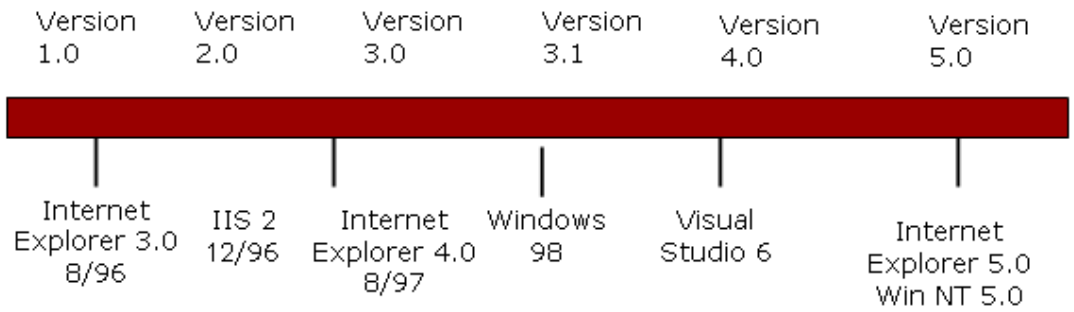
VBScript 是大小写不敏感的，例如下面的脚本中，定义变量时使用小写“str”，使用变量时使用大写“STR”：

```
Dim str
str = "hello world!"
Msgbox STR
```

VBS 可在 Windows 本地执行（依靠 WSH（Windows 脚本宿主）来执行），也可在 IIS 中执行，从而实现 WEB 页面的动态效果。

VBS 的发展历史

VBS 早在 1996 年就出现了，时至今日，已经发展到 5.6 版本，QTP 使用的是 VBS 的 5.6 版本。



VBS 的数据类型

我们会看到很多脚本语言都是数据类型无关的，通常，脚本语言为了简化和方便起见，会对数据类型进行自动的转换。VBS 也一样，只有一种数据类型 – Variant。

Variant 既能是数字，又能是字符串，还能是各种类型的对象，我们在使用变量时可以不声明变量是字符串还是数字，VBS 在执行时会自动识别和转变类型。这不禁让人想起“孙大圣”来，“变变变”。

例如下面的脚本：

```
Dim var
var = 123
Msgbox "输入的是：" & var
```

Var 变量赋值的是数字，但是通过 Msgbox 显示的是字符串。

然而，这种不预先声明变量类型而直接使用的做法虽然比较方便，但是会带来脚本可读性和可理解性的问题，一般规范的脚本编写要求还是尽量能预先定义好变量的数据类型，然后再使用。

在 VBS 中，提供了 SubType 来支持这种做法。SubType 的数据类型包括下面的各种类型：

- Empty：未初始化的 Variant。对于数值变量而言为 0，对于字符串变量而言，则为零长度的字符串。
- Null：不包含任何有效数据的 Variant。
- Boolean：布尔变量，值为 True 或 False。
- Byte：0 到 255 之间的整数。
- Integer：-32768 到 32768 之间的整数。
- Currency：-922 337 203 685 477.5808 到 922 337 203 685 477.5807。
- Long：-2 147 483 648 到 2 147 483 647 之间的整数。
- Single：单精度浮点数，负数范围从-3.402823E38 到-1.401298E-45 之间，正数范围从 1.401298E-45 到 3.402823E38 之间。
- Double：双精度浮点数，负数范围从 -1.79769313486232E308 到 -4.94065645841247E-324 之间，正数范围从 4.94065645841247E-324 到 1.79769313486232E308 之间。
- Date（Time）：日期型，日期范围从公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日。
- String：字符串型，最大长度可为 20 亿个字符。

- Object：包含对象。
- Error：包含错误号。

可用 VarType 函数来返回某个数据的 Variant 子类型，如下面的代码：

```
Dim VarTypeCheck
VarTypeCheck = VarType(300) ' 返回 2,表示整型
MsgBox VarTypeCheck
VarTypeCheck = VarType(#10/19/62#)    ' 返回 7，表示日期型
MsgBox VarTypeCheck
VarTypeCheck = VarType("VBScript")    ' 返回 8，表示字符型
MsgBox VarTypeCheck
```

这可以用我自己来打个比喻，我是一个人（Variant），具有所有普通人所具备的基本特征。但是在我来到这个世界之前，我是“Null”的，我甚至还不是“我”…

刚出生的时候，我是“Empty”的，我既没有钱，也不会说话…

在父母生下我之前，他们不知道我是男的还是女的，但是他们坚信我要么是男孩，要么是女孩，还好“Boolean”不会有第三个值…

后来我长大了，在人生这个大舞台上的不同场合，我要扮演的角色是不一样的，在公司，我是测试经理，回到家，我要孝顺父母，教导小孩，侍候老婆，因此我需要随时在 Byte、Integer、Long、Single、Double 之间来回切换，因为有时候我需要非常地精确，有时候我要简单点…

虽然我坚信人的一生有无限的可能，赚的钱可以高达“922 337 203 685 477.5807”，但是一个不小心，也可能成为负资产，身价掉到“-922 337 203 685 477.5807”…

有时候会觉得人生好郁闷啊，虽然脑海里储存了 20 亿个“String”可以用来描述这种郁闷的感觉，但是很多时候我就是说不出话来…

后来，我想想，从公元 100 年 1 月 1 日到现在，甚至我敢打赌，直到公元 9999 年 12 月 31 日，我们所有人其实都只是一个“Object”而已，都会经过生老病死，因此，有些东西又何必那么执着呢？偶尔出一下“Error”也没什么大不了的嘛！

VBS 的变量

何谓变量？

简单而言，可变的东西就叫变量，不可变的东西叫常量。

常量也叫恒量，当然你也可以把它称为“永恒”，这是一般女性都比较相信的一样东西。

但是世界上真正永恒不变的恐怕就只有“变化”这个东西了。

常量是相对的，在 VBS 中，一般常量在函数体内永恒不变，例如下面脚本：

```
Function TestConstVar
   Const localVar    = 100
   Msgbox localVar
End Function
```

用 Const 声明一个常量 localVar，其取值为 100，然后在 TestConstVar 这个函数体中的任何部分使用 localVar 都能获取到不变的值 100。

但是如果跳出了这个函数体，则将不再能获取到 localVar 的值（所以所“永恒是相对的”），例如下面脚本中，第一行代码将得到消息框显示“100”的值，而第二句的 Msgbox 将显示一个空值：

```
TestConstVar ' 调用 TestConstVar  函数
Msgbox localVar ' 在 TestConstVar  函数之外使用其内部定义的常量

Function TestConstVar
   Const localVar    = 100
   Msgbox localVar
End Function
```

Tips：定义常量的好处是让代码可读性和可维护性更强些，通常在集中的一个地方，例如脚本的最开始定义全局的常量，这样易于后续的维护和修改。

既然“永恒是不现实的”，那么我们只能期待好好处理“变量”了。我们尝试把那些容易变化的东西用一个名字记住它，然后就可以随时知道它的变化了、掌控它的变化了，也许这才是真正的“永恒”。

如何定义“永恒”

在 VBS 中，可以用 Dim 来声明一个变量，例如：
Dim MyVariant

也可以不预先定义，而是在变量赋值时直接使用变量，例如：
MyVariant = 100
Msgbox MyVariant

当然，按照规范的 VBS 代码编码规范而言，我们不推荐这样做，规范的做法是用 Dim 预先定义变量，并且在开始的时候使用“Option Explicit”，例如下面脚本：
Option Explicit
Dim MyVariant
MyVariant = 100
Msgbox CStr(MyVaraint)
这样定义后，执行第四句时就会报错，这样就可避免由于“笔误”造成的困扰了。

请注意，上面的脚本中，定义的是“MyVariant”，使用的是“MyVaraint”，一字之差可能造成你的脚本莫名其妙地出错，并且这个错误在你发现这 一字之差之前是很难发现的，因为如果不使用“Option Explicit”的话，它是不会主动提示错误的。

Tips：定义的变量之间可以用逗号隔开，这样可以在一行中定义多个变量，这样会节省不少代码行呢!
Dim Top,Left,Right,Bottom
不过，如果你的老板是以代码行来计算你的工作量的话，建议还是一行定义一个变量...

VBS 的数组

从变量到数组

变量只能存储一个值，如果要想存储多个值，就要考虑使用数组了。

定义数组可以用下面的简单方式：
Dim MyArray(10)

在这里要出一个简单的数学题：上面定义的数组中可以存储多少个值呢？

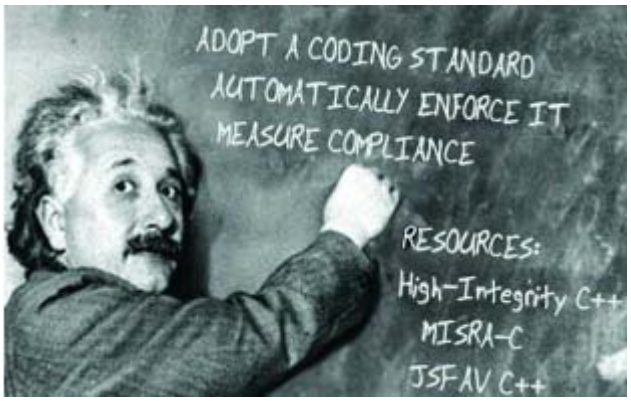
回答是 10 个的“扣 10 分！”

答案是 11 个，因为 VBS 的数组是从 0 开始计数的。可以用 UBOUND 函数来获取数组的最高 Index：

```
Dim MyArray(10)
For I=0 to 10
  MyArray(I) = I
Next
Msgbox UBOUND(MyArray)
```

用一维数组到多维数组

当然，如果一维的数组不够用，不足以表达的话，也可以使用二维、三维，最高去到 60 维的数组，但是我想大部分像我一样智力水平的人是很难理解 60 维的空间是怎样的情形，也许只有爱因斯坦这样的天才可以搞明白它。



一般比较常用的是 2、3 维的数组，例如可以用下面定义的二维数组来代表一个表格的数据：
Dim MyTable(5,10)
MyTable 这个数组可以表示一个 6 行 11 列的表格。

从固定数组到动态数组

数组给了我们扩展变量存储空间的可能性，但是很多时候我们无法预知需要定义一个多大的数组，就像当初预计只是生一个小孩，因此买个两房一厅的房子就够住了，岂料到意外出现了，又来了一个“小天使”，两房一厅就不够用了，因此要考虑“扩容”。

在 VBS 中，可以通过 Redim 来重新定义数组的大小，例如下面的例子：

```
Dim MyArray()
Redim MyArray(2)
MyArray(0) = "客厅"
MyArray(1) = "小 yoyo 的房间"
MyArray(2) = "我和老婆的房间"
' 如果另外一个“小天使”降临了
Redim Preserve MyArray(3)
MyArray(3) = "小天使的房间"
For I = 0 to UBOUND(MyArray)
    Msgbox MyArray(I)
```

Next

注意如果要使用动态数组，则定义数组时不要输入数组大小，例如 Dim MyArray()，然后在后续使用数组之前，通过 Redim 来重新定义数组的大小。

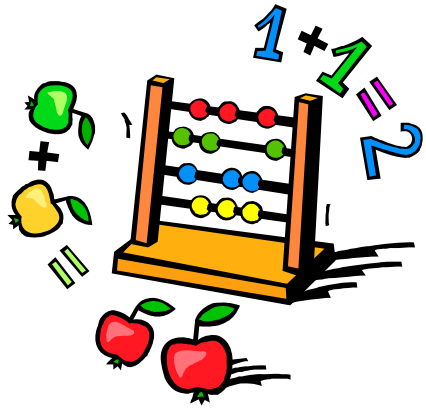
Tips：
Preserve 这个关键字很关键，如果不使用 Preserve 的话，则“小天使”的到来会把我和老婆还有小 yoyo 都踢出去，这样的结局会很糟糕：（

VBS 的运算符

什么是运算符？

如果程序中仅仅能做变量的定义和赋值的话，未免太枯燥了，这好比自己跟自己玩，给自己穿不同的衣服，但是没有其他人做出评价一样。如果变量之间能互相交互，有时候你借我一下钱，有时候我借你一下钱，时不时比较一下谁更帅，如果情投意合的话就结合在一起，嗯…这样的话这个世界才精彩。

为了让变量之间能交互，VBS 提供了运算符。VBS 支持广泛的运算符，包括数学运算符、比较运算符、连接运算符、逻辑运算符。



数学运算

数学运算符包括：
加法：+
减法：-
乘法：*
除法：\

在 VBS 中，如果是数字相加的话，就用“+”号把两个数字连接再赋值即可：

```
Dim a,b,sum
a = 1
b = 1
sum = a + b
Msgbox sum
```

如果 a 和 b 换成是字符串，用“+”连接会等到什么结果呢？

```
Dim a,b,sum
a = "hello"
b = " world!"
sum = a + b
Msgbox sum
```

没错，结果是“hello world!”因为 VBS 能自动识别变量类型，如果是字符串的话，加法操作就变成了字符串的连接操作了，真聪明！

但是，按照规范的做法，我们不鼓励这样使用“+”来连接字符串串，而是用“&”操作符来连接字符串，目的是为了避免混淆，提高代码的可读性和规范性。

除法用“\”来连接两个数字，例如：

```
Dim a,b,Result
a = 100
b = 9
Result = a \b
Msgbox Result
```

结果是 11，哎？100 块钱 9 个人瓜分，每个人得到 11 块钱，还剩 1 块钱哪去了？

嗯，VBS 帮你保留了，因为使用的是“\”运算符，进行的是整除运算，不能整除的就留给 VBS“享用”了。如果想一分不剩地把它瓜分掉的话，就应该掉个方向，使用“/”运算符，进行浮点运算：

```
Dim a,b,Result
a = 100
b = 9
Result = a /b
Msgbox Result
```

这次得到的就是 11.1111111111111，可惜我国货币最小单位是“分”，否则后面的“小钱”也能瓜分掉。

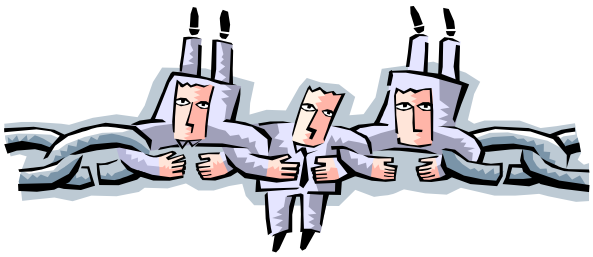
连接运算

如果我们把前面的加法中的“+”换成“&”的话，会出现什么结果呢？

```
Dim a,b,sum
a = 1
b = 1
sum = a & b
Msgbox sum
```

结果是“11”，哇，不错嘛，简直就是变魔术，两个 1 块钱硬币碰一下就变成了 11 块钱。不过，不要开心得太早，VBS 帮你做的可不是加法运算，而是连接运算，它仅仅帮你把两个“1”并排在一起展现给你瞧瞧而已。

“&”用于连接字符串，当然，如果给定的变量不是字符串的话，VBS 会自动把它们转换成字符串再连接在一起。



比较运算

如果有个人，腰缠万贯，找到你，然后非要跟你比一下谁更“幸福”，怎么办呢？我的办法如下：

```
Dim a,b,Result
a = 1000000000
b = "My World"
Result = a > b
Msgbox Result
```

结果是 False，也就是说 a 不大于 b，哈哈，我赢了！（阿 Q 精神万岁！）下次记得找同类型的东西进行比较啰！

VBS 支持以下比较运算符：

等于：=

小于：<

小于或等于：<=

大于：>

大于或等于：>=

不等于：<>
比较的结果要么是 True，要么是 False。

逻辑运算

VBS 支持的逻辑运算操作符包括：
与：And
或：Or
非：Not
异或：Xor

```
Dim a,b,Result
a = True
b = False
Result = a And b
Msgbox Result
```

对于与操作（And），只要参与运算的变量中有一个是 False，则结果为 False，就像现在的计划生育政策一样，父母双方，只要有其中一位是城市户口的，就不允许生第二个小孩。
相比而言或操作（Or）就宽松很多了，只要有其中一位是农村户口的，就允许生第二个小孩，什么时候我国计划生育政策能这么宽松就好了…

VBS 的条件语句

有了变量和数组，可以暂存各种数据，通过变量间的赋值和运算，可以使变量们进行交互。但是如果你是上帝，你也许希望可以更加灵活地控制它们的交互行为，不能随随便便就让一个变量给赋值了，或者随随便便就让一个变量与另外一个变量结合了。

在 VBS 中，可以通过条件语句来限制、控制变量交互行为。VBS 的条件语句有两种，分别是 If...Then...Else 和 Select Case 。

If...Then...Else

先来看 If...Then...Else。
如果你向一位心慕已久的美丽女生求婚，也许要满足一定的条件才行：
如果
　　你有房子和汽车
那么
　　“OK，我答应嫁给你”
否则
　　“现在我们都还不成熟，还是等等吧…”

这在 VBS 中可以这样表达:

```
Dim House,Car
House = null
Car = null
IF IsNull(House) or IsNull(Car) Then
   Msgbox "现在我们都还不成熟，还是等等吧…"
Else
   Msgbox "OK，我答应嫁给你"
End IF
```

当然，条件有可能还要更苛刻些，例如人要长得英俊啦，要有幽默感啦，要专一啦…

有些女孩不会那么苛刻，但是有自己的评分表，满 80 分就 OK：

例如：
如果你很有钱
　那么得 90 分
否则，如果你很英俊
　那么得 85 分
否则，如果你很有才
　那么得 80 分
否则
　“你找别人去吧!”
这在 VBS 中可以这样表达：

```
Dim Money,handsome,talent,Score
Money = 1000
handsome = null
talent = "非常有才"
IF Money > 1000000000 then
   Score = 90
ElseIF NOT IsNull(handsome) then
   Score =85
ElseIF NOT IsNull(talent)    then
   Score =80
Else
   Msgbox "你找别人去吧！"
End IF
IF Score >= 80 then
   Msgbox "OK!我嫁给你吧"
End IF
```

幸好，我还属于比较有才的那种人，否则我老婆当初可能就会对我说：“你找别人去吧!”

Select Case

想当初还是太急了吧？！如果让你重新选择的话，你其实是有很多选择余地的，因为也许你不知道暗恋你的女孩有那么多吧？！她们各具风格和特色，有温柔可爱型的、有身材火辣型的、有贤慧典雅型的。

忽然间理解了韦小宝的烦恼，真不知道该选哪个好，还是写个 VBS 脚本来帮你选一下吧：

```
Dim Style
Style = "身材火辣型"
Select Case Style
Case "温柔可爱型"
        Msgbox "我喜欢!"
Case "身材火辣型"
        Msgbox "超喜欢!"
Case "贤慧典雅型"
        Msgbox "嗯，跟她结婚比较合适!"
Case Else
        Msgbox "你不是我的 Style!"
End Select
```

VBS 的循环语句

大部分人都不能活到 100 岁，为什么呢？每年春夏秋冬 365 天，每天 24 小时，大部分人在重复着每天的生活，但是每天都潜伏着危机，一个不小心，你就不能完成这 100 次的循环。

在 VBS 中，有 4 种方式表达循环：
Do…Loop
While…Wend
For…Next
For…Each

Do…Loop

如果你一生平平安安，快快乐乐的生活到老的话，那真是太幸福了！

```
Age = 0
Do Until Age = 100
   Age = Age + 1
   Msgbox Age
Loop
```

但是很多人没有你这么幸福，上帝在适当的时候就会召见他的：

```
Age = 0
Do Until Age = 100
   Age = Age + 1
   Msgbox Age
   IF Age = 27 Then
      Msgbox "天妒英才，英年早逝"
      Exit Do
   End IF
Loop
```

Tips：VBS 支持把 Until 放到后面，例如：

```
Age = 0
Do
   Age = Age + 1
   Msgbox Age
   IF Age = 27 Then
```

```
    Msgbox "天妒英才，英年早逝"
    Exit Do
  End IF
Loop Until Age = 100
```

While…Wend

正所谓“命里有时终须有，命里无时莫强求”，上帝早就安排好了，只要时机成熟，就能结果。

我曾经看到一对夫妻，非常想要小孩，但是由于种种原因，妻子老是怀不上，我就跟他们说:“放心吧!也许老天正在考验你们，只要不断努力，你们会有的”

```
Dim Try,Got_Baby
Try = 0
Got_Baby = False
While Got_Baby = False
  Try = Try +1
  IF Try = 10 Then
    Got_Baby = True
    Msgbox "恭喜你！经过不懈努力，你终于怀孕了！"
  Else
    Msgbox "Just Try again!"
  End IF
Wend
```

For…Next

挫折总会有的，每天都碰到一些挫折不是更好吗？“小小挫折等于激励!”，总比那些度日如年，每年做重复的事情、每天在做重复的事情、甚至每分钟都在做重复的事情的人，来得精彩吧？！这种人，让他活到一百岁又怎样呢？

```
Dim Year,Day,Hour,Minute,Second
For Year = 1 To 100
  For Day = 1 To 365
    For Hour = 1 To 24
      For Minute = 1 To 60
```

```
            For Second = 1 To 60
               Msgbox CStr(I) & "这样重复地数日子，杀掉我算了！"
            Next
         Next
      Next
   Next
Next
```

郑重声明：如果你运行上述脚本，如果想结束运行，请打开Windows任务管理器，找到Wscript.exe进程，并把它结束掉。

For…Each

如果上天只给你再有 10 天的命，你会做什么呢？你会选择马上结束自己的生命，还是认真地活过这 10 天中的每一天呢?

```
Dim CountDownDay
CountDownDay = Array("画一天的画","跟老婆聊一天","带上小孩去动物园玩一天","和老爸一起抽烟、喝茶、饮酒过一天","带老妈去坐火车游一天","找小学、中学、大学的同学们聚一天","拜访小学、中学、大学的老师一天","骑上单车游一天","用一天的时间把所有以前想吃而不敢吃、不舍得吃的东西都吃遍","躺在床上静静地睡一天")
For Each element In CountDownDay
   Msgbox element
Next
```

VBS 的程序

如果把所有脚本一行行地堆砌在一起，当然也可以把所有功能实现，但是这种代码被称为“意大利面条”（所有功能模块都搅在一起了），或者被称为“铁板程序”（一行行堆上去）。

为了使代码可读性更强、可维护性和可重用性更强，需要把这些堆砌在一起的脚本按不同的功能、过程进行分类，封装成函数或子过程，函数、过程之间互相可以调用，从而增强代码的可重用性和可维护性；这样一个函数或过程中的脚本也不会太多，不会看得眼花缭乱的，因此也就增强了代码的可读性。

在 VBS 中，程序分为两种：Sub 程序和 Function 程序

Sub

在 VBS 中，Sub 类型的程序是指那些执行了脚本但是不返回结果的程序。如果把工人烧焊的整个过程分解成调用多个 Sub 程序的话，可以用 VBS 这样写：

```
Sub 戴上手套
…
End Sub
Sub 戴上安全护帽
…
End Sub
Sub 准备好烧焊设备
…
End Sub
Sub 拿起烧焊枪指向烧焊点
…
End Sub
```

当然，“准备好烧焊设备”这个过程可能有些复杂，所以可以把里面的代码再细分以下，独立出更多的子过程来（Sub）。

如果烧焊工需要根据自己的心情来挑选不同颜色的手套的话，则需要在定义“戴上手套”这

个 Sub 的同时定义参数了，例如：
Sub 戴上手套（手套颜色）
…
End Sub

Tips：在代码中调用 Sub 可以有两种方式，一种是用 Call 语句来调用，例如：
Call　戴上手套（红颜色手套）
注意用括号把输入参数括起来，另外一种则直接使用，例如：
戴上手套　红颜色手套

Function

Sub 是不返回任何结果的，如果烧焊工在“戴上安全护帽”这个过程中没有很好地检查佩戴的效果的话，可能会造成很严重的后果，可能会在后续的烧焊过程中把眼睛弄瞎了！

为此，应该改成用 Function 封装“戴上安全护帽”这个过程，在调用这个过程时判断返回的值是否为“佩戴正确”，如果是，才继续后续操作。

```
Function 戴上安全护帽
   佩戴安全护帽
   检查佩戴情况
   IF 佩戴正确 Then
      戴上安全护帽 = True
   Else
      戴上安全护帽 = False
   End IF
End Function
```

注意，在 VBS 中，返回值是赋值给函数名的，例如：戴上安全护帽 = True

这样，就可以在调用时判断是否正确佩戴了安全护帽：

```
戴上手套（红颜色手套）
IF 戴上安全护帽 = True Then
   准备好烧焊设备
   拿起烧焊枪指向烧焊点
End IF
```

THE END