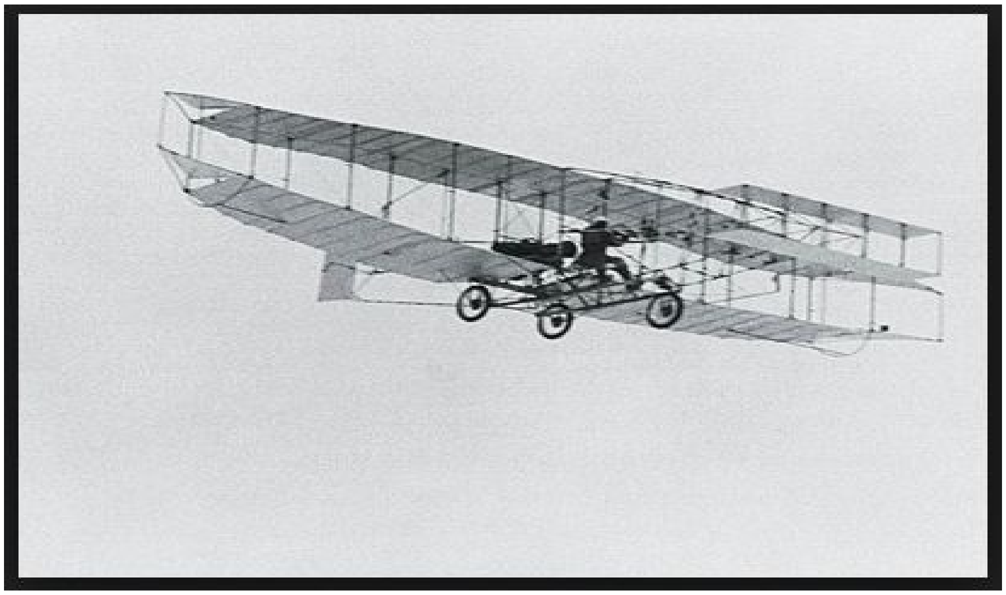
Chapter -6

Library and debugging in UFT



In this playhouse of infinite forms I have had my play, and here have I caught sight of him that is formless.

- Rabindranath Tagore (from *Gitanjali)*

Reusable code is essential for efficient automation suite. Function, sub procedure, classes, actions are the way to write reusable code. UFT support various means to enable you to write code in modular way. Modularity increase the maintainability and lead to minimal effort for maintenance.

Reusable functions in Library:

Reusable code should reside in library and UFT tests should call the reusable functions.

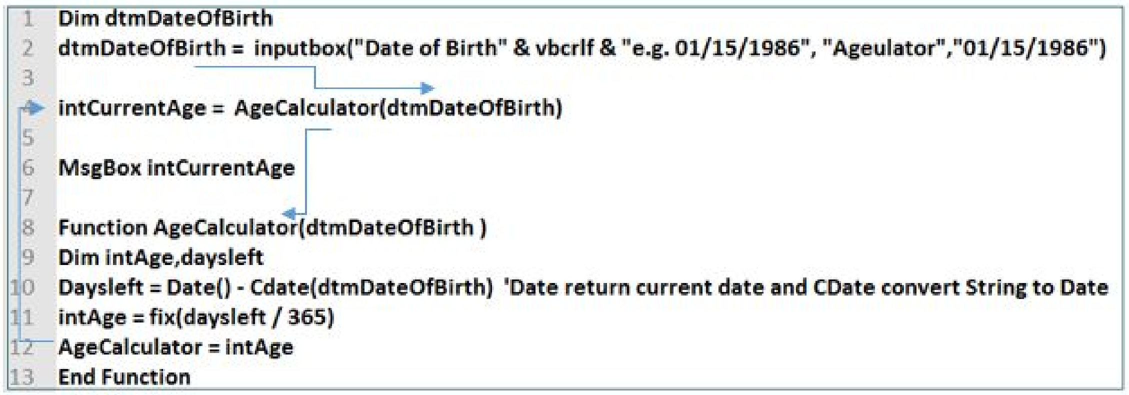
Function is a unit of logic to accomplish certain tasks and if required returns the result. Test engineers need to write reusable function and call them in test scripts.

Time for exercise TBD [Johnny bravo pic, laughter challenge type pic] ….

Exercise – Use external functions in Test Script

Step1: Write the VBScript Age Calculator function

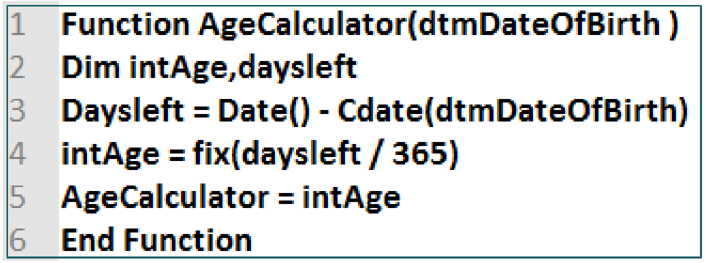
Open Notepad | Write code as given below | Save in "C:/Test/AgeCalculator.vbs" | double click on AgeCalculator.vbs to run VBScript program.



```
Dim dtmDateOfBirth
dtmDateOfBirth = inputbox("Date of Birth" & vbcrlf & "e.g. 01/15/1986", "Ageulator","01/15/1986")

intCurrentAge = AgeCalculator(dtmDateOfBirth)

MsgBox intCurrentAge

Function AgeCalculator(dtmDateOfBirth )
Dim intAge,daysleft
Daysleft = Date() - Cdate(dtmDateOfBirth) 'Date return current date and CDate convert String to Date
intAge = fix(daysleft / 365)
AgeCalculator = intAge
End Function
```

Step2: Modify VBScript program by removing lines to only include function but no statement to call function.

Open Notepad | Remove the lines except in Functions as given below | Save in "C:/Test/AgeCalculator.vbs" | double click on AgeCalculator.vbs to run VBScript program | Verify that you get no pop-up



```
Function AgeCalculator(dtmDateOfBirth )
Dim intAge,daysleft
Daysleft = Date() - Cdate(dtmDateOfBirth)
intAge = fix(daysleft / 365)
AgeCalculator = intAge
End Function
```

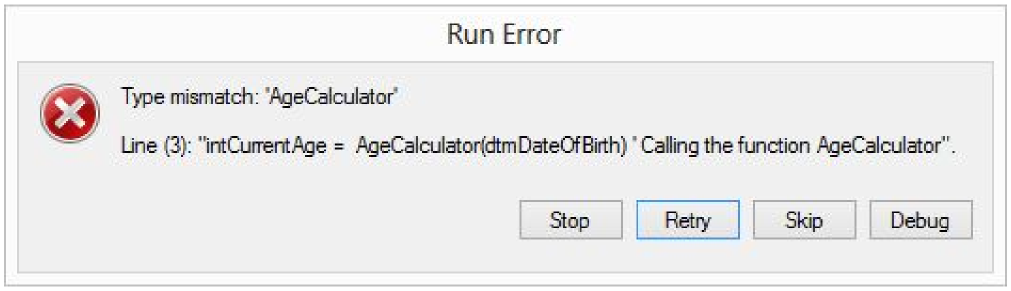
Step3: Record a Google Search Test

Open UFT | New | Test | Select "GUI Test" with name "AgeInGoogle" | Create | Record | Select first radio button in web | OK | Open Internet Explorer | Google.com | Search with "28 Year old UFT STAR" | Enter | Click on Image | Close Internet Browser | Stop | Save

Step4: Call and reuse Function in UFT Script

Navigate to UFT | Modify the Test "AgeInGoogle" by inserting following lines in the beginning of script| Run | Error

```
Dim dtmDateOfBirth
dtmDateOfBirth = inputbox("Date of Birth" & vbcrlf & "e.g. 01/15/1986", "Ageulator","01/15/1986")
intCurrentAge = AgeCalculator(dtmDateOfBirth)
MsgBox intCurrentAge
```


Run Error

Type mismatch: 'AgeCalculator'

Line (3): "intCurrentAge = AgeCalculator(dtmDateOfBirth) ' Calling the function AgeCalculator".

Step5: Link the Library to test - ExecuteFile

Navigate to UFT | Modify the Test AgeInGoogle by inserting ExecuteFile method at first lines in the beginning of script| Run

```
ExecuteFile "C:/Test/AgeCalculator.vbs"
```

Step6: Link the Library to test - **LoadFunctionLibrary**

Navigate to UFT | Remove ExecuteFile method from Script | Modify the Test AgeInGoogle by inserting LoadFunctionLibrary method at first lines in the beginning of script| Run

```
LoadFunctionLibrary "C:/Test/AgeCalculator.vbs"
```

Step7: Link the Library to test – Test Settings

Navigate to UFT | Open Test AgeInGoogle | Remove "LoadFunctionLibrary" or ExecuteFile method from Script | File | Settings… | Resources | Click "+" | Provide .vbs file path "C:/Test/AgeCalculator.vbs" | If pop-up appear click "No" | Check Syntax | Apply | OK | Save | run

Concept of Exercise – Use external functions in Test Script

In step-4 we have function and test but we have no link between test and .vbs function. This can be accomplish by mainly following ways

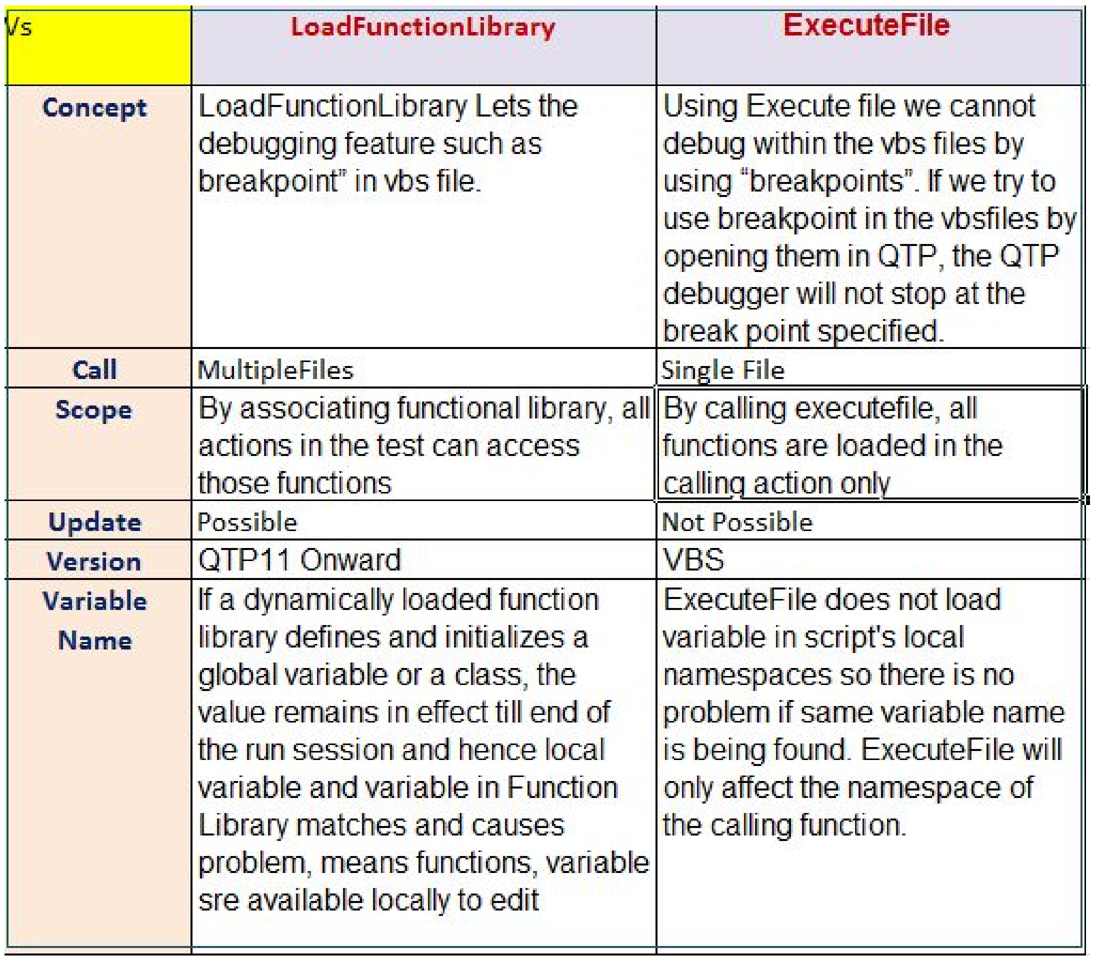
1. VBScript ExecuteFile method in code
2. UFT LoadFunctionLibrary method in code
3. Associate Library by UFT Test Settings by UFT GUI
4. Programmatically Associate Library by UFT Object model

ExecuteFile will keep file available during Action execution (not Test). ExecuteFile is VBScript concept. User cannot debug function by ExecuteFile.

LoadFunctionLibrary will keep file available during whole test execution (not only action, unlike ExecuteFile method). LoadFunctionLibrary is UFT concept and available from QTP v11.0 onward. User can debug function by LoadFunctionLibrary. User can also associate multiple files.

LoadFunctionLibrary "C:\Test\MyData1.vbs", "C:\Test\MyData2.vbs"

Associate Library by UFT Test Settings: This is manual way to associate functional library with test. If you notice the solution explorer listed out the associated library with test.



Vs	LoadFunctionLibrary	ExecuteFile
Concept	LoadFunctionLibrary Lets the debugging feature such as breakpoint" in vbs file.	Using Execute file we cannot debug within the vbs files by using "breakpoints". If we try to use breakpoint in the vbsfiles by opening them in QTP, the QTP debugger will not stop at the break point specified.
Call	MultipleFiles	Single File
Scope	By associating functional library, all actions in the test can access those functions	By calling executefile, all functions are loaded in the calling action only
Update	Possible	Not Possible
Version	QTP11 Onward	VBS
Variable Name	If a dynamically loaded function library defines and initializes a global variable or a class, the value remains in effect till end of the run session and hence local variable and variable in Function Library matches and causes problem, means functions, variable sre available locally to edit	ExecuteFile does not load variable in script's local namespaces so there is no problem if same variable name is being found. ExecuteFile will only affect the namespace of the calling function.

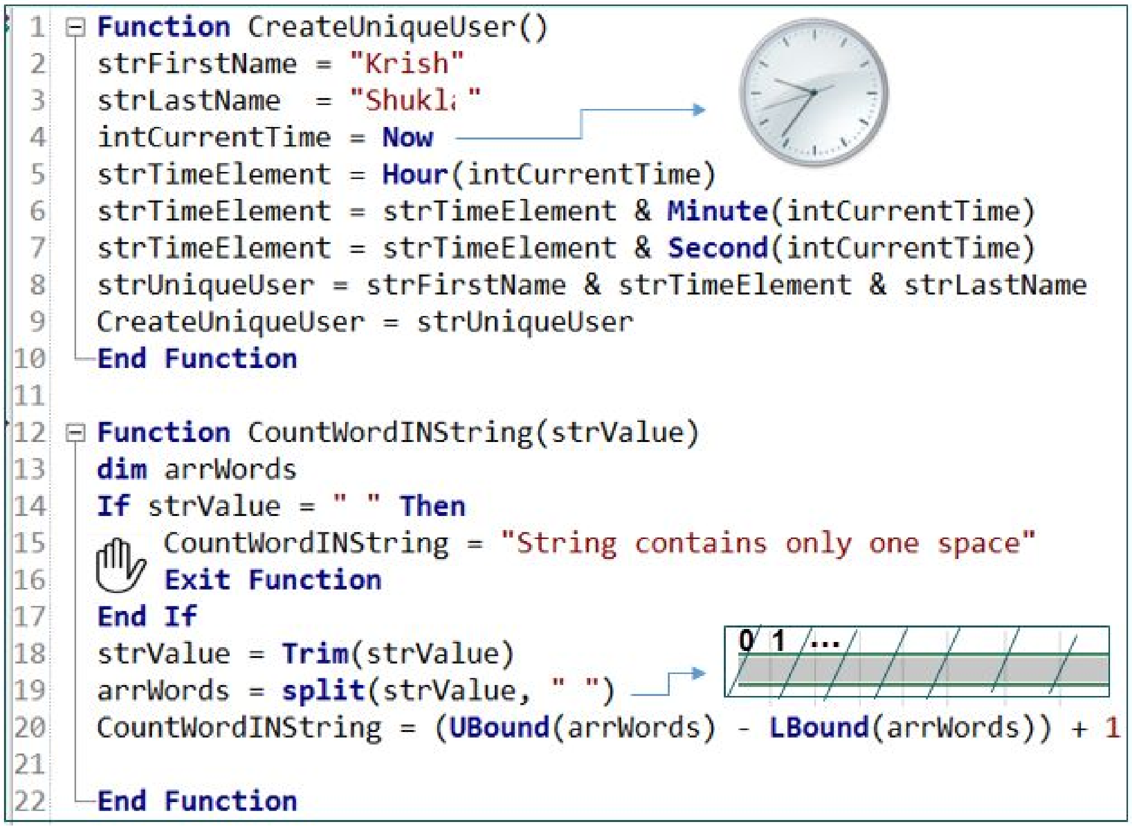
.QFL files

Programmer can write functions in .qfl file instead of .vbs files. Library of .qfl files work in similar fashion as .vbs file. We can put breakpoint in .qfl file. The .vbs library associated with LoadFunctionLibrary only enable programmers to debug functions.

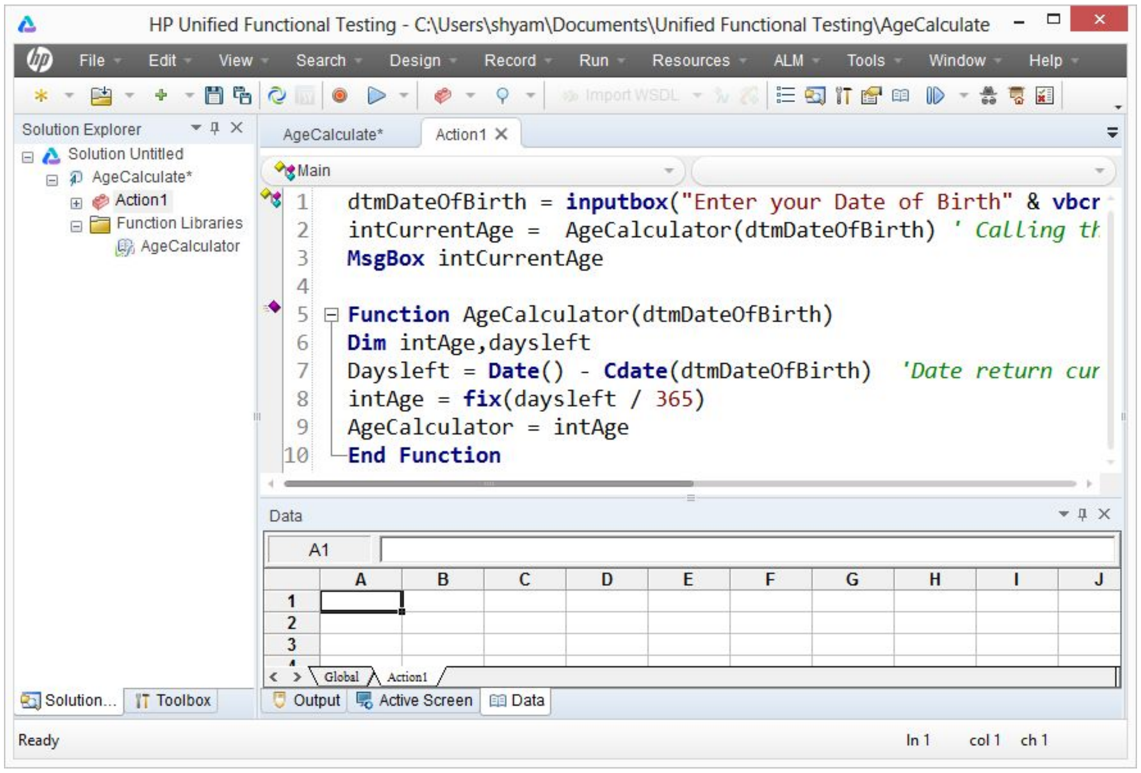
Advantages of Function Library:

User can assign many functions in a single .vbs file and call then multiple time in any test. Unlike action, functions are light weighted and do not have any OR or Datatable. Repetitive task e.g. database connection, error logging, recovery functions, windows process, task and service management functions etc. should be written in the function library.

Following are two example of generic functions to create unique user on each call and count the words in string. Programmer can build various reusable function to minimize the effort to reinvent the wheel every time by thoughtful designing functional library.



```
Function CreateUniqueUser()
strFirstName = "Krish"
strLastName  = "Shukla"
intCurrentTime = Now
strTimeElement = Hour(intCurrentTime)
strTimeElement = strTimeElement & Minute(intCurrentTime)
strTimeElement = strTimeElement & Second(intCurrentTime)
strUniqueUser = strFirstName & strTimeElement & strLastName
CreateUniqueUser = strUniqueUser
End Function

Function CountWordINString(strValue)
dim arrWords
If strValue = " " Then
    CountWordINString = "String contains only one space"
    Exit Function
End If
strValue = Trim(strValue)
arrWords = split(strValue, " ")
CountWordINString = (UBound(arrWords) - LBound(arrWords)) + 1

End Function
```



```
dtmDateOfBirth = inputbox("Enter your Date of Birth" & vbcr
intCurrentAge =  AgeCalculator(dtmDateOfBirth) ' Calling th

MsgBox intCurrentAge

Function AgeCalculator(dtmDateOfBirth)
Dim intAge,daysleft
Daysleft = Date() - Cdate(dtmDateOfBirth)  'Date return cur
intAge = fix(daysleft / 365)
AgeCalculator = intAge
End Function
```

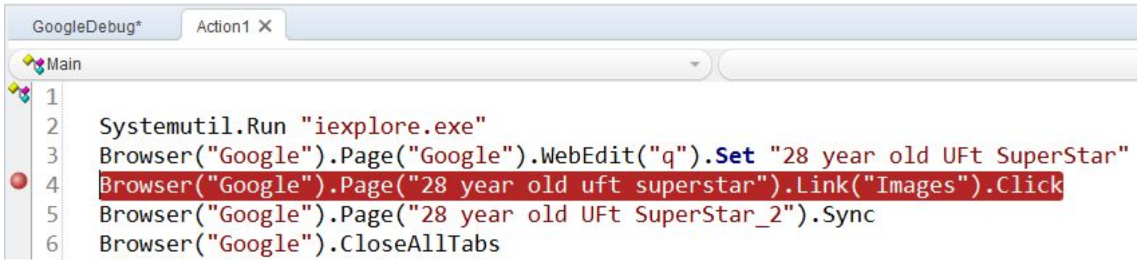
Debugging:

Debugging is a process of finding the problem in program by analysing it during execution.

Exercise 1: Understanding of debugging in UFT

Step1: Record a Google Search script

Open UFT | New | Test | Select "GUI Test" with name GoogleDebug | Create | Record | Select first radio button in web | OK | Open Internet Explorer | Google.com | Search with "28 Year old UFT STAR" | Enter | Click on Image | Close Internet Browser | Stop | Save



```

Systemutil.Run "iexplore.exe"
Browser("Google").Page("Google").WebEdit("q").Set "28 year old UFt SuperStar"
Browser("Google").Page("28 year old uft superstar").Link("Images").Click
Browser("Google").Page("28 year old UFt SuperStar_2").Sync
Browser("Google").CloseAllTabs
```

Step 2: Insert Breakpoint. Breakpoint will held the script execution at the specific point. User can step-by-step debug the script or run the script again by pressing F5-function key.

Navigate to GoogleDebug test in UFT | Mouse cursor at following link | Run | Insert/Remove break point | Run test | Press F10 function Key | again Press F10 Function Key | press F5 function Key

```
Browser("Google").Page("28 year old uft superstar").Link("Images").Click
```

Step 3: Associate function with script

Navigate to UFT | Open Test GoogleDebug | Remove LoadFunctionLibrary or ExecuteFile method from Script if appear in script | File | Settings… | Resources | Click "+" | Provide .vbs file path "C:/Test/AgeCalculator.vbs" | If pop-up appear click "No" | Check Syntax | Apply | OK | Save

Step 4: Call function in Test Script.

Navigate to GoogleDebug   in UFT |Modify test with following lines in the beginning of the script | Run

```
Dim dtmDateOfBirth
dtmDateOfBirth = inputbox("Date of Birth" & vbcrlf & "e.g. 01/15/1986", "Ageulator","01/15/1986")
intCurrentAge = AgeCalculator(dtmDateOfBirth)
MsgBox intCurrentAge
```

Step 4: Step-In to function in Test Script using F11

Navigate to GoogleDebug   in UFT |mouse cursor on following line | Press F9 | Run | Press F11

```
intCurrentAge = AgeCalculator(dtmDateOfBirth)
```

Step 5: Step-Over to function in Test Script using F10

Navigate to GoogleDebug   in UFT |mouse cursor on following line | Press F9 if breakpoint not inserted| View | Debug | LocalVariable | Run | Press F10

```
intCurrentAge =  AgeCalculator(dtmDateOfBirth)
```

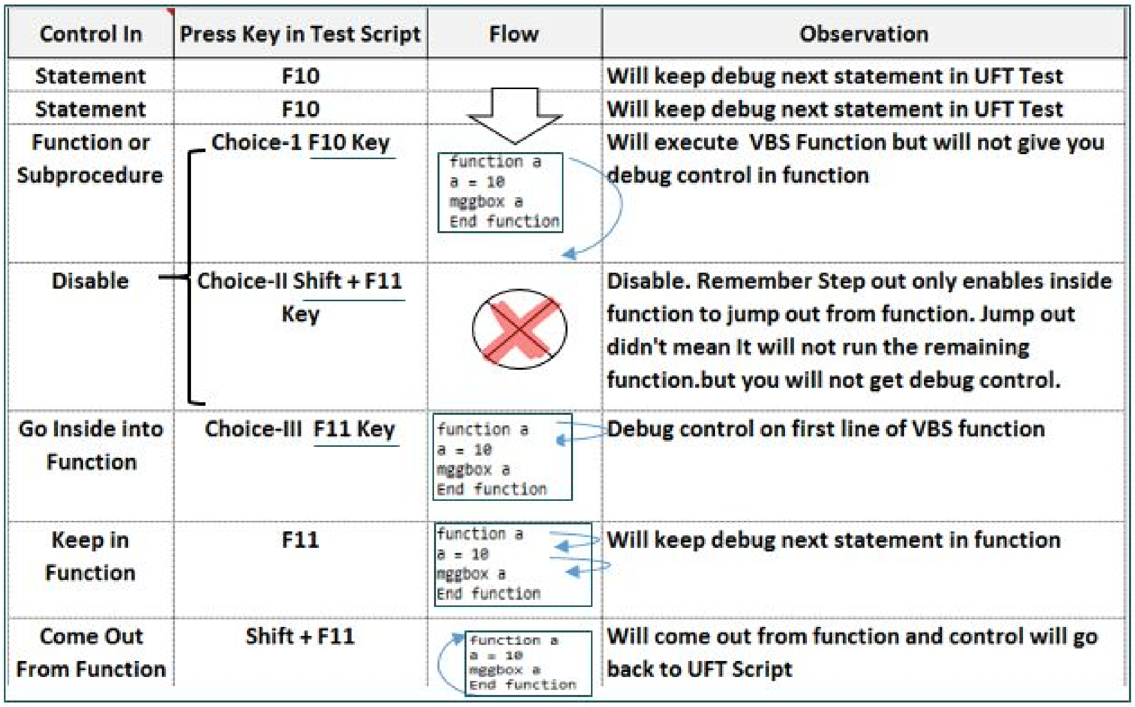
Step 6: Step-Out to function in Test Script using F10

Navigate to GoogleDebug   in UFT |mouse cursor on following line | Press F9 if breakpoint not inserted | Run | Press F11 | Now you are in function | press F11 again | Verify that F11 keeps you in function | Press Shift + F11 instead of F11 | Verify control jump Step-out from function | Press F10 | Run | Save

```
intCurrentAge =  AgeCalculator(dtmDateOfBirth)
```

Concept of Debug Script - Step-In (F11), Step-Over (F10) and Step-Out (Shift+F11)

Test Engineer can debug test script and function by using Step-In (F11), Step-Over (F10) and Step-Out (Shift+F11) function keys. Step-Over keep executing test script. If any function encounter during script, Step-Over just execute that function without giving any debug-control to user in function. Step-In is just opposite of that and it transfer debug control in to the function. User can debug the steps in function with F11 key. User can come out from function by pressing Step-Over "Shift+F11" keys. It does not mean that function will not execute after pressing for Step-Over but it will not give any debug control to user in function and come back to test script after finishing the function.



Control In	Press Key in Test Script	Flow	Observation
Statement	F10		Will keep debug next statement in UFT Test
Statement	F10		Will keep debug next statement in UFT Test
Function or Subprocedure	Choice-1 F10 Key	function a a = 10 msgbox a End function	Will execute VBS Function but will not give you debug control in function
Disable	Choice-II Shift + F11 Key		Disable. Remember Step out only enables inside function to jump out from function. Jump out didn't mean It will not run the remaining function.but you will not get debug control.
Go Inside into Function	Choice-III F11 Key	function a a = 10 msgbox a End function	Debug control on first line of VBS function
Keep in Function	F11	function a a = 10 msgbox a End function	Will keep debug next statement in function
Come Out From Function	Shift + F11	function a a = 10 msgbox a End function	Will come out from function and control will go back to UFT Script

Watching the Variable

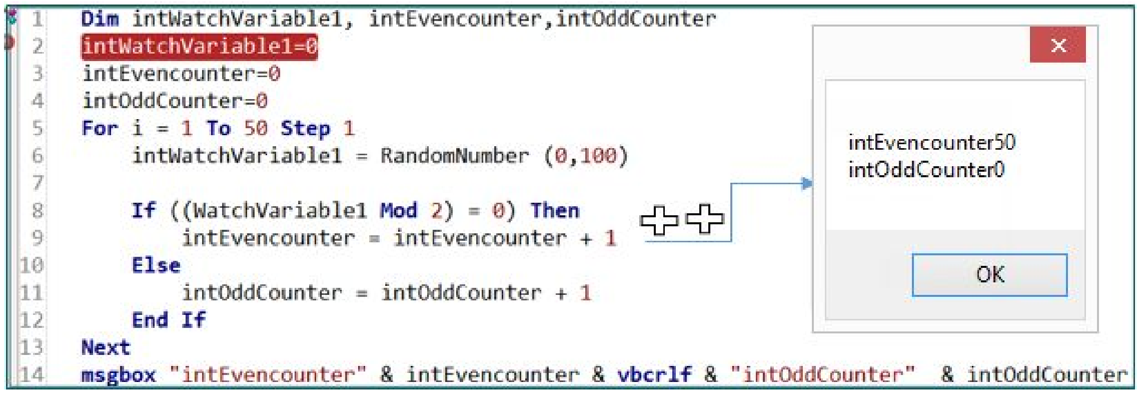
Suppose we need to track variables of interest. In that case, we can put them on watch to see their activities and any change during script execution

Exercise – Watch variable values during run-time.

Step 1: Create a test with logical error

Open UFT | New | Test | Select "GUI Test" with name WatchTheAge | Create | Write code as given below with breakpoint | Save | Run

This program checking the distribution of even and odd occurrence by the random number generated by using RandomNumber function.



```
Dim intWatchVariable1, intEvencounter,intOddCounter
intWatchVariable1=0
intEvencounter=0
intOddCounter=0
For i = 1 To 50 Step 1
    intWatchVariable1 = RandomNumber (0,100)

    If ((WatchVariable1 Mod 2) = 0) Then
        intEvencounter = intEvencounter + 1
    Else
        intOddCounter = intOddCounter + 1
    End If
Next
msgbox "intEvencounter" & intEvencounter & vbcrlf & "intOddCounter"  & intOddCounter
```

```
intEvencounter50
intOddCounter0
```

The following output is only giving integer value. Is RandomNumber function is so predictable? Something is wrong with code. We can put watch on intWatchVariable1 variable.

Step 2: Rectify error with watch on variable

Navigate to WatchTheAge test in UFT |Press F9 on first line in script | F5 | View | Debug | Watch | Click "+" | intWatchVariable1 | OK | Verify expression in Watch | F10 | Verify the following line never go in else branch | Debug all program

```
If ((WatchVariable1 Mod 2) = 0) Then
```

We were watching variable intWatchVariable1 but if condition using variable WatchVariable1. We need to rectify error by replacing above line with following line

```
If ((intWatchVariable1 Mod 2) = 0) Then
```

Step3: Run the test

Navigate to WatchTheAge test in UFT | Run

A Note on Debugging

Debugging is a process to rectify errors in code. MsgBox , print, InputBox, breakpoints , watch , on error resume next, err object, boolean flags, wait, comments, VBScript verification and conversion functions should employ together to rectify any error in code.

Maintenance and Update Run Mode:

Exercise - Maintenance run mode in UFT

STEP1: Create a Test

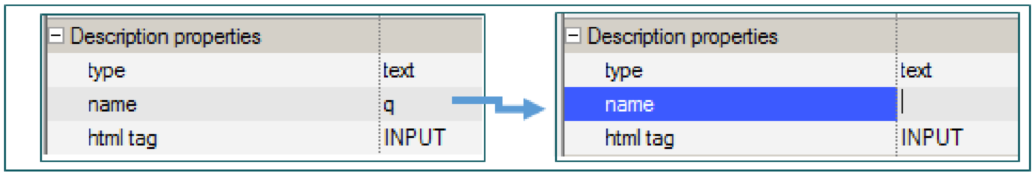
☝ *Navigate to UFT | New | Test | GUI Test | "MaintenanceRunModeTest" | Click Create | Record | Select first radio button in web | http://Google.co.uk | OK | Search with "krishna wikipedia" | Google Search button | Click on "Krishna - Wikipedia, the free encyclopedia" link | Close Browser | Save | Stop recording | Save*

STEP2: Run test

☝ *Navigate to "MaintenanceRunModeTest" test in UFT | F5 key | View | Last Run results | ALT + V + X | Close*

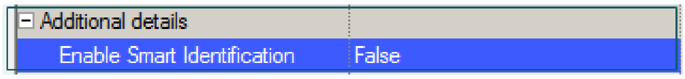
STEP3: Intentionally induce error in Object

☝ *Navigate to "MaintenanceRunModeTest" test in UFT | Resources | Object Repository | go to "q" object | remove name property value "q" | Set "Enable Smart Identification" to false | Close | Save*



Description properties	
type	text
name	q
html tag	INPUT

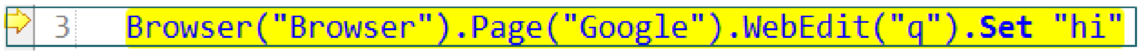
Description properties	
type	text
name	
html tag	INPUT



Additional details	
Enable Smart Identification	False

STEP4: Run test

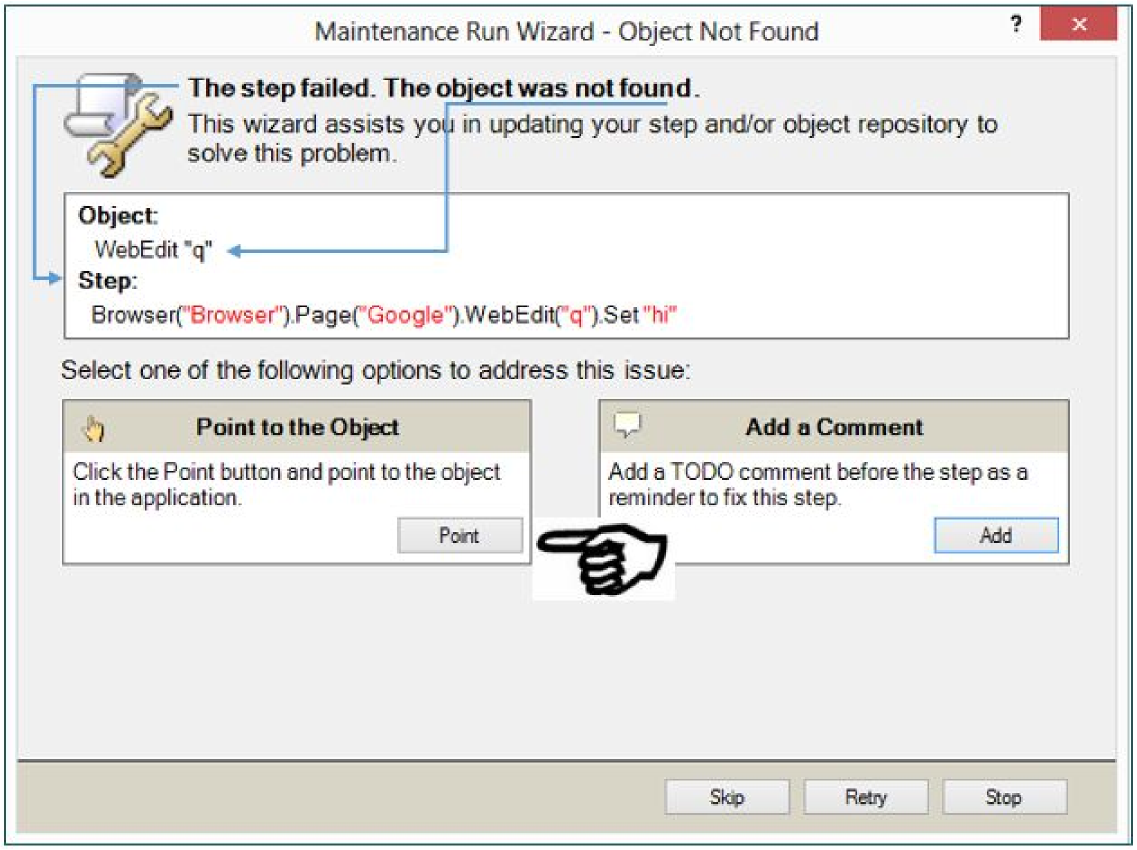
☝ *Navigate to "MaintenanceRunModeTest" test in UFT | Run | on error click Debug | Stop*



```
Browser("Browser").Page("Google").WebEdit("q").Set "hi"
```

STEP5: Run test in Maintenance Run Mode

☝ *Navigate to "MaintenanceRunModeTest" test in UFT | Run Menu | Maintenance Run Mode | Run | Click on Point | Click on Google search box | OK | OK | Finish | Resources | Object Repository | go to "q" object | verify name property of "q" object | Close | Save*


Maintenance Run Wizard - Object Not Found

The step failed. The object was not found.

This wizard assists you in updating your step and/or object repository to solve this problem.

Object:

WebEdit "q"

Step:

```
Browser("Browser").Page("Google").WebEdit("q").Set "hi"
```

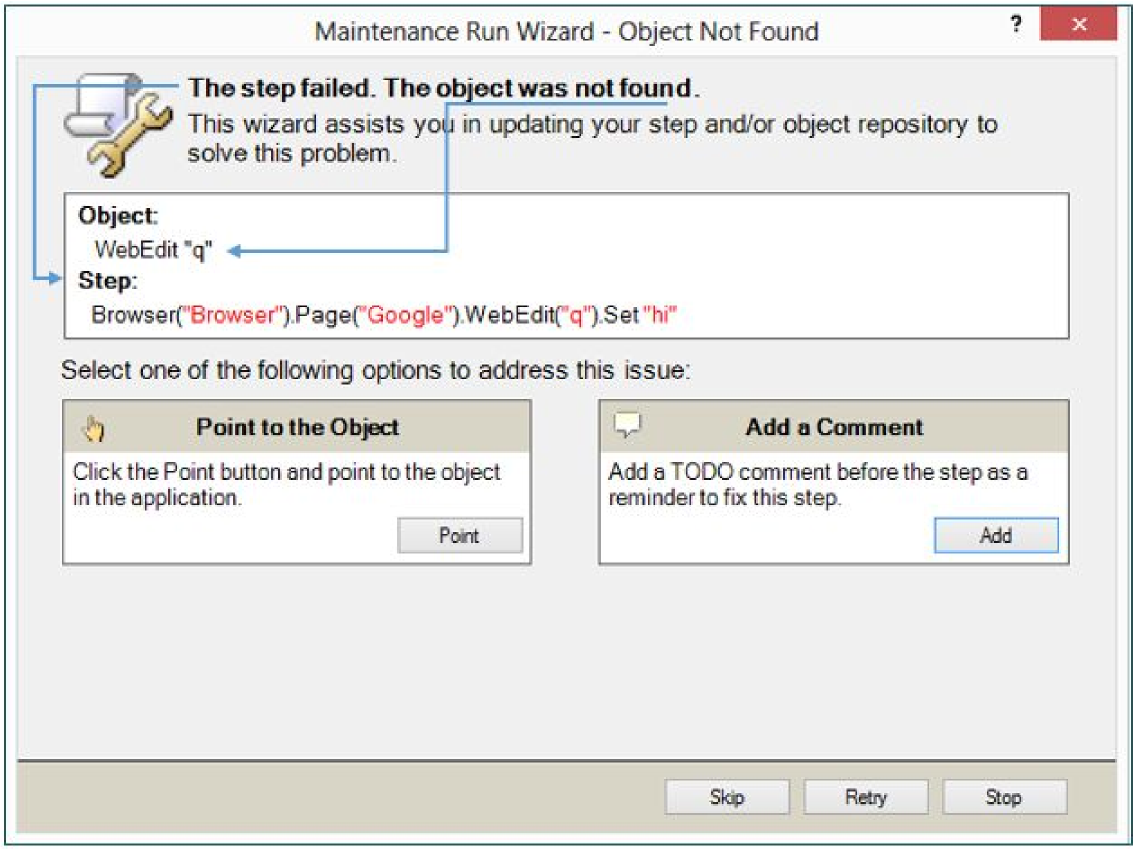
Select one of the following options to address this issue:

Point to the Object

Click the Point button and point to the object in the application.

Add a Comment

Add a TODO comment before the step as a reminder to fix this step.



Maintenance Run Wizard - Object Not Found

The step failed. The object was not found.

This wizard assists you in updating your step and/or object repository to solve this problem.

Object:

WebEdit "q"

Step:

Browser("Browser").Page("Google").WebEdit("q").Set "hi"

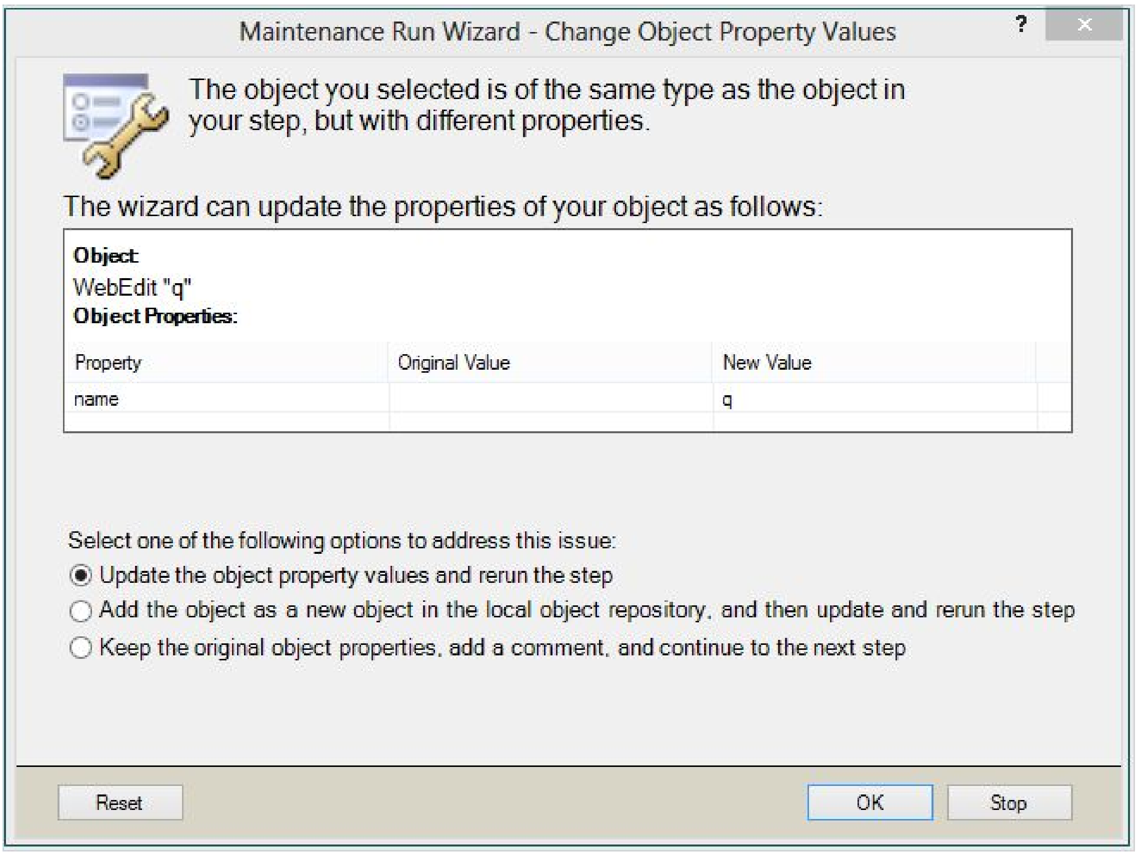
Select one of the following options to address this issue:

Point to the Object

Click the Point button and point to the object in the application.

Add a Comment

Add a TODO comment before the step as a reminder to fix this step.



Maintenance Run Wizard - Change Object Property Values

The object you selected is of the same type as the object in your step, but with different properties.

The wizard can update the properties of your object as follows:

Object:
WebEdit "q"
Object Properties:

Property	Original Value	New Value
name		q

Select one of the following options to address this issue:

- ● Update the object property values and rerun the step
- ○ Add the object as a new object in the local object repository, and then update and rerun the step
- ○ Keep the original object properties, add a comment, and continue to the next step

STEP6: Run test in Normal Mode

Navigate to "MaintenanceRunModeTest" test in UFT | Run by F5 key

Exercise - Update run mode in UFT

STEP1: Create a Test

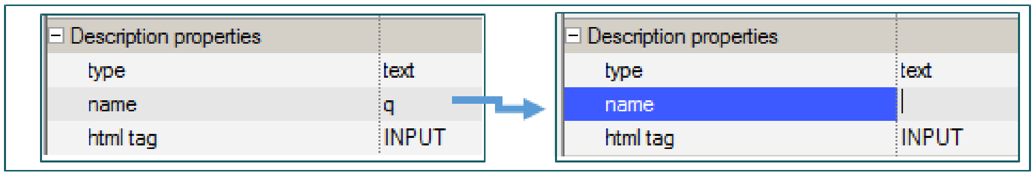
Navigate to UFT | New | Test | GUI Test | "UpdateRunModeTest" | Click Create | Record | Select first radio button in web | http://Google.co.uk | OK | Search with "krishna wikipedia" | Google Search button | Click on "Krishna - Wikipedia, the free encyclopedia" link | Close Browser | Save | Stop recording | Save

STEP2: Run test

Navigate to "UpdateRunModeTest" test in UFT | F5 key | View | Last Run results | ALT + V + X | Close

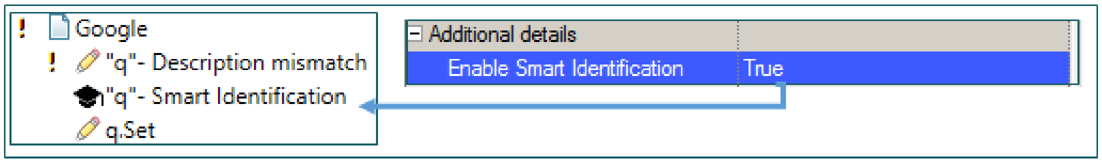
STEP3: Intentionally induce error in Object

Navigate to "UpdateRunModeTest" test in UFT | Resources | Object Repository | go to "q" object | remove name property value "q" | Set "Enable Smart Identification" to true | Close | Save | Run | View | Last Run Result | ALT + V + X



Description properties	
type	text
name	q
html tag	INPUT

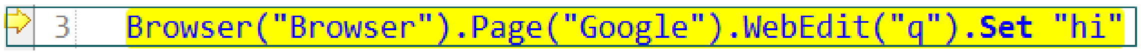
Description properties	
type	text
name	
html tag	INPUT



Additional details	
Enable Smart Identification	True

STEP4: Run test

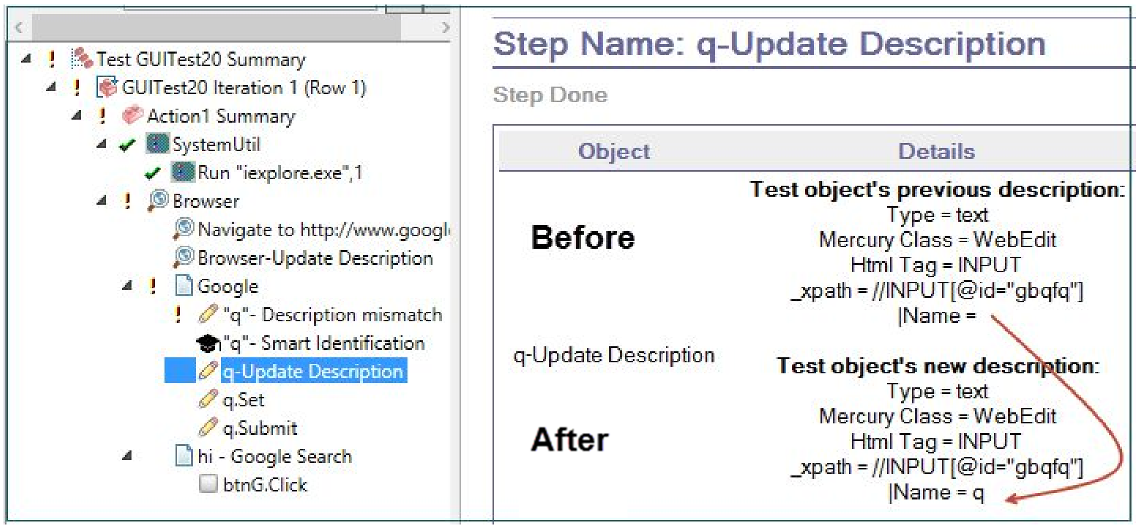
Navigate to "UpdateRunModeTest" test in UFT | Run | on error click Debug | Stop



```
Browser("Browser").Page("Google").WebEdit("q").Set "hi"
```

STEP5: Run test in Update Run Mode

Navigate to "UpdateRunModeTest" test in UFT | Run Menu | Update Run Mode | Run | Save | View | Last Run Result | ALT + V + X | Resources | Object Repository | go to "q" object | verify name property of "q" object | Close | Save



Step Name: q-Update Description

Step Done

Object		Details

Before	q-Update Description	Test object's previous description:
		Type = text
		Mercury Class = WebEdit
		Html Tag = INPUT
		_xpath = //INPUT[@id="gbqfq"]
		|Name =

After		Test object's new description:
		Type = text
		Mercury Class = WebEdit
		Html Tag = INPUT
		_xpath = //INPUT[@id="gbqfq"]
		|Name = q

Concept of Exercise - Maintenance run mode in UFT and Exercise – Update run mode in UFT

Quick test professional test run options for maintenance & updating the test object properties are two ways and here we provided what’s is the basic difference for better understanding.

Maintenance Mode :
If QTP doesn't recognize the test object by means of using defined identification properties, tool will pop up message with expected and actual properties giving an option to select the new properties, update and save the current test.

When object has changed in application and QTP is able to recognize the object by smart identification (or any other way) then Maintenance Run mode provides a dialog displaying the changed properties. Here user can update or keep the same original object property.

When object has changed and QTP is not able to recognize the object then Maintenance Run Mode asks user find the object in application and update one/many/all properties associated with that object.

Update Mode :
QTP would recognize the test object by non-identified or stored object properties, tool would not pop up message with stored and actual properties giving an option to select the new properties, update and save the current test. Rather it would update on self-methods and save's the current test. In Update Run mode object properties are updated directly without providing any alert dialog.

Update Run mode updates all properties of objects and checkpoints in case of failures. So if a object gets recognized using Smart identification then correct properties are updated in the OR. In case expected and actual values of checkpoints are different then they are updated as per the actuals.

Maintenance run mode is used when you want to update objects which are failing during execution. So if the object cannot be recognized at all with current settings instead of giving an error QTP will give a dialog to point the object to a new object if required.