***UFT Assignment Questions***

### **1. Introduction to UFT:**

* **Q1:** What is UFT (Unified Functional Testing)? How is it different from other test automation tools like Selenium or QTP?

**Answer**: **UFT (Unified Functional Testing)** is a commercial test automation tool developed by Micro Focus (formerly by HP). It is used for **functional and regression testing** of software applications. UFT supports a wide range of platforms, including desktop, web, mobile, and enterprise applications.

It uses **VBScript** as its scripting language and provides a **user-friendly interface** for both technical and non-technical users. UFT combines GUI testing and API testing into a single platform, enabling end-to-end automated testing.

**Differences Between UFT, Selenium, and QTP:**

| **Feature** | **UFT** | **Selenium** | **QTP** |
| --- | --- | --- | --- |
| **Developer** | Micro Focus | Open-source community | HP (now merged into UFT) |
| **License** | Commercial (paid) | Open-source (free) | Commercial (merged into UFT) |
| **Supported Applications** | Web, desktop, mobile, API | Web only | Web and desktop |
| **Scripting Language** | VBScript | Java, Python, C#, JavaScript, etc. | VBScript |
| **Ease of Use** | User-friendly with built-in IDE | Requires programming knowledge | Script-based interface |
| **Integration** | Built-in support for ALM, LoadRunner | Requires setup for CI tools | Limited integration (now outdated) |
| **Test Types** | Functional, regression, API testing | Primarily functional and regression | Functional and regression testing |

* **Q2:** List the key features of UFT. Explain how it supports functional, regression, and GUI testing.

**Answer**: **Key Features of UFT (Unified Functional Testing)**

**1. Cross-Platform and Multi-Browser Support**

* Tests web, desktop, mobile, and enterprise apps across different browsers and operating systems.

**2. GUI Testing**

* Captures and replays user actions on the graphical interface (buttons, menus, text fields).
* Uses Object Repository to recognize UI elements consistently.

**3. API Testing**

* Allows testing of service layers (SOAP, REST APIs) without a GUI.
* Supports end-to-end testing by combining API and GUI tests.

**4. Keyword-Driven Framework**

* Uses a table-based format for test steps, making test creation easier for non-programmers.

**5. Integration with ALM and CI Tools**

* Integrates with Micro Focus ALM (Application Lifecycle Management) and CI/CD tools like Jenkins, enabling test management and automation in DevOps pipelines.

**6. Smart Object Recognition**

* Uses AI and image-based object recognition to identify dynamic or complex UI elements reliably.

**7. Reusability and Modularity**

* Supports reusable functions, actions, and libraries, making test maintenance easier.

**8. VBScript Scripting**

* Allows test customization using VBScript for complex logic and validations.

**9. Reporting and Logging**

* Automatically generates detailed test reports with screenshots, pass/fail status, and error messages.

**How UFT Supports Functional, Regression, and GUI Testing**

**Functional Testing**

* UFT verifies that each function of the application works as intended by automating test cases based on user requirements.
* Example: Logging in, placing an order, or submitting a form.

**Regression Testing**

* After changes or bug fixes, UFT can rerun previously created automated test scripts to ensure that existing features still function correctly.
* Saves time by executing large test suites without manual effort.

**GUI Testing**

* UFT simulates user interactions with the application’s interface by recording mouse clicks, keystrokes, and UI behavior.
* It uses an object-based approach to identify and interact with elements like buttons, checkboxes, and dropdowns.
* **Q3:** What are the different types of objects that UFT can recognize? Give examples of each type.

**Answer**: **Types of Objects Recognized by UFT (Unified Functional Testing)**

UFT identifies and interacts with **objects** in the application's user interface using its **Object Recognition Mechanism**. These objects are classified into several types based on their nature and behavior within the application.

**1. Standard GUI Objects**

These are common interface elements found in most applications.

* **Examples:**
  + Button – A clickable control (e.g., "Submit" or "OK" button)
  + Edit – Text input fields (e.g., login fields)
  + CheckBox – A toggle input (e.g., "Remember me" checkbox)
  + RadioButton – A selectable option in a group
  + ListBox – A box displaying a list of items
  + ComboBox – A dropdown list allowing selection

**2. Web Objects**

Used when testing web applications.

* **Examples:**
  + WebEdit – HTML input text field
  + WebButton – HTML button element
  + WebLink – Hyperlink on a webpage
  + WebTable – Table structure in a webpage
  + WebCheckBox – Checkbox on a form
  + WebRadioGroup – Group of radio buttons

**3. ActiveX Objects**

Used in applications developed with ActiveX controls.

* **Examples:**
  + ActiveXListBox – List box from an ActiveX control
  + ActiveXComboBox – Dropdown created with ActiveX

**4. Windows Objects**

Used for desktop or Windows-based applications.

* **Examples:**
  + WinButton – Button in a Windows application
  + WinEdit – Text box in a Windows form
  + WinMenu – Menu bar or menu item
  + WinList – List control
  + WinTab – Tab control with multiple pages

**5. SAP, Oracle, Java, .NET, and Custom Objects**

UFT supports specialized add-ins for enterprise applications, enabling recognition of complex UI components.

* **SAP Examples:** SAPTable, SAPEdit, SAPButton
* **Java Examples:** JavaEdit, JavaButton, JavaList
* **Oracle Examples:** OracleList, OracleButton
* **.NET Examples:** SwfButton, SwfEdit, SwfTable

**6. Custom or Virtual Objects**

Used when UFT cannot recognize an object by default.

* **Examples:**
  + VirtualButton – A custom UI element trained to behave like a button
  + VirtualObject – Any UI region that is mapped to a standard object type

### **2. Creating and Running a Basic Test in UFT:**

* **Q4:** Create a simple test in UFT to open the Notepad application, type a text message, and save the file. Include the steps to record and run the test.

**Answer**: '========================================

' Verify text on a web page using Text Checkpoint

'========================================

' Step 1: Launch browser and navigate to a page

SystemUtil.Run "chrome.exe", "https://example.com"

' Step 2: Wait for page to load

Browser("micClass:=Browser").Sync

' Step 3: Add a Text Checkpoint dynamically using .CheckProperty

Dim expectedText

expectedText = "Welcome, Ananya!"

' Option A: Use WebElement to check text

If Browser("micClass:=Browser").Page("micClass:=Page").WebElement("innertext:=" & expectedText).Exist(5) Then

Reporter.ReportEvent micPass, "Text Checkpoint", "Text '" & expectedText & "' is present on the page."

Else

Reporter.ReportEvent micFail, "Text Checkpoint", "Text '" & expectedText & "' is NOT found on the page."

* **Q5:** Write a simple UFT script to open a web browser, navigate to a website (e.g., www.google.com), and perform a Google search.

**Answer**: 4 '=============================

' Google Search using UFT

'=============================

' Step 1: Launch the browser and navigate to Google

SystemUtil.Run "chrome.exe", "https://www.google.com"

' Step 2: Wait for the browser to load completely

Browser("micClass:=Browser").Sync

' Step 3: Enter search term into the input box

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("name:=q").Set "UFT Automation"

' Optional wait (let suggestions load)

Wait 2

' Step 4: Click the Google Search button

' Since Google dynamically updates buttons, we use index or outertext property

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("name:=btnK", "index:=1").Click

' Step 5: Wait and Report

Wait 3

Reporter.ReportEvent micPass, "Search Executed", "Searched for 'UFT Automation' on Google"

### **3. Object Repository and Object Identification:**

* **Q6:** What is an object repository in UFT? Explain the difference between "Local Object Repository" and "Shared Object Repository."

**Answer**: An **Object Repository (OR)** in UFT (Unified Functional Testing) is a storage location where **UFT stores information about the objects** (UI elements) it interacts with during automated testing.  
It maps objects from the application under test to logical names used in the test scripts, allowing UFT to identify and interact with those objects reliably during execution.

Each object in the repository contains:

* Logical Name (used in script)
* Class (type of object, like Button or Edit box)
* Properties (used to uniquely identify the object)

**Types of Object Repositories in UFT**

**1. Local Object Repository (LOR)**

* Automatically created and stored **within the individual test**.
* Only accessible to that specific test.
* Objects are added when using **recording** or manually.
* Best for **simple tests** or one-time use.

**Example Use Case:**  
A test script that automates a login form and does not need object reuse across multiple tests.

**2. Shared Object Repository (SOR)**

* Stored **externally as a separate file** (.tsr format).
* Can be **linked to multiple test scripts**, promoting reusability.
* Managed and edited using the **Object Repository Manager**.
* Ideal for **large projects** with many tests that interact with common objects.

**Example Use Case:**  
Multiple test cases that all use the same home page or dashboard elements in a web application.

**Key Differences Between Local and Shared Object Repository**

| **Feature** | **Local Object Repository** | **Shared Object Repository** |
| --- | --- | --- |
| **Storage** | Inside the test | Separate .tsr file |
| **Reusability** | Only in that test | Across multiple tests |
| **Management Tool** | Object Repository window | Object Repository Manager |
| **Best Suited For** | Small, individual test cases | Large projects with common objects |
| **Ease of Maintenance** | Harder to maintain across tests | Easier to update and reuse objects |

* **Q7:** Explain the concept of "Object Identification" in UFT. How does UFT recognize objects on the application being tested?

**Answer**: **Object Identification** in UFT refers to the process of how UFT **recognizes and interacts with objects** (such as buttons, text fields, checkboxes) in the application under test.

UFT uses a set of **properties** and **object classes** to uniquely identify each object in the user interface so it can simulate user actions during test execution.

**How UFT Recognizes Objects**

1. **Object Class**
   * UFT first determines the **type or class** of the object (e.g., WebButton, WinEdit, WebLink).
   * Each class has a predefined set of identification properties.
2. **Mandatory Properties**
   * These are key properties required to uniquely identify the object.
   * Example: For a WebButton, UFT might use html tag, name, or type.
3. **Assistive Properties**
   * If the mandatory properties are not sufficient to distinguish the object, UFT uses additional assistive properties.
   * Example: index, innerText, html id.
4. **Ordinal Identifier**
   * When both mandatory and assistive properties are still not enough to identify the object, UFT uses positional information like:
     + Index (position in code)
     + Location (position on screen)

**Object Identification Mechanism Components**

* **Object Repository**: Stores logical names and properties of objects.
* **Object Spy**: A tool used to view the properties and values of any object on the screen.
* **Object Identification Settings**: Allows customization of how UFT recognizes different object classes.

**Example:**

If you are automating a login form with a "Submit" button, UFT might identify it as:

* Class: WebButton
* Name: Submit
* HTML Tag: INPUT
* Type: submit

**Smart Object Recognition (Optional Feature)**

UFT also has a **Smart Identification** feature. If UFT cannot find an object with the recorded properties during runtime, it uses a set of backup identification rules (smart identification) to locate it.

### **4. Checkpoints and Verification:**

* **Q10:** What are checkpoints in UFT? Write a script to add a "Text Checkpoint" to verify that a specific text appears on a web page.

**Answer**: **Checkpoints** in UFT (Unified Functional Testing) are verification points that compare the actual outcome of a test during execution with the expected result. They are used to **validate values, text, properties, database entries, and other aspects of the application** under test.

UFT provides several types of checkpoints, including:

* **Standard Checkpoint** – Verifies object properties
* **Text Checkpoint** – Verifies text displayed on the screen
* **Bitmap Checkpoint** – Compares screen images
* **Database Checkpoint** – Verifies database values
* **XML Checkpoint** – Validates XML data

**What is a Text Checkpoint?**

A **Text Checkpoint** checks that a specific **static text** (visible on the application or web page) appears exactly as expected during test execution.

**Sample Script to Add a Text Checkpoint in UFT (Web Page Example)**

'========================================

' Verify text on a web page using Text Checkpoint

'========================================

' Step 1: Launch browser and navigate to a page

SystemUtil.Run "chrome.exe", "https://example.com"

' Step 2: Wait for page to load

Browser("micClass:=Browser").Sync

' Step 3: Add a Text Checkpoint dynamically using .CheckProperty

Dim expectedText

expectedText = "Welcome, Ananya!"

' Option A: Use WebElement to check text

If Browser("micClass:=Browser").Page("micClass:=Page").WebElement("innertext:=" & expectedText).Exist(5) Then

Reporter.ReportEvent micPass, "Text Checkpoint", "Text '" & expectedText & "' is present on the page."

Else

Reporter.ReportEvent micFail, "Text Checkpoint", "Text '" & expectedText & "' is NOT found on the page."

* **Q11:** Explain the difference between "Standard Checkpoints" and "Database Checkpoints" in UFT. Give an example of when you would use each.

**Answer**: **Standard Checkpoint**

**Definition:**  
A Standard Checkpoint is used to verify the **properties** of objects in the application’s user interface during runtime.

**What It Checks:**

* Object properties such as text, value, enabled/disabled, visible/hidden, etc.

**When to Use:**  
Use when you want to **validate the appearance or behavior of UI elements** in your application.

**Example Use Case:**  
You want to verify that after a successful login, the “Welcome, User” message appears on the homepage.

vb

Copy code

Browser("LoginPage").Page("HomePage").WebElement("WelcomeMessage").Check CheckPoint("WelcomeMessage")

**2. Database Checkpoint**

**Definition:**  
A Database Checkpoint is used to verify the **contents of a database** accessed by the application.

**What It Checks:**

* Records, fields, and values returned by a SQL query
* Allows comparison between expected and actual data in a connected database

**When to Use:**  
Use when you need to **validate backend data** after performing an operation in the application.

**Example Use Case:**  
After submitting a registration form, you want to verify that the user's details were correctly inserted into the database.

vbscript

Copy code

' Database checkpoint inserted during test recording, linked to a SQL query

DbTable("UserDatabase").Check CheckPoint("UserRecordExists")

* **Q12:** How can you handle dynamic objects using UFT? Explain with an example of handling dynamic buttons that change text based on user interactions.

**Answer**: **Handling Dynamic Objects in UFT (Unified Functional Testing)**

In UFT, **dynamic objects** are UI elements whose properties (like name, id, text, etc.) change during runtime — often based on user interaction, data, or application state.  
To test these effectively, UFT provides multiple strategies to identify such objects **without relying on fixed property values**.

**Common Techniques to Handle Dynamic Objects**

**1. Descriptive Programming**

Instead of relying on the Object Repository, use scripting to describe the object based on stable or partial properties.

**2. Regular Expressions in Object Repository**

Use regular expressions for property values that change but follow a predictable pattern.

**3. Smart Identification**

Enable UFT’s smart identification feature to let it automatically find the closest match when an object’s properties don’t match exactly.

**Example: Handling Dynamic Button Text Using Descriptive Programming**

**Scenario:** UFT Script Using Descriptive Programming

' Step 1: Open the browser and navigate to the page

SystemUtil.Run "chrome.exe", "https://example.com/dynamic-button"

' Step 2: Create a description object for the dynamic button

Dim oButton

Set oButton = Description.Create

oButton("micClass").Value = "WebButton"

oButton("innertext").Value = "Click Me|Clicked!" ' Regex to match either text

' Step 3: Click the dynamic button

Browser("micClass:=Browser").Page("micClass:=Page").WebButton(oButton).Click

' Step 4: Verify button was clicked (optional)

If Browser("micClass:=Browser").Page("micClass:=Page").WebButton("innertext:=Clicked!").Exist(5) Then

Reporter.ReportEvent micPass, "Dynamic Button", "Button clicked and text changed successfully."

Else

Reporter.ReportEvent micFail, "Dynamic Button", "Button text did not change as expected."

End If

**Q13:** What is parameterization in UFT? Why is it important for automating tests? Demonstrate how to parameterize a test using input data (e.g., user credentials for a login page).

**Answer**: **Parameterization** in UFT (Unified Functional Testing) is the process of replacing hard-coded values in a test script with **variable parameters**. This allows the same test to run multiple times with different sets of input data.

**Why is Parameterization Important?**

* **Increases test coverage**: Validates functionality using multiple data inputs.
* **Enhances flexibility**: Easily change test inputs without modifying the script.
* **Supports data-driven testing**: Automates scenarios like form submissions, logins, and transactions using different user data.

**Types of Parameters in UFT**

1. **Test/Action Parameters** – Pass values between tests or actions.
2. **Data Table Parameters** – Retrieve data from the built-in Data Table (similar to Excel).
3. **Environment Variables** – Use global or user-defined values across tests.
4. **Random Numbers / Custom Code** – Generate dynamic data during runtime.

**Example: Parameterize Login Credentials Using Data Table**

**Scenario:**

'===============================

' Parameterized Login Script

'===============================

' Open the login page

SystemUtil.Run "chrome.exe", "https://example.com/login" ' Replace with actual URL

' Loop through all rows in the DataTable

rowCount = DataTable.GetRowCount

For i = 1 To rowCount

DataTable.SetCurrentRow(i)

' Get values from parameterized columns

username = DataTable.Value("Username", dtGlobalSheet)

password = DataTable.Value("Password", dtGlobalSheet)

' Fill login form

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=user").Set username

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=pass").Set password

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("html id:=loginBtn").Click

' Log result

Reporter.ReportEvent micDone, "Login", "Tested with username: " & username

* **Q14:** Create a test that accepts input parameters (e.g., username and password) from an Excel file and performs a login using that data.

**Answer:** '=============================================

' Data-Driven Login Test from Excel in UFT

'=============================================

' Load Excel file using DataTable.ImportExternal

DataTable.ImportSheet "C:\UFT\_Data\LoginData.xlsx", 1, "Global"

' Get total rows (excluding header)

Dim rowCount

rowCount = DataTable.GetRowCount

' Launch browser

SystemUtil.Run "chrome.exe", "https://example.com/login" ' Change to your actual login URL

' Loop through each row of data

For i = 1 To rowCount

DataTable.SetCurrentRow(i)

' Read username and password from Excel

user = DataTable.Value("Username", dtGlobalSheet)

pass = DataTable.Value("Password", dtGlobalSheet)

' Set credentials in form

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=user").Set user

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=pass").Set pass

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("html id:=loginBtn").Click

' Log step

Reporter.ReportEvent micPass, "Login Attempt", "Tried with Username: " & user & " and Password: " & pass

* **Q15:** What are the different types of parameters available in UFT (e.g., test, action, and data table parameters)? Explain their use with examples.

**Answer**: **Types of Parameters in UFT (Unified Functional Testing)**

UFT provides several types of parameters to support **data-driven testing**, **modular test design**, and **reusability**. Each type of parameter serves a specific purpose within the test or action flow.

**1. Test Parameters**

* **Scope:** Available to the entire test and can be passed from external sources like test batches or test calls.
* **Use Case:** Pass values like environment settings or login type to multiple actions within the test.

**Example:**

Suppose a test has a parameter named EnvironmentType.

vbscript

If TestArgs("EnvironmentType") = "Production" Then

' Navigate to production site

Else

' Navigate to test site

End If

To define:

* Go to **File > Settings > Parameters > Test Parameters**
* Add EnvironmentType with a default value

**2. Action Parameters**

* **Scope:** Specific to individual actions (reusable actions).
* **Use Case:** Pass input/output values between actions for better modularization.

**Example:**

Let’s say you create a reusable login action with two parameters: Username, Password.

Inside the action:

vbscript

Browser("LoginPage").Page("LoginPage").WebEdit("username").Set Parameter("Username")

Browser("LoginPage").Page("LoginPage").WebEdit("password").Set Parameter("Password")

Browser("LoginPage").Page("LoginPage").WebButton("Login").Click

To define:

* Right-click the action > **Action Properties > Parameters**
* Add input parameters (e.g., Username, Password)

When calling the action:

vbscript

Copy code

RunAction "LoginAction", oneIteration, "user1", "pass1"

**3. Data Table Parameters**

* **Scope:** Pulls input values from the test’s **Data Table** (Global or Action sheet).
* **Use Case:** Used for data-driven testing where test steps are repeated with different data sets.

**Example:**

vbscript

Browser("LoginPage").Page("LoginPage").WebEdit("username").Set DataTable("Username", dtGlobalSheet)

Browser("LoginPage").Page("LoginPage").WebEdit("password").Set DataTable("Password", dtGlobalSheet)

To define:

* Open the Data Table at the bottom
* Add columns: Username, Password
* Enter different rows of test data

UFT will automatically iterate through each row unless instructed otherwise.

**4. Environment Variables**

* **Scope:** Global variables, available throughout the test.
* **Types:** Built-in or user-defined
* **Use Case:** Store environment-specific data like URLs, global credentials, paths.

**Example:**

vbscript

SystemUtil.Run Environment("AppURL")

To define:

* Go to **File > Settings > Environment > User-Defined**
* Add or load variables from an .xml file

### **6. Actions and Function Libraries:**

* **Q16:** What is an action in UFT? How does it help in organizing your test scripts? Create an example of a reusable action for logging into a web application.

**Answer**: An **Action** in UFT (Unified Functional Testing) is a modular section of a test that contains its own set of steps, data, and objects. Actions help you organize, reuse, and maintain test scripts efficiently by dividing a large test into **logical, manageable parts**.

**Types of Actions in UFT**

| **Action Type** | **Description** |
| --- | --- |
| **Non-reusable Action** | Can be called only in the test in which it is defined. |
| **Reusable Action** | Can be called multiple times within the same or other tests. |
| **External Action** | A read-only version of a reusable action from another test. |

**How Actions Help in Organizing Test Scripts**

* **Modularity**: Break tests into smaller, manageable components.
* **Reusability**: Reuse login, logout, or navigation steps across tests.
* **Readability**: Each action focuses on a single task, improving clarity.
* **Maintainability**: Changes made to one action update all associated test calls.

**Example: Reusable Action for Logging into a Web Application**

A) Create the Action:

In your UFT Test, go to Keyword View or Action menu.

Click: Insert > Call to New Action

Name it: LoginAction

In Action Properties, set Reusable Action = True

B) Add This Code Inside the Action:

vbscript

Edit

' LoginAction - Reusable login action

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=user").Set "admin"

Browser("micClass:=Browser").Page("micClass:=Page").WebEdit("html id:=pass").Set "admin123"

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("html id:=loginBtn").Click

Reporter.ReportEvent micPass, "Login", "User successfully logged in."

C) Call This Action from Another Test:

In any other test:

Go to Insert > Call to Existing Action

Browse to the test where LoginAction is saved

Select it and it will appear in the test flow 16

* **Q17:** Explain the concept of "Function Libraries" in UFT. How do you create and associate a function library with your test?

**Answer**: **Function Library** in UFT (Unified Functional Testing) is a file that contains **user-defined functions and reusable code**, stored separately from the test script. These libraries help you modularize your automation framework by allowing code reuse across multiple tests and actions.

Function Libraries are typically saved with the extension .vbs (VBScript file) or .qfl (QuickTest Function Library).

**Benefits of Using Function Libraries**

* **Code Reusability**: Write once, use in many tests.
* **Maintainability**: Centralized updates to common functions.
* **Modularity**: Organize business logic or utility functions separately.
* **Scalability**: Supports large test automation frameworks.

**How to Create a Function Library in UFT**

1. **Create the File**:
   * Open UFT.
   * Go to **File > New > Function Library**.
   * A blank script editor opens.
   * Write your reusable functions.

**Example:**

vbscript

' Function to perform login

Function Login(username, password)

Browser("LoginPage").Page("LoginPage").WebEdit("username").Set username

Browser("LoginPage").Page("LoginPage").WebEdit("password").Set password

Browser("LoginPage").Page("LoginPage").WebButton("Login").Click

End Function

1. **Save the Library**:
   * Save the file as LoginLibrary.vbs or LoginLibrary.qfl.

**How to Associate a Function Library with a Test**

**Method 1: Using Test Settings**

* Go to **File > Settings > Resources** tab.
* Click **+ (Add)** to browse and add the .vbs or .qfl file.
* Click **OK**.

This makes the functions available in that test.

**Method 2: Using ExecuteFile in the Script**

You can also associate the library dynamically using:

vbscript

ExecuteFile "C:\UFT\Libraries\LoginLibrary.vbs"

**How to Use a Function from the Library in Your Test**

vbscript

' Call the login function with parameters

Login "admin@test.com", "admin123"

* **Q18:** Write a simple function in a UFT function library that accepts two numbers as inputs and returns their sum. Call this function from your test script.

**Answer**: '----------------------------------

' File: MathLibrary.vbs

' Description: Contains a simple addition function

'----------------------------------

Function AddNumbers(num1, num2)

AddNumbers = num1 + num2

End Function

### **7. Descriptive Programming:**

* **Q19:** What is Descriptive Programming in UFT, and when would you use it? Write a UFT script using descriptive programming to click a button on a webpage (e.g., a "Submit" button).

**Answer**: **Descriptive Programming (DP)** in UFT is a method that allows you to **interact with objects without storing them in the Object Repository**.  
Instead, object properties are described directly in the test script.

**When to Use Descriptive Programming**

Use DP when:

* The object is **not present** in the Object Repository.
* You want to **bypass** the repository for **dynamic objects** or on-the-fly identification.
* You're working with **data-driven** or **loop-based** test cases involving similar objects.
* You want to improve **maintainability** in large frameworks by separating object logic from repositories.

**Example: Clicking a "Submit" Button on a Web Page Using Descriptive Programming**

'===============================

' Click "Submit" using Descriptive Programming

'===============================

' Step 1: Launch the browser and navigate to a webpage

SystemUtil.Run "chrome.exe", "https://example.com"

' Step 2: Sync browser (wait for page to load)

Browser("micClass:=Browser").Sync

' Step 3: Define the "Submit" button using description

Dim oButton

Set oButton = Description.Create

oButton("micClass").Value = "WebButton"

oButton("html tag").Value = "INPUT"

oButton("type").Value = "submit"

oButton("name").Value = "submit" ' OR you can use "value" := "Submit" depending on actual property

' Step 4: Click the button

Browser("micClass:=Browser").Page("micClass:=Page").WebButton(oButton).Click

' Step 5: Log result

Reporter.ReportEvent micPass, "Submit Button", "Submit button clicked successfully."

**Explanation of the Script**

* **Browser("micClass:=Browser")** and **Page("micClass:=Page")**: Describe the browser and page generically.
* **WebButton(...)**:
  + "html tag:=INPUT": Specifies the HTML element is an <input> tag.
  + "type:=submit": Indicates it's a submit-type button.
  + "value:=Submit": The visible label/text on the button.

You can add or replace properties depending on how the button is defined (like name, id, or innertext).

* **Q20:** Explain the syntax for Descriptive Programming in UFT. Write a script that uses descriptive programming to interact with a web element based on its properties (e.g., link text, tagname, etc.).

**Answer**: **Descriptive Programming (DP)** in UFT allows you to define and interact with objects **directly in the script using their properties**, instead of storing them in the Object Repository.

**Syntax for Descriptive Programming**

vb

Copy code

objectType("property1:=value1", "property2:=value2", ...).operation

* objectType: UFT test object class (e.g., WebEdit, WebButton, Link, etc.)
* "property:=value": A **property-value pair** used to identify the object
* .operation: The action you want to perform on that object (e.g., .Click, .Set, .Select)

**Using a Description Object (Advanced Syntax)**

You can also use a Description object to define properties more cleanly:

vbscript

Copy code

Set desc = Description.Create

desc("html tag").Value = "A"

desc("innertext").Value = "Contact Us"

Browser("micClass:=Browser").Page("micClass:=Page").Link(desc).Click

**Example Script: Interacting with a Link Using Descriptive Programming**

Script: Click a Link Using Descriptive Programming

This script:

Opens Chrome

Navigates to a webpage

Finds and clicks a link with innertext = "About Us"

vbscript

Copy

Edit

' Step 1: Launch browser

SystemUtil.Run "chrome.exe", "https://example.com"

' Step 2: Wait for page load

Browser("micClass:=Browser").Page("micClass:=Page").Sync

' Step 3: Define the link using descriptive programming

Set linkDesc = Description.Create

linkDesc("micClass").Value = "Link"

linkDesc("innertext").Value = "About Us"

linkDesc("html tag").Value = "A"

' Step 4: Click the link

Browser("micClass:=Browser").Page("micClass:=Page").Link(linkDesc).Click

' Step 5: Report

Reporter.ReportEvent micPass, "Descriptive Click", "Successfully clicked the 'About Us' link."

* **Q21:** How does UFT handle dynamic objects with Descriptive Programming? Provide an example using a dynamic link or button.

**Answer**: **UFT handles dynamic objects using Descriptive Programming (DP)** by identifying objects at runtime using their properties instead of relying on the Object Repository. This is especially useful for dynamic objects whose properties (like name, index, or ID) change frequently.

**How UFT Handles Dynamic Objects with DP:**

* Instead of recording or storing objects, you define their properties directly in code.
* DP helps bypass Object Repository, making scripts more flexible and maintainable.

**Example: Handling a Dynamic Link or Button**

Suppose you have a button with a dynamic html id like "btn\_12345" (changes every time). You can use a pattern to match it:

vb

Copy code

' Example for a dynamic button

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("html tag:=BUTTON", "html id:=btn\_\d+").Click

Or for a dynamic link with changing text:

vbscript

' Example for a dynamic link where part of the text changes

Browser("micClass:=Browser").Page("micClass:=Page").Link("innertext:=Order.\*", "html tag:=A").Click

**Explanation:**

* "html id:=btn\_\d+" uses a **regular expression** to match any button whose ID starts with "btn\_" followed by digits.
* "innertext:=Order.\*" matches any link whose visible text starts with "Order".

**Key Benefits of Using DP for Dynamic Objects:**

* Avoids maintenance of the Object Repository.
* Works well with objects created dynamically at runtime.
* Allows handling objects with unpredictable or changing properties.

You can also create an object using the **Description object**:

vbscript

Set objDesc = Description.Create

objDesc("html tag").Value = "A"

objDesc("innertext").Value = "Click Here"

Browser("micClass:=Browser").Page("micClass:=Page").Link(objDesc).Click

### **8. Synchronization and Wait Statements:**

* **Q22:** Why is synchronization important in UFT? What are the different synchronization techniques you can use to make sure your script waits for an element to be available?

**Answer**: **Why is Synchronization Important in UFT?**

**Synchronization** in UFT (Unified Functional Testing) ensures that the **test script waits for the application to be ready** before performing any actions.  
Without synchronization, UFT might try to interact with UI elements **before they are fully loaded or available**, causing the test to fail unnecessarily.

**Common Reasons for Synchronization Issues:**

* Web elements loading at different speeds
* Delayed responses from the server
* Animation or dynamic UI changes
* AJAX-based operations

**Synchronization Techniques in UFT**

**1. Wait Method**

* Forces the script to pause for a fixed number of seconds.
* Syntax:

vbscript

Copy code

Wait 5 ' Pauses for 5 seconds

* **Use When**: You know exactly how long the delay will be (not preferred for dynamic waits).

**2. Sync Method**

* Waits for the **browser page** to finish loading.
* Syntax:

vb

Copy code

Browser("micClass:=Browser").Page("micClass:=Page").Sync

* **Use When**: Waiting for the full page to load.

**3. Exist Method**

* Waits until a specific object exists or is visible on the page.
* Syntax:

vbscript

Copy code

If Browser("micClass:=Browser").Page("micClass:=Page").WebButton("name:=Login").Exist(10) Then

' Proceed with click or other action

End If

* **Use When**: Dealing with dynamic elements or waiting for specific controls to appear.

**4. WaitProperty Method**

* Waits until a specified object property meets a particular value within a timeout.
* Syntax:

vbscript

Copy code

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("name:=Submit") \_

.WaitProperty "enabled", True, 15

* **Use When**: You’re waiting for a button to become enabled or visible.

**5. Smart Synchronization (Smart Wait)**

* Automatically enabled in newer UFT versions for web apps.
* UFT detects object readiness more intelligently.
* Can be configured in **Test Settings > Run > Object Synchronization Timeout**.

**Summary Table**

| **Synchronization Technique** | **Purpose** | **Use Case Example** |
| --- | --- | --- |
| Wait | Static delay | Pause 5 seconds before next step |
| Sync | Waits for full page load | After navigating to a new web page |
| Exist(timeout) | Checks if object appears | Wait for login button to appear |
| WaitProperty | Waits for specific property change | Wait until submit button is enabled |
| Smart Synchronization | Auto-waits for web objects (if enabled) | Handle AJAX or dynamic web loading |

* **Q23:** Write a script that uses the Sync method and Wait method to ensure UFT waits for a page to load before performing actions like clicking a button.

**Answer**: '===============================

' Sync and Wait Example in UFT

'===============================

' Step 1: Launch Browser and Navigate to a page

SystemUtil.Run "iexplore.exe", "https://example.com"

' Step 2: Wait for browser to load completely

Browser("micClass:=Browser").Sync ' Dynamic wait (waits for page ready)

' Step 3: Optional static wait for UI elements

Wait(3) ' Wait for 3 seconds (can be adjusted)

' Step 4: Click a button after page is ready

Browser("micClass:=Browser").Page("micClass:=Page").WebButton("html id:=submitBtn").Click

' Step 5: Report success

Reporter.ReportEvent micPass,

* **Q24:** How would you handle synchronization issues when testing a slow application or a page with dynamic content?

**Answer**: **Handling Synchronization Issues in UFT for Slow or Dynamic Applications**

When testing **slow applications** or pages with **dynamic content (e.g., AJAX, JavaScript-heavy UI)**, synchronization becomes critical to avoid false test failures. Here’s how you can effectively manage synchronization in UFT:

**1. Use Exist Method with Timeout**

Checks if an object appears within a given time.

vbscript

If Browser("micClass:=Browser").Page("micClass:=Page") \_

.WebElement("innertext:=Loading...").Exist(20) Then

Browser("micClass:=Browser").Page("micClass:=Page") \_

.WebElement("innertext:=Loading...").WaitProperty "visible", False, 30

End If

**Use When**: The element appears after a delay or asynchronously.

**2. Use WaitProperty to Wait for Specific States**

Waits until an object property reaches the expected value.

vbscript

Browser("micClass:=Browser").Page("micClass:=Page") \_

.WebButton("name:=Submit").WaitProperty "enabled", True, 15

**Use When**: Waiting for a button to be clickable or a field to become visible.

**3. Set Global Synchronization Timeout**

Applies a timeout for all object operations.

* Go to: **File > Settings > Run > Object Synchronization Timeout**
* Set it to a higher value (e.g., 30–60 seconds)

**Use When**: General slowness across the entire application.

**4. Use Sync Method for Web Applications**

Waits for the browser page to fully load.

vb

Browser("micClass:=Browser").Page("micClass:=Page").Sync

**Use When**: After navigation or page reload.

**5. Avoid Hard-Coded Wait Unless Necessary**

vbscript

Wait 5 ' Static wait – not recommended unless unavoidable

**Use When**: The delay is known and consistent (least preferred).

**6. Use Smart Synchronization (UFT Feature)**

In recent versions of UFT, smart synchronization is automatically applied for certain web apps.  
Make sure **Smart Identification and Synchronization** is enabled in settings.

**Best Practices**

* Prefer **dynamic waits** (Exist, WaitProperty) over **static waits** (Wait)
* Combine multiple techniques if needed for different page elements
* Handle AJAX or dynamic content using **conditional loops** when applicable

### **9. Error Handling and Recovery:**

* **Q25:** How can you add exception handling in UFT to handle pop-ups or alerts that appear unexpectedly during the test execution?

**Answer**: To handle pop-ups or alerts in **UFT (Unified Functional Testing)**, you can use **exception handling** techniques like **Recovery Scenarios** and **programmatic error handling**. Here's a brief explanation:

**1. Using Recovery Scenarios:**

* **Recovery Scenario Manager** in UFT helps you handle unexpected events like pop-ups.
* Steps:
  + Open **Recovery Scenario Manager** from the Resources menu.
  + Define a new recovery scenario with:
    - **Trigger Event** (e.g., pop-up window appears).
    - **Recovery Operation** (e.g., click "OK" or "Close").
    - **Post-Recovery Test Run Option** (e.g., repeat or continue test).
  + Associate the scenario with your test.

**2. Programmatic Handling using VBScript:**

* You can use conditional checks and error-handling code like:

vbscript

If Dialog("text:=Unexpected Error").Exist(2) Then

Dialog("text:=Unexpected Error").WinButton("text:=OK").Click

End If

* Or use **On Error Resume Next**:

vbscript

On Error Resume Next

' your test step here

If Err.Number <> 0 Then

' handle error or log it

Err.Clear

End If

On Error GoTo 0

### **10. Test Results and Reporting:**

* **Q26:** Explain how UFT generates test results. How do you view and analyze the test results after running a test in UFT?

**Answer**: **UFT generates detailed test results** for each test execution to help identify successes, failures, and issues during automation. Here's a brief explanation:

**How UFT Generates Test Results:**

* After running a test, UFT automatically creates a **Test Results Report**.
* It logs the status of each step: **Pass, Fail, Warning, or Done**.
* Information included:
  + Execution time
  + Expected vs. actual outcomes
  + Screenshots (if configured)
  + Error messages or recovery steps
  + Custom reporter logs (using Reporter.ReportEvent)

**How to View and Analyze Test Results:**

1. **Automatically Opened Report:**
   * When the test finishes, UFT opens the **Test Results Viewer**.
   * It displays a tree structure of actions and steps.
2. **Navigating the Report:**
   * You can expand nodes to see:
     + **Step Details**
     + **Messages** logged using Reporter.ReportEvent
     + Screenshots (if available)
3. **Analyzing Failures:**
   * Failed steps are marked in red.
   * Click on them to see:
     + The line of code
     + The error description
     + Any associated recovery operation
4. **Exporting Results:**
   * You can export results to **HTML** or **PDF** formats from the viewer.
5. **Test Results Location:**
   * By default, results are saved in the test folder under TestName\Report.

* **Q27:** What is the difference between the "Test Results" tab and the "Run-Time Data Table" in UFT? How would you use them to debug a failing test?

**Answer**:

| **Feature** | **Test Results Tab** | **Run-Time Data Table** |
| --- | --- | --- |
| **Purpose** | Displays execution summary and step-wise outcome of the test | Shows the actual data used during test execution |
| **Content** | Step status (Pass/Fail), error messages, screenshots, log messages | Input/output data used in parameterized steps |
| **Location** | Opens automatically after test run or via View > Test Results | Found under Data > Run-Time Data Table in UFT |
| **Usage in Debugging** | Helps identify which step failed and why | Helps verify whether correct data was used during the failing step |
| **Customization** | Can be customized with Reporter.ReportEvent | Can be modified using script or DataTable editor |
| **File Saved As** | HTML/XML report in the test results folder | .xls file located in the test folder (Default.xls for design-time, runtime version created on execution) |

**How to Use Them for Debugging a Failing Test:**

* **Test Results Tab:**
  + Check which step failed and view the associated message or error.
  + Look at screenshots or recovery actions if present.
* **Run-Time Data Table:**
  + Verify the data passed to that step.
  + Check for incorrect, missing, or unexpected values in input/output fields.
* **Q28:** Write a script that generates a custom report in UFT after executing a test case. This report should include test steps, status (pass/fail), and any relevant messages.

**Answer**: '=========================

' Custom Report Generator

'=========================

' Set the report file path Dim reportFilePath

reportFilePath = "C:\Users\Ananya\Desktop\UFT\_Custom\_Report.txt"

' Create or open the report file for writing

Set fso = CreateObject("Scripting.FileSystemObject")

Set reportFile = fso.CreateTextFile(reportFilePath, True)

' Write header

reportFile.WriteLine "-----------------------------" reportFile.WriteLine " UFT Custom Test Report" reportFile.WriteLine "-----------------------------" reportFile.WriteLine "Test Execution Time: " & Now reportFile.WriteLine ""

'=======================

' STEP 1 - Launch Notepad

'=======================

On Error Resume Next

SystemUtil.Run "notepad.exe" If Err.Number = 0 Then

reportFile.WriteLine "Step 1: Launch Notepad - PASS" Else

reportFile.WriteLine "Step 1: Launch Notepad - FAIL (" & Err.Description & ")" End If

Err.Clear

'=======================

' STEP 2 - Type Text

'=======================

Wait(2)

Window("Notepad").WinEditor("Edit").Set "This is a sample automation report demo."

If Window("Notepad").WinEditor("Edit").GetROProperty("text") = "This is a sample automation report demo." Then

reportFile.WriteLine "Step 2: Type Text - PASS"