Robot Framework Scenario based Interview Questions :

**Scenario: You need to set up Robot Framework in a new environment. How do you proceed?**

# Install Robot Framework

pip install robotframework

# Verify installation

robot –version

**Scenario: You want to run a simple test case to check if a webpage loads successfully. Write a basic test case.**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Variables \*\*\*

${URL} http://example.com

\*\*\* Test Cases \*\*\*

Page Should Load

Open Browser ${URL} Chrome

Page Should Contain Example Domain

[Teardown] Close Browser

**Scenario: A test case needs to use different URLs in different environments (e.g., dev, test, prod). How do you handle this?**

\*\*\* Variables \*\*\*

${URL} ${ENVIRONMENT\_URL}

\*\*\* Test Cases \*\*\*

Page Should Load

Open Browser ${URL} Chrome

Page Should Contain Example Domain

[Teardown] Close Browser

Run with:

robot --variable ENVIRONMENT\_URL:http://dev.example.com test\_suite.robot

**Scenario: You need to import a custom Python library in Robot Framework. How do you do it?**

# custom\_library.py

class CustomLibrary:

def custom\_keyword(self):

return "Hello, Robot Framework!"

\*\*\* Settings \*\*\*

Library custom\_library.py

**Scenario: Write a test case that uses a custom keyword to log a message.**

# custom\_library.py

class CustomLibrary:

def log\_message(self, message):

print(message)

\*\*\* Settings \*\*\*

Library custom\_library.py

\*\*\* Test Cases \*\*\*

Log Custom Message

Log Message Hello from custom keyword!

**Scenario: How do you create a data-driven test that verifies multiple user logins using Robot Framework?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Variables \*\*\*

${URL} http://example.com/login

\*\*\* Test Cases \*\*\*

Login Test

[Template] Login User

admin admin123

user1 password1

user2 password2

\*\*\* Keywords \*\*\*

Login User

[Arguments] ${username} ${password}

Open Browser ${URL} Chrome

Input Text id=username ${username}

Input Text id=password ${password}

Click Button id=login

Page Should Contain Welcome, ${username}

[Teardown] Close Browser

**Scenario: You need to capture a screenshot when a test fails. How do you implement this in Robot Framework?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

Suite Teardown Capture Page Screenshot

\*\*\* Variables \*\*\*

${URL} http://example.com

\*\*\* Test Cases \*\*\*

Failing Test

Open Browser ${URL} Chrome

Page Should Contain This text does not exist

[Teardown] Close Browser

**Scenario: How do you use tags to organize and run specific sets of test cases?**

\*\*\* Test Cases \*\*\*

Smoke Test

[Tags] smoke

Open Browser http://example.com Chrome

Page Should Contain Example Domain

[Teardown] Close Browser

Regression Test

[Tags] regression

Open Browser http://example.com Chrome

Page Should Contain Example Domain

[Teardown] Close Browser

Run with:

robot --include smoke test\_suite.robot

**Scenario: You need to run a keyword before and after each test case in a suite. How do you set this up?**

\*\*\* Settings \*\*\*

Suite Setup Suite Setup Keyword

Suite Teardown Suite Teardown Keyword

\*\*\* Keywords \*\*\*

Suite Setup Keyword

Log Starting the test suite

Suite Teardown Keyword

Log Finishing the test suite

\*\*\* Test Cases \*\*\*

Test Case 1

Log Running Test Case 1

Test Case 2

Log Running Test Case 2

**Scenario: You need to use variables defined in a variable file. How do you do this?**

# variables.py

URL = "http://example.com"

USERNAME = "user"

PASSWORD = "pass"

\*\*\* Settings \*\*\*

Variables variables.py

\*\*\* Test Cases \*\*\*

Test Case Using Variables

Open Browser ${URL} Chrome

Input Text id=username ${USERNAME}

Input Text id=password ${PASSWORD}

Click Button id=login

[Teardown] Close Browser

**Scenario: You need to run the same test case with multiple sets of data and verify different outcomes. How do you achieve this using Test Template?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Variables \*\*\*

${URL} http://example.com/login

\*\*\* Test Cases \*\*\*

Verify User Login

[Template] Login and Verify

admin admin123 Welcome, Admin

user1 password1 Welcome, User1

user2 password2 Welcome, User2

\*\*\* Keywords \*\*\*

Login and Verify

[Arguments] ${username} ${password} ${welcome\_message}

Open Browser ${URL} Chrome

Input Text id=username ${username}

Input Text id=password ${password}

Click Button id=login

Page Should Contain ${welcome\_message}

[Teardown] Close Browser

**Scenario: You need to run the same test case with multiple sets of data and verify different outcomes. How do you achieve this using Test Template?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Variables \*\*\*

${URL} http://example.com/login

\*\*\* Test Cases \*\*\*

Verify User Login

[Template] Login and Verify

admin admin123 Welcome, Admin

user1 password1 Welcome, User1

user2 password2 Welcome, User2

\*\*\* Keywords \*\*\*

Login and Verify

[Arguments] ${username} ${password} ${welcome\_message}

Open Browser ${URL} Chrome

Input Text id=username ${username}

Input Text id=password ${password}

Click Button id=login

Page Should Contain ${welcome\_message}

[Teardown] Close Browser

**Scenario: Write a test case that checks the response status of an API call using Robot Framework and Python.**

# api\_library.py

import requests

class APILibrary:

def check\_status\_code(self, url, expected\_status):

response = requests.get(url)

if response.status\_code != expected\_status:

raise AssertionError(f"Expected {expected\_status} but got {response.status\_code}")

\*\*\* Settings \*\*\*

Library api\_library.py

\*\*\* Test Cases \*\*\*

Check API Status

Check Status Code https://api.example.com 200

**Scenario: A test case requires you to check for a file download and verify its content. How do you approach this?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

Library OperatingSystem

\*\*\* Variables \*\*\*

${URL} http://example.com/download

${FILE\_PATH} /path/to/downloaded/file.txt

\*\*\* Test Cases \*\*\*

Verify File Download

Open Browser ${URL} Chrome

Click Link id=download\_link

Wait Until File Exists ${FILE\_PATH} 10s

File Should Exist ${FILE\_PATH}

${content}= Get File ${FILE\_PATH}

Should Contain ${content} Expected Content

[Teardown] Close Browser

**Scenario: How do you handle conditional logic in Robot Framework test cases?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Variables \*\*\*

${URL} http://example.com

\*\*\* Test Cases \*\*\*

Conditional Logic Example

Open Browser ${URL} Chrome

${is\_logged\_in}= Run Keyword And Return Status Page Should Contain Log Out

Run Keyword If ${is\_logged\_in} Log Out User

... ELSE Log In User

\*\*\* Keywords \*\*\*

Log Out User

Click Link id=logout

Log In User

Input Text id=username user

Input Text id=password pass

Click Button id=login

Page Should Contain Welcome, user

**Scenario: How do you perform parallel test execution in Robot Framework?**

pip install robotframework-pabot

pabot --processes 4 path\_to\_test\_suite.robot

1. **Scenario: You need to integrate Robot Framework tests with a CI/CD pipeline. What steps do you take?**
   * **Answer:**
     + Install necessary dependencies in the CI/CD environment (e.g., Python, Robot Framework, SeleniumLibrary).
     + Configure the CI/CD pipeline to run Robot Framework tests using a command like **robot path\_to\_test\_suite.robot**.
     + Collect and store test results as artifacts.
     + Optionally, configure notifications or integrations for test results.
2. **Scenario: How do you handle exceptions in Robot Framework to ensure that specific cleanup actions are performed even if a test case fails?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Test Cases \*\*\*

Test With Cleanup

[Setup] Setup Test

[Teardown] Cleanup Test

Run Keyword And Ignore Error Failing Keyword

\*\*\* Keywords \*\*\*

Setup Test

Log Setting up the test

Cleanup Test

Log Cleaning up after the test

Failing Keyword

Fail This is an intentional failure

**Scenario: How do you manage large test suites with reusable keywords?**

* **Answer:**
  + Organize keywords into resource files.
  + Use libraries for common functionalities.
  + Follow a modular structure for test cases.

\*\*\* Settings \*\*\*

Resource common\_keywords.robot

\*\*\* Test Cases \*\*\*

Test Case 1

Common Keyword arg1 arg2

Test Case 2

Common Keyword arg3 arg4

**Scenario: You need to wait for an element to become visible on the page before interacting with it. How do you achieve this?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Test Cases \*\*\*

Wait For Element Example

Open Browser http://example.com Chrome

Wait Until Element Is Visible id=some\_element 10s

Click Element id=some\_element

[Teardown] Close Browser

**Scenario: You need to perform database operations within your Robot Framework tests. How do you integrate this?**

\*\*\* Settings \*\*\*

Library DatabaseLibrary

\*\*\* Variables \*\*\*

${DB Connection} psycopg2 dbname=test user=postgres password=secret

\*\*\* Test Cases \*\*\*

Database Test

Connect To Database ${DB Connection}

${result}= Query SELECT \* FROM users WHERE id=1

Log ${result}

Disconnect From Database

**Scenario: How do you create a custom Robot Framework listener to extend logging functionalities?**

# custom\_listener.py

from robot.api import logger

from robot.api.deco import keyword

from robot.api import ExecutionListener

class CustomListener(ExecutionListener):

def start\_test(self, name, attributes):

logger.info(f"Starting test: {name}")

def end\_test(self, name, attributes):

logger.info(f"Ending test: {name}")

# Import in Robot Framework

\*\*\* Settings \*\*\*

Library custom\_listener.py

**Scenario: You need to run a specific test case multiple times with different data sets and log the results separately. How do you achieve this?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

\*\*\* Test Cases \*\*\*

Data Driven Test

[Template] Log In And Verify

${USERNAME} ${PASSWORD} ${EXPECTED\_RESULT}

user1 pass1 Welcome, User1

user2 pass2 Welcome, User2

user3 pass3 Welcome, User3

\*\*\* Keywords \*\*\*

Log In And Verify

[Arguments] ${username} ${password} ${expected\_result}

Open Browser http://example.com/login Chrome

Input Text id=username ${username}

Input Text id=password ${password}

Click Button id=login

Page Should Contain ${expected\_result}

[Teardown] Close Browser

**Scenario: How do you implement a retry mechanism in Robot Framework for flaky tests?**

\*\*\* Settings \*\*\*

Library SeleniumLibrary

Retry Keyword 3x Retry On Failure

\*\*\* Test Cases \*\*\*

Flaky Test

[Tags] retry

Open Browser http://example.com Chrome

Page Should Contain Example Domain

[Teardown] Close Browser

\*\*\* Keywords \*\*\*

Retry On Failure

Run Keyword And Ignore Error Flaky Keyword

Flaky Keyword

Log This might fail sometimes

Fail Intentional failure

Other important questions :

1. **What is Robot Framework?**
   * Robot Framework is an open-source automation framework for acceptance testing and acceptance test-driven development (ATDD). It uses a keyword-driven approach to create test cases.
2. **How do you install Robot Framework?**
   * Robot Framework can be installed using pip: **pip install robotframework**.
3. **What are the key components of Robot Framework?**
   * The key components are Test Cases, Keywords, Libraries, Variables, and Settings.
4. **What file formats are used in Robot Framework for test cases?**
   * Test cases in Robot Framework are usually written in plain text format with **.robot** or **.txt** extensions.

**Intermediate Questions**

1. **How do you create a test suite in Robot Framework?**
   * A test suite in Robot Framework is created by writing multiple test cases in a single **.robot** file. A directory containing multiple such files can also be considered a test suite.
2. **Explain the concept of keywords in Robot Framework.**
   * Keywords are reusable sets of actions that can be used to build test cases. They can be either built-in, user-defined, or from external libraries.
3. **What is a library in Robot Framework, and how do you import one?**
   * Libraries provide additional keywords that you can use in your test cases. They can be imported using the **Library** setting in the settings section of the test suite file, e.g., **Library SeleniumLibrary**.
4. **How do you execute Robot Framework tests?**
   * Tests can be executed using the command: **robot path\_to\_test\_suite.robot**.
5. **How can you pass arguments to a test case in Robot Framework?**
   * Arguments can be passed to test cases using the variable syntax, e.g., **robot --variable VAR\_NAME:VALUE path\_to\_test\_suite.robot**.

**Advanced Questions**

1. **How do you handle data-driven tests in Robot Framework?**
   * Data-driven tests can be handled using the **For** loop syntax in Robot Framework or by using **Test Template** in combination with variables.
2. **What are some common Robot Framework built-in libraries?**
   * Common built-in libraries include BuiltIn, Collections, DateTime, OperatingSystem, and String.
3. **How can you integrate Robot Framework with Selenium?**
   * By using the SeleniumLibrary, which provides keywords to interact with web browsers.
4. **Explain how to create a custom library in Robot Framework using Python.**
   * A custom library in Robot Framework is created by writing Python classes and methods, then importing the library in the Robot Framework test suite. Each method in the class can be used as a keyword.
5. **How do you use tags in Robot Framework?**
   * Tags are used to categorize test cases. They can be added to test cases using the **[Tags]** setting and can be used to include or exclude tests from execution.
6. **How do you capture screenshots on test failure in Robot Framework?**
   * Using the **Capture Page Screenshot** keyword from the SeleniumLibrary, which can be added to the teardown section of the test case or suite.

**Practical Questions**

1. **Explain how to run a specific test case in a suite.**
   * A specific test case can be run using the **--test** option followed by the test case name, e.g., **robot --test "Test Case Name" path\_to\_suite.robot**.
2. **Describe how you would log custom messages during test execution.**
   * Custom messages can be logged using the **Log** keyword, e.g., **Log This is a custom log message**.
3. **What are some best practices for writing Robot Framework test cases?**
   * Use descriptive names for test cases and keywords.
   * Keep test cases and keywords small and focused.
   * Use variables for values that may change.
   * Organize test cases and keywords logically.
   * Regularly refactor to remove duplication and improve readability.
4. **How do you use setup and teardown in Robot Framework?**
   * Setup and teardown are used to perform actions before and after a test case or suite. They are specified using **[Setup]** and **[Teardown]** settings in the test case or suite level.