Stages

1. Robot Framework & Selenium
2. Python Selenium
3. Python – pytest
4. Behave – BDD
5. API – postman & api automation

Robot Framework - <https://github.com/balaji-githubstore/robot_framework_citijan23.git>

pytest framework - <https://github.com/balaji-githubstore/robot_framework_citijan23/tree/pytest-framework>

python-selenium - <https://github.com/balaji-githubstore/robot_framework_citijan23/tree/python-selenium>

Web Automation – Selenium

Mobile Automation (android/ios/windows) – Appium

API Automation – Request Lib

Robot Framework:-

<https://robotframework.org/>

<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html>

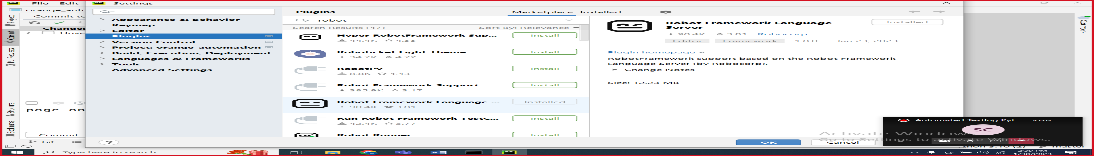
* Keyword driven framework
* Installation
* Libraries
  + Standard Libraries
    - Builtin, Operating System, String, Collections...
  + External Libraries
    - Selenium library
    - HTTP RequestsLibrary
    - Appium library

Installation:

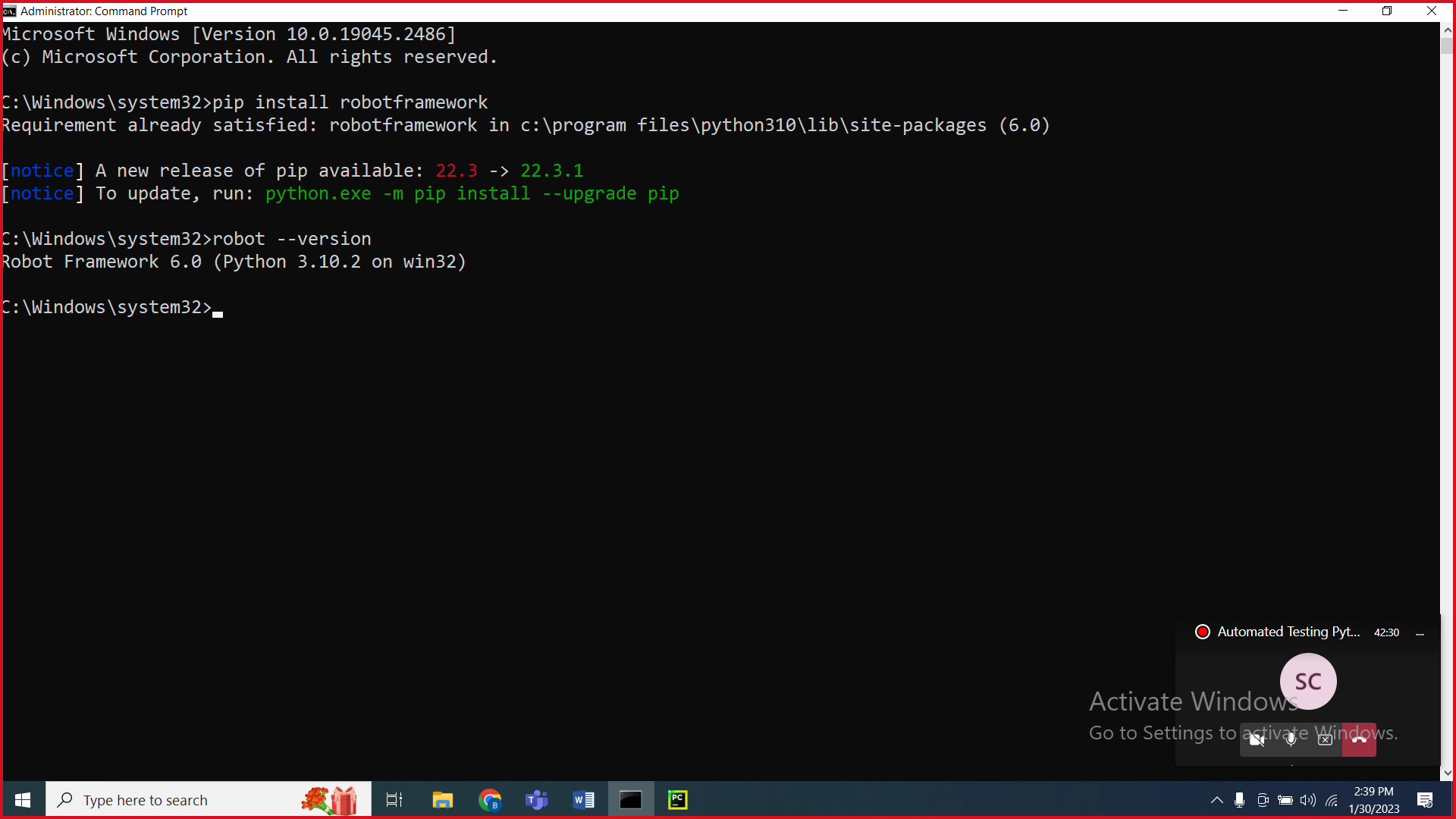
1. Install Python
   1. Add Path to environment “Path” variable

C:\Program Files\Python310

C:\Program Files\Python310\Scripts

1. Pycharm / visual studio code IDE
   1. Robocorp plugin
2. Install robotframework

pip install robotframework



Steps to create project for robot framework

1. Create robot project
2. Create suite folder and then create suite file (.robot)
3. Sections in robot framework
   1. Settings
   2. Test Cases
   3. Variable
      1. Scalar variable ($)
      2. List (@)
      3. Dictionary (&)
   4. Keywords

Selenium Library

1. Selenium library

pip install --upgrade robotframework-seleniumlibrary

1. Selenium Keyword doc

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>

1. Click, type, Select
2. To inspect – tagname, attributes, text or not
3. Locating Stratergies
   1. Basic locators – id, name, classname, tagname, link, partial link
   2. Advance locators – xpath, css

When duplicate locators are there then findelement/get webelement picks the first element

1. To inspect – ctrl+shift+c
2. For page load – wait for page load to complete
3. Get WebElement/FindElement 🡪 it checks for presence of element in 0.5s
4. Synchronization
   1. Unconditional wait

Sleep 5s 🡪 not recommended

* 1. Conditional wait
     1. Implicit wait
        1. Default implicit wait – 0s
        2. Applicable for Get WebElement and Get WebElements
        3. Example: Implicit wait – 30s
           1. If element not present – it will check for 30s and then throw exception
           2. If element present – it will do the operation immediately
           3. Polling time – 0.5s (how frequently it checks)
     2. Explicit wait
        1. Exact condition
        2. Polling time – 0.5s

1. Dropdown
   1. With select tag
      1. Select From List By Label
      2. Select From List By Value
      3. Select From List By Index – starts at 0
   2. Without select tag
      1. Click Element to solve it
2. Multiple tabs/windows, alert, frame
3. Mutliple tabs/windows
   1. Switch Window title
   2. Use NEW & MAIN
4. Close Browser vs Close window
   1. Close window/ driver.close() – close the current tab/session3
   2. Close Browser/ driver.quit() – close the browser and also kills the driver associated to it.

Faker Lib

1. Install

pip install robotframework-faker

1. Alert – javascript alert
   1. Handle Alert
2. Frame – html embedded into another html
   1. Even though locator is correct, we used to get Element with locator 'name=fldLoginUserId' not found.
   2. To check for tagname iframe or frame
   3. Select frame
3. Actions – mouse/keyboards
4. Javascripts
   1. Click on hidden elementss
   2. Type on read only element
   3. Scroll page

Javascript – click & type

document.querySelector('#email').click()

document.querySelector('#email').value='hello'

1. Git architecture

Project (local system) 🡪 local repository (local system) 🡪 remote repo (github, aws code commit, bit bucket)

1. Test Setup & Tear Down – runs before and after each test case
2. Robot Framework
   1. Data-Driven Framework – Test Template in robot framework
      1. Create a keyword with test data as an arguments
      2. Declare the template in setting sections
      3. Create Test case and pass the test data(arguments)
   2. Data-Driven using excel – Test Template & Excel
      1. Install

pip install --upgrade robotframework-datadriver

pip install --upgrade robotframework-datadriver[XLS]

* + 1. Make sure the arguments in test template in present as a header in the excel sheet
    2. Declare settings section

Library DataDriver file=../test\_data/openemr\_data.xlsx sheet\_name=InvalidLoginTest

Python Selenium

1. Create a project
2. Create package and python file
3. How to call methods in python?
   1. Methods in module 🡪 modulename.methodname()
   2. Class
      1. Static method 🡪 Classname.methodname()
      2. Non-static method 🡪
         * 1. Create object
           2. Call the method using the objectref.methodname()
4. Install selenium for python

pip install selenium

1. Webelement-> **driver.find\_element(By.NAME,"UserTitle")**
2. Click, type, select
3. Multiple tabs/windows, alert, frame – switch\_to
4. Mutliple tabs/windows
   1. driver.window\_handles 🡪 list of string (all session id detail)
5. Actions – mouse/keyboards
   1. May not throw proper error
   2. May not work in headless mode
   3. Do not distrub the mouse/keyword
6. Frame – html embedded into another html
   1. Even though locator is correct, we used to get Element with locator 'name=fldLoginUserId' not found.

selenium.common.exceptions.**NoSuchElementException**:

* 1. To check for tagname iframe or frame
  2. Select frame

1. Javascripts
   1. Click on hidden elementss
   2. Type on read only element
   3. Scroll page

Javascript – click & type

document.querySelector('#email').click()

document.querySelector('#email').value='hello'

Pytest – Test Framework in Python

Hybrid Framework

1. Pytest – Python Testing Framework
2. Data Driven Framework
3. Page object model – design pattern

Packages:

tests-> contains test class, test methods

base 🡪 browser and report config

pages-> page object class, methods

utitlites -> resusable code for working with excel, db, json..

Steps to create customized framework using pytest

1. Install

**pip install pytest**

1. Create project
2. Create package “tests”
3. Create module inside tests package 🡪 test\_login.py
4. Create a test class (TestLoginUI) and test methods (test\_title())
5. Every test method should have minimum one assertion. Assertions decides wether test method is pass or fail

**pip install assertpy**

1. Pytest fixtures

Scope🡪 functions

@pytest.fixture(scope="function", autouse=True)  
def setup(self):  
 *# will run before each test method* print("browser launch")  
 yield  
 *# will run after each test method always* print("browser close")

1. autouse=True 🡪 that fixtures will be running always depends on the scope
2. Reuse logics – create methods

Reuse variable, methods – then we can inheritance

1. We created a parent class WebDriverWrapper for browser config
2. Created test method for valid and invalid login under TestLogin class
3. Data Driven Framework – using pytest mark parameterization
   1. Create a test method with parameter/arguments
   2. Add @pytest.mark.parametrize to the test method and supply the arguments
4. @pytest.mark.parametrize(  
    "username,password,expected\_title",  
    [  
    ["admin", "pass", "OpenEMR"],  
    ["accountant", "accountant", "OpenEMR"]  
    ]  
   )

Reference

|  |  |  |
| --- | --- | --- |
| **Strategy** | **Match based on** | **Example** |
| id | Element id. | id:example |
| name | name attribute. | name:example |
| identifier | Either id or name. | identifier:example |
| class | Element class. | class:example |
| tag | Tag name. | tag:div |
| xpath | XPath expression. | xpath://div[@id="example"] |
| css | CSS selector. | css:div#example |
| dom | DOM expression. | dom:document.images[5] |
| link | Exact text a link has. | link:The example |
| partial link | Partial link text. | partial link:he ex |
| sizzle | Sizzle selector deprecated. | sizzle:div.example |
| data | Element data-\* attribute | data:id:my\_id |
| jquery | jQuery expression. | jquery:div.example |
| default | Keyword specific default behavior. | default:example |

2. Pytest

### **Fixture scopes**

Fixtures are created when first requested by a test, and are destroyed based on their scope:

* function: the default scope, the fixture is destroyed at the end of the test.
* class: the fixture is destroyed during teardown of the last test in the class.
* module: the fixture is destroyed during teardown of the last test in the module.
* package: the fixture is destroyed during teardown of the last test in the package.
* session: the fixture is destroyed at the end of the test session.

Assignments

# Task 1 (Robot Framework)

1. Navigate onto <https://www.salesforce.com/in/form/signup/freetrial-sales/>
2. Enter first name as “John”
3. Enter last name as “wick”
4. Enter work email as “john@gmail.com”
5. Select Job title as “IT Manager”
6. Select Employees as “101-500 employees”
7. Select country as “United Kingdom”
8. Do not fill the phone number
9. Click on check box
10. Click on start my free trial
11. Get the error message displayed “Enter a valid phone number”

# Task 2 (Complete in python)

1. Navigate onto https://www.online.citibank.co.in/
2. Close if any pop up comes
3. Click on Login
4. Click on Forgot User ID?
5. Choose Credit Card
6. Enter credit card number as 4545 5656 8887 9998
7. Enter cvv number
8. Enter date as “14/04/2022”
9. Click on Proceed
10. Get the text and print it “Please accept Terms and Conditions”

# Task 3 (Using Python Selenium)

1. Navigate onto https://www.medibuddy.in/
2. Click on Not Now button
3. Click on Login
4. Click on I have an Insurance/Corporate Account
5. Click on Login using Username & Password
6. Enter username as john
7. Enter password as john123
8. Click on show password
9. Click log in
10. Get the error message shown and print it in terminal

# Task 4 (robot framework – selenium)

1. Navigate onto https://nasscom.in/
2. Click on Login and then click on register
3. Enter First name as admin
4. Enter Last name as pass
5. Enter email address as admin@gmail.com
6. Enter company name as Google
7. Select IT Consulting from dropdown
8. No need to automate CAPTCHA
9. Click on Register

# Day 3 - Task 1 (Add in python - pytest framework)

1. Navigate onto http://demo.openemr.io/b/openemr/
2. Update username as admin
3. Update password as pass
4. Select language as English (Indian)
5. Click on the login button
6. Click on Patient  Click New Search
7. Add the first name, last name
8. Update DOB as today's date driver.findElement(By.id("form\_DOB")).sendKeys("2021-12-");
9. Update the gender
10. . Click on the create new patient button above the form
11. . Click on confirm create new patient button.
12. . Print the text from alert box (if any error before handling alert add 5 sec wait)
13. . Handle alert
14. Close the Happy Birthday popup
15. Assert the added patient name