Selenium:

=========

Selenium is an open-source tool that automates web browsers. Its a interface that allows you to write test scripts in programming languages like Ruby, Java, NodeJS, PHP, Perl, Python, and C#, among others.

Selenium Suite consists of the following components:

-->Selenium Integrated Development Environment (IDE)

-->Selenium Remote Control (RC)

-->Selenium WebDriver

-->Selenium Grid

Selenium WebDriver:

==========

Selenium WebDriver is an automated testing framework used for the validation of web applications.

Selenium WebDriver executes test scripts through browser with specific drivers.

Pre-Requisites:

==========

download Chrome,IE & Firefox drivers

Basic packages to import:

==========

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

Syntax for giving path:

==========

```
syntax: driver=webdriver.Browser_Name(executable_path="")
```

for chrome browser:

---

driver=webdriver.Chrome(executable_path="")

for IE browser:

---

driver=webdriver.IE(executable_path="")

for firefox browser:

---

driver=webdriver.Firefox(executable_path="")

Basic web driver commands:

==========

-->get():The command launches a new browser and opensthe specified URL in the browser instance.

syntax: driver.get()

-->CurrentUrl():This command is used to retrieve the URL of the webpage the user is currently accessing and returns a string value

syntax: driver.currenturl()

-->Title():This command is used to retrieve the title of the webpage the user is currently working on.

syntax: driver.title()

-->Page_Source():This command is used to retrieve the html code of the webpage the user is currently working on.

syntax: driver.page_source()

-->Time.Sleep():This method will stop the execution of the script for the specified duration of time, irrespective of whether the element is found or not on the web page.

syntax:time.sleep()

-->close():This command is used to close the webpage the user is currently working on.

syntax: driver.close()

-->Quit():This command is used to quit all the webpages the user is currently working on.

syntax: driver.quit()

Locators:

=========

-->Find_Element_by_Id:We can use find_element_by_id method when you know the id attribute of an element. This method returns the first matching element if there is more than one match.

syntax:driver.find_element_by_id()

-->Find_Element_by_Name:We can use find_element_by_name method when you know the name attribute of an element. Find Element By name method returns the first occurring element which has a matching name attribute.

syntax:driver.find_element_by_name()

-->Find_Element_by_Tag_Name:Tagname is a tag used to form that particular element; This method returns the first matching element, if there is no match then raises a NoSuchElementException

syntax:driver.find_element_by_tag_name()

-->Find_Element_by_Xpath: This is a combination of attributes to find the element, as when we try to find elements using other locators you may have other matches also along with the target match.

syntax: xpath=(//button[@name=""]) - matches with particular name

xpath = (//*[@value='']) - here * denotes everything

-->Find_Element_by_Link_Text: This method is used to find elements when the hyperlink is static.

syntax:driver.find_element_by_link_text()

-->Find_Element_by_Partial_Link_Text:This method is used when a certain part of the string keeps changing.

syntax:find_element_by_partial_link_text()

-->Find_Element_by_CSS_Selector:In this the first element with the matching CSS selector will be returned. If no element has a matching CSS selector, a NoSuchElementException will be raised.

syntax:find_elements_by_css_selector()

-->Find_Elements:The above discussed methods will only one matching element even there more than one, but still they will return one element. Sometimes we may need to find all the matching

elements on the webpage, in such cases we can use find_elements method, these also will follow the same principles, but these will return all the matching elements

find_elements_by_name()

find_elements_by_xpath()

find_elements_by_link_text()

find_elements_by_partial_link_text()

find_elements_by_tag_name()

find_elements_by_class_name()

find_elements_by_css_selector()

Navigation Commands:

==========

-->Back():This command is used to take back the user to the previous webpage in the web browser.

syntax:driver.back()

-->Forward():This command is used to take forward the user to the previous webpage in the web browser.

syntax:driver.forward()

-->Refresh():This command is used to refresh the webpage in the web browser.

syntax:driver.refresh()

Conditional Statements:

==========

-->Is_Displayed():is_displayed method is used to check if element is displayed or not. It returns a boolean value True or False.

syntax:driver.is_displayed()

-->Is_Enabled():is_enabled method is used to check if element is enabled or not. It returns a boolean value True or False.

syntax:driver.is_enabled()

-->Is_Displayed():is_selected method is used to check if element is selected or not. It returns a boolean value True or False.

syntax:driver.is_selected()

Waits:

======

-->Implicitly_wait():An implicit wait directs the WebDriver to poll the DOM for a certain amount of time (as mentioned in the command) when trying to locate an element that is not visible immediately.

syntax:driver.implicitly_wait()

-->Explicitly_wait():The explicit wait is used to tell the Web Driver to wait for specific conditions or the maximum time limit before throwing an Exception.

syntax:driver.explicitly_wait()

DropDowns:

==========

Dropdowns are one of the general elements present in any webpage after buttons and text bars dropdowns are a more frequently available element.

Selenium python provides Select class, with help of select class we can handle dropdowns on the webpage.

The main methods we use in this are:

-->select_by_index(int index)

-->select_by_value(String value)

-->select_by_visible_text(String text)

Alerts:

=======

An alert on the webpage used to get the attention of the user to perform some operation on the alert in the webpage.

We can handle alerts using the switch_to_alert() method present in selenium python, with the help of this switch_to_alert() method.

syntax: driver.switch_to_alert()

-->Switch_to_Alert().Accept():Accept the popUp by clicking OK button.

syntax: switch_to_alert().accept()

-->Switch_to_Alert().Dismiss():Dismiss the popUp by clicking OK button.

syntax: switch_to_alert().dismiss()

Frames:

=======

A frame is an element that keeps a document within another document in a page.

We can find the frame using different attribute present in the frame like

syntax: driver.switch_to_frame("")

1.Using ID

2.Using Name

3.Using Index

Windows Handeling:

==========

-->current_window_handle() : current_window_handle method in selenium python returns the current(active) browser's GU ID. It returns GU ID as a string value.

syntax: driver.current_window_handle()

-->switch_to_window() : switch_to method in selenium python helps user to switch between windows, frames, elements, alerts. switch_to_window method switches the control from the current browser window to the target browser window.

syntax: driver.switch_to_window()

Scroling:

=========

Scrolling is an essential feature for using any web page.

we can perform scrolling by three ways.

-->Scroll_Down_Page_by_Pixel():We can use the pixel method for scrolling upto specfic height.

syntax: driver.execute_script(("window.scrollBy(0, Y)")) [Y is height of the pixel]

-->Scroll_Down_Page_till_element_found():It is used to scroll down the page till the element is found.

syntax:
driver.execute_script(("arguments[0].scrollintoView();",Element)

-->Scroll_to_end_of_page():it is used to scroll till end of a web page.

syntax: driver.execute_script(("window.scrollBy(0,document.body.scrollHeight)")

Screenshot:

==========

We can take a screenshot of a webpage in selenium python using save_screenshot() method, save_screenshot () will take the complete page as a screenshot.

syntax: driver.save_screenshot("filename")

The other way to take a screenshot is by using get_screenshot_as_file,in this we can take screenshot as a file it will suppourt only .jpg format.

syntax: get_screenshot_as_file("filename")

Mouse Actions:

==========

Mouse actions are simulating the mouse events like operation performed using mouse.

handling Mouse & keyboard events and mouse events including actions such as Drag and Drop or clicking multiple elements with control key are done using the ActionChains

-->Mouse Hover: Sometimes we may need to hover on an element to see it's properties like color, highlight, background, and font to perform hover, we should use a method called mouseMove().

basic code: actions = ActionChains(driver)

actions.move_to_element(menu)

actions.perform()

-->Drag & Drop: drag and drop() drags source element into the target element or to the place, drag and drop() method accepts two parameters, 1. Source: which you want to drag, 2. Target: where you want to drag.

synatx: element = driver.find_element_by_name("source")

syntax: target = driver.find_element_by_name("target")

-->Double Click: DoubleClick() method simulates the double click of the mouse, we have to pass the element that to be double-clicked.

syntax: actions.double_click(element).perform()

-->Right click(): The element which has to be right clicked ,the click on the present mouse position is performed.

syntax: action.context_click(button).perform()

Upload File:

==========

We can upload files with the help of the send_keys method.we should identify the element which does the task of selecting the file path that has to be uploaded.

syntax: s = driver.find_element_by_xpath("")

s.send_keys("path of the file to be uploaded")

Data Driven:

==========

An Excel spreadsheet document is called a workbook. A single workbook is saved in a file with the .xlsx extension.

Each workbook can contain multiple sheets. The sheet the user is currently viewing is called the active sheet.

Each sheet has columns and rows. A box at a particular column and row is called a cell.

OpenPyxl:

OpenPyXL is a library used to read and write data in Excel.

Cookies:

========

A cookie is a piece of information from the website and saved by your web browser.It stores the login information like user name or email and password.

An HTTP cookie is also known as a web cookie, a browser cookie or an Internet cookie.Cookies are a way of remembering users and their interaction with the site by storing information in the cookie file as key-value pairs.

To return the list of all cookies.

syntax: driver.get_cookies()

-->Add_Cookie():This method is used to create and add the cookie.

syntax: driver.add_cookies()

-->Delete_Cookie():This method is used to delete the cookie.

syntax: driver.delete_cookies()

-->Delete_All_cookies(): This method is used to delete all cookies.

syntax: delete_all_cookies()

UnitTest Framework():

====================

Unit testing is a software testing method by which individual units of source code, such as functions, methods, and class are tested to determine whether they are fit for use.

t\we can use Unittest to organize our code

-->Skip Test():In Unit test it is possible to skip individual test method or TestCase class, conditionally or unconditionally.

syntax: @unittest.skip("conditon")

-->SetUp(): setUp method is defined, the test runner will run that method prior to each test. It is executed before each test.

```
syntax:  def setUp(self):
             pass
```

-->TearDown(): Teardown method is defined the test runner will run that method prior to each test. It is executed after each test.

```
syntax:  def tearDown(self):
             pass
```

PageObjectModel:

==========

Page Object Model is a design pattern that is used mostly in web ui testing for enhancing maintenance of the test code and reducing duplication of it

Page Object Modelprovides us creating a non-fragile test by means of reducing duplicated codes.

Page Object Modeldesign implementation separates the page objects and tests by achieving an abstraction.

PyTest:

=======

PyTest is a testing framework that allows users to write test codes.It is used to write simple and scalable test cases for UI,databases and APIs.

Pytest requires the test method names to start with "test".

-->Assertions:Pytest assertions are checks that return either True or False status.If an assertion fails in a test method, then that method execution is stopped there. The remaining code in that test method is not executed.

-->Run Parallel Tests in Parallel with Pytest:

-->Pytest Fixtures:

Fixtures are used when we want to run some code before every test method. So instead of repeating the same code in every test we define fixtures.

syntax: @pytest.fixture

-->Pytest Parameterized: Parameterizing a test is used to run a test against multiple sets of arguments.

synatx: @pytest.mark.parametrize

-->Skip Tests: If we dont want to run a test case or a test case is not relevant for a particular time. Then we use skip the tests.

syntax: @pytest.mark.skip