Selenium:

— Pradhvi Vardhan

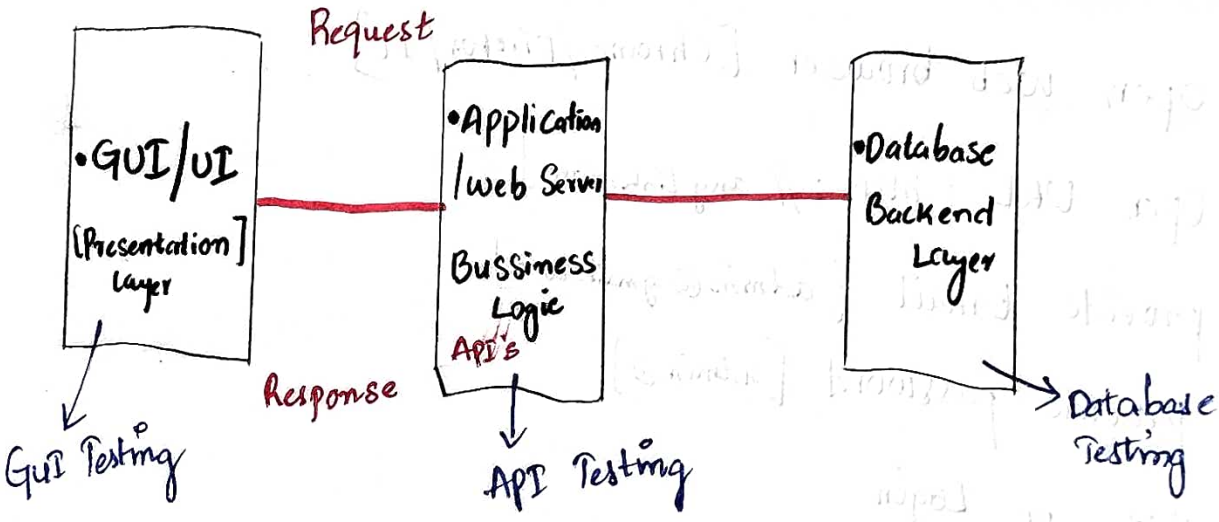
• Web driver

→ Web driver is One of the Component in Selenium.

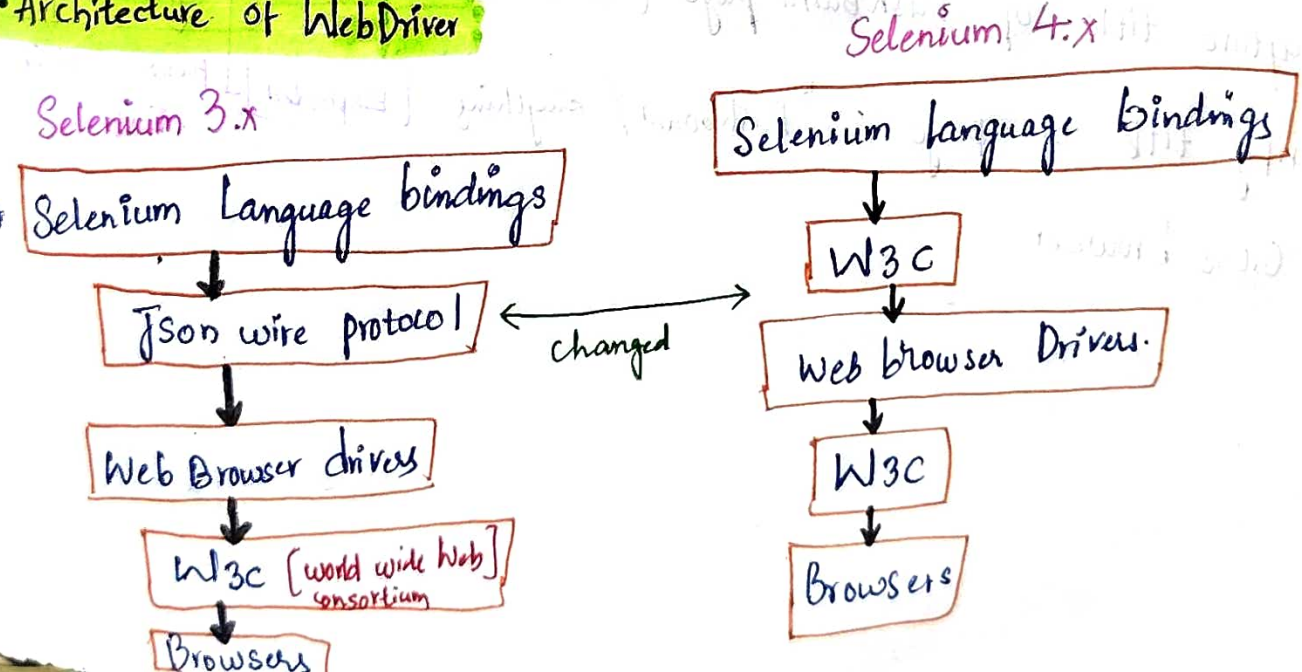
→ web driver is "Module" which contains

- fire fox browser → Firefox()
- Chrome brower → Chrome()
- Edge browser → Edge()

→ Web driver is an "API" [Application Programming Inteface]



• Architecture of WebDriver



Setup & Configure Webdriver in Pycharm

- Pre-requisities :
  - Python
  - Pycharm
- Web browser Specific Drivers
  - Chrome
  - Edge
  - Firefox
  - Safari

Test Case

1. Open web browser [Chrome / FireFox / IE]
2. Open URL [https://anylink.com]
3. provide Email [admin@gmail.com]
4. provide password [admin@]
5. Click on Login
6. Capture title of Dashboard page [Actual title]
7. Verify title of page : "Dashboard / anything [Expected]

Equal / Not Equal
⇓ ⇓
Pass Fail
Test

8. Close browser

CODE

```
from Selenium import Webdriver
from Selenium.Webdriver.Chrome.Service import Service
from Selenium.webdriver.common.by import By

Service_Object = Service ["C:\\Users\\User\\Driver\\Chrome.exe")
                                                    ↓ package
driver = Webdriver.Chrome [Service = Service_obj)

driver.get ("Link of web") # application link / web links

driver.find_element (By.NAME, "txtUsername").Send_Keys ("User")
driver.find_element (By.ID, "txtPassword").Send_Keys ("123")

driver.find_element (By.ID, "btnlogin").click()

# Actual / Expected

actual_title = driver.title
expected_title = "Orange"
if actual_title == expected_title:
    print ("test passed")

else:
    print ("Test Failed")

driver.close()
```

Locators :

↳ We can identify various Elements on the Web using "Locators".

↳ Locaters are addresses that identify a Web element Uniquely with in the page.

↳ They are

Single web element:
* Id
* name
* Link text / partial Link test

More Than Web Element:
* Class name
* Tag name

* Customized Locaters

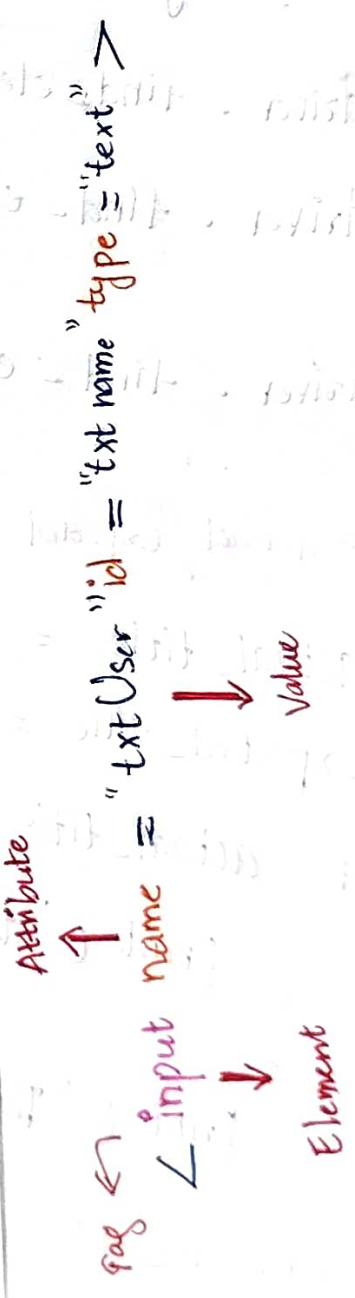
→ CSS Selector :

Combinations:
* Tag and ID
* Tag and class
* Tag and Attribute
* Tag, class and attribute

→ Xpath :

* Absolute Xpath
* Relative Xpath

* HTML structure.



<a = link , href = hyper link Reference

<input = text

Id

driver.find_element(By.ID, "xxx").send_keys("xxx")

HTML Link :

<input class = "search" type = "Text", id = "xxx" name = "search"
placeholder = "sea" value autocomplete = "off">

Name

driver.find_element(By.NAME, "xxx").click()

HTML Link :

▶button type = "Submit", name = "xxx" class = "btn",></button>

Link Test / Partial link Test

driver.find_element(By.LINK_TEST, "Register").click()

driver.find_element(By.PARTIAL_LINK_TEST, "Reg").click()

∴ link Test = We have Mention Total value

Partial Link Test = But, Here we can give min value of attribute

Class Name

```
sliders = driver.find_elements(By.CLASS_NAME, "homeslide")
print(len(sliders))
```

HTML :

```
<UL id = "home slide" style = "max">
  <Li class = "home slide" , style, >
  <Li class = " , style >
  <li class = " "
  < li class = " "
```

Tag Name

```
links = driver.find_elements(By.TAG_NAME, "a")
print(len(links))
```

CSS Selectors

optional

- Tag & Id ⇒ tag # Value of Id
- Tag & Class ⇒ tagname . Value of class
- Tag & attribute ⇒ tagname [attribute = Value]
- Tag & class & attribute ⇒ tagname . Value of class [attribute = Value]

Tag & ID

```
driver.find_element(By.CSS_SELECTOR, "input#email").
    send_keys("xxxx")
```

ID → `#email`

Tag → `input`

Tag & class

```
driver.find_element(By.CSS_SELECTOR, "input.inputtext").
    send_keys("xxxx")
```

class → `.inputtext`

Tag → `input`

Tag & attribute

```
driver.find_element(By.CSS_SELECTOR, "input[place="Email"]").
    send_keys("xxxx")
```

attribute → `[place="Email"]`

Tag → `input`

Email :

Tag & class & attribute

```
driver.find_element(By.CSS_SELECTOR, "input.inputtext[autocomplete="current-password"]").
    send_keys("xxx")
```

Tag → `input`

class → `.inputtext`

attribute → `[autocomplete="current-password"]`

Email :

Password : xxx

Here, we use combination of values, Incase we have same Id, name we can access with different attributes

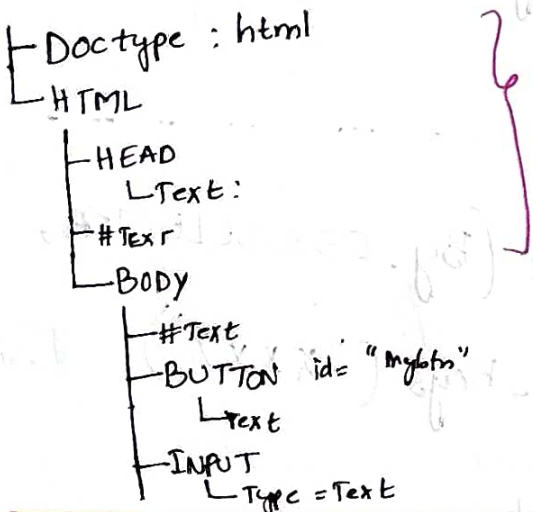
Xpath :- Selectorhub

↳ Xpath is defined as "XML path"

↳ it is a Syntax (or) Language for "Finding any Element" on Web page using XML path expression.

↳ Xpath is used to find the location of any element on webpage using HTML DOM [Document Object Model]

Structure



↳ Xpath is an "Address of the Element"

Types of Xpaths

1. Absolute / Full xpath [unstable]
2. Relative / partial xpath

We use (Always) → 2.

Because,
* if developer introduced new element than absolute xpath will be Broken
* if developer changed the location then absolute xpath will be Broken

• Absolute [Full Xpath]

Ex:-

* /html /body /nav /div /div [2] /ul [3] /Li [1] /a

* /html /body / div [1] /div [3] /div [1] / img

↳ Absolute xpath starts from root html node

↳ Absdute xpath always starts with / [single slash]

↳ In Absolute xpath we use only tags / nodes

• Relative [Partial Xpath]

↳ Relative xpath directly jump to element on "DOM"

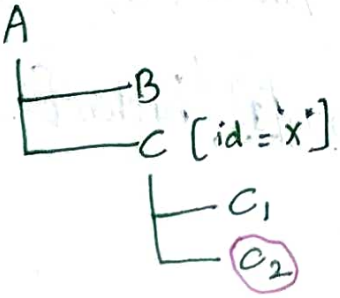
↳ it always starts with // [double slash]

↳ it always use attributes

Ex: //*[@ id = "header-navbar"] /ul [3] /Li [1] /a

//*[@ id = "divlogo"] /img

⇒ DOM :



He we have to Find "C2"

- Absolute (Full) — /A/C/C2
- Partial (Relative) — //C[@id = "x"]/C2

X-Path options

- AND = (By.XPATH, "//input [@name = "xxx" (Right) @place = "xxx" (Right)].SendKeys
- OR = [By.XPATH, "//input [@name = "xxx" (Right) OR @place = "xxx" (wrong)].click()
- CONTAINS()
- STARTS-WITH()
- Text() :- If you want to find/Identify the Inner Text, we use Text Function

CODE

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By

Service_object = Service("c||user||chromedriver.exe")

driver = webdriver.Chrome(service = Service_object)

driver.get(anylink)

# absolute x path
driver.find_element(By.XPATH,"/html[1]/body[1]/div[1]/
    div[2]/div[2]/form[1]/input[1]").send_keys("XXX")

driver.find_element(By.XPATH," /html[1]/body[1]/
    div[6]/div[1]/div[2]/form[1]/button[1]").
    click()
```

Relative Xpath

use single ' '

```
driver.find_element(By.XPATH, "//input[@id='small-search']").Send_keys("T-shirt")
driver.find_element(By.XPATH, "//button[@type='Submit']").click()
```

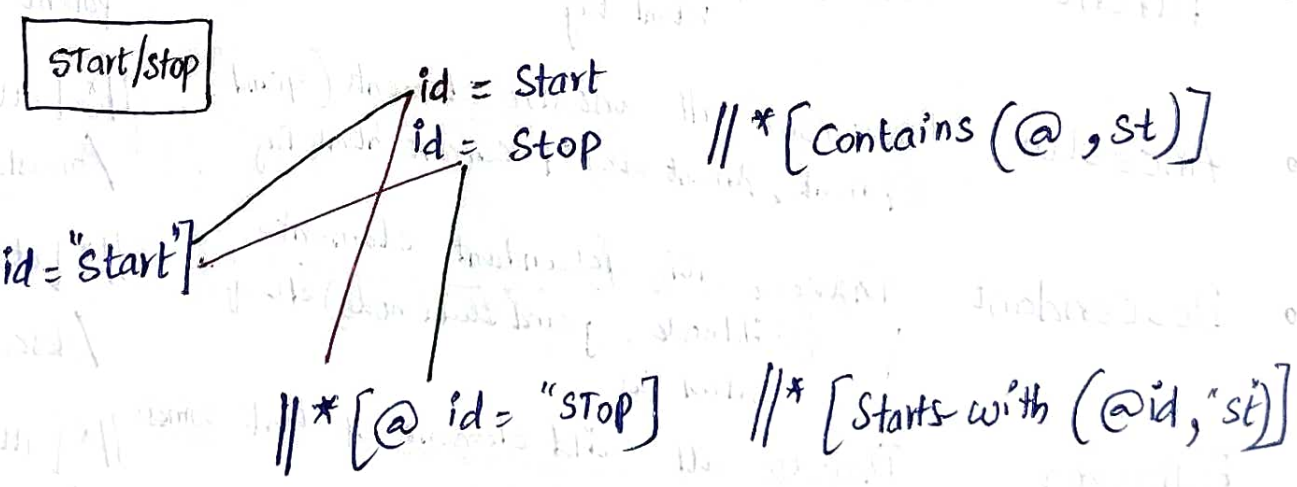
or

```
driver.find_element(By.XPATH, "//input[@id='small_search' or @type='text']).Send_keys("T-shirt")
```

AND

```
driver.find_element(By.XPATH, "//Button[@type='submit' and @class='button-I']).click()
```

Contains / Starts with



Contains

```
driver.find_element(By.XPATH, "//input*[contains(@id, 'search')]")
```

starts-with

```
driver.find_element(By.XPATH, "//button*[starts-with[@name, 'submit']]").click
```

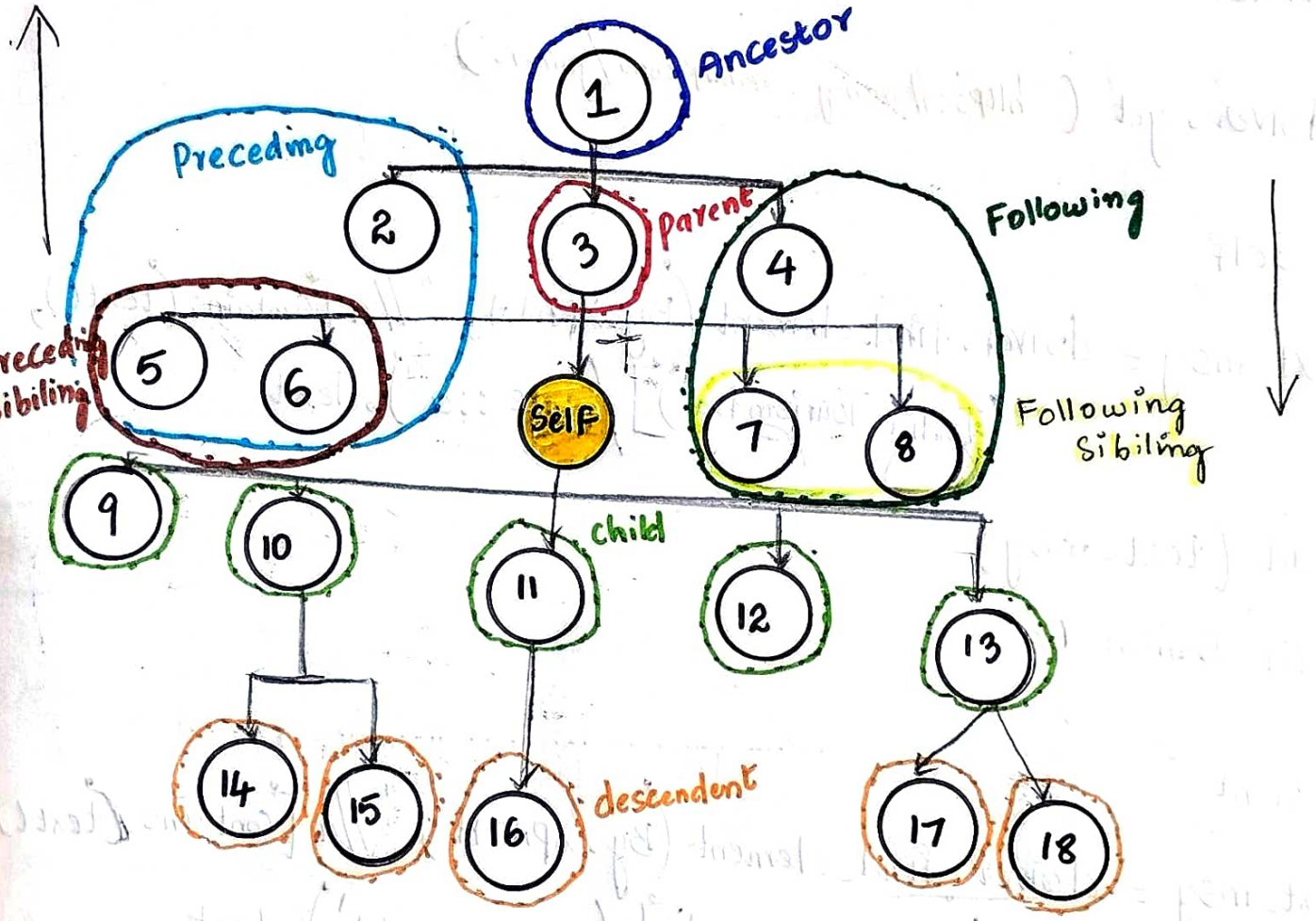
Text

```
driver.find_element(By.xpath, "//div['text()= 'category']").click()
```

Xpath axes

• child	Traverse all child elements of Current html tag	//*[attribute = 'value']/child::tagname
• Parent	Traverse Parent element of Current html tag	//*[attribute = 'value']/parent::tagname
• Ancestor	Traverse all ancestor elements (Grand parent, parent etc) of current html Tag	//*[attribute = 'value']/ancestor::tagname
• Descendant	Traverse all descendant elements (Childnode, grand child node) etc of html Tag	//*[attribute = "value"]/descent::tagname
• Following	Traverse all element of that comes after Current html tag	//*[attribute = value]/following::Tagname
• Following-Sibiling	Traverse from current Html tag to Next Sibling HTML tag.	//current html tag[@attribute = 'v'/Following-sibling::sibling tag[@attribute = 'value']

• Preceding	Traverse all nodes that comes before the Current html tag.	//*[attribute = 'value']/ preceding :: tagname.
• Preceding - Sibling	Traverse From Current Html Tag To Previous Sibling HTml Tag.	//Current html tag [@attribute = 'value']/preceding - sibling : Previous tag [@attribute = 'valu



" Xpath Axes "

Code

```
from Selenium import Webdriver
from Selenium.Webdriver.Common.by import By
from Selenium.webdriver.Chrome.Service import Service

Service_object = Service("C||User||chromedriver.exe")
driver = Webdriver.Chrome(Service = Service_object)
driver.get("https://money.reddif.com/gainers")
```

Self

```
text_msg = driver.find_element(By.XPATH, "//a[Contains(text(),
"India Tourism Da")]/self::a").Text

Print(test_msg)
```

o/p: India Tourism Da

Parent

```
text_msg = driver.find_element(By.XPATH, "//a[Contains(text(),
"India Tourism Da")]/parent::td').text

Print(text_msg)
```

o/p: India Tourism Da

Child

```
child = driver.find_elements(By.XPATH, "//a[contains(text(),
        'India Tourism Da')]/ancestor::tr/child::td")
Print(len(child)
```

o/p: 5

Ancestor

```
text_msg = driver.find_element(By.XPATH, "//a[contains(text(),
        'India Tourism Da')]/ancestor::tr").text
Print(text_msg)
```

o/p: India Tourism Da B 355.40 356.15 + 0.21

Descendent

```
Descendent = driver.find_elements[By.XPATH, "//a[contains(text()
        'India Tourism Da')]/descendant::*")
```

↳ /ancestor::tr

```
Print(len(Descendent)
```

o/p: 7

Following ↓

```
Following = driver.find_elements[By.XPATH, "//a[contains.(text()
        'India Tourism Da')]/ancestor::tr/Following::*")
Print(Len(Following)
```

o/p: 5055

Following-sibling

```
following sibling = driver.find_elements(By.XPATH, "//a[contains(text(),
        'India Tourism Da')]/ancestor::tr/following-sibling::*")

Print (Len (following sibling)
```

O/p: 618

Preceding

```
Preceding = driver.find_elements(By.XPATH, "//a[contains(text(),
        'India Tourism Da')]/ancestor::tr/preceding::*")

Print (Len (preceding)
```

Preceding-sibling

```
preceding sibling = driver.find_elements (By.XPATH, "//a[contains
        (text(), 'India Tourism Da')]/ancestor::tr/preceding
        Sibling::*")

Print (Len (preceding sibling)
```

O/p: 1541

```
Driver.close()
```

Commands

1. Application Commands
2. Conditional Commands
3. Browser Commands
4. Navigational Commands
5. Wait Commands.

⇒ Application Commands [Access Through "Driver"]

- get() - opening the application URL
- title - to Capture the title of Current Webpage
- Current_url - to Capture the Current url of webpage
- Page_Source - to Capture Source code of page.

Code

```
from Selenium import Webdriver
from Selenium.webdriver.common.by import By
from Selenium.Webdriver.Chrome.service import Service
Service_object = Service("C:\\user\\chromedriver.exe")
driver = Webdriver.Chrome(service = Service_object)
driver.get("http://opensource-demo.orangehrmlive.com")
```

Print (driver.title)

o/p: # ORANGE ORM

Print (driver.Current_url)

o/p: https://orangedemo-hrm.live

Print (driver.Page_source)

o/p: page source details.

⇒ Conditional Commands [Access Through "Web Element"]

- is_displayed () → Returns the boolean Values.
- is_enabled ()
- is_selected () → it is used For Radio buttons / checkboxes

CODE

driver.get ('https://demo.nopecommerce.com/Register')

is_displayed & is_enabled

Searchbox = driver.find_element (By.XPATH, "//input [@placeho
= 'Search store']")

Print ("display status:", searchbox.is_displayed ())

print ("enabled status:", Searchbox.is_enabled ())

o/p: display status : True

enabled status : True

is_selected

male_radiobutton = driver.find_element(By.XPATH, "//input[@Va
='M']")

Female_radiobutton = driver.find_element(By.XPATH, "//input[@
Value = 'F']")

Print ("Default parameters")

Print ("male radio :", male_radiobutton.is_selected())

Print ("Female radio :", Female_radiobutton.is_selected())

O/p :- Default parameter

male radio : False

Female radio : False

- male_radiobutton.click()

Print ("After clicking the male")

Print ("male radio :", male_radiobutton.is_selected())

Print ("Female radio :", Female_radiobutton.is_selected())

O/p : After clicking the male

male radio : True

Female radio : False

- Female_radiobutton.click()

Print ("after clicking Female")

Print ("male radio :", male_radiobutton.isSelected())

print ("Female radio :", Fe_male_radiobutton.is_selected())

o/p :- after clicking Female

male radio : False

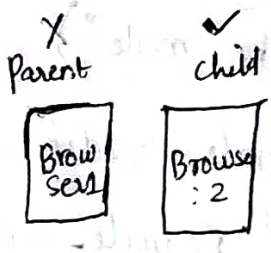
Female radio : True

⇒ Browser Commands [Access through "Driver"]

- close() - Close "Singe" browser Window [where driver focused]
- quit() - Close "Multiple" browser windows [this will kill the Proces

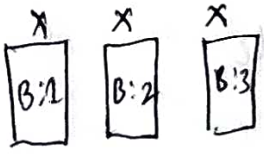
CODE

driver.close()



Here it closes Firstly open Brower [parent] & keeps the child Browser.

driver.quit()



cuts whole power and kill the Proces

⇒ Navigational Commands [Access Through "Driver"]

- back()
- forward()
- refresh()

```
driver.get("https://www.snapdeal.com/")
driver.get("https://www.amazon.com/")
driver.back() # snapdeal
driver.forward() # amazon
driver.refresh() # reloads page/browser
driver.quit()
```

→ Wait commands [synchronization] problem

- implicit wait
- Explicit wait
- time.sleep(time) → python. [Performance of script is very Poor]

  * if the element is not available within time mentioned, still there is chance of Expection.

implict wait

driver.implicitly_wait(10)

Advantages :

1. Single statement
2. Performance will not be reduced. [if the element is available with in the time, it proceed to excute further Statements

Dis Advantages :

1. If the element is not Available within the time mentioned, still there is a chance of getting Exception.

Code

```
from Selenium import Webdriver
from Selenium.webdriver.Commons.by import By
from Selenium.Webdriver.Chrome.Service import Sevice
```

```
Service_object = Service("c||user||chromedriver.exe")
driver = webdriver.Chrome(service = service_object)

driver.implicitly_wait(10)

driver.get("https://www.google.com")

driver.maximize_window()

    Search = driver.find_element(By.NAME, "q")

Search.send_keys("selenium")

Search.submit()

driver.find_element(By.XPATH, "//h3[text() = "Selenium"]").
                                    click()
```

Explicit Wait

* Explicit Wait works based on "Condition"

Advantages

* More effectively Works

Dis-Advantages

* Multiple places
* Feel Some difficulty

```
from Selenium import webdriver
from Selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as E

Service_object = Service("C:\\user\\chromedriver.exe")
driver = webdriver.Chrome(service = Service_object)
mywait = WebDriverWait(driver, 10, poll_frequency = 2,
                       ignored_exceptions = Exception)
driver.get("https://www.google.com/")
Search = driver.find_element(By.NAME, "q")
Search.send_keys("Selenium")
Search.submit()
Searchlink = mywait.until(EC.presence_of_element_located
             ((By.XPATH, "//h3[text()='Selenium']")))
Searchlink.click()
```

webdriver — time (seconds) — checking frequency (seconds) — ignore

• Handle "Checkbox"

```
driver.get("https://itera-qa.azurewebsites")
```

1. Select the Specific checkbox

```
driver.find_element(By.XPATH, "//div[4]//label[1]//input[1]").
```

Gives one Element: Monday ☑

2. Select the Multiple Checkbox

```
Checkbox = driver.find_elements(By.Xpath, "//input[@type =
            'checkbox' and Contains(@id, 'day')]")
Print(len(Checkbox)
```

Approach: 1

```
for i in range(len(checkbox)):
    checkbox[i].click()
```

o/p:
- monday ☑
- Tuesday ☑
- Wensday ☑
- Thursday ☑
- Friday ☑
- Saturday ☑
- Sunday ☑

Approach: 2

```
for i in checkbox:
    i.click()
```

3. Checking Multiple box with Choice

```
for i in checkbox:
    Weekdays = i.get_attribute("id")
    if Weekdays == 'Monday' or Weekdays = "Sunday":
        i.click()
```

Monday ☑ Wensday ☐ Friday ☐
Tuesday ☐ Thursday ☐ Saturday ☐
Sunday ☑

4. Select Last two boxes

```
for i in range (len (checkbox)-2, len (checkbox):
    checkbox[i].click()
```

∴ total number of elements − 2 = starting index

range (5,7) ⇒ 6,7

o/p : Monday ☐ Wensday ☐ Friday ☐ Sunday ☑
Tuesday ☐ Thursday ☐ Saturday ☑

5. Select First two boxes

```
for i in range (len(checkbox)):
    if i<2:
        checkbox[i].click()
```

Monday ☑
Tuesday ☑

6. Clear all check box

approach I :I

```
for i in checkbox:
    if i.is_selected():
        i.click()
```

Links

1. internal link
2. External link
3. broken link : which doesn't have Target page
   greater Than 400
   >= 400
   < = 300 (good)

- click on link [internal]

```
driver.get("https://demo.nopcommerce.com/")
# link_text
driver.find_element(By.LINK_TEXT, "Digital Downloads").click()
# partial_link_text
driver.find_element(By.PARTIAL_LINK_TEXT, "Digital").click
```

o/p :- #95

- Find all links on the Page [External]

```
links = driver.find_elements(By.TAG_NAME, "a")
Print(len(links)
```

o/p: #95

```
for i in links:
    Print(i.text)
```

names of Links

- **Broken links**

```
import requests

driver.get("https://www.deadlinkcity.com")

driver.maximize_window()

allLinks = driver.find_elements(By.TAG_NAME, "a")

Print (len(allLinks)

count = 0

for i in allLinks:

    Url = i.get_attributes("href")

    try:

        response = requests.head(Url)

    expect:

        None

    if respone.status_code >= 400:

        Print (Url, "broken links")

        count = +1

    else:

        Print (Url, "Valid links")

Print ("broken links", count)
```

Find_Element V/S find_Elements

- Difference between "find_element" & "find_elements"

```
driver.get("https://demo.nopcommerce.com/")
```

find_element() → Returns Single Webelement

1. Locater matching with Single webelement

```
element = driver.find_element(By.XPATH, "//input[@id='small-searchterms']")
element.send_keys("Apple") # Apple
```

2. Locater matching with Multiple webelement

```
element = driver.find_element(By.XPATH, "//div[@class='footer']//a")
print(element.text) # prints first link from the footer : "Sitemap"
```

3. Element not available then throw 'No Such Element Exception'

```
login_element = driver.find_element(By.XPATH, "//div[@'log']
login_element.click() # No Such Element Exception Found
```

find_elements() - "Returns Multiple Webelements.

1. Locater matching Single webelement

```
elements = driver.find_elements (By.XPATH, "//input[@id='small-search-terms']")

Print (Len (elements))  # find_elements gives Webelement in : List

elements[0].Send_keys ("Apple")
```

2. Locater matching with Multiple web element

```
elements = driver.find_elements (By.XPATH, "//div[@class='Footer']//a")

Print (Len (elements))  #23 webelements

Print ((elements[1].text))

(or)

for i in elements:
    Print (i.text)   # gives names of 23 elements
```

3. Element No Available : Zero [0]

```
elements = driver.find_elements (By.LINK_TEXT, "log")

Print (Len (elements))

# gives [0] in List
```

But, doesn't give "No such element Found"

text V/s get-attribute

- text → Returns 'inner text' of the element
- get_attribute () → Returns Values of any attribute of web element

Ex:-
- get_attribute ('value')
- get_attribute ('Id')
- get_attribute ('name')

```
driver.get ("https://admin-demo.nopcommerce")
emailbox = driver.find_element (By.XPATH, "//input [@value = "admin@yourstore.com"]")
emailbox.clear()
emailbox.Send_keys ("abcdef@gmail.com")
Print ("result of test", emailbox.text)
Print ("result of get_attribute", emailbox.get_attribute ("value"))
```

<input id='123', name='xyz', email=></inp

o/p:- result of test : None [# Because Text, Has No inner text]
result of get_attribute : abcdef@gmail.com.

Button

```
button = driver.find_element (By.XPATH, "//button [@type = "Submit"]")
Print ("result of test", button.text)
Print ("result of get_att", button.get_attribute ('value'))
```

o/p:- result of Text : LOGIN [# Here Text contains Inner Text]
result of get_att :

Handle Dropdowns

```
from selenium.webdriver.support.select import Select

driver.get("https://www.opencart.com/index.Register")

drop_element = Select(driver.find_element(By.XPATH,
                "//select[@id='input-country']"))
```

1. Select option from dropdown

```
.drop_element.Select_by_visible_text("India")

.drop_element.Select_by_index(10) # argentina

.drop_element.Select_by_value("3") # algeria
```

2. Select all options from dropdown

```
alloptions = drop.element.options

print("selecting all options:", len(alloptions))

for i in alloptions:
    print(i.text)
```

o/p: # prints all options

3. Select option with inbuild-options

```
for i in alloptions:
    if i.text == "India":
        i.click()
        break
```

4. If there is "no Select" option

```
alloptions = driver.find_elements(By.XPATH, "//*[@id.
                                  input_country"]/option")

Print(len(alloptions)
```

Frames / iFrames :- Switch_to.frame()

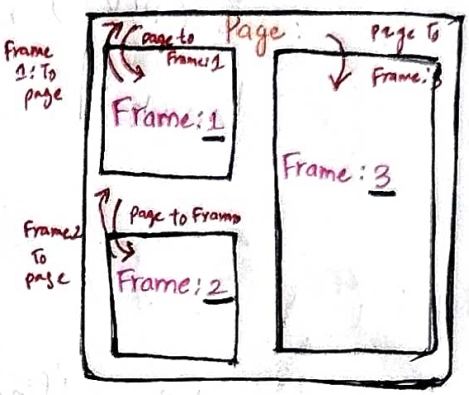
→ Switch_to.frame(name of the frame)

→ Switch_to.frame(id of the frame)

→ Switch_to.frame(Webelement)

→ Switch_to.frame(0)

→ Switch_to.default_content()



code:

```
driver.get("https://www.selenium.dev/selenium/docs/api")
driver.maximize_window()
```

1st Frame.

```
driver.switch_to.frame("Package List Frame")  # name of the Frame.
driver.find_element(By.LINK_TEXT, "org.openqa.selenium").click()
driver.switch_to.default_content()  # go back To Main page.
```

2nd Frame

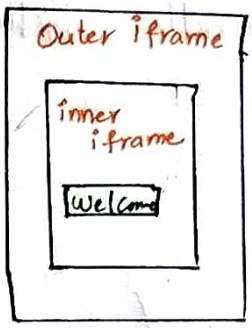
```
driver.switch_to.frame("Package Frame")
driver.find_element(By.LINK_TEXT, "WebDriver").click()
driver.switch_to.default_content()  # go back To Main page
```

3rd Frame.

driver. Switch_to. Frame ("Class Frame")

driver. find_element (By. XPATH, "/html/body/header/nav/div). Clic

Inner Frames



driver. get ("http://demo.automationtesting.in/Frames")

driver. maximize_window ()

driver. find_element (By. XPATH, "//a[normalize-space() = 'IFrame with in an IFrame']"). click()

Outer Frame

Outer Frame = driver. find_element (By. XPATH, "//iframe[@scr = "Multiple Frames. html"]")

driver. switch_to. frame (Outer Frame) # webelement

Inner Frame

inner Frame = driver. find_element (By. XPATH, "/html/body/section/div/iframe")

driver. switch_to. frame (inner Frame)

Enter Element

driver. find_element (By. XPATH, "//input[@type = 'text']"). Send_Keys ("Welcome")

driver. switch_to. Parent_frame () # directly switch to Parent frame (outer frame)

Browser Windows

Switch_to.window (Window ID)

Current_window_handle : Return WindowID of "Single browser" window

window_handle : Return Window ID of "Multiple browser" windows.

code :

```
driver.get ("https://rahulshettyacademy.com/AutomationPractice/")
```

Single Window

```
Window id = driver.current_window_handle  # single window / current
Print (window id)  # Prints Unique Id Everytime
```

CDWindow - 81ECCBB02E9BB54F88B4D520C7CACEA2

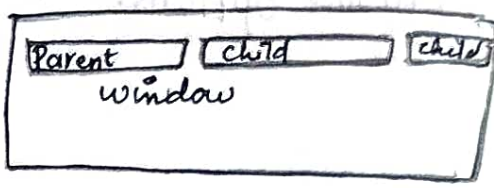
Multiple Windows

Approach #1

```
driver.find_element (By.XPATH, "//button[@id='openwindow']").click()
WindowIDs = driver.window_handles  # Multiple windows.
                                   # Gives list of Elements
Parent window = WindowIDs[0]  # CD-window - 7C4C...
Child Window = WindowIDs[1]  # CD-window - E1632...
Print (Parent window, Child window)
```

driver.switch_to.window(Parent Window)

Print("Parent window title", driver.title)



o/p: Pratice page

driver.switch_to.window(child Window)

Print("child window title", driver.title)

- **Approach#2**

```
for i in WindowIDs:
    driver.switch_to.window(i)
    print(driver.title)
```

To close particular Windows

```
for i in windowIDs:
    driver.switch_to.window(i)
    if driver.title == "Partice page":   # close particular window
        driver.close()
```

or), driver.title == "xyz"

```
driver.quit()
```

Chrome options: Notification Popup.

```
driver = webdriver.Chrome (service = Service, Object,
                           options = ops)
ops = webdriver.ChromeOptions()
ops.add_argument ("--disable-notifications")
driver.get ("https://www.where-am-i.co/")
```

Web Table / HTml Table

- Static WebTable :- Data remains Same Everytime
- Dynamic WebTable :- Data always changes (or) updated.

html : < Table

< tbody > → Table body

< tr > -- < /tr > Table row

Table data < td > Hi < /td >

< td > Hello < /td >

< td > Mama < /td >

< /tbody >

< /table >

Table header

< th > BookName < /th >

< th > Author < /th >

Column

HTML Table

Book Name	Author	Subject	Price
Learn Selenium	Amit	Selenium	300
Learn Java	Mukesh	Java	500
Learn Js	Animesh	Java script	300
Master in Selenium	Mukesh	Selenium	3000
Master in Java	Amod	JAVA	2000
Master in J.S	Amit	Java Script	1000

rows

Code:

```
driver.get("https://Testautomationpratice.blogspot.com/")
```

1. Count no. of rows and columns in table

```
rows = len(driver.find_elements(By.XPATH, "//table[@name='BookTable']//tr"))

columns = len(driver.find_elements(By.XPATH, "//table[@name='BookTable']//tr[1]/th"))

Print(rows, columns)
```

o/p: 7, 4

2. Read specific row & column Data

```
data = driver.find_element(By.XPATH, "//table[@name='BookTable']//tr[5]/td[1]").text
```

row — 5, column — 1

```
Print(data)
```

o/p: Master in Selenium

3. Read all data from table

```
for row in range(2, rows+1):
    for col in range(1, columns+1):
```

```
data = driver.find_element(By.XPATH, "//table[@name='Book Table']//tr["+str(row)+"]/td["+str(col)+"]").text
Print(data, end=" ")
Print()
```

O/P :- Total Table

4. read the data according to Condition

Condition = [list books who's name == Mukesh]

```
for row in range(2, rows+1):
    authorname = driver.find_element(By.XPATH, "//table[@name='book Table']/tbody/tr["+str(row)+"]/td[2]").text
```

↳ column(2) = author name

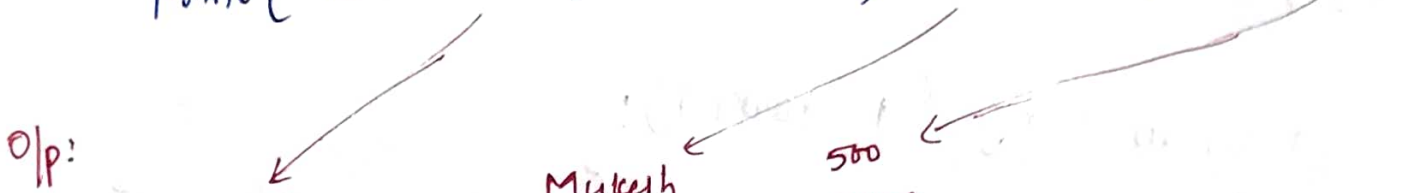
```
    if authorname == "Mukesh":
        bookname = driver.find_element(By.XPATH, "//table[@name='book Table']/tbody/tr["+str(row)+"]/td[1]").text
```

↳ colum(1) = book name

```
        Price = driver.find_element(By.XPATH, "//table[@name='book Table']/tbody/tr["+str(row)+"]/td[4]").text
```

↳ column(4) = price

Print (bookname , " " , authorname , " " , Pri



O/P:

Learn Java Mukesh 500
Master in Selenium Mukesh 3000

Dynamic Table

driver . get ("https://openOrange.html")

Admin → User Management → Users

driver . find_element (By.XPATH, "//*[@id= menu_admin_view
Admin Module']/b"). click()

driver . find_element (By.XPATH, "//*[@id = menu_admin_
UserManagement']"). click()

driver . find_element (By.XPATH, "//*[@ id= menu_admin_
ViewSystem Users']"). click()

total rows numbers of Table

rows = Len (driver . find_elements (By.XPATH, "//Table [@id=
'resultTable'] // tbody /tr"))

Print ("total number of rows in Table:", rows)

```
Count = 0

for row in range (1, rows+1):
    Status = driver.find_element (By.XPATH, "//table [@id=
             'resultTable']/tbody/tr [" +str(row)+ "]/td[5]").
             text
    if status == "Enabled":
        Count = Count + 1

Print ("Total no. of User:", rows)
Print ("No. of Enabled User":, Count)
Print ("No. of disabled User:", (rows - count))

driver.close ()
```

Date picker :

* Standard : Same in Every Webpage.
* Non - Standard [customized] : different From One Web From another webpage

```
driver.get ("https://jqueryui.com/datepicker")
```

1. Standard :- `driver.switch_to.frame (0)`

```
driver.find_element (By.XPATH, "//input [@id='datepicker']").
        Send_keys ("07/03/1997")  # mm/dd/yy
```

2. Non - Standard

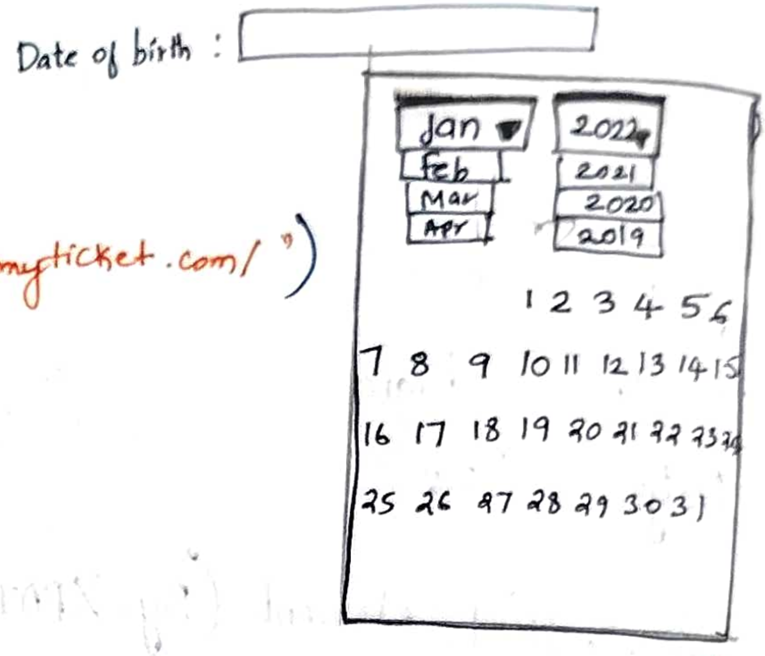
Date : #mm/dd/yyyy


```
year = "2022"
month = "March"
date = "22"

driver.find_element(By.XPATH, "//input[@id='datepicker']").click()

While True:
    mn = driver.find_element(By.XPATH, "//span[@class='ui-datepicker-month']").text

    yr = driver.find_element(By.XPATH, "//span[@class='ui-datepicker-year']").text

    if mn == month and yr == year:
        break
    else:
        driver.find_element(By.XPATH, "//span[@class='ui-icon ui-icon-triangle-w']").click()

dates = driver.find_elements(By.XPATH, "//Table[@class='ui-datepicker-calender']/tbody/tr/td/a")

for i in dates:
    if i.text == dates:
        i.click()
        break

driver.close()
```

picks Month and year

picks dates

3. dropdown datepicker




```
driver.get("https://www.dummyticket.com/")
driver.implicitly_wait(20)
driver.maximize_window()

# click to go another page.
driver.find_element(By.XPATH, "/html[1]/body[1]/div[1][div]1").click

# select date of birth /By dropdown.
driver.find_element(By.XPATH, "//input[@id='dob']").click()

month = Select(driver.find_element(By.XPATH, "//select[@aria-label=
                  'Select month']"))
month.select_by_visible_text('Jan')    # selects the month

year = Select(driver.find_element(By.XPATH, "//select[@aria-lab
               ='Select year']"))  # selects year

dates = driver.find_elements(By.XPATH, "//table[@class='ui-date
              Picker-Calendar']//tbody/tr/td/a")  # select date

for i in dates:
    if i.text == "20":
        i.click()
        break
```

Mouse Operations

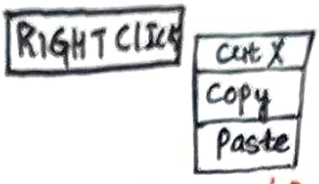
* ActionChains()

1. Mouse hover --> move_to_element (element)
2. Right Click --> Context_click (element)
3. Double Click --> double_click (element)
4. Drag and Drop --> drag and drop (element)

Mouse hover

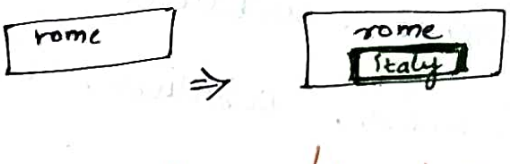
```
from Selenium import Webdriver
from Selenium.webdriver import ActionChains
from Selenium.webdriver.chrome.Service import Service
from Selenium.webdriver.Common.by import By

Service_object = Service("C//user//chromedriver.exe")

driver = webdriver.Chrome(service = Service_object)

driver.get("https://jqueryui.com/datepicker/")

demo = driver.find_element(By.XPATH, "//a[normalize-space()='Demos']")

act = ActionChains(driver)

act.move_to_element.click().perform()
```

Right click :



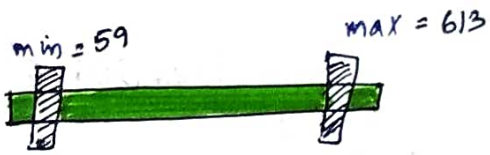

```
driver.get("https://swisnl.github.io/jQuery-contextMenu/demo.html")

button = driver.find_element(By.XPATH, "//span[@class='context-menu-one btn btn-neutral']")

act = ActionChains(driver)
act.context_click(button).perform() # perform Right Click

driver.find_element(By.XPATH, "//span[@normalize-space()='copy']").click # copy

driver.switch_to.alert.accept() # closes Alert Window
```

Drag & Drop :




```
driver.get("https://www.dhtmlgoodies.com/scripts/drag-down-html2")
driver.implicitly.wait(10)

act = ActionChains(driver)

rome = driver.find_element(By.ID, "box6")
italy = driver.find_element(By.ID, "box106")

act.drag_and_drop(rome, italy).perform()
```

Sliders




```
driver.get("https://www.jqueryscript.net/demo/price-range")

min = driver.find_element(By.XPATH, "(//span[1]")
max = driver.find_element(By.XPATH, "(//span[2]")

Print("Before draging sliders")

Print(min.location)  # {"x": 59, "y": 251}
Print(max.location)  # {"x": 613, "y": 251}

act = ActionChain(driver)

act.drag_and_drop_by_offset(min, 100, 0).perform()

act.drag_and_drop_by_offset(max, -40, 0).perform()

Print("After draging sliders")

Print(min.location)  # {"x": 159, "y": 251}

Print(max.location)  # {"x": 574, "y": 251}
```

Scrolling Page :

By Java Script's

```
driver.get("https://www.Countries-of-the-World.com/Flags-of-world")
driver.maximize_window()
```

1. Scroll down page by Pixel

```
driver.excute_Script("window.scrollBy(0,3000)","")
Value = driver.excute_script("return window.pageYOffset;")
Print(Value) #3000
```

2. Scroll down by visible_text

```
flag = driver.find_element(By.XPATH,"//td[normalize=INDIA()]
driver.excute_script("arguments[0].ScrollIntoView();",flag)
Value = driver.excute_script("return window.pageYOffset;")
Print(Value) #5003.20019
```

3. Scroll down page till End

```
driver.excute_script("Window.scrollBy(0,document.body.
                      ScrollHeight)")
Value = driver.excute_script("return window.pageYOffset;")
Print(Value) #5998.3999
```

4. Scroll back to Starting Point

```
driver.excute_script("window.scrollBy(0,-document.body.scrollHeight)")
```

KeyBoard Actions:

```
Driver = webdriver.chrome(Service = Service_obj)

Driver.get("https://text-Compare.com/")
# driver.maximize_window()
Input1 = driver.find_Element(By.XPATH."//textarea[@id='input Text 1']")
Input2 = driver.find_Element(By.XPATH."//textarea[@id='input Text 2']")
Input1.Send_Keys('Welcome ra mawa')
```

#ctrl A

```
act = Actionchains(driver)
# act.Key_down(Keys.CONTROL)
# act.Send_Keys("a")
# act.Key_up(Keys.CONTROL)
# act.perform()
act.Key_down(Keys.CONTROL).Send_Keys("a").Key_up(Keys.CONTROL).perform()
```

ctrl c

```
# act.key_down(Keys.CONTROL)
# act.send_keys("a")
# act.key_up(Keys.CONTROL)
# act.perform()
act.key_down(Keys.CONTROL).send_keys("c").key_up
(Keys.CONTROL).perform()
```

ctrl TAB — act.send_keys(Keys.TAB).perform()

```
# act.key_down(Keys.CONTROL) --> # ctrl P
# act.send_keys("a")
# act.key_up(Keys.CONTROL)
# act.perform()
act.key_down(Keys.CONTROL).send_keys("v").key_up
(Keys.CONTROL).perform()
```

Download file.

```
import os
locations = os.getcwd()
def chrome_setup():
    from selenium.webdriver.chrome.service import Service
Service_obj = Service("c:\\Users\\User\\Drivers\\chromedriver.exe")
```

download in desired location.

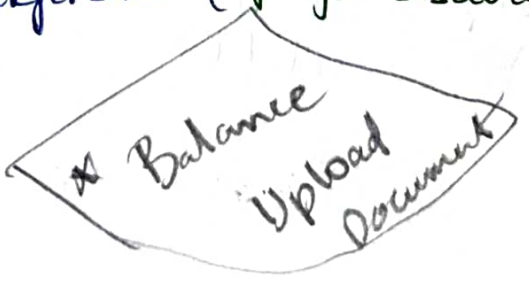
```
prefrences = {"download.default_directory": location}
Ops = Webdriver.ChromeOptions()
Ops.add_experimental_option('prefs', prefrences).
driver = Webdriver.Chrome(Service = service_obj, options = Ops)
retuen driver
def Edge_setup():-
    from Selenium.webdriver.Edge.service import Service
    Service_obj: Service = Service("C://users//user// Drivers //msedge
    driver.exe").
    # download in desired loction.
    prefrences = {'download.default_directory': location}
    Ops = webdriver.EdgeOptions()
    Ops.add_experimental_option('prefs'.prefrences)
    driver = Webdriver.Edge(Service = Service_obj, options =ops)
    retuen driver
# Automation Code:-
# driver = chrome_setup()
driver = Edge_setup()
driver.get("https://filesamples.com/formats/docx")
driver.maximize_window()
driver.find_Element(By.XPATH"//div[@id='output']//div[2]//a[1]).
                                                    click()
```

def firefox_setup():-

```
from selenium.webdriver.firefox.service import Service
Serv_obj = Service("C:\Drivers\geckodriver-v0.29.1-win64\geckodriver.exe")

# settings
Ops = webdriver.FireFoxOptions()
Ops.set_preferences("Browser.helperApps.neverAsk.saveToDisk",
                    "application/msword")
Ops.set_preference("browser.download.manager.showWhenStarting", False)
Ops.set_preference('Browser.download.folderList', 2) # 0-desktop
                                                     1-downloads folder 2-Desired loc
Ops.set_preference("browser.download.dir," location).
driver = webdriver.Firefox(Service=Serv_obj, options=ops)
return driver
```

Pdf. download.

```
Prefrences = {"download.default_directory": location, "plugins.alw
              -ays-open-pdf-externally": True}.
Ops.set_preference("browser.helperApps.neverAsk.saveToDisk",
                   "application/pdf") #Mine tag.
Ops.set_preference("pdfjs.disabled", True) # For pdf.
```



Boostrap Drop Down

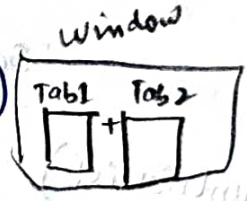
```
driver.get("https://www.dummyticket.com/")
driver.maximize_window()
Selectticket = driver.find_element(By.XPATH, "(//a[contains(@class,
                                    button)[2]")
# Scrolling down
driver.excute_Script("arguments[0].scrollIntoView();",
                                    Selectticket)
# Click on option
driver.find_element(By.XPATH, "//span[@id = "select2-billing
                                    -country]").click()
# Bootstrap Drop Down
Countrieslist = driver.find_elements(By.XPATH, "//ul[@id=
                    "select2-billing-country]//li")
                    ↳ Gives list on options
Print(Len(countrieslist)
o/p: 249
for i in Countrieslist:
    if i.text == "India":
        i.click()
        break
```

ScreenShot

```
driver.get("facebook.com")
driver.save_screenshot("C://user//selenium//home.png")
                        # Total path + Filename
import os
driver.save_screenshot(os.getcwd()+"//home.png")
                        # By importing Os 'module'.
driver.get_screenshot_as_file(os.getcwd()+"//home.png")
driver.get_screenshot_as_base64()
```

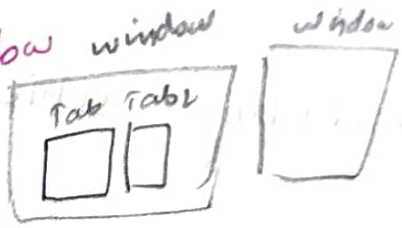
Tabs And Windows

```
driver.get("http://demo.nopCommerce.com")
```




```
reglink = Keys.CONTROL + Keys.RETURN
driver.find_element(By.LINK_TEXT, "Register").send_keys(reglink)
# Open New Tab & Switches New Tab
driver.get("https://facebook.com")
driver.switch_to.new_window("Tab")
driver.get("https://instagram.com")
```

open New window & Switches to new-window




```
driver.get("https://facebook.com")
driver.switch_to.new_window("window")
driver.get("https://instagram.com")
```

Handling Cookies [data stored in web browser]

```
driver.get("https://demo.nopCommerce.com/")
```

number of cookies in page.

```
cookiee = driver.get_cookies()
Print(len(cookiee)
```

o/p: 3

Adding New Cookiee to page.

```
driver.add_cookie({"name": "MyCookies", "Value": "1234"})
cookies = driver.get_cookies()
print("after adding new cookie:", len(cookies))
```

o/p: 4

deleting cookie

```
driver.delete_cookie("Mycookie")
cookies = driver.get_cookies()
print("after deleting cookie:", len(cookies))
```

o/p: 3

After deleting "all" cookies.

```
driver.delete_all_cookies()
cookies = driver.get_cookies()
print(len(cookies))
```

o/p: 0

Headless Mode: "without opening browser: it gives result".

```
from Selenium import Webdriver
def headless_chrome():
    from selenium.webdriver.chrome.service import Service
    service = Service("C:\\usr\\chromedriver.exe")
    ops = Webdriver.ChromeOptions()
    ops.headless = True
    driver = Webdriver.Chrome(service=Service, options=ops)
    return driver
```

Same For: Edge, Firefox()

```
driver = headless_chrome()
driver.get("facebook.com")
print(driver.title)  # facebook
print(driver.URL_current)  # https://facebook.com.
driver.close()
```

Data driven Testing

Openpyxl ⇒ We Can Work with Excel Files (.xlsx)

25/03/23