#navigation commands

driver.get("https://rahulshettyacademy.com/AutomationPractice/") #opens a url in browser

driver.back()

driver.forward()

driver.refresh()

#Alert handling

alt = driver.switch\_to.alert *#switching to pop-up*alt.accept()

# Press the Cancel button

alert.dismiss()

Information:

**Isdisplayed():**

# Navigate to the url

driver.get("https://www.selenium.dev/selenium/web/inputs.html")

# Get boolean value for is element display

is\_email\_visible = driver.find\_element(By.NAME, "email\_input").is\_displayed()’

IS\_enabled():

# Navigate to url

driver.get("https://www.selenium.dev/selenium/web/inputs.html")

# Returns true if element is enabled else returns false

value = driver.find\_element(By.NAME, 'button\_input').is\_enabled()

**Is Selected():**

This method determines if the referenced Element is Selected or not. This method is widely used on Check boxes, radio buttons, input elements, and option elements.

Returns a boolean value, True if referenced element is selected in the current browsing context else returns false.

# Navigate to url

driver.get("https://www.selenium.dev/selenium/web/inputs.html")

# Returns true if element is checked else returns false

value = driver.find\_element(By.NAME, "checkbox\_input").is\_selected()

## Tag Name:

It is used to fetch the [TagName](https://www.w3.org/TR/webdriver/" \l "dfn-get-element-tag-name) of the referenced Element which has the focus in the current browsing context.

# Navigate to url

driver.get("https://www.selenium.dev/selenium/web/inputs.html")

# Returns TagName of the element

attr = driver.find\_element(By.NAME, "email\_input").tag\_name

**Size and Position:**

It is used to fetch the dimensions and coordinates of the referenced element.

The fetched data body contain the following details:

* X-axis position from the top-left corner of the element
* y-axis position from the top-left corner of the element
* Height of the element
* Width of the element

# Navigate to url

driver.get("https://www.selenium.dev/selenium/web/inputs.html")

# Returns height, width, x and y coordinates referenced element

res = driver.find\_element(By.NAME, "range\_input").rect

Window handling:

#command to open a new tab in current window and switch to new window

driver.switch\_to.new\_window('tab')

#command to open a new window and switch to new window

driver.switch\_to.new\_window('window')

#closing a window or tab

driver.close()

#switch back to old tab or window

original\_window = driver.current\_window\_handle

driver.switch\_to.new\_window('window')

driver.switch\_to.window(original\_window)

#window management

width = driver.get\_window\_size().get('width')  
height = driver.get\_window\_size().get('height')  
print(width, height)

#set window size

driver.set\_window\_size(1024, 768)

#maximize

driver.maximize\_window()

#minimize

driver.minimize\_window()

#fullscreen

driver.fullscreen\_window()

#take screenshot

driver.save\_screenshot('./image.png')

## Get CSS Value:

Retrieves the value of specified computed style property of an element in the current browsing context.

# Navigate to Url

driver.get('https://www.selenium.dev/selenium/web/colorPage.html')

# Retrieves the computed style property 'color' of linktext

cssValue = driver.find\_element(By.ID, "namedColor").value\_of\_css\_property('background-color')

#element screenshot

ele = driver.find\_element(By.CSS\_SELECTOR, 'h1')

driver.save\_screenshot('./image.png')

**Keyboard actions:**

Note: this keyboard actions will work on web elements only not on directly browser like when you want to ctrl+tab to switch to new tab pr window we need to use driver.switch\_to.new\_window(‘tab’) or driver.switch.to\_currentwindow like that

Examples:

When we want to use two keys at a time we will use keyup and down to hold one key and to select another key

If we want to use only one key we will send that button directly using send\_keys

To do ctrl+a when we did some addition of messege in text field

a\_ch = ActionChains(driver)  
a\_ch.key\_down(Keys.CONTROL).send\_keys('a').key\_up(Keys.CONTROL).perform()

To do ctrl+c when we want to copy some messege

a\_ch.key\_down(Keys.CONTROL).send\_keys('c').key\_up(Keys.CONTROL).perform()

To got to next field using TAB button

a\_ch.send\_keys(Keys.TAB).perform()

To submit clicking ENTER

a\_ch.send\_keys(Keys.ENTER).perform()