### ****Task 2:****

**Locators for web elements in Selenium**

* By CSS ID: find\_element\_by\_id
* By CSS class name: find\_element\_by\_class\_name
* By name attribute: find\_element\_by\_name
* By DOM structure or xpath: find\_element\_by\_xpath
* By link text: find\_element\_by\_link\_text
* By partial link text: find\_element\_by\_partial\_link\_text
* By HTML tag name: find\_element\_by\_tag\_name

While all these locators return single elements, one may use the .find\_elements() method to find multiple elements.

****Locate Elements by CSS ID:****

This is by far the simplest method of locating an element. The CSS ID, stored in the id attribute of an HTML DOM element, is unique for every element in the page by design. Thus, an ID can uniquely identify an element.

To use this feature, one needs to call the .find\_element\_by\_id() method of the webdriver class. Here is the usage for it.

from selenium import webdriver

driver = webdriver.Chrome('./chromedriver')

driver.get("https://www.python.org")

search\_bar = driver.find\_element\_by\_id("id-search-field")

If there is no DOM element with the ID that one is searching for, a No Such Element Exception is raised, which one can account for, by using a try-catch block.

Theoretically, every DOM element on a page should have a unique ID. However, in real life, one does not commonly observe this. Most elements may not have an ID or may encounter two elements with the same ID. In such cases, one needs to use a different strategy to identify a DOM element uniquely.

### ****Locate Elements by CSS Class Name:****

A second strategy of locating elements on a page is to search by the class name. The class name is stored in the class attribute of an HTML tag. By design, a CSS class applies to a group of DOM elements. The .find\_element\_by\_class\_name() method only returns the first element with the matching class. It raises a No Such Element Exception if there is no element with the given class name. Here is how to use the method in the driver.

from selenium import webdriver

driver = webdriver.Chrome('./chromedriver')

driver.get("https://www.python.org")

# Returns first element with matching class

first\_search\_bar = driver.find\_element\_by\_class\_name("id-class-name")

### ****Locate Elements by Name Attributes:****

In HTML5, form elements often have a name attribute associated with them. The .find\_element\_by\_name() method only returns the first element with the matching class. If there are multiple elements of the same name, the first matched element would be returned. No matching elements result in a No Such Element Exception error.

Consider the following form:

<form id="loginForm"><input name="name" type="text" value="First Name" /><input name="name" type="text" value="Last Name" /><input name="email" type="text" value="Business Email" /><input name="password" type="password" /><input name="continue" type="submit" value="Sign Me Up" /></form>

The following code returns the email form element.

email\_input = driver.find\_element\_by\_name("email")

However, the following code only returns the first name form element.

name\_input = driver.find\_element\_by\_name("name")

Using the .find\_element\_by\_name() method, it is not possible to get to the last name input form field in the example. This brings us to the next locator.

### ****Locate Elements by DOM structure or XPath:****

If one has failed to identify an element by ID, class or name, one would need to locate the element through its XML path. This process may also be implemented while reading an XML document. In this tutorial, we explore the use of relative paths, as absolute paths are prone to errors with the slightest change in the HTML structure.

We will use the .find\_element\_by\_xpath() method to locate an appropriate element in the document. The argument that the .find\_element\_by\_xpath() method takes is the path to the element.

To find the email input field in the above example of an HTML form, one may use the following code:

email\_input = driver.find\_element\_by\_xpath("//form[input/@name='email']")

This code snippet searches for the first form element of the page. Within this form, it searches for an input with the name which equals the value email, thus narrowing down to the required element.

Next, let us try to locate the first and last names input element of the form above.

first\_name = driver.find\_element\_by\_xpath("//form[@id='loginForm']/input[1]")

last\_name = driver.find\_element\_by\_xpath("//form[@id='loginForm']/input[2]")

The method first searches for a form with the ID login form and then selects the first and second input elements of the form as the first and last names.

### ****Other Single Element Locators:****

In addition to the popular methods that we have discussed, there are a few other element locators in the Selenium WebDriver that testers may wish to explore.

One can locate elements by their HTML tag name using the .find\_element\_by\_tag\_name() method.

page\_heading = driver.find\_element\_by\_tag\_name('h1')

One can also search for a hyperlink element using the link text. One can either use the .find\_element\_by\_link\_text() method to search for the exact link’s text, or .find\_element\_by\_partial\_link\_text() method to search for partial text.

# Exact Link Text

click\_here\_link = driver.find\_element\_by\_link\_text('Click Here')

# Partial Link Text

click\_here\_link = driver.find\_element\_by\_partial\_link\_text('Click')

### ****Locate Multiple Elements:****

In this tutorial, we have discussed methods that locate only single elements. One may want to select a group of elements and then iterate through them. The .find\_elements() method helps in finding multiple elements in the DOM structure.

Here are common examples of the usage of the .find\_elements() method. To find all input elements of a form with ID loginForm, use the following snippet –

from selenium.webdriver.common.by import By

all\_inputs = driver.find\_elements(By.XPATH, '//form[@id='loginForm']/input')

To locate all elements with a class name, use the following code –

from selenium.webdriver.common.by import By

all\_elements = driver.find\_elements(By.CLASS\_NAME, 'my-css-class')

In addition to this, the By class has the following attributes:

* By.ID: search using the CSS ID
* By.LINK\_TEXT: search using exact link text
* By.PARTIAL\_LINK\_TEXT: search using partial link text
* By.NAME: search using the name attribute
* By.TAG\_NAME: search using the HTML tag name

### ****Final Thoughts on Locators in Selenium:****

With this, we come to the end of the tutorial on locators in Selenium using Python. We discussed a variety of approaches to select elements within an HTML page. We first looked at single element selectors and then moved on to multiple element selectors in the Selenium WebDriver.