**Installation:**

* Download from <https://www.python.org/downloads/> and install (it will install Interpreter)
* Install IDE for python i.e. Pycharm from <https://www.jetbrains.com/pycharm/>
* Python has it’s own IDLE to write simple programs. It gets installed when you install Python
* Python gets installed at C:\Users\DELL\AppData\Local\Programs\Python by default
* Note that PIP gets installed when you install Python. PIP is the standard package manager for python. It allows you to install and manage additional packages that are not part of python standard library.
* **For Selenium,**
* you just have to run below command in command prompt to get install Selenium for Python. That’s it. Selenium package will get injected into python folder.

C:/> **pip install selenium**

If you wanted to see installed selenium version and location, use below command

C:/> **pip show selenium**

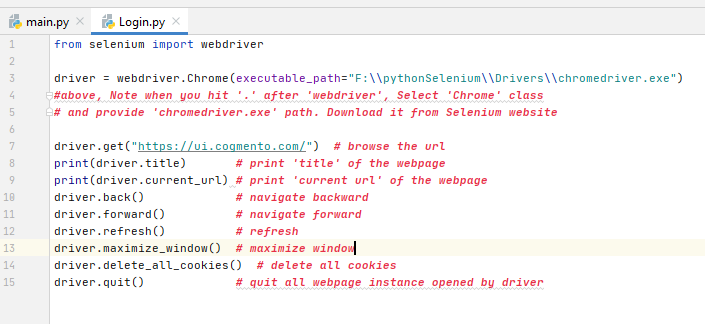
If you wanted to upgrade installed selenium, use below command

C:/> **pip install -U selenium**

* PIP stands for either "Pip Installs Packages" or "Pip Installs Python
* **READ BELOW DETAILS (IMP)**
* When you create a Project in Pycharm IDE, by default it uses it’s own python environment (and not the one which we install at **C:\Users\DELL\AppData\Local\Programs\Python**)
* For each project, Pycharm creates it’s own python environment at project level. ‘F:\Python Programs’ is my project path, then you can see python.exe at “F:\Python Programs\venv\Scripts\python.exe”.
* To check this, click on pycharm version displayed on Pycharm IDE right bottom > Interpreter Settings
* Here, you will not see installed Selenium package since it gets installed in different location; not in the project.
* **So, if you wanted to use the python environment where selenium package is installed then**
* **Python Interpreter Setting screen, click config button displayed next the default path > show All.**
* **You might see your python path; then you can select it.**
* If you are not able to see your python path, then click + icon > Virtualenv Environment > Existing Environment > select or navigate your base python environment ("**C:\Users\DELL\AppData\Local\Programs\Python\Python38-32\python.exe**") > Select ‘make available to all projects’ > OK > Select it in Python Interpreter list > OK
* Now your base python environment has been set to all projects and selenium package will be displayed on Interpreter Settings screen.
* Now, whatever the package you will install from command prompt using PIP command, that package will be available for your project.
* Note, even though you don’t set base python environment for your project in pycharm, when you use selenium jargons, pycharm will show error and give suggestions to install it. By clicking on it, you can install selenium package in your pycharm python environment.

**WebDriver**:

* When you write ‘webdriver’, it will give suggestion to import it from Selenium. Import it.
* When you hit '.' after 'webdriver', Select 'Chrome' **class**
* And provide the chromerdriver.exe path which is you downloaded from Selenium website and saved on your local machine
* See different methods for webdriver



**Locators**:

|  |  |  |
| --- | --- | --- |
| **Locator Type** | **BASIC Syntax** | **Example** |
| **Xpath** | //tagname[@attribute=value] | "//input[@name='email']"  driver.find\_element\_by\_xpath("//input[@name='email']") |
| **Css selector** | tagname[attribute=value]  OR  [attribute=value] | "input[name='email']"  driver.find\_element\_by\_css\_selector("input[name='email']") |
| **ID** | No syntax | "id"  driver.find\_element\_by\_id("exampleFormControlSelect1") |
| **Name** | No syntax | "name"  driver.find\_element\_by\_name("name") |
| **Class name** | No syntax | "class-name"  driver.find\_element\_by\_class\_name("btn-success") |
| **Link text** | No syntax | "link-text"  driver.find\_element\_by\_link\_text("Genealogies") |
| **Partial link text** | No syntax | "partialtext"  driver.find\_element\_by\_partial\_link\_text("partialtext") |
| **Tag name** | No syntax | "tag-name"  driver.find\_element\_by\_tag\_name("span") |

* XPATH Locator Without Tag Name

Replace tag name with asterisk (\*) sign, if you don’t want to give tag name or for dynamic tag name

*Syntax:* //\*[attribute=value]

*E.g.*: //\*[@name = ‘email’]

* XPATH based on text

*Syntax:* //tagname[contains(text(), ‘actual-text’)]

*E.g.*: //span[contains(text(),'Users Info')]

* XPATH by traversing tags

*Syntax:* ParentTag/ChildTag

*E.g.:* //div[@class='product-action']/button

* XPATH: Select Parent Locator from Child

*Syntax:* XPATH/parent::tagname

*E.g.*: //\*[title="test"]/parent::div

* CSS Selector Syntax Without Tag Name

Remove tag name

*Syntax*: [attribute=value]

*E.g.*: [class\*=‘alert-succes’]

* CSS Selector by traversing to nth child

*Syntax*: Tagname:nth-child(x)

*E.g.*: div:nth-child(1)

* CSS Selector from Tag and ClassName

Replace spaces with period(.)to use more than one class name

*Syntax*: Tagname.ClassName OR .ClassName

*E.g.*: input.search-keyword OR .search-keyword

For multiple classes

.search-keyword.alert-success

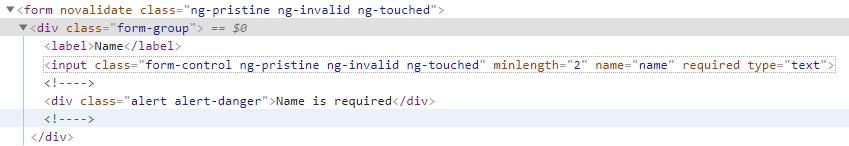
* CSS Selector from Tag and ID

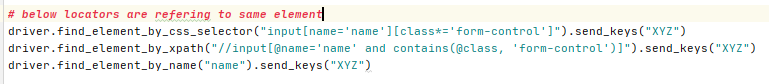
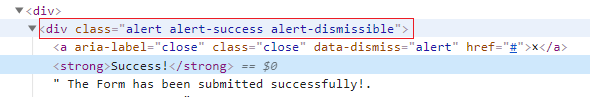
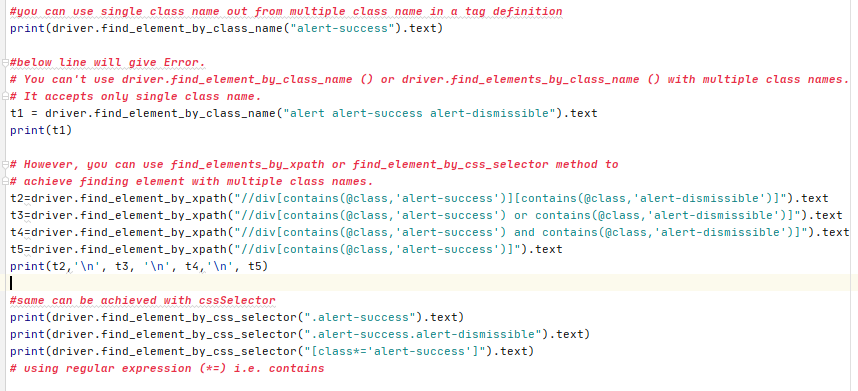
Use # for ID

Syntax: TagName#ID OR #ID

e.g. input#emailID OR #emailID

More examples:



* Xpath: //input[@name='name' and contains(@class, 'form-control')]
* CSS selector: input[name='name'] [class\*='form-control']
* 
* Lets see using Class Name Locator
* 
* you can use single class name out from multiple class name in a tag definition
* You can't use driver.find\_element\_by\_class\_name () or driver.find\_elements\_by\_class\_name () with multiple class names.  
   It accepts only single class name.
* However, you can use find\_elements\_by\_xpath or find\_element\_by\_css\_selector method to achieve finding element with multiple class names.
* same can be achieved with cssSelector
* see below e.g. for same
* 

Identify Static Drop down: