Imports:-

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.color import Color
from selenium.webdriver.support.ui import Select
```

Environment variables:-

```
URL = os.environ.get('APP_URL')
```

browser Declarations :-

```
browser = webdriver.Chrome(executable_path=os.path.expanduser('~/chromedriver'))
                          .Ie
                          .Firefox
browser.get(URL)      # onload event has fired
browser.find_element_by_name('username').send_keys(UNAME)
browser.find_element_by_name('password').send_keys(PWD)

browser.find_element_by_name('password').submit()
try:
        time.sleep(2)
        if
browser.find_element(By.XPATH,"//div//input[@name='validation_code']").is_displayed():

browser.find_element(By.XPATH,"//div//input[@name='validation_code']").send_keys('UITES
T')
        browser.find_element(By.XPATH,"//div//input[@name='validation_code']").submit()
except:
        pass
```

Wait for command:-

```
WebDriverWait(browser,60).until(EC.element_to_be_clickable((By.CLASS_NAME,'leftfloat'))
)
```

ele.**clear**() to clear element values

```
spots = browser.find_element**s**(By.XPATH,"//*[local-name()='circle' and @r='2']")
```

Color check:-

```
spot_color = spots[n].value_of_css_property('stroke')
Color.from_string(spot_color).hex
```

Action chains:- drag and drop

```
a = ActionChains(browser)
a.move_to_element(wtdspots[len(wtdspots)-6]).click(wtdspots[len(wtdspots)-6]).perform()
action_chains.drag_and_drop(element, target).perform()
```

Close/Quit :-

close: closes tab only Quit: closes browser

Select :-

```
from selenium.webdriver.support.ui import Select
select = Select(driver.find_element_by_name('name'))
select.select_by_index(index)
select.select_by_visible_text("text")
select.select_by_value(value)
options = select.options
all_selected_options = select.all_selected_options

select = Select(driver.find_element_by_id('id'))
select.deselect_all()
```

switch windows:-

```
driver.switch_to_window("windowName")
```

Switch frame

```
driver.switch_to_frame(“frameName”)
driver.switch_to_frame(“frameName.childFrame”)
driver.switch_to_default_content()
```

Popups:-

```
alert = driver.switch_to_alert()   #accept, dismiss, read
Alert(driver).accept()
Alert(driver).dismiss()
```

Navigation:-

```
driver.forward()
driver.back()
```

Cookies:-

```
# Go to the correct domain
driver.get("http://www.example.com")

# Now set the cookie. This one's valid for the entire domain
cookie = {‘name’ : ‘foo’, ‘value’ : ‘bar’}
driver.add_cookie(cookie)
```

# And now output all the available cookies for the current URL
driver.get_cookies()

Locating:-

- find_element_by_id
- find_element_by_name
- find_element_by_xpath
- find_element_by_link_text
- find_element_by_partial_link_text
- find_element_by_tag_name
- find_element_by_class_name
- find_element_by_css_selector

Absolute Vs Relative Xpath:-

Absolute-- from home
Relative -- from near properties

Implicit Vs Explicit waits:-

Implicit -- poll the DOM for a certain amount of time
driver.implicitly_wait(10)
Explicit -- wait for a certain condition

WebDriverWait(browser,60).until(EC.element_to_be_clickable((By.CLASS_NAME,'lefloat')))

Expected Conditions

- title_is
- title_contains
- presence_of_element_located
- visibility_of_element_located
- visibility_of
- presence_of_all_elements_located
- text_to_be_present_in_element
- text_to_be_present_in_element_value
- frame_to_be_available_and_switch_to_it
- invisibility_of_element_located
- element_to_be_clickable - it is Displayed and Enabled.
- staleness_of
- element_to_be_selected
- element_located_to_be_selected
- element_selection_state_to_be
- element_located_selection_state_to_be
- alert_is_present

Page source

Driver.page_source

Page details
driver.execute_script("return window.location;")

Scrolling:
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

Lambda:
```
# Function definition is here
sum = lambda arg1, arg2: arg1 + arg2;


# Now you can call sum as a function
print "Value of total : ", sum( 10, 20 )
print "Value of total : ", sum( 20, 20 )
```

Regular Expression:
`re.match(pattern, string, flags=0)`

re.sub()
re.search()

`browser.save_screenshot('screenie.png')`

Datepicker

```
      driver.get('https://jqueryui.com/datepicker/')
        time.sleep(3)
        #define frame
        frame=driver.find_element_by_tag_name('iframe')
        #switch to frame
        driver.switch_to_frame(frame)
        #find date picker
        datepicker=driver.find_element_by_id('datepicker')
        #click on the datepicker to show the calendar
        datepicker.click()
```

Example for page object
https://justin.abrah.ms/python/selenium-page-object-pattern--the-key-to-maintainable-tests.html

Contains xpath :

The <Comment> tag contains two text nodes and two <br> nodes as children.

Your xpath expression was

//*[contains(text(),'ABC')]
To break this down,

* is a selector that matches any element (i.e. tag) -- it returns a node-set.
The [] are a conditional that operates on each individual node in that node set. It matches if any of the individual nodes it operates on match the conditions inside the brackets.
text() is a selector that matches all of the text nodes that are children of the context node -- it returns a node set.
contains is a function that operates on a string. If it is passed a node set, the node set is converted into a string by returning the string-value of the node in the node-set that is first in document order. Hence, it can match only the first text node in your <Comment> element -- namely BLAH BLAH BLAH. Since that doesn't match, you don't get a <Comment> in your results.
You need to change this to

//*[text()[contains(.,'ABC')]]
* is a selector that matches any element (i.e. tag) -- it returns a node-set.
The outer [] are a conditional that operates on each individual node in that node set -- here it operates on each element in the document.
text() is a selector that matches all of the text nodes that are children of the context node -- it returns a node set.
The inner [] are a conditional that operates on each node in that node set -- here each individual text node. Each individual text node is the starting point for any path in the brackets, and can also be referred to explicitly as . within the brackets. It matches if any of the individual nodes it operates on match the conditions inside the brackets.
contains is a function that operates on a string. Here it is passed an individual text node (.). Since it is passed the second text node in the <Comment> tag individually, it will see the 'ABC' string and be able to match it.

XPATH

XPath Examples

All of the following examples use this sample XML code. You can find more documentation on XPath expressions at W3Schools.com

```
<root xmlns:foo="http://www.foo.org/" xmlns:bar="http://www.bar.org">
	<actors>
		<actor id="1">Christian Bale</actor>
		<actor id="2">Liam Neeson</actor>
```

```
            <actor id="3">Michael Caine</actor>
        </actors>
        <foo:singers>
            <foo:singer id="4">Tom Waits</foo:singer>
            <foo:singer id="5">B.B. King</foo:singer>
            <foo:singer id="6">Ray Charles</foo:singer>
        </foo:singers>
</root>
```

1. Select the document node

/

---

2. Select the 'root' element

/root

---

3. Select all 'actor' elements that are direct children of the 'actors' element.

/root/actors/actor

---

4. Select all 'singer' elements regardless of their positions in the document.

//foo:singer

---

5. Select the 'id' attributes of the 'singer' elements regardless of their positions in the document.

//foo:singer/@id

---

6. Select the textual value of first 'actor' element.

//actor[1]/text()

---

7. Select the last 'actor' element.

//actor[last()]

---

8. Select the first and second 'actor' elements using their position.

//actor[position() < 3]

---

9. Select all 'actor' elements that have an 'id' attribute.

//actor[@id]

---

10. Select the 'actor' element with the 'id' attribute value of '3'.

//actor[@id='3']

---

11. Select all 'actor' nodes with the 'id' attribute value lower or equal to '3'.

//actor[@id<=3]

---

12. Select all the children of the 'singers' node.

*/root/foo:singers/**

---

13. Select all the elements in the document.

*//**

---

14. Select all the 'actor' elements AND the 'singer' elements.

//actor|//foo:singer

---

15. Select the name of the first element in the document.

name(//[1])*

---

16. Select the numeric value of the 'id' attribute of the first 'actor' element.

number(//actor[1]/@id)

---

17. Select the string representation value of the 'id' attribute of the first 'actor' element.

string(//actor[1]/@id)

---

18. Select the length of the first 'actor' element's textual value.

string-length(//actor[1]/text())

---

19. Select the local name of the first 'singer' element, i.e. without the namespace.

local-name(//foo:singer[1])

---

20. Select the number of 'singer' elements.

count(//foo:singer)

---

21. Select the sum of the 'id' attributes of the 'singer' elements.

sum(//foo:singer/@id)

ancestor::	*The ancestor axis sends you to parents and above, all the way up to the root node.*
parent:: *or* ***. .***	*The parent axis sends you up a short distance, to the immediate parent of the context node.*
child:: *or* ***/***	*The child axis* ***(which is the default)*** *sends you down to the immediate child of the context node.*
descendant:: *or* ***//***	*The descendent axis sends you down to the children and their children, and their children's children, etc.*
preceding-sibling::	*The preceding-sibling axis sends you to the "left" to the siblings that come before the context node (the big brothers and sisters, or earliest children of a parent).*
following-sibling::	*The following-sibling axis sends you to the "right" to the siblings that come after the context node (the little brothers and sisters, or younger children of a parent.*
self:: *or* ***.***	*There's actually a "self" axis designating the current context node.*
attribute:: *or* ***@***	*One last axis that's sort of in its own parallel universe: the attribute (@) axis! You can follow one of the paths up or down or left or right among elements, and if you want to locate attributes in particular, or want to locate only the elements with a particular attribute or attribute value, you move to the @ axis.*

CSS

. notation to select the class

 # notation to select the id

http://www.testingexcellence.com/css-selectors-selenium-webdriver-tutorial/

XL File handling

```
from openpyxl import Workbook, load_workbook
wb = load_workbook("C:/Users/bamaddu/Desktop/Book1.xlsx")
wb.get_sheet_names()
sheet = wb.get_sheet_by_name('Sheet1')
sheet.title
```