Selenium is a popular browser automation framework used for web testing automation. This cheat sheet provides a quick reference to the most common Selenium commands when using Python, focusing on Selenium 4.x and the latest Selenium Grid.

Driver Initialization

To start, you need to import the required libraries and initialize the WebDriver for the specific browser you are targeting.

```
from selenium import webdriver
```

Chrome	`driver = webdriver.Chrome(executable_path="path/to/chromedriver")`
Firefox	`driver = webdriver.Firefox(executable_path="path/to/geckodriver")`
Safari	`driver = webdriver.Safari(executable_path="path/to/safaridriver")`
Edge	`driver = webdriver.Edge(executable_path="path/to/msedgedriver")`

Locating Elements

By ID:	`driver.find_element_by_id("username")`
By Name:	`driver.find_element_by_name("password")`
By Class Name:	`driver.find_element_by_class_name("submit-btn")`
By Tag Name:	`driver.find_element_by_tag_name("h1")`
By CSS Selector:	`driver.find_element_by_css_selector("#login-form > div > input[type='submit']")`
By XPath:	`driver.find_element_by_xpath("//input[@type='submit']")`
By Link Text:	`driver.find_element_by_link_text("Sign Up")`
By Partial Link Text:	`driver.find_element_by_partial_link_text("Sign")`

Pytest

1. Test functions: Define test functions by prefixing their names with “test_”. Pytest will automatically discover and run these functions as test cases.

```
def test_example_function():
  # Test code here
```

2. Fixtures are used to set up and tear down resources, like test data or objects. They can be created using the `@pytest.fixture` decorator and are passed as function arguments to the test functions that need them.

```
import pytest
@pytest.fixture
  def example_fixture():
    # Set up code here
    yield
    # Tear down code here

  def test_example_function(example_fixture):
    # Test code here
```

Unittest

1. Test classes: Create test classes by inheriting from `unittest.TestCase`.

```
import unittest
class ExampleTestCase(unittest.TestCase):
```

2. Test methods: Define test methods within the test classes by prefixing their names with “test_”.

```
class ExampleTestCase(unittest.TestCase):
  def test_example_method(self):
    # Test code here
```

3. setUp and tearDown: These methods are used to set up and tear down resources for each test method. Override them in your test class as needed.

```
class ExampleTestCase(unittest.TestCase):
  def setUp(self):
    # Set up code here
  def tearDown(self):
    # Tear down code here
  def test_example_method(self):
    # Test code here
```

4. setUpClass and tearDownClass: These class methods are used to set up and tear down resources for the entire test class. Override them in your test class and use the `@classmethod` decorator.

```
class ExampleTestCase(unittest.TestCase):
  @classmethod
    def setUpClass(cls):
      # Set up code here
  @classmethod
    def tearDownClass(cls):
      # Tear down code here
  def test_example_method(self):
    # Test code here
```

Working with Files

Upload a file:

```
file_input = driver.find_element_by_id("file-upload")
file_input.send_keys("path/to/your/file.txt")
```

Read data from a text file:

```
with open("path/to/your/file.txt", "r") as file:
data = file.read()
```

Read data from a CSV file:

```
import csv
with open("path/to/your/file.csv", "r") as csvfile:
csv_reader = csv.reader(csvfile)
  for row in csv_reader:
    print(row)
```

Read data from an Excel file:

```
import openpyxl (install with pip install openpyxl)
workbook = openpyxl.load_workbook("path/to/your/file.xlsx")
worksheet = workbook.active
for row in worksheet.iter_rows():
  for cell in row:
    print(cell.value)
```

Selenium Navigators

Navigate to a URL:

```
driver.get("https://www.example.com")
```

Refresh the page:

```
driver.refresh()
```

Navigate forward in browser history:

```
driver.forward()
```

Navigate back in browser history:

```
driver.back()
```

Working with Windows

Get the current window handle:

```
driver.current_window_handle
```

Get all window handles:

```
driver.window_handles
```

Switch to a specific window:

```
driver.switch_to.window("window_handle")
```

Switch to the last opened window:

```
driver.switch_to.window(driver.window_handles[-1])
```

Close the current window:

```
driver.close()
```

Working with Frames

Switch to a frame by name or ID:

```
driver.switch_to.frame("frame_id")
```

Switch to a frame by index:

```
driver.switch_to.frame(0) # Switch to the first frame by index
```

Switch to a frame using a WebElement:

```
frame_element = driver.find_element_by_id("frame_id")
 driver.switch_to.frame(frame_element)
```

Switch back to the main content:

```
driver.switch_to.default_content()
```

Selenium Operations

Lauch a Webpage:

```
driver.get("https://www.example.com")
```

Click a button:

```
button = driver.find_element_by_id("button_id")
button.click()
```

Accept an alert pop-up:

```
alert = driver.switch_to.alert
alert.accept()
```

Print the page title:

```
print(driver.title)
```

Implicit wait:

```
driver.implicitly_wait(10) # Waits up to 10 seconds for elements to appear
```

Explicit wait:

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
element = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, "element_id")))
```

Sleep:

```
import time
time.sleep(5) # Pause the script for 5 seconds
```

Clear the input field text:

```
input_field = driver.find_element_by_id("input_field_id")
input_field.clear()
```

Disable a field (set the ‘disabled’ attribute):

```
field = driver.find_element_by_id("field_id")
driver.execute_script("arguments[0].setAttribute('disabled', true)", field)
```

Enable a field (remove the ‘disabled’ attribute):

```
field = driver.find_element_by_id("field_id")
driver.execute_script("arguments[0].removeAttribute('disabled')", field)
```

Selenium Grid

Start the hub:

```
java -jar selenium-server-standalone-x.y.z.jar -role hub
```

Start a node:

```
java -jar selenium-server-standalone-x.y.z.jar -role node -hub
```

Server:

```
http://localhost:4444/ui/index.html
```