Appium with Python - UI Testing Cheat Sheet

Installation & Setup

```
# Install Appium
npm install -g appium

# Install Python client
pip install Appium-Python-Client

# Install additional dependencies
pip install selenium pytest allure-pytest
```

Basic Setup & Driver Initialization

Android Setup

```
from appium import webdriver
from appium.options.android import UiAutomator2Options

# Android capabilities
options = UiAutomator2Options()
options.platform_name = "Android"
options.device_name = "emulator-5554"
options.app_package = "com.example.app"
options.app_activity = "MainActivity"
options.automation_name = "UiAutomator2"

# Initialize driver
driver = webdriver.Remote("http://localhost:4723", options=options)
```

iOS Setup

```
from appium.options.ios import XCUITestOptions

# iOS capabilities
options = XCUITestOptions()
options.platform_name = "iOS"
options.device_name = "iPhone 14"
options.bundle_id = "com.example.app"
options.automation_name = "XCUITest"

driver = webdriver.Remote("http://localhost:4723", options=options)
```

Common Capabilities

```
options.new_command_timeout = 60
options.implicit_wait_timeout = 10
options.app_wait_timeout = 20000
options.no_reset = True  # Don't reset app state
options.full_reset = False  # Don't uninstall app
```

Element Locators

Find Elements

```
from appium.webdriver.common.appiumby import AppiumBy
from selenium.webdriver.common.by import By

# By ID
element = driver.find_element(AppiumBy.ID, "com.example:id/button")
element = driver.find_element(By.ID, "button_id")

# By Class Name
element = driver.find_element(AppiumBy.CLASS_NAME, "android.widget.Button")

# By XPath
element = driver.find_element(AppiumBy.XPATH, "//android.widget.Button[@text='Login']")

# By Accessibility ID
element = driver.find_element(AppiumBy.ACCESSIBILITY_ID, "login_button")

# By Text (Android)
element = driver.find_element(AppiumBy.ANDROID_UIAUTOMATOR, 'text("Login")')

# By Resource ID (Android)
element = driver.find_element(AppiumBy.ANDROID_UIAUTOMATOR, 'resourceId("com.example:id/butt

# By Predicate String (iOS)
element = driver.find_element(AppiumBy.IOS_PREDICATE, "label == 'Login'")

# By Class Chain (iOS)
element = driver.find_element(AppiumBy.IOS_CLASS_CHAIN, "**/XCUIElementTypeButton[`label ==
```

Find Multiple Elements

```
elements = driver.find_elements(AppiumBy.CLASS_NAME, "android.widget.TextView")
```

Basic Actions

Click & Tap

```
# Click element
element.click()

# Tap by coordinates
from appium.webdriver.common.touch_action import TouchAction
TouchAction(driver).tap(x=100, y=200).perform()

# Multiple taps
TouchAction(driver).tap(element, count=2).perform()
```

Text Input

```
# Clear and type text
element.clear()
element.send_keys("Hello World")

# Hide keyboard
driver.hide_keyboard()

# Check if keyboard is shown
driver.is_keyboard_shown()
```

Swipe & Scroll

```
# Swipe by coordinates
driver.swipe(start_x=100, start_y=500, end_x=100, end_y=100, duration=1000)

# Scroll to element (Android)
driver.find_element(AppiumBy.ANDROID_UIAUTOMATOR,
    'new UiScrollable(new UiSelector().scrollable(true)).scrollIntoView(text("Target Text")

# Scroll down/up
def scroll_down():
    size = driver.get_window_size()
    start_x = size['width'] // 2
    start_y = size['height'] * 0.8
    end_y = size['height'] * 0.2
    driver.swipe(start_x, start_y, start_x, end_y, 1000)
```

Waits & Synchronization

Implicit Wait

```
driver.implicitly_wait(10)  # seconds
```

Explicit Waits

```python
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

wait = WebDriverWait(driver, 10)

# Wait for element to be present
element = wait.until(EC.presence_of_element_located((AppiumBy.ID, "element_id")))

# Wait for element to be clickable
element = wait.until(EC.element_to_be_clickable((AppiumBy.ID, "button_id")))

# Wait for text to be present
wait.until(EC.text_to_be_present_in_element((AppiumBy.ID, "label"), "Expected Text"))
```

Custom Wait Conditions

```python
def element_has_text(locator, text):
    def _predicate(driver):
        element = driver.find_element(*locator)
        return text in element.text
    return _predicate

wait.until(element_has_text((AppiumBy.ID, "status"), "Success"))
```

Gestures & Touch Actions

Touch Actions

```python
from appium.webdriver.common.touch_action import TouchAction

# Press and hold
TouchAction(driver).press(element).wait(2000).release().perform()

# Drag and drop
action = TouchAction(driver)
action.press(source_element).move_to(target_element).release().perform()

# Pinch (zoom out)
from appium.webdriver.common.multi_action import MultiAction

action1 = TouchAction(driver)
action2 = TouchAction(driver)
action1.press(x=200, y=300).move_to(x=150, y=250).release()
action2.press(x=300, y=400).move_to(x=350, y=450).release()
```

```
multi_action = MultiAction(driver)
multi_action.add(action1, action2)
multi_action.perform()
```

Advanced Gestures (W3C Actions)

```
from selenium.webdriver import ActionChains
from selenium.webdriver.common.actions import interaction
from selenium.webdriver.common.actions.action_builder import ActionBuilder
from selenium.webdriver.common.actions.pointer_input import PointerInput

# Tap
actions = ActionChains(driver)
actions.w3c_actions = ActionBuilder(driver, mouse=PointerInput(interaction.POINTER_TOUCH, "t
actions.w3c_actions.pointer_action.move_to_location(100, 200)
actions.w3c_actions.pointer_action.pointer_down()
actions.w3c_actions.pointer_action.pointer_up()
actions.perform()
```

Device Interactions

Device Management

```
# Get device orientation
orientation = driver.orientation  # 'PORTRAIT' or 'LANDSCAPE'

# Set orientation
driver.orientation = 'LANDSCAPE'

# Get window size
size = driver.get_window_size()  # {'width': 1080, 'height': 1920}

# Take screenshot
driver.save_screenshot("screenshot.png")

# Get page source
source = driver.page_source
```

App Management

```
# Install app
driver.install_app("/path/to/app.apk")

# Launch app
driver.activate_app("com.example.app")

# Close app
```

```python
driver.terminate_app("com.example.app")

# Remove app
driver.remove_app("com.example.app")

# Check if app is installed
is_installed = driver.is_app_installed("com.example.app")
```

System Actions

```python
# Press Android back button
driver.back()

# Press home button (Android)
driver.press_keycode(3)

# Lock/unlock device
driver.lock()
driver.unlock()

# Open notifications (Android)
driver.open_notifications()
```

Common Test Patterns

Page Object Model

```python
from appium.webdriver.common.appiumby import AppiumBy

class LoginPage:
    def __init__(self, driver):
        self.driver = driver
        self.username_field = (AppiumBy.ID, "username")
        self.password_field = (AppiumBy.ID, "password")
        self.login_button = (AppiumBy.ID, "login_btn")

    def login(self, username, password):
        self.driver.find_element(*self.username_field).send_keys(username)
        self.driver.find_element(*self.password_field).send_keys(password)
        self.driver.find_element(*self.login_button).click()

    def is_login_button_displayed(self):
        return self.driver.find_element(*self.login_button).is_displayed()
```

Base Test Class

```
import pytest
from appium import webdriver

class BaseTest:
    @pytest.fixture(autouse=True)
    def setup(self):
        # Setup driver
        self.driver = webdriver.Remote("http://localhost:4723", options)
        yield
        # Teardown
        self.driver.quit()

    def take_screenshot(self, name):
        self.driver.save_screenshot(f"{name}.png")
```

Debugging & Troubleshooting

Element Information

```
# Get element attributes
element.get_attribute("text")
element.get_attribute("content-desc")
element.get_attribute("resource-id")
element.get_attribute("class")
element.get_attribute("enabled")

# Element properties
element.text
element.location  # {'x': 100, 'y': 200}
element.size      # {'width': 200, 'height': 50}
element.is_displayed()
element.is_enabled()
element.is_selected()
```

Logging & Error Handling

```
import logging

# Setup logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Error handling
try:
    element = driver.find_element(AppiumBy.ID, "element_id")
    element.click()
except Exception as e:
    logger.error(f"Element not found: {e}")
    driver.save_screenshot("error_screenshot.png")
```

Useful Utilities

Wait for Element Helper

```
def wait_for_element(driver, locator, timeout=10):
    wait = WebDriverWait(driver, timeout)
    return wait.until(EC.presence_of_element_located(locator))

def wait_and_click(driver, locator, timeout=10):
    element = wait_for_element(driver, locator, timeout)
    element.click()
    return element
```

Retry Decorator

```
import time
from functools import wraps

def retry(times=3, delay=1):
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            for i in range(times):
                try:
                    return func(*args, **kwargs)
                except Exception as e:
                    if i == times - 1:
                        raise e
                    time.sleep(delay)
        return wrapper
    return decorator

@retry(times=3, delay=2)
def find_and_click(driver, locator):
    driver.find_element(*locator).click()
```

Testing Frameworks Integration

pytest Example

```
import pytest
```

```python
class TestLogin:
    def test_successful_login(self, driver):
        login_page = LoginPage(driver)
        login_page.login("user@example.com", "password123")

        # Assert successful login
        assert driver.find_element(AppiumBy.ID, "welcome_message").is_displayed()

    @pytest.mark.parametrize("username,password", [
        ("invalid@email.com", "wrongpass"),
        ("", "password123"),
        ("user@example.com", ""),
    ])
    def test_login_failures(self, driver, username, password):
        login_page = LoginPage(driver)
        login_page.login(username, password)

        error_element = driver.find_element(AppiumBy.ID, "error_message")
        assert error_element.is_displayed()
```

Performance & Best Practices

Optimize Element Location

```python
# Prefer ID over XPath
# Good
element = driver.find_element(AppiumBy.ID, "button_id")

# Avoid complex XPath
# Bad
element = driver.find_element(AppiumBy.XPATH, "//div[@class='container']//button[contains(@c

# Use UiAutomator for Android
element = driver.find_element(AppiumBy.ANDROID_UIAUTOMATOR,
    'resourceId("com.example:id/button")')
```

Resource Management

```python
# Always quit driver
try:
    # Test code
    pass
finally:
    driver.quit()

# Use context managers
from contextlib import contextmanager
```

```python
@contextmanager
def appium_driver(capabilities):
    driver = webdriver.Remote("http://localhost:4723", capabilities)
    try:
        yield driver
    finally:
        driver.quit()

# Usage
with appium_driver(options) as driver:
    # Test code
    pass
```

Common Issues & Solutions

Element Not Found

- Use explicit waits instead of sleep()
- Verify element locator with Appium Inspector
- Check if element is in a different context (webview)

Flaky Tests

- Implement proper waits
- Use retry mechanisms
- Stabilize test environment
- Handle dynamic content properly

Performance Issues

- Minimize XPath usage
- Use efficient locators
- Implement proper cleanup
- Avoid unnecessary screenshots