Here are **10 UI Testing Scenarios** in **Scenario → Question → Answer** format using **Selenium Python** 🚀

**1️⃣ Login Page - Account Lock After 3 Wrong Attempts**

**📝 Scenario:** A user enters the wrong password three times and gets locked out.

**❓ Question:** How do you automate testing for account lock after multiple failed login attempts?

**✅ Answer:**

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

# Invalid data dictionary (can contain multiple sets of data)

invalid\_data = {

"user1": {"username": "invalid\_user1", "password": "wrong\_password1"},

"user2": {"username": "invalid\_user2", "password": "wrong\_password2"},

"user3": {"username": "invalid\_user3", "password": "wrong\_password3"}

}

# Assuming you have already initialized the driver

driver = webdriver.Chrome()

# Navigate to the login page

driver.get("https://example.com/login")

username = driver.find\_element(By.ID, "username")

password = driver.find\_element(By.ID, "password")

login\_button = driver.find\_element(By.ID, "login-button")

# Loop through the dictionary and perform login attempts with invalid data

for user\_key, credentials in invalid\_data.items():

for attempt in range(3): # Retry the login 3 times for each set of invalid credentials

username.send\_keys(credentials["username"])

password.send\_keys(credentials["password"])

login\_button.click()

time.sleep(2) # Wait for the error message to appear

# Clear the fields for the next attempt

username.clear()

password.clear()

print(f"Attempt {attempt + 1} for {user\_key} completed!")

print("Test completed!")

driver.quit()

**2️⃣ Handling Dynamic Drop-down Suggestions**

**📝 Scenario:** A search box shows dynamic auto-suggestions as the user types.

**❓ Question:** How do you select an option from an auto-suggest dropdown in Selenium?

**✅ Answer:**

driver.get("https://example.com/search")

search\_box = driver.find\_element(By.ID, "search")

search\_box.send\_keys("Tes")

time.sleep(2) # Wait for suggestions

suggestions = driver.find\_elements(By.CSS\_SELECTOR, ".suggestion-item")

for suggestion in suggestions:

if "Test Automation" in suggestion.text:

suggestion.click()

print("✅ Suggestion clicked successfully!")

break

driver.quit()

**3️⃣ Broken Links & Images Detection**

**📝 Scenario:** A webpage has broken images and links.

**❓ Question:** How do you detect broken links and images in Selenium?

**✅ Answer:**

import requests

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

links = driver.find\_elements(By.TAG\_NAME, "a")

for link in links:

url = link.get\_attribute("href")

if url:

response = requests.head(url)

if response.status\_code >= 400:

print(f"❌ Broken link: {url} - {response.status\_code}")

images = driver.find\_elements(By.TAG\_NAME, "img")

for img in images:

img\_url = img.get\_attribute("src")

response = requests.head(img\_url)

if response.status\_code >= 400:

print(f"❌ Broken image: {img\_url}")

print("✅ Broken link/image check complete!")

driver.quit()

**4️⃣ Handling Alerts & Popups**

**📝 Scenario:** A confirmation alert appears when a user tries to delete an item.

**❓ Question:** How do you handle JavaScript alerts in Selenium?

**✅ Answer:**

from selenium.webdriver.common.alert import Alert

driver.get("https://example.com/delete")

delete\_button = driver.find\_element(By.ID, "delete")

delete\_button.click()

alert = Alert(driver)

assert "Are you sure" in alert.text, "Alert text mismatch!"

alert.accept() # Click "OK"

print("✅ Alert handled successfully!")

driver.quit()

**5️⃣ File Upload Testing**

**📝 Scenario:** A website allows users to upload only PDF files.

**❓ Question:** How do you automate file uploads using Selenium?

**✅ Answer:**

driver.get("https://example.com/upload")

upload\_button = driver.find\_element(By.ID, "file-upload")

upload\_button.send\_keys("/path/to/sample.pdf") # Replace with actual file path

submit\_button = driver.find\_element(By.ID, "upload-submit")

submit\_button.click()

message = driver.find\_element(By.ID, "upload-success")

assert "Upload successful" in message.text, "Upload failed!"

print("✅ File upload validation passed!")

driver.quit()

**6️⃣ Page Load Performance**

**📝 Scenario:** A webpage should load within 3 seconds for a good user experience.

**❓ Question:** How do you measure page load time in Selenium?

**✅ Answer:**

import time

start\_time = time.time()

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

end\_time = time.time()

load\_time = end\_time - start\_time

assert load\_time < 3, f"❌ Page took too long to load: {load\_time:.2f}s"

print(f"✅ Page loaded in {load\_time:.2f} seconds")

driver.quit()

**7️⃣ Handling Stale Element Exception**

**📝 Scenario:** A page refresh makes elements stale, causing Selenium to throw an error.

**❓ Question:** How do you handle stale elements in Selenium?

**✅ Answer:**

from selenium.common.exceptions import StaleElementReferenceException

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

retry\_count = 0

max\_retries = 3

while retry\_count < max\_retries:

try:

button = driver.find\_element(By.ID, "refresh-button")

button.click()

print("✅ Button clicked successfully!")

break

except StaleElementReferenceException:

print("Retrying due to stale element...")

retry\_count += 1

driver.quit()

**8️⃣ Shopping Cart Update Validation**

**📝 Scenario:** Adding an item to the cart should update the cart count.

**❓ Question:** How do you verify that the shopping cart count updates correctly?

**✅ Answer:**

driver.get("https://example.com/shop")

add\_to\_cart = driver.find\_element(By.ID, "add-item")

add\_to\_cart.click()

time.sleep(2)

cart\_count = driver.find\_element(By.ID, "cart-count")

assert cart\_count.text == "1", "Cart count mismatch!"

print("✅ Cart count updated correctly!")

driver.quit()

**9️⃣ Handling Multiple Tabs**

**📝 Scenario:** Clicking a link should open a new tab with the Terms & Conditions page.

**❓ Question:** How do you switch between multiple tabs in Selenium?

**✅ Answer:**

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

terms\_link = driver.find\_element(By.LINK\_TEXT, "Terms & Conditions")

terms\_link.click()

driver.switch\_to.window(driver.window\_handles[1])

assert "Terms" in driver.title, "New tab did not load correctly!"

driver.close()

driver.switch\_to.window(driver.window\_handles[0])

print("✅ Multi-tab handling successful!")

driver.quit()

**🔟 Responsive Design Validation**

**📝 Scenario:** The website layout should adjust for mobile, tablet, and desktop.

**❓ Question:** How do you test responsive design using Selenium?

**✅ Answer:**

screen\_sizes = {

"Mobile": (375, 812),

"Tablet": (768, 1024),

"Desktop": (1366, 768)

}

for device, size in screen\_sizes.items():

print(f"Testing on {device} with resolution {size}")

driver.set\_window\_size(size[0], size[1])

time.sleep(2)

menu = driver.find\_element(By.ID, "nav-menu")

if device == "Mobile":

assert not menu.is\_displayed(), "Navigation menu should be collapsed."

else:

assert menu.is\_displayed(), "Navigation menu should be visible."

print(f"✅ UI validation passed for {device}")

driver.quit()

**1️⃣1️⃣ Drag and Drop Functionality**

**📝 Scenario:** A Kanban board allows users to drag tasks from "To Do" to "In Progress".

**❓ Question:** How do you automate drag-and-drop functionality in Selenium?

**✅ Answer:**

from selenium.webdriver.common.action\_chains import ActionChains

driver.get("https://example.com/kanban")

source = driver.find\_element(By.ID, "task-todo")

target = driver.find\_element(By.ID, "task-inprogress")

actions = ActionChains(driver)

actions.drag\_and\_drop(source, target).perform()

assert "In Progress" in target.text, "❌ Task not moved successfully!"

print("✅ Drag and drop validated!")

driver.quit()

**1️⃣2️⃣ Scroll to Element & Infinite Scroll Testing**

**📝 Scenario:** A webpage has an infinite scroll, and an element appears only after scrolling.

**❓ Question:** How do you scroll to a specific element in Selenium?

**✅ Answer:**

driver.get("https://example.com/infinite-scroll")

target\_element = driver.find\_element(By.ID, "load-more")

driver.execute\_script("arguments[0].scrollIntoView();", target\_element)

assert target\_element.is\_displayed(), "❌ Element not visible after scroll!"

print("✅ Scrolling functionality validated!")

driver.quit()

**1️⃣3️⃣ Date Picker - Selecting a Future Date**

**📝 Scenario:** A date picker allows users to book appointments only 7 days in the future.

**❓ Question:** How do you select a date dynamically in a date picker?

**✅ Answer:**

from datetime import datetime, timedelta

future\_date = (datetime.today() + timedelta(days=7)).strftime("%d/%m/%Y")

driver.get("https://example.com/datepicker")

date\_input = driver.find\_element(By.ID, "appointment-date")

date\_input.send\_keys(future\_date)

selected\_date = date\_input.get\_attribute("value")

assert selected\_date == future\_date, f"❌ Date selection failed! Expected: {future\_date}, Got: {selected\_date}"

print("✅ Date picker validation successful!")

driver.quit()

**1️⃣4️⃣ Captcha Handling (Manual Intervention Required)**

**📝 Scenario:** A signup page has a CAPTCHA that requires manual intervention.

**❓ Question:** How do you automate form submission when CAPTCHA is present?

**✅ Answer:**

driver.get("https://example.com/signup")

# Fill form fields

driver.find\_element(By.ID, "username").send\_keys("testuser")

driver.find\_element(By.ID, "password").send\_keys("securepass")

input("🚀 Solve the CAPTCHA manually and press Enter to continue...") # Wait for user input

driver.find\_element(By.ID, "submit-button").click()

assert "Welcome" in driver.page\_source, "❌ Signup failed!"

print("✅ Form submission with CAPTCHA handled successfully!")

driver.quit()

**1️⃣5️⃣ Handling Shadow DOM Elements**

**📝 Scenario:** A UI component is inside a **Shadow DOM**, making it difficult to locate.

**❓ Question:** How do you interact with elements inside Shadow DOM in Selenium?

**✅ Answer:**

driver.get("https://example.com/shadow")

shadow\_host = driver.find\_element(By.ID, "shadow-host")

shadow\_root = driver.execute\_script("return arguments[0].shadowRoot", shadow\_host)

shadow\_element = shadow\_root.find\_element(By.CSS\_SELECTOR, ".shadow-button")

shadow\_element.click()

print("✅ Successfully interacted with Shadow DOM element!")

driver.quit()

**1️⃣6️⃣ Verifying CSS Styles Dynamically**

**📝 Scenario:** A button should change color when hovered over.

**❓ Question:** How do you verify CSS changes using Selenium?

**✅ Answer:**

driver.get("https://example.com/button")

button = driver.find\_element(By.ID, "hover-button")

original\_color = button.value\_of\_css\_property("background-color")

ActionChains(driver).move\_to\_element(button).perform()

hover\_color = button.value\_of\_css\_property("background-color")

assert original\_color != hover\_color, "❌ Button color did not change on hover!"

print("✅ CSS hover effect validated!")

driver.quit()

**1️⃣7️⃣ Handling iFrames in Web Pages**

**📝 Scenario:** A webpage contains an embedded video inside an iframe.

**❓ Question:** How do you switch to an iframe and interact with its elements?

**✅ Answer:**

driver.get("https://example.com/video")

iframe = driver.find\_element(By.TAG\_NAME, "iframe")

driver.switch\_to.frame(iframe)

play\_button = driver.find\_element(By.CLASS\_NAME, "play")

play\_button.click()

print("✅ Video play button clicked inside iframe!")

driver.switch\_to.default\_content() # Switch back to the main page

driver.quit()

**1️⃣8️⃣ Handling Multi-Select Drop-down**

**📝 Scenario:** A form has a **multi-select dropdown** where users can select multiple options.

**❓ Question:** How do you automate multi-selection in Selenium?

**✅ Answer:**

from selenium.webdriver.support.ui import Select

driver.get("https://example.com/multi-select")

multi\_select = Select(driver.find\_element(By.ID, "options"))

multi\_select.select\_by\_visible\_text("Option 1")

multi\_select.select\_by\_visible\_text("Option 3")

selected\_options = [opt.text for opt in multi\_select.all\_selected\_options]

assert "Option 1" in selected\_options and "Option 3" in selected\_options, "❌ Multi-selection failed!"

print("✅ Multi-select dropdown validation passed!")

driver.quit()

**1️⃣9️⃣ Verifying Text Alignment and Font Style**

**📝 Scenario:** A heading on a page should be center-aligned and bold.

**❓ Question:** How do you verify text alignment and font style?

**✅ Answer:**

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

heading = driver.find\_element(By.TAG\_NAME, "h1")

alignment = heading.value\_of\_css\_property("text-align")

font\_weight = heading.value\_of\_css\_property("font-weight")

assert alignment == "center", "❌ Heading is not center-aligned!"

assert font\_weight == "700", "❌ Heading is not bold!"

print("✅ Text alignment and font style validated!")

driver.quit()

**2️⃣0️⃣ Handling File Downloads**

**📝 Scenario:** Clicking a "Download PDF" button should download a file.

**❓ Question:** How do you automate file downloads using Selenium?

**✅ Answer:**

import os

download\_dir = "/path/to/download/folder" # Set your download folder

chrome\_options = webdriver.ChromeOptions()

prefs = {"download.default\_directory": download\_dir}

chrome\_options.add\_experimental\_option("prefs", prefs)

driver = webdriver.Chrome(options=chrome\_options)

driver.get("https://example.com/download")

driver.find\_element(By.ID, "download-pdf").click()

time.sleep(3) # Wait for download

assert os.path.exists(os.path.join(download\_dir, "sample.pdf")), "❌ File download failed!"

print("✅ File downloaded successfully!")

driver.quit()

**2️⃣1️⃣ Captcha Handling in Login Forms**

**📝 Scenario:** A login page has a CAPTCHA that prevents automated logins.

**❓ Question:** How do you handle CAPTCHA automation in Selenium?

**✅ Answer:**  
Selenium **cannot bypass** CAPTCHA (by design), but you can handle it using:

1. **Manual intervention** (Pause script and ask the tester to enter CAPTCHA).
2. **Third-party services** (e.g., 2Captcha, OCR-based APIs).
3. **Disabling CAPTCHA** for test environments (ask developers).

Example for manual intervention:

input("Enter CAPTCHA manually and press Enter to continue...")

login\_button.click()

**2️⃣2️⃣ Verifying Tooltip Text on Hover**

**📝 Scenario:** A button displays a tooltip when hovered.

**❓ Question:** How do you verify tooltip text using Selenium?

**✅ Answer:**

from selenium.webdriver import ActionChains

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

tooltip\_element = driver.find\_element(By.ID, "tooltip-button")

ActionChains(driver).move\_to\_element(tooltip\_element).perform()

tooltip\_text = driver.find\_element(By.CLASS\_NAME, "tooltip-text").text

assert tooltip\_text == "Expected Tooltip", "Tooltip text mismatch!"

print("✅ Tooltip validation successful!")

driver.quit()

**2️⃣3️⃣ Checking Session Timeout Behavior**

**📝 Scenario:** The user should be logged out after 5 minutes of inactivity.

**❓ Question:** How do you verify session timeout in Selenium?

**✅ Answer:**

import time

driver.get("https://example.com/login")

time.sleep(300) # Wait for 5 minutes

session\_message = driver.find\_element(By.ID, "session-expired")

assert "Session expired" in session\_message.text, "Session timeout not working!"

print("✅ Session timeout validated!")

driver.quit()

**2️⃣4️⃣ Verifying Text Alignment on a Webpage**

**📝 Scenario:** The login button should be center-aligned.

**❓ Question:** How do you check if a button is aligned correctly using Selenium?

**✅ Answer:**

button = driver.find\_element(By.ID, "login-button")

# Get button location and size

x\_position = button.location['x']

width = button.size['width']

page\_width = driver.execute\_script("return document.body.clientWidth;")

assert abs((page\_width / 2) - (x\_position + width / 2)) < 5, "Button is not center-aligned!"

print("✅ Button alignment is correct!")

driver.quit()

**2️⃣5️⃣ Validating Placeholder Text in Input Fields**

**📝 Scenario:** The search box should have a placeholder text "Search here...".

**❓ Question:** How do you verify placeholder text in an input field?

**✅ Answer:**

search\_box = driver.find\_element(By.ID, "search")

placeholder = search\_box.get\_attribute("placeholder")

assert placeholder == "Search here...", "Placeholder text mismatch!"

print("✅ Placeholder validation passed!")

driver.quit()

**2️⃣6️⃣ Handling Multi-Step Form Navigation**

**📝 Scenario:** A multi-step form requires navigation between steps.

**❓ Question:** How do you verify step-by-step navigation in a multi-step form?

**✅ Answer:**

driver.get("https://example.com/form")

driver.find\_element(By.ID, "next-step").click()

assert "Step 2" in driver.page\_source, "Step 2 not loaded!"

driver.find\_element(By.ID, "next-step").click()

assert "Step 3" in driver.page\_source, "Step 3 not loaded!"

print("✅ Multi-step form navigation validated!")

driver.quit()

**2️⃣7️⃣ Testing Mobile Navigation Menu (Hamburger Menu)**

**📝 Scenario:** On mobile screens, the navigation menu should collapse into a hamburger menu.

**❓ Question:** How do you verify the hamburger menu in a mobile view?

**✅ Answer:**

driver.set\_window\_size(375, 812) # Simulating mobile screen

menu\_button = driver.find\_element(By.ID, "hamburger-menu")

menu\_button.click()

menu = driver.find\_element(By.ID, "mobile-nav")

assert menu.is\_displayed(), "Mobile menu is not visible!"

print("✅ Mobile navigation menu works!")

driver.quit()

**2️⃣8️⃣ Ensuring Lazy-Loaded Images Are Displayed**

**📝 Scenario:** Some images are loaded only when scrolled into view.

**❓ Question:** How do you verify lazy-loaded images in Selenium?

**✅ Answer:**

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2) # Wait for images to load

lazy\_images = driver.find\_elements(By.CSS\_SELECTOR, "img.lazy")

for img in lazy\_images:

assert img.get\_attribute("src") != "", "Lazy-loaded image not displayed!"

print("✅ Lazy-loaded images are displayed correctly!")

driver.quit()

**2️⃣9️⃣ Testing Error Message for Invalid Form Inputs**

**📝 Scenario:** An invalid email should show an error message.

**❓ Question:** How do you verify form validation error messages?

**✅ Answer:**

driver.get("https://example.com/signup")

email\_field = driver.find\_element(By.ID, "email")

email\_field.send\_keys("invalid-email")

driver.find\_element(By.ID, "submit").click()

error\_message = driver.find\_element(By.CLASS\_NAME, "error-message")

assert "Invalid email" in error\_message.text, "Error message incorrect!"

print("✅ Form validation tested successfully!")

driver.quit()

**3️⃣0️⃣ Testing Page Scroll and Sticky Elements**

**📝 Scenario:** The website has a sticky navigation bar that remains fixed when scrolling.

**❓ Question:** How do you verify sticky elements using Selenium?

**✅ Answer:**

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

nav\_bar = driver.find\_element(By.ID, "navbar")

initial\_position = nav\_bar.location['y']

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

final\_position = nav\_bar.location['y']

assert initial\_position == final\_position, "Sticky navbar is not working!"

print("✅ Sticky navigation bar verified!")

driver.quit()

**Final Thoughts**

These **10 Intermediate UI Testing Scenarios** help you cover **real-world challenges** in **Selenium Python** 🚀

Need **more advanced** scenarios or **help with specific automation challenges?** Let me know! 🎯