PassQueen
a.k.a dumpsqueen.com

A4Q Certified Selenium Tester Foundation

iSQI CSeT-F

Version Demo

Total Demo Questions: 7

Total Premium Questions: 72

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Selenium WebDriver	50
Topic 2, Selenium Grid	1
Topic 3, Test Automation Frameworks	21
Total	72

QUESTION NO: 1

Which would be the best set of methods called by a Page Object that corresponds to the following dialog box?

A. log_in, cancel_log_in, remind_password

B. log_in, cancel_log_in. clear_user, clear_password

C. enter_login. enter_password, log_in, cancel

D. push_button(button), enter__text(text). clickjink(link)

ANSWER: C

Explanation:

`enter_login. enter_password, log_in, cancel` is the best Page Object method set because it represents the meaningful user interactions offered by a typical login dialog: supplying a login identifier, supplying a password, submitting the credentials, and cancelling the dialog. Page Object methods should model the services and business-oriented actions that a page or component provides to a test, rather than expose low-level implementation details such as buttons, links, or generic text-entry mechanics. This gives test code an expressive, readable vocabulary: a test can enter credentials and submit or cancel without needing to know the locators, element types, or sequencing required by the dialog's internal implementation.

Keeping these interactions in the Page Object also centralizes UI knowledge. If the login dialog's markup, selectors, or input handling changes, the corresponding Page Object implementation can be updated while tests that invoke these methods retain their intent and remain easier to maintain. Selenium's guidance describes Page Objects as an abstraction that reduces duplication and isolates page-specific details from test code. The Selenium project's Page Object Models guidance and the test-practice documentation support designing this interface around the page's user-facing behavior.

QUESTION NO: 2

Which of the following practices can reduce duplicated Selenium interaction code in a test suite? Select three options

A. Create page objects for pages or reusable components

B. Use reusable helper methods for common operations

C. Centralize locators used by the test suite

D. Repeat the same element interactions in every test case

E. Add fixed delays before all interactions

ANSWER: A B C

Explanation:

Creating page objects, using reusable helper methods, and centralizing locators are practices that reduce duplication in a Selenium test suite. A page object can encapsulate the elements and operations of a page or component, while helper methods can provide common operations such as logging in or waiting for a page state. Centralized locators reduce the number of places that must be changed when the UI evolves. Repeating low-level click and locator code in every test increases maintenance effort, and embedding arbitrary fixed delays does not provide reuse or robust synchronization. See the Selenium Page Object Models guidance and the Selenium waiting strategies documentation.

QUESTION NO: 3

Which of the following are useful responsibilities of a test automation framework? Select three options

A. Providing reusable test execution support

B. Providing test reporting support

C. Managing configuration or test data

D. Replacing test analysis and test design

E. Guaranteeing that the SUT contains no defects

ANSWER: A B C

Explanation:

Providing reusable support for test execution, test reporting, and configuration or test-data management are useful responsibilities of a test automation framework. A framework gives tests a consistent structure and centralizes technical concerns that would otherwise be duplicated in individual scripts. For example, it may create browser sessions, load environment-specific configuration, capture evidence after failures, and produce execution reports. The framework supports the implementation and execution of tests; it does not replace the need to design test cases or decide whether the product behaviour is correct. See the Selenium test practices documentation.

QUESTION NO: 4

Which of the following statements about Selenium Grid are true? Select two options

A. It can run WebDriver tests on remote machines

B. It can support execution against different browser and platform combinations

C. It automatically converts manual test cases into automated tests

D. It is primarily a capture/playback browser extension

E. It removes the need to maintain automated test code

ANSWER: A B

Explanation:

Selenium Grid enables WebDriver tests to run on remote machines and can distribute execution across different browser and platform combinations. This makes it useful when a test suite must provide cross-browser coverage or when parallel execution is needed to reduce execution time. A Grid session is requested with desired browser capabilities or options, and the Grid routes that request to a suitable available node.

Selenium Grid does not create test cases, record user actions, or eliminate the need for a test framework and test code. It is execution infrastructure for remote and distributed browser automation. The official documentation describes Grid as a way to run tests on different machines against different browser and operating-system combinations. See the Selenium Grid documentation.

QUESTION NO: 5

Assume that within the ACME com homepage there exists the following edit box:

```
<input id="id-search-field" name="q" type="search" role="textbox"
class="search-field" placeholder="Search" value="" tabindex="1"
/>
```

and that it is the only text box with the name " q " Look at the following lines of code:

```
I.  drv.get_screenshot_as_file("python_homepage.png")
II. from selenium import webdriver
III. edt_search = drv.find_element_by_name("q")
IV. drv = webdriver.Ie()
V.  drv.get("http://acme.com")
```

What is the correct sequence of above lines to take a screenshot of ACME.com homepage?

A. IV, II, V, I. III

B. II, IV. I, V. III

C. II, IV, III, V, I

D. II. IV. V. I. III

ANSWER: C

Explanation:

The correct sequence is **II, IV, III, V, I**. A Selenium test must first create the browser driver, because every subsequent browser action requires an active `WebDriver` session. It must then navigate that driver to the ACME.com homepage. Once the page has been loaded, locating the unique text box through its `name` attribute confirms that Selenium is operating on the intended page and that the required page element is available. Selenium's `findElement` operation is performed through the initialized driver and therefore necessarily follows browser creation and navigation.

After the page context and its edit box have been established, the driver can be cast to `TakesScreenshot` and asked for a screenshot file. The screenshot is initially returned as a file object or temporary file, so the final operation copies that captured file to its intended persistent destination. This ordering ensures that the saved image represents the loaded homepage rather than an uninitialized browser or an incomplete navigation state. Selenium documents screenshot capture through the `TakesScreenshot` interface at Selenium TakesScreenshot API; element lookup is described in the Selenium locator documentation.

QUESTION NO: 6

In the web application you are testing, you need to select several options in a dropdown menu

Which of the following is the BEST approach for selecting a dropdown option using WebDriver?

A. Use the switch_to class to switch to the dropdown element, and then click on the option in the dropdown

B. Click on the dropdown option using its relative XPath

C. Click on the dropdown option using its absolute XPath

D. Click on the dropdown element and then click on the option in the dropdown

E. Use Selenium's Select class with the dropdown element, then select each required option by visible text, value, or index.

ANSWER: E

Explanation:

Use Selenium's `Select` support class for a standard HTML `<select>` dropdown, selecting entries through a stable semantic property such as their visible text, value, or index. This is the purpose-built WebDriver API for select lists and is particularly appropriate when the control permits multiple selections: after locating the `<select>` element, a test can create a `Select` object and call methods such as `select_by_visible_text()` for each required entry. This expresses the test's intent directly and lets Selenium interact with the browser's native select-list behavior rather than relying on fragile page-structure details. Selenium documents support for selecting and clearing options in both ordinary and multi-select lists, including use of the `multiple` attribute to identify controls that support more than one selected option. In practice, locate

the dropdown with a resilient locator, then use the `Select` API to select the required values. See Selenium's Select List documentation and the Selenium Select API reference.

QUESTION NO: 7

If you need to test the content within a specific frame in a web page, which one of the following is the BEST approach for gaining access to the frame?

A. Get handles for the open frames and switch to the frame with that handle

B. Execute JavaScript window_open code to open the desired frame

C. Create a WebDriver object for the frame and use the gotframeQ method using the object

D. Use the switch_to class to switch to the desired frame

ANSWER: D

Explanation:

Use the `switch_to` class to switch to the desired frame. Selenium WebDriver operates in the context of the top-level document by default, so elements located inside an `iframe` or frame cannot be located until the driver changes its current browsing context to that frame. In Python, this is done through `driver.switch_to.frame(...)`. The frame may be identified with a WebElement, its name or ID attribute, or an index where appropriate. For example, a robust approach is to wait until the frame is available and then switch to it: `WebDriverWait(driver, 10).until(EC.frame_to_be_available_and_switch_to_it((By.ID, "frame_id")))`. Once the context has changed, normal WebDriver locators and actions apply to the content contained in that frame. If the test subsequently needs to interact with the parent document, it should restore the appropriate context with `driver.switch_to.parent_frame()` or `driver.switch_to.default_content()`. Selenium's official documentation describes frame switching as a required context change before interacting with framed content. See the Selenium frames documentation and the Python SwitchTo API reference.