1. What is the role of “Desired Capabilities” in Appium, and Why they are Important?

Answer: Desired Capabilities are Shown in the form of JSON and its in the form of key-value pairs which enable the Appium to interact with the Devices/Emulators/Simulators.

Importance:

a. Cross-Platform: Based on the desired Capabilities appium is going to interact with the desired device or application. Based on user interest the Device is chosen either it is Android or iOS.

b. Built-in functionality like noReset to make sure before the test all the previous doings are in reset.

c. Correctness of choosing devices where we have to provide the key-value pairs like appPackage, appActivity, deviceName, etc. Without the desired Capabilities appium is not going to establish connection with the server to the device.

Example:

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("platformName", "Android");
capabilities.setCapability("platformVersion","13.0");
capabilities.setCapability("deviceName", "Pixel_6_Pro");
capabilities.setCapability("automationName", "UiAutomator2");
capabilities.setCapability("app", "path/to/app.apk");
capabilities.setCapability("noReset", true);
```

2. In what programming languages can you write test scripts for Appium, and how does Appium support this flexibility?

Answer: Programming Languages are Java, Python, JavaScript, Ruby, C#, PHP, Kotlin, Swift, and Go.

Flexibility Support of Appium:

a. Appium provides Client Libraries for different programming language which acts as the bridge between the test scripts written in any programming language and the Appium Server.
b. Based on Web Driver protocol Appium server communicates with the test scripts via HTTP Request.
c. Cross platform compactibility: Based on desired Capabilities we can choose the type of device to be tested, Regardless of the programming language used.
d. We have the ability to choose the drivers like UiAutomator2 for Android, XCUITest for iOS where we don't have to change programming language.

3. What is the purpose of Appium Inspector, and How does it Assist testers in creating automation Scripts?

Answer. Appium Inspector comes with the Appium, Which is a powerful tool and Main purpose of Appium enables testers to gather UI Element locator identification for automating Real Devices/Virtual Devices based on Desired Capabilities set by User in the test Script.
Purpose of Appium Inspector:
a. Inspecting the UI element shown in the inspector.
b. We can identify the locator address.
c. It supports both the Android and iOS apps.
d. Visual Representation of the Mobile Automation.
Appium Assists testers in creating automation Scripts by:
a. While inspecting each of the element in the appium inspector which shows the necessary details of the elements. Necessary details for the device automation are xpath, text, cssSelector, className, name etc.
b. Testers can interact with the apps before choosing the correct apps in the scripts.

c. Locators are Compactible for both Android and iOS if we are testing same app.

4. Describe the Process of locating and interacting with UI elements using Appium?

Answer. Start the Appium server then Run the Device and open the appium inspector with enabling CORS(https://youtu.be/_M9qW5YtETE).

```
MobileElement element =
driver.findElement(By.id("com.example:id/username"));
MobileElement element =
driver.findElement(MobileBy.AccessibilityId("loginButton
"));
MobileElement element =
driver.findElement(By.xpath("//android.widget.Button[@te
xt='Submit']"));
MobileElement element =
driver.findElement(By.className("android.widget.TextView
"));
MobileElement element =
driver.findElement(MobileBy.AndroidUIAutomator("text(\"L
ogin\")"));
```

Operating the Element:

For clicking the element:

```
    element.click();
```

Entering the input:

```
    element.sendKeys("test_user");
```

Clearing the Field:

```
    element.clear();
```

Getting the Text:

```
    String value = element.getText();
```

Element is Displayed/Enabled:

```
    boolean isVisible = element.isDisplayed();
    boolean isEnabled = element.isEnabled();
```
5. Explain the concept of “implicit wait” and “explicit wait” in Appium?

Answer. These wait commands are used to reduce the Dynamic loading or synchronization problems. These ensure the elements are perfectly loaded before action to be performed by user.

Implicit Wait:

a. Performs in the global Scope means applied to all the elements.

b. It is less flexible or fixed time out.

c. It slow down the execution if uses excessively.

d. It is mostly used for simple and uniform time wait.

Explicit Wait:

a. It can be specific for certain element.

b. It is Highly flexible.(Can have user defined conditions).

c. Wait for various conditions(Checking for clickable or visibility).

d. Conditional and Dynamic specific wait.