**1) What is Selenium and what is composed of?**

Selenium is a suite of tools for automated web testing. It is composed of

**Selenium IDE** (Integrated Development Environment): It is a tool for recording and playing back. It is a Firefox plug-in.

**Selenium Remote Control (RC)** – Selenium RC is a server that allows a user to create test scripts in the desired programming language. It also allows executing test scripts within the large spectrum of browsers.

Selenium RC requires server to be started before executing the test scripts.

Selenium RC doesn't explicitly provide aids to handle alerts, navigations, and dropdowns.

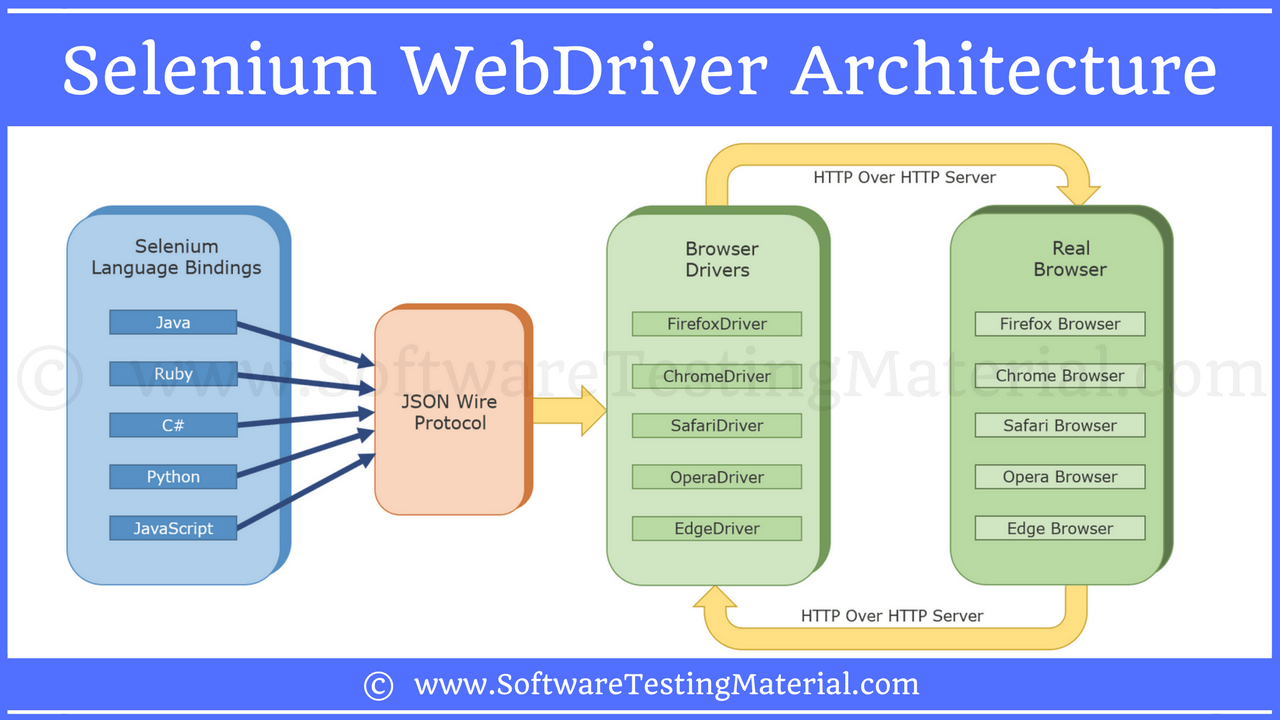
Selenium RC doesn't support testing of iPhone/Android applications.

**Web Driver**: Web Driver is a different tool altogether that has various advantages over Selenium RC. Web Driver directly communicates with the web browser and uses its native compatibility to automate. It provides the APIs for a variety of languages like Java, .NET, PHP, etc. With most of the browsers Web driver and RC work.

**Grid**: With the help of Grid you can distribute tests on multiple machines so that test can be run parallel which helps in cutting down the time required for running in browser test suites.

**Selenium Architecture?**

Selenium Web Driver uses the JSON wire protocol to communicate between client libraries and different drivers (that is, Firefox Driver, IE Driver, Chrome Driver, and so on) implementations.



**Pandas:**

Data Science or Data Analytics is a process of analyzing large set of data points to get answers on questions related to that data set.

**Pandas deals with the following three data structures −**

Series

DataFrame

Panel

|  |  |  |
| --- | --- | --- |
| **Data Structure** | **Dimensions** | **Description** |
| Series | 1 | 1D labeled homogeneous array, size is immutable. |
| Data Frames | 2 | General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns. |
| Panel | 3 | General 3D labeled, size-mutable array. |

**Pandas-DataFrames:**

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Features of DataFrame**

Potentially columns are of different types

Size – Mutable

Labeled axes (rows and columns)

Can Perform Arithmetic operations on rows and columns

**Pandas-Panel:**

A panel is a 3D container of data. The term Panel data is derived from econometrics and is partially responsible for the name pandas − pan(el)-da(ta)-s.

The names for the 3 axes are intended to give some semantic meaning to describing operations involving panel data. They are −

items − axis 0, each item corresponds to a DataFrame contained inside.

major\_axis − axis 1, it is the index (rows) of each of the DataFrames.

minor\_axis − axis 2, it is the columns of each of the DataFrames.

**Create a Series from dict**

A dict can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index. If index is passed, the values in data corresponding to the labels in the index will be pulled out.

s2=pd.Series({'a':1,'b':2,'c':3})

print(s2)

**Observe** − Dictionary keys are used to construct index.

**Create a Series from Scalar**

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index

s4=pd.Series(5,[0,1,2])

print(s4)

**Data Frame:**

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

**Features of DataFrame**

Potentially columns are of different types

Size – Mutable

Labeled axes (rows and columns)

Can Perform Arithmetic operations on rows and columns

**Create a DataFrame from Dict**

import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}

df = pd.DataFrame(data)

print df

**Create a DataFrame from List of Dicts**

List of Dictionaries can be passed as input data to create a DataFrame. The dictionary keys are by default taken as column names.

import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

df = pd.DataFrame(data)

print df

**ExcelWriter**: Class for writing DataFrame objects into excel sheets, default is to use xlwt for xls, openpyxl for xlsx. See DataFrame.to\_excel for typical usage

**Example**:

writer = pd.ExcelWriter("D:/Test/DataFrame.xlsx")

df2.to\_excel(writer,'kumar',index=True)

df1.to\_excel(writer,'suresh',index=True)

writer.save()

**Column Selection, Addition, and Deletion:**

data=[['a',1],['b',2],['c',3]]

columns=['Name','age']

df1=pd.DataFrame(data=data,columns=columns)

data={'Name':['suresh','kumar'],'Age':[22,30]}

df2=pd.DataFrame(data)

**#Addition of column**

df2['Gender']=pd.DataFrame(['Male','Male'])

print(df2)

**#Addition of Row**

df2=pd.DataFrame({'Name':['kadi'],'Age':[32]})

df1=df1.append(df2)

**#Selection of column**

print(df2['Name'])

**#Selection of Row by label/Index**

print (df2.loc['b'])

**#Selection of Row by interger location**

Print(df2.iloc[0])

**#Selection of multiple Rows by interger location**

Print(df2.iloc[0:1])

**##Deletion of column**

del df2['Age']

**Deletion of Rows**

Use index label to delete or drop rows from a DataFrame. If label is duplicated, then multiple rows will be dropped.

If you observe, in the above example, the labels are duplicate. Let us drop a label and will see how many rows will get dropped.

df1=df1.drop(0)

print(df1)

**NumPy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

**How to handle multiple windows in Python?**

You can do it by using **window\_handles** and **switch\_to\_window** method.

Before clicking the link first store the window handle as

window\_before = driver.window\_handles[0]

After clicking the link store the window handle of newly opened window as

window\_after = driver.window\_handles[1]

Then execute the switch to window method to move to newly opened window

driver.switch\_to\_window(window\_after)

**Scenarios that cannot be automated using Selenium?**

1)Video streaming scenarios

2)Code reading scenarios - QR Code

3)Bar code scanners

4)Hardware testing

5)CAPTCHA image

**dir() function in Python:**

dir() is a powerful inbuilt function in Python3, which returns list of the attributes and methods of any object (say functions , modules, strings, lists, dictionaries etc.)

**Syntax**:

dir({object})

**Help function in Python**

The python help function is used to display the documentation of modules, functions, classes, keywords etc.

The help function has the following syntax:

**Help ([object])**

If the help function is passed without an argument, then the interactive help utility starts up on the console.

Let us check the documentation of the print function in python console.

For e.g.: Help (print)

What is **flask** in Python**?**

Flask is a web application framework written in Python.

**Exploratory Testing** is an important activity in an agile environment as it can help software testers to keep up with the rapid development pace of agile software projects.

First, a brief intro on agile methodology and exploratory testing:

In agile methodology, software is released in small iterations. Each iteration goes through planning, estimation, development, integration, testing and release. Because of frequent releases, test automation becomes ever so important as developers need to get quick feedback on the status of the application. The automated checks serve as regression tests to ensure that with each release the software has not regressed.

Exploratory Testing is defined as simultaneous learning, test design and test execution.

Exploratory Testing is also complementary to test automation; that is while automated checks are checking for regression issues, Exploratory Testing focuses on new features which have been developed. This is important because each sprint typically lasts only couple of weeks, which doesn’t allow sufficient time for scripting tests cases and executing them later against the application. On the other hand, exploratory testing in agile environment allows testers to get familiar with the domain and the application and on each iteration, that understanding is enhanced and hence testers become more efficient.Exploratory Testing is an important activity in an agile environment as it can help software testers to keep up with the rapid development pace of agile software projects.

First, a brief intro on agile methodology and exploratory testing:

In agile methodology, software is released in small iterations. Each iteration goes through planning, estimation, development, integration, testing and release. Because of frequent releases, test automation becomes ever so important as developers need to get quick feedback on the status of the application. The automated checks serve as regression tests to ensure that with each release the software has not regressed.

Exploratory Testing is defined as simultaneous learning, test design and test execution.

Exploratory Testing is also complementary to test automation; that is while automated checks are checking for regression issues, Exploratory Testing focuses on new features which have been developed. This is important because each sprint typically lasts only couple of weeks, which doesn’t allow sufficient time for scripting tests cases and executing them later against the application. On the other hand, exploratory testing in agile environment allows testers to get familiar with the domain and the application and on each iteration, that understanding is enhanced and hence testers become more efficient.

**Important points:**

Exploratory Testing is an important activity in an agile environment as it can help software testers to keep up with the rapid development pace of agile software projects.

In agile methodology, software is released in small iterations. Each iteration goes through planning, estimation, development, integration, testing and release

Exploratory Testing is defined as simultaneous learning, test design and test execution.

**Automation Challenges:**

Testing always under time constraint

Skilled resources for the Automation

Requirements keep changing

Effective Team collaboration

Choosing the right tool

**What is POM?**

Page Object Model is a design pattern to create Object Repository for web UI elements.

Under this model, for each web page in the application, there should be corresponding page class.

This Page class will find the Web Elements of that web page and also contains Page methods which perform operations on those Web Elements.

Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay ().

**When will you automate a test?**

Automation in preferred in following cases

Repetitive Tasks

Smoke and Sanity Tests

Test with multiple data set

Regression test cases

**On what basis you can map the success of automation testing?**

By following criteria, the success of automation testing can be mapped

1) Defect Detection Ratio

2) Automation execution time and time savings to release the product

3) Reduction in Labour & other costs

**Type of Objects in a programming language:**

1. Attributes – color, gender, Nationality etc…

2. Actions-Walk, see, smell etc…

No need to call \_\_init\_\_ method explicitly, whenever class is constructed init method is called automatically.

We have to create an object for the ’student’ class.

**Explaining about Automation Framework**

**Programming Language**

* We are using Java language in our project, even though selenium supports multiple languages since huge communities/groups available in net )

**Type of Framework,**

* We are using Unittestest ~~/ Hybrid Framework~~ by using page object model
* As per Page Object Model we have maintained class for every web page, each web page separate class and that holds the functionality and members of that web page,
* We have separate classes for every individual test.
* We have separate packages for pages & tests

**Base Class:**

* It contains all the common functions used by all pages, this class responsible for loading configuration from property files/setup.json file
* Initializing web driver, implicit wait, extent reports, logs, and also creating objects of file input stream to read data from ~~property files~~/Testdata.json

**Page Object Design Model, Functions (i.e. Utils),**

* Repetitive tasks will be created as functions such as (i.e. Waits, Actions, Capture screen shots, Accessing Excel Sheets and sending emails) and will use entire framework
* Achieving re usability

**~~Property files~~,/Setup.json**

* It stores the information that reminds static throughout the framework such as (i.e. browser specific info, application url and screen shot path etc.,)

**Unittest/Pytest,**

* We use unittest annotation for assertion, grouping and parallel execution etc.,

**Parameterization (i.e. Json /Excel files),**

* All the historical test data will kept in side json /excel sheet, by using Apache POI we handle data driven testing.

**~~Maven,~~**

* ~~We use maven for build execution and dependency purpose~~

**Error Screenshots,Sending Emails,**

**Version Control (i.e. Git, SVN),**

* We use Git to store our code repository

**Continuous Integration (i.e. Jenkins)**

* By using Jenkins we execute test cases on daily basis and also nightly execution based on the schedule,
* Results/Reports will be send to every stack holders/peers

**HTML TestRunner Reports:**

For the reporting purpose, we are using HTML TestRunner Reports. It generates beautiful HTML reports. We use the HTML reports for maintaining logs and also to include the screenshots of failed test cases in the HTML Report.

**Defect Life Cycle:**

New, Assigned, Open, Fixed, Retest, Verified and Closed.

Retest🡪Reopen

Open🡪Deferred, Duplicated, Rejected and Not a Bug.

**Syntax to capture and save the screenshot.**

File screenshotFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

**Syntax to store it in our local drive**

FileUtils.copyFile(screenshotFile, new File("filename\_with\_path"));

**4. A) How to scroll down to the bottom of the page?**

browser.execute\_script ("arguments[0].scrollintoview(true);",webelemet)

**2) How will you find an element using Selenium?**

In Selenium every object or control in a web page is referred as an element, there are different ways to find an element in a web page they are

ID

Name

Tag

Attribute

CSS

Linktext

PartialLink Text

Xpathetc

Xpath syntax - $x(“//input[@type=’text’]”)

**STLC Life Cycle?**

Requirement Analysis

Test Planning

Test case development

Test Environment setup

Test Execution

Test Cycle closure

**Explain Entry and Exit Criteria?**

**Examples for Entry Criteria:**

Verify if the Test environment is available and ready for use.

Verify if test tools installed in the environment are ready for use.

Verify if Testable code is available.

Verify if Test Data is available and validated for correctness of Data.

**Examples of Exit Criteria:**

Verify if all tests planned have been run.

Verify if the level of requirement coverage has been met.

Verify if there are NO Critical or high severity defects that are left outstanding.

Verify if all high risk areas are completely tested.

Verify if software development activities are completed within the projected cost.

Verify if software development activities are completed within the projected timelines.

**How a tester should be?**

Testing the application through customers eye.

Tester should think ‘Out of the box’.

Testers job is not only finding defects in the software,but also to improve the quality of the product.

**Difference between Test Strategy and Test Plan?**

**Test strategy** is a high level document which defines the approach for software testing. It is basically derived from the Business Requirement document. Test strategy is developed by project manager or business analyst. It is kind of static document which sets the standards for testing so not updated often.

**Test plan** is derived from SRS (Software Requirement Specification) which is prepared by test lead or manager. The main goal of test plan is to include all the details related to testing such as what to test, when to test, how to test and who will be the tester. Test plan is often not updated but if there is some new feature or change is introduced then it has to be updated accordingly.

**What are Test Deliverables?**

Test cases Documents, Test Plan, Testing Strategy, Test Scripts, Test Data, Test Trace-ability Matrix, Test Results/reports, Test summary report, Install/config guides, Defect Report and Release notes

**3) Mention what is the use of X-path?**

X-Path is used to find the Web Element in web pages. It is also useful in identifying the dynamic elements.

Refer Complete Guide on XPath

**Siblings in Selenium-Web Driver:**

**Parent:**

Parent filter selects the parent element of the current element.

**Syntax**://tagname/parent::tagname

**Usage**: browser.find\_element\_by\_xpath(“//h1/parent::div”)

This XPath will select *parent div element* of h1 tag

**Child:**

Child filter selects the child element of the current element.

**Syntax**://tagname/child::tagname

**Usage**: browser.find\_element\_by\_xpath(“//footer/ child::p”)

This XPath will select *child p element* of footer tag

**following Sibling:**

Following -sibling filter selects all siblings’ elements after the current element.

**Syntax**://tagname/ following-sibling::tagname

**Usage**: browser.find\_element\_by\_xpath(“//tr[6]/ following-sibling::tr”)

This XPath will select *all tr elements of same parent* after sixth tr tag.

**Following:**

Following filter selects all elements after the current element.

**Syntax**://tagname/ following::tagname

**Usage**: browser.find\_element\_by\_xpath(“//tr[6]/ following::\*”)

This XPath will select *all elements* after sixth tr tag.

**Preceding Sibling:**

Preceding-sibling filter selects all siblings’ elements before the current element.

**Syntax**://tagname/ preceding-sibling::tagname

**Usage**: browser.find\_element\_by\_xpath(“//tr[6]/ preceding-sibling::tr”)

This XPath will select *all tr elements of same parent* before sixth tr tag.

**Preceding:**

Preceding filter selects all elements before the current element.

**Syntax**://tagname/ preceding::tagname

**Usage**: browser.find\_element\_by\_xpath(“//tr[6]/ preceding::\*”)

This XPath will select *all elements* before sixth tr tag.

**4) Explain the difference between single and double slash in X-path?**

Single slash ( / ) start selection from the document node

It allows you to create ‘absolute’ path expressions

Double Slash ‘// ’

Double slash ( // ) start selection matching anywhere in the document

It enables to create ‘relative’ path expressions

5) List out the technical challenges with Selenium? Technical challenges with Selenium are

Selenium supports only web based applications

It does not support the Bitmap comparison

For any reporting related capabilities have to depend on third party tools

**6) What is the difference between verify and assert commands?**

**Assert**: Assert allows checking whether an element is on the page or not. The test will stop on the step failed, if the asserted element is not available. In other words, the test will terminate at the point where check fails.

**Verify**: Verify command will check whether the element is on the page, if it is not then the test will carry on executing. In verification, all the commands are going to run guaranteed even if any of test fails.

**6. What is the difference between driver. Close () and driver. Quit () methods?**

driver. close(): To close current Web Driver instance – Close current tab (**Note**: if browser has only one tab opened –Closing the tab will close the browser)

driver.quit(): To close all the opened Web Driver instances – quit browser window

**6a) how do you select an option from drop down list?**

From selenium.webdriver.support.ui import Select

select\_by\_value

select\_by\_index

select\_by\_visible\_text

**Example**:

Select = Select (driver.find\_element\_by\_id ('fruits01'))

**# select by visible text**

select.select\_by\_visible\_text('Banana')

**# Select by value**

select.select\_by\_value ('1')

**Actionchains in Selenium?**

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

ActionChains are a way to automate low level interactions such as

mouse movements, mouse button actions, key press, and context menu interactions.

This is useful for doing more complex actions like hover over and drag and drop.

actions = ActionChains(driver)

actions.move\_to\_element(menu)

actions.click(hidden\_submenu)

actions.perform()

**Working with Checkboxes in Selenium Web driver in Python – Example?**

We can first see if the checkbox is selected using **is\_selected()** method. Then using click method we can perform the operations such as selecting or deselecting the checkboxes as illustrated in the below example.

**Is\_Enabled()** is the method used to verify if the web element is enabled or disabled within the webpage. **Is\_Enabled()** is primarily used with buttons. **Is\_Selected()** is the method used to verify if the web element is selected or not. **Is\_Selected()** method is predominantly used with radio buttons, dropdowns and checkboxes.

**Instance Method:**

Instance method (i.e. method ()) must be called by a class instanceand we have pass ‘**self’** as the argument, this is most common method in python. Any method you create will automatically be created as an instance method.

Unlike regular methods, both static and class methods are decorators.

**Class method:**

A class method is a method which is bound to the class and not the object of the class.

They have the access to the state of the class as it takes a class parameter that points to the class and not the object instance.

It can modify a class state that would apply across all the instances of the class. For example it can modify a class variable that will be applicable to all the instances.

**Static method:**

A static method is also a method which is bound to the class and not the object of the class.

A static method can’t access or modify class state.

It is present in a class because it makes sense for the method to be present in class.

**Types of Asserts in unittest?**

assertAlmostEquel, assertAlmostEquels, assertEquel, assertEquels, assertFalse,assertGreater, assertGreaterEquel,assertIn,assertTrue,assertNotEquel,assertRaises.

**With Keyword in Python?**

with open('/etc/passwd') as f:

for line in f:

print(line)

The “**with**” statement invokes what Python calls a “context manager” on f. That is, it assigns f to be the new file instance, pointing to the contents of /etc/password. Within the block of code opened by “with”, file is opened, and can be read from freely.

However, once Python exits from the “with” block, the file is automatically closed.

**Explanation 2:**

**With** statement is used to wrap the execution of a block of code within methods defined by the context manager.

Context manager is a class that implements \_\_enter\_\_ and \_\_exit\_\_ methods.

First the \_\_enter\_\_ method is called, then the code within **with** statement is executed and finally the \_\_exit\_\_ method is called. \_\_exit\_\_ method is called even if there is an error. It basically closes the file stream.

**Default Arguments in Python:**

Python allows function arguments to have default values; if the function is called without the argument, the argument gets its default value.

Default arguments:

Python has a different way of representing syntax and default values for function arguments. Default values indicate that the function argument will take that value if no argument value is passed during function call. The default value is assigned by using assignment (=) operator. Below is a typical syntax for default argument. Here, parameter has a default value.

**Example:**

Def defaultargs(name, age=34,gender='Male'):

Print name,age,gender

defaultargs('suresh',34,'male')

**Explain \*args and \*\*kwargs in Python?**

1)\*args:

The special syntax \*args in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list.

**The syntax is to use the symbol \* to take in a variable number of arguments; by convention, it is often used with the word args.**

What \*args allows you to do is take in more arguments than the number of formal arguments that you previously defined. With \*args, any number of extra arguments can be tacked on to your current formal parameters (including zero extra arguments).

For example: we want to make a multiply function that takes any number of arguments and able to multiply them all together. It can be done using \*args.

Using the \*, the variable that we associate with the \* becomes an iterable meaning you can do things like iterate over it, run some higher order functions such as map and filter, etc.

defsuresh(\*argv):

fori in argv:

printi

suresh('suresh1','kumar')

2)\*kwargs:

The special syntax \*\*kwargs in function definitions in python is used to pass a keyworded, variable-length argument list. **We use the name kwargs with the double star. The reason is because the double star allows us to pass through keyword arguments (and any number of them).**

A keyword argument is where you provide a name to the variable as you pass it into the function.

One can think of the kwargs as being a dictionary that maps each keyword to the value that we pass alongside it. That is why when we iterate over the kwargs there doesn’t seem to be any order in which they were printed out.

defkumar(hi1,\*\*kwargs):

fork,v in kwargs.items():

printk,v

kumar(name='suresh', age=34)

**What is diff between function and methods in Python?**

**Answer:**

**Method** -Instance or Object of the Class

------

Method can't be called by its Name

We need to invoke the class by reference of that class in which it is defined

i.e., method is defined within and hence they are dependent on that class

(DEPENDENT)

It may or may not return any data

**Function**

**-------**

Functions can be called by only its Name as it is defined independently (INDEPENDENT)

Functions does not deal with class

It may or may not return any data

**How to find id of a variable?**

a=1

Print Id(a) – it returns integer (long integer)

**How to amend Key in a Dictionary?**

**Syntax:**dictionary[new\_key] = dictionary[old\_key]

del dictionary[old\_key]

a = {"a":1,"b":2}

a["c"]=a["b"]

del a["b"]

Print a

**6c) Framework folder structure**

Env - where you define the browsers, which are using for execution

Syslibrary - Common method is place here to retrieve the data from XML files

Testscripts - Individual test scripts will code here

Screenhots - will store screenshots for failure test cases

Testsuite - where you keep all the test cases here to run as a suite

Testreport - once ran the suite file, html report is generated and saved here

Data- all your test data will be placed here

Object - all your object properties will be place here

Library - All the classes and corresponding methods will be written here

7)What is JUnit Annotations and what are different types of annotations which are useful?

Setup

tearDown

Setupclass

tearDownclass

**Python Concepts?**

1. Os and sys
2. List
3. Dictionaries
4. Reading data from a file
5. Classes and Methods
6. Unittest
7. Json
8. Traceback
9. HTMLTestRunner
10. Pytest

8) What are the advantages of Selenium?

It supports C#, PHP, Java, Perl, Python

It supports different OS like Windows, Linux and Mac OS

It has got powerful methods to locate elements (Xpath, DOM,and CSS)

**8. A) How to upload documents in Selenium?**

Windows file uploads does not support selenium, so we have rely on AutoIT.

Download and Install AutoIT, then go to Editor, here we use ‘SciTE’.

ControlFocus(‘title’,’text’,’controlID’)- ‘Fileupload’,””,’Edit1’

ControlSetText(‘title,text’,’controlID,’newtext’’)-‘Fileupload’,””,’Edit1’,’location of the file’

ControlClick(‘(‘title,text’,’controlID)-‘Fileupload’,””,Button1’

Do save the file in any appropriate folder

We cannot execute .au3 files as it is, so we have to compile it first based on x32 or x64 bit.

It converts into .exe

Provide above .exe at run time in selenium (os.system (location of the executable path))

9) Why testers should opt for Selenium and not QTP? Selenium is more popular than QTP as

Selenium is an open source whereas QTP is a commercial tool

Selenium is used specially for testing web based applications while QTP can be used for testing client server application also

Selenium supports Firefox, IE, Opera, and Safari on operating systems like Windows, Mac, and Linux etc. however QTP is limited to Internet Explorer on Windows.

Selenium supports many programming languages like Ruby, Perl, and Python whereas QTP supports only VB script

21) What are the features of TestNG and list some of the functionality in TestNG which makes it more effective?

TestNG is a testing framework based on JUnit and NUnit to simplify a broad range of testing needs, from unit testing to integration testing. And the functionality which makes it efficient testing framework are

Support for annotations

Support for data-driven testing

Flexible test configuration

Ability to re-execute failed test cases

**22) Mention what is the difference between implicit wait and explicit wait?**

Implicit Wait: Sets a timeout for all successive Web Element searches. For the specified amount of time it will try looking for element again and again before throwing a **NoSuchElementException**. It waits for elements to show up.

Explicit Wait: It is a one-timer, used for a particular search

Implicit wait is set for the entire duration of the web Driver object. Suppose, you want to wait for a certain duration, let's say 5 seconds before each element or a lot of elements on the webpage load.

Now, you wouldn't want to write the same code again and again. Hence, implicit wait. However, if you want to wait for only one element, use explicit.

The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception". The default setting is 0. Once we set the time, web driver will wait for that time before throwing an exception.

**Explicit wait code:**

from selenium import webdriver

from selenium.webdriver.common.by import By

fromselenium.webdriver.support.ui import WebDriverWait

fromselenium.webdriver.support import expected\_conditions as EC

browser = webdriver.Chrome()

browser.get('https://google.com')

wait= WebDriverWait(browser,2)

wait.until(EC.presence\_of\_element\_located(By.ID,'button'))

**24) Explain what is the difference between find elements () and find element ()?**

Find element ():

It finds the first element within the current page using the given “locating mechanism”. It returns a single Web Element

Find Elements (): Using the given “locating mechanism” find all the elements within the current page. It returns a list of web elements.

**30) What is Object Repository?**

An object repository is an essential entity in any UI automations which allows a tester to store all objects that will be used in the scripts in one or more centralized locations rather than scattered all over the test scripts.

**31) Explain how Selenium Grid works?**

Selenium Grid sent the tests to the hub. These tests are redirected to Selenium Web driver, which launch the browser and run the test. With entire test suite, it allows for running tests in parallel.

32) Can we use Selenium grid for performance testing?

Yes. But not as effectively as a dedicated performance testing tool like Load runner.

33) List the advantages of Web driver over Selenium Server?

If you are using Selenium-Web Driver, you don’t need the Selenium Server as it is using totally different technology

Selenium Server provides Selenium RC functionality which is used for Selenium 1.0 backwards compatibility

Selenium Web driver makes direct calls to browser using each browsers native support for automation, while Selenium RC requires selenium server to inject Javascript into the browser

34) Mention what are the capabilities of Selenium Web Driver or Selenium 2.0?

WebDriver should be used when requiring improvement support for

Handling multiple frames, pop ups, multiple browser windows and alerts

Page navigation and drag & drop

Ajax based UI elements

Multi browser testing including improved functionality for browser not well supported by Selenium 1.0

89) Mention why to choose Python over Java in Selenium?

Few points that favor Python over Java to use with Selenium is,

Java programs tend to run slower compared to Python programs.

Java uses traditional braces to start and ends blocks, while Python uses indentation.

Java employs static typing, while Python is dynamically typed.

Python is simpler and more compact compared to Java

86) To enter values onto text boxes what is the command that can be used?

To enter values onto text boxes we can use command send keys ()

87) How do you identify an object using selenium?

To identify an object using Selenium you can use

isElementPresent(String locator)

isElementPresent takes a locator as the argument and if found returns a Boolean

Implicit wait is set for the entire duration of the web Driver object. Suppose, you want to wait for a certain duration, let's say 5 seconds before each element or a lot of elements on the webpage load.

Now, you wouldn't want to write the same code again and again. Hence, implicit wait. However, if you want to wait for only one element, use explicit.

The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception". The default setting is 0. Once we set the time, web driver will wait for that time before throwing an exception.

**Testing Activities:**

Test Planning

Test Implementation

Maintainability, Reliability, Flexibility, Efficiency, Portability, Robustness, and Usability - these are the main attributes in test automation

Reliability is the degree to which an assessment tool produces stable and consistent results.

* **Oops concepts**
  + Object-oriented
  + Inheritance
  + Encapsulation
  + Polymorphism
  + Abstraction

**17. What is a “unit test” in Python?**

The unit testing framework of Python is known as “unittest”.

It supports the sharing of setups, automation testing, shutdown code for tests, and aggregation of tests into collections, among others.

**21. How do you copy an object in Python?**

copy.copy () or copy.deepcopy()

**22. What is tuple?**

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists.

The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. ‘Tuple' object does not support item assignment 'tuple' object doesn't support item deletion

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2= (1, 2, 3, 4, 5);

tup3= ("a", "b", "c", "d")

**23. Built-in Methods in List?**

Append, insert, remove, pop, reverse, count and index

**24. Built-in Methods in Tuple?**

Count and index

**Range () –** This returns a list of numbers created using range () function.

**xrange() –** This function returns the generator object that can be used to display numbers only by looping. Only particular range is displayed on demand and hence called “lazy evaluation“.

Both are implemented in different ways and have different characteristics associated with them. The points of comparisons are:

1. Return Type Memory Operation Usage Speed
2. Memory
3. Operation Usage
4. Speed

**Difference between ‘Copy’ and ‘deep copy’**

The difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists or class instances):

* A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original.
* A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

**Deep copy:** Deep copy creates a copy of the object and elements of the object

**Shallow Copy:** Shallow copy creates a copy of the object but references each element of the object

Here's a little demonstration:

Import copy

a = [1, 2, 3]

b = [4, 5, 6]

c = [a, b]

Using normal assignment operating to copy:

d = c

Print id(c) == id (d) # True - d is the same object as c

Print id(c [0]) == id (d [0]) # True - d [0] is the same object as c [0]

Using a shallow copy:

d = copy.copy(c)

Print id(c) == id (d) # False - d is now a new object

Print id(c [0]) == id (d [0]) # True - d [0] is the same object as c [0]

Using a deep copy:

d = copy.deepcopy(c)

Print id(c) == id (d) # False - d is now a new object

Print id(c [0]) == id (d [0]) # False - d [0] is now a new object

**How to reverse a string:**

>>> 'hello world'[::-1]

'dlrowolleh'

This is [extended slice](http://docs.python.org/2/whatsnew/2.3.html#extended-slices) syntax. It works by doing [begin:end:step] - by leaving begin and end off and specifying a step of -1, it reverses a string.

**‘Self’ in Python?**

**self** represents the instance of the class. By using the "**self**" keyword we can access the attributes and methods of the class in **python**.

**Keywords in Python?**

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

**What is ‘as’ in Python?**

**as** is used to create an alias while importing a module. It means giving a different name (user-defined) to a module while importing it.

**try**...**except** blocks are used to catch exceptions in Python.

**Single Inheritance?**

Where a Child class acquires the members of a base class. We can make use of functions and variables of base class into child class.

**Encapsulation?**

In an object oriented **python** program, you can restrict access to methods and variables. This can prevent the data from being modified by accident and is known as **encapsulation**. We cannot access private method outside the class.

class A:

def \_\_init\_\_(self):

self.\_\_updatesoftware()

def drive(self):

self.\_\_updatesoftware()

print"drive"

def \_\_updatesoftware(self):

print"software updated"

obj=A()

obj.drive()

**Private Variables**: private variables can modify inside the class and we cannot modify private variables outside the class.

class A:

def \_\_init\_\_(self):

self.\_\_name='suresh'

self.\_\_age=31

def drive(self):

print("name:",self.\_\_name)

print("age:",self.\_\_age)

defsetage(self,age1):

self.\_\_age=age1

print("age:",self.\_\_age)

obj=A()

obj.drive()

obj.\_\_age=40

obj.drive()

**Polymorphisam?**

**Polymorphism** means that different types respond to the same function. Polymorphism is a method which behaves differently for different objects.

The same operation results in different behaviors depending on the type of data is given

class dog:

def sound(self):

print"bow bow"

class cat:

def sound(self):

print"mmmmm"

defmakesound(animaltype):

animaltype.sound()

dogobj=dog()

makesound(dogobj)

catobj=cat()

makesound(catobj)

**What is Abstraction?**

Its main goal is to handle complexity by hiding unnecessary details from the user.

**How to print Even and Odd numbers:**

L= [0, 1, 2, 3, 4, 5, 6, 76]

even = []

odd = []

for j in L:

if (j%2==0):

even.append(j)

else:

odd.append(j)

Print even

Print odd

**How to swap numbers:**

X = 1

Y = 2

X,y = y,x

Print x

Print y

**Print number from 0 to 14**

x=1

For x in range (0, 15):

x = x\*1

Print x,

**Print only even numbers between 0 to 14**

x=1

For x in range (0, 15):

x = x\*2

Print x,

**How to print string number?**

String = "1+2+3"

String = int("1") +int("2") + int("3")

Print string

1. **filter()**

Filter lets us filter in some values based on conditional logic.

The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

>>>list(filter(lambda x:x>5,range(8)))

[6, 7]

1. **map()**

Map applies a function to every element in an iterable.

map() function returns a list of the results after applying the given function to each item of a given iterable (list, tuple etc.)

>>>list(map(lambda x:x\*\*2,range(8)))

[0, 1, 4, 9, 16, 25, 36, 49]

1. **reduce()**

Reduce repeatedly reduces a sequence pair-wise until no more elements are left in the sequence.

>>>fromfunctools import reduce

>>> reduce(lambda x,y:x-y,[1,2,3,4,5])

-13

**How do you configure Jenkins?**

Go to Manage Jenkins/Configure system:

Home directory – default

# No.of executors - This refers to the total number of concurrent job executions that can take place on the Jenkins machine. This can be changed based on requirements

Jenkins Location – Default will be <http://localhost:8080/>

If you want customized one, go to C:\Windows\System32\drivers\etc\host – change localhost to any other name.

## Email Notification

In the email Notification area, you can configure the SMTP settings for sending out emails. This is required for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.

Default user E-mail suffix - @causeway.com

**How to Run Jenkins as a Service?**

java -jar jenkins.war --httpPort=8282

**How to remove spaces between strings?**

**## Remove the Starting Spaces in Python**

string1="    This is Test String to strip leading space"

Printstring1

Printstring1.lstrip ()

**## Remove the Trailing or End Spaces in Python**

string2="This is Test String to strip trailing space     "

Printstring2

Printstring2.rstrip ()

**## Remove the whiteSpaces from Beginning and end of the string in Python**

string3="    This is Test String to strip leading and trailing space      "

Printstring3

Printstring3.strip ()

**## Remove all the spaces in python**

string4="&nbsp;&nbsp;&nbsp; This is Test String to test all the spaces&nbsp; "

Printstring4

Printstring4.replace (" ", "")

**Remove the space between the words in python**–O/P should be sureshkumarkadi

s = "suresh kumarkadi"

t = s.replace ('h ',"")

Print t – Result is sureshkumarkadi

**What is generator function?**

* Generator function contains one or more yield statement.
* When called, it returns an object (iterator) but does not start execution immediately.
* Methods like \_\_iter\_\_ () and \_\_next\_\_ () are implemented automatically. So we can iterate through the items using next ().
* Once the function yields, the function is paused and the control is transferred to the caller.
* Local variables and their states are remembered between successive calls.
* Finally, when the function terminates, StopIteration is raised automatically on further calls.

**# A simple generator function**

**defmy\_gen():**

n = 1

print('This is printed first')

# Generator function contains yield statements

yield n

n += 1

print('This is printed second')

yield n

n += 1

print('This is printed at last')

yield n

# **using for loop**

for item in my\_gen():

print(item)

**How do you perform string slicing?**

string = 'suresh'

print string[:-1]

print string[-1:]

s = "c(1)a(2)t(3)a(4)n(5)d(6)a(7)p(8)p(9)l(10)e(11)"

s = "(-11)c(-10)a(-9)t(-8)a(-7)n(-6)d(-5)a(-4)p(-3)p(-2)l(-1)e"

Test = s [6:] # from starting it will print the letters after 6 - apple

test1 = s [-5:] #fromending it will print the letters from -5 - apple

test2 = s [:-5] # From Ending it will print the letters after -5 - catand

test3 = s [:6] #from starting it will print the letters up to 6 - catand

Print test

Print test1

Print test2

Print test3

**How do you designed your framework?**

Explain about the folder structure and how you design each test script and how you collect individual test scripts as a suite plus generating Html reports.

**How do you design your individual test script?**

**Try:**

Open browser

Function 1

…………..

…………..

Function n

Do assertion/validation

Clear the test data to ready for next run

**Except Exception:**

Take screenshot

Exception handling

Fail the test script

**Finally**:

Close browser

**What are objects//builtin libraries in python?**

List, tuple, string, Dictionary, float, int and Boolean etc.….

**How do you upload files in Selenium?**

**Selenium cannot find ids/xpath/css for windows popup, hence** Automation of uploading files in windows can be done using Autoit.

**How to handle unexpected alert in webpage?**

Alert = browser.switch\_to\_alert()

Alert. Accept () # Click on OK button

Alert. Dismiss () # Click on Cancel button

**How to switch between two frames?**

Frame = browser.switch\_to\_frame (‘frame reference’) #It will navigate to desired frame based on the frame reference.

Frame. Click ()

**How to switch over the elements in iframes using Web Driver commands:**

Basically, we can switch over the elements in frames using 3 ways.

1)By Index

2)By Name or Id

3)By Web Element

**How to remove extensions from a file?**

Import os

Path = os.path.splitext ("E:\projectautomation\Object\locators.json")

Print path [0] - E:\projectautomation\Object\locators

Print path [1] - .json

**What is Set in Python?**

Set is a collection of items not in any particular order. A Python set is similar to this mathematical definition with below additional conditions.

* The elements in the set cannot be duplicates.
* The elements in the set are immutable (cannot be modified) but the set as a whole is mutable.
* There is no index attached to any element in a python set. So they do not support any indexing or slicing operation.

newset = set(['Telugu','English','Maths'])

newset.add('Sci')

newset.discard('Telugu')

printnewset

**What is init method in python?**

\_\_init\_\_ is a special method in Python classes, it is the constructor method for a class.

**class Student(object):**

**""" Returns a ```Student``` object with the given name, branch and year. """**

def \_\_init\_\_(self, name, branch, year):

self.name = name

self.branch = branch

self.year = year

print("A student object is created.")

defprint\_details(self):

""" Prints the details of the student. """

print("Name:", self.name)

print("Branch:", self.branch)

print("Year:", self.year)

std1 = Student('suresh','ece',2018)

std1.print\_details()

\_\_init\_\_ is called whenever an object of the class is constructed. That means whenever we will create a student object we will see the message “A student object is created” in the prompt. You can see the first argument to the method is self. It is a special variable which points to the current object (like this in C++). The object is passed implicitly to every method available in it, but we have to get it explicitly in every method while writing the methods. Example shown below. Remember to declare all the possible attributes in the \_\_init\_\_ method itself. Even if you are not using them right away, you can always assign them as None.

**What are Stubs?**

For example, in a situation where one has three different modules: Login, Home, User. Suppose login module is ready for test, but the two minor modules Home and User, which are called by Login module are not ready yet for testing. At this time, a piece of dummy code is written, which simulates the called methods of Home and User. These dummy pieces of code are the stubs.

**SQL Queries:**

The following SQL statement selects all the records in the "Customers" table:

SELECT \* FROM Customers;

The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

SELECT CustomerName, City FROM Customers;

The following SQL statement selects all (and duplicate) values from the "Country" column in the "Customers" table:

SELECT Country FROM Customers;

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

SELECT DISTINCT Country FROM Customers;

The following SQL statement lists the number of different (distinct) customer countries:

SELECT COUNT(DISTINCT Country) FROM Customers;

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

SELECT \* FROM Customers  
WHERE Country='Mexico';

The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

The AND operator displays a record if all the conditions separated by AND is TRUE.

The OR operator displays a record if any of the conditions separated by OR is TRUE.

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

SELECT \* FROM Customers  
ORDER BY Country;

The MIN() function returns the smallest value of the selected column.The MAX() function returns the largest value of the selected column.

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

The COUNT() function returns the number of rows that matches a specified criteria.

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

The AVG() function returns the average value of a numeric column.

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

The SUM() function returns the total sum of a numeric column.

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

**Update Syntax:**

UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;

UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

**Delete Syntax:**

DELETE FROM table\_name WHERE condition;

**Syntax – INNER JOIN🡪**Returns records that have matching values in both tables

The basic syntax of the INNER JOIN is as follows.

SELECT table1.column1, table2.column2...

FROM table1

INNER JOIN table2

ON table1.common\_field = table2.common\_field;

**Syntax – LEFT JOIN🡪**Return all records from the left table, and the matched records from the right table

SELECT table1.column1, table2.column2...

FROM table1

LEFT JOIN table2

ON table1.common\_field = table2.common\_field;

**Syntax – RIGHTJOIN🡪**Return all records from the right table, and the matched records from the left table.

SELECT table1.column1, table2.column2...

FROM table1

RIGHT JOIN table2

ON table1.common\_field = table2.common\_field;

**Syntax – FULLJOIN🡪**Return all records when there is a match in either left or right table.

SELECT table1.column1, table2.column2...

FROM table1

FULL JOIN table2

ON table1.common\_field = table2.common\_field;

**Syntax – SELFJOIN🡪**is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement

SELECT a.column\_name, b.column\_name...

FROM table1 a, table1 b

WHERE a.common\_field = b.common\_field;

**Python has five standard data types?**

Numbers, String, List, Tuple and Dictionary

**How is memory managed in Python?**

1. Python memory is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have an access to this private heap and interpreter takes care of this Python private heap.
2. The allocation of Python heap space for Python objects is done by Python memory manager. The core API gives access to some tools for the programmer to code.
3. Python also have an inbuilt garbage collector, which recycle all the unused memory and frees the memory and makes it available to the heap space.

**What is Multithreading?**

Running several threads is similar to running several different programs concurrently, but with the following benefits −

* Multiple threads within a process share the same data space with the main thread and can therefore share information or communicate with each other more easily than if they were separate processes.
* Threads sometimes called light-weight processes and they do not require much memory overhead; they are cheaper than processes.

A thread has a beginning, an execution sequence, and a conclusion. It has an instruction pointer that keeps track of where within its context it is currently running.

* It can be pre-empted (interrupted)
* It can temporarily be put on hold (also known as sleeping) while other threads are running - this is called yielding.
* t1 = threading.Thread(target=print\_square, args=(10,))
* t2 = threading.Thread(target=print\_cube, args=(10,))
* target: the function to be executed by thread
* args: the arguments to be passed to the target function
* In above example, we created 2 threads with different target functions:
* To start a thread, we use start method of Thread class.
* t1.start()
* t2.start()
* Once the threads start, the current program (you can think of it like a main thread) also keeps on executing. In order to stop execution of current program until a thread is complete, we use join method.
* t1.join()
* t2.join()

**Responsibilities & Qualities of a Lead Automation Test Engineer**

Responsibilities & Qualities of a Lead Automation Test Engineer are:   
The Senior Test Automation Engineer will lead the effort to build new test frameworks and or extend existing frameworks. The tool/s accommodates both functional and load testing. These frameworks allow us to perform end-to-end testing in multi-component environments. He will working with the test engineers to design and develop a reusable architecture for the test automation. The Framework or Automation Architecture should consist the following:

* Prepare Reusable functions, which improve the robustness, re-usability, and maintainability of their test scripts.
* The framework should be designed in such a way that it increases and speeds up their productivity.
* The engineer also must support the framework/s, for example, supporting dev/qa with issues using the tool. The engineer will implement automation test scripts. Integration with the test management tool is also, planned.
* The Senior Test Automation Engineer must be able to take on leadership responsibilities and influence the direction of the automation effort, and its schedule and prioritization.
* The engineer will work with management, developers, and quality assurance personnel, to meet these goals.
* Additionally the Test Automation Engineer will also be involved in supporting the build master implement/improve build test processes, environments, and scripts. These build tests ensure that the code drops to quality assurance are of the highest quality.
* ADDITIONAL
* He will provide a practical approach to complex product testing, specifically in the areas of the automation of test cases for the purposes of regression testing. You will be a creative and proactive thinker and you will make use of current technologies to provide extensible automation infrastructures.
* He will review product requirements, functional and design specifications to determine and prepare automated test cases.
* He will work closely with other QC team members to automate the execution and verification of reports created by the various company products.
* He will work closely with various Dev team members to understand testing objectives and ensure that problems are resolved in a timely and efficient manner.
* He will be part of a team focusing on automation of an identified set of migration tests, checking they run correctly and working within the infrastructure. The team would focus on develop and test these automation buckets which would be executed by other teams.

**Download a image/file from web using python?**

# imported the requests library

import requests

image\_url = "https://www.python.org/static/community\_logos/python-logo-master-v3-TM.png"

# URL of the image to be downloaded is defined as image\_url

r = requests.get(image\_url) # create HTTP response object

printr.status\_code

# send a HTTP request to the server and save

# the HTTP response in a response object called r

with open("G:\Testing\sureshresume.docx",'w') as f:

# Saving received content as a png file in

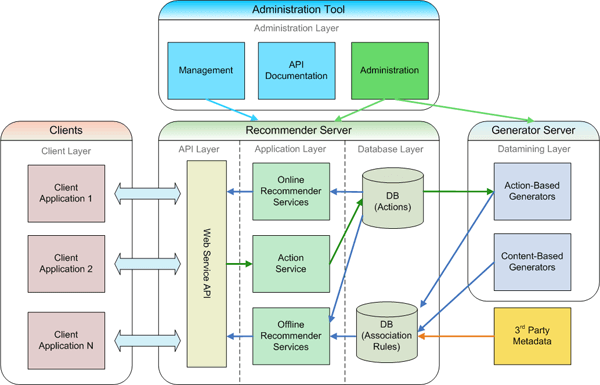
# binary format

# write the contents of the response (r.content)

# to a new file in binary mode.

f.write(r.content)

**Architecture of a web application?**



**Presentation layer:**

This is the top most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

**Application layer:**

The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.

**Database layer:**

Here information is stored and retrieved from a database or file system. The information is then passed back to the application layer for processing, and then eventually back to the user.

**How to find broken links in a webpage?**

Import requests

from selenium import webdriver

browser = webdriver.Chrome('C:\Selenium\chromedriver.exe')

browser.get('https://google.co.in/')

#links = browser.find\_elements\_by\_css\_selector("a")

links = browser.find\_elements\_by\_tag\_name("a")

print links

for link in links:

r = requests.head(link.get\_attribute('href'))

print r

print(link.get\_attribute('href'), r.status\_code)

Collect all the links in the web page based on <a> tag.

Send HTTP request for the link and read HTTP response code.

Find out whether the link is valid or broken based on HTTP response code.

Repeat this for all the links captured.

**23. What is List?**

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5];

list3 = ["a", "b", "c", "d"]

**24. What is Dictionary?**

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

Print "dict ['Name']: “dict ['Name']

Print "dict ['Age']: “dict ['Age']

**What are python generators?**

A generator calculates the values on the fly and forgets them, so it does not have any overview about the own result set.

Generators are especially useful for memory-intensive tasks, where there is no need to keep all of the elements of a memory-heavy list accessible at the same time. Calculating a series of values one-by-one can also be useful in situations where the complete result is never needed, yielding intermediate results to the caller until some requirement is satisfied and further processing stops.

## Generators

Generators are iterators, but **you can only iterate over them once**. It’s because they do not store all the values in memory, **they generate the values on the fly**:

**Yield Statement:**

To master yield, you must understand that **when you call the function, the code you have written in the function body does not run.** The function only returns the generator object, this is a bit tricky.

Then, your code will be run each time when for uses the generator.

It creates the values as you need them with a special technique called yielding

Now the hard part:

The first time the for calls the generator object created from your function, it will run the code in your function from the beginning until it hits yield, then it’ll return the first value of the loop. Then, each other call will run the loop you have written in the function one more time, and return the next value, until there is no value to return.

**Explain how to delete a file in Python?**

os.remove (filename) or os.unlink(filename)

**How do you connect MySQLDB in Python? And how do you retrieve the data and validate?**

Install Python2.7 MYSQL Connector then import MySQLdb

**# Open database connection**

#hostname #username # password #default schema name

db = MySQLdb.connect("bg-etender-db","root","root","etender" )

**# prepare a cursor object using cursor () method**

Cursor = db. Cursor ()

**# prepare SQL query to SELECT a record from the database.**

sql = "SELECT \* FROM etender.tender\_item\_value where id=%s" %tenderid

Try:

**# Execute the SQL commands**

Cursor. Execute (sql)

**# Fetch all the rows in a list of lists.**

Results = cursor.fetchall ()

**For row1 in results:**

Print row1

Return row1

#rate = row [4]

#value = row [7]

#print (rate)

# now print fetched result

#print "rate=%s, value=%s" % \

# (rate, value)

**Except Exception:**

Print "Error: unable to fetch data"

**Finally**:

# disconnect from server

Db. Close ()

**Python Features?**

#### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

#### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

#### 3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX and Macintosh etc. So, we can say that Python is a portable language.

#### 5) Free and Open Source

Python language is freely available at address. The source-code is also available. Therefore it is open source.

#### 6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

#### 7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

#### 8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

#### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

**What is docstring in Python?**

Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods. As you can see, even for a relatively simple function, documenting using comments quickly makes it unpleasant and difficult to read. So, to solve this, the docstring was introduced. A docstring is simply a multi-line string that is not assigned to anything. It is specified in source code that is used to document a specific segment of code. Unlike conventional source code comments, the docstring should describe what the function does, not how.

**List Comprehensions in python?**

List comprehensions provide a concise way to create lists.

It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The expressions can be anything, meaning you can put in all kinds of objects in lists.

The result will be a new list resulting from evaluating the expression in the

context of the for and if clauses which follow it.

The list comprehension always returns a result list.

**The basic syntax is**

[ expression for item in list if conditional ]

**Simple Example:**

x = [i for i in range(10)]

print x

**How do you split the words?**

split() – uses a regex pattern to “split” a given string into a list.

srting1 = "suresh $ kumar $x kadi"

Words = srting1.split (" ")

Print words – Result is ['Suresh', '$', 'Kumar', '$x', 'kadi']

Print words [0]–Suresh

### ****How can you generate random numbers in Python?****

Random module is the standard module that is used to generate the random number. The method is defined as

Import random

Random.randomrange(10)

**How to swap key and key value in a dictionary?**

my\_dict = {'x':1, 'y':2, 'z':3}

swap = dict((value:key) for key, value in my\_dict.iteritems())

Print swap

**What is TestCase?**

A TEST CASE is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.

**What are decorators?**

By definition, a **decorator** is a function that takes another function and extends the behavior of the latter function without explicitly modifying it. The decorator acts as a wrapper

**What are Iterators?**

1. **\_\_iter\_\_** method that is called on initialization of an iterator. This should return an object that has a next or \_\_next\_\_ (in Python 3) method.
2. **next ( \_\_next\_\_ in Python 3)** The iterator next method should return the next value for the iterable. When an iterator is used with a ‘for in’ loop, the for loop implicitly calls next() on the iterator object. This method should raise a StopIteration to signal the end of the iteration

**JSON is Unlike XML Because**

* JSON doesn't use end tag
* JSON is shorter
* JSON is quicker to read and write
* JSON can use arrays

The biggest difference is:

 XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

**Why JSON is better than XML**

XML is much more difficult to parse than JSON.  
JSON is parsed into a ready-to-use JavaScript object.

**Git** is a revision control system, a tool to manage your source code history.

**GitHub** is a hosting service for **Git** repositories.

So they are not the same thing: **Git** is the tool, **GitHub** is the service for projects that use **Git**

**F9 – Debug**

**F4 – Run to Cursor**

**F7 – Step Into – goes to detailed functions**

**F8 – Step Over – will go to next line**

**Shift + F8 – Step out**

**Ctrl+Alt+F9 – Abort debugging**

* When Python interpreter reads a source file, it will execute all the code found in it.
* When Python runs the “source file” as the main program, it sets the special variable (\_\_name\_\_) to have a value (“\_\_main\_\_”).
* When you execute the main function, it will then read the “if” statement and checks whether \_\_name\_\_ does equal to \_\_main\_\_.
* In Python **“if\_\_name\_\_== “\_\_main\_\_”** allows you to run the Python files either as **reusable modules or standalone programs**.

**Typical XML structure:**

<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>

filepath = ET.parse('E:\Demoproject\Data\Data.xml')

printfilepath

root = filepath.getroot()

print root

for root1 in root.findall('automation'):

print root1

value = root1.find('url').text

print value

**25.Lampda function (Anonymous function)?**

Python, anonymous function is a function that is defined without a name.

While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.

Hence, anonymous functions are also called lambda functions.

Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

lambda arguments: expression

# Program to show the use of lambda functions

Double

Double = lambda x: x \* 2

Print (double (5))

In the above program, lambda x: x \* 2 is the lambda function. Here x is the argument and x \* 2 is the expression that gets evaluated and returned.This function has no name.

It returns a function object which is assigned to the identifier double.

We can now call it as a normal function. The statement

double = lambda x: x \* 2

lambda operator can have any number of arguments, but it can have only one expression. It cannot contain any statements and it returns a functi/

on object which can be assigned to any variable.

add = lambda x,y : x+y

print add(1,2)

In **lambda x, y: x + y;**x and y are arguments to the function and x + y is the expression which gets executed and its values is returned as output.

**lambda x, y: x + y**returns a function object which can be assigned to any variable, in this case function object is assigned to the add variable.

add = lambda x,y : x+y

print add(1,2)

Use lambda functions when an anonymous function is required for a short period of time.

**Pylint uses?**

Exactly one space required after comma (1, 2, and 3) ->bad-whitespace

Unnecessary semicolon (unnecessary-semicolon)

Invalid constant name

Unused import unit test

Bad indentation. Found 8 spaces, expected 4 (bad-indentation)

**Does Selenium supports drag and drop?**

Yes,it supports drag and drop. We have two functions available here

       actions.drag\_and\_drop\_by\_offset(stagesreorder[2],500,308)

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

**Common Exceptions in Selenium WebDriver**

There is a complete list of ***Exceptions in Selenium WebDriver*** mentioned in the Selenium Doc which you may or may not encounter in course of your testing. Hence in this article we will focus on some *most common exceptions in Selenium WebDriver*,

1. **ElementNotVisibleException**: Although an element is present in the DOM, it is not visible (cannot be interacted with). E.g. Hidden Elements – defined in HTML using type=”hidden”.
2. **ElementNotSelectableException**: Although an element is present in the DOM, it may be disabled (cannot be clicked/selected).
3. **InvalidSelectorException**: Selector used to find an element does not return a WebElement. Say XPath expression is used which is either syntactically invalid or does not select WebElement.
4. **NoSuchElementException**: WebDriver is unable to identify the elements during run time, i.e. FindBy method can’t find the element.
5. **NoSuchFrameException**: WebDriver is switching to an invalid frame, which is not available.
6. **NoAlertPresentException**: WebDriver is switching to an invalid alert, which is not available.
7. **NoSuchWindowException**: WebDriver is switching to an invalid window, which is not available.
8. **StaleElementReferenceException**: The referenced element is no longer present on the DOM page (reference to an element is now Stale). E.g. The Element belongs to a different frame than the current one OR the user has navigated away to another page.
9. **SessionNotFoundException**: The WebDriver is performing the action immediately after ‘quitting’ the browser.
10. **TimeoutException**: The command did not complete in enough time. E.g. the element didn’t display in the specified time. Encountered when working with waits.
11. **WebDriverException**: The WebDriver is performing the action immediately after ‘closing’ the browser.

**What is Exception Handling?**

Exception handling refers to the anticipation, detection, and resolution of exception, i.e. the block of code that processes the exception object.

How to Handle Exceptions in Selenium WebDriver

**Try/Catch:** A method catches an exception using a combination of the try and catch keywords. Try is the start of the block and Catch is at the end of try block to handle the exceptions. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following

1.What is inbuilt page object model in selenium web driver?

2.What is Page Factory?

3.How to initialize page objects?

4.Method or Constructor for page object initialization?

5.Advantages of page factory?

6.Lazy initialization?

7.How to handle stale element exception?

Jenkins and GitHub related Interview questions

**1. How do you maintain your Automation code?**

We do develop our automation scripts in our local machine, oncedeployed will committhe changes to Gitrepository. Incase, more than one person working on the same project.

If Gitrepository has latest code but not you’re local one, at the time of Check-In the code tool will

Ask you to 'pull with merge’. When you do this operation, whatever changes you have done

Will be pushed to Gitrepository and at the same time latest code from Git will be Check-Out

To your local repository.

2.How to you 'merge' the code to different repositories?

GitHub has provision to merge the code to different repositories.We have an option availble

in GitHub called 'Merge Requests'.Cliking this option,will ask you to select 'Source Brnach'

and 'Target branch' and then click on 'Compare branches and continue' button.Another window will

open then Submit new merge request.Then another window will open,click on 'merge' button to

merge the code from one repositoty to another one.

3.How do you create nodes in Jenkins?

Jenkins has provision to create 'n' number of nodes to run automation scripts in different machines.

Go to 'Manage jenkins' there you can see 'Manage Nodes' provision to create nodes.when creating a node,

most important one to select Launch method as 'Launch agent via Java Web Start'(For Windows) and save the details.

Saving the details will generate 'slave-agent.jnlp'.Run the this file in any machine, where you wish to run

automation scripts.

4.How do you Run automation scripts in different machines?

Jenkins has an option 'Restrict where this project can be run' enabling this will ask you toprovide Label expression. You have to provide 'Node name' (you should be knowing how to create nodes)as the label expression and save the details. When you run Jenkins job, it automatically goes to node machineand there it will run our automations scripts.

Note:By default, if you run any Jenkins job it looks for 'Windows batch command'for example :( python C:\eTenderPreStaging\TestSuite\SmoketestSuite-Prestaging.py) local to Jenkins server.

5. How do you schedule your jobs?

Jenkins has an option 'Poll SCM' enabling this will ask you to provide Schedule. (Fore.g. H/5 \* \* \* \*)

In the example, what it does -->it checks the git repository for every 5 minutes, in case if there are

Any changes it automatically Check-out the latest code and run the automation.

(It’s purely depends on how you schedule, you can schedule for hourly basis and different options available in Jenkins)

**Headless browser Testing:**

1. Firefox (56+ version)

2. Chrome (59 + version)