

ROBOT FRAMEWORK

Building Automation Testing Using
Robot Framework



PRE-REQUISITES

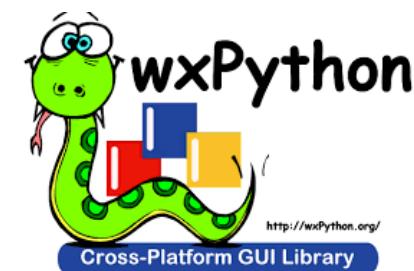
You need to complete these before



python



Robot Framework



SeleniumLibrary

ROBOT FRAMEWORK

Introduction robot
framework



Introduction Robot Framework



is a Python-based



extensible keyword-driven automation framework for acceptance testing



acceptance test driven development (ATDD),



behavior driven development (BDD)



robotic process automation (RPA)

Introduction Robot Framework



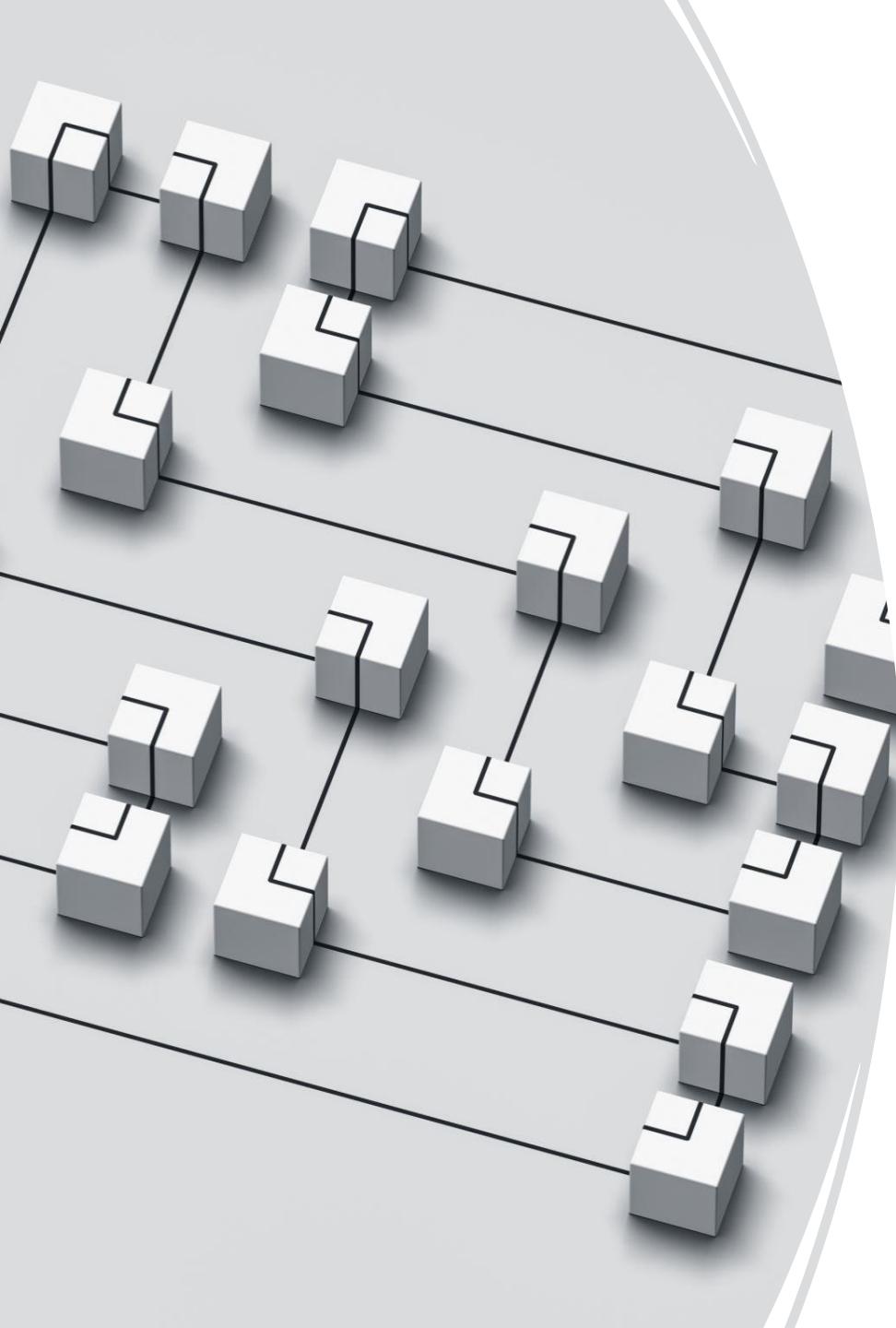
free to use without licensing costs



easy syntax, utilizing human-readable keywords



Its capabilities can be extended by libraries implemented with Python, Java or many other programming languages

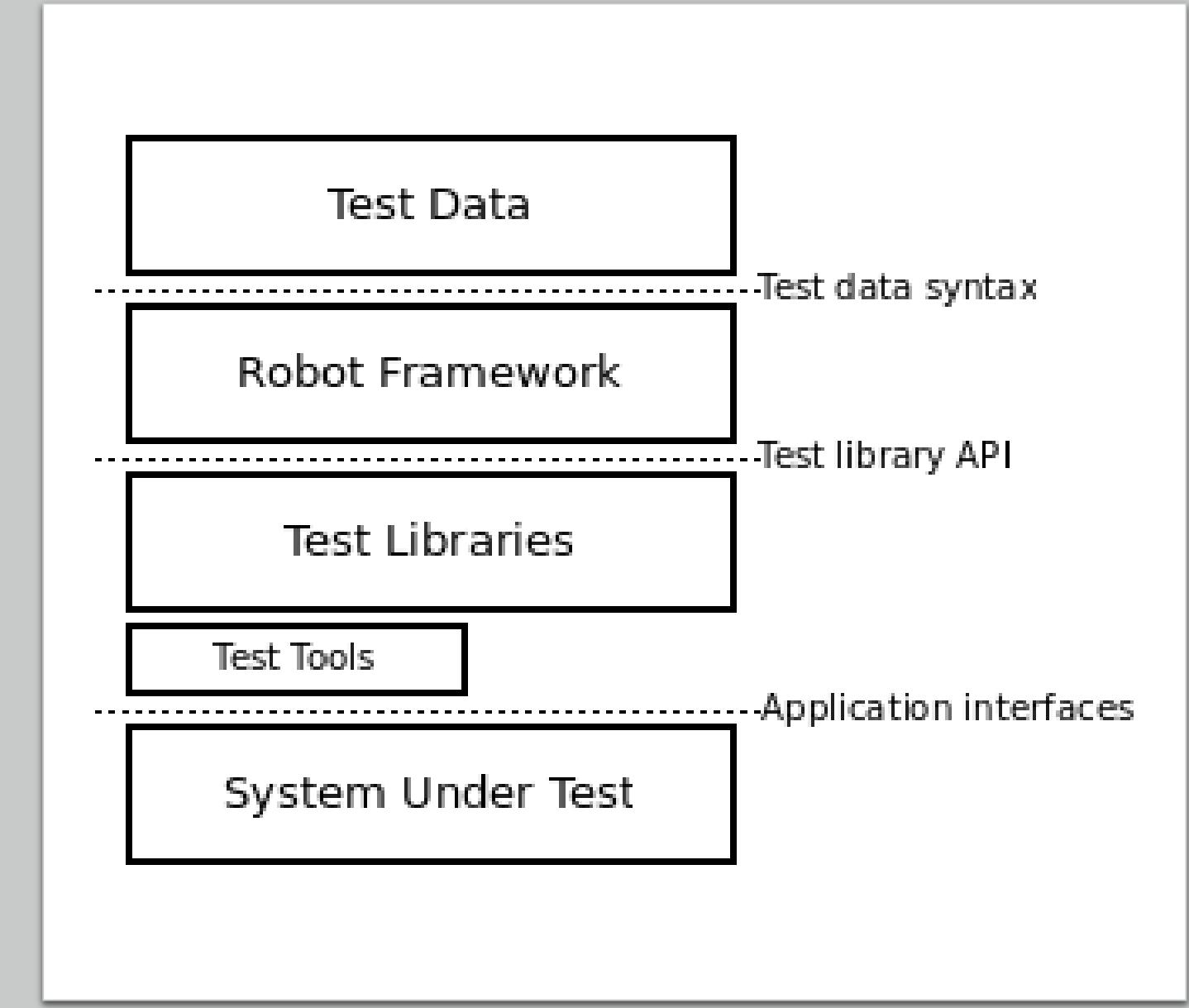


Why Robot Framework

- Enables easy-to-use tabular syntax for creating test cases in a uniform way.
- Provides ability to create reusable higher-level keywords from the existing keywords.
- Provides easy-to-read result reports and logs in HTML format.
- Is platform and application independent.
- Provides a simple library API for creating customized test libraries which can be implemented natively with Python.
- Supports creating data-driven test cases.
- Has built-in support for variables, practical particularly for testing in different environments.

High-level architecture

- Robot Framework is a generic, application and technology independent framework
- It has a highly modular architecture illustrated in the diagram



Features of Robot Framework

- **Keywords**

Robot framework comes with built-in keywords available with robot framework, keywords available from the libraries like Selenium Library (open browser, close browser, maximize browser, etc.). We can also create user-defined keywords, which are a combination of other user-defined keywords or built-in or library keywords. We can also pass arguments to those keywords, which make the user-defined keywords like functions that can be reused.

- **Variables**

Robot framework supports variables – scalar, list and dict. Variables in robot framework are easy to use and are of great help while writing complex test cases.

Features of Robot Framework

- Libraries

Robot framework has support for a lot of external libraries like Selenium Library, Database Library, FTP Library and http library. Selenium Library is mostly used as it helps to interact with the browsers and helps with web application and UI testing. Robot framework also has its own built-in libraries for strings, date, numbers etc.

- Resources

Robot framework also allows the import of robot files with keywords externally to be used with test cases. Resources are very easy to use and are of great help when we need to use some keywords already written for other test projects.

- Data driven test cases

Robot framework supports keyword driven style test cases and data driven style. Data driven works with high-level keyword used as a template to the test suite and the test cases are used to share data with the high-level keyword defined in the template. It makes the work very easy for testing UI with different inputs.

Features of Robot Framework

- Test Case Tagging

Robot framework allows to tag test-cases so that we can either run the tagged test-cases or skip the tagged testcases. Tagging helps when we want to run only a group of test cases or skip them.

- Reports and Logs

Robot framework provides all the details of test suite, test case execution in the form of report and logs. All the execution details of the test case are available in the log file. The details like whether the test case has failed or passed, time taken for execution, steps followed to run the test case are provided.

Limitation robot framework for automation testing

- Robot framework does not support parallel execution
- Hard to customize html report
- Robot framework is hard to maintain
- Some error are difficult to debug
- Robot framework has strict indentation rules

Installing Robot Framework

- Robot Framework is implemented with Python, so you need to have Python installed.
- On Windows machines, make sure to add Python to PATH during installation.



Install python



Version	Operating System
Gzipped source tarball	Source releases
XZ compressed source tarball	Source releases
macOS 64-bit installer	macOS
Windows help file	Windows
Windows x86-64 embeddable zip file	Windows
Windows x86-64 executable installer	Windows
Windows x86-64 web-based installer	Windows
Windows x86 embeddable zip file	Windows
Windows x86 executable installer	Windows
Windows x86 web-based installer	Windows

Check python use command prompt

Open cmd



Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19043.1706]
```

```
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>python --version
```

```
python 3.7.8
```

```
C:\Windows\system32>
```

Python --version

Install robot framework



The image shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window title bar includes the text "Administrator: Command Prompt", the operating system version "Microsoft Windows [Version 10.0.19043.1706]", and copyright information "(c) Microsoft Corporation. All rights reserved.". The main area of the window displays the following command-line session:
C:\Windows\system32>python --version
Python 3.7.8
C:\Windows\system32>pip install robotframework_

Type pip install robotframework

Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19043.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>python --version
Python 3.7.8

C:\Windows\system32>pip install robotframework
Collecting robotframework
  Using cached robotframework-5.0.1-py3-none-any.whl (639 kB)
Installing collected packages: robotframework
Successfully installed robotframework-5.0.1
WARNING: You are using pip version 20.1.1; however, version 22.1.1 is available.
You should consider upgrading via the 'c:\program files\python37\python.exe -m pip install --upgrade pip' command.

C:\Windows\system32>pip install wxpython==4.0.7
Collecting wxpython==4.0.7
  Downloading wxPython-4.0.7-cp37-cp37m-win_amd64.whl (23.0 MB)
    |██████████| 23.0 MB 6.4 MB/s
Collecting pillow
  Using cached Pillow-9.1.1-cp37-cp37m-win_amd64.whl (3.3 MB)
Collecting numpy; python_version >= "3.0"
  Using cached numpy-1.21.6-cp37-cp37m-win_amd64.whl (14.0 MB)
Collecting six
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pillow, numpy, six, wxpython
Successfully installed numpy-1.21.6 pillow-9.1.1 six-1.16.0 wxpython-4.0.7
WARNING: You are using pip version 20.1.1; however, version 22.1.1 is available.
You should consider upgrading via the 'c:\program files\python37\python.exe -m pip install --upgrade pip' command.

C:\Windows\system32>
```

Install wxpython

```
You should consider upgrading via the "C:\Program Files\Python37\python.exe -m pip install --upgrade pip" command.

rst C:\Windows\system32>pip install wxpython==4.0.7
Collecting wxpython==4.0.7
  Downloading wxPython-4.0.7-cp37-cp37m-win_amd64.whl (23.0 MB)
    |████████████████████████████████| 23.0 MB 6.4 MB/s
Collecting pillow
  Using cached Pillow-9.1.1-cp37-cp37m-win_amd64.whl (3.3 MB)
Collecting numpy; python_version >= "3.0"
  Using cached numpy-1.21.6-cp37-cp37m-win_amd64.whl (14.0 MB)
Collecting six
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pillow, numpy, six, wxpython
Successfully installed numpy-1.21.6 pillow-9.1.1 six-1.16.0 wxpython-4.0.7
WARNING: You are using pip version 20.1.1; however, version 22.1.1 is available.
You should consider upgrading via the "C:\Program Files\Python37\python.exe -m pip install --upgrade pip" command.
```

Type `pip install wxpython==4.0.7`

Install ride

```
C:\Windows\system32>pip install robotframework-ride
Processing c:\users\syazw\appdata\local\pip\cache\wheels\fc\7e\ad\62316f036476f8a3eba3830dcb12649d85d209e5a42416c348\robotframework_ride-1.7.4.2-py3-none-any.whl
Collecting PyPubSub
  Using cached Pypubsub-4.0.3-py3-none-any.whl (61 kB)
Collecting Pygments
  Using cached Pygments-2.12.0-py3-none-any.whl (1.1 MB)
Collecting Pywin32
  Using cached pywin32-304-cp37-cp37m-win_amd64.whl (12.2 MB)
Requirement already satisfied: wxPython<=4.0.7.post2 in c:\program files\python37\lib\site-packages (from robotframework-ride) (4.0.7)
Requirement already satisfied: pillow in c:\program files\python37\lib\site-packages (from wxPython<=4.0.7.post2->robotframework-ride) (9.1.1)
Requirement already satisfied: six in c:\program files\python37\lib\site-packages (from wxPython<=4.0.7.post2->robotframework-ride) (1.16.0)
Requirement already satisfied: numpy; python_version >= "3.0" in c:\program files\python37\lib\site-packages (from wxPython<=4.0.7.post2->robotframework-ride) (1.21.6)
Installing collected packages: PyPubSub, Pygments, Pywin32, robotframework-ride
```

Type pip install robotframework-ride

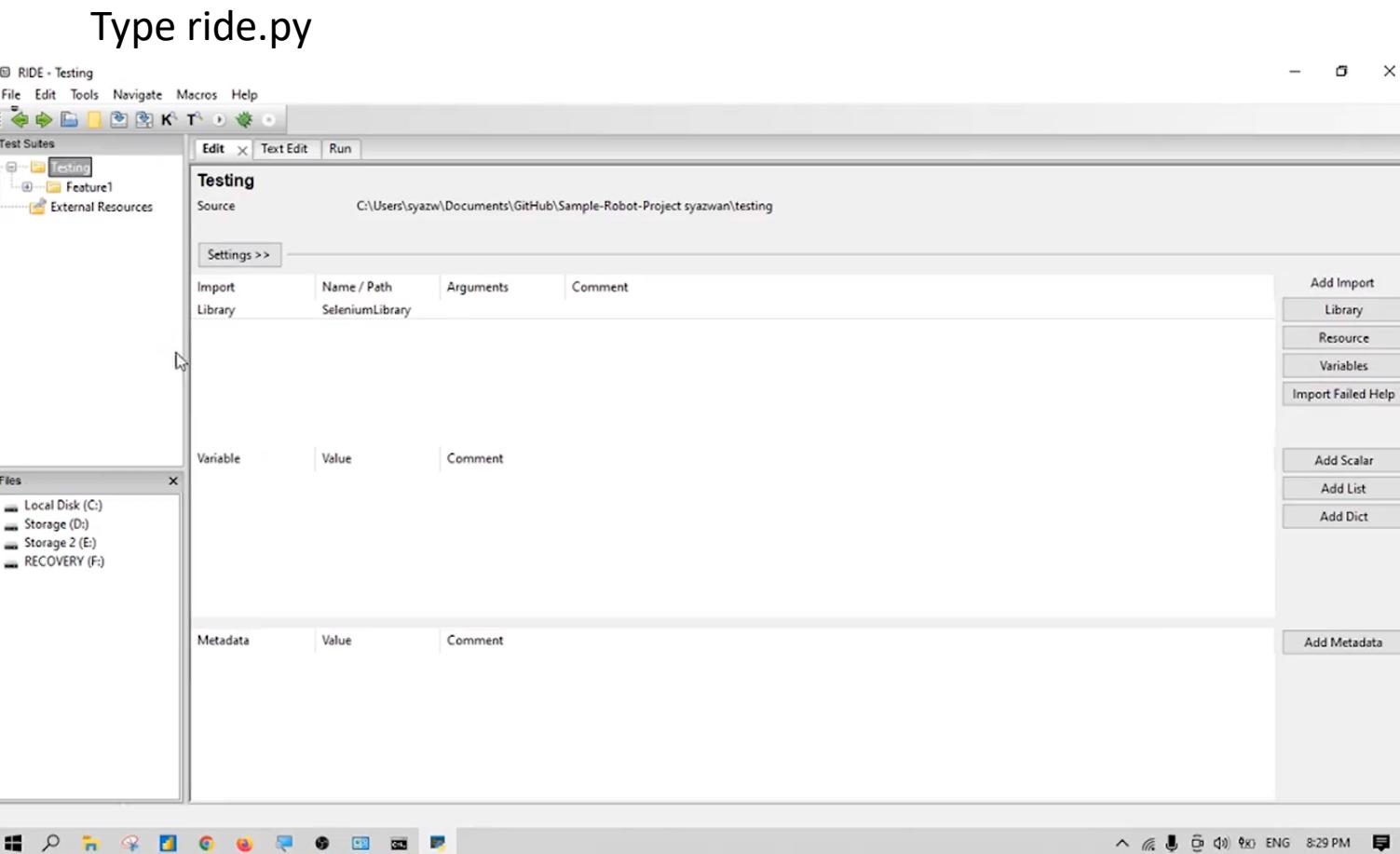
Install Selenium Library

```
C:\Windows\system32>pip install --upgrade robotframework-seleniumlibrary
Collecting robotframework-seleniumlibrary
  Using cached robotframework_seleniumlibrary-6.0.0-py2.py3-none-any.whl (95 kB)
Collecting selenium>=4.0.0
  Using cached selenium-4.1.5-py3-none-any.whl (979 kB)
Requirement already satisfied, skipping upgrade: robotframework>=3.2.2 in c:\program files\python37\lib\site-packages (from robotframework-seleniumlibrary) (5.0.1)
Collecting robotframework-pythonlibcore>=2.2.1
  Using cached robotframework_pythonlibcore-3.0.0-py2.py3-none-any.whl (9.9 kB)
Collecting urllib3[secure,socks]~=1.26
  Using cached urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
```

Type pip install –upgrade robotframework-seleniumlibrary

Open ride robot framework

```
You should consider upgrading via the C:\Program Files\Python37\python.exe -m pip install --upgrade pip command.  
C:\Windows\system32>ride.py
```



Test data sections

Different sections in data	
Section	Used for
Settings	1) Importing test libraries resource files and variable files 2) Defining metadata for test suites and test cases
Variables	Defining variables that can be used elsewhere in the test data.
Test Cases	Creating test cases from available keywords.
Tasks	Creating tasks using available keywords. Single file can only contain either tests or tasks.
Keywords	Creating user keywords from existing lower-level keywords
Comments	Additional comments or data. Ignored by Robot Framework.

Keywords in Selenium Library

List of keywords that used in this project

Keywords	Deceptions
Open browser	Opens a given browser instance to the given url address
Close Browser	Closes browser window/tab
Click Element	Click the element identified by locator(element)
Input Text	Types the given text into the text field identified by locator
Select From List by Label	Selects options from selection list locator by given value
Choose File	Inputs the file_path into the file input field locator.
Element Text Should Be	Verifies that element locator contains exact the text expected

Syntax of Keywords in Table in Edit tab.

1st Column	2nd column	3rd column
Open browser	[link of website]	[webdriver] : chrome
Close Browser		
Click Element	Path of element (xpath,html, etc)	
Input Text	Path of element	Test data
Select From List by Label	Path of element	Value
Choose File	Path of element	Directory of test data file
Element Text Should Be	Path of element	text that should be

- For Choose File, recommend to save test data file into folder that save test suite. Then on third column, type '\${CURDIR}/file name'

If want to find more library keywords, can refer to:

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html#library-documentation-top>

Copy Element (Xpath) in webpage

Steps:

- i. Choose element that want to get code
- ii. Right click and then click on Inspect
- iii. Inspect coding tab is shown on right (or left/below) and it will highlight code that represent

opensource-demo.orangehrmlive.com/index.php/auth/login

The screenshot shows the OrangeHRM login page. A context menu is open over the password input field, displaying options like 'Emoji', 'Win+Period', 'Undo' (Ctrl+Z), 'Redo' (Ctrl+Shift+Z), 'Cut' (Ctrl+X), 'Copy' (Ctrl+C), 'Paste' (Ctrl+V), 'Paste as plain text' (Ctrl+Shift+V), 'Select all' (Ctrl+A), 'Spell check', 'Writing Direction', 'Block element...', and 'Inspect'. The 'Inspect' option is highlighted. To the right, the browser's developer tools are open, showing the 'Elements' tab with the DOM structure and the 'Styles' tab with the applied CSS rules.

Elements Console Sources

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>...</head>
  ...<body data-new-gr-c-s-check-loaded="14.1062.0" data-gr-ext-installed> == $0
    <div id="wrapper">...</div>
    <!-- wrapper -->
    <script type="text/javascript">...</script>
    <div id="tiptip_holder" style="max-width:200px;">...</div>
  </body>
  <grammarly-desktop-integration data-grammarly-shadow-root="true">...</grammarly-desktop-integration>
</html>
```

html body

Styles Computed Layout Event Listeners DOM Breakpoints

```
element.style { }
body {
  background-color: #FFFFFF;
  height: 700px;
  font-family: Roboto, Arial, Helvetica, sans-serif;
}
body {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 13px;
  color: #5d5d5d;
}
body {
  line-height: 1;
}
```

iv. Right click on the highlight code and click Copy -> Copy XPath

The screenshot shows the OrangeHRM login page and the Chrome developer tools Elements panel.

OrangeHRM Login Page:

- The page title is "LOGIN Panel".
- The logo on the left features an orange fruit with the text "HR for ALL" around it.
- The login form has two input fields: one for "Username" and one for "Password".
- A "LOGIN" button is at the bottom.
- A link "Forgot your password?" is also present.
- A status message at the top right says "(Username : Admin | Password : admin)".

Developer Tools Elements Panel:

- The "Elements" tab is selected.
- The DOM tree shows the HTML structure of the login page.
- An input field with the ID "txtUsername" is highlighted.
- A context menu is open over the highlighted element, showing options like "Copy", "Copy element", "Copy XPath", and "Copy full XPath".
- The "Copy XPath" option is highlighted.

Open Directory Project in RIDE

1. Open RIDE
2. Click 'File' on top left. Then choose Open Directory.
3. Then choose project folder that consist testing script

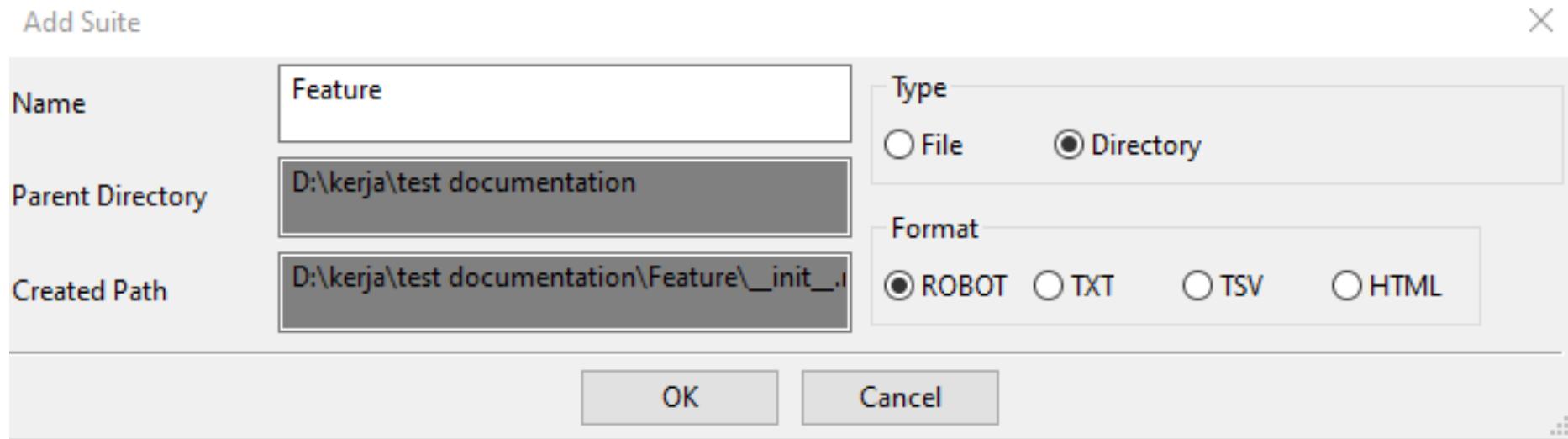
For this tutorial choose 'Robotframework Testing from documentation' folder inside 'Robotframework_ProjectAssignment_Group2' GitHub repository.

First testing

Success Adding Candidate

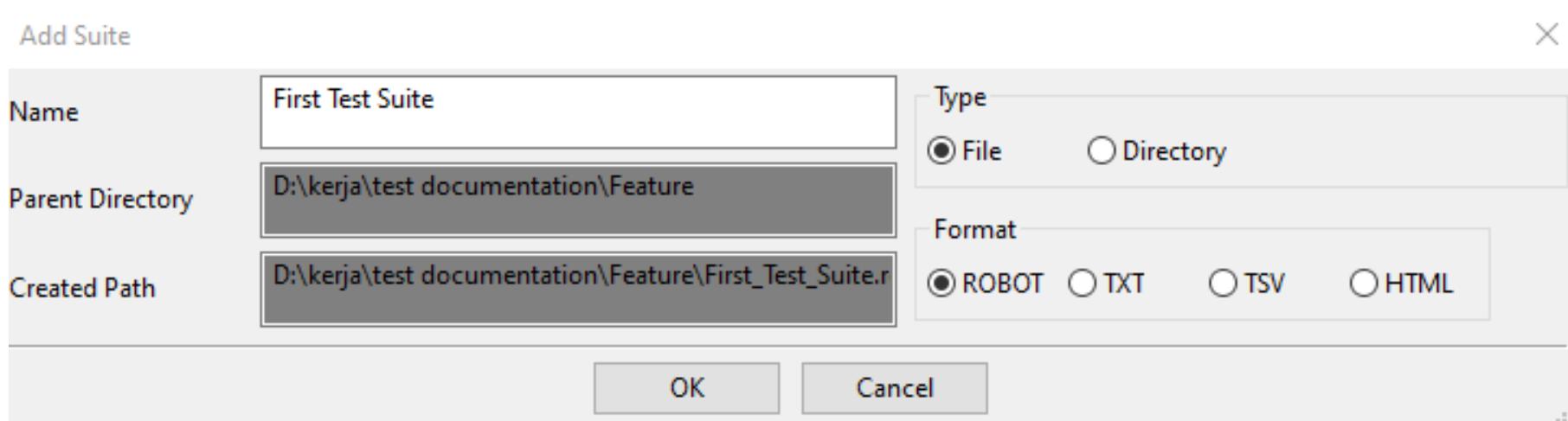
Step 1: Create Directory/folder

- i. Create new directory/folder (optional) for saving step definitions:
Right click on main directory (on left side of RIDE) and choose ‘New Suite’.
- ii. In Add Suite window, change Type from ‘File’ into ‘Directory’. Then click ‘OK’. ‘Feature’ directory can be found under main directory



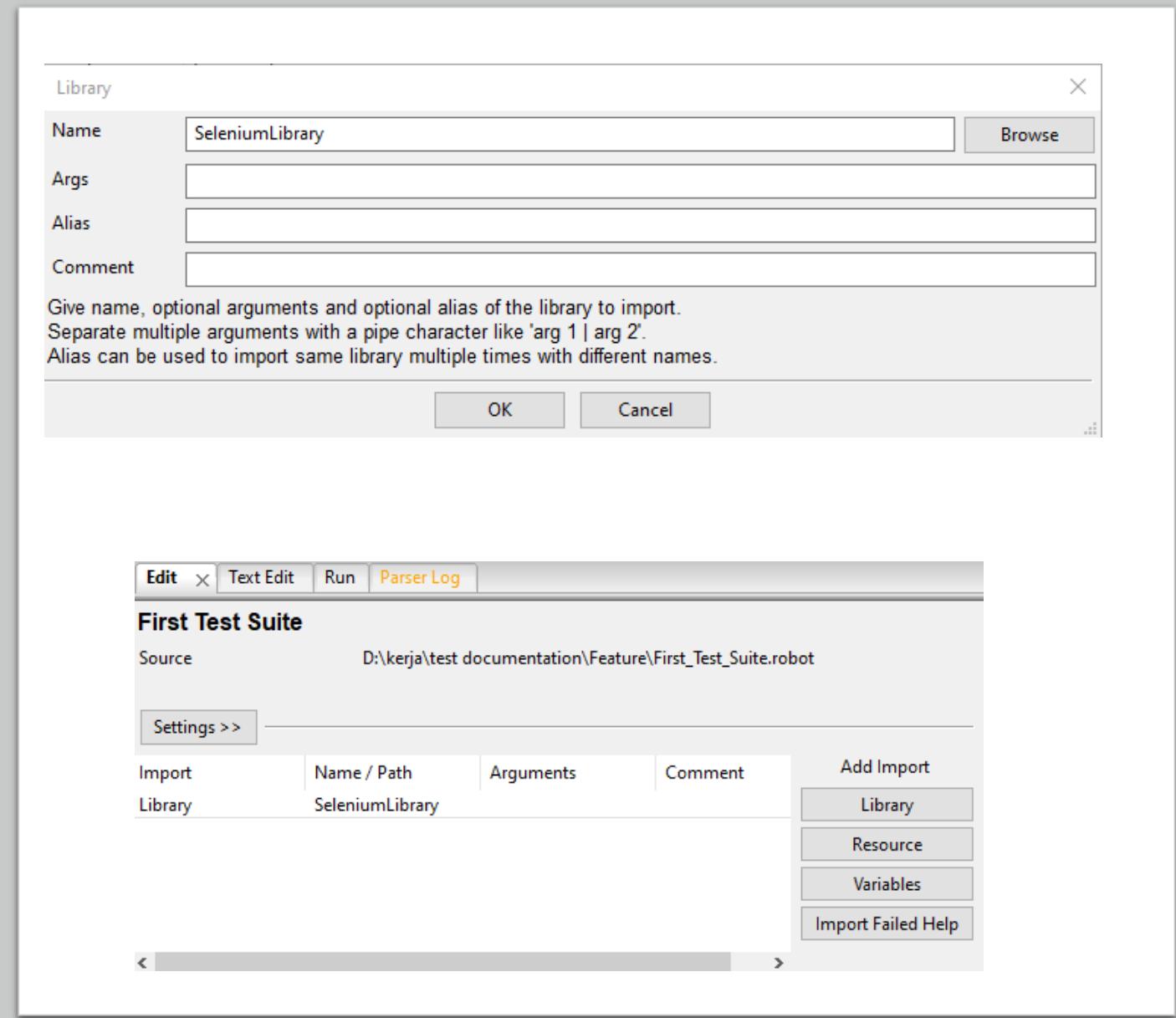
Step 2: Create Test Suite

- i. Right click on Feature Directory and choose ‘New Suite’.
- ii. In Add Suite window, named it as ‘First Test Suite’ and then click OK. This test suite can be found under Feature.



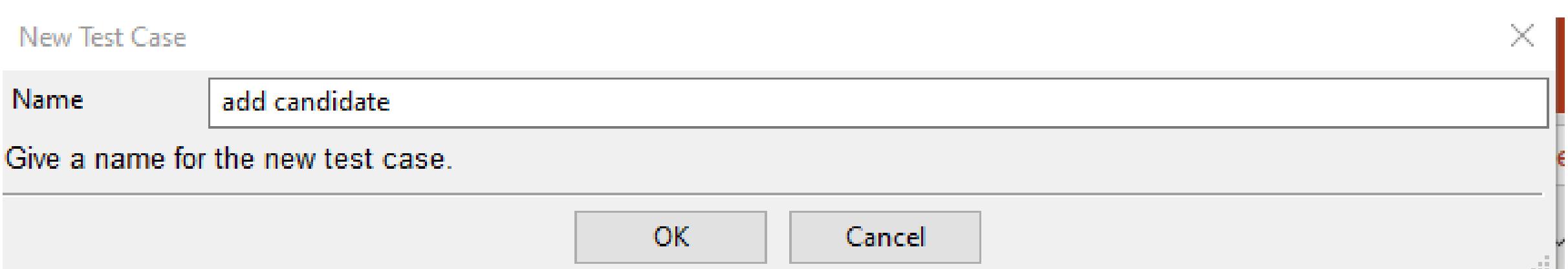
Step 3: Import Selenium Library

- i. Click on library under 'import' and import Selenium Library by typing: 'SeleniumLibrary'. Then click OK.
- ii. On Edit Section (2nd picture), if colour of SeleniumLibrary is black, it means this library is valid. If colour is red, it means this library is invalid



Step 4: Create Test Case

- i. Right click on main directory (on left side of RIDE) and choose ‘New Test Case’.
- ii. In Add Suite window, named it as ‘add candidate’ and then click OK. This testcase can be found under Test Suite.



Step 5: Put test steps into Test Case

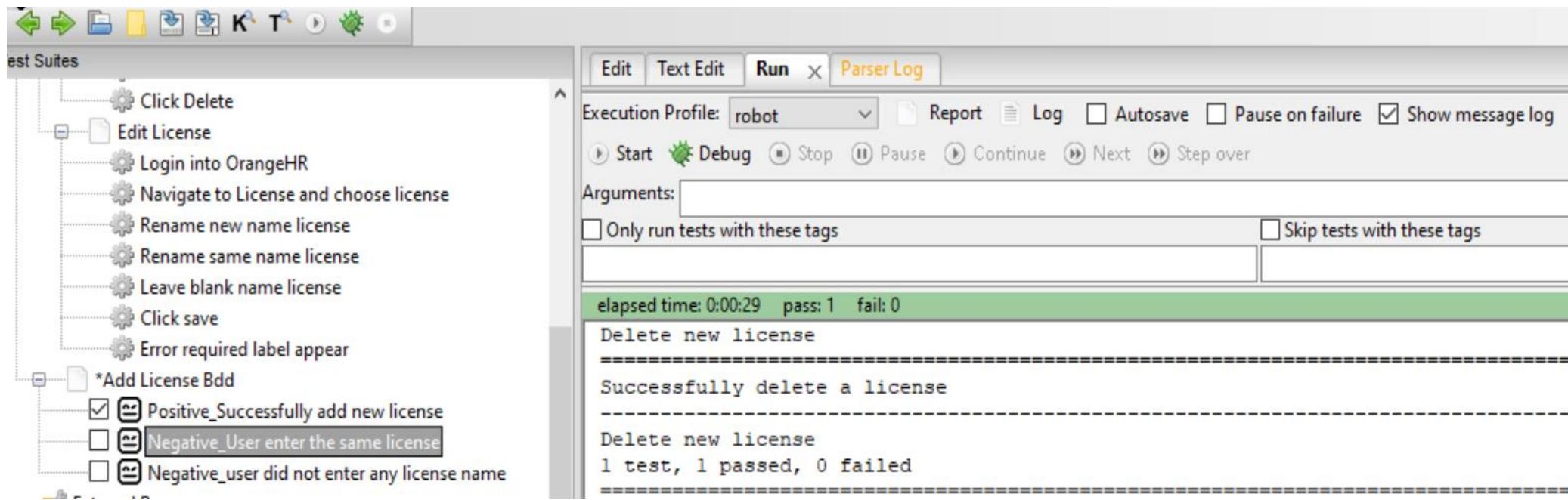
- i. Click on test case and click Edit tab.
- ii. In First column, put library keywords based on steps that have planned for this testing
- iii. Copy element (Xpath) of object inside webpage. Then put elements into second column
- iv. Put value (only for steps that involve test data) into third column

The screenshot shows a software interface for editing a test case titled "add candidate". The window has tabs at the top: "Edit", "Text Edit", and "Run". Below the title, there is a "Settings >>" button. The main area is a table with three columns:

1	open browser	https://opensource-demo.orangehrmlive.com
2	input_text	xpath=//*[@id="txtUsername"]
3	Sleep	2
4	input_text	xpath=//*[@id="txtPassword"]
5	Sleep	3
6	click element	xpath=//*[@id="btnLogin"]
7	Sleep	3
8	click element	xpath=//*[@id="menu_recru"]
9	Sleep	1
10	click element	xpath=//*[@id="btnAdd"]
11	Sleep	1
12	input_text	xpath=//*[@id="addCandidateName"]
13	Sleep	1
14	input_text	xpath=//*[@id="addCandidateEmail"]
15	Sleep	1
16	input_text	xpath=//*[@id="addCandidateAliakhalid@kmail.com"]
17	Sleep	1
18	Select From List By Label	xpath=//*[@id="addCandidateJobTitle"]
19	Sleep	1
20	Choose File	xpath=//*[@id="addCandidateResume"]
21	click element	xpath=//*[@id="btnSave"]
22	Sleep	3

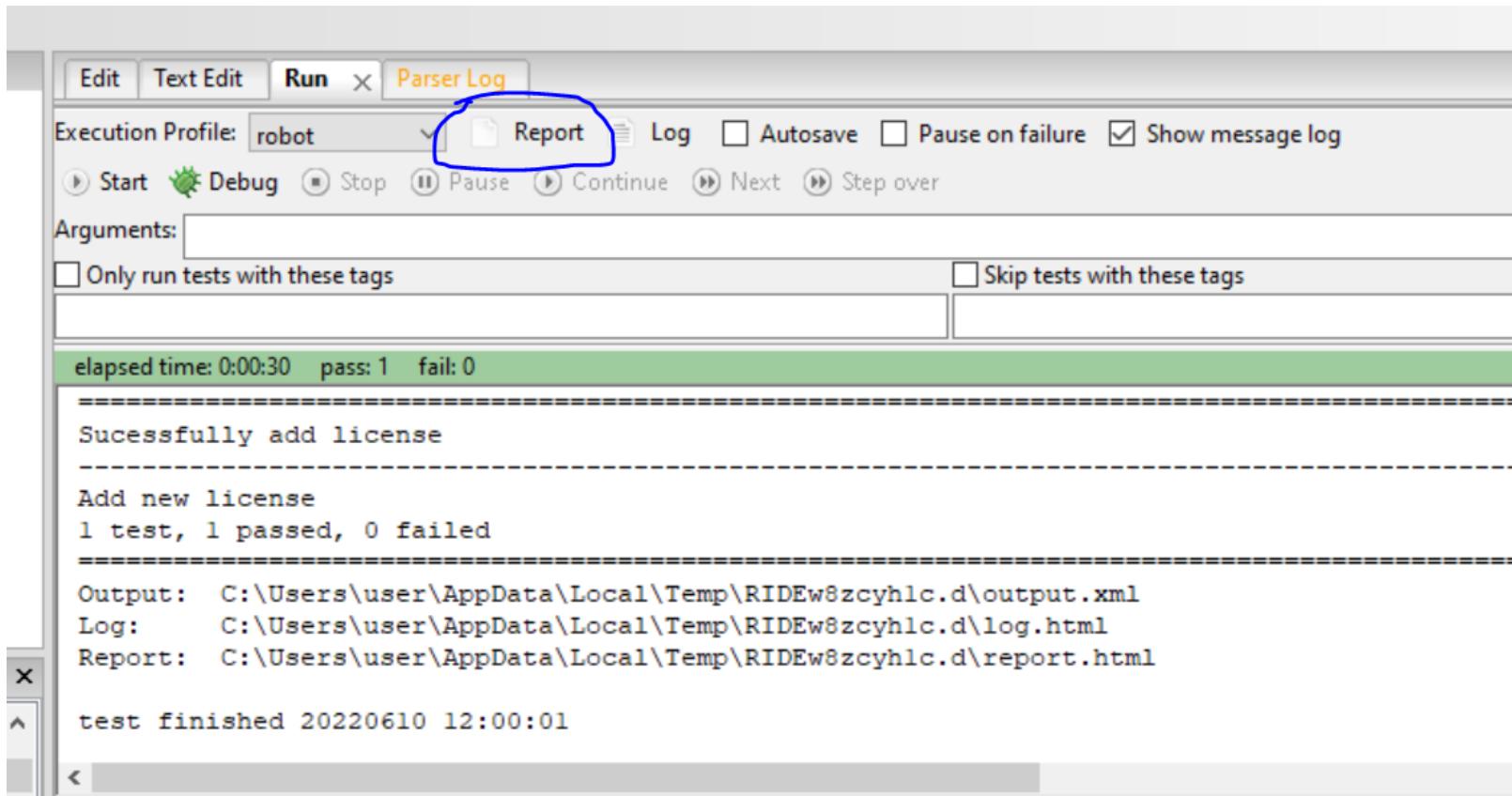
STEP 6: Run test case

- i. Tick the test case you want to run.
- ii. Go to ‘Run’ tab
- iii. Click ‘Start’ robot button.



STEP 7: Test case report

- Click report button on ‘Run’ tab to check the report.



Example of test case report

Add new license Report

Generated
20220610 12:00:01 UTC+08:00
7 minutes 31 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20220610 11:59:31.170
End Time:	20220610 12:00:01.325
Elapsed Time:	00:00:30.155
Log File:	log.html

Test Statistics

Total Statistics		Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests		1	1	0	0	00:00:30	<div style="width: 100%; background-color: #6aa84f; height: 10px;"></div>

Statistics by Tag		Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags							<div style="width: 0%; background-color: #e0e0e0; height: 10px;"></div>

Statistics by Suite		Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Add new license		1	1	0	0	00:00:30	<div style="width: 100%; background-color: #6aa84f; height: 10px;"></div>

Test Details

All Tags Suites Search

Suite:

Test:

Include:



To make testing process become proper, advance and easily, there are several features/setting that we will learn:

- Test Setup
 - Test Teardown
 - Variable
 - Keywords File
 - Resource File
- 

Test Setup and Teardown

Test Setup and Test Teardown

Test Setup: keyword/step that is executed before a test case(s)

Test Teardown: Keyword/step that is executed after a test case (s)

- Both can be set in Test Suite.
- Steps
 - i. Go to test suite in Edit tab
 - ii. Click Setting
 - iii. Find Setup and Teardown and click 'Edit'
 - iv. Put keyword/step

0 Add And Delete Candidate Nonbdd

Source D:\kerja\pakwanoranghr\Candidate Feature Test Suites\0_add_and_delete_candida

Settings <<

Documentation

Edit Clear

Suite Setup

Suite Teardown

Test Setup

Test Teardown

Test Template

Test Timeout

Force Tags

<Add New>

Default Tags

<Add New>

Edit Clear

Edit Clear

Edit Clear

Edit Clear

Edit Clear

Edit Clear

Import Library Name / Path SeleniumLibrary Arguments Comment Add Import

Resource Variables

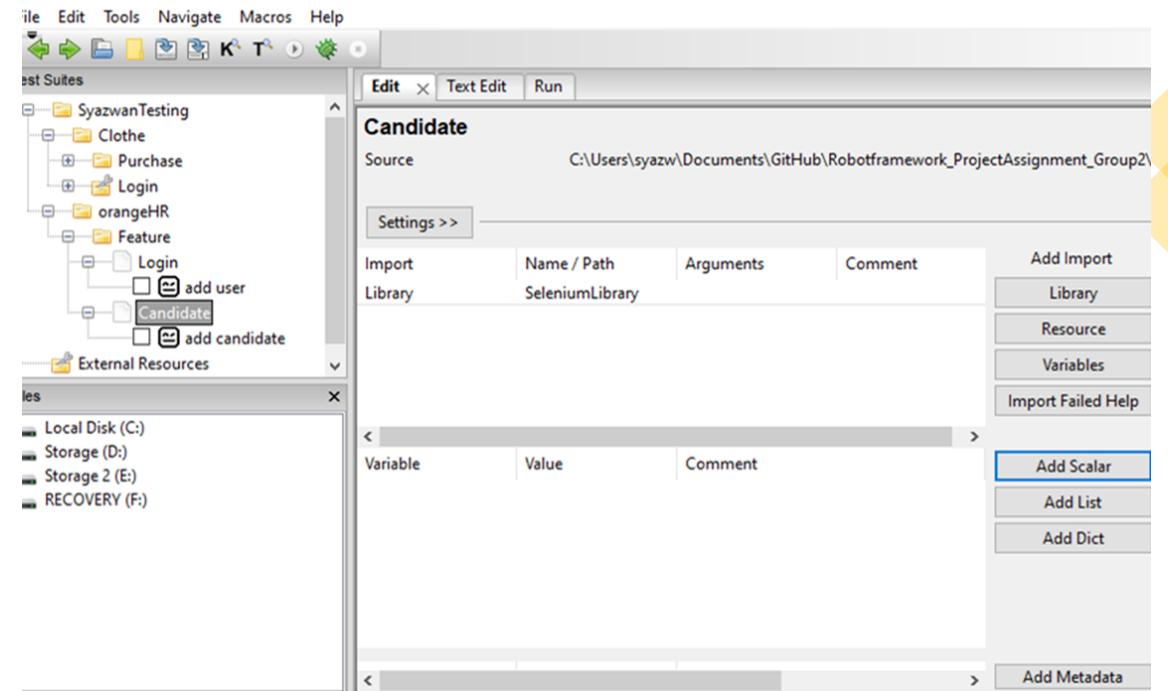
The screenshot displays a software interface for managing test configurations. At the top, it shows the source path: D:\kerja\pakwanoranghr\Candidate Feature Test Suites\0_add_and_delete_candida. Below this, there's a 'Settings <<' button. The main area contains several sections: 'Documentation' with 'Edit' and 'Clear' buttons; 'Suite Setup' and 'Suite Teardown' both with 'Edit' and 'Clear' buttons; 'Test Setup' set to 'Open Browser | Chrome' with 'Edit' and 'Clear' buttons; 'Test Teardown' set to 'Close Browser' with 'Edit' and 'Clear' buttons; 'Test Template' with 'Edit' and 'Clear' buttons; 'Test Timeout' with a red asterisk and 'Edit' and 'Clear' buttons; 'Force Tags' and 'Default Tags' each with a '' button and 'Edit' and 'Clear' buttons. At the bottom, there are tabs for 'Import' (selected), 'Library', 'Name / Path' (set to 'SeleniumLibrary'), 'Arguments', 'Comment', and an 'Add Import' section with buttons for 'Library', 'Resource', and 'Variables'.

- But only one keyword/step can be set for both.
- To have keywords/steps in setup and teardown, keywords file should be created and set to both

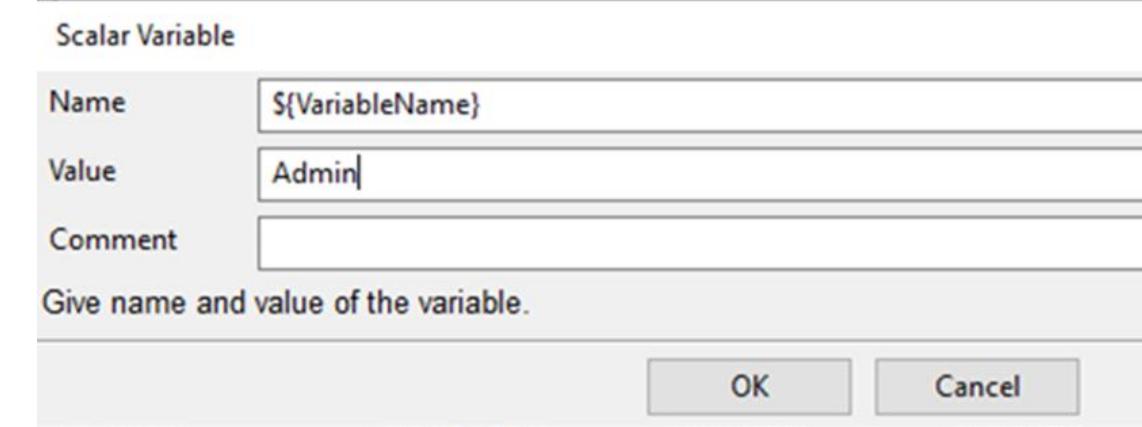
Variable, Keywords File and Resource File

Steps (using edit tab)

i. Navigate to Test Suit



ii. Click on 'Add Scalar' on right bottom (1st picture)



iii. Add Scalar window is popup, write variable name inside {} and put value that you want (2nd picture)

- iv. Now go to test case
- v. To add variable, write \${VariableName} on third column as shown below:

3	input_text	xpath=//*[@id="txtUsername"]\${VariableName}
---	------------	----------------------------------------------

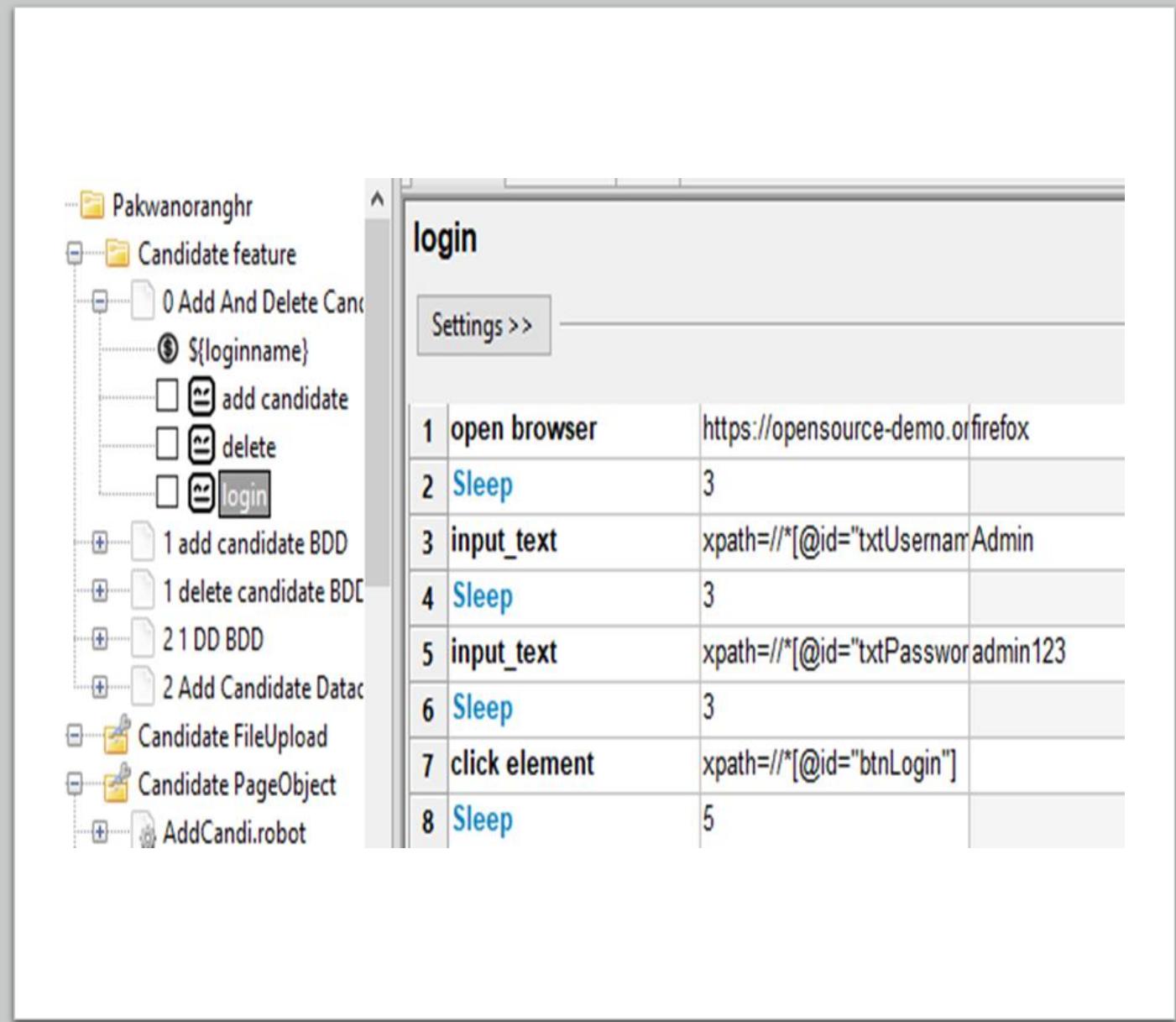
*If colour of Variable is green, it means this variable is existing. But if its colour is light purple/pink, it means this variable is not existing.

Keywords File

Combining existing keywords/test steps together

Steps:

- i. Select any test case inside test suite. Inside test case, there are many keywords/test steps.
- ii. Main objective for this is creating keywords file so it can be set on Test Setup.
- iii. Select on test suite and then right click-> New User Keywords to create keywords file

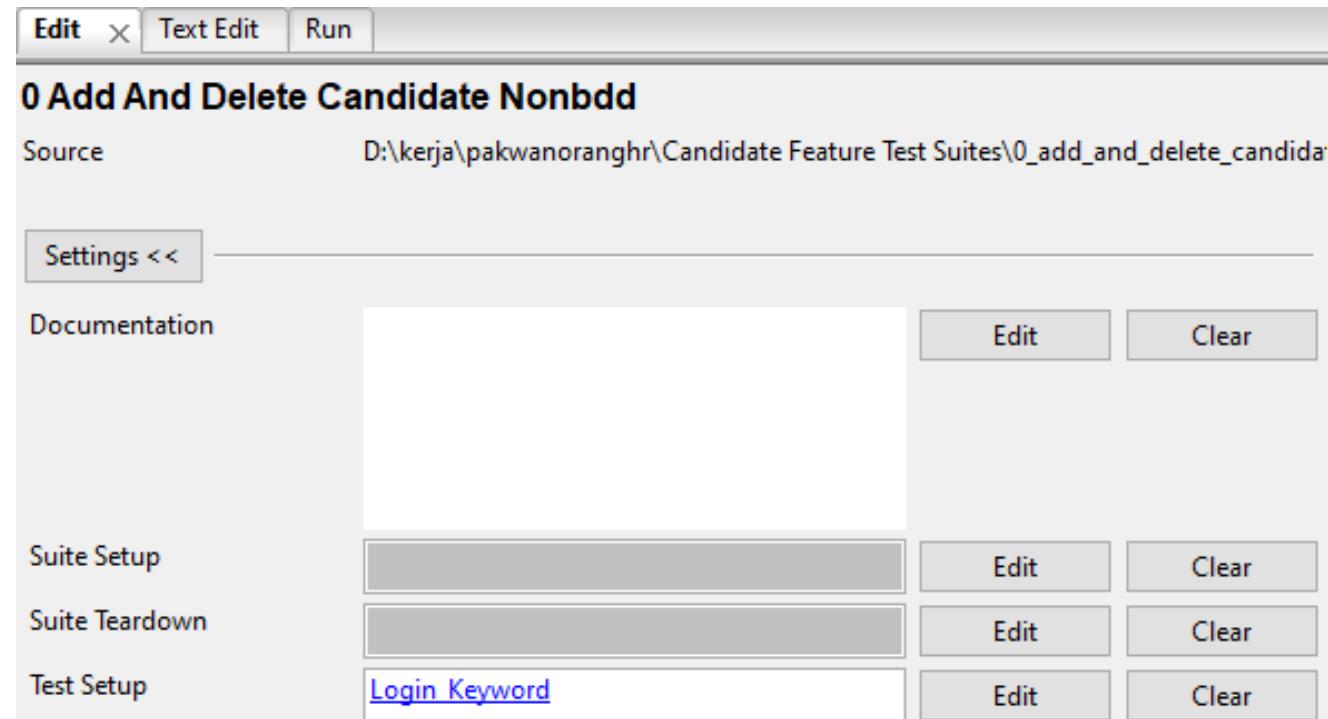


iv. Inside keywords file, copy/write keywords/test steps

The screenshot shows a software interface for test automation. On the left, a tree view displays a project structure under 'Pakwanoranghr'. The 'Candidate feature' node is expanded, showing several test cases and a 'Login_Keyword' component. The 'Login_Keyword' component is highlighted with a dark gray background. On the right, a detailed view of the 'Login_Keyword' component is shown in a table format. The table has three columns: Step Number, Keyword, and Argument/Value.

1	open browser	https://opensource-demo.orangehrmlive.com/index.php/auth/login
2	Sleep	3
3	input_text	xpath=//*[@id="txtUsername"]
4	Sleep	3
5	input_text	xpath=//*[@id="txtPassword"]
6	Sleep	3
7	click element	xpath=//*[@id="btnLogin"]
8	Sleep	5

- v. After that, go to login test cases and delete (if you want).
- vi. Go to Test Suite -> Setting. And then put keywords file name on Test Setup



- vii. This test case can be run

There is Keywords file that can link with variables.

i. Create keywords file and name it with any name include the variable

Example: Add candidate \${firstname} \${lastname}

ii. Put keywords/steps that related to those variable. After that, put variable names on third column and row that related

The screenshot shows a software interface for test automation. On the left, a 'Test Suites' tree view displays a project structure with a 'Pakwanoranghr' folder containing a 'Candidate Feature Test Suites' folder. Inside this folder are several test cases: '0 Add And Delete Candidate Nonbdd', which contains steps like 'S{fname}', 'S{lname}', 'S{email}', 'S{resume}', 'add candidate' (marked with a checkmark), and 'delete'; and 'Add candidate \${firstname} and \${lastname}' (also marked with a checkmark). On the right, a detailed view of the 'Add candidate \${firstname} and \${lastname}' step is shown. The title bar says 'Add candidate \${firstname} and \${lastname}'. Below it is a 'Settings >>' button. A table lists the steps with their descriptions and variable mappings:

Step	Description	Variable Mappings
1	input_text	xpath='//*[@id="addCandidateForm"]' value='\${firstname}'
2	Sleep	1
3	input_text	xpath='//*[@id="addCandidateForm"]' value='\${lastname}'
4		
5		
6		

- iii. Go to test case Add Candidate, replace those two input steps with:

Add candidate \${fname} \${lname}

Refer 1st picture

Note: Make sure name of variables are same with variable in test suite, not with keywords file name (refer 2nd pivoture)

- iv. Test case can be run.

1	Sleep	1
2	click element	xpath=//*[@id="menu_recru
3	Sleep	1
4	click element	xpath=//*[@id="btnAdd"]
5	Sleep	1
6	Add candidate \${fname} and \${lname}	
7	Sleep	1
8	input_text	xpath=//*[@id="addCandidateForm\${email}]
9	Sleep	1
10	Select From List By Label	xpath=//*[@id="addCandidateFormLabel"]
11	Sleep	1
12	click element	xpath=//*[@id="btnSave"]
13	Sleep	3
14	click element	xpath=//*[@id="btnBack"]
15	Sleep	2

Edit Text Edit Run

0 Add And Delete Candidate Nonbdd

Source D:\kerja\pakwanoranghr\Candidate Feature Test Suites\0_

Settings >>

Import Library	Name / Path SeleniumLibrary	Arguments	Add Import
< >		Library	
< >		Resource	
< >		Variables	
< >		Import Failed Help	
Variable		Value	
\${fname}		Ali	
\${lname}		Khalid	



For more detail by looking test suite that used setup, teardown, variable and keyword file, can refer to '1 Adding Candidate More Setting' test suite



Resource File

- *Keywords and variables from test suite A cannot be used on another test suits*
- *Resource files provide a mechanism (keywords and variable) for sharing them*

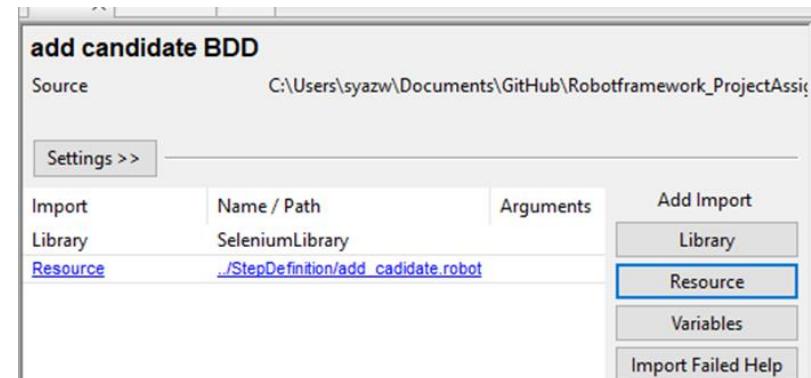
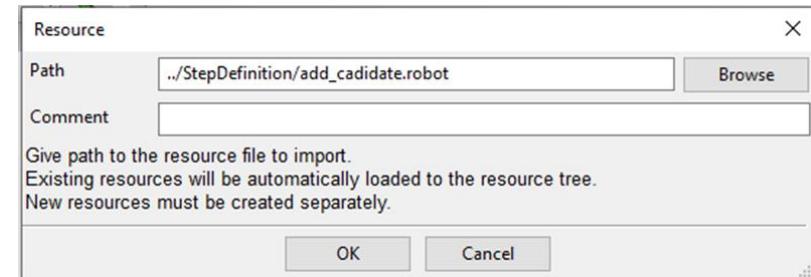
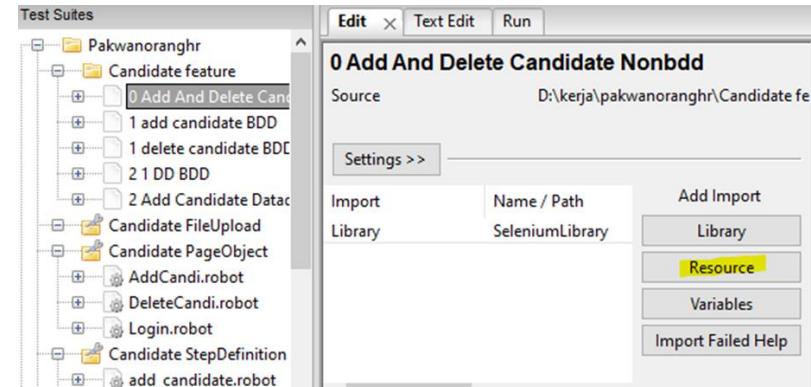
Steps:

- i. Create scenarios resource by right click on main folder (or other folder/directory) and click ‘New Resource’.
- ii. ‘New resource file’ window will be popped up. Give a name and click ‘OK’.
- iii. Click on that scenarios resource file
- iv. You can create variables or keywords inside resource file.

v. Go to any test suite. Then, click ‘Resource’ under Add Import

vi. ‘Resource’ window will be popped up. Browse scenario resource file as shown in below and click OK.

vii. The list of added library and resource is shown. If library and resource are invalid, colour of library and resource word become red.



Behavior-driven development

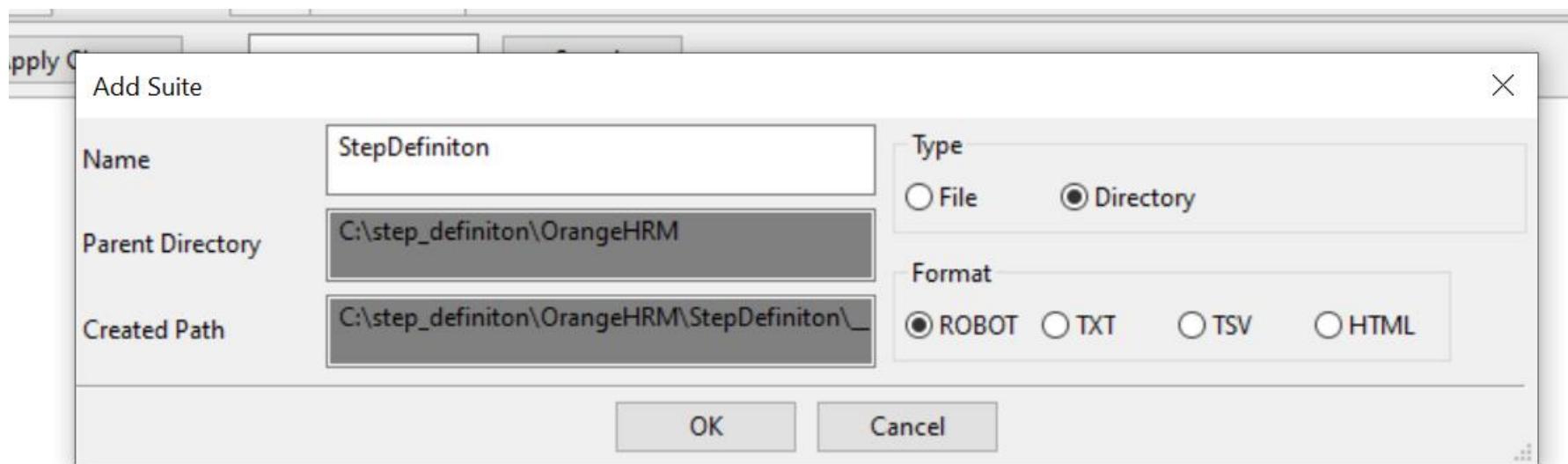


Behavior-driven development

- BDD is method when an application is documented and designed around the behavior a user expects to experience when interacting with it.
- BDD helps to avoid bloat, excessive code, unnecessary features or lack of focus from developer and others.

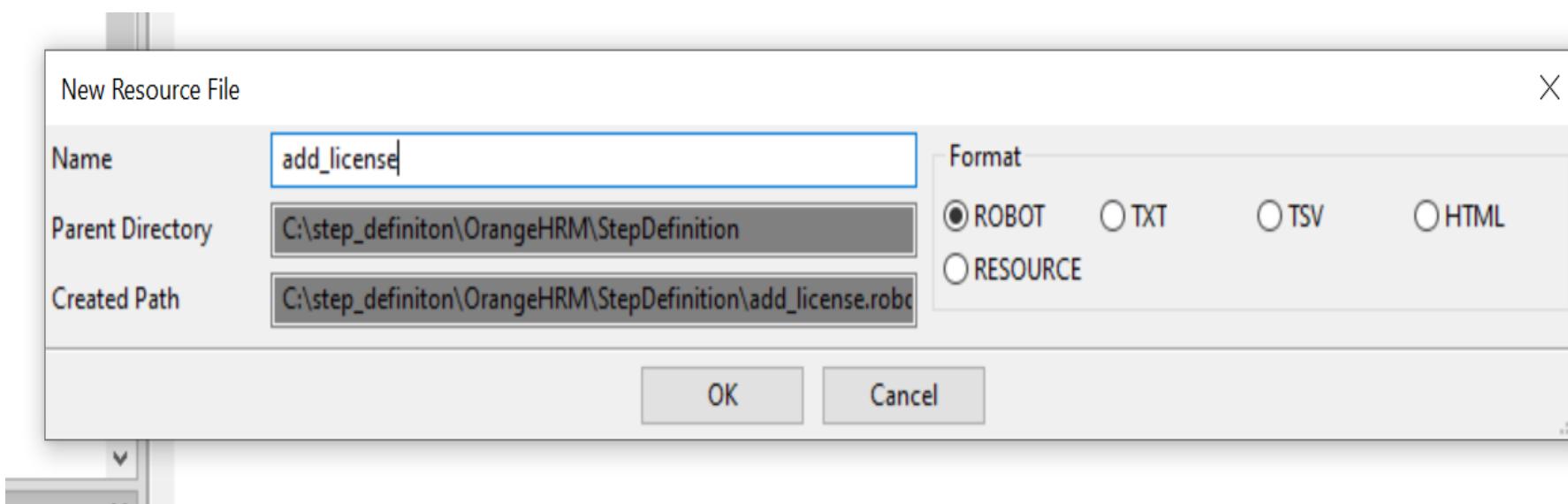
STEP 1 : Create StepDefinition directory folder

- i. Create new project
- ii. Right-click on the project and choose new suite. Select the type as ‘Directory’ and format ‘ROBOT’.
- iii. Name the file as ‘StepDefinition’ then click OK.



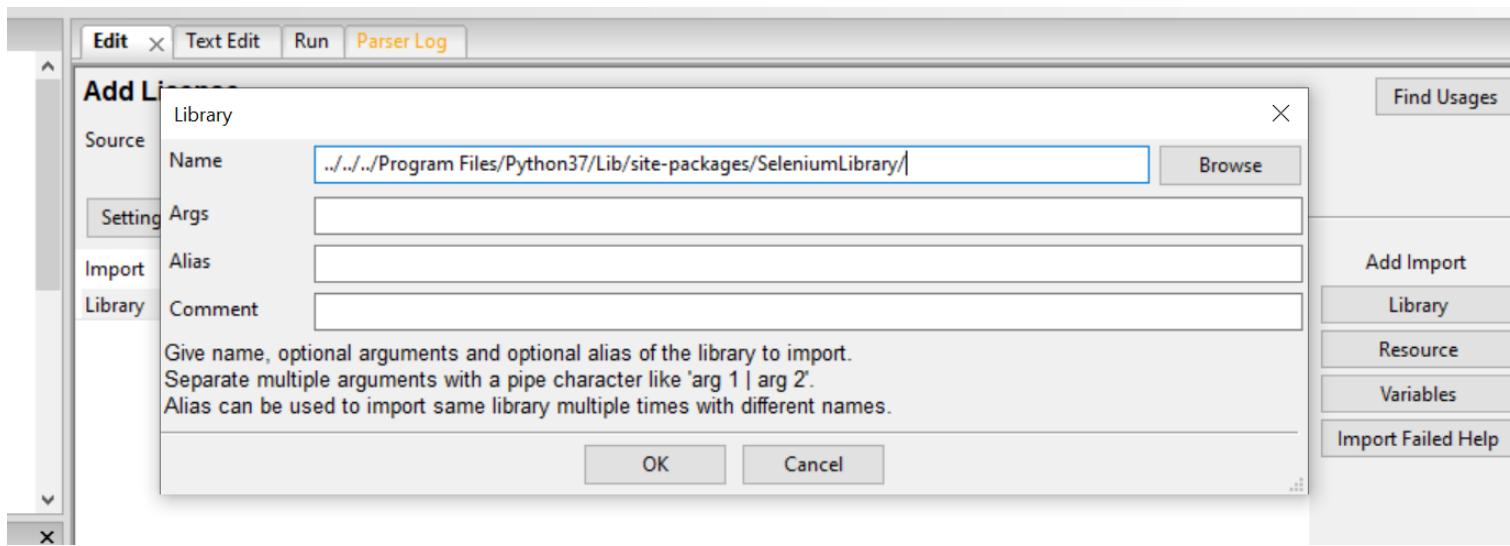
STEP 2: Create Step Definition directory folder

- i. Right-click on the StepDefinition file and choose new resource.
Name the file. Eg: Add_license.robot
- ii. Choose format ‘ROBOT’. Then click ‘OK’



STEP 3: Import library & resources

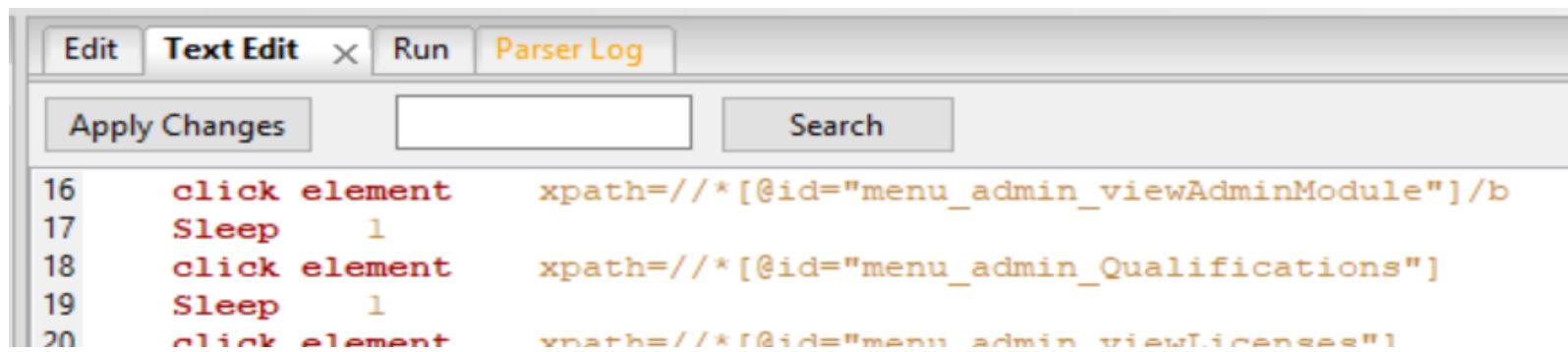
- i. Click on the 'add_license.robot' resources file
- ii. Click edit. Then click on 'Library' on the right-side.
- iii. Browse and choose your Selenium library folder.
- iv. Click 'OK'
- v. Click 'Resources' to add other resources file



STEP 4 :Create user keyword

There are two ways to create user-keyword

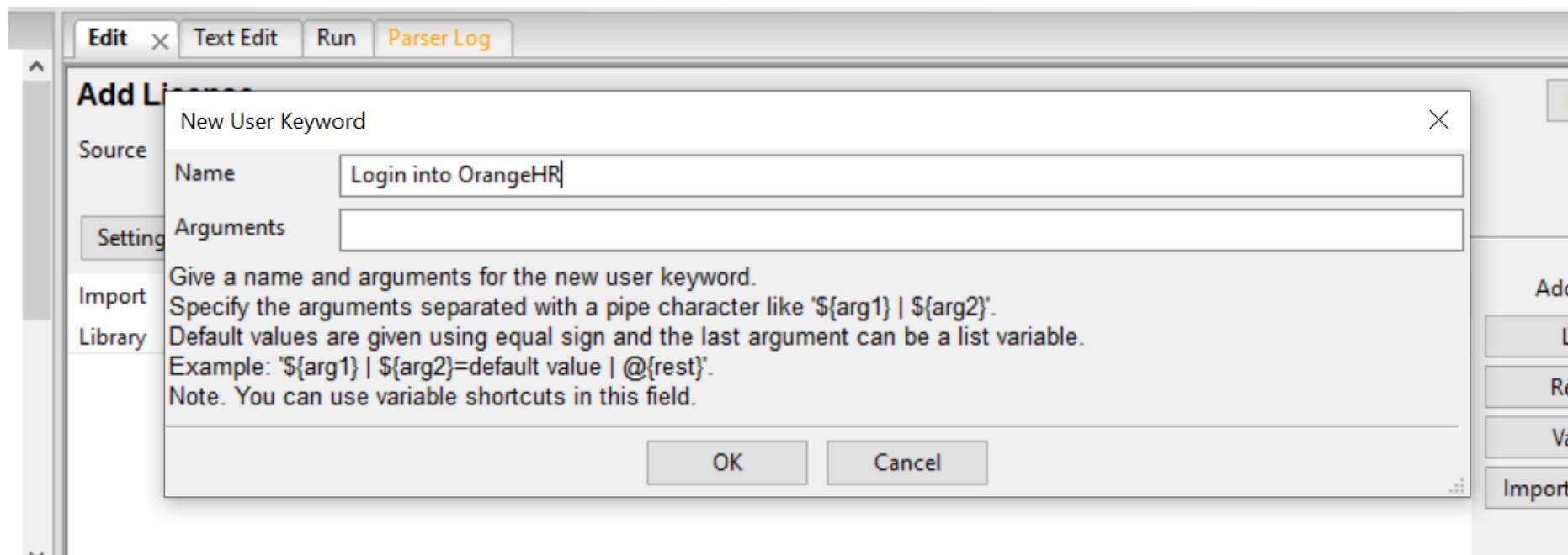
- i. Tab 'Edit'
- ii. Tab 'Text Edit'



```
16 click element    xpath=//*[@id="menu_admin_viewAdminModule"]/b
17 Sleep      1
18 click element    xpath=//*[@id="menu_admin_Qualifications"]
19 Sleep      1
20 click element    xpath=//*[@id="menu_admin_viewLicenses"]
```

STEP 4.1: Create user keyword (Tab ‘edit’)

- i. Right-click on resource file eg: add_license.robot
- ii. Select new user keyword
- iii. Enter new user keyword and click ‘OK’



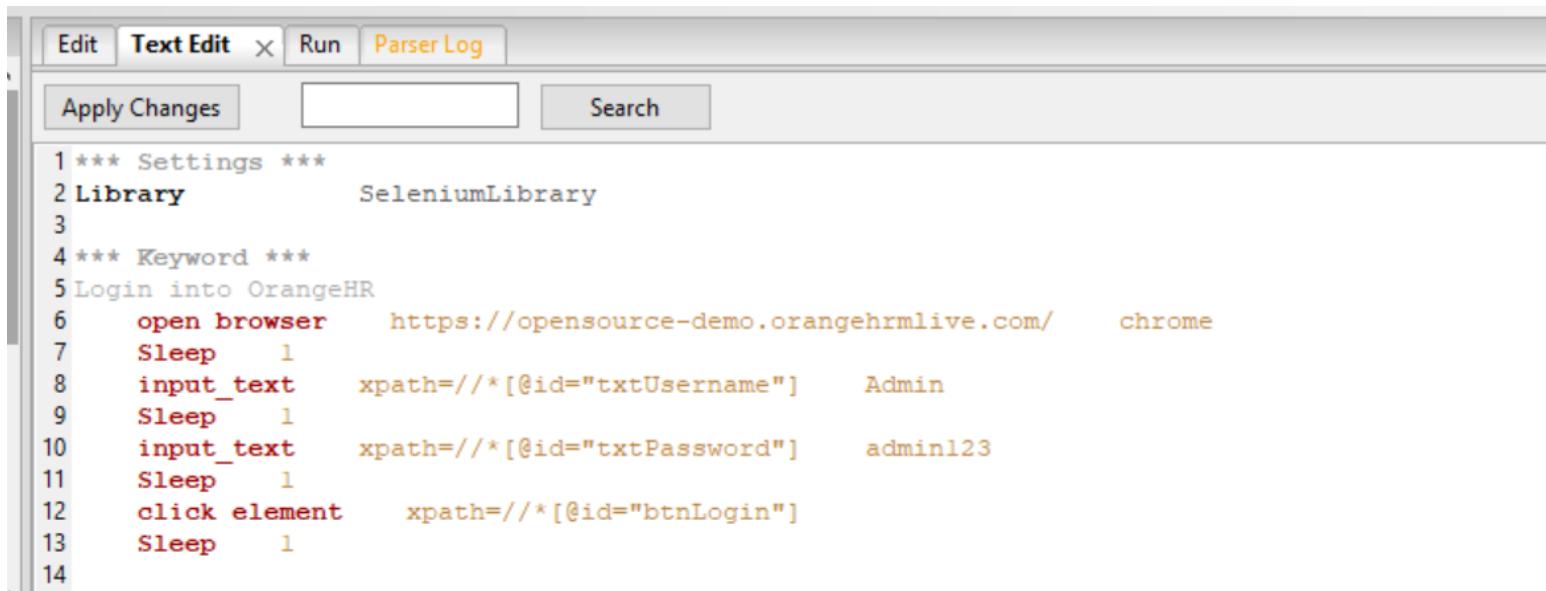
Write steps, xpath and value into scenario file as shown in below:
Then, save the file.

The screenshot shows a software interface for test automation. At the top, there's a menu bar with 'Edit', 'Text Edit', 'Run', and 'Parser Log'. Below the menu is a title 'Login into OrangeHR' and a 'Settings >>' button. To the right of the title is a 'Find' button. The main area is a grid table with 13 rows and 4 columns. The first column contains step numbers (1-13). The second column contains step descriptions: 'open browser', 'Sleep', 'input_text', 'Sleep', 'input_text', 'Sleep', 'click element', 'Sleep', an empty row, and three empty rows at the bottom. The third column contains URLs or XPaths: 'https://opensource-demo.orangehrm.com', 'xpath=//*[@id="txtUsername"]', 'Admin', 'xpath=//*[@id="txtPassword"]', 'admin123', 'xpath=//*[@id="btnLogin"]', '1', and three empty rows. The fourth column contains three empty rows.

1	open browser	https://opensource-demo.orangehrm.com	
2	Sleep	1	
3	input_text	xpath=//*[@id="txtUsername"]	
4	Sleep	1	
5	input_text	xpath=//*[@id="txtPassword"]	
6	Sleep	1	
7	click element	xpath=//*[@id="btnLogin"]	
8	Sleep	1	
9			
10			
11			
12			
13			

STEP 4.2 : Create user keyword (Tab ‘Text edit’)

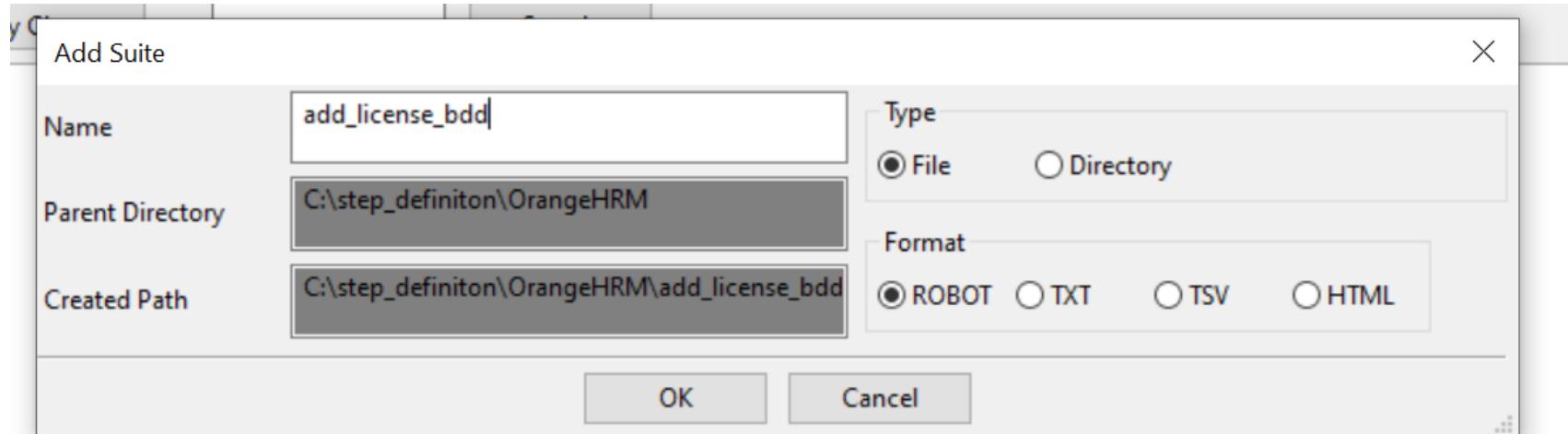
- i. Directly click on ‘Text Edit’ Tab
- ii. Write steps, xpath and value into scenario file as shown in below:
- iii. Then save the file



```
1 *** Settings ***
2 Library           SeleniumLibrary
3
4 *** Keyword ***
5 Login into OrangeHRM
6   open browser    https://opensource-demo.orangehrmlive.com/    chrome
7   Sleep      1
8   input_text    xpath=//*[@id="txtUsername"]    Admin
9   Sleep      1
10  input_text   xpath=//*[@id="txtPassword"]    admin123
11  Sleep      1
12  click element  xpath=//*[@id="btnLogin"]
13  Sleep      1
14
```

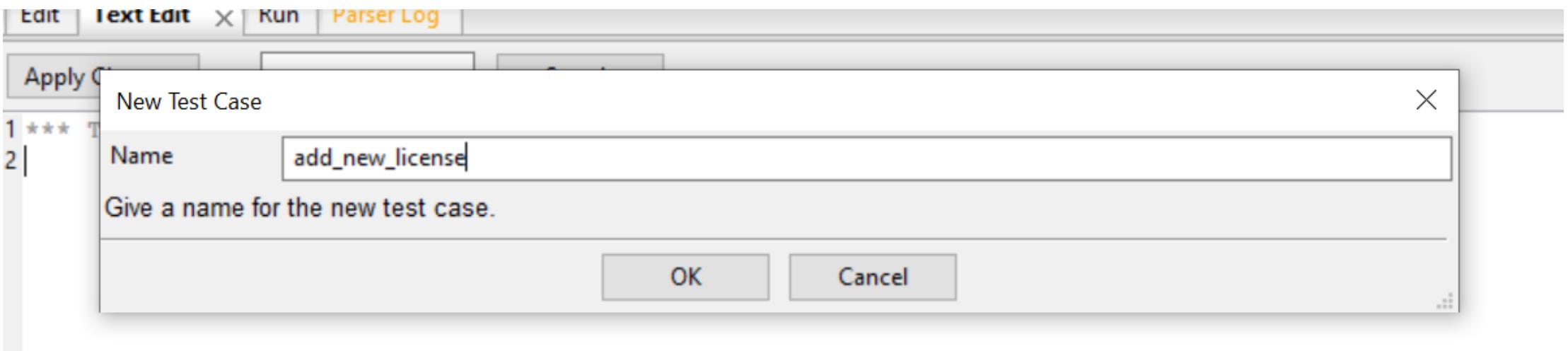
STEP 5 : Create new suite for BDD

- i. Right-click on feature folder and choose new suite
- ii. Name the BDD file eg: add_license_bdd
- iii. Choose type ‘File’ and format ‘ROBOT’ then click ‘OK’



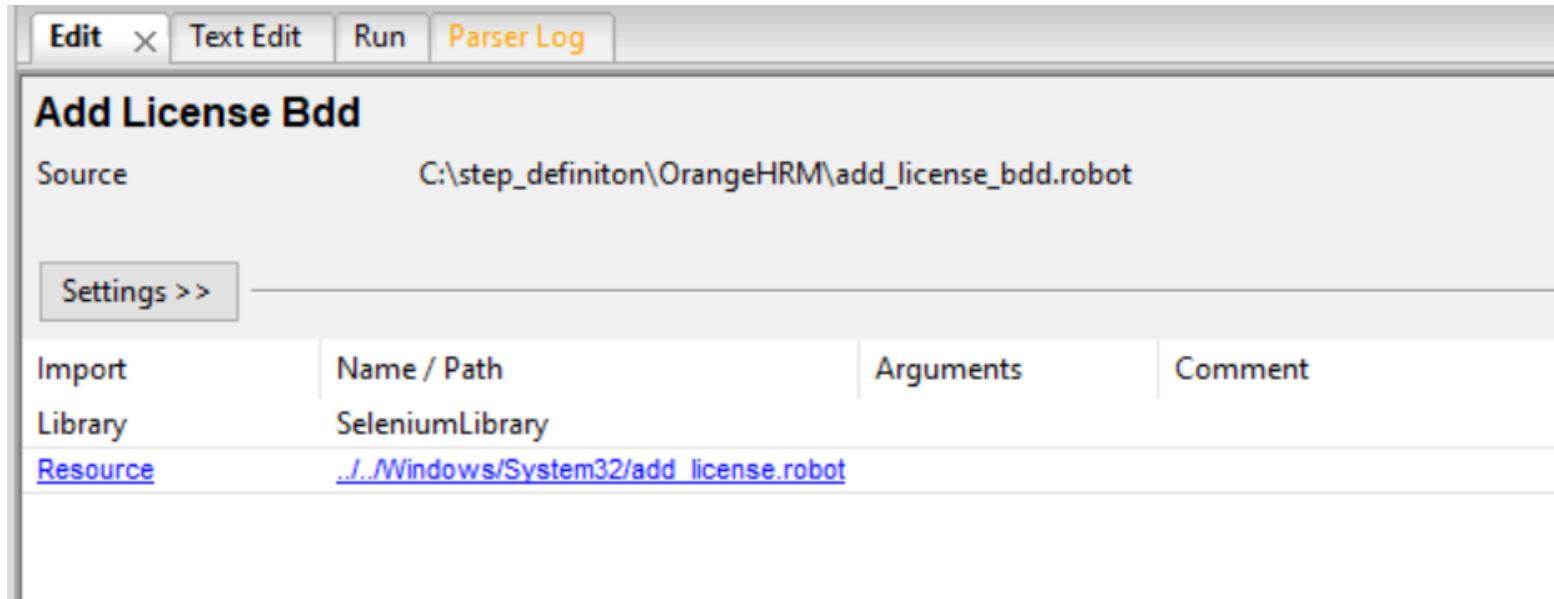
STEP 6 : Create new test case

- i. Right-click on BDD test suite file and choose new test case.
- ii. Name the tase case. Eg: add_new_license
- iii. Click 'OK'



STEP 7 : Import library and resources

- i. Click on library and import Selenium Library
- ii. Click on resources and import your resources file
- iii. Then, click 'OK'



STEP 8 : Write scenarios

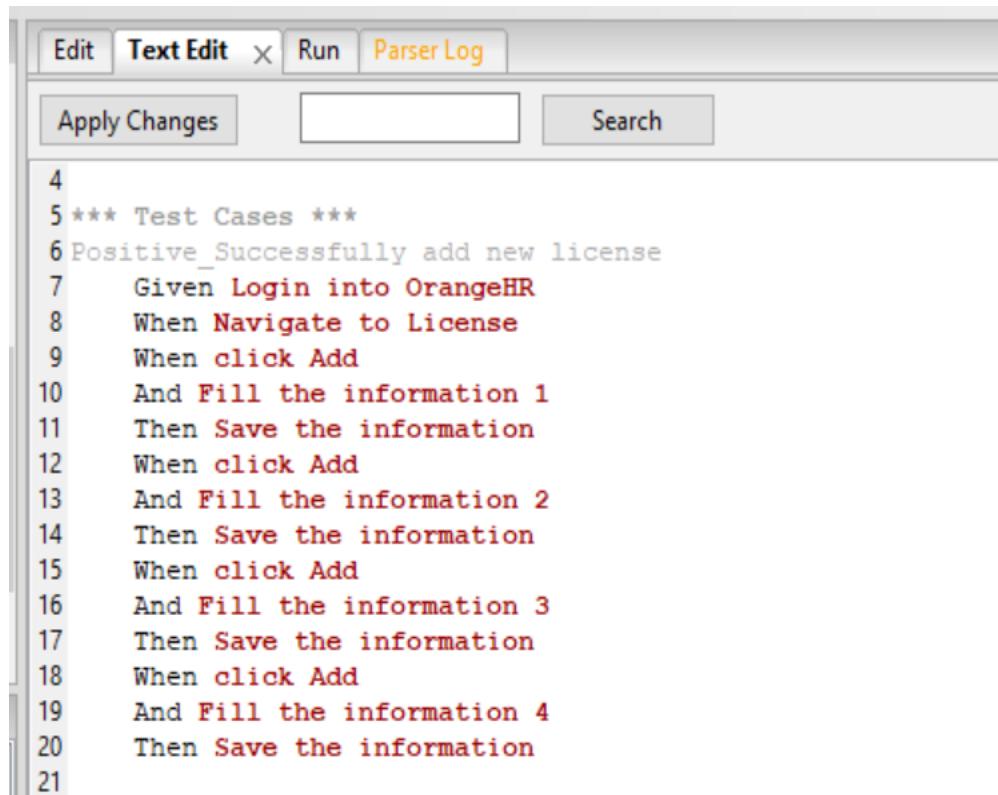
- i. You can write and edit your scenarios through ‘Edit’ tab
- ii. If the colour turn out ‘blue’ means the scenarios written are valid from your resources file.

The screenshot shows a software interface for writing and editing scenarios. The title bar includes tabs for 'Edit', 'Text Edit', 'Run', and 'Parser Log'. The main window is titled 'Positive_Successfully add new license'. On the left, there are sections for 'Documentation', 'Setup', 'Teardown', 'Timeout', 'Template', and 'Tags'. A button labeled '<Add New>' is located next to the 'Tags' section. Below these sections is a table containing the following steps:

1	Given Login into OrangeHR
2	When Navigate to
3	When click Add
4	And Fill the information
5	Then Save the
6	When click Add
7	And Fill the information
8	Then Save the
9	When click Add

STEP 9 : Write scenarios

- i. You can also write and edit your scenarios through ‘Text Edit’ tab



The screenshot shows a software interface titled 'Text Edit'. The window has a menu bar with 'Edit', 'Text Edit', 'Run', and 'Parser Log'. Below the menu is a toolbar with 'Apply Changes', an empty text input field, and a 'Search' button. The main area contains a numbered list of steps:

```
4
5 *** Test Cases ***
6 Positive_Successfully add new license
7 Given Login into OrangeHR
8 When Navigate to License
9 When click Add
10 And Fill the information 1
11 Then Save the information
12 When click Add
13 And Fill the information 2
14 Then Save the information
15 When click Add
16 And Fill the information 3
17 Then Save the information
18 When click Add
19 And Fill the information 4
20 Then Save the information
21
```

STEP 10: Run test case

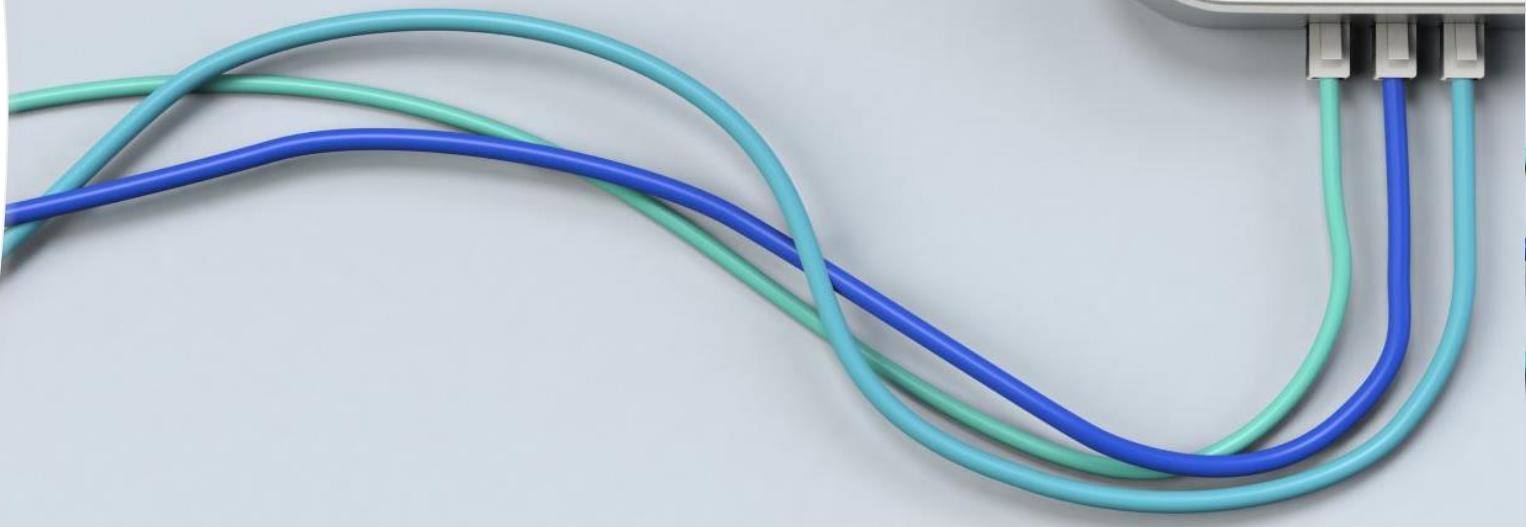
- i. Tick the test case you want to run.
- ii. Go to ‘Run’ tab
- iii. Click ‘Start’ robot button.

STEP 11: Test case report

- i. Click report button on ‘Run’ tab to check the report.

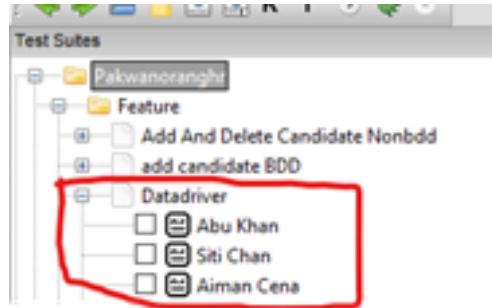
Data Driven (DD)

*Learn creating keywords and
variable first



Steps:

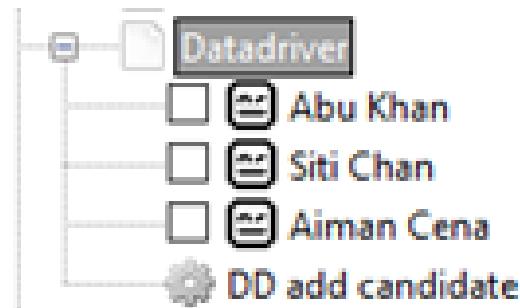
1. Create test suite for data driven testing (e.g: named as Data Driven)
2. Create test cases that represent data. For this example, create three test cases



3. Click on test case and click Edit tab. Inside test cases, put data value for testing. For example, each test cases have three data value such as first name, last name and email. Then, put data value to another test cases

1	Abu	Khan	abukhan@kmail.com
2			
3			

4. On Data Driven test suite, create Keywords (exp: named as DD add candidate)



5. Click on Data Driven test suite and then click on Text Edit tab.

6. Under **Setting**, type *Test Template DD add candidate* (name for keyword)

```
1 *** Settings ***
2 Test Template      DD add candidate
3 Library           SeleniumLibrary
4
5 *** Test Cases ***
6 Abu Khan
7   Abu     Khan    abukhan@kmail.com
8
9 Siti Chan
10  Siti    Chan    sitichan@kmail.com
11
12 Aiman Cena
13  Aiman   Cena    aimancena@kmail.com
14
15 *** Keywords ***
16 DD add candidate
17
```

7. Under 'DD add candidate', type [Arguments] name of variables that will be used as following:

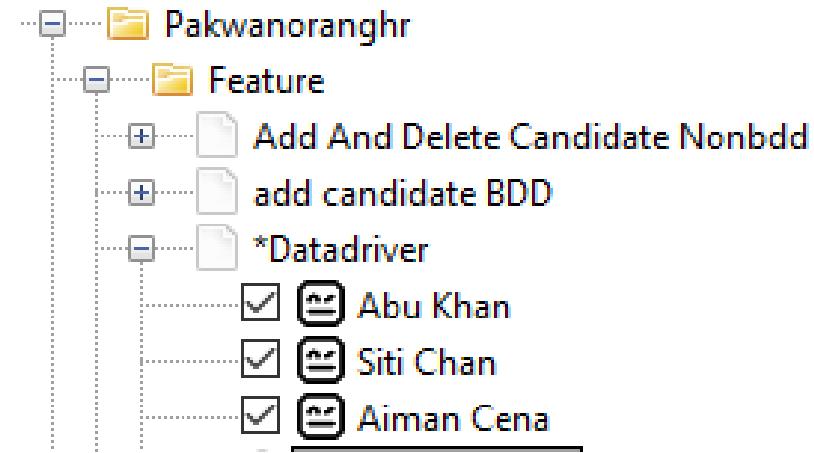
```
1 *** Settings ***
2 Test Template      DD add candidate
3 Library           SeleniumLibrary
4
5 *** Test Cases ***
6 Abu Khan
7   Abu    Khan    abukhan@kmail.com
8
9 Siti Chan
10  Siti   Chan    sitichan@kmail.com
11
12 Aiman Cena
13  Aiman  Cena    aimancena@kmail.com
14
15 *** Keywords ***
16 DD add candidate
17  [Arguments]    ${dd_fname}    ${dd_lname}    ${dd_email} ←
18
19
```

8. Then, type steps (either on Edit or Test Edit tab) in ‘DD add candidate’ keywords. For value, put name of variables on third columns for some steps as shown below:

Edit			Text Edit	Run	Print Log
DD add candidate					
Settings >>					
1	open browser	https://opensource-demo.orangehrmlive.com			
2	Sleep	1			
3	input_text	xpath=//*[@id="txtUsername"]			
4	Sleep	1			
5	input_text	xpath=//*[@id="txtPassword"]			
6	Sleep	1			
7	click element	xpath=//*[@id="btnLogin"]			
8	Sleep	1			
9	click element	xpath=//*[@id="menu_recruitment_viewCandidateList"]			
10	Sleep	1			
11	click element	xpath=//*[@id="btnAdd"]			
12	Sleep	1			
13	input_text	xpath=//*[@id="addCandidate.firstName"]			
14	Sleep	1			
15	input_text	xpath=//*[@id="addCandidate.lastName"]			
16	Sleep	1			
17	input_text	xpath=//*[@id="addCandidate.email"]			
18	Sleep	1			
19	Select From List By Label	xpath=//*[@id="addCandidate.vacancy"]			Senior QA Lead
20	Sleep	1			
21	click element	xpath=//*[@id="btnSave"]			
22	Sleep	3			
23	click element	xpath=//*[@id="btnBack"]			
24	Sleep	2			

```
DD add candidate
[Arguments]    ${dd_fname}    ${dd_lname}    ${dd_email}
Sleep    1
input_text    xpath=//*[@id="addCandidate(firstName]"]
Sleep    1
input_text    xpath=//*[@id="addCandidate(lastName]"]
Sleep    1
input_text    xpath=//*[@id="addCandidate(email]"]
Sleep    1
Select From List By Label    xpath=//*[@id="addCandidate(vacancy]"]
Sleep    1
click element    xpath=//*[@id="btnSave"]
Sleep    3
click element    xpath=//*[@id="btnBack"]
Sleep    2
```

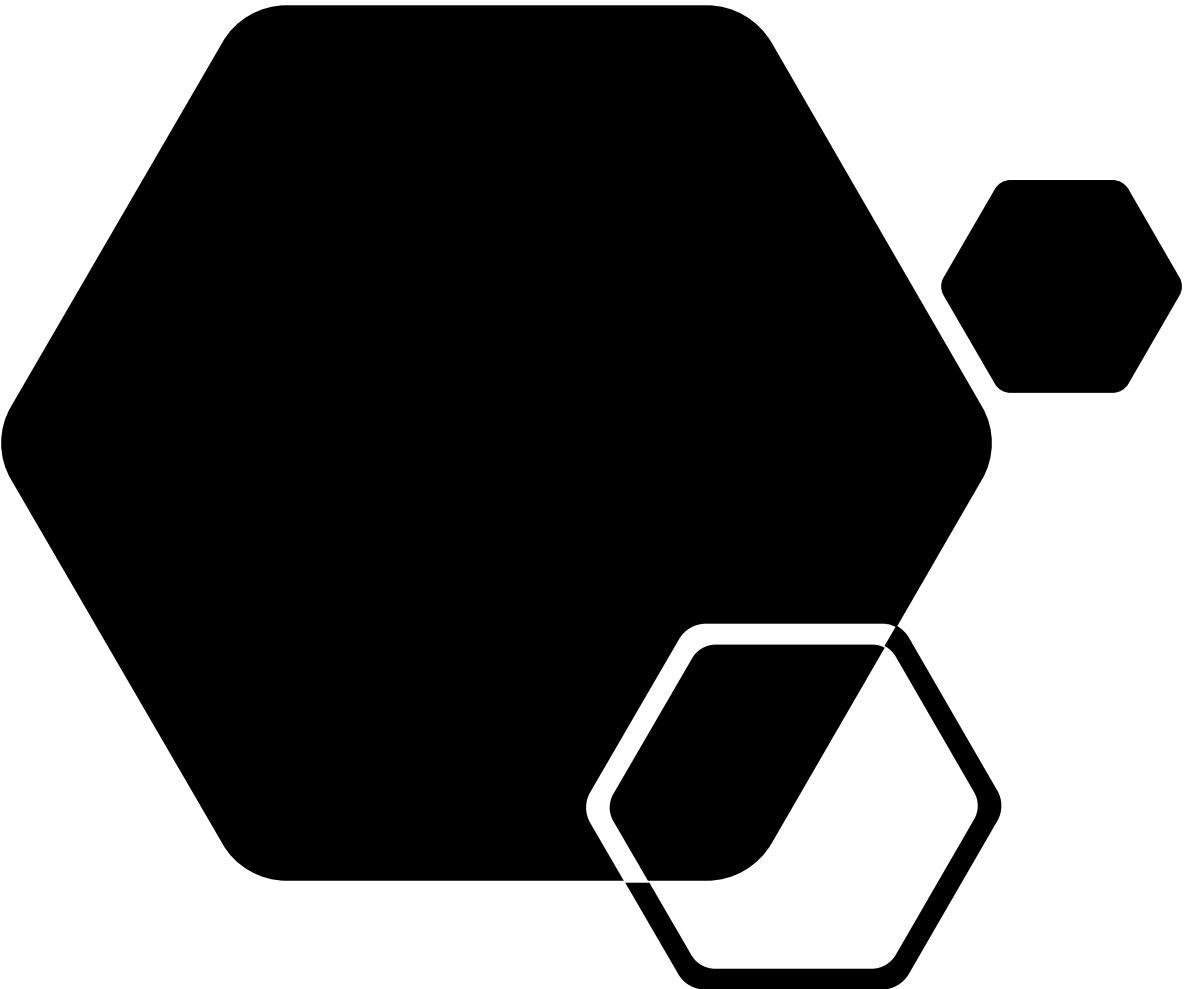
9. Tick test cases.



10. Go to *Run* tab, and then click *Start* . It runs ticked test cases.

Data Driven + BDD

*Learn creating keywords and variable
first



Steps:

Take Success Add Candidate BDD
for example

1. Create new Test Suite which import StepDefinition resources file.
2. Create test cases that represent data (refer to 4 fill information).
3. Create Keywords and put argument and BDD step (from previous BDD test case)
4. Set 'Test Template' with name of keywords

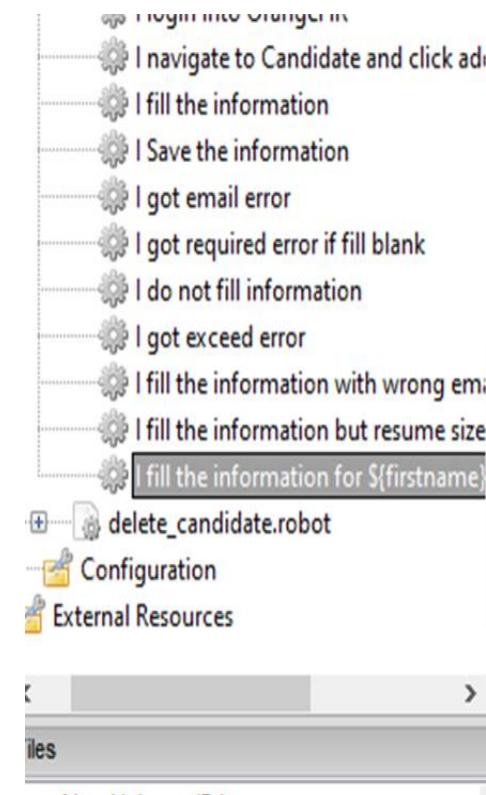
The screenshot shows a test template in a Robot Framework editor. The template includes settings for a Selenium library and three test cases with data. It also defines a keyword named 'DD_Positive_add_candidate_success_bdd' with arguments for first name, last name, and email, and describes the steps for logging in, navigating to the candidate page, filling information, and saving it.

```
1 *** Settings ***
2 Test Template      DD_Positive_add_candidate_success_bdd
3 Library           SeleniumLibrary
4 Resource          ../Candidate StepDefinition/add_candidate.robot
5
6 *** Test Cases ***
7 Abu Khan
8     Abu      Khan      abukhan@kmail.com
9
10 Siti Chan
11    Siti     Chan      sitichan@kmail.com
12
13 Aiman Cena
14    Aiman    Cena      aimancena@kmail.com
15
16 *** Keywords ***
17 DD_Positive_add_candidate_success_bdd
18     [Arguments]  ${dd_fname}  ${dd_lname}  ${dd_email}
19     Given I login into OrangeHR
20     When I navigate to Candidate and click add
21     And I fill the information
22     Then I Save the information
```

5. We want step 'I fill information' be used for data driven. So, recall create keywords file will variable and create keywords file and name it as:

'I fill information \${firstname}, \${lastname} and \${email}'

6. Inside Keywords file, change value (in third column and row that involve only) into name of variable



The screenshot shows the Robot Framework Test Case Editor. A context menu is open over the keyword 'I fill the information for \${firstname}'. The menu items listed are: 'I navigate to Candidate and click add', 'I fill the information', 'I Save the information', 'I got email error', 'I got required error if fill blank', 'I do not fill information', 'I got exceed error', 'I fill the information with wrong emai', 'I fill the information but resume size', and 'I fill the information for \${firstname}'.

I fill the information for \${firstname}, \${lastname} and \${email}			
Settings >>			
1	input_text	\${ac_lname}	\${firstname}
2	Sleep	1	
3	input_text	\${ac_fname}	\${lastname}
4	Sleep	1	
5	input_text	\${ac_email}	\${email}
6	Sleep	1	
7	Select From List By Label	\${ac_jobva}	Senior QA Lead
8	Sleep	1	
9	Choose File	\${ac_uploadresume}	D://kerja/pakwanoranghr/Fil
10	Sleep	1	

7. Go back to Test Suite. Change 'I fill information' into

'I fill information `${dd_fname}`, `${dd_lname}` and `${dd_email}`'

*Note: name of variables must be same with argument variable

```
16 *** Keywords ***
17 DD_Positive_add_candidate_success_bdd
18     [Arguments]    ${dd_fname}    ${dd_lname}    ${dd_email}
19     Given I login into OrangeHR
20     When I navigate to Candidate and click add
21     And I fill the information for ${dd_fname}, ${dd_lname} and ${dd_email}
22     Then I Save the information
23
```

8. Go to *Run* tab, and then click *Start*. It runs ticked test cases.

Reference

- <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html>
- https://www.tutorialspoint.com/robot_framework/index.htm
- <https://youtu.be/ErTN5rE6t8s>
- <https://youtube.com/playlist?list=PLhW3qG5bs-L9I2I8K8dEhw6HXy-Z-33w3>