* **Robot Framework Features**
* Robot Framework is a generic test automation framework
* Robot Framework is a Python-based keyword-driven test automation framework
* By default Robot Framework creates an XML output file and a log and a report in HTML format.
* Robot Framework is open source software
* Official sit is for more information <http://robotframework.org/>
* Enables easy-to-use tabular syntax for [creating test cases](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#creating-test-cases) in a uniform way.
* Provides ability to create reusable [higher-level keywords](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#creating-user-keywords) from the existing keywords.
* Provides easy-to-read result [reports](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#report-file) and [logs](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#log-file) in HTML format.
* Is platform and application independent?
* Provides a simple [library API](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#creating-test-libraries) for creating customized test libraries which can be implemented natively with either Python or Java.
* Provides a [command line interface](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#id175) and XML based [output files](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#output-file) for integration into existing build infrastructure (continuous integration systems).
* Provides support for Selenium for web testing, Java GUI testing, running processes, Telnet, SSH, and so on.
* Supports creating [data-driven test cases](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#data-driven-style).
* Has built-in support for [variables](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#variables), practical particularly for testing in different environments.
* Provides [tagging](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#tagging-test-cases) to categorize and [select test cases](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#selecting-test-cases) to be executed.
* Enables easy integration with source control: [test suites](http://robot-framework.readthedocs.org/en/master/autodoc/robot.running.html#robot.running.model.TestSuite) are just files and directories that can be versioned with the production code.
* Provides [test-case](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#test-setup-and-teardown) and [test-suite](http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#suite-setup-and-teardown) -level setup and teardown.
* The modular architecture supports creating tests even for applications with several diverse interfaces.
* **INSTALL ROBOT FRAMEWORK**

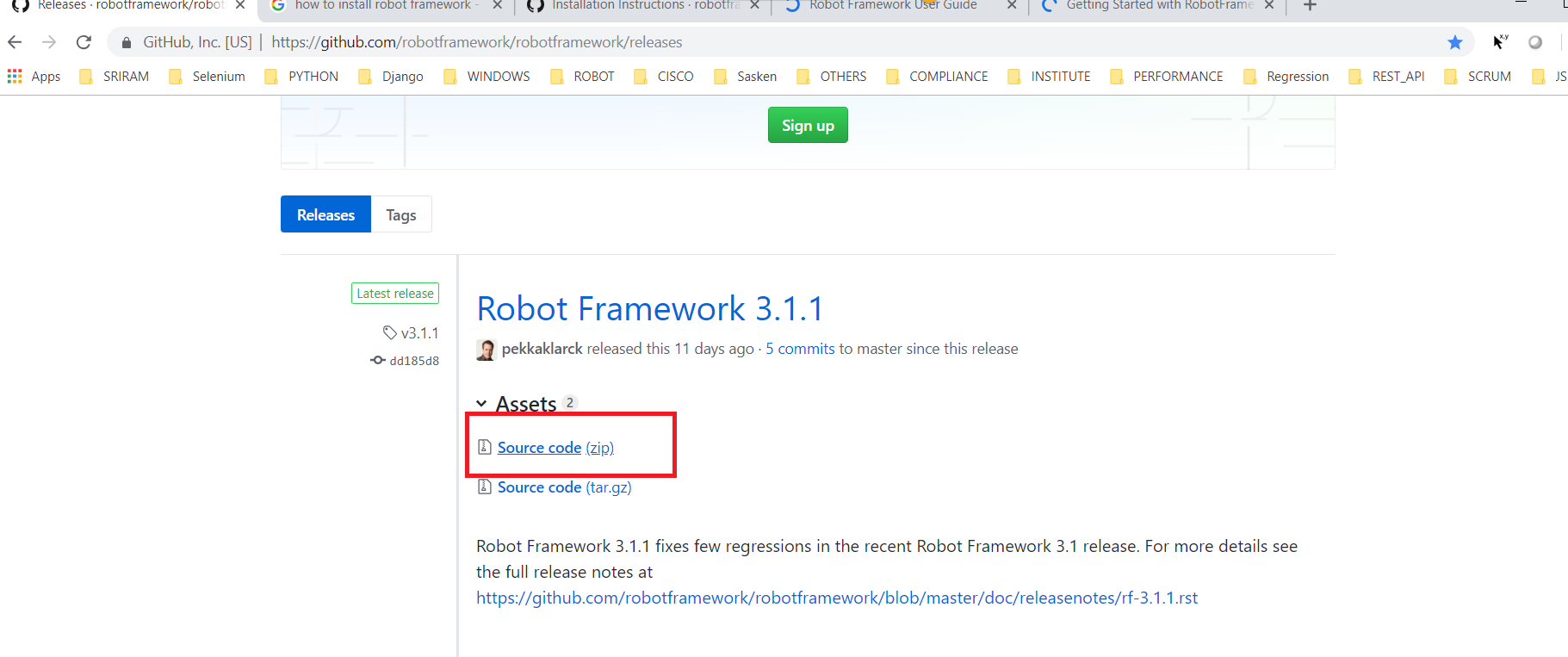
**pip install robotframework**

OR

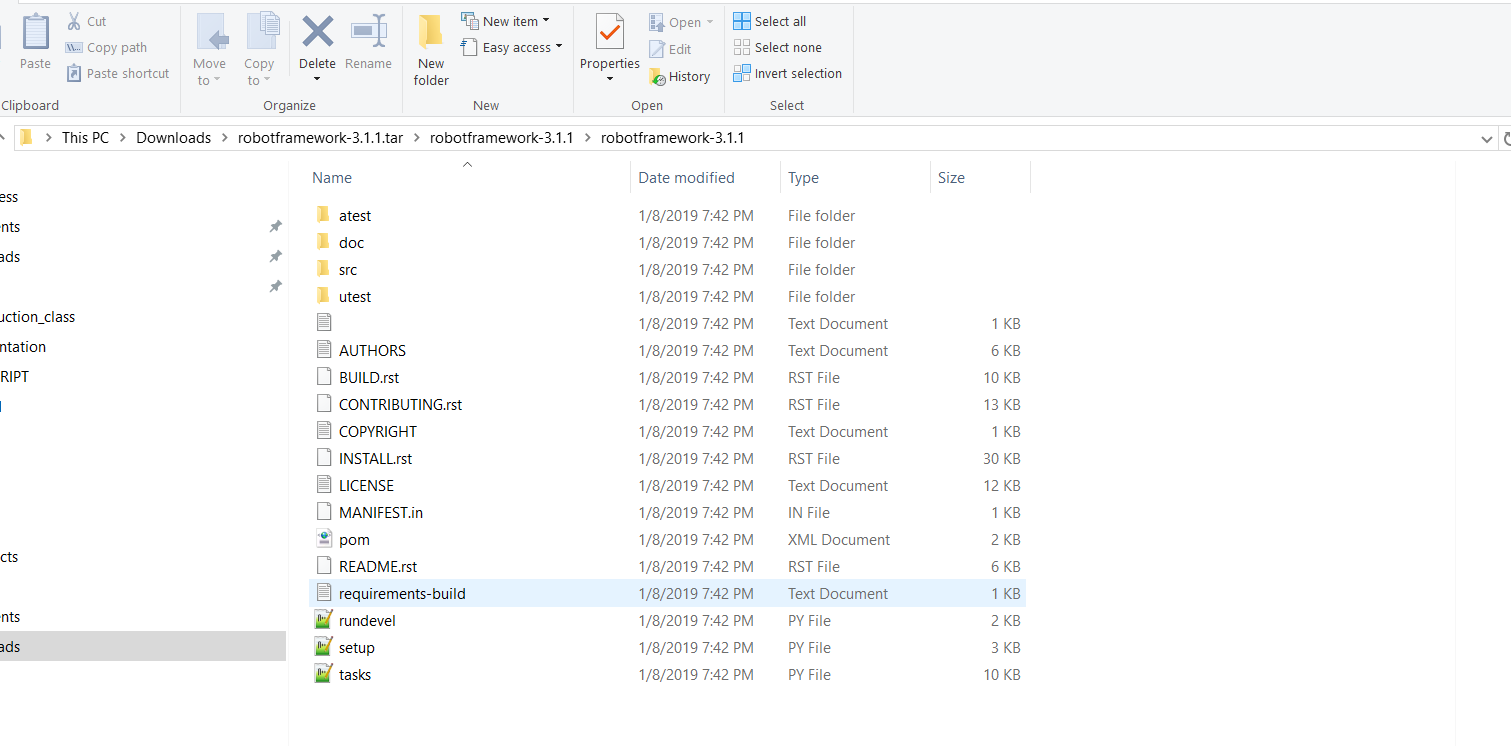
Download tar.gz file form below path

<https://github.com/robotframework/robotframework/releases>

* Click on Source code.(zip) to download



* UNZIP Sourcecode.zip file after unzipping you will see below files



* Then run this command :- python setup.py install
* Run below command to know installed or not

**robot –version**

**IMP Features in robot Framework**

**[Teardown]**

**Tear down will execute after executing the test case**

**[Setup]**

**Robot Framework has similar test setup and teardown functionality as many other test automation frameworks. In short, a test setup is something that is executed before a test case**

**Suite Setup :--**

**It will execute after executing the all test cases**

**Suite Teardown:-**

**It will execute before executing the all test cases**

**[Tags]**

**Used to give alias name to the test case**

**With tags, you can include or exclude test cases to be executed.**

**[Documentation]**

**Allows you to set a free documentation for a test case. That text is shown in the command line output, as well as the resulting test logs and test reports.**

**# To know robot version**

**pybot --version**

**robot --version**

**# paasing values to variable through command line =>**

**pybot -v ONE:10 Loop.txt**

**# create log file ==>**

**pybot -b debug.log test2\_list.robot**

**# validate the test data ==>**

**pybot --dryrun test2\_list.robot**

**# To run particular test case ==>**

**pybot -t "Add Two Values" first\_test.txt**

**robot -t Addition -t "Test Case 3" first\_test.txt**

**pybot -t Add\* first\_test.txt**

**# TO run a test case based on tag =>**

**pybot -i test2 first\_test.txt**

**# TO run all test cases except given tag =>**

**pybot -e test2 first\_test.txt**

**# Change logs path ==>**

**pybot -d C:\Users\smellamp\Desktop\SRIRAM\INSTITUTION\COURSES\ROBOT\_FRAMWEORK\Logs test2\_list.robot**