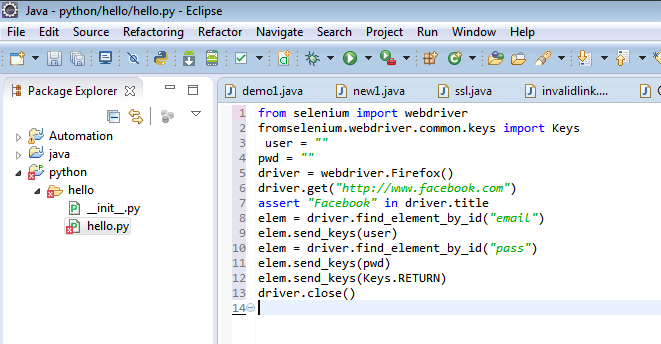
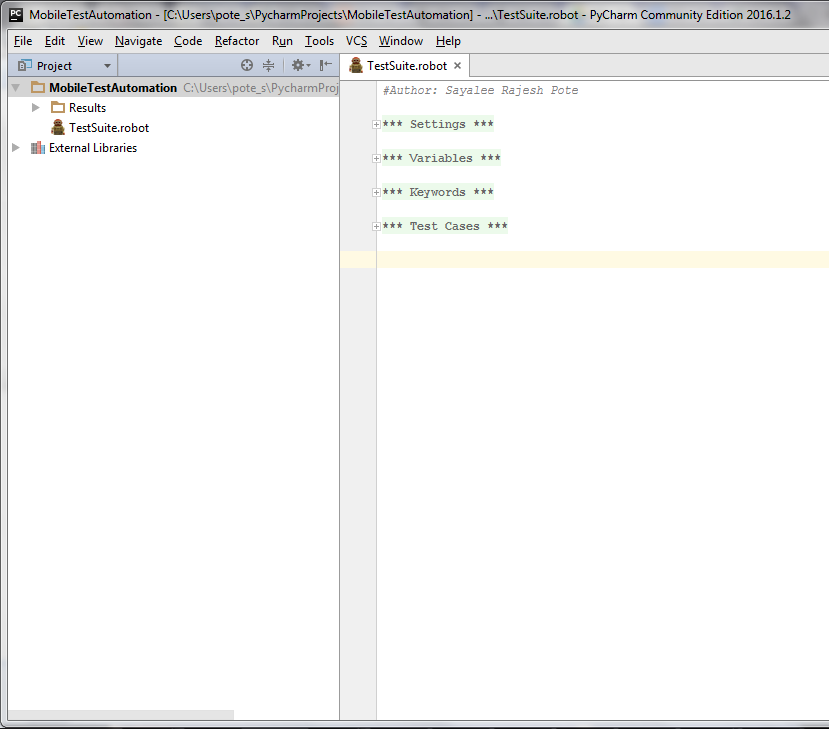
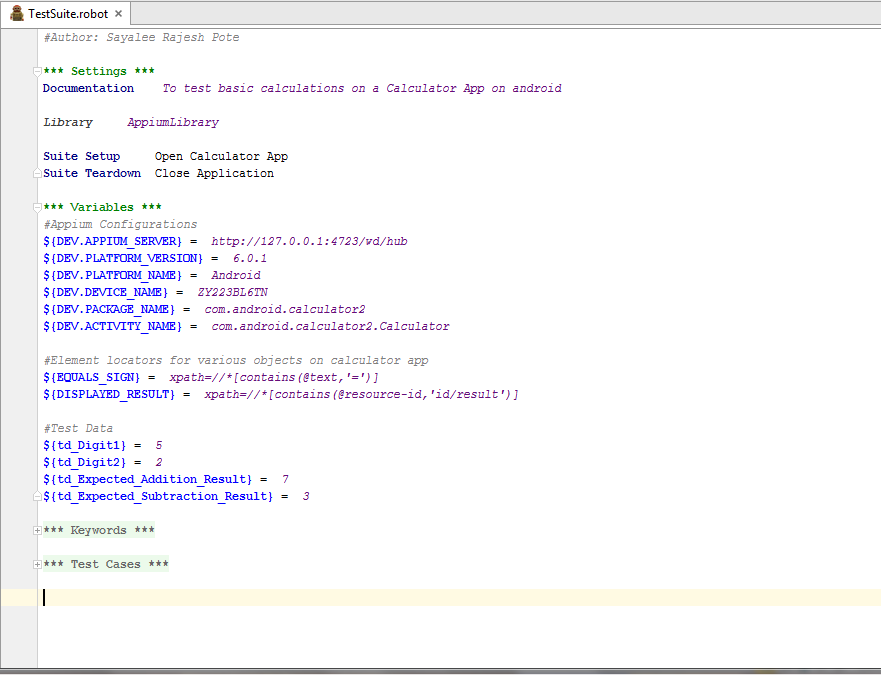
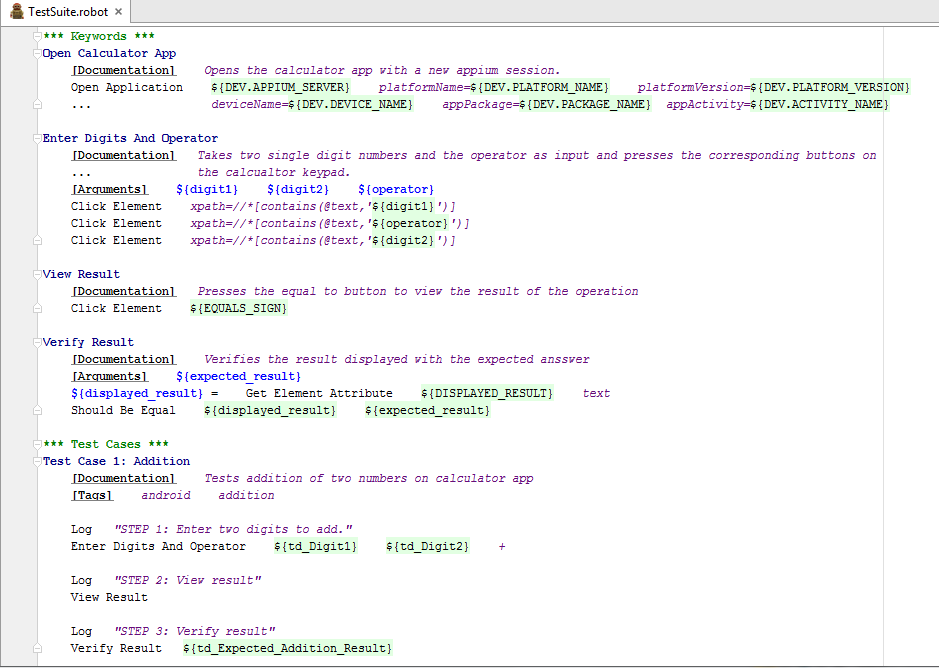
RobotFramework

* UI Automator is a UI testing framework suitable for cross-app functional UI testing across system and installed apps.
* The UI Automator testing framework provides a set of APIs to build UI tests that perform interactions on user apps and system apps. The UI Automator APIs allows you to perform operations such as opening the Settings menu or the app launcher in a test device. The UI Automator testing framework is well-suited for writing black box-style automated tests, where the test code does not rely on internal implementation details of the target app.
* The key features of the UI Automator testing framework include the following:
* A viewer to inspect layout hierarchy. For more information
* An API to retrieve state information and perform operations on the target device. For more information.
* APIs that support cross-app UI testing. For more information.
* The uiautomatorviewer tool is located in the <android-sdk>/tools/ directory.
* The UI Automation Test Library (UIA Test Library) is an API that is called by the *driver* application in an automated testing scenario.
* The driver is the application that obtains an automation element ([IUIAutomationElement](https://docs.microsoft.com/en-us/windows/desktop/api/UIAutomationClient/nn-uiautomationclient-iuiautomationelement) object) from a control that requires verification, and provides it to the UI Automation Test Library
* **UI** Automator is an Android **UI** framework for mobile **testing** developed and maintained by Google. Its key features include **cross**-**app functional testing**, the ability to **test** multiple **apps**, and switch between installed and system**apps**
* Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.
* Selenium is the first automated web testing tool that **allowed users to use a programming language they prefer**s the first automated web testing tool that **allowed users to use a programming language they prefer**
* 
* Snapshot of a code using python with selenium.
* <https://blog.codecentric.de/en/2012/03/robot-framework-tutorial-overview/> check this for introduction to automated testing.
* Keyword is a function or method that can be used to test (or help test) one (smaller or bigger) aspect of the SuT (“System under Test”)
* [Robot Framework](http://robotframework.org/) is a generic open source test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach
* This framework we are using with sublime text as a plugin.
* Higher-level keywords: Those are really testing a concrete aspect of the business logic of the system under test.
* Lower-level keywords: To keep the implementation of the higher-level keywords at a decent size one is often breaking down the required functionality to several lower-level keywords.
* Technical keywords: Those provide the technical implementation to access and thus test the system.
* Test cases are written on external files called resource files
* Testsuites: This is where testcases (checks) are implemented. Typically every project will at least have one Testsuite. In bigger projects it makes for sure sense to divide the test functionality further into different Testsuites.
* Resource Files: As it makes much sense from a test-design point of view one will almost always define own higher-level keywords. In turn this means that one will typically have own Resource Files. Especially in product development or in some long-running projects it makes sense to implement certain keywords centrally that fit to the product/project very well and can then be used by all teams.
* Test Library: Typically you are not required to write new technical keywords – for which you would need to implement a new Test Library – unless you are using a very specific technology. Nevertheless this is possible and does not always neccessarily mean a lot of effort.
* The Robot Framework itself and its Core Libraries are written in Python.
* With Jython it is possible to run Python code in a Java Virtual Machine.
* Remote Libraries enable us to write Test Libraries in basically any programming language supporting the XML-RPC protocol and running them on different machines (than the one running the Robot Framework as such) if required.
* Rpc-Xml is a spec and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet. It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.
* To start with this framework for android these things have to be done:
  + A **wrapper function** is a subroutine in a software library or a computer program whose main purpose is to call a second subroutine or a system call with little or no additional computation.
  + This module is a Python wrapper of Android uiautomator testing framework. It works on Android 4.1+ simply with Android device attached via adb, no need to install anything on Android device.
  + Refer <https://github.com/xiaocong/uiautomator> for complete example and explanation of example.
* A test suite is a .robot file which can be written and executed using a python IDE. The basic skeleton of a test suite written using robot framework’s syntax consists of the following sections.
* This is how test suite looks like:



* <https://www.xoriant.com/blog/product-engineering/robot-framework-mobile-test-automation.html>
* settings – This section consists of the test suite documentation, imports of libraries and resource files, suite and test level setup and teardown. (Fig 4)
* Variables – This section consists of all the variable declarations for the variables used in the test suite. (Fig 4)
* Keywords – This section consists of the higher level keywords formed using in built keywords from robot’s standard libraries and appium library. (Fig 5)
* Test Cases – This section consists of all the test cases that belong to the test suite. (Fig 5)
* 
* 
* UIAutomator is a tool used to obtain the locators of all the elements on a particular android application. It is a part of the android SDK.
* XPath uses "path like" syntax to identify and navigate nodes in an XML document
* The locators in the form of xpaths were obtained using UIAutomator for the calculator app
* Test Reports and Logs: test suite can be executed using the following command on python terminal:
  + *pybot -d Results\TestSuite TestSuite.robot*
* On execution of the test suite, report and log files are created in the form of HTML documents. These files have a detailed summary of the test case execution and all the necessary statistics related to the test case execution
* In conclusion, the appium library of robot framework facilitates automation of test cases for mobile applications with a simple tabular syntax, which is easy to read and is platform independent, without altering the source code of the application under test. The keyword driven approach of robot framework ensures the reusability and readability of the test cases, thus making the automation framework robust and tester friendly.