Understand what is automation framework.

An Automation Framework is a set of guidelines like coding standards , test-data handling , object repository etc. which when followed during automation scripting produce beneficial outcomes like increase code re-usability, higher portability, reduced script maintenance cost etc. Mind you these are just guidelines and not rules; they are not mandatory and you can still script without following the guidelines. But you will miss out on the advantages of having a Framework.

Language is not the constraint for creating any type of automation framework. If you are using Selenium, you can use any language. Here in your case, it is python.

There are some basic automation frameworks available.

1. Keyword Driven : The Keyword-Driven or Table-Driven framework requires the development of data tables and keywords, independent of the test automation tool used to execute them. Tests can be designed with or without the Application. In a keyword-driven test, the functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test.
2. Data Driven : In this Framework, while Test case logic resides in Test Scripts, the Test Data is separated and kept outside the test scripts. Test Data is read from the external files such as : Excel sheets, Text Files, CSV Files, DataBases and are loaded into the variables inside the test script. Variables are used for both Input values and for verification/validation of the values.
3. Hybrid : As the name suggests this framework is the combination of one or more frameworks discussed above pulling from their strengths and trying to mitigate their weaknesses. This hybrid test automation framework is what most frameworks evolve into over time and multiple projects.

Most of the companies go for Hybrid, as we can leverage keyword, test data approach as well as POM (Page Object Model design pattern which is booming market).

My recommendation would be to use the Hybrid framework which would be real benefit for your tests.

Test automation frameworks based on Python continues to become popular – just like the programming language. However, the test automation frameworks from different developers vary in terms of features, performance, supported platforms, support, efficiency and more.

In its simplest form, a framework is a set of tools, libraries, best practices, and some assumptions that various teams rely on when testing software. In most cases, the testing needs may vary according to the app and the environment.

As such, when looking for a test automation framework, the software developers and testing teams need to consider a wide range of factors. These include the quality of the framework, its effectiveness, ease of use, complexity, test case simplicity, flexibility, and more.

**Top test automation frameworks**

Developers and QA teams may encounter problems deciding on which test framework will work for their unique environment. To make the selection work easier, here are the top 5 Python frameworks for test automation services in 2019.

**Robot Framework**

The Robot framework is one of the most popular tools for test automation and ideal for a wide range of testers. It is based on the keyword-driven approach that makes it easy to create tests which are also readable. Also, it has a wide range of tools, libraries and APIs that make it easy to extend its features.

The Robot is an open source framework that is easy to customize and expand. The framework is ideal for testers and usually great for those familiar with the keyword-driven test tools.

Robot Framework is a cross-platform tool (Linux, MacOS, and Windows) ideal for acceptance testing. The open source framework is compatible with several other tools and libraries that make it more advanced and robust. This is one of the simplest frameworks; it supports parallel testing and has a scope for API extension.

The Robot framework is easier to use because of its many in-built libraries and is a great tool for beginners with little experience in developing applications. However, it may have limitations when running complex projects.

**Pytest**

Pytest is one of the best test automation frameworks for small and less complex projects. Most developers prefer the framework for their unit testing automation. Usually, the complete solution consists of several sub-packages and has the best test automation techniques. It has the ability to support the automation of unit, integration, end to end, and functional tests

This is an open source framework that is easy to run and use for both simple complex projects. It is suitable for development and QA teams for all types of testing; as well as individuals and groups working on open source projects. The tool is also suitable for static code analysis and is able to write effective test cases and support multiple IDE.

The framework supports parallel tests and easily integrates with other frameworks and tools. In addition, it allows running selective tests and stop when it encounters a failure in the applications. This means that the teams are able to skip some specific test cases based on certain conditions they have set. Also, the PyTest supports the running of test suites or cases with several different parameters.

Pytest is compatible with many Python versions, including the latest and other frameworks such as the Nose out, Unittest, and Doctest (usually in the out of the box state). The framework provides comprehensive reports about the failures hence enabling developers to easily and quickly identify and address the problem areas.

Currently, there are over 300 plugins for extending the features of the Pytest framework and increase its coverage. And this makes it possible to extend its functionalities further.

**Jasmine**

Jasmine is one of the best front-end testing automation frameworks. This is a behavior driven development (BDD) framework that supports Python as well as Ruby and JavaScript unit test automation. However, it focuses more on the applications business value rather than the technical details.

Jasmine allows testing for both asynchronous and DOM-less test cases. It has a simple, user-friendly and readable syntax. The clean and straightforward syntax makes it easy to write basic tests. Also, the framework allows the testers to run the server and client side test cases in parallel. By and large, this is an ideal framework that combines the server-client unit testing.

Jasmine does not rely on the web browsers, JavaScript frameworks or DOM. The tool does not have external dependencies and is usually available in a ready-to-use state. However, for it to run, it needs the Karma tests runner. Karma is a tool that allows the teams to run the Jasmine tests using the command line. It can also watch the changes in the development files and automatically re-run the tests.

The framework that focuses on behavior-driven development, supports automation of unit testing activities has a very active community for support issues.

**Unit-test**

This is an easy-to-use unit testing framework that supports reuse and organization of tests. The UnitTest framework supports sharing of the setup as well as shutdown codes. It supports important concepts such as test fixture, test case, test suite, and test runner in an object-oriented manner.

This is a flexible framework that does not require the installation of additional modules. It is also an easy to use tool that allows running individual test cases easily. In addition, the framework has some command-line capability, supports simple test discovery, gives the ability to re-use test code, and skip individual or an entire class of test methods.

The framework is suitable for people without high levels of technical skills in Python, has a concise output and the ability to generate the test reports very fast (within milliseconds).

**RedwoodHQ**

The RedwoodHQ framework works differently from the others in that it establishes a website interface that allows several people to connect to a single web location and run the tests together. This is a user-friendly framework for testers and has many features to support complete automation and management of the testing processes.

Usually, the RedwoodHQ framework enables writing tests in Python and other languages such as C#, Java, etc. The framework supports parallel testing and is compatible with continuous integration tools such as Jenkins and TeamCity. The user-friendly framework supports most of the testing processes in any project and is suitable for front-end and back-end testing.

The framework has action keywords that help teams to quickly create and modify the tests. For instance, when creating a test script, the tester only needs to find the action he or she intends to perform, after which they just need to drag it into the test case and finally add the parameters the function requires.

An inbuilt IDE makes it easier to create, modify and execute test cases. It also supports running tests in parallel. In addition, the framework keeps a history or record of all the test runs, hence making it easier to refer to previous cases.

**Choosing the best testing automation framework**

Generally, a good testing framework should have the best scripts, test cases, assumptions as well as techniques to run each module and code in order to identify the flaws and weaknesses. However, each testing project is unique and will require a different set of the above parameters.

Some of the considerations when selecting a framework include the resource it requires, functionality, reporting, integrating with third-party tools, and test driver. Also, the teams need to understand the test requirements. Ideally, it is good to break the project into smaller manageable test scenarios and then research on the frameworks to see the features and how well they meet the requirements.

The choice of the right framework will also depend on what the developer intends to spend. Generally, there are both open source and premium frameworks. Each model has its benefits and limitations.