# Introduction

Robot Framework is an **open-source Python-based** **automation framework for testing**.

The framework has a rich ecosystem with various libraries and tools developed as separate projects. For more information, see <http://robotframework.org>.

# Installation

<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#installation-instructions>

$ pip install robotframework

# Supported Tools

## VS Code Extension

### Language Server

[Robot Framework Language Server](https://marketplace.visualstudio.com/items?itemName=robocorp.robotframework-lsp) extension for VSCode.

### Code Format

By default, VSCode Robot Framework use a built-in formatter, but it's not very helpful. We prefer [**robotidy**](https://robotidy.readthedocs.io/en/stable/) instead.

You just need two extra steps:

1. From terminal:

$ pip install robotframework-tidy

1. In VSCode settings.json, add:

"robot.codeFormatter": "robotidy"

# Quick Start

## Test Approaches

There are several different ways in which test cases may be written.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Description** | **Example** | **Details** |
| **Keyword-Driven** | All tests contain a workflow constructed from keywords. | \*\*\* Test Cases \*\*\*  Push button      Push button    1      Result should be    1  Push multiple buttons      Push button    1      Push button    2      Result should be    12 | Keyword-driven and Gherkin-driven approaches are sometimes considered as workflow test.   * Generally have these phases:   + **Precondition** (optional – often in setup)   + **Action** (mandatory – do something to the system)   + **Verification** (mandatory – validate results)   + **Cleanup** (optional – always in teardown to make sure it is executed) * Different tests can have different abstraction levels.   + Tests for a detailed functionality are more precise.   + End-to-end tests can be on very high level.   + One test should use only one abstraction level * Different styles:   + More technical tests for lower level details and integration tests.   + "Executable specifications" **act as requirements**.   + Use *domain language* – Gherkin.   + Everyone should always understand. * **No complex logic** on the test case level.   + No for loops or if/else constructs.   + Use variable assignments with care.   + Test cases **should not look like scripts**! * **Max 10 steps**, preferably less. |
| **Gherkin-Driven**  **(Behavior-Driven)** | Similar to the keyword-driven, except that keywords use higher abstraction level and their arguments are embedded into the keyword names. | \*\*\* Test Cases \*\*\*  Addition      Given calculator has been cleared      When user types "1 + 1"      and user pushes equals      Then result is "2"  \*\*\* Keywords \*\*\*  Calculate      [Arguments]    ${expr}    ${expected}      Push buttons    C${expr}=      Result should be    ${expected} |
| **Data-Driven** | Tests use a keyword specified in the Test Template setting, that in turn uses other keywords.  This is very more suitable to writing testcases with same workflow repeated multiple times. | \*\*\* Test Cases \*\*\*    Expression    Expected  Addition              12 + 2 + 2    16                        2 + -3        -1  Subtraction           12 - 2 - 2    8                        2 - -3        5  Multiplication        12 \* 2 \* 2    48                        2 \* -3        -6 | * One high-level keyword per test.   + **Different arguments create different tests**.   + **One test can run the same keyword multiple times** to validate multiple related variations * If the keyword is implemented as a user keyword, it typically contains a similar workflow as workflow tests.   + Unless needed elsewhere, it is a good idea to create it in the same file as tests using it. * Recommended to use the *test template* functionality.   + No need to repeat the keyword multiple times.   + Easier to test multiple variations in one test. * Possible, and recommended, to name column headings * If a really big number of tests is needed, consider generating them based on an external model. |

Find demostration in <https://github.com/robotframework/RobotDemo> or folder *Robot Framework/RobotDemo*.

## Running Demo

To execute a specific test suite:

$ robot keyword\_driven.robot

To execute multiple test suites at once:

$ robot keyword\_driven.robot data\_driven.robot gherkin.robot

To execute all test suites in a directory recursively:

$ cd RobotDemo

$ robot .

# Robot Syntax

## File Types

### Test Suite File

These files have .robot extension. **All** [**test cases**](#_Test_Case) **of a** [**test suite**](#_Test_Suite) are defined here.

This should be the first thing to look at to know what the test does.

### Test Library File

These are.py file. And [library](#_Test_Libraries) is just a Python class with methods that create **low-level keywords** used by the test cases.

### Test Resource File

[Resource files](#_Test_Resources) are .resource files, containing **variables** and **higher-level user keywords**.

### Test Report Files

After running tests, you will **get report and log in HTML format**.

* report.html
* log.html

## Sessions

Robot Framework data is defined in different sections, often also called tables, listed below:

| **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. |

Different sections are recognized by their header row. The **recommended header format is \*\*\* Settings \*\*\***, but the header is **case-insensitive**, surrounding **spaces are optional**, and the number of asterisk characters can vary as long as there is **at least one asterisk** in the beginning. For example, also \*settings would be recognized as a section header.

## Space Rules

There are two ways:

### Space Separated Format

Keywords and their arguments are separated from each others with **two or more spaces**. Or alternatively one or more tab characters.

The recommendation is using **4 spaces** for both indentaions and separations.

**\*\*\* Settings \*\*\***

Documentation     Example using the space separated format.

Library           OperatingSystem

**\*\*\* Variables \*\*\***

${MESSAGE}        Hello, world!

**\*\*\* Test Cases \*\*\***

My Test

    [Documentation]    Example test.

    Log    ${MESSAGE}

    My Keyword    ${CURDIR}

Another Test

    Should Be Equal    ${MESSAGE}    Hello, world!

**\*\*\* Keywords \*\*\***

My Keyword

    [Arguments]    ${path}

    Directory Should Exist    ${path}

### Pipe Separated Format

Keywords and their arguments are separated from each others with a pipe character surrounded with spaces (**|**).

| \*\*\* Settings \*\*\*   |

| Documentation      | Example using the pipe separated format.

| Library            | OperatingSystem

| \*\*\* Variables \*\*\*  |

| ${MESSAGE}         | Hello, world!

| \*\*\* Test Cases \*\*\* |                 |               |

| My Test            | [Documentation] | Example test. |

|                    | Log             | ${MESSAGE}    |

|                    | My Keyword      | ${CURDIR}     |

| Another Test       | Should Be Equal | ${MESSAGE}    | Hello, world!

| \*\*\* Keywords \*\*\*   |                        |         |

| My Keyword         | [Arguments]            | ${path} |

|                    | Directory Should Exist | ${path} |

## Test Suites

### Settings

The following settings in the **\*\*\* Settings \*\*\*** section are used to customize the suite:

|  |  |
| --- | --- |
| **Setting** | **Used For** |
| [Name] | Used for setting a custom suite name. The default name is created based on the file or directory name.  Name is shown in reports and logs. |
| [Documentation] | Specify a test suite documentation. It is shown in reports and logs. |
| [Metadata] | Used for setting [free suite metadata](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#free-suite-metadata) as name-value pairs. It is shown in reports and logs. |
| [Suite Setup] | Specify test setup.  It is executed BEFORE running any of the suite's test cases or child test suites.  If it fails, all test cases in it and its child test suites are immediately assigned a fail status and ignore executed. |
| [Suite Teardown] | Specify test teardown.  It is executed AFTER running any of the suite's test cases or child test suites.  It is executed even if the setup of the same suite fails. If the suite teardown fails, all test cases in the suite are marked failed, regardless of their original execution status. |

Examples:

**\*\*\* Settings \*\*\***

Name             Custom suite name

Documentation    An example suite documentation with \*some\* \_formatting\_.

...              Long documentation can be split into multiple lines.

Metadata         Version            2.0

Metadata         Robot Framework    http://robotframework.org

Metadata         Platform           ${PLATFORM}

Metadata         Longer Value

...              Longer metadata values can be split into multiple

...              rows. Also \*simple\* \_formatting\_ is supported.

## Test Cases

### Creating Test Cases

Test cases are constructed in **\*\*\* Test Cases \*\*\*** sections from the available keywords.

The **first column** in the test case section contains **test case names**.

The **second column** normally has **keyword names**. An exception to this rule is setting variables from keyword return values, when the second and possibly also the subsequent columns contain variable names and a keyword name is located after them. In either case, columns after the keyword name contain possible arguments to the specified keyword.

**\*\*\* Test Cases \*\*\***

Valid Login

    Open Login Page

    Input Username    demo

    Input Password    mode

    Submit Credentials

    Welcome Page Should Be Open

Setting Variables

    Do Something    first argument    second argument

    ${value} =    Get Some Value

    Should Be Equal    ${value}    Expected value

### Settings

#### Local Settings

Test cases can also have their own settings. Setting names are always in the second column, where keywords normally are, and their values are in the subsequent columns. Setting names have square brackets around them to distinguish them from keywords. The available settings are listed below and explained later in this section.

|  |  |
| --- | --- |
| **Setting** | **Used For** |
| [Documentation] | Specify a test case documentation. |
| [Setup] | Specify test setup. |
| [Teardown] | Specify test teardown. |
| [Template] | Specify the template keyword to use. The test itself will contain only data to use as arguments to that keyword. |
| [Tags] | Specify tag(s) for test cases.  They're helpful for **classifying test cases** and user keywords. Thus, they help to **collect statistics**, **select test to execute**, **remove duplicates**, etc. Note that tag comparisons are case-, space- and underscore-**insensitive**.  They are **free text** and Robot itself has no special meaning for them except for the reserved tags. |
| [Timeout] | Set a test case timeout. |

Example:

**\*\*\* Test Cases \*\*\***

Test With Settings

    [Documentation]    Another dummy test

    [Tags]    dummy    owner-johndoe

    Log    Hello, world!

#### Global Settings

The **\*\*\* Settings \*\*\*** session can have the following test case related settings. These are **mainly default values** for the test case specific settings listed earlier.

|  |  |
| --- | --- |
| **Setting** | **Used For** |
| [Test Setup] | Default values for test setup. |
| [Test Teardown] | Default values for test teardown. |
| [Test Tags] | Tags all tests in the suite will get in addition to their possible own tags. |
| [Test Template] | Default template keyword to use. |
| [Test Timeout] | Default value for test case timeout. |

### Variables

Variables are an integral feature of Robot Framework, and they can be used in most places.

#### Global Variables

The most common source for variables are **\*\*\* Variables \*\*\*** section in suite files and resource files. However, note that values here are **always strings** (not [numbers](#_Number_Variables)) and they cannot be created dynamically. If either of these is a problem, [variable files](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#variable-files) can be used instead.

Example:

**\*\*\* Variables \*\*\***

${NAME}         Robot Framework

${VERSION}      2.0

${EMPTY}

${ROBOT}        ${NAME} ${VERSION}

${MULTILINE}    First line.

...             Second line.

...             Third line.

...             separator=\n

@{MANY}         one         two      three      four

&{USER}       name=Matti    address=xxx         phone=123

It is also possible, but not obligatory, to use the equals sign **=** after the variable name to make assigning variables slightly more explicit.:

\*\*\* Variables \*\*\*

${NAME} =       Robot Framework

${VERSION} =    2.0

#### Local Variables

**Return values from keywords** can also be set into variables. This allows communication between different keywords.

Any value returned by a keyword can be assigned to a scalar variable. Having the equals sign **=** after the variable name is not obligatory, but it makes the assignment more explicit:

**\*\*\* Test Cases \*\*\***

Returning

    ${x} =    Get X    an argument

    Log    We got ${x}!

Although a value is assigned to a scalar variable, it can be used as a list variable if it has a list-like value and as a dictionary variable if it has a dictionary-like value:

**\*\*\* Test Cases \*\*\***

Example

    ${list} =    Create List    first    second    third

    Length Should Be    ${list}    3

    Log Many    @{list}

#### Data Types

|  |  |
| --- | --- |
| **Type** | **Syntax** |
| Scalar | ${SCALAR} |
| List | @{LIST} |
| Dictionary | &{DICT} |
| Environment Variable | %{ENV\_VAR} |

##### Scalar

###### String

**\*\*\* Variables \*\*\***

${GREET}         Hello

${NAME}       World

**\*\*\* Test Cases \*\*\***

Constants

    Log    Hello

    Log    Hello, world!!

Variables

    Log    ${GREET}                # Hello

    Log    ${GREET}, ${NAME}!!     # Hello, World!!

When a scalar variable is used alone, like in ${GREET} above, it is replaced with its value as-is and the value can be any object.

If the variable is not used alone, but having other text or variables around, like ${GREET}, ${NAME}!! above, its value is first converted into a string and then concatenated with the other data.

###### Number

**\*\*\* Test Cases \*\*\***

Example 1A

    Connect    example.com    80       # Connect gets two strings as arguments

Example 1B

    Connect    example.com    ${80}    # Connect gets a string and an integer

Example 2

    Do X    ${3.14}    ${-1e-4}        # Do X gets floating point numbers 3.14 and -0.0001

It is possible to create also from binary, octal, and hexadecimal values using 0b, 0o and 0x prefixes, respectively:

**\*\*\* Test Cases \*\*\***

Example

    Should Be Equal    ${0b1011}    ${11}

    Should Be Equal    ${0o10}      ${8}

    Should Be Equal    ${0xff}      ${255}

    Should Be Equal    ${0B1010}    ${0XA}

###### Boolean

**\*\*\* Test Cases \*\*\***

Boolean

    Set Status    ${true}               # Set Status gets boolean "true" as an argument

    Create Y    something   ${false}    # Create Y gets a string and boolean "false"

###### None/null

**\*\*\* Test Cases \*\*\***

None

    Do XYZ    ${None}                   # Do XYZ gets Python None as an argument

##### List

The following test cases, Constants and List Variables, are equivalent:

**\*\*\* Variables \*\*\***

@{USER} robot secret

**\*\*\* Test Cases \*\*\***

Constants

    Login    robot    secret

List Variable

    Login    @{USER}

Also, *list expansion* can be used in combination with [list item access](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#accessing-sequence-items) making these usages possible:

**\*\*\* Test Cases \*\*\***

Nested container

    ${nested} =    Evaluate    [['a', 'b', 'c'], {'key': ['x', 'y']}]

    Log Many    @{nested}[0]         # Logs 'a', 'b' and 'c'.

    Log Many    @{nested}[1][key]    # Logs 'x' and 'y'.

Slice

    ${items} =    Create List    first    second    third

    Log Many    @{items}[1:]         # Logs 'second' and  'third'.

##### Dict

The following test cases, Constants and Dict Variable, are equivalent:

**\*\*\* Variables \*\*\***

&{USER} name=robot password=secret

**\*\*\* Test Cases \*\*\***

Constants

    Login    name=robot    password=secret

Dict Variable

    Login    &{USER}

A certain value of a dictionary variable can be accessed with the syntax ${NAME}[key], where key is the name of the selected value. Keys are considered to be strings, but non-strings keys can be used as variables. If a key is a string, it is possible to access its value also using attribute access syntax ${NAME.key}.

**\*\*\* Variables \*\*\***

&{USER} name=robot password=secret

**\*\*\* Test Cases \*\*\***

Dictionary variable item

    Login    ${USER}[name]    ${USER}[password]

    Title Should Be    Welcome ${USER}[name]!

Key defined as variable

    Log Many    ${DICT}[${KEY}]    ${DICT}[${42}]

Attribute access

    Login    ${USER.name}    ${USER.password}

    Title Should Be    Welcome ${USER.name}!

##### Env Var

Env vars are limited to string values. It is possible to specify a default value, that is used if the env var does not exists, by separating the variable name and the default value with an equal sign like %{ENV\_VAR\_NAME=default value}.

Env vars, which is set in the OS before the test execution, are available during it, and it's possible to create new ones with the keyword Set Environment Variable or delete existing ones with the keyword Delete Environment Variable, both available in the OperatingSystem library. Because env vars are global, ones set in one test case can be used in other test cases executed after it. However, changes to env vars are not effective after the test execution.

#### Changing Variable Scope Dynamically

Way 1: <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#toc-entry-332>

Way 2 (recommended): <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#toc-entry-331>

#### Built-In Variables

Robot Framework provides some built-in variables that are available automatically.

##### OS

Built-in variables related to the operating system ease making the test data operating-system-agnostic.

| **Variable** | **Explanation** |
| --- | --- |
| ${CURDIR} | Absolute path to the directory where the test data file is located. This variable is case-sensitive. |
| ${TEMPDIR} | Absolute path to the system temporary directory.  In UNIX-like systems this is typically */tmp*, and in Windows *c:\Documents and Settings\<user>\Local Settings\Temp*. |
| ${EXECDIR} | Absolute path to the directory where test execution was started from. |
| ${/} | System directory path separator. / in UNIX-like systems and \ in Windows. |
| ${:} | System path element separator. : in UNIX-like systems and ; in Windows. |
| ${\n} | System line separator. \n in UNIX-like systems and \r\n in Windows. |

##### Testcase

| **Variable** | **Explanation** | **Available** |
| --- | --- | --- |
| ${TEST NAME} | The name of the current test case. | Test case |
| @{TEST TAGS} | Contains the tags of the current test case in alphabetical order. Can be modified dynamically using *Set Tags* and *Remove Tags* keywords. | Test case |
| ${TEST DOCUMENTATION} | The documentation of the current test case. Can be set dynamically using using *Set Test Documentation* keyword. | Test case |
| ${TEST STATUS} | The status of the current test case, either PASS or FAIL. | [Test teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#test-setup-and-teardown) |
| ${TEST MESSAGE} | The message of the current test case. | [Test teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#test-setup-and-teardown) |
| ${PREV TEST NAME} | The name of the previous test case, or an empty string if no tests have been executed yet. | Everywhere |
| ${PREV TEST STATUS} | The status of the previous test case: either PASS, FAIL, or an empty string when no tests have been executed. | Everywhere |
| ${PREV TEST MESSAGE} | The possible error message of the previous test case. | Everywhere |
| ${SUITE NAME} | The full name of the current test suite. | Everywhere |
| ${SUITE SOURCE} | An absolute path to the suite file or directory. | Everywhere |
| ${SUITE DOCUMENTATION} | The documentation of the current test suite. Can be set dynamically using using *Set Suite Documentation* keyword. | Everywhere |
| &{SUITE METADATA} | The free metadata of the current test suite. Can be set using *Set Suite Metadata* keyword. | Everywhere |
| ${SUITE STATUS} | The status of the current test suite, either PASS or FAIL. | [Suite teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#suite-setup-and-teardown) |
| ${SUITE MESSAGE} | The full message of the current test suite, including statistics. | [Suite teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#suite-setup-and-teardown) |
| ${KEYWORD STATUS} | The status of the current keyword, either PASS or FAIL. | [User keyword teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#user-keyword-teardown) |
| ${KEYWORD MESSAGE} | The possible error message of the current keyword. | [User keyword teardown](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#user-keyword-teardown) |
| ${LOG LEVEL} | Current [log level](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#log-levels). | Everywhere |
| ${OUTPUT DIR} | An absolute path to the [output directory](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#output-directory) as a string. | Everywhere |
| ${OUTPUT FILE} | An absolute path to the [output file](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#output-file) as a string or a string NONE if the output file is not created. | Everywhere |
| ${LOG FILE} | An absolute path to the [log file](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#log-file) as a string or a string NONE if the log file is not created. | Everywhere |
| ${REPORT FILE} | An absolute path to the [report file](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#report-file) as a string or a string NONE if the report file is not created. | Everywhere |
| ${DEBUG FILE} | An absolute path to the [debug file](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#debug-file) as a string or a string NONE if the debug file is not created. | Everywhere |
| &{OPTIONS} | A dictionary exposing command line options. The dictionary keys match the command line options and can be accessed both like ${OPTIONS}[key] and ${OPTIONS.key}. Available options:   * ${OPTIONS.exclude} (--exclude) * ${OPTIONS.include} (--include) * ${OPTIONS.skip} (--skip) * ${OPTIONS.skip\_on\_failure} (--skip-on-failure) * ${OPTIONS.console\_width} (--console-width)   ${OPTIONS} itself was added in RF 5.0 and ${OPTIONS.console\_width} in RF 7.1. More options can be exposed later. | Everyw |

### Keywords

#### Creating and Calling Keywords

Keywords are created in the **\*\*\* Keywords \*\*\*** section in suite files and resource files.

Also user keywords are created from keywords, either from library keywords or other user keywords.

In this example, we have 5 keywords: Open Login Page, Open Browser, Title Should Start With, Get Title, and Should Start With.

**\*\*\* Keywords \*\*\***

Open Login Page

    Open Browser    http://host/login.html

Title Should Start With

    [Arguments]    ${expected}

    ${title} =    Get Title

    Should Start With    ${title}    ${expected}

Keywords can have zero or more arguments, and some arguments may have default values. It is also possible that a keyword accepts any number of arguments.

**\*\*\* Test Cases \*\*\***

Example

    No Operation

    Create Directory    ${TEMPDIR}/stuff

    Copy File    ${CURDIR}/file.txt    ${TEMPDIR}/stuff

    Create File    ${TEMPDIR}/empty.txt

    Create File    ${TEMPDIR}/utf-8.txt         content=Here content is not empty

    Create File    ${TEMPDIR}/iso-8859-1.txt    Here content is not empty with encoding    encoding=ISO-8859-1

Remove Files ${TEMPDIR}/f1.txt ${TEMPDIR}/f2.txt ${TEMPDIR}/f3.txt

Note: Above keywords are built-in in [OperatingSystem](https://robotframework.org/robotframework/latest/libraries/OperatingSystem.html) library.

#### Settings

Keywords can have settings like test cases:

|  |  |
| --- | --- |
| **Setting** | **Used For** |
| [Documentation] | Specify a keyword documentation. |
| [Setup] | Specify test setup.  Unlike tes tcases, it is not possible to specify a common setup to all keywords in a certain file. |
| [Teardown] | Specify test teardown.  Unlike tes tcases, it is not possible to specify a common teardown to all keywords in a certain file. |
| [Arguments] | Specify the keyword arguments. |
| [Return] | Specifies keyword return values  Deprecated in Robot Framework 7.0, the RETURN statement should be used instead. |
| [Tags] | Tag keyword. |
| [Timeout] | Set a keyword timeout. |

Example:

**\*\*\* Keywords \*\*\***

One Argument

    [Documentation]    This is a keyword with one argument

    [Arguments]    ${arg\_name}

    Log    Got argument ${arg\_name}

Three Arguments

    [Tags]  Example

    [Arguments]    ${arg1}    ${arg2}    ${arg3}

    Log    1st argument: ${arg1}

    Log    2nd argument: ${arg2}

    Log    3rd argument: ${arg3}

One Argument With Default Value

    [Arguments]    ${arg}=default value

    Log    Got argument ${arg}

One Required And One With Default

    [Arguments]    ${required}    ${optional}=default

    Log    Required: ${required}

    Log    Optional: ${optional}

Return One Value

    [Arguments]    ${arg}

    [Documentation]    Return a value unconditionally.

    ...                Notice that keywords after RETURN are not executed.

    ${value} =    Convert To Upper Case    ${arg}

    RETURN    ${value}

    Fail    Not executed

Return Three Values

    RETURN    a    b    c

Return One Value (Old style)

    [Arguments]    ${arg}

    ${value} =    Convert To Upper Case    ${arg}

    [Return]    ${value}

Setup and teardown

    [Setup]       Log    New in RF 7!

    Do Something

    [Teardown]    Log    Old feature.

Using variables

    [Setup]       ${SETUP}

    Do Something

    [Teardown]    ${TEARDOWN}

### Test Case Failures

A test case **fails if any of the keyword it uses fails**. Normally this means that execution of that test case is stopped, possible test teardown is executed, and then execution continues from the next test case. It is also possible to use special [continuable failures](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#continue-on-failure) if stopping test execution is not desired.

### Test Setup and Teardown

A test setup is executed before a test case, and a test teardown is executed after a test case. In Robot Framework, setups and teardowns are just normal keywords with possible arguments.

**\*\*\* Settings \*\*\***

Test Setup       Open Application    App A

Test Teardown    Close Application

**\*\*\* Test Cases \*\*\***

Default values

    [Documentation]    Setup and teardown from setting section

    Do Something

Overridden setup

    [Documentation]    Own setup, teardown from setting section

    [Setup]    Open Application    App B

    Do Something

No teardown

    [Documentation]    Default setup, no teardown at all

    Do Something

    [Teardown]

No teardown 2

    [Documentation]    Setup and teardown can be disabled also with special value NONE

    Do Something

    [Teardown]    NONE

Using variables

    [Documentation]    Setup and teardown specified using variables

    [Setup]    ${SETUP}

    Do Something

    [Teardown]    ${TEARDOWN}

### Test Template

Test templates **convert normal keyword-driven test cases into data-driven tests**. Whereas the body of a keyword-driven test case is constructed from keywords and their possible arguments, test cases with template contain only the arguments for the template keyword. Instead of repeating the same keyword multiple times per test and/or with all tests in a file, it is possible to use it only per test or just once per file.

#### Simple Example

Specify the template for an individual test case using the [Template] setting.

Below two tests are functionally fully identical:

**\*\*\* Test Cases \*\*\***

Normal test case

    Example keyword    first argument    second argument

Templated test case

    [Template]    Example keyword

    first argument    second argument

#### More Complex Example

The below example has six separate tests, one for each invalid user/password combination, and the example below **illustrates how to have only one test with all the combinations**. This is where we can see the benefits of using test template:

**\*\*\* Settings \*\*\***

Test Template    Login with invalid credentials should fail

**\*\*\* Test Cases \*\*\***                USERNAME         PASSWORD

Invalid User Name                 invalid          ${VALID PASSWORD}

Invalid Password                  ${VALID USER}    invalid

Invalid User Name and Password    invalid          invalid

Empty User Name                   ${EMPTY}         ${VALID PASSWORD}

Empty Password                    ${VALID USER}    ${EMPTY}

Empty User Name and Password      ${EMPTY}         ${EMPTY}

## Test Libraries

### Importing Test Libraries

Test libraries are imported using the Library setting in the \*\*\* Setting \*\*\* section and having the library name in the subsequent column. Unlike most of the other data, the library name is both **case- and space-sensitive**. If a library is in a package, the **full name including the package name** must be used.

It's **possible for a library to have arguments**. Both the library name and arguments can be set using variables.

**Test libraries can be imported in suite files or resource files**. All the keywords in the imported library are available in that file. With resource files, they are also available in other files using them.

Example 1: Library is a directory

**\*\*\* Settings \*\*\***

Library    OperatingSystem

Library    my.package.TestLibrary

Library    MyLibrary    arg1    arg2

Library    ${LIBRARY}

Example 2: Library is a file. The path is considered relative to the directory where current test data file is situated similarly as paths to resource and variable files.

**\*\*\* Settings \*\*\***

Library    PythonLibrary.py

Library    relative/path/PythonDirLib/    arg1    arg2

Library    ${RESOURCES}/Example.class

Example 3: Setting custom name to library. The name is shown in test logs before keyword names.

**\*\*\* Settings \*\*\***

Library    SomeLibrary    localhost        1234    AS    LocalLib

Library    SomeLibrary    server.domain    8080    AS    RemoteLib

### Standard Libraries / Built-in Libraries

#### Normal Standard Libraries

The available normal standard libraries are listed below with links to their documentations:

* [BuiltIn](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html)
* [Call Method](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Call%20Method)
* [Catenate](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Catenate)
* [Comment](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Comment)
* [Continue For Loop](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Continue%20For%20Loop)
* [Continue For Loop If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Continue%20For%20Loop%20If)
* [Convert To Binary](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Binary)
* [Convert To Boolean](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Boolean)
* [Convert To Bytes](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Bytes)
* [Convert To Hex](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Hex)
* [Convert To Integer](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Integer)
* [Convert To Number](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Number)
* [Convert To Octal](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20Octal)
* [Convert To String](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Convert%20To%20String)
* [Create Dictionary](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Create%20Dictionary)
* [Create List](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Create%20List)
* [Evaluate](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Evaluate)
* [Exit For Loop](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Exit%20For%20Loop)
* [Exit For Loop If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Exit%20For%20Loop%20If)
* [Fail](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Fail)
* [Fatal Error](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Fatal%20Error)
* [Get Count](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Count)
* [Get Length](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Length)
* [Get Library Instance](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Library%20Instance)
* [Get Time](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Time)
* [Get Variable Value](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Variable%20Value)
* [Get Variables](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Get%20Variables)
* [Import Library](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Import%20Library)
* [Import Resource](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Import%20Resource)
* [Import Variables](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Import%20Variables)
* [Keyword Should Exist](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Keyword%20Should%20Exist)
* [Length Should Be](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Length%20Should%20Be)
* [Log](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Log)
* [Log Many](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Log%20Many)
* [Log To Console](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Log%20To%20Console)
* [Log Variables](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Log%20Variables)
* [No Operation](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#No%20Operation)
* [Pass Execution](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Pass%20Execution)
* [Pass Execution If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Pass%20Execution%20If)
* [Regexp Escape](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Regexp%20Escape)
* [Reload Library](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Reload%20Library)
* [Remove Tags](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Remove%20Tags)
* [Repeat Keyword](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Repeat%20Keyword)
* [Replace Variables](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Replace%20Variables)
* [Reset Log Level](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Reset%20Log%20Level)
* [Return From Keyword](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Return%20From%20Keyword)
* [Return From Keyword If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Return%20From%20Keyword%20If)
* [Run Keyword](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword)
* [Run Keyword And Continue On Failure](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Continue%20On%20Failure)
* [Run Keyword And Expect Error](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Expect%20Error)
* [Run Keyword And Ignore Error](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Ignore%20Error)
* [Run Keyword And Return](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Return)
* [Run Keyword And Return If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Return%20If)
* [Run Keyword And Return Status](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Return%20Status)
* [Run Keyword And Warn On Failure](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20And%20Warn%20On%20Failure)
* [Run Keyword If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If)
* [Run Keyword If All Tests Passed](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If%20All%20Tests%20Passed)
* [Run Keyword If Any Tests Failed](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If%20Any%20Tests%20Failed)
* [Run Keyword If Test Failed](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If%20Test%20Failed)
* [Run Keyword If Test Passed](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If%20Test%20Passed)
* [Run Keyword If Timeout Occurred](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20If%20Timeout%20Occurred)
* [Run Keyword Unless](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keyword%20Unless)
* [Run Keywords](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Run%20Keywords)
* [Set Global Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Global%20Variable)
* [Set Library Search Order](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Library%20Search%20Order)
* [Set Local Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Local%20Variable)
* [Set Log Level](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Log%20Level)
* [Set Suite Documentation](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Suite%20Documentation)
* [Set Suite Metadata](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Suite%20Metadata)
* [Set Suite Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Suite%20Variable)
* [Set Tags](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Tags)
* [Set Task Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Task%20Variable)
* [Set Test Documentation](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Test%20Documentation)
* [Set Test Message](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Test%20Message)
* [Set Test Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Test%20Variable)
* [Set Variable](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Variable)
* [Set Variable If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Set%20Variable%20If)
* [Should Be Empty](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20Empty)
* [Should Be Equal](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20Equal)
* [Should Be Equal As Integers](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20Equal%20As%20Integers)
* [Should Be Equal As Numbers](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20Equal%20As%20Numbers)
* [Should Be Equal As Strings](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20Equal%20As%20Strings)
* [Should Be True](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be%20True)
* [Should Contain](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Contain)
* [Should Contain Any](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Contain%20Any)
* [Should Contain X Times](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Contain%20X%20Times)
* [Should End With](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20End%20With)
* [Should Match](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Match)
* [Should Match Regexp](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Match%20Regexp)
* [Should Not Be Empty](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20Empty)
* [Should Not Be Equal](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20Equal)
* [Should Not Be Equal As Integers](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20Equal%20As%20Integers)
* [Should Not Be Equal As Numbers](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20Equal%20As%20Numbers)
* [Should Not Be Equal As Strings](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20Equal%20As%20Strings)
* [Should Not Be True](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Be%20True)
* [Should Not Contain](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Contain)
* [Should Not Contain Any](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Contain%20Any)
* [Should Not End With](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20End%20With)
* [Should Not Match](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Match)
* [Should Not Match Regexp](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Match%20Regexp)
* [Should Not Start With](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Not%20Start%20With)
* [Should Start With](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Start%20With)
* [Skip](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Skip)
* [Skip If](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Skip%20If)
* [Sleep](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Sleep)
* [Variable Should Exist](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Variable%20Should%20Exist)
* [Variable Should Not Exist](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Variable%20Should%20Not%20Exist)
* [Wait Until Keyword Succeeds](https://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Wait%20Until%20Keyword%20Succeeds)
* [Collections](https://robotframework.org/robotframework/latest/libraries/Collections.html)
* [DateTime](https://robotframework.org/robotframework/latest/libraries/DateTime.html)
* [Dialogs](https://robotframework.org/robotframework/latest/libraries/Dialogs.html)
* [OperatingSystem](https://robotframework.org/robotframework/latest/libraries/OperatingSystem.html)
* [Process](https://robotframework.org/robotframework/latest/libraries/Process.html)
* [Screenshot](https://robotframework.org/robotframework/latest/libraries/Screenshot.html)
* [String](https://robotframework.org/robotframework/latest/libraries/String.html)
* [Telnet](https://robotframework.org/robotframework/latest/libraries/Telnet.html)
* [XML](https://robotframework.org/robotframework/latest/libraries/XML.html)

#### Remote library

In addition to the normal standard libraries listed above, there is also Remote library that is totally different than the other standard libraries. It does not have any keywords of its own but it works as a proxy between Robot Framework and actual test library implementations. These libraries can be running on other machines than the core framework and can even be implemented using languages not supported by Robot Framework natively.

### External libraries

Any test library that is **not one of the standard libraries** is an external library. The Robot Framework open source community has implemented several generic libraries, such as SeleniumLibrary and SwingLibrary, which are not packaged with the core framework. A list of publicly available libraries can be found from <http://robotframework.org>.

## Test Resources

Resource files are imported using the Resource setting in the \*\*\* Settings \*\*\* section so that the path to the resource file is given as an argument to the setting.

If the resource file path is absolute, it is used directly. Otherwise, the resource file is first searched relatively to the directory where the importing file is located. If the file is not found there, it is then searched from the directories in Python's module search path.

Example:

**\*\*\* Settings \*\*\***

Documentation     An example resource file

Library           SeleniumLibrary

Resource          ${RESOURCES}/common.resource

**\*\*\* Variables \*\*\***

${HOST}           localhost:7272

${LOGIN URL}      http://${HOST}/

${WELCOME URL}    http://${HOST}/welcome.html

${BROWSER}        Firefox

**\*\*\* Keywords \*\*\***

Open Login Page

    [Documentation]    Opens browser to login page

    Open Browser    ${LOGIN URL}    ${BROWSER}

    Title Should Be    Login Page

Input Name

    [Arguments]    ${name}

    Input Text    username\_field    ${name}

Input Password

    [Arguments]    ${password}

    Input Text    password\_field    ${password}

## Control Structures

Various structures can be used to **control the test execution flow**. These are **familiar** from most programming languages as they allow conditional execution, repeatedly executing a block of keywords and fine-grained error handling.

For readability reasons these structures should be used judiciously, and more complex use cases should be preferably implemented in test libraries.

### Loops

#### FOR Loops

**\*\*\* Test Cases \*\*\***

Simple for loop

    FOR    ${animal}    IN    cat    dog

        Log    ${animal}

        Log    2nd keyword

    END

    Log    Outside loop

Another simple for loop

    FOR    ${var}    IN    one    two    ${3}    four    ${five}

    ...    kuusi    7    eight    nine    ${last}

        Log    ${var}

    END

Nested for loops

    [Arguments]    @{table}

    FOR    ${row}    IN    @{table}

        FOR    ${cell}    IN    @{row}

            Handle Cell    ${cell}

        END

    END

#### FOR-IN-RANGE Loop

**\*\*\* Test Cases \*\*\***

Only upper limit

    [Documentation]    Loops over values from 0 to 9.

    FOR    ${index}    IN RANGE    10

        Log    ${index}

    END

Start and end

    [Documentation]    Loops over values from 1 to 10.

    FOR    ${index}    IN RANGE    1    11

        Log    ${index}

    END

Also step given

    [Documentation]    Loops over values 5, 15, and 25.

    FOR    ${index}    IN RANGE    5    26    10

        Log    ${index}

    END

Negative step

    [Documentation]    Loops over values 13, 3, and -7.

    FOR    ${index}    IN RANGE    13    -13    -10

        Log    ${index}

    END

Arithmetic

    [Documentation]    Arithmetic with variable.

    FOR    ${index}    IN RANGE    ${var} + 1

        Log    ${index}

    END

Float parameters

    [Documentation]    Loops over values 3.14, 4.34, and 5.54.

    FOR    ${index}    IN RANGE    3.14    6.09    1.2

        Log    ${index}

    END

#### WHILE Loops

**\*\*\* Test Cases \*\*\***

Example

    VAR    ${rc}   1

    WHILE    ${rc} != 0

        ${rc} =    Keyword that returns zero on success

    END

Limit as iteration count

    WHILE    True    limit=100

        Log    This is run 100 times.

    END

    WHILE    True    limit=10 times

        Log    This is run 10 times.

    END

    WHILE    True    limit=42x

        Log    This is run 42 times.

    END

Limit as time

    WHILE    True    limit=10 seconds

        Log    This is run 10 seconds.

    END

No limit

    WHILE    True    limit=NONE

        Log    This runs forever.

    END

#### BREAK and CONTINUE in Loops

**\*\*\* Test Cases \*\*\***

BREAK with FOR

    ${text} =    Set Variable    zero

    FOR    ${var}    IN    one    two    three

        IF    '${var}' == 'two'    BREAK

        ${text} =    Set Variable    ${text}-${var}

    END

    Should Be Equal    ${text}    zero-one

CONTINUE with FOR

    ${text} =    Set Variable    zero

    FOR    ${var}    IN    one    two    three

        IF    '${var}' == 'two'    CONTINUE

        ${text} =    Set Variable    ${text}-${var}

    END

    Should Be Equal    ${text}    zero-one-three

### IF/ELSE Condition

**\*\*\* Test Cases \*\*\***

IF

   IF    ${rc} > 0

       Some keyword

       Another keyword

   END

IF ELSE

    IF    ${rc} > 0

        Some keyword

    ELSE

        Another keyword

    END

IF ELSE IF ELSE

    IF    $rc > 0

        Positive keyword

    ELSE IF    $rc < 0

        Negative keyword

    ELSE IF    $rc == 0

        Zero keyword

    ELSE

        Fail    Unexpected rc: ${rc}

    END

Inline IF

    IF    $condition1    Keyword    argument

    IF    $condition2    RETURN

Inline IF/ELSE

    IF    $condition    Keyword    argument    ELSE    Another Keyword

Inline IF/ELSE IF/ELSE

    IF    $cond1    Keyword 1    ELSE IF    $cond2    Keyword 2    ELSE IF    $cond3    Keyword 3    ELSE    Keyword 4

Nested IF/ELSE

    [Arguments]    @{items}    ${log\_values}=True

    IF    not ${items}

        Log to console    No items.

    ELSE IF    len(${items}) == 1

        IF    ${log\_values}

            Log to console    One item: ${items}[0]

        ELSE

            Log to console    One item.

        END

    ELSE

        Log to console    ${{len(${items})}} items.

        IF    ${log\_values}

            FOR    ${index}    ${item}    IN ENUMERATE    @{items}    start=1

                Log to console    Item ${index}: ${item}

            END

        END

    END

### TRY/EXCEPT Syntax

**\*\*\* Test Cases \*\*\***

First example

    TRY

        Some Keyword

    EXCEPT    Error message

        Error Handler Keyword

    END

    Keyword Outside

Multiple EXCEPT branches

    TRY

        Some Keyword

    EXCEPT    Error message    # Try matching this first.

        Error Handler 1

    EXCEPT    Another error    # Try this if above did not match.

        Error Handler 2

    EXCEPT    ${message}       # Last match attempt, this time using a variable.

        Error Handler 3

    END

Multiple messages with one EXCEPT

    TRY

        Some Keyword

    EXCEPT    Error message    Another error    ${message}    # Match any of these.

        Error handler

    END

Match any error

    TRY

        Some Keyword

    EXCEPT               # Match any error.

        Error Handler

    END

Match any after testing more specific errors

    TRY

        Some Keyword

    EXCEPT    Error message    # Try matching this first

        Error Handler 1

    EXCEPT                     # Match any that did not match the above.

        Error Handler 2

    END

TRY/EXCEPT/ELSE/FINALLY

    TRY

        Some keyword

    EXCEPT

        Log    Error occurred!

    ELSE

        Log    No error occurred.

    FINALLY

        Log    Always executed.

    END

TRY/FINALLY

    Open Connection

    TRY

        Use Connection

    FINALLY

        Close Connection

    END

# Executing Test Cases

<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#executing-test-cases>

# Custom Test Libraries

<https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#extending-robot-framework>

# Robot Best Practices

## Most Important Goals When Writing Test Cases

* Easy to understand
* Easy to maintain
* Fast to execute

Details:

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

<https://www.slideshare.net/slideshow/robot-framework-dos-and-donts/38564910#4>