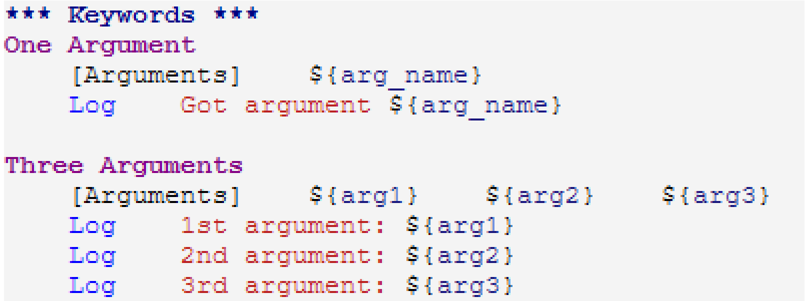
Arguments

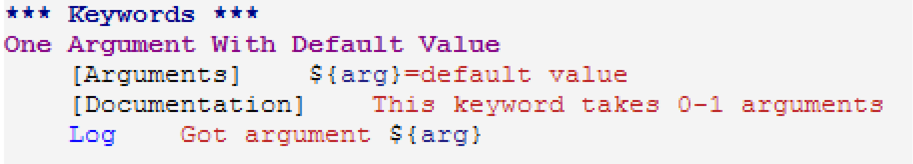
- Positional arguments
  - Arguments passed to keyword are assigned to variables defined in the Arguments setting in according to their position



```
*** Keywords ***
One Argument
    [Arguments]    ${arg_name}
    Log    Got argument ${arg_name}

Three Arguments
    [Arguments]    ${arg1}    ${arg2}    ${arg3}
    Log    1st argument: ${arg1}
    Log    2nd argument: ${arg2}
    Log    3rd argument: ${arg3}
```

- Arguments can have default values
  - Default argument syntax is space sensitive

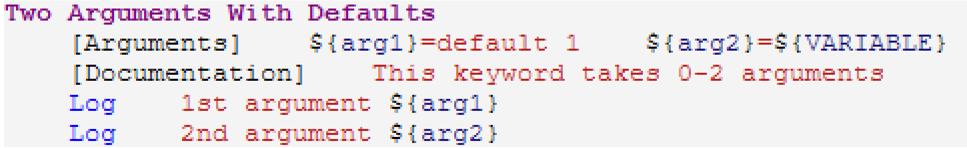


```
*** Keywords ***
One Argument With Default Value
    [Arguments]    ${arg}=default value
    [Documentation]    This keyword takes 0-1 arguments
    Log    Got argument ${arg}
```

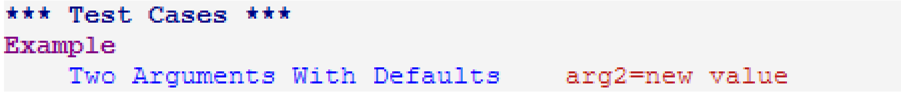
  - Default arguments must come after the positional arguments

Named arguments

- If there are multiple arguments with default values and only one of the values needs to be overridden the argument can be named



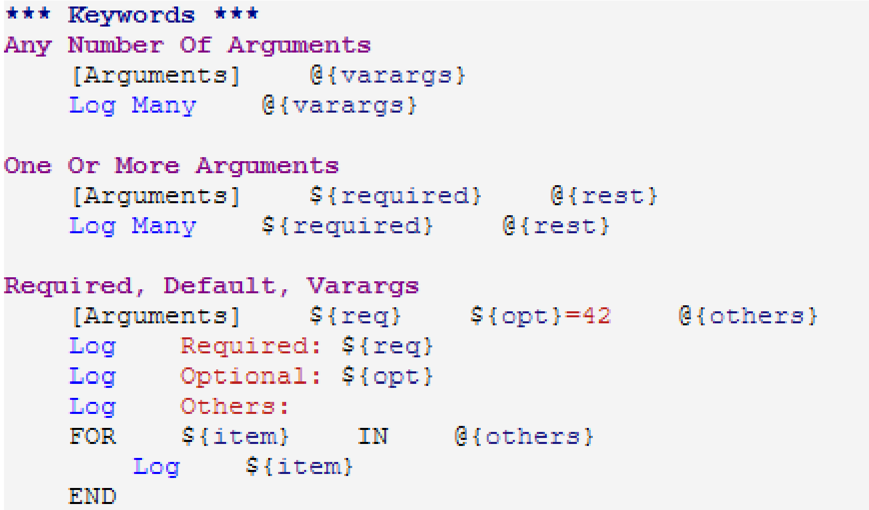
```
Two Arguments With Defaults
    [Arguments]    ${arg1}=default 1    ${arg2}=${VARIABLE}
    [Documentation]    This keyword takes 0-2 arguments
    Log    1st argument ${arg1}
    Log    2nd argument ${arg2}
```



```
*** Test Cases ***
Example
    Two Arguments With Defaults    arg2=new value
```

Variable number of arguments

- Variable number of arguments is enabled simply by specifying a list variable as an argument
    - The variable argument must be the last (or only) argument of the keyword



```
*** Keywords ***
Any Number Of Arguments
    [Arguments]    @{varargs}
    Log Many    @{varargs}

One Or More Arguments
    [Arguments]    ${required}    @{rest}
    Log Many    ${required}    @{rest}

Required, Default, Varargs
    [Arguments]    ${req}    ${opt}=42    @{others}
    Log    Required: ${req}
    Log    Optional: ${opt}
    Log    Others:
    FOR    ${item}    IN    @{others}
        Log    ${item}
    END
```

List expansion

- If you pass a list variable as an argument to a keyword the list is expanded in to individual arguments
- If variable users contains a list: [root, admin, dialout, guest]
  Then passing it to keyword expands the list

  ```
  My Special Keyword    @{users}
  ```

  is same as

  ```
  My Special Keyword    root    admin    dialout    guest
  ```

- You can prevent list expansion and pass a list as a single parameter by using it like a scalar variable

  ```
  My Special Keyword    ${users}
  ```

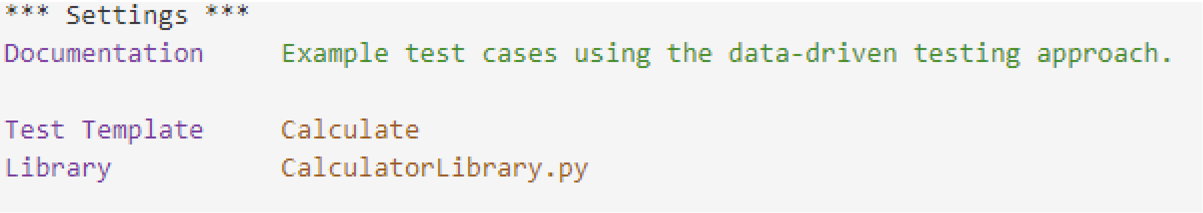
  is same as

  ```
  My Special Keyword    [root, admin, dialout, guest]
  ```

  Single argument

Data-driven style

- Data-driven style is used when same test needs to be executed multiple times with different input and/or output data
- The test to be executed is typically a user user keyword that defines the workflow of an individual test
- A test template is used to define the test to execute and the test case(s) define only the argument that are passed to the actual test
- There are two ways to specify the test template
  - In the *Settings* section



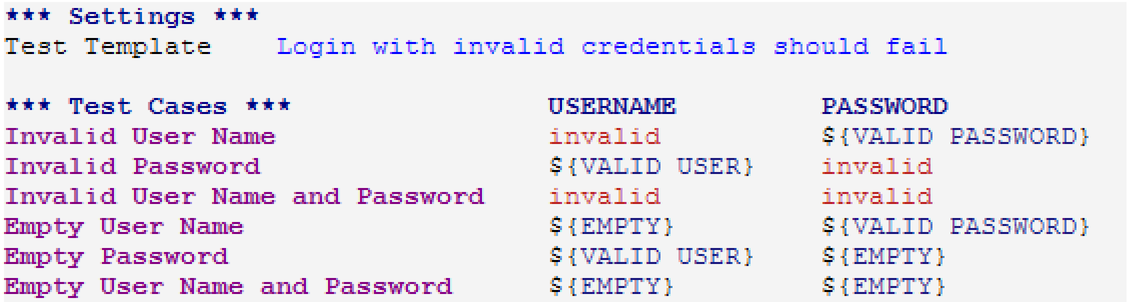
```
*** Settings ***
Documentation     Example test cases using the data-driven testing approach.

Test Template     Calculate
Library           CalculatorLibrary.py
```

  - In a test case (discussed on the following slides)

Template

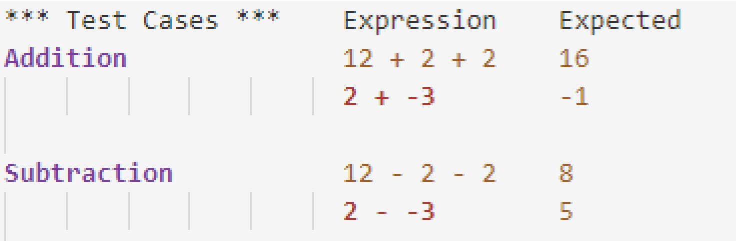
- Test template simply specifies that a test repeats the same keyword for each line in the test case
  - When template from *Settings* is applied the test data follows immediately after the test case name



```
*** 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}
```

  - Is also possible to group multiple test cases under the same name



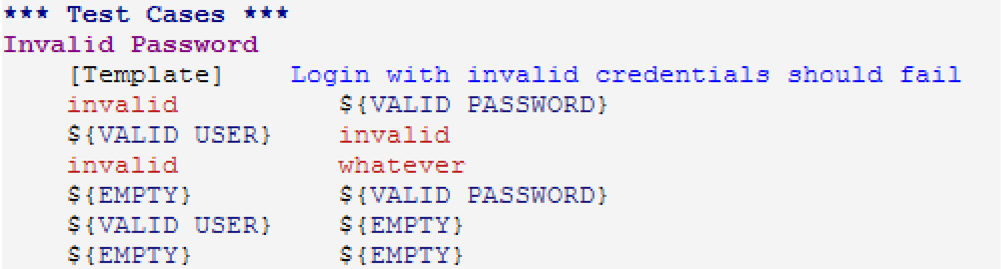
```
*** Test Cases ***     Expression     Expected
Addition               12 + 2 + 2     16
                       2 + -3         -1

Subtraction            12 - 2 - 2     8
                       2 - -3         5
```

  - Note that all cells after the section header name are ignored so we can use them to document the arguments passed to the template

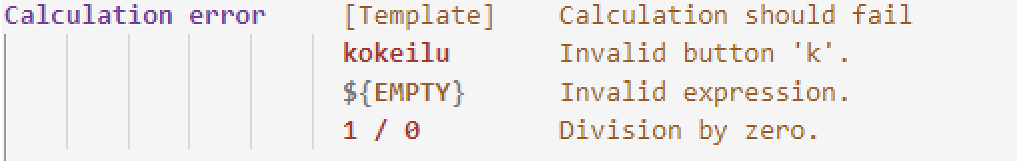
Template

- Template can also be specified locally inside a test case



```
*** Test Cases ***
Invalid Password
    [Template]    Login with invalid credentials should fail
    invalid           ${VALID PASSWORD}
    ${VALID USER}     invalid
    invalid           whatever
    ${EMPTY}          ${VALID PASSWORD}
    ${VALID USER}     ${EMPTY}
    ${EMPTY}          ${EMPTY}
```

- Or we can override a template specified in the Settings section



```
Calculation error      [Template]    Calculation should fail
                       kokeilu       Invalid button 'k'.
                       ${EMPTY}      Invalid expression.
                       1 / 0         Division by zero.
```