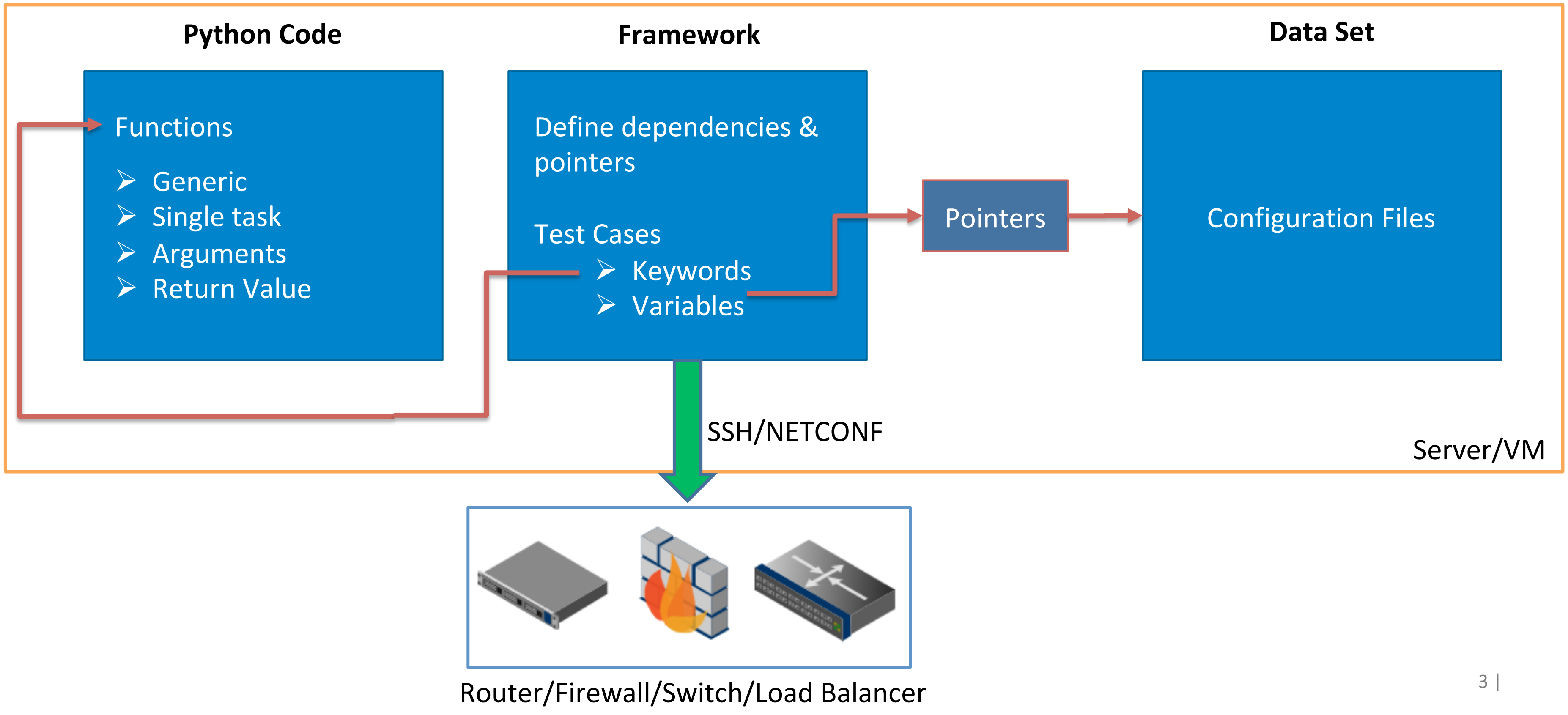
Automating Device Certifications with Robot Framework

Pratik Lotia

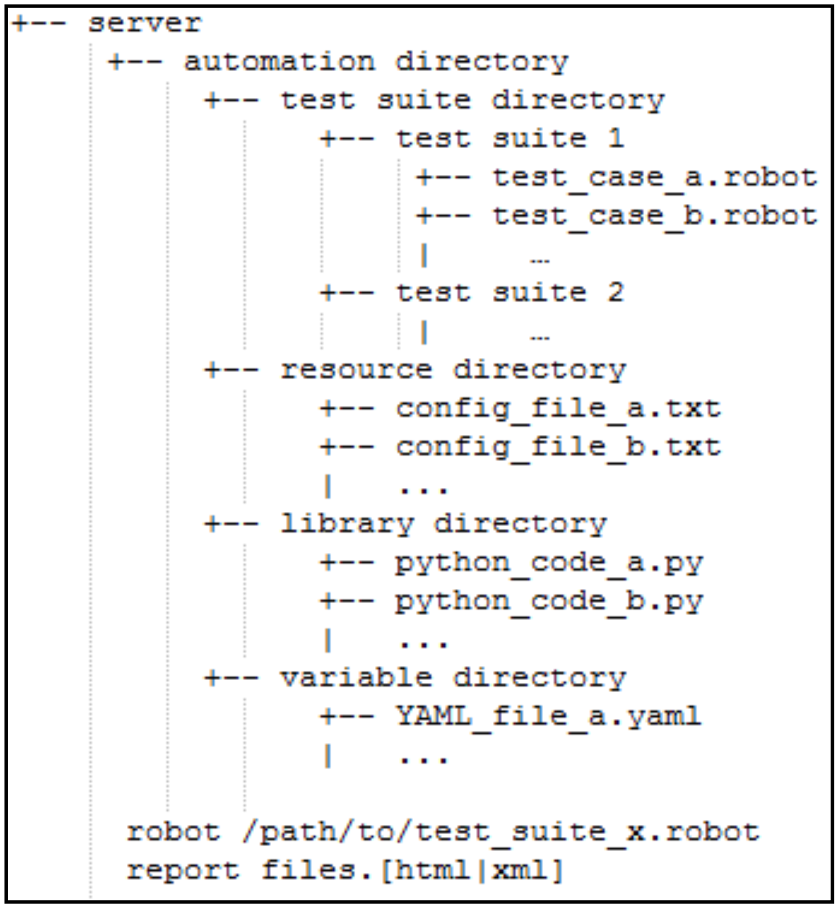
Robot Framework Introduction

- Open source generic test automation framework for acceptance testing
- Keyword driven approach supported with several libraries in Python & Java
- Ideal implementation with high level tests pre-written and network engineers using keywords to develop framework
- Not specifically made for network based testing
- Data driven test cases
- OS and application independent

Robot Framework Major Components



Structure

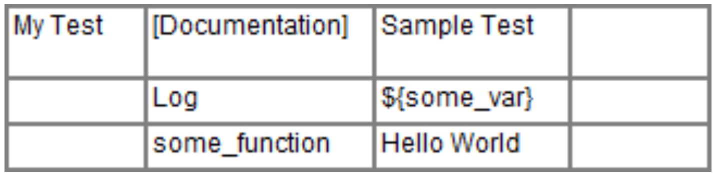


```
+-- server
     +-- automation directory
          +-- test suite directory
                +-- test suite 1
                     +-- test_case_a.robot
                     +-- test_case_b.robot
                     |      …
                +-- test suite 2
                     |      …
          +-- resource directory
                +-- config_file_a.txt
                +-- config_file_b.txt
                |    ...
          +-- library directory
                +-- python_code_a.py
                +-- python_code_b.py
                |    ...
          +-- variable directory
                +-- YAML_file_a.yaml
                |    ...

       robot /path/to/test_suite_x.robot
       report files.[html|xml]
```

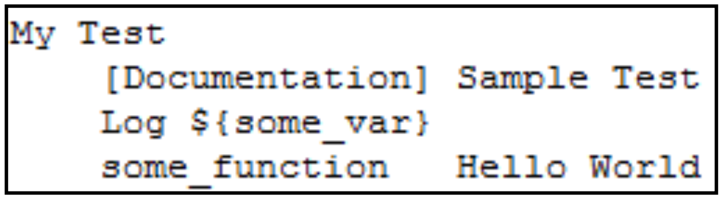
Framework Format

➢ Extension based

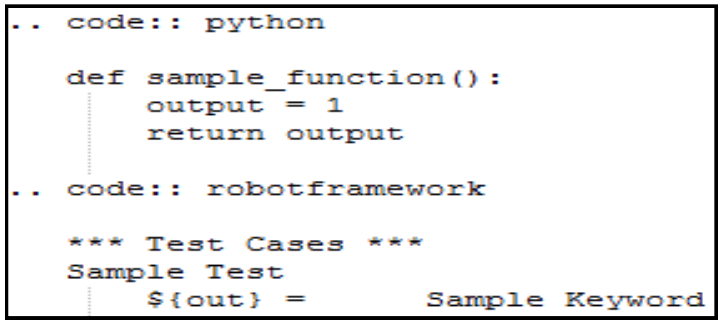
- HTML
- TSV – spreadsheet, programmatic
- Plain text
- reST (HTML compiled)



My Test	[Documentation]	Sample Test	
	Log	${some_var}	
	some_function	Hello World	



```
My Test
    [Documentation]  Sample Test
    Log  ${some_var}
    some_function    Hello World
```

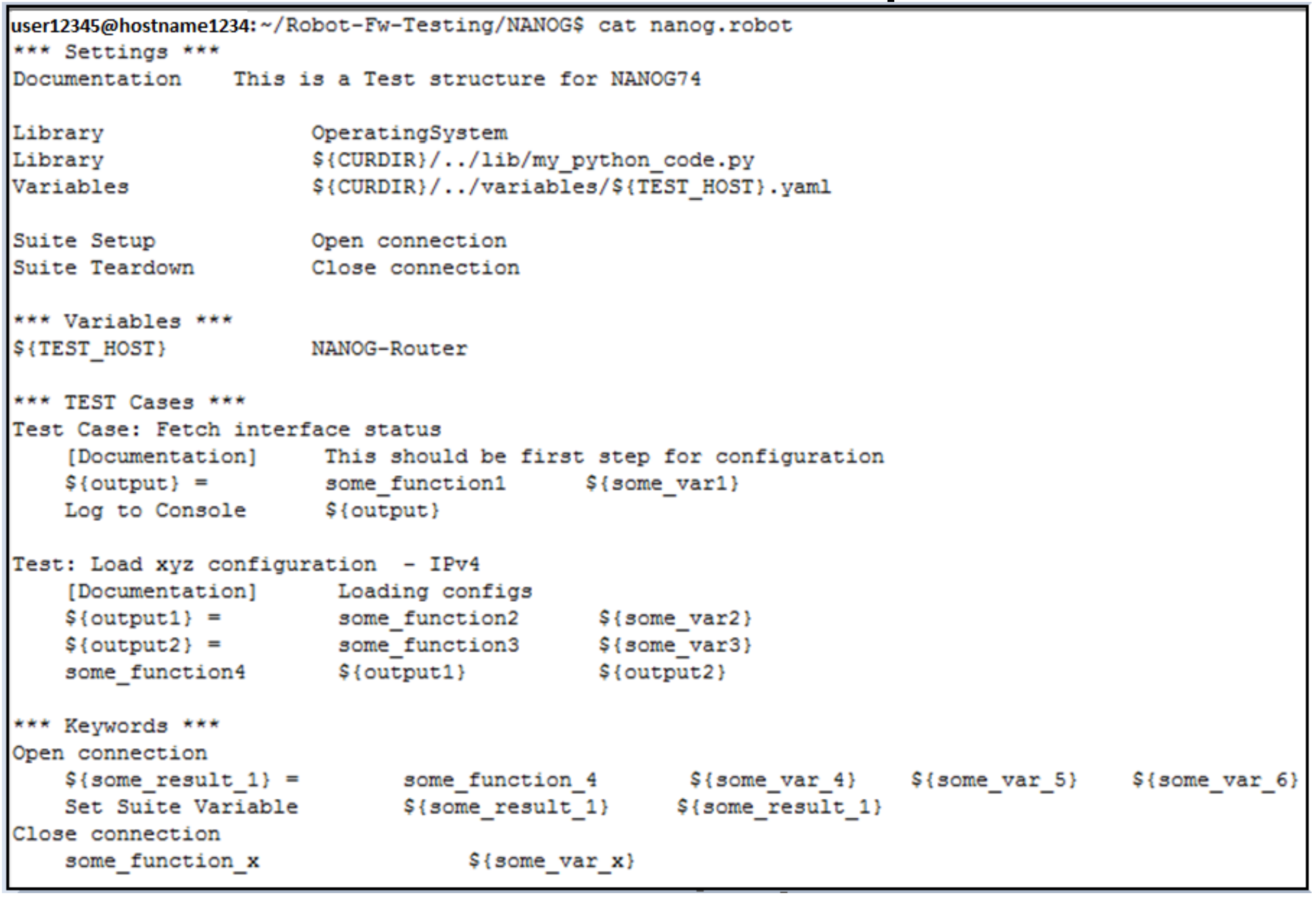


```
.. code:: python

   def sample_function():
       output = 1
       return output

.. code:: robotframework

   *** Test Cases ***
   Sample Test
       ${out} =      Sample Keyword
```

Framework Structure

- Modular model
- Structure combines
    - Settings
    - Pre-test setup
    - Test criteria
    - Post-test cleanup
- Each Test has a true/false outcome
- Each Test has 1 or more functions
- Top-down approach for Test Case
    - One fail, all fail model

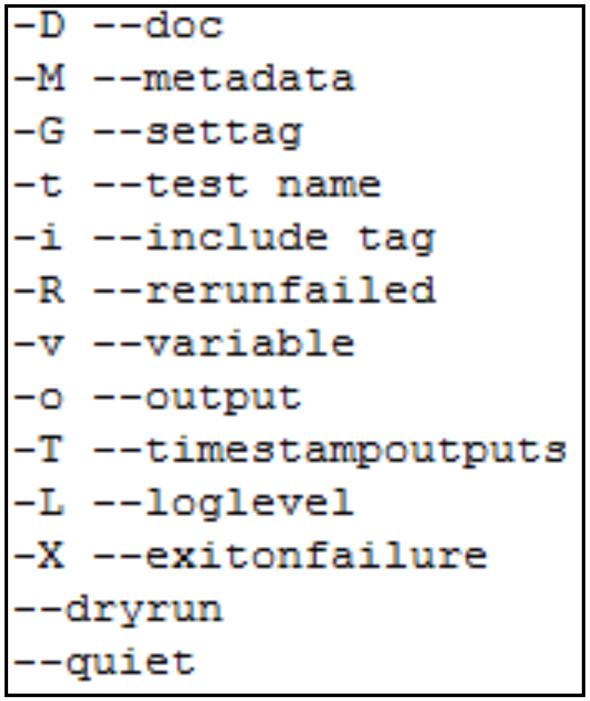
Framework Sample



```
user12345@hostname1234:~/Robot-Fw-Testing/NANOG$ cat nanog.robot
*** Settings ***
Documentation    This is a Test structure for NANOG74

Library                 OperatingSystem
Library                 ${CURDIR}/../lib/my_python_code.py
Variables               ${CURDIR}/../variables/${TEST_HOST}.yaml

Suite Setup             Open connection
Suite Teardown          Close connection

*** Variables ***
${TEST_HOST}            NANOG-Router

*** TEST Cases ***
Test Case: Fetch interface status
    [Documentation]     This should be first step for configuration
    ${output} =         some_function1      ${some_var1}
    Log to Console      ${output}

Test: Load xyz configuration  - IPv4
    [Documentation]      Loading configs
    ${output1} =         some_function2      ${some_var2}
    ${output2} =         some_function3      ${some_var3}
    some_function4       ${output1}          ${output2}

*** Keywords ***
Open connection
    ${some_result_1} =       some_function_4       ${some_var_4}    ${some_var_5}    ${some_var_6}
    Set Suite Variable       ${some_result_1}     ${some_result_1}
Close connection
    some_function_x                ${some_var_x}
```

Robot Command Options

- robot /path/to/file.robot
- Options to:
  - Set documentation
  - Set suite, report name
  - Set tags, variables
  - Rerun failed tests
  - Run/exclude certain tests
  - Set logging level, output level
  - Set timestamp
  - Error handling



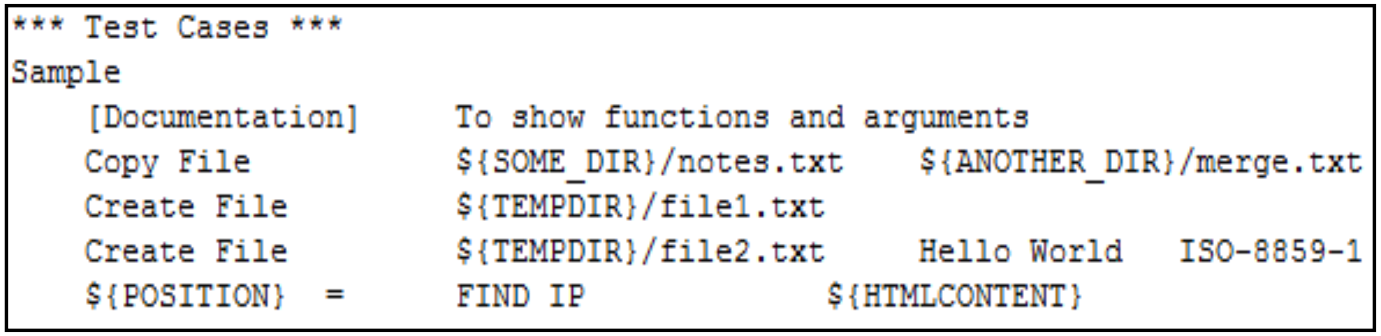
```
-D --doc
-M --metadata
-G --settag
-t --test name
-i --include tag
-R --rerunfailed
-v --variable
-o --output
-T --timestampoutputs
-L --loglevel
-X --exitonfailure
--dryrun
--quiet
```

Style Conventions

Field	Convention	Example/Comments
Variable	Name inside ${}	${mgt_ip_addr)
Function Name	Keyword(s) with or w/ space	Check Interface Config check_interface_config
Passing Variables	Leave >4 whitespaces between function & variable	Show Interface ${mgt_ip_addr} ${user} ${passwd}
Notes	Under [Documentation] in test cases	Test case: check interface config [Documentation] Load and verify IP
Variables	Define separately yaml file	Helps to keep framework generic and data driven
Framework	Tabular model	Equal spacing

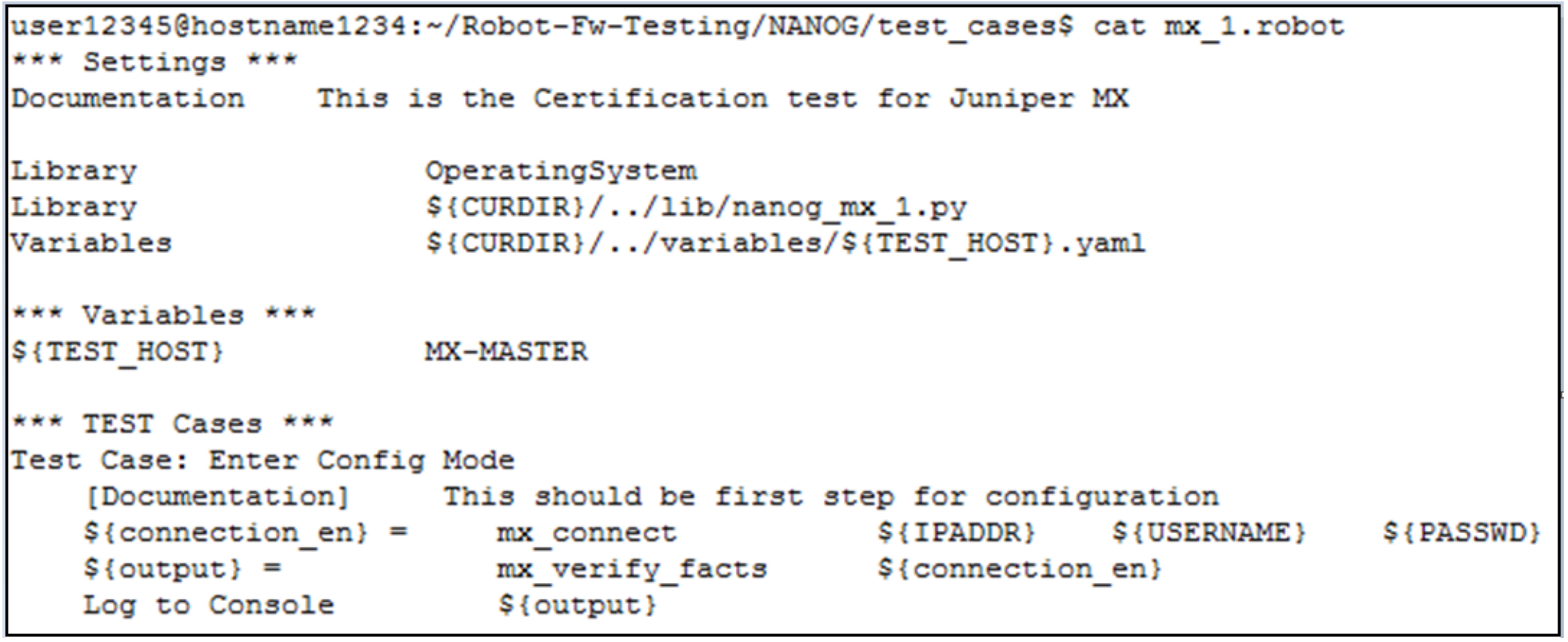
Creating Test Case

- Whitespaces ignored*
- Keyword (What?)
- Library + Python code
- Arguments:
  - Mandatory
  - Default
- Return Value
- Single Test



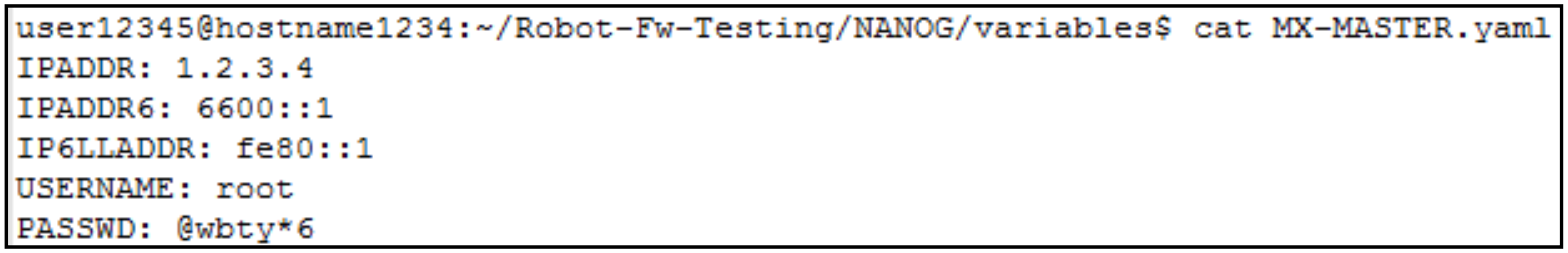
```
*** Test Cases ***
Sample
    [Documentation]     To show functions and arguments
    Copy File           ${SOME_DIR}/notes.txt    ${ANOTHER_DIR}/merge.txt
    Create File         ${TEMPDIR}/file1.txt
    Create File         ${TEMPDIR}/file2.txt     Hello World   ISO-8859-1
    ${POSITION}  =      FIND IP             ${HTMLCONTENT}
```

Example 1 – Operational Status of Device – Framework



```
user12345@hostname1234:~/Robot-Fw-Testing/NANOG/test_cases$ cat mx_1.robot
*** Settings ***
Documentation    This is the Certification test for Juniper MX

Library                 OperatingSystem
Library                 ${CURDIR}/../lib/nanog_mx_1.py
Variables               ${CURDIR}/../variables/${TEST_HOST}.yaml

*** Variables ***
${TEST_HOST}            MX-MASTER

*** TEST Cases ***
Test Case: Enter Config Mode
    [Documentation]     This should be first step for configuration
    ${connection_en} =     mx_connect          ${IPADDR}    ${USERNAME}    ${PASSWD}
    ${output} =            mx_verify_facts     ${connection_en}
    Log to Console         ${output}
```

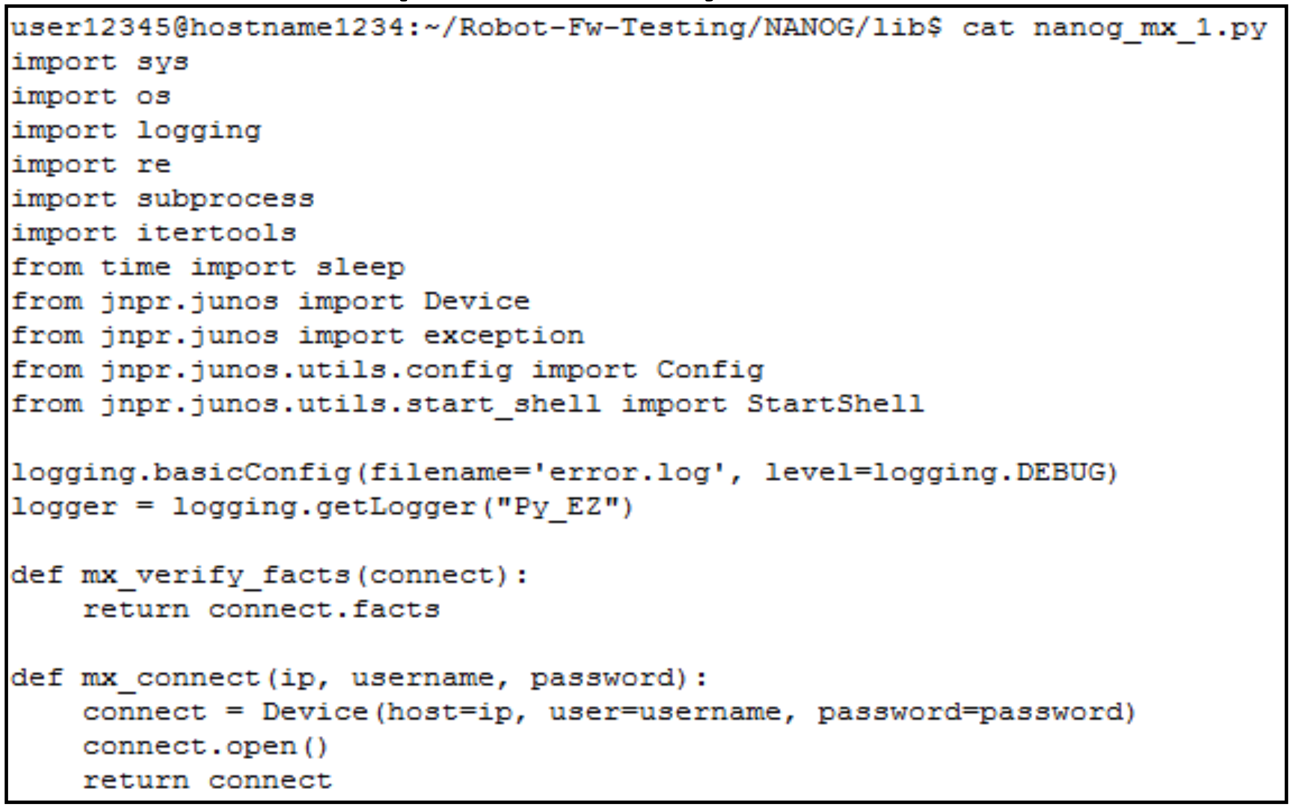
Example 1 – Variables



```
user12345@hostname1234:~/Robot-Fw-Testing/NANOG/variables$ cat MX-MASTER.yaml
IPADDR: 1.2.3.4
IPADDR6: 6600::1
IP6LLADDR: fe80::1
USERNAME: root
PASSWD: @wbty*6
```

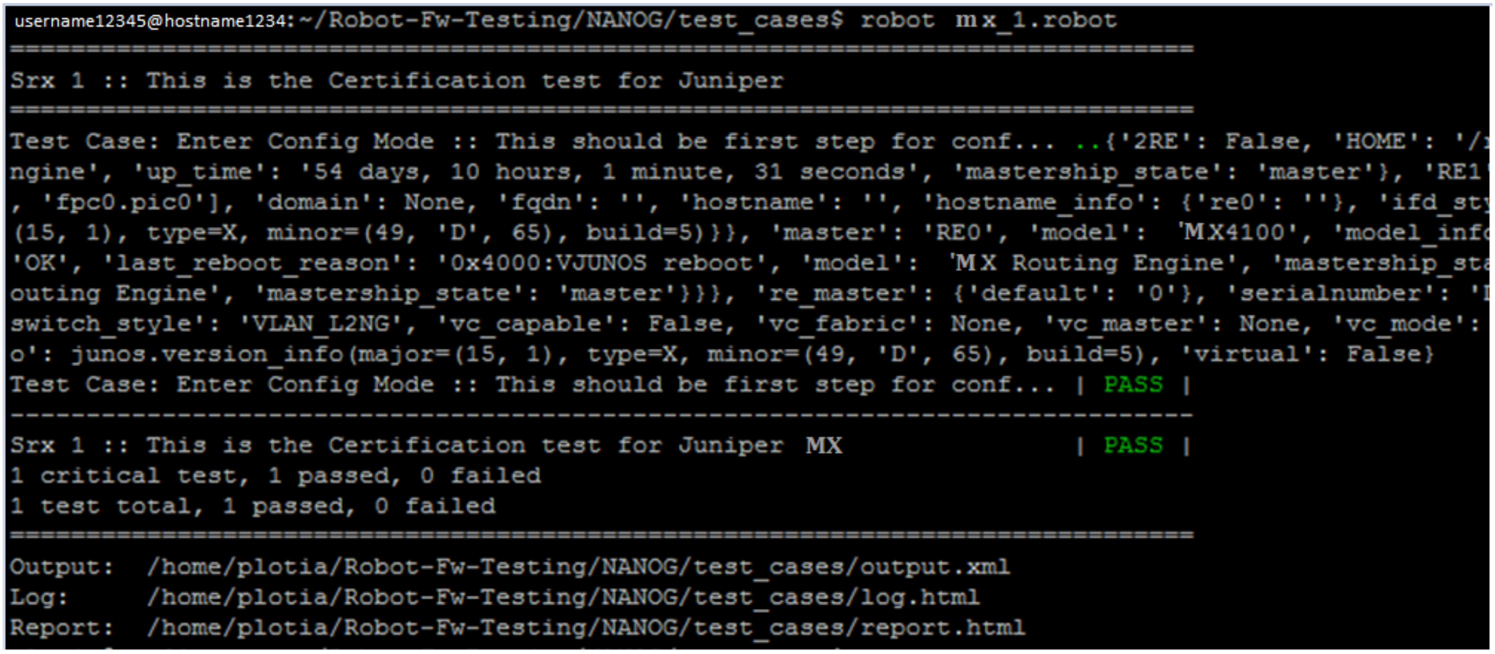
*Fake credentials on this and subsequent slides

Example 1 – Python Code



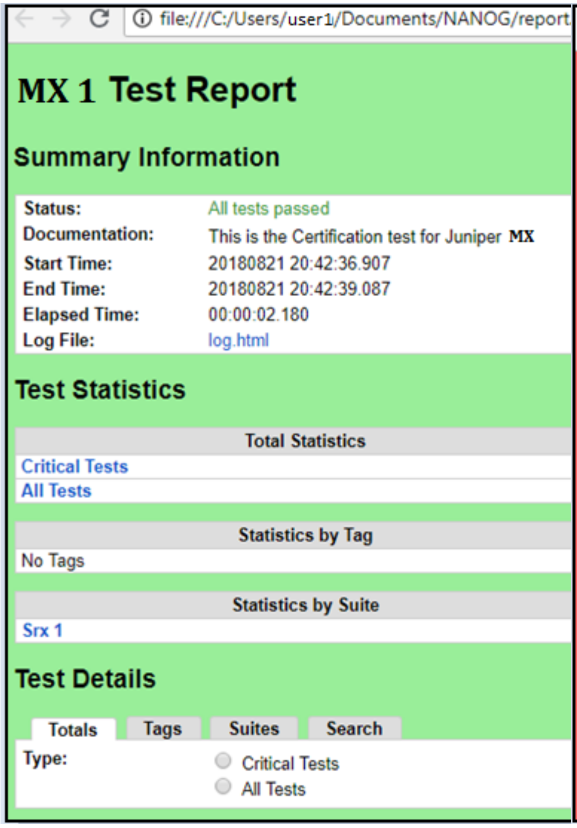
```
user12345@hostname1234:~/Robot-Fw-Testing/NANOG/lib$ cat nanog_mx_1.py
import sys
import os
import logging
import re
import subprocess
import itertools
from time import sleep
from jnpr.junos import Device
from jnpr.junos import exception
from jnpr.junos.utils.config import Config
from jnpr.junos.utils.start_shell import StartShell

logging.basicConfig(filename='error.log', level=logging.DEBUG)
logger = logging.getLogger("Py_EZ")

def mx_verify_facts(connect):
    return connect.facts

def mx_connect(ip, username, password):
    connect = Device(host=ip, user=username, password=password)
    connect.open()
    return connect
```

Example 1 - Results



```
username12345@hostname1234:~/Robot-Fw-Testing/NANOG/test_cases$ robot mx_1.robot
==============================================================================
Srx 1 :: This is the Certification test for Juniper
==============================================================================
Test Case: Enter Config Mode :: This should be first step for conf... ..{'2RE': False, 'HOME': '/
ngine', 'up_time': '54 days, 10 hours, 1 minute, 31 seconds', 'mastership_state': 'master'}, 'RE1'
, 'fpc0.pic0'], 'domain': None, 'fqdn': '', 'hostname': '', 'hostname_info': {'re0': ''}, 'ifd_sty
(15, 1), type=X, minor=(49, 'D', 65), build=5)}}, 'master': 'RE0', 'model': 'MX4100', 'model_info
'OK', 'last_reboot_reason': '0x4000:VJUNOS reboot', 'model': 'MX Routing Engine', 'mastership_sta
outing Engine', 'mastership_state': 'master'}}}, 're_master': {'default': '0'}, 'serialnumber': 'I
switch_style': 'VLAN_L2NG', 'vc_capable': False, 'vc_fabric': None, 'vc_master': None, 'vc_mode':
o': junos.version_info(major=(15, 1), type=X, minor=(49, 'D', 65), build=5), 'virtual': False}
Test Case: Enter Config Mode :: This should be first step for conf... | PASS |
------------------------------------------------------------------------------
Srx 1 :: This is the Certification test for Juniper MX                | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /home/plotia/Robot-Fw-Testing/NANOG/test_cases/output.xml
Log:     /home/plotia/Robot-Fw-Testing/NANOG/test_cases/log.html
Report:  /home/plotia/Robot-Fw-Testing/NANOG/test_cases/report.html
```

Results – Executive Summary



file:///C:/Users/user1/Documents/NANOG/report

MX 1 Test Report

Summary Information

Status:	All tests passed
Documentation:	This is the Certification test for Juniper MX
Start Time:	20180821 20:42:36.907
End Time:	20180821 20:42:39.087
Elapsed Time:	00:00:02.180
Log File:	log.html

Test Statistics

Total Statistics
Critical Tests
All Tests

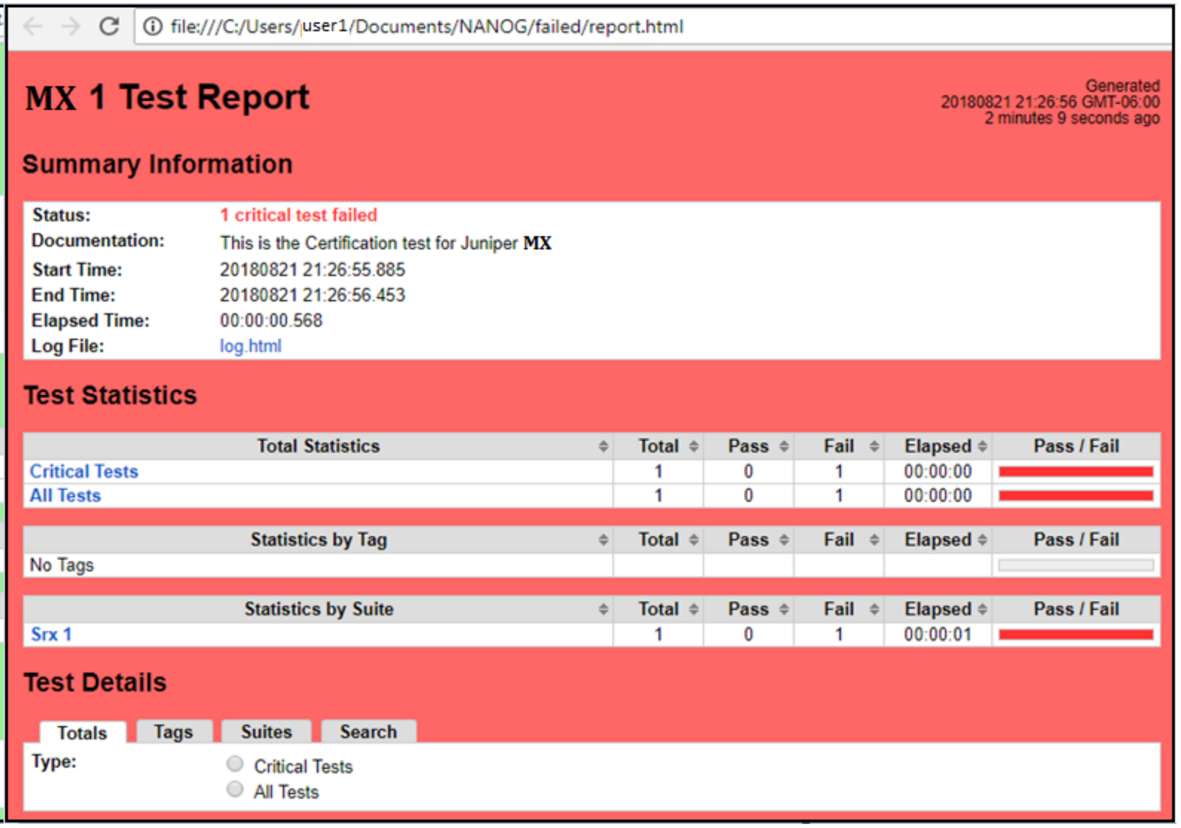
Statistics by Tag
No Tags

Statistics by Suite
Srx 1

Test Details

Totals Tags Suites Search

Type: Critical Tests / All Tests



file:///C:/Users/user1/Documents/NANOG/failed/report.html

MX 1 Test Report

Generated
20180821 21:26:56 GMT-06:00
2 minutes 9 seconds ago

Summary Information

Status:	1 critical test failed
Documentation:	This is the Certification test for Juniper MX
Start Time:	20180821 21:26:55.885
End Time:	20180821 21:26:56.453
Elapsed Time:	00:00:00.568
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	0	1	00:00:00	🟥
All Tests	1	0	1	00:00:00	🟥

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Srx 1	1	0	1	00:00:01	🟥

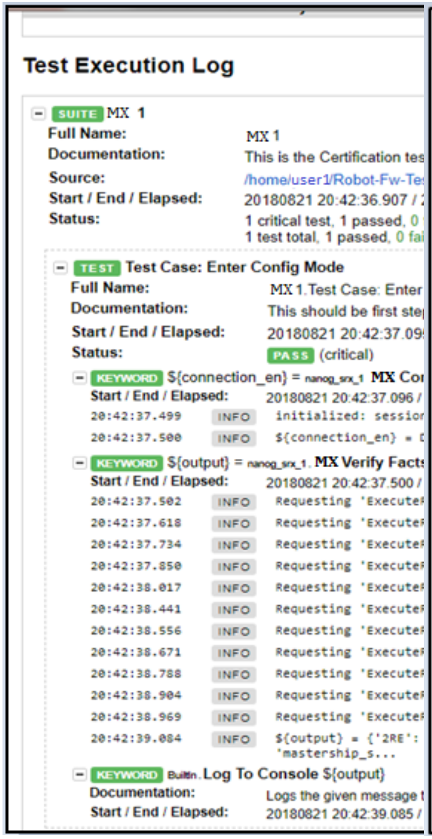
Test Details

Totals Tags Suites Search

Type: Critical Tests / All Tests

*Failed test may be a result of misconfiguration and not a failure of the device

Results – Detailed Logs



Test Execution Log

SUITE MX 1

Full Name:	MX 1
Documentation:	This is the Certification tes
Source:	/home/user1/Robot-Fw-Tes
Start / End / Elapsed:	20180821 20:42:36.907 / 2
Status:	1 critical test, 1 passed, 0 f 1 test total, 1 passed, 0 fai

TEST Test Case: Enter Config Mode

Full Name:	MX 1.Test Case: Enter
Documentation:	This should be first step
Start / End / Elapsed:	20180821 20:42:37.09
Status:	PASS (critical)

KEYWORD ${connection_en} = nanog_srx_1 MX Cor

Start / End / Elapsed: 20180821 20:42:37.096 /

```
20:42:37.499   INFO   initialized: sessior
20:42:37.500   INFO   ${connection_en} = [
```

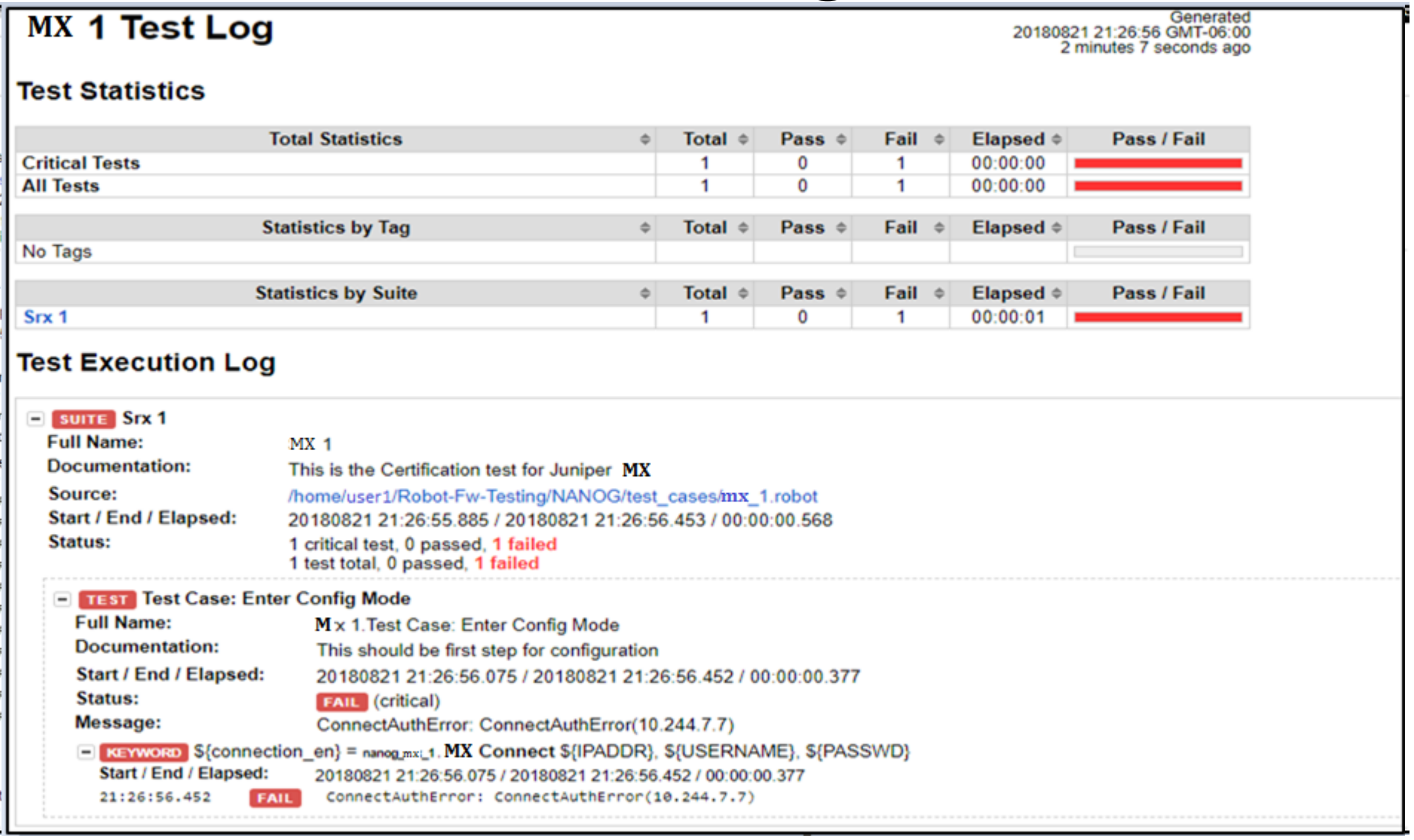
KEYWORD ${output} = nanog_srx_1. MX Verify Facts

Start / End / Elapsed: 20180821 20:42:37.500 /

```
20:42:37.502   INFO   Requesting 'Execute
20:42:37.618   INFO   Requesting 'Execute
20:42:37.734   INFO   Requesting 'Execute
20:42:37.850   INFO   Requesting 'Execute
20:42:38.017   INFO   Requesting 'Execute
20:42:38.441   INFO   Requesting 'Execute
20:42:38.556   INFO   Requesting 'Execute
20:42:38.671   INFO   Requesting 'Execute
20:42:38.788   INFO   Requesting 'Execute
20:42:38.904   INFO   Requesting 'Execute
20:42:38.969   INFO   Requesting 'Execute
20:42:39.084   INFO   ${output} = {'2RE':
                      'mastership_s...
```

KEYWORD BuiltIn. Log To Console ${output}

Documentation:	Logs the given message t
Start / End / Elapsed:	20180821 20:42:39.085 /



MX 1 Test Log

Generated
20180821 21:26:56 GMT-06:00
2 minutes 7 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	0	1	00:00:00	🟥
All Tests	1	0	1	00:00:00	🟥

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Srx 1	1	0	1	00:00:01	🟥

Test Execution Log

SUITE Srx 1

Full Name:	MX 1
Documentation:	This is the Certification test for Juniper MX
Source:	/home/user1/Robot-Fw-Testing/NANOG/test_cases/mx_1.robot
Start / End / Elapsed:	20180821 21:26:55.885 / 20180821 21:26:56.453 / 00:00:00.568
Status:	1 critical test, 0 passed, 1 failed 1 test total, 0 passed, 1 failed

TEST Test Case: Enter Config Mode

Full Name:	Mx 1.Test Case: Enter Config Mode
Documentation:	This should be first step for configuration
Start / End / Elapsed:	20180821 21:26:56.075 / 20180821 21:26:56.452 / 00:00:00.377
Status:	FAIL (critical)
Message:	ConnectAuthError: ConnectAuthError(10.244.7.7)

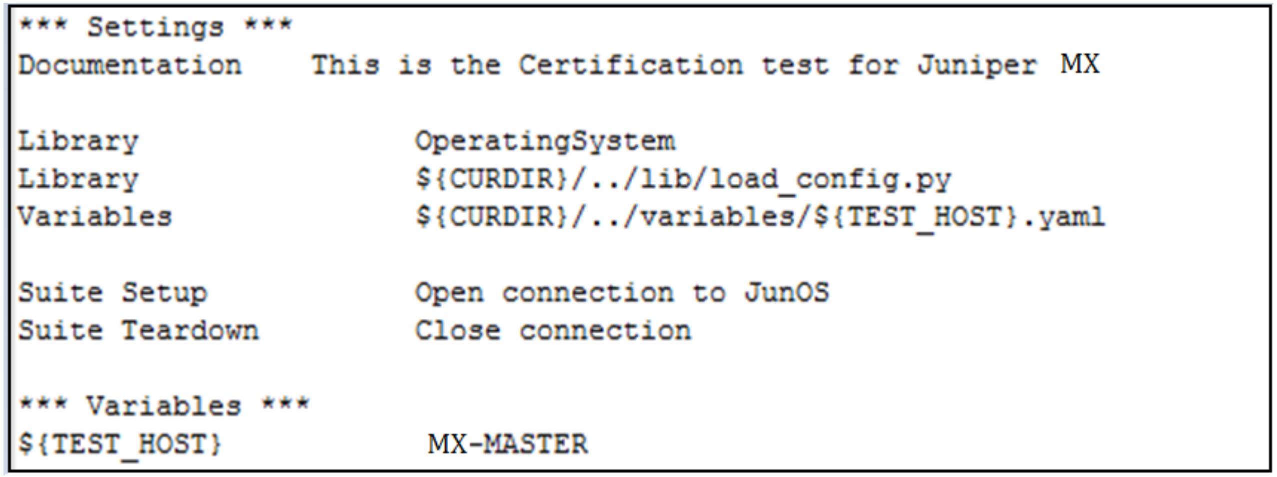
KEYWORD ${connection_en} = nanog_mx_1. MX Connect ${IPADDR}, ${USERNAME}, ${PASSWD}

Start / End / Elapsed: 20180821 21:26:56.075 / 20180821 21:26:56.452 / 00:00:00.377

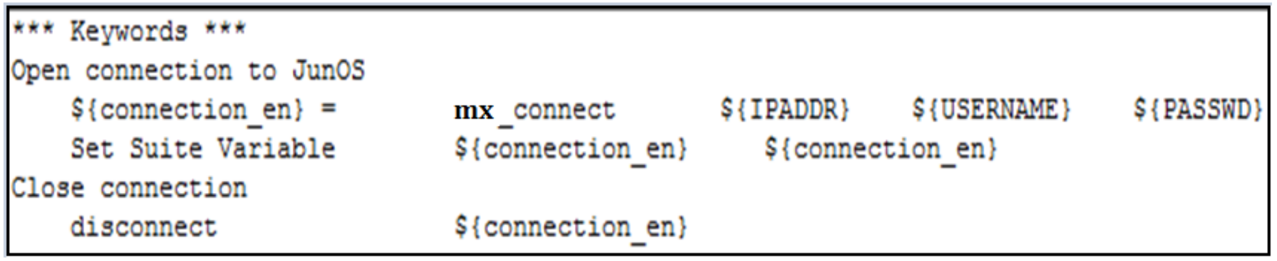
```
21:26:56.452   FAIL   ConnectAuthError: ConnectAuthError(10.244.7.7)
```

*Failed test may be a result of misconfiguration and not a failure of the device

Suite Setup and Teardown



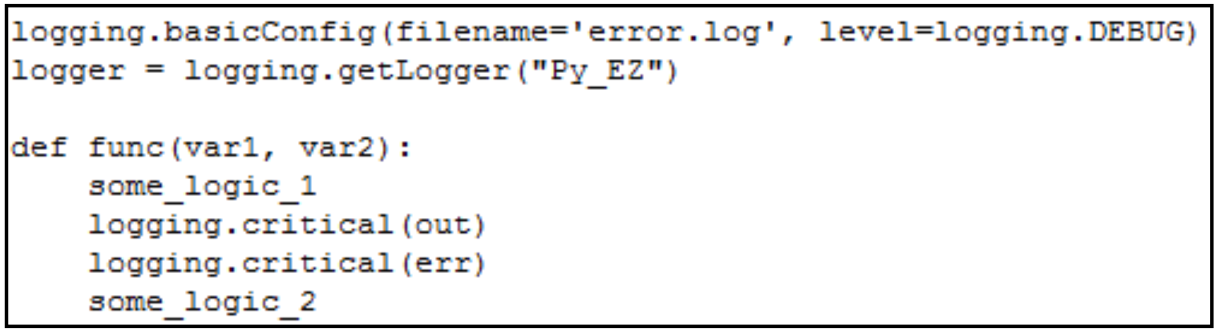
```
*** Settings ***
Documentation    This is the Certification test for Juniper MX

Library                  OperatingSystem
Library                  ${CURDIR}/../lib/load_config.py
Variables                ${CURDIR}/../variables/${TEST_HOST}.yaml

Suite Setup              Open connection to JunOS
Suite Teardown           Close connection

*** Variables ***
${TEST_HOST}              MX-MASTER
```



```
*** Keywords ***
Open connection to JunOS
    ${connection_en} =       mx_connect      ${IPADDR}    ${USERNAME}    ${PASSWD}
    Set Suite Variable       ${connection_en}    ${connection_en}
Close connection
    disconnect               ${connection_en}
```

Troubleshooting Errors

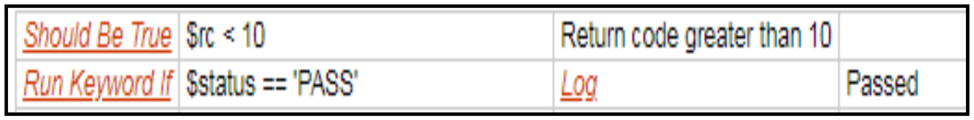
- Default errors are minimal
- Tedious to look at html for errors
- Logging module in Python
- Similar to print statements
- Prints while running tests



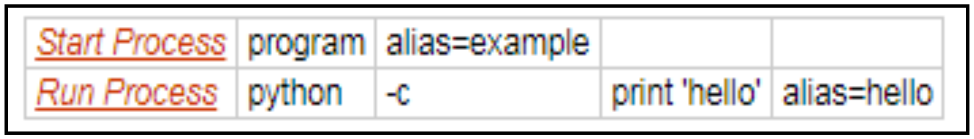
```
logging.basicConfig(filename='error.log', level=logging.DEBUG)
logger = logging.getLogger("Py_EZ")

def func(var1, var2):
    some_logic_1
    logging.critical(out)
    logging.critical(err)
    some_logic_2
```

Libraries

- Standard Libraries
  - Built-in
    - Run with conditions
    - Evaluation
    - Matching expected behavior
  - Process oriented
    - Control process execution
    - Fetch process attributes
    - Switch process



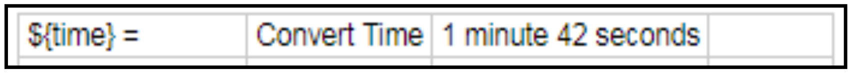
Should Be True	$rc < 10	Return code greater than 10	
Run Keyword If	$status == 'PASS'	Log	Passed



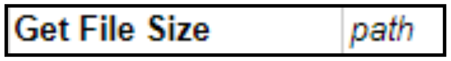
Start Process	program	alias=example		
Run Process	python	-c	print 'hello'	alias=hello

Libraries

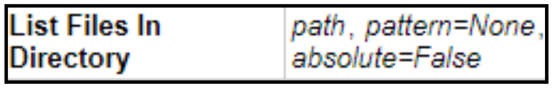
- Standard Libraries
  - DateTime
    - Date and Time conversions
    - Adding time/date
    - Subtracting time/date
  - OS level functions
    - Directory changes/verification
    - File changes/verification (copy, size)
    - Environment variables
    - Merge/List



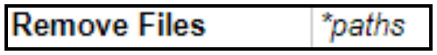
${time} =	Convert Time	1 minute 42 seconds	



Get File Size	*path*



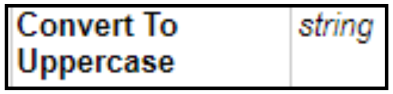
List Files In Directory	*path*, *pattern=None*, *absolute=False*



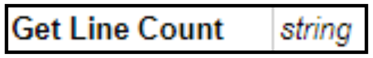
Remove Files	**paths*

Libraries

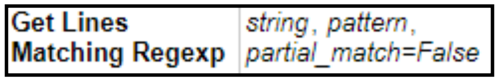
- Standard Libraries
  - String functions
    - Length control/verification
    - Behavior matching
    - Byte conversion
  - Collections
    - Control Lists/Dictionaries
    - Behavior matching
  - Dynamic input, Telnet
  - Screenshots



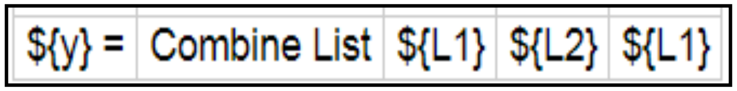
Convert To Uppercase	*string*



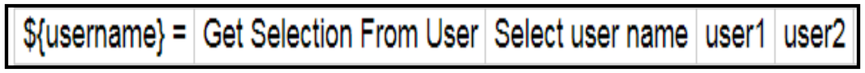
Get Line Count	*string*



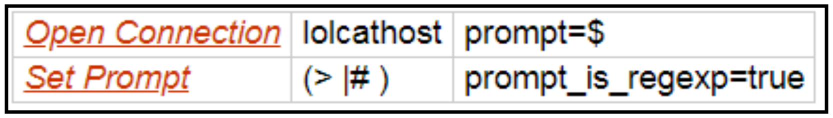
Get Lines Matching Regexp	*string, pattern, partial_match=False*



${y} =	Combine List	${L1}	${L2}	${L1}



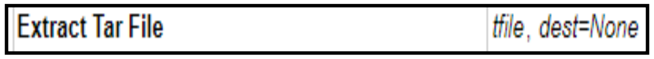
${username} =	Get Selection From User	Select user name	user1	user2



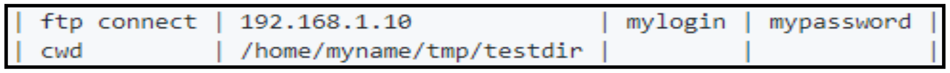
Open Connection	lolcathost	prompt=$
Set Prompt	(> |#)	prompt_is_regexp=true

Libraries

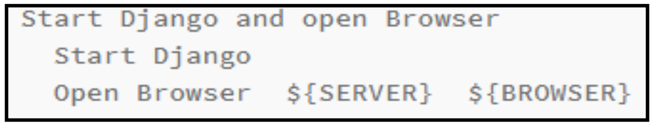
- Extended Libraries
  - Selenium, Selenium with Angular JS
  - Suds (SOAP), MQTT, Faker
  - SSH, Nc client, Django, FTP
  - Database, HTTP, Archive



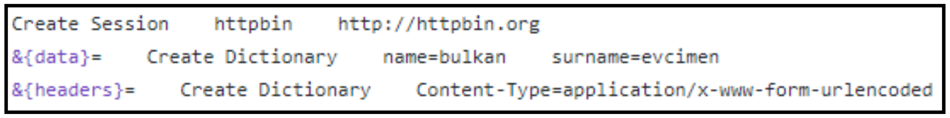
```
Extract Tar File                                   tfile, dest=None
```



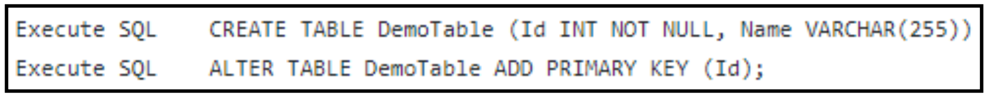
```
| ftp connect | 192.168.1.10              | mylogin | mypassword |
| cwd         | /home/myname/tmp/testdir |         |            |
```



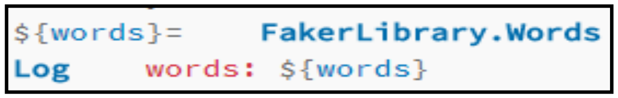
```
Start Django and open Browser
   Start Django
   Open Browser   ${SERVER}   ${BROWSER}
```



```
Create Session     httpbin     http://httpbin.org
&{data}=     Create Dictionary     name=bulkan     surname=evcimen
&{headers}=     Create Dictionary     Content-Type=application/x-www-form-urlencoded
```



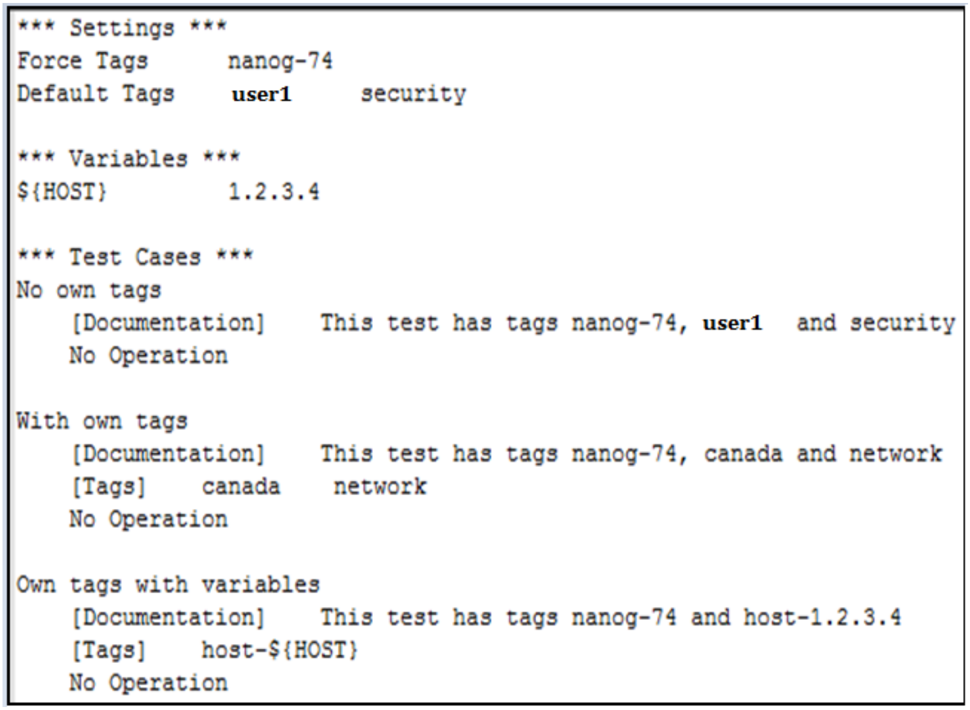
```
Execute SQL    CREATE TABLE DemoTable (Id INT NOT NULL, Name VARCHAR(255))
Execute SQL    ALTER TABLE DemoTable ADD PRIMARY KEY (Id);
```



```
${words}=     FakerLibrary.Words
Log      words:  ${words}
```

Tagging

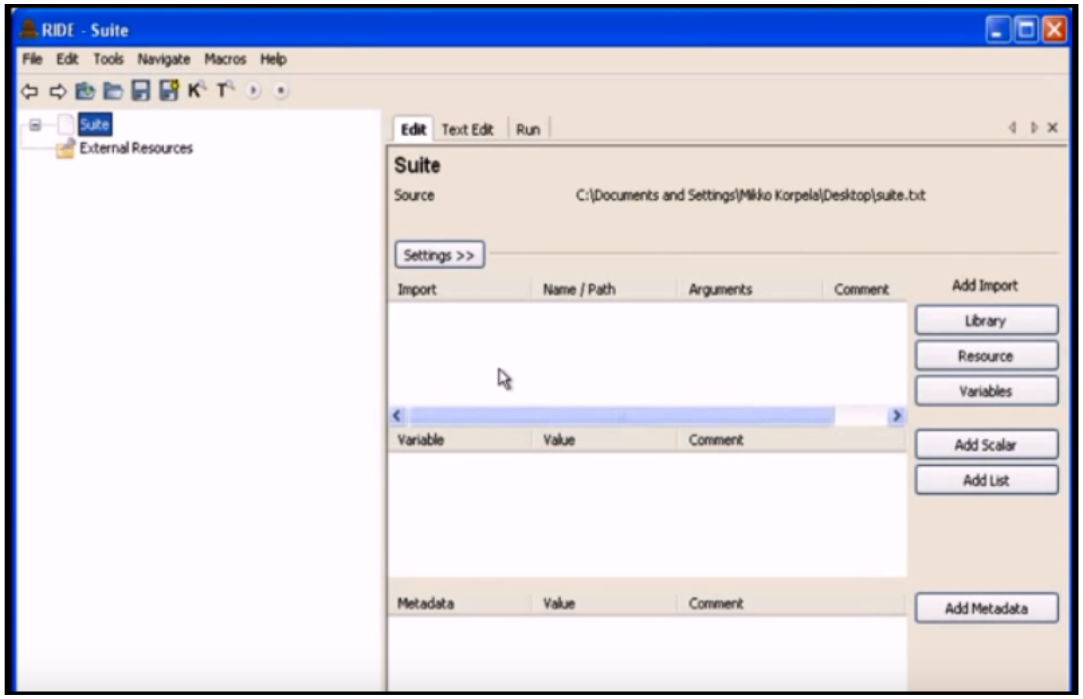
- Classifying test cases & providing metadata
- Report shows statistics based on tags
- Include/Exclude execution of specific
- Tags
- Tags for critical, non-critical, trivial
- Types
  - Force tags
  - Default tags
  - Customized tags



```
*** Settings ***
Force Tags      nanog-74
Default Tags     user1     security

*** Variables ***
${HOST}         1.2.3.4

*** Test Cases ***
No own tags
    [Documentation]    This test has tags nanog-74, user1   and security
    No Operation

With own tags
    [Documentation]    This test has tags nanog-74, canada and network
    [Tags]    canada    network
    No Operation

Own tags with variables
    [Documentation]    This test has tags nanog-74 and host-1.2.3.4
    [Tags]    host-${HOST}
    No Operation
```

Editor

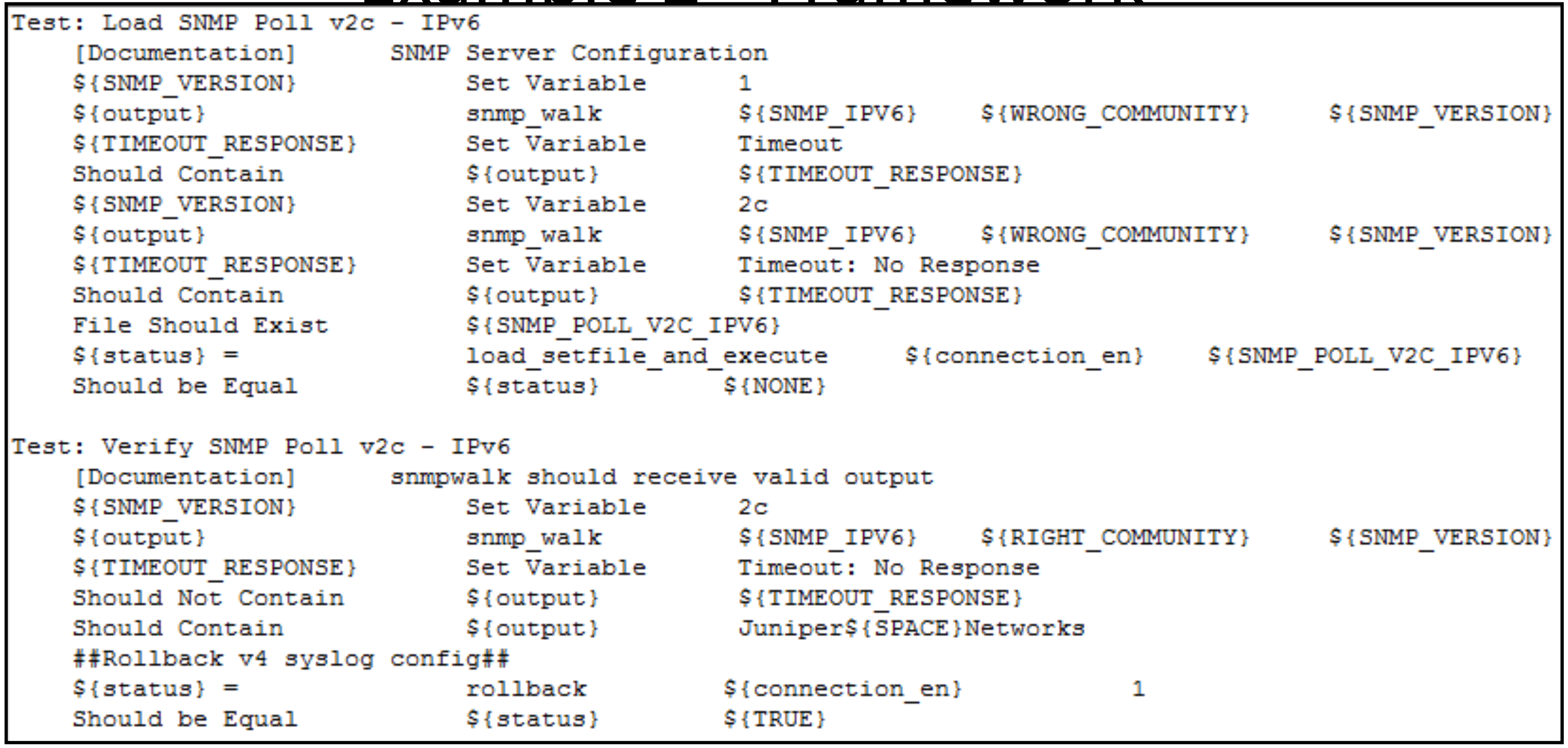
- RIDE – Standalone editor
- Plugins for various editors
  - Eclipse
  - Sublime
  - Vim
  - Emacs
  - Gedit
  - Notepad++



Source C:\Documents and Settings\Mikko Korpela\Desktop\suite.txt

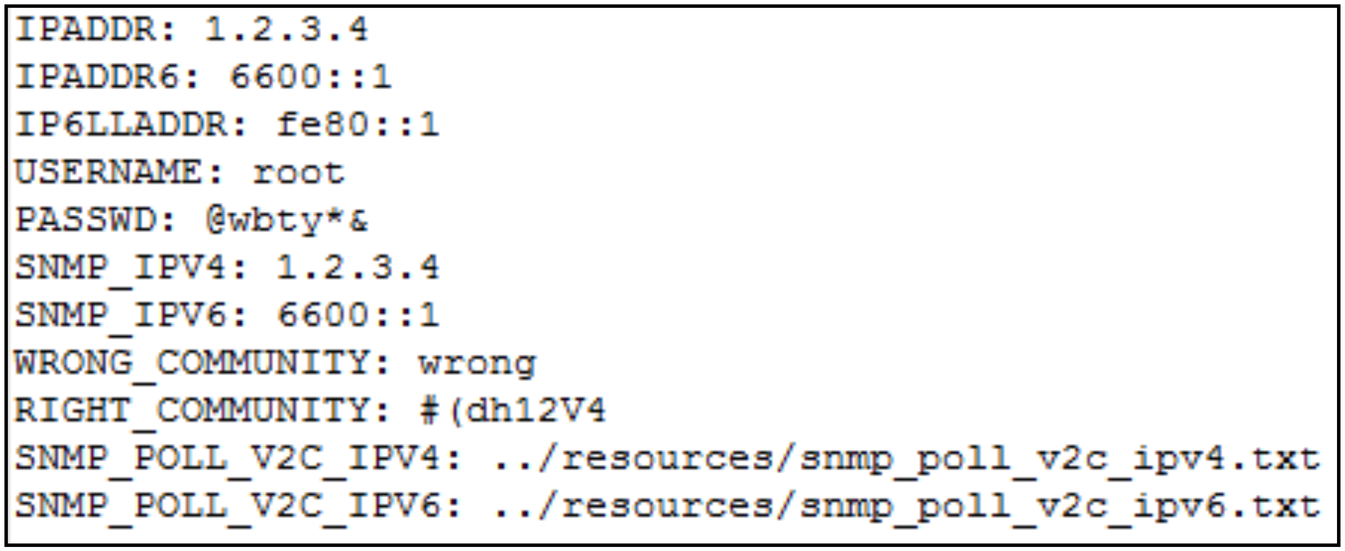
*https://www.youtube.com/watch?feature=player_embedded&v=6F_xGKdoN1E

Example 2 - Framework



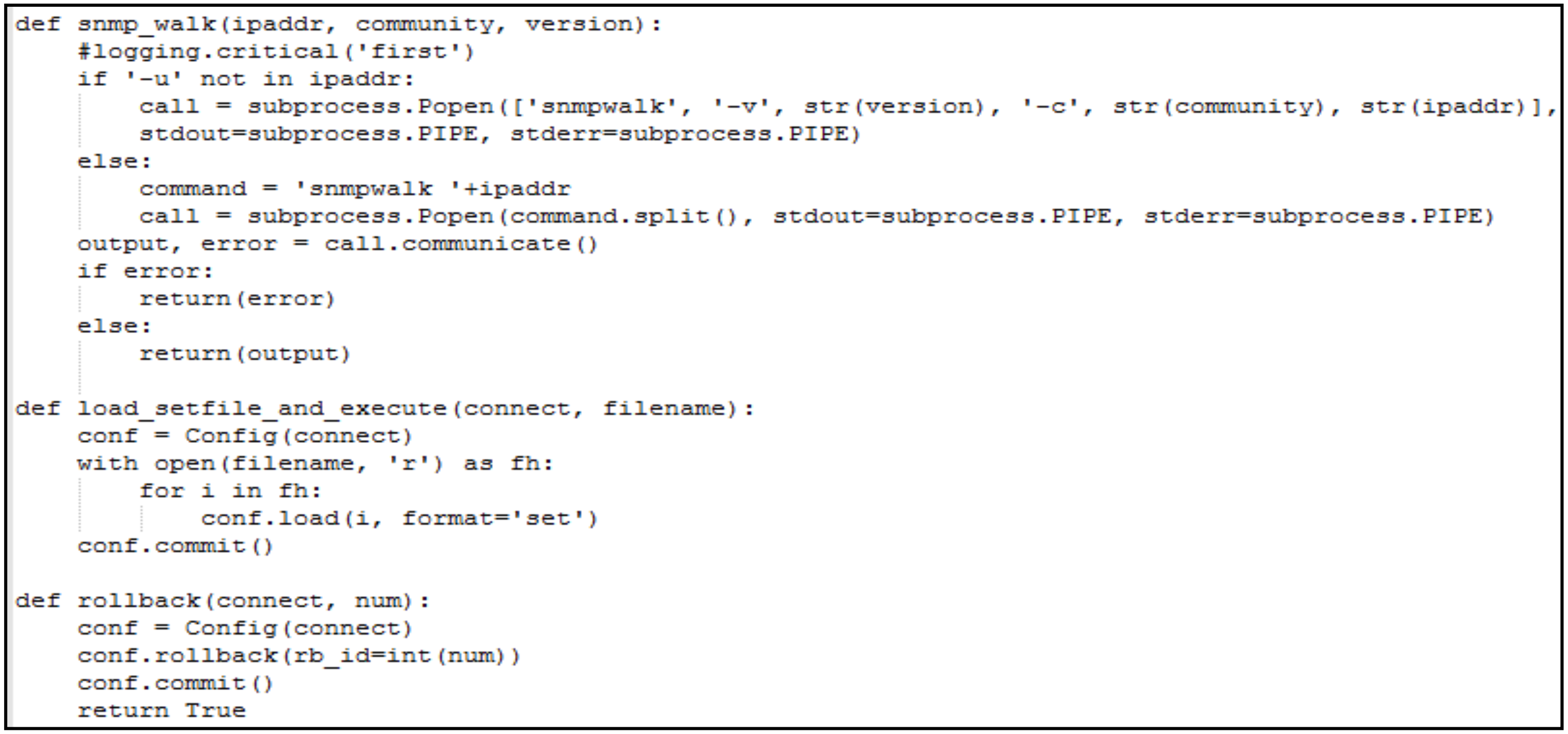
```
Test: Load SNMP Poll v2c - IPv6
    [Documentation]      SNMP Server Configuration
    ${SNMP_VERSION}           Set Variable      1
    ${output}                 snmp_walk         ${SNMP_IPV6}    ${WRONG_COMMUNITY}     ${SNMP_VERSION}
    ${TIMEOUT_RESPONSE}       Set Variable      Timeout
    Should Contain            ${output}         ${TIMEOUT_RESPONSE}
    ${SNMP_VERSION}           Set Variable      2c
    ${output}                 snmp_walk         ${SNMP_IPV6}    ${WRONG_COMMUNITY}     ${SNMP_VERSION}
    ${TIMEOUT_RESPONSE}       Set Variable      Timeout: No Response
    Should Contain            ${output}         ${TIMEOUT_RESPONSE}
    File Should Exist         ${SNMP_POLL_V2C_IPV6}
    ${status} =               load_setfile_and_execute     ${connection_en}    ${SNMP_POLL_V2C_IPV6}
    Should be Equal           ${status}        ${NONE}

Test: Verify SNMP Poll v2c - IPv6
    [Documentation]      snmpwalk should receive valid output
    ${SNMP_VERSION}           Set Variable      2c
    ${output}                 snmp_walk         ${SNMP_IPV6}    ${RIGHT_COMMUNITY}     ${SNMP_VERSION}
    ${TIMEOUT_RESPONSE}       Set Variable      Timeout: No Response
    Should Not Contain        ${output}         ${TIMEOUT_RESPONSE}
    Should Contain            ${output}         Juniper${SPACE}Networks
    ##Rollback v4 syslog config##
    ${status} =               rollback         ${connection_en}           1
    Should be Equal           ${status}        ${TRUE}
```

Example 2 - Variables



```
IPADDR: 1.2.3.4
IPADDR6: 6600::1
IP6LLADDR: fe80::1
USERNAME: root
PASSWD: @wbty*&
SNMP_IPV4: 1.2.3.4
SNMP_IPV6: 6600::1
WRONG_COMMUNITY: wrong
RIGHT_COMMUNITY: #(dh12V4
SNMP_POLL_V2C_IPV4: ../resources/snmp_poll_v2c_ipv4.txt
SNMP_POLL_V2C_IPV6: ../resources/snmp_poll_v2c_ipv6.txt
```

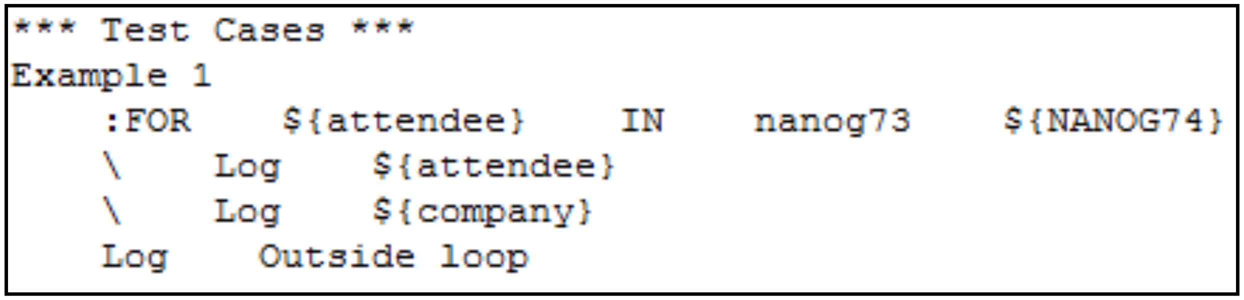
Example 2 – Python Code



```
def snmp_walk(ipaddr, community, version):
    #logging.critical('first')
    if '-u' not in ipaddr:
        call = subprocess.Popen(['snmpwalk', '-v', str(version), '-c', str(community), str(ipaddr)],
        stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    else:
        command = 'snmpwalk '+ipaddr
        call = subprocess.Popen(command.split(), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = call.communicate()
    if error:
        return(error)
    else:
        return(output)

def load_setfile_and_execute(connect, filename):
    conf = Config(connect)
    with open(filename, 'r') as fh:
        for i in fh:
            conf.load(i, format='set')
    conf.commit()

def rollback(connect, num):
    conf = Config(connect)
    conf.rollback(rb_id=int(num))
    conf.commit()
    return True
```

Loops

- 'For' Loop
- Repetitive tasks
- Keyword/Variable



```
*** Test Cases ***
Example 1
    :FOR    ${attendee}    IN    nanog73    ${NANOG74}
    \    Log    ${attendee}
    \    Log    ${company}
    Log    Outside loop
```

Additional Tools

- Rebot
  - Process XML output
  - Generate html reports
  - Combine or Merge reports

```
rebot output.xml
rebot output1.xml output2.xml
rebot --merge --name Sample --critical regression original.xml merged.xml
robot --rerunfailed output1.xml --output rerun.xml tests
rebot --merge original.xml rerun.xml
```

- Libdoc
  - Generate Documentation

```
python -m robot.libdoc test/resource.html doc/resource_doc.html
```

- Tidy
  - Cleanup / Change format

```
python -m robot.tidy [options] inputfile [outputfile]
```

- DbBot
  - Reports to SQLite
  - Unify storage of reports

```
python -m dbbot.run atest/testdata/one_suite/output.xml
```

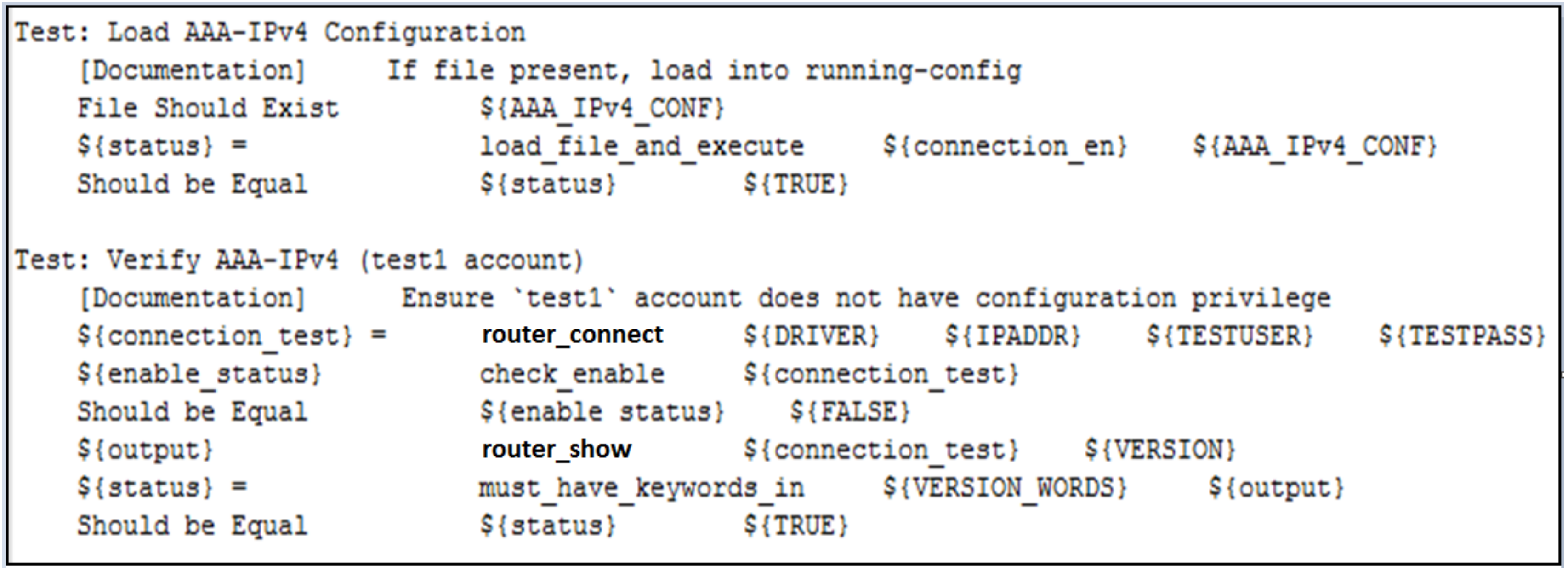
Additional Tools

- Robot Corder
  - Record GUI actions
  - HTML framework generation
- Pabot
  - Parallel execution
  - Time
- Fixml
  - Fixing incomplete xml results
- Mabot
  - Manual tests with compatible outputs

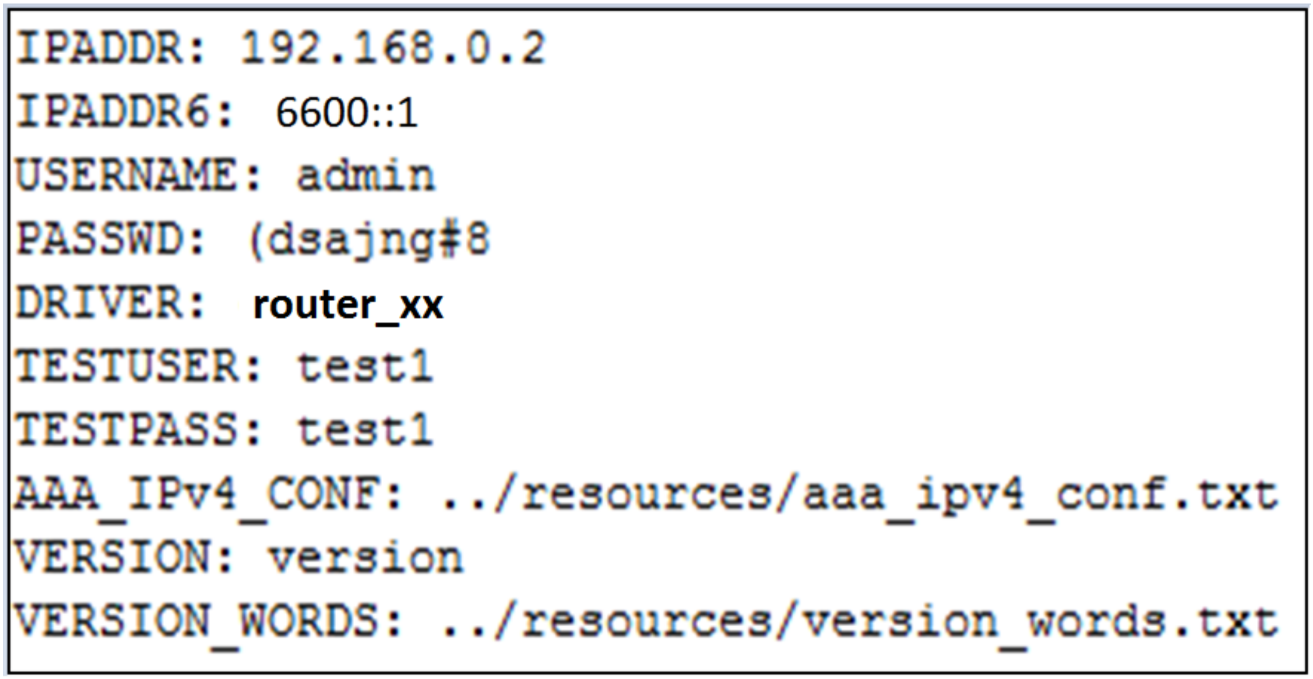
API

- Running code via code!
- API functions to run tests along with options
- Includes all basic tools such as rebot, libdoc, tidy
- Retrieve results
- Customize reports HTML/XML format
- Use standard libraries with API
- Specify variables and resources

Example 3 - Framework

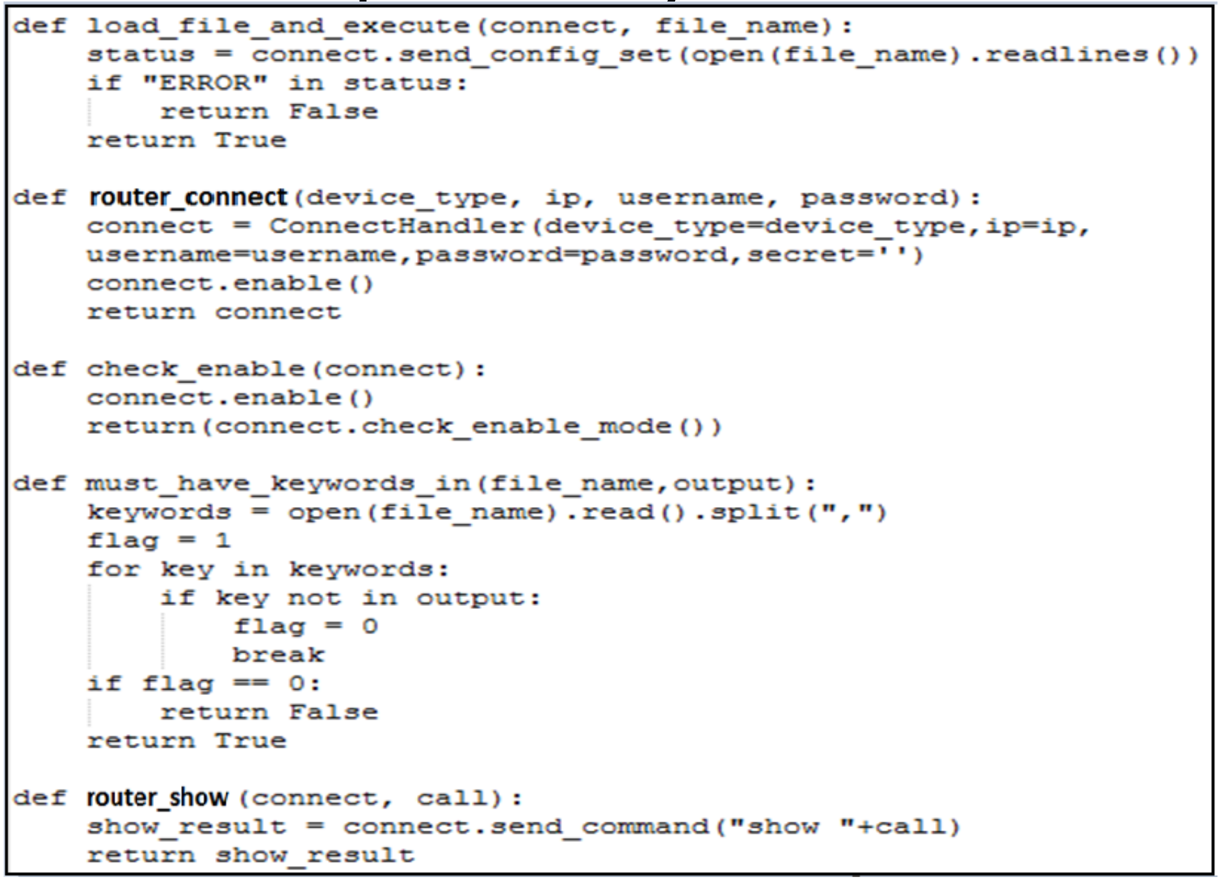

```
Test: Load AAA-IPv4 Configuration
    [Documentation]     If file present, load into running-config
    File Should Exist         ${AAA_IPv4_CONF}
    ${status} =               load_file_and_execute     ${connection_en}    ${AAA_IPv4_CONF}
    Should be Equal           ${status}        ${TRUE}

Test: Verify AAA-IPv4 (test1 account)
    [Documentation]      Ensure `test1` account does not have configuration privilege
    ${connection_test} =      router_connect      ${DRIVER}     ${IPADDR}     ${TESTUSER}     ${TESTPASS}
    ${enable_status}          check_enable      ${connection_test}
    Should be Equal           ${enable status}    ${FALSE}
    ${output}                 router_show       ${connection_test}    ${VERSION}
    ${status} =               must_have_keywords_in     ${VERSION_WORDS}     ${output}
    Should be Equal           ${status}        ${TRUE}
```

Example 3 - Variables



```
IPADDR: 192.168.0.2
IPADDR6: 6600::1
USERNAME: admin
PASSWD: (dsajng#8
DRIVER: router_xx
TESTUSER: test1
TESTPASS: test1
AAA_IPv4_CONF: ../resources/aaa_ipv4_conf.txt
VERSION: version
VERSION_WORDS: ../resources/version_words.txt
```

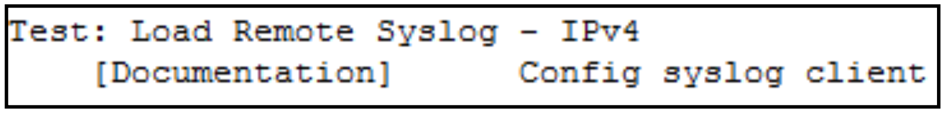
Example 3 – Python Code



```
def load_file_and_execute(connect, file_name):
    status = connect.send_config_set(open(file_name).readlines())
    if "ERROR" in status:
        return False
    return True

def router_connect(device_type, ip, username, password):
    connect = ConnectHandler(device_type=device_type,ip=ip,
    username=username,password=password,secret='')
    connect.enable()
    return connect

def check_enable(connect):
    connect.enable()
    return(connect.check_enable_mode())

def must_have_keywords_in(file_name,output):
    keywords = open(file_name).read().split(",")
    flag = 1
    for key in keywords:
        if key not in output:
            flag = 0
            break
    if flag == 0:
        return False
    return True

def router_show (connect, call):
    show_result = connect.send_command("show "+call)
    return show_result
```

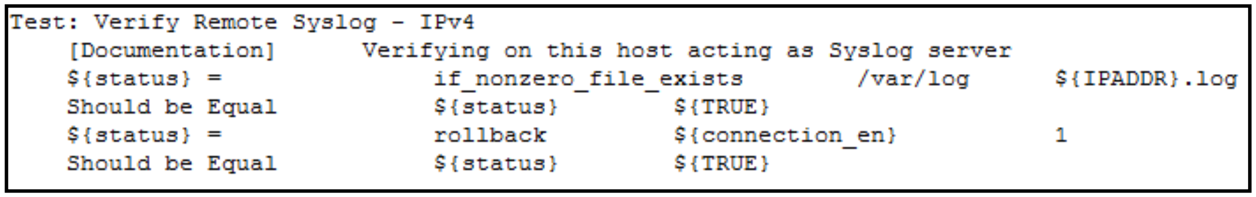
Dos and Don'ts

- Dos
  - Documentation (options)
  - Short & easy naming
  - What, not how



```
Test: Load Remote Syslog - IPv4
    [Documentation]      Config syslog client
```

  - Tabular uniformity



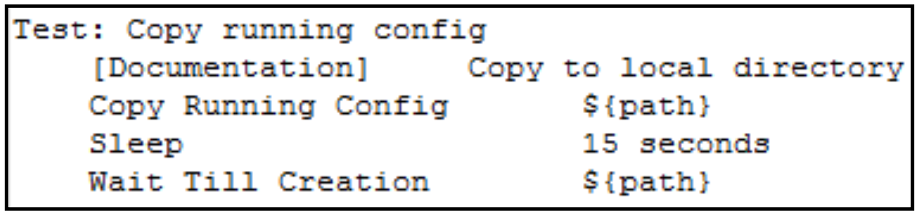
```
Test: Verify Remote Syslog - IPv4
    [Documentation]      Verifying on this host acting as Syslog server
    ${status} =               if_nonzero_file_exists       /var/log      ${IPADDR}.log
    Should be Equal           ${status}        ${TRUE}
    ${status} =               rollback         ${connection_en}              1
    Should be Equal           ${status}        ${TRUE}
```

  - Generic and simple framework

Dos and Don’ts

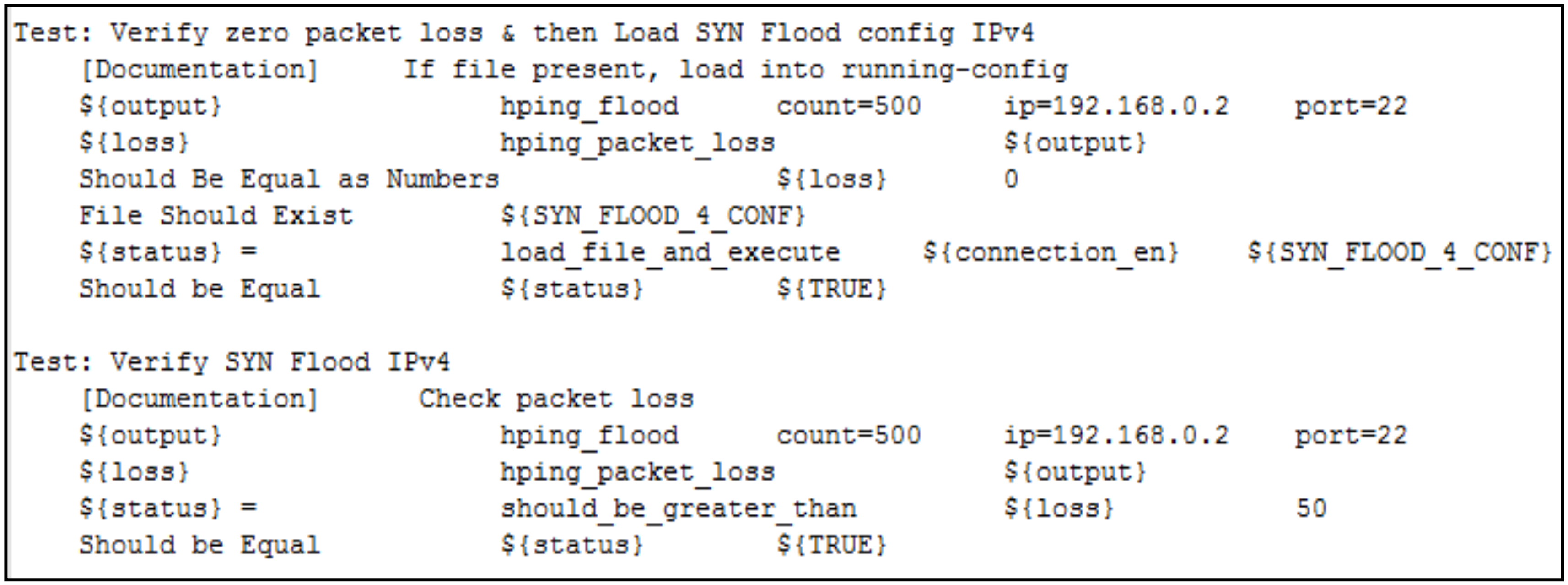
- Dos
  - Logic in code
  - Data driven
  - Checks
  - Syntax (Given, When, Then)

- Don’ts
  - Dependencies
  - Granular test
  - Hardcoded variables
  - Sleeping in place of polling

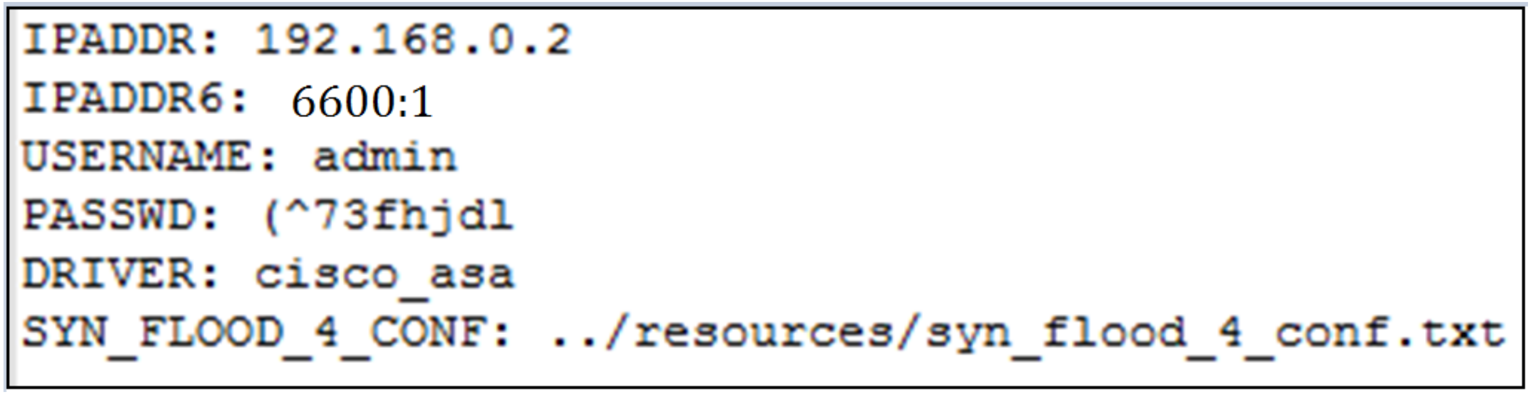


```
Test: Copy running config
    [Documentation]     Copy to local directory
    Copy Running Config       ${path}
    Sleep                     15 seconds
    Wait Till Creation        ${path}
```

Example 4 - Framework

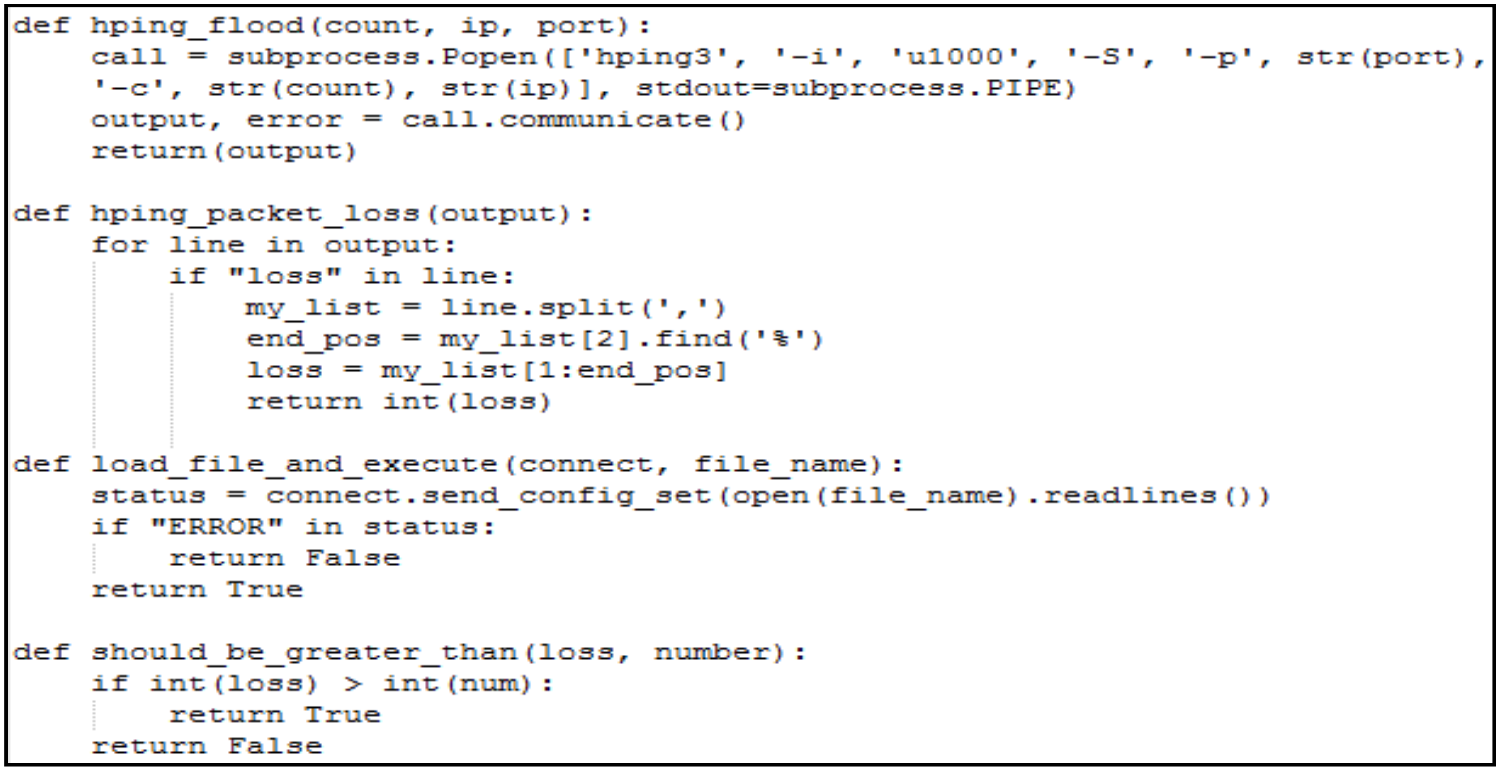

```
Test: Verify zero packet loss & then Load SYN Flood config IPv4
    [Documentation]     If file present, load into running-config
    ${output}                hping_flood      count=500     ip=192.168.0.2    port=22
    ${loss}                  hping_packet_loss              ${output}
    Should Be Equal as Numbers                ${loss}       0
    File Should Exist        ${SYN_FLOOD_4_CONF}
    ${status} =              load_file_and_execute     ${connection_en}    ${SYN_FLOOD_4_CONF}
    Should be Equal          ${status}        ${TRUE}

Test: Verify SYN Flood IPv4
    [Documentation]      Check packet loss
    ${output}                hping_flood      count=500     ip=192.168.0.2    port=22
    ${loss}                  hping_packet_loss              ${output}
    ${status} =              should_be_greater_than         ${loss}           50
    Should be Equal          ${status}        ${TRUE}
```

Example 4 - Variables

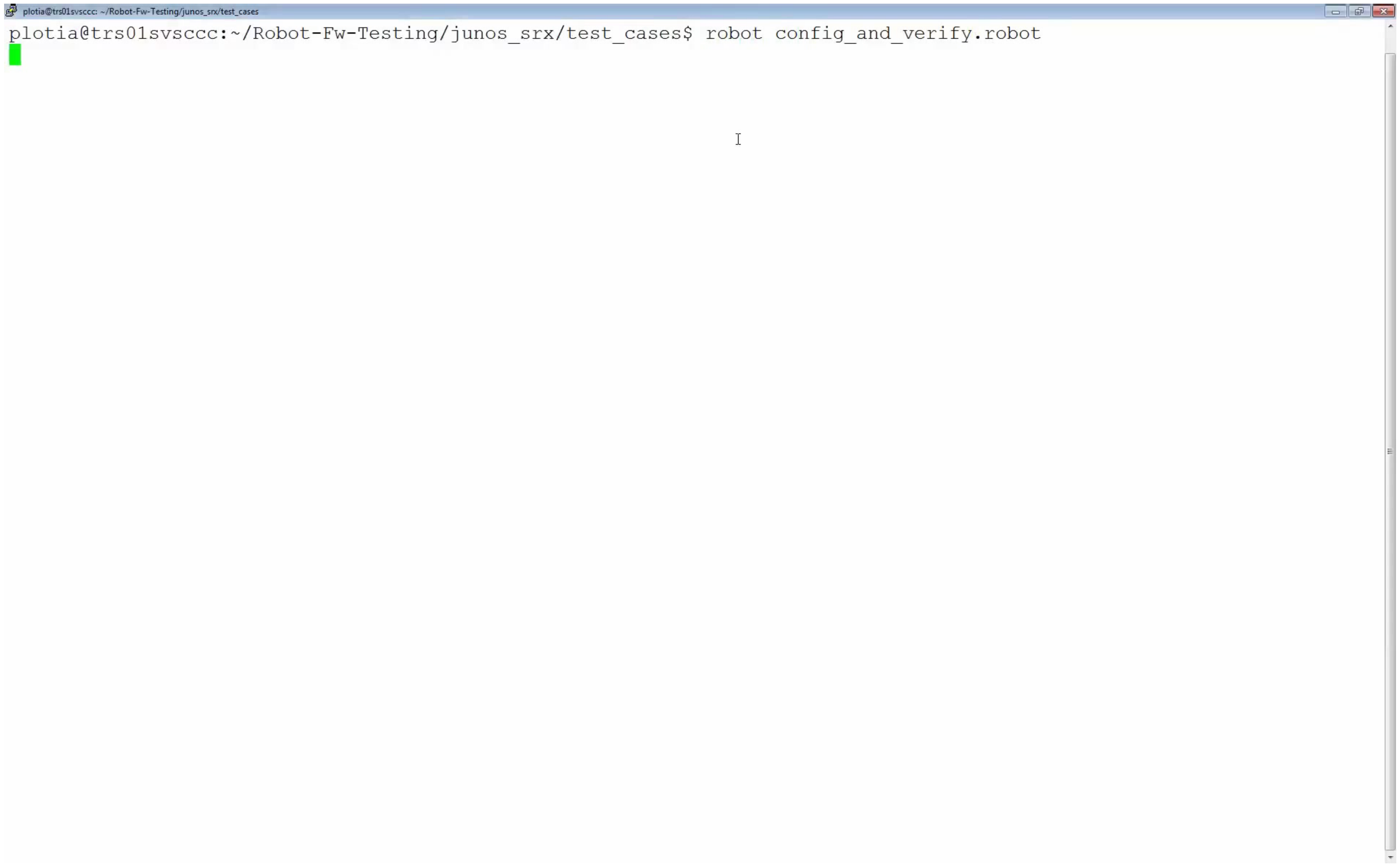


```
IPADDR: 192.168.0.2
IPADDR6: 6600:1
USERNAME: admin
PASSWD: (^73fhjdl
DRIVER: cisco_asa
SYN_FLOOD_4_CONF: ../resources/syn_flood_4_conf.txt
```

Example 4 – Python Code



```
def hping_flood(count, ip, port):
    call = subprocess.Popen(['hping3', '-i', 'u1000', '-S', '-p', str(port),
    '-c', str(count), str(ip)], stdout=subprocess.PIPE)
    output, error = call.communicate()
    return(output)

def hping_packet_loss(output):
    for line in output:
        if "loss" in line:
            my_list = line.split(',')
            end_pos = my_list[2].find('%')
            loss = my_list[1:end_pos]
            return int(loss)

def load_file_and_execute(connect, file_name):
    status = connect.send_config_set(open(file_name).readlines())
    if "ERROR" in status:
        return False
    return True

def should_be_greater_than(loss, number):
    if int(loss) > int(num):
        return True
    return False
```



plotia@trs01svsccc: ~/Robot-Fw-Testing/junos_srx/test_cases

```
plotia@trs01svsccc:~/Robot-Fw-Testing/junos_srx/test_cases$ robot config_and_verify.robot
```

Summary

- Robot Automation Framework provides several use case scenarios for network automation
- Keyword based acceptance driven tests
- Reuse generic test libraries
- Separation of components allow customization and ease of understanding
- Simplify automation of workflows

Resources

- https://github.com/robotframework/robotframework/blob/master/INSTALL.rst
- http://www.slideshare.net/pekkaklarck/robot-framework-introduction
- https://github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst
- http://robotframework.org/robotframework/#user-guide
- https://github.com/robotframework/HowToWriteGoodTestCases/blob/master/HowToWriteGoodTestCases.rst
- http://robotframework.org/robotframework/#standard-libraries
- https://robot-framework.readthedocs.io/en/latest/

Charter COMMUNICATIONS Spectrum

pratik.lotia@charter.com

@pratiklotia

Thank You

- Questions?

Backup slides