San Jose State University
Department of Electrical Engineering
EE104, Fall 2023, Pham

Hardware Test Framework
Google's Spintop OpenHTF

Contents

Hardware Test Module

This module will leverage Google's Open-source Hardware Testing Framework (OpenHTF) with an additional Spintop layer on top of the OpenHTF.

Spintop-OpenHTF is an opinionated fork of OpenHTF to bring a more standard approach to hardware testbench development.

References

- https://spintop-openhtf.readthedocs.io/en/latest/index.html
- https://github.com/google/openhtf
- https://www.youtube.com/watch?app=desktop&v=bC5YhAo1kHc&ab_channel=GoogleTechTalks

Test Engineering Intro

- https://www.youtube.com/watch?v=1QVF04l1eCI&ab_channel=LifeatGoogle
- https://www.youtube.com/watch?v=YUDKYqzomgg&ab_channel=Uplatz

Overview

OpenHTF is a Python library that provides a set of convenient abstractions designed to remove as much boilerplate as possible from hardware test setup and execution, so test engineers can focus primarily on test logic. It aspires to do so in a lightweight and minimalistic fashion. It is general enough to be useful in a variety of hardware testing scenarios, from the lab bench to the manufacturing floor.

We will go over 3 parts:

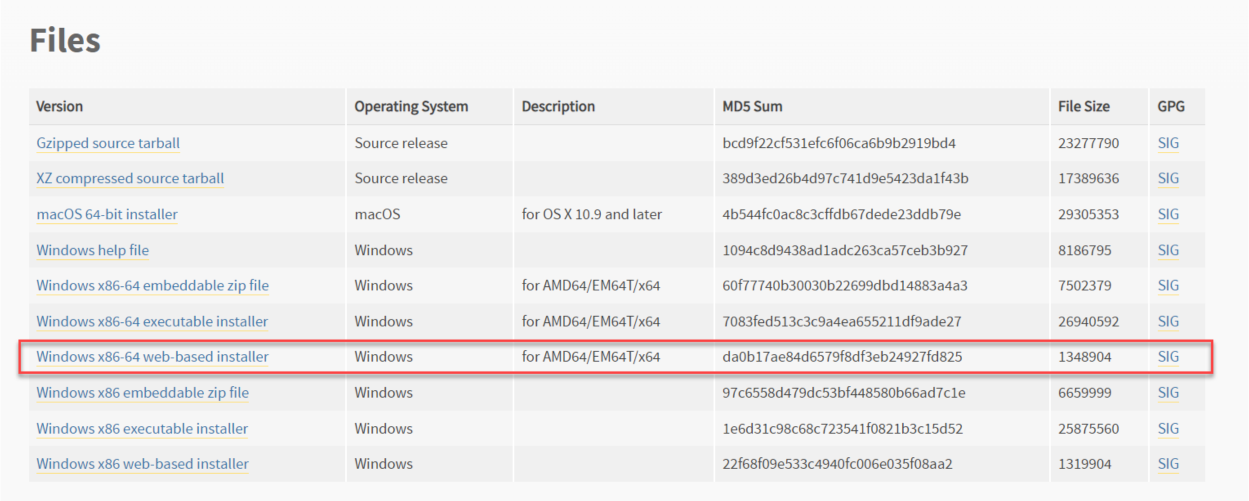
Part 1: Installations

Part 2: Run the tutorials to get acquaintance to the framework and how to write a test

Part 3: Develop your own hardware tests

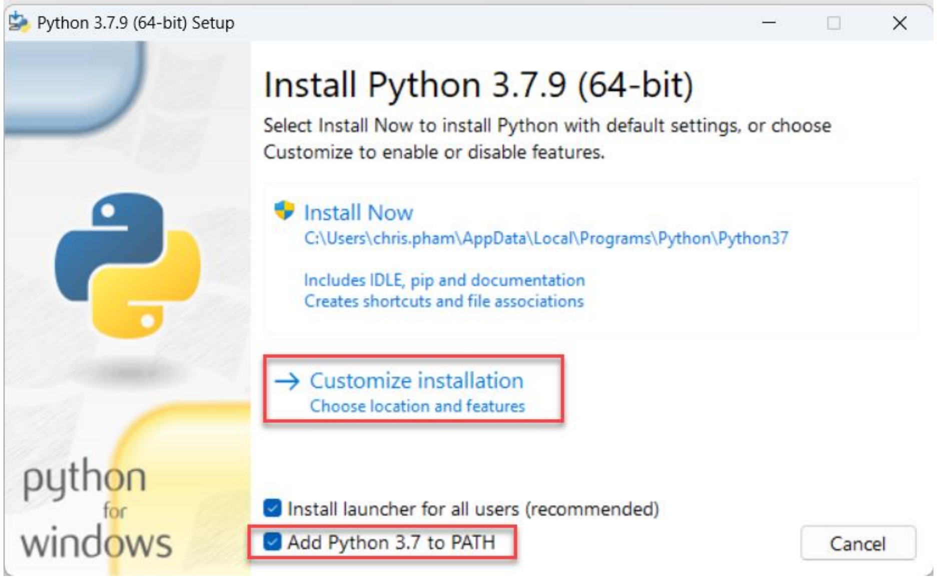
PART 1: Install OpenHTF

```
# In Part 1, you will follow the steps below to  install Python 3.7.9, create a virtual environment &
#  upgrade pip and pyopenssl, install OpenHTF, then install Spintop-OpenHTF.
# Note that OpenHTF only works for python version >=3.7.0 and <3.8.0

====================================
#STEP 1: Install Python 3.7.9 from https://www.python.org/downloads/release/python-379/
# You can use the web-based installer
# Let's do Customize Installation and install it to C:\Python379 for simplicity
```



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		bcd9f22cf531efc6f06ca6b9b2919bd4	23277790	SIG
XZ compressed source tarball	Source release		389d3ed26b4d97c741d9e5423da1f43b	17389636	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	4b544fc0ac8c3cffdb67dede23ddb79e	29305353	SIG
Windows help file	Windows		1094c8d9438ad1adc263ca57ceb3b927	8186795	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	60f77740b30030b22699dbd14883a4a3	7502379	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	7083fed513c3c9a4ea655211df9ade27	26940592	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	da0b17ae84d6579f8df3eb24927fd825	1348904	SIG
Windows x86 embeddable zip file	Windows		97c6558d479dc53bf448580b66ad7c1e	6659999	SIG
Windows x86 executable installer	Windows		1e6d31c98c68c723541f0821b3c15d52	25875560	SIG
Windows x86 web-based installer	Windows		22f68f09e533c4940fc006e035f08aa2	1319904	SIG



Python 3.7.9 (64-bit) Setup

Install Python 3.7.9 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

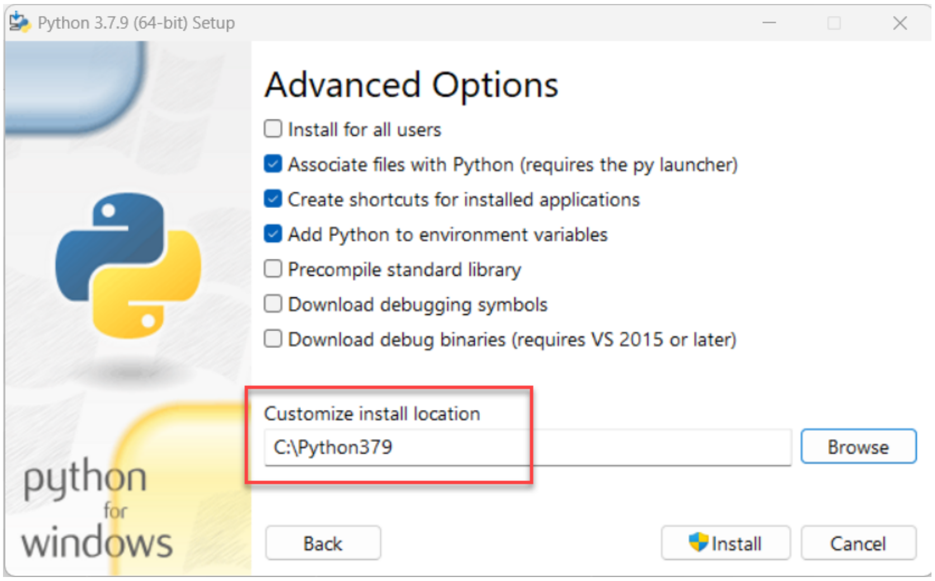
Install Now
C:\Users\chris.pham\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)

☑ Add Python 3.7 to PATH



Python 3.7.9 (64-bit) Setup

Advanced Options

☐ Install for all users

☑ Associate files with Python (requires the py launcher)

☑ Create shortcuts for installed applications

☑ Add Python to environment variables

☐ Precompile standard library

☐ Download debugging symbols

☐ Download debug binaries (requires VS 2015 or later)

Customize install location
C:\Python379

====================================

#STEP 2: pip install * everything that you need in this new C:\Python379 folder
.\Script\pip install <package_name>
Where <package_name> can be numpy and other packages that you have been using.

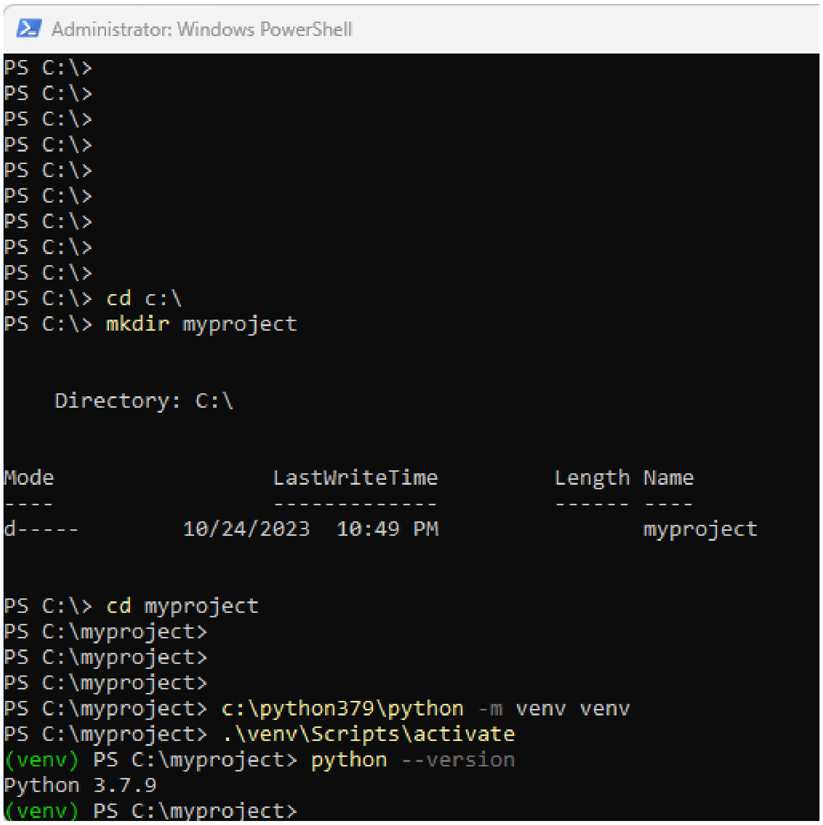
====================================

#STEP 3: Create a Virtual Environment & upgrade pip
#Follow steps 1 and 2 from https://pypi.org/project/spintop-openhtf/

#Create Folder from C:\
cd c:\
mkdir myproject
cd myproject

#Creates new venv in the folder 'venv' using Python3.7.9
# Notice this command: C:\Python379\python -m venv venv
C:\Python379\python -m venv venv
venv\Scripts\activate
 #Note: if you want to deactivate the venv, just type "deactivate" at the command line
 # or your commands are still running in the background in this virtual environment.

#Optional: Type this command to verify that you are indeed running Python 3.7.9
# (venv) PS C:\myproject> python --version
# Python 3.7.9

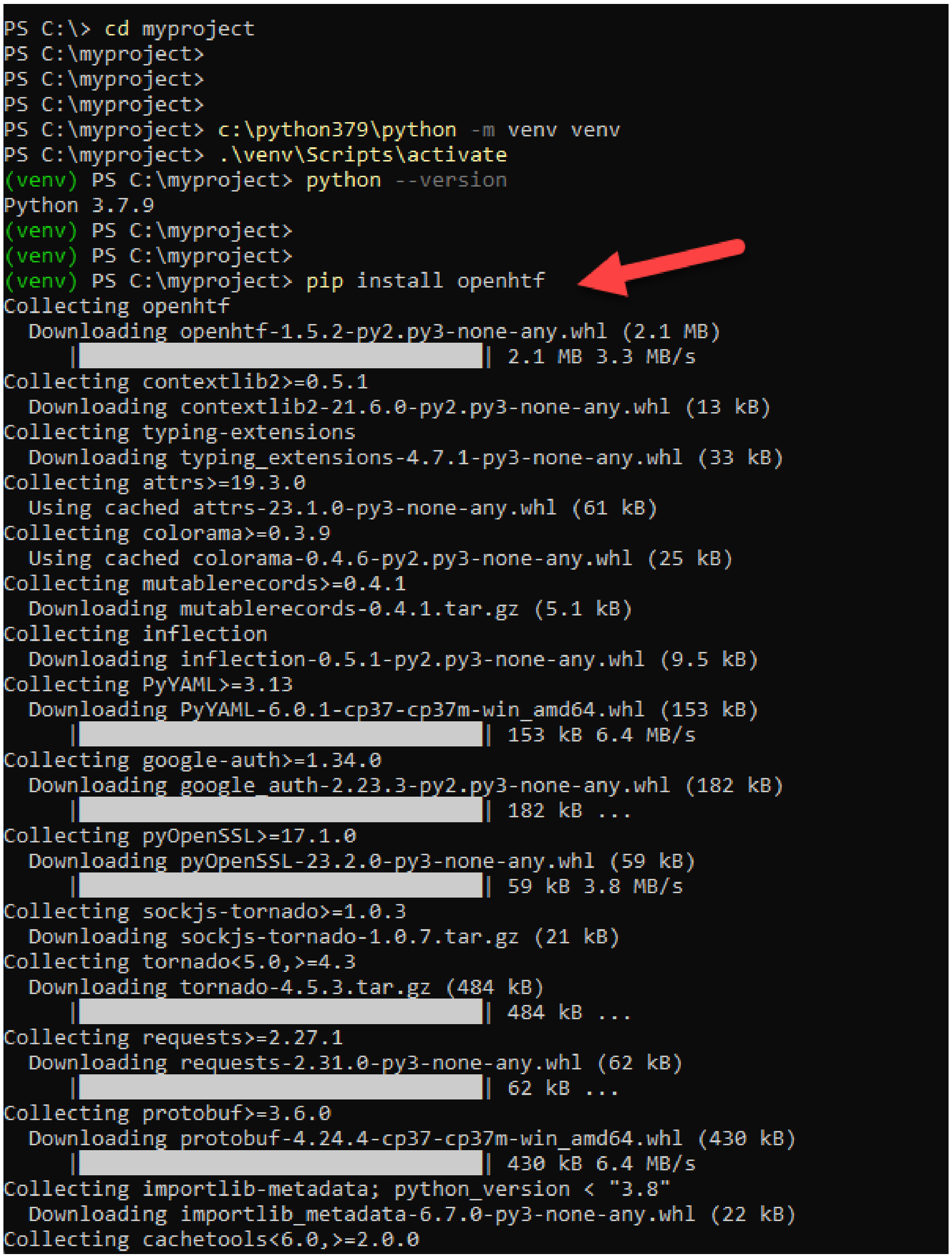


```
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\>
PS C:\> cd c:\
PS C:\> mkdir myproject


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/24/2023  10:49 PM                myproject


PS C:\> cd myproject
PS C:\myproject>
PS C:\myproject>
PS C:\myproject>
PS C:\myproject> c:\python379\python -m venv venv
PS C:\myproject> .\venv\Scripts\activate
(venv) PS C:\myproject> python --version
Python 3.7.9
(venv) PS C:\myproject>
```

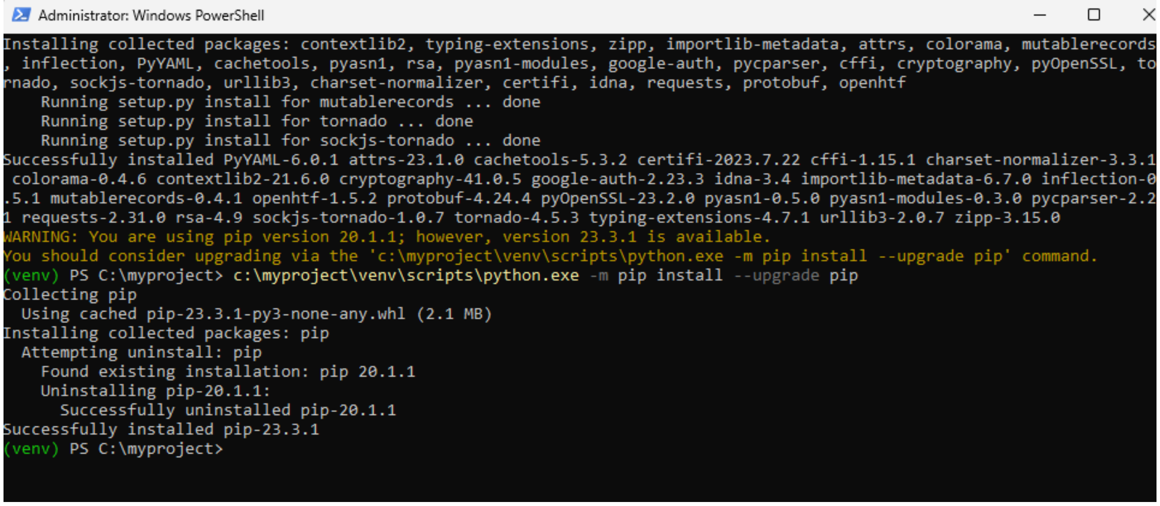
====================================

#STEP 4: Install OpenHTF
#Follow instructions from https://github.com/google/openhtf

pip install openhtf



```
PS C:\> cd myproject
PS C:\myproject>
PS C:\myproject>
PS C:\myproject>
PS C:\myproject> c:\python379\python -m venv venv
PS C:\myproject> .\venv\Scripts\activate
(venv) PS C:\myproject> python --version
Python 3.7.9
(venv) PS C:\myproject>
(venv) PS C:\myproject>
(venv) PS C:\myproject> pip install openhtf
Collecting openhtf
  Downloading openhtf-1.5.2-py2.py3-none-any.whl (2.1 MB)
     |████████████████████████████████| 2.1 MB 3.3 MB/s
Collecting contextlib2>=0.5.1
  Downloading contextlib2-21.6.0-py2.py3-none-any.whl (13 kB)
Collecting typing-extensions
  Downloading typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Collecting attrs>=19.3.0
  Using cached attrs-23.1.0-py3-none-any.whl (61 kB)
Collecting colorama>=0.3.9
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting mutablerecords>=0.4.1
  Downloading mutablerecords-0.4.1.tar.gz (5.1 kB)
Collecting inflection
  Downloading inflection-0.5.1-py2.py3-none-any.whl (9.5 kB)
Collecting PyYAML>=3.13
  Downloading PyYAML-6.0.1-cp37-cp37m-win_amd64.whl (153 kB)
     |████████████████████████████████| 153 kB 6.4 MB/s
Collecting google-auth>=1.34.0
  Downloading google_auth-2.23.3-py2.py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB ...
Collecting pyOpenSSL>=17.1.0
  Downloading pyOpenSSL-23.2.0-py3-none-any.whl (59 kB)
     |████████████████████████████████| 59 kB 3.8 MB/s
Collecting sockjs-tornado>=1.0.3
  Downloading sockjs-tornado-1.0.7.tar.gz (21 kB)
Collecting tornado<5.0,>=4.3
  Downloading tornado-4.5.3.tar.gz (484 kB)
     |████████████████████████████████| 484 kB ...
Collecting requests>=2.27.1
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB ...
Collecting protobuf>=3.6.0
  Downloading protobuf-4.24.4-cp37-cp37m-win_amd64.whl (430 kB)
     |████████████████████████████████| 430 kB 6.4 MB/s
Collecting importlib-metadata; python_version < "3.8"
  Downloading importlib_metadata-6.7.0-py3-none-any.whl (22 kB)
Collecting cachetools<6.0,>=2.0.0
```

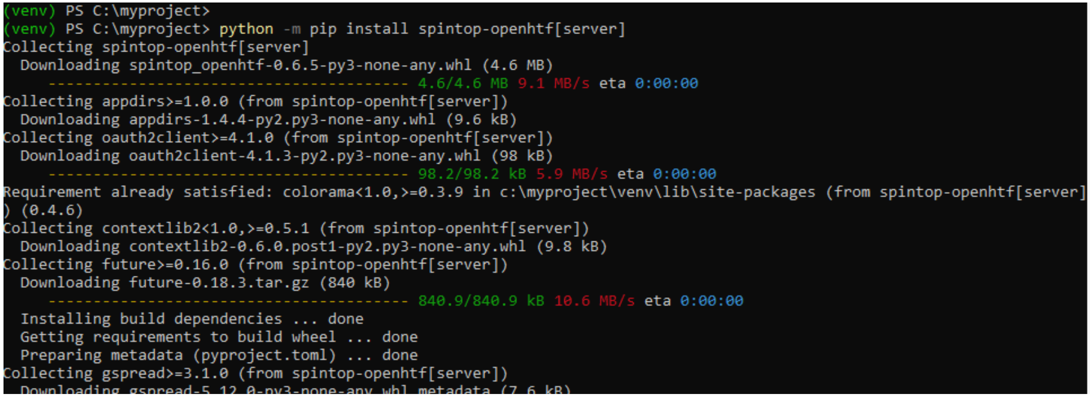


```
Installing collected packages: contextlib2, typing-extensions, zipp, importlib-metadata, attrs, colorama, mutablerecords
, inflection, PyYAML, cachetools, pyasn1, rsa, pyasn1-modules, google-auth, pycparser, cffi, cryptography, pyOpenSSL, to
rnado, sockjs-tornado, urllib3, charset-normalizer, certifi, idna, requests, protobuf, openhtf
    Running setup.py install for mutablerecords ... done
    Running setup.py install for tornado ... done
    Running setup.py install for sockjs-tornado ... done
Successfully installed PyYAML-6.0.1 attrs-23.1.0 cachetools-5.3.2 certifi-2023.7.22 cffi-1.15.1 charset-normalizer-3.3.1
 colorama-0.4.6 contextlib2-21.6.0 cryptography-41.0.5 google-auth-2.23.3 idna-3.4 importlib-metadata-6.7.0 inflection-0
.5.1 mutablerecords-0.4.1 openhtf-1.5.2 protobuf-4.24.4 pyOpenSSL-23.2.0 pyasn1-0.5.0 pyasn1-modules-0.3.0 pycparser-2.2
1 requests-2.31.0 rsa-4.9 sockjs-tornado-1.0.7 tornado-4.5.3 typing-extensions-4.7.1 urllib3-2.0.7 zipp-3.15.0
WARNING: You are using pip version 20.1.1; however, version 23.3.1 is available.
You should consider upgrading via the 'c:\myproject\venv\scripts\python.exe -m pip install --upgrade pip' command.
(venv) PS C:\myproject> c:\myproject\venv\scripts\python.exe -m pip install --upgrade pip
Collecting pip
  Using cached pip-23.3.1-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.1.1
    Uninstalling pip-20.1.1:
      Successfully uninstalled pip-20.1.1
Successfully installed pip-23.3.1
(venv) PS C:\myproject>
```

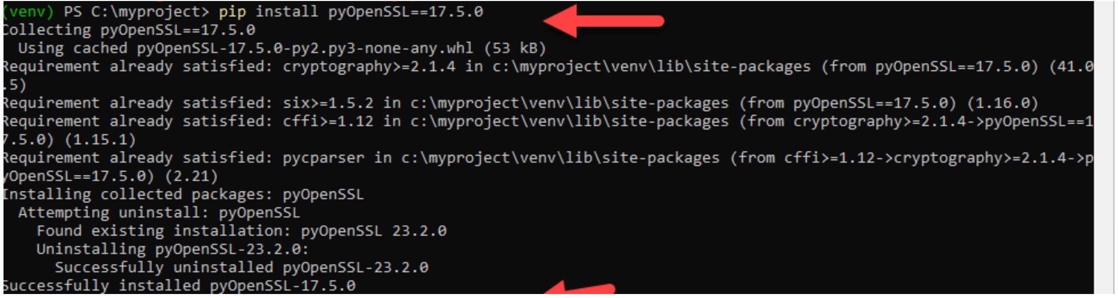
=====================================

#STEP 5: Install Spintop-OpenHTF & upgrade pyopenssl to latest version
#Follow steps 3 from https://pypi.org/project/spintop-openhtf/
python -m pip install spintop-openhtf[server]

pip install pyOpenSSL==17.5.0



```
(venv) PS C:\myproject>
(venv) PS C:\myproject> python -m pip install spintop-openhtf[server]
Collecting spintop-openhtf[server]
  Downloading spintop_openhtf-0.6.5-py3-none-any.whl (4.6 MB)
     ---------------------------------------- 4.6/4.6 MB 9.1 MB/s eta 0:00:00
Collecting appdirs>=1.0.0 (from spintop-openhtf[server])
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting oauth2client>=4.1.0 (from spintop-openhtf[server])
  Downloading oauth2client-4.1.3-py2.py3-none-any.whl (98 kB)
     ---------------------------------------- 98.2/98.2 kB 5.9 MB/s eta 0:00:00
Requirement already satisfied: colorama<1.0,>=0.3.9 in c:\myproject\venv\lib\site-packages (from spintop-openhtf[server]
) (0.4.6)
Collecting contextlib2<1.0,>=0.5.1 (from spintop-openhtf[server])
  Downloading contextlib2-0.6.0.post1-py2.py3-none-any.whl (9.8 kB)
Collecting future>=0.16.0 (from spintop-openhtf[server])
  Downloading future-0.18.3.tar.gz (840 kB)
     ---------------------------------------- 840.9/840.9 kB 10.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting gspread>=3.1.0 (from spintop-openhtf[server])
  Downloading gspread-5.12.0-py3-none-any.whl.metadata (7.6 kB)
```



```
(venv) PS C:\myproject> pip install pyOpenSSL==17.5.0
Collecting pyOpenSSL==17.5.0
  Using cached pyOpenSSL-17.5.0-py2.py3-none-any.whl (53 kB)
Requirement already satisfied: cryptography>=2.1.4 in c:\myproject\venv\lib\site-packages (from pyOpenSSL==17.5.0) (41.0
.5)
Requirement already satisfied: six>=1.5.2 in c:\myproject\venv\lib\site-packages (from pyOpenSSL==17.5.0) (1.16.0)
Requirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib\site-packages (from cryptography>=2.1.4->pyOpenSSL==1
7.5.0) (1.15.1)
Requirement already satisfied: pycparser in c:\myproject\venv\lib\site-packages (from cffi>=1.12->cryptography>=2.1.4->p
yOpenSSL==17.5.0) (2.21)
Installing collected packages: pyOpenSSL
  Attempting uninstall: pyOpenSSL
    Found existing installation: pyOpenSSL 23.2.0
    Uninstalling pyOpenSSL-23.2.0:
      Successfully uninstalled pyOpenSSL-23.2.0
Successfully installed pyOpenSSL-17.5.0
```

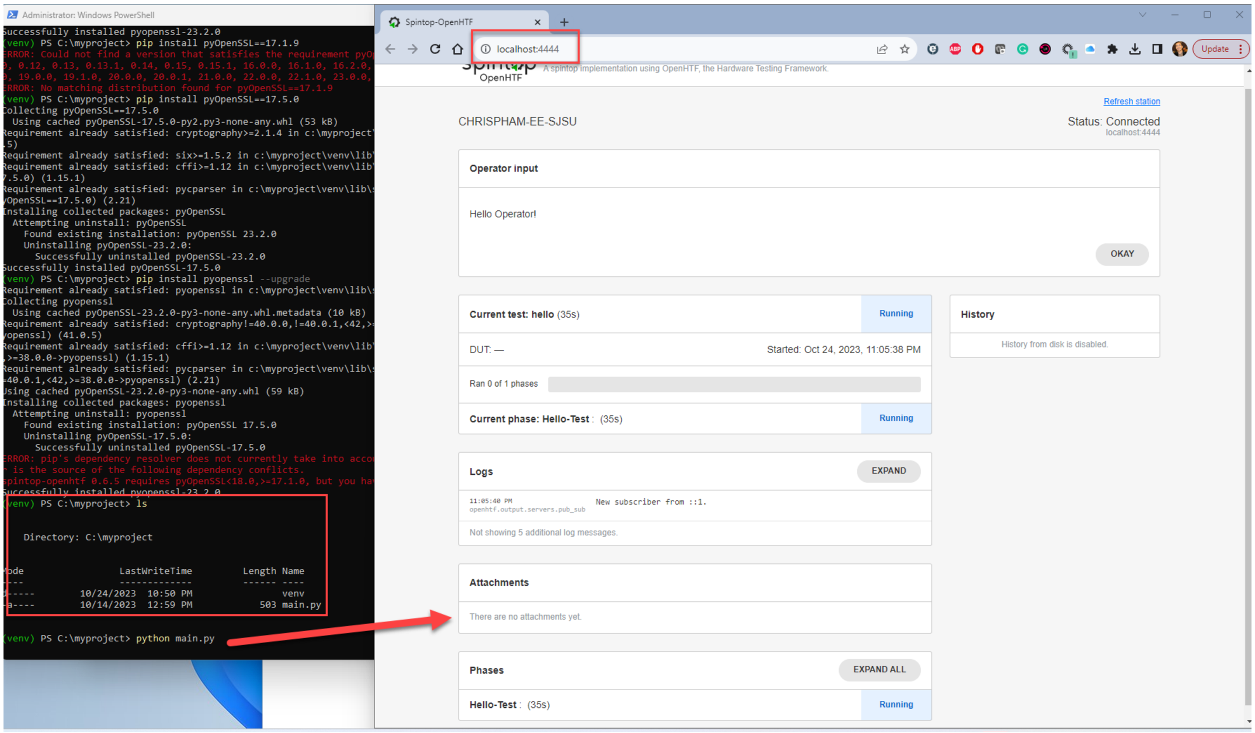
# Now you are ready to run some test tutorials and learn from them.

PART 2: Run the Tutorials

```
=========================================================
#TUTORIAL 1: First Testbench Tutorial & Web Interface Tutorial
# from https://spintop-openhtf.readthedocs.io/en/latest/docs/first-testbench.html
# and from https://spintop-openhtf.readthedocs.io/en/latest/docs/web-app/ref.html
=========================================================
#Create file main.py below, and save it to C:\myproject folder
```

```python
# main.py
import os
import openhtf as htf
from openhtf.plugs.user_input import UserInput
from spintop_openhtf import TestPlan

""" Test Plan """

# This defines the name of the testbench.
plan = TestPlan('hello')

@plan.testcase('Hello-Test')
@plan.plug(prompts=UserInput)
def hello_world(test, prompts):
   prompts.prompt('Hello Operator!')
   test.dut_id = 'hello' # Manually set the DUT Id to same value every test

if __name__ == '__main__':
   plan.no_trigger()
   plan.run()
```

```
#-------------------
# From the C:\myproject folder, run this command to test the GUI interface and run a Hello-Test
(venv) PS C:\myproject> python main.py

# You will have to respond YES to allow the firewall to passthrough your python access.
# You will see a new browser pops up with URL  "localhost:4444"
# Run the test several times by pressing OKAY key for a few times and observe the browser GUI interface,
#  and the console outputs.
# When you are done with observations, press "CTRL-C" keys from the PowerShell console to quit the test.
```



Administrator: Windows PowerShell

```
Successfully installed pyopenssl-23.2.0
(venv) PS C:\myproject> pip install pyOpenSSL==17.1.9
ERROR: Could not find a version that satisfies the requirement pyO
0, 0.12, 0.13, 0.13.1, 0.14, 0.15, 0.15.1, 16.0.0, 16.1.0, 16.2.0,
0, 19.0.0, 19.1.0, 20.0.0, 20.0.1, 21.0.0, 22.0.0, 22.1.0, 23.0.0,
ERROR: No matching distribution found for pyOpenSSL==17.1.9
(venv) PS C:\myproject> pip install pyOpenSSL==17.5.0
Collecting pyOpenSSL==17.5.0
  Using cached pyOpenSSL-17.5.0-py2.py3-none-any.whl (53 kB)
Requirement already satisfied: cryptography>=2.1.4 in c:\myproject
.5)
Requirement already satisfied: six>=1.5.2 in c:\myproject\venv\lib
Requirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib
7.5.0) (1.15.1)
Requirement already satisfied: pycparser in c:\myproject\venv\lib\
yOpenSSL==17.5.0) (2.21)
Installing collected packages: pyOpenSSL
  Attempting uninstall: pyOpenSSL
    Found existing installation: pyOpenSSL 23.2.0
    Uninstalling pyOpenSSL-23.2.0:
      Successfully uninstalled pyOpenSSL-23.2.0
Successfully installed pyOpenSSL-17.5.0
(venv) PS C:\myproject> pip install pyopenssl --upgrade
Requirement already satisfied: pyopenssl in c:\myproject\venv\lib\
Collecting pyopenssl
  Using cached pyOpenSSL-23.2.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: cryptography!=40.0.0,!=40.0.1,<42,>
yopenssl) (41.0.5)
Requirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib
,>=38.0.0->pyopenssl) (1.15.1)
Requirement already satisfied: pycparser in c:\myproject\venv\lib\
=40.0.1,<42,>=38.0.0->pyopenssl) (2.21)
Using cached pyOpenSSL-23.2.0-py3-none-any.whl (59 kB)
Installing collected packages: pyopenssl
  Attempting uninstall: pyopenssl
    Found existing installation: pyOpenSSL 17.5.0
    Uninstalling pyOpenSSL-17.5.0:
      Successfully uninstalled pyOpenSSL-17.5.0
ERROR: pip's dependency resolver does not currently take into acco
r is the source of the following dependency conflicts.
spintop-openhtf 0.6.5 requires pyOpenSSL<18.0,>=17.1.0, but you ha
Successfully installed pyopenssl-23.2.0
(venv) PS C:\myproject> ls


    Directory: C:\myproject


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/24/2023  10:50 PM                venv
-a----        10/14/2023  12:59 PM            503 main.py


(venv) PS C:\myproject> python main.py
```

A spintop implementation using OpenHTF, the Hardware Testing Framework.

Refresh station

CHRISPHAM-EE-SJSU

Status: Connected
localhost:4444

Operator input

Hello Operator!

OKAY

Current test: hello (35s) — Running

DUT: — Started: Oct 24, 2023, 11:05:38 PM

Ran 0 of 1 phases

Current phase: Hello-Test : (35s) — Running

History

History from disk is disabled.

Logs EXPAND

```
11:05:40 PM                          New subscriber from ::1.
openhtf.output.servers.pub_sub
```

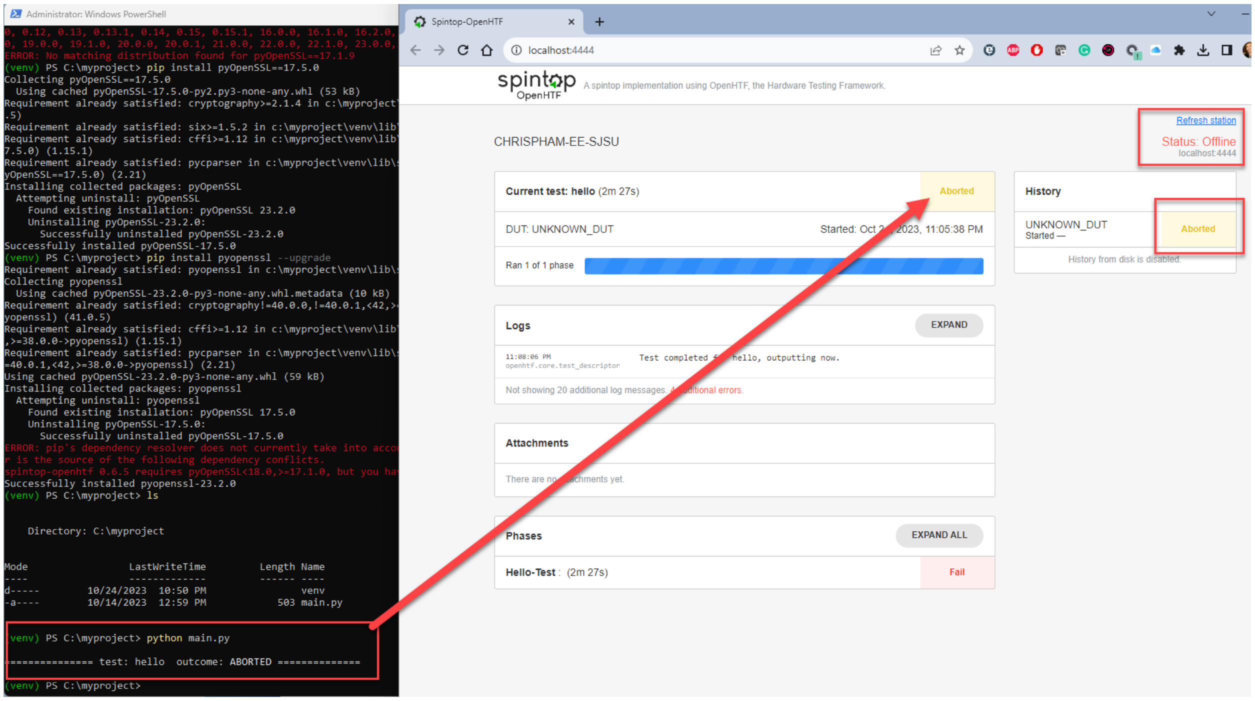
Not showing 5 additional log messages.

Attachments

There are no attachments yet.

Phases EXPAND ALL

Hello-Test : (35s) — Running



Administrator: Windows PowerShell

```
0, 0.12, 0.13, 0.13.1, 0.14, 0.15, 0.15.1, 16.0.0, 16.1.0, 16.2.0,
0, 19.0.0, 19.1.0, 20.0.0, 20.0.1, 21.0.0, 22.0.0, 22.1.0, 23.0.0,
ERROR: No matching distribution found for pyOpenSSL==17.1.9
(venv) PS C:\myproject> pip install pyOpenSSL==17.5.0
Collecting pyOpenSSL==17.5.0
  Using cached pyOpenSSL-17.5.0-py2.py3-none-any.whl (53 kB)
Requirement already satisfied: cryptography>=2.1.4 in c:\myproject
.5)
Requirement already satisfied: six>=1.5.2 in c:\myproject\venv\lib
Requirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib
7.5.0) (1.15.1)
Requirement already satisfied: pycparser in c:\myproject\venv\lib\
yOpenSSL==17.5.0) (2.21)
Installing collected packages: pyOpenSSL
  Attempting uninstall: pyOpenSSL
    Found existing installation: pyOpenSSL 23.2.0
    Uninstalling pyOpenSSL-23.2.0:
      Successfully uninstalled pyOpenSSL-23.2.0
Successfully installed pyOpenSSL-17.5.0
(venv) PS C:\myproject> pip install pyopenssl --upgrade
Requirement already satisfied: pyopenssl in c:\myproject\venv\lib\
Collecting pyopenssl
  Using cached pyOpenSSL-23.2.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: cryptography!=40.0.0,!=40.0.1,<42,>
yopenssl) (41.0.5)
Requirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib
,>=38.0.0->pyopenssl) (1.15.1)
Requirement already satisfied: pycparser in c:\myproject\venv\lib\
=40.0.1,<42,>=38.0.0->pyopenssl) (2.21)
Using cached pyOpenSSL-23.2.0-py3-none-any.whl (59 kB)
Installing collected packages: pyopenssl
  Attempting uninstall: pyopenssl
    Found existing installation: pyOpenSSL 17.5.0
    Uninstalling pyOpenSSL-17.5.0:
      Successfully uninstalled pyOpenSSL-17.5.0
ERROR: pip's dependency resolver does not currently take into acco
r is the source of the following dependency conflicts.
spintop-openhtf 0.6.5 requires pyOpenSSL<18.0,>=17.1.0, but you ha
Successfully installed pyopenssl-23.2.0
(venv) PS C:\myproject> ls


    Directory: C:\myproject


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/24/2023  10:50 PM                venv
-a----        10/14/2023  12:59 PM            503 main.py


(venv) PS C:\myproject> python main.py

============== test: hello  outcome: ABORTED ==============

(venv) PS C:\myproject>
```

A spintop implementation using OpenHTF, the Hardware Testing Framework.

Refresh station

CHRISPHAM-EE-SJSU

Status: Offline
localhost:4444

Current test: hello (2m 27s) — Aborted

DUT: UNKNOWN_DUT Started: Oct 24, 2023, 11:05:38 PM

Ran 1 of 1 phase

History

UNKNOWN_DUT
Started — — Aborted

History from disk is disabled.

Logs EXPAND

```
11:08:06 PM                          Test completed for hello, outputting now.
openhtf.core.test_descriptor
```

Not showing 20 additional log messages. 4 additional errors.

Attachments

There are no attachments yet.

Phases EXPAND ALL

Hello-Test : (2m 27s) — Fail

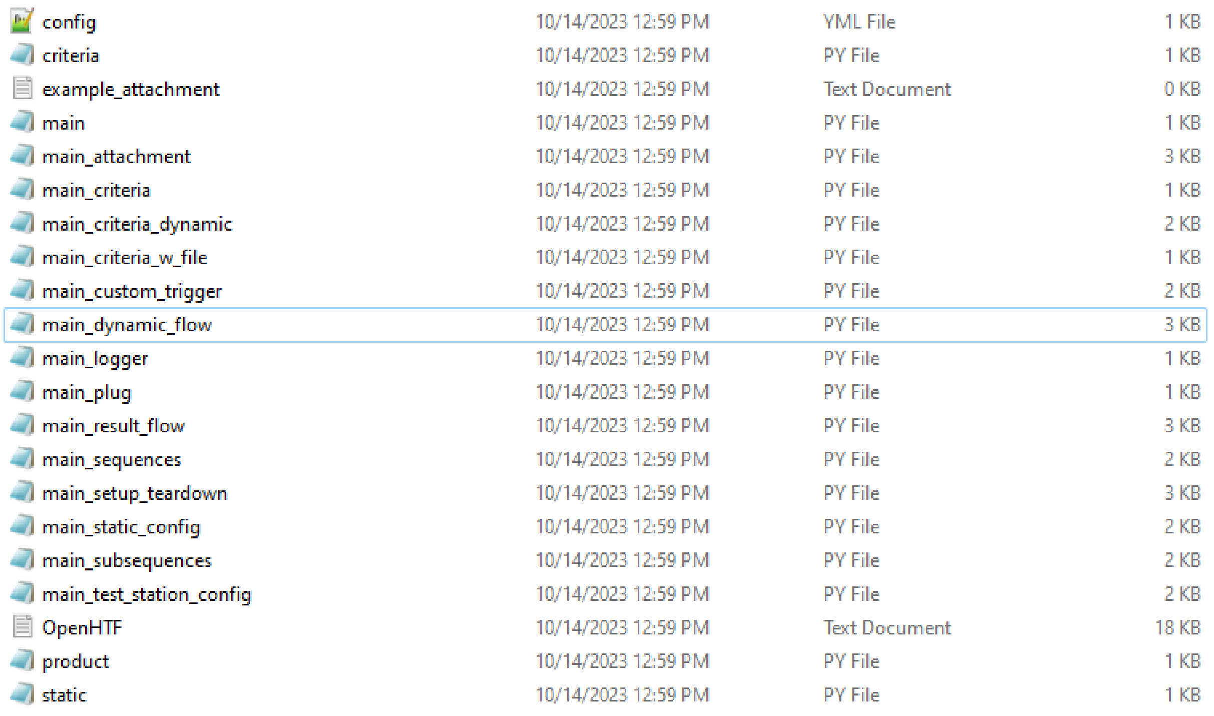
# When you are done with observations, press "CTRL-C" keys from the console to quit the test.

==

#TUTORIAL 2: Forms and Tester Feedback Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/form/ref.html
==

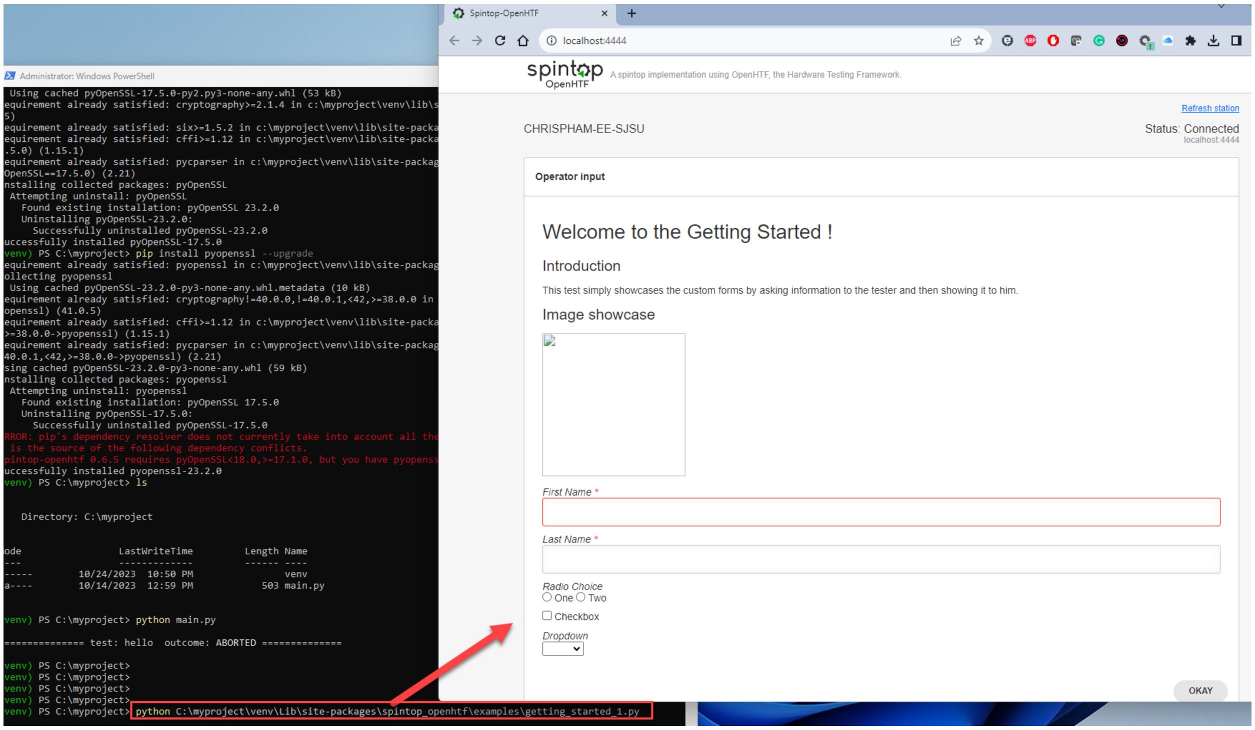
#For simplicity, I have all the source code files in this zip file on Canvas: Spintop_OpenHTF.zip
# Unzip and copy all files to C:\myproject folder
# Notice the config.yml file and static.py file



config	10/14/2023 12:59 PM	YML File	1 KB
criteria	10/14/2023 12:59 PM	PY File	1 KB
example_attachment	10/14/2023 12:59 PM	Text Document	0 KB
main	10/14/2023 12:59 PM	PY File	1 KB
main_attachment	10/14/2023 12:59 PM	PY File	3 KB
main_criteria	10/14/2023 12:59 PM	PY File	1 KB
main_criteria_dynamic	10/14/2023 12:59 PM	PY File	2 KB
main_criteria_w_file	10/14/2023 12:59 PM	PY File	1 KB
main_custom_trigger	10/14/2023 12:59 PM	PY File	2 KB
main_dynamic_flow	10/14/2023 12:59 PM	PY File	3 KB
main_logger	10/14/2023 12:59 PM	PY File	1 KB
main_plug	10/14/2023 12:59 PM	PY File	1 KB
main_result_flow	10/14/2023 12:59 PM	PY File	3 KB
main_sequences	10/14/2023 12:59 PM	PY File	2 KB
main_setup_teardown	10/14/2023 12:59 PM	PY File	3 KB
main_static_config	10/14/2023 12:59 PM	PY File	2 KB
main_subsequences	10/14/2023 12:59 PM	PY File	2 KB
main_test_station_config	10/14/2023 12:59 PM	PY File	2 KB
OpenHTF	10/14/2023 12:59 PM	Text Document	18 KB
product	10/14/2023 12:59 PM	PY File	1 KB
static	10/14/2023 12:59 PM	PY File	1 KB

#Read the webpage carefully: https://spintop-openhtf.readthedocs.io/en/latest/docs/form/ref.ht

#Now execute this command to observe the output and test behavior:
(venv) PS C:\myproject> python C:\myproject\venv\Lib\site-packages\spintop_openhtf\examples\getting_started_1.py
#Learn from the source code and introduce one change at a time to see how the output or behavior changes.


Administrator: Windows PowerShell
Spintop-OpenHTF
localhost:4444
spintop
OpenHTF
A spintop implementation using OpenHTF, the Hardware Testing Framework.
Refresh station
OKAY

```
 Using cached pyOpenSSL-17.5.0-py2.py3-none-any.whl (53 kB)
equirement already satisfied: cryptography>=2.1.4 in c:\myproject\venv\lib\s
5)
equirement already satisfied: six>=1.5.2 in c:\myproject\venv\lib\site-packa
equirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib\site-packa
.5.0) (1.15.1)
equirement already satisfied: pycparser in c:\myproject\venv\lib\site-packag
OpenSSL==17.5.0) (2.21)
nstalling collected packages: pyOpenSSL
 Attempting uninstall: pyOpenSSL
   Found existing installation: pyOpenSSL 23.2.0
   Uninstalling pyOpenSSL-23.2.0:
     Successfully uninstalled pyOpenSSL-23.2.0
uccessfully installed pyOpenSSL-17.5.0
venv) PS C:\myproject> pip install pyopenssl --upgrade
equirement already satisfied: pyopenssl in c:\myproject\venv\lib\site-packag
ollecting pyopenssl
 Using cached pyOpenSSL-23.2.0-py3-none-any.whl.metadata (10 kB)
equirement already satisfied: cryptography!=40.0.0,!=40.0.1,<42,>=38.0.0 in
openssl) (41.0.5)
equirement already satisfied: cffi>=1.12 in c:\myproject\venv\lib\site-packa
>=38.0.0->pyopenssl) (1.15.1)
equirement already satisfied: pycparser in c:\myproject\venv\lib\site-packag
40.0.1,<42,>=38.0.0->pyopenssl) (2.21)
sing cached pyOpenSSL-23.2.0-py3-none-any.whl (59 kB)
nstalling collected packages: pyopenssl
 Attempting uninstall: pyopenssl
   Found existing installation: pyOpenSSL 17.5.0
   Uninstalling pyOpenSSL-17.5.0:
     Successfully uninstalled pyOpenSSL-17.5.0
RROR: pip's dependency resolver does not currently take into account all the
 is the source of the following dependency conflicts.
pintop-openhtf 0.6.5 requires pyOpenSSL<18.0,>=17.1.0, but you have pyopenss
uccessfully installed pyopenssl-23.2.0
venv) PS C:\myproject> ls


   Directory: C:\myproject


ode                 LastWriteTime         Length Name
---                 -------------         ------ ----
-----        10/24/2023  10:50 PM                venv
a----        10/14/2023  12:59 PM            503 main.py


venv) PS C:\myproject> python main.py

============= test: hello  outcome: ABORTED =============

venv) PS C:\myproject>
venv) PS C:\myproject>
venv) PS C:\myproject>
venv) PS C:\myproject>
venv) PS C:\myproject> python C:\myproject\venv\Lib\site-packages\spintop_openhtf\examples\getting_started_1.py
```

CHRISPHAM-EE-SJSU

Status: Connected
localhost:4444

Operator input

Welcome to the Getting Started !

Introduction

This test simply showcases the custom forms by asking information to the tester and then showing it to him.

Image showcase

First Name *

Last Name *

Radio Choice
○ One ○ Two

□ Checkbox

Dropdown

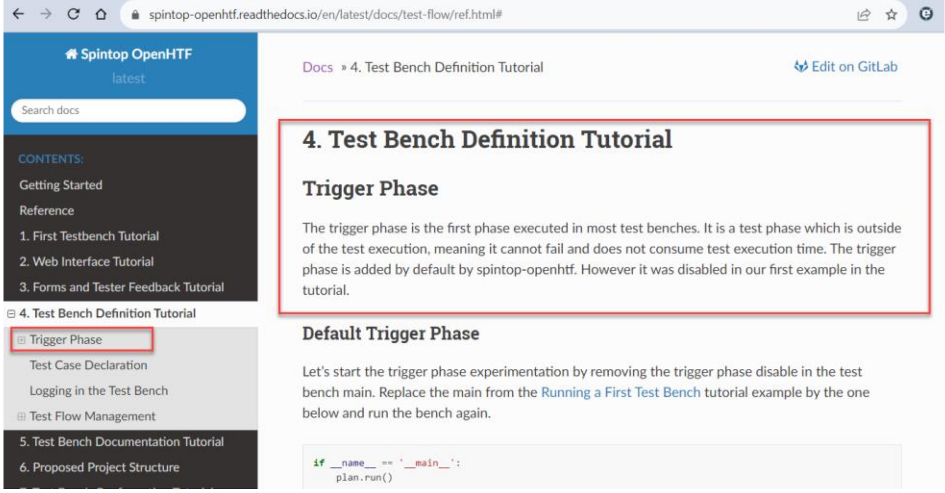
#Now you can implement the tutorial from https://spintop-openhtf.readthedocs.io/en/latest/docs/form/ref.html

===

#TUTORIAL 3: Test Bench Definition Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#
===
#Do the following sub-tutorials:

#++++++++++++++++++++++++++++
#Trigger Phase
https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#trigger-phase


spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#
Spintop OpenHTF
latest
Search docs
CONTENTS:
Getting Started
Reference
1. First Testbench Tutorial
2. Web Interface Tutorial
3. Forms and Tester Feedback Tutorial
4. Test Bench Definition Tutorial
Trigger Phase
Test Case Declaration
Logging in the Test Bench
Test Flow Management
5. Test Bench Documentation Tutorial
6. Proposed Project Structure
Docs » 4. Test Bench Definition Tutorial
Edit on GitLab

4. Test Bench Definition Tutorial

Trigger Phase

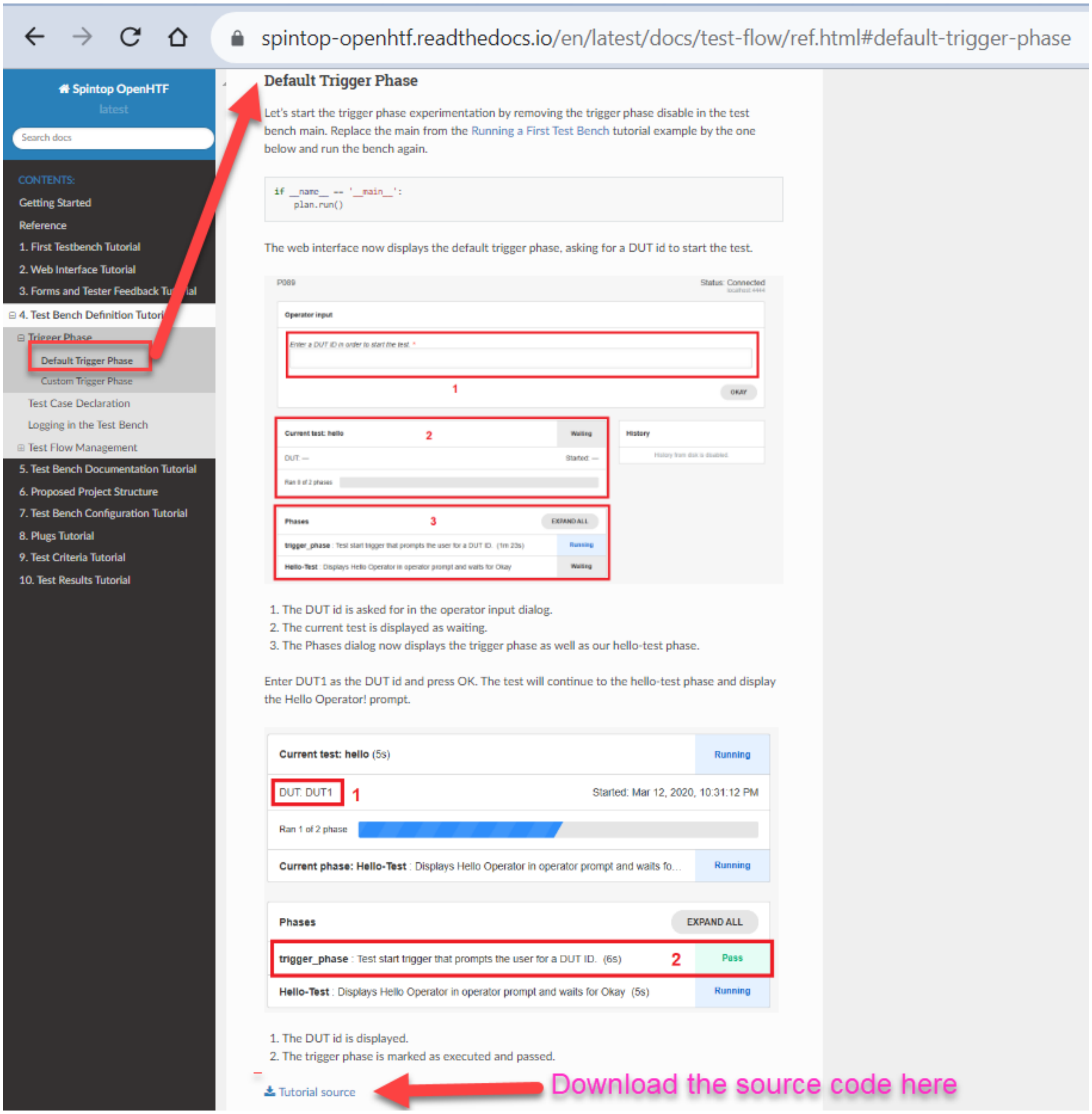
The trigger phase is the first phase executed in most test benches. It is a test phase which is outside of the test execution, meaning it cannot fail and does not consume test execution time. The trigger phase is added by default by spintop-openhtf. However it was disabled in our first example in the tutorial.

Default Trigger Phase

Let's start the trigger phase experimentation by removing the trigger phase disable in the test bench main. Replace the main from the Running a First Test Bench tutorial example by the one below and run the bench again.

```
if __name__ == '__main__':
    plan.run()
```

#++++++++++++++++++++++++++++
#Default Trigger Phase
https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#default-trigger-phase
#There is the source code at the end for you to download and try out to compare with your own code
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/e8722e5b3816e5ffc89dec9f6cefdcdd/main_w_trigger_phase.py


spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#default-trigger-phase
Spintop OpenHTF
latest
Search docs
CONTENTS:
Getting Started
Reference
1. First Testbench Tutorial
2. Web Interface Tutorial
3. Forms and Tester Feedback Tutorial
4. Test Bench Definition Tutorial
Trigger Phase
Default Trigger Phase
Custom Trigger Phase
Test Case Declaration
Logging in the Test Bench
Test Flow Management
5. Test Bench Documentation Tutorial
6. Proposed Project Structure
7. Test Bench Configuration Tutorial
8. Plugs Tutorial
9. Test Criteria Tutorial
10. Test Results Tutorial

Default Trigger Phase

Let's start the trigger phase experimentation by removing the trigger phase disable in the test bench main. Replace the main from the Running a First Test Bench tutorial example by the one below and run the bench again.

```
if __name__ == '__main__':
    plan.run()
```

The web interface now displays the default trigger phase, asking for a DUT id to start the test.

P089
Status: Connected
localhost:4444
Operator input
Enter a DUT ID in order to start the test. *
1
OKAY
Current test: hello
2
Waiting
DUT: —
Started: —
Ran 0 of 2 phases
History
History from disk is disabled.
Phases
3
EXPAND ALL
trigger_phase : Test start trigger that prompts the user for a DUT ID. (1m 23s)
Running
Hello-Test : Displays Hello Operator in operator prompt and waits for Okay
Waiting

1. The DUT id is asked for in the operator input dialog.
2. The current test is displayed as waiting.
3. The Phases dialog now displays the trigger phase as well as our hello-test phase.

Enter DUT1 as the DUT id and press OK. The test will continue to the hello-test phase and display the Hello Operator! prompt.

Current test: hello (5s)
Running
DUT: DUT1
1
Started: Mar 12, 2020, 10:31:12 PM
Ran 1 of 2 phase
Current phase: Hello-Test : Displays Hello Operator in operator prompt and waits fo...
Running
Phases
EXPAND ALL
trigger_phase : Test start trigger that prompts the user for a DUT ID. (6s)
2
Pass
Hello-Test : Displays Hello Operator in operator prompt and waits for Okay (5s)
Running

1. The DUT id is displayed.
2. The trigger phase is marked as executed and passed.

Tutorial source

Download the source code here

#++++++++++++++++++++++++++++++
#Custom Trigger Phase
https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#custom-trigger-phase
#There is the source code at the end for you to download and try out to compare with your own code
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/44dac268e008e9e1e671251ce6b60828/main_custom_trigger.py

Click on this icon to download the source code:

Trigger is using the custom form:

```
@plan.trigger('Configuration')
@plan.plug(prompts=UserInput)
def trigger(test, prompts):
    """Displays the configuration form"""
    response = prompts.prompt_form(FORM_LAYOUT)
    pprint (response)
```

#++++++++++++++++++++++++++++++
#Test Case Declaration
# https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#test-case-declaration &
#Logging in the Test Bench
#https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#logging-in-the-test-bench
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/4eea3aadc61462d7516505af71276ee8/main_logger.py

The test case is declared as:

```
@plan.testcase('Hello-Test')
@plan.plug(prompts=UserInput)
def hello_world(test, prompts):
    prompts.prompt('Hello Operator!')
    test.dut_id = 'hello' # Manually set the DUT Id to same value every test
```

#++++++++++++++++++++++++++++++
#Test Flow Management
pip install static
#Download this file first and save in the same directory:
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py
https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#test-flow-management

In the context of a spintop-openhtf test bench, test flow management consists in selecting the test cases to execute dynamically depending on

- the input of the operator during the trigger phase,
- the results and outcomes of the previous test phases.

#There is the source code at the end for you to download and try out to compare with your own code
https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b22f330ec404e341be9cf22c07191de9/main_result_flow.py
# Note: if test does not execute, download this file first and save in the same directory:
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py

#++++++++++++++++++++++++++++++
#Test Hierarchy

```
Test Plan
└── Sleep Sequence
    └── Sleep Test 1
    └── Sleep Test 2
```

#https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#test-hierarchy
#Defining Sequences or PhaseGroups
# https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#defining-sequences-or-phasegroups
#There is the source code at the end for you to download and try out to compare with your own code
#https://spintop-openhtf.readthedocs.io/en/latest/_downloads/15ed953e90823175106c3a4041a57723/main_sequences.py
# Note: if test does not execute, download this file first and save in the same directory:
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py

To define a *test sequence* within your *test plan*, simply use the TestSequence module.

```
from spintop_openhtf import TestPlan, TestSequence

sequence = TestSequence('Sleep Sequence')
```

To add *test cases* to the sequence, instead of to the *test plan* itself, simply use the sequence instead of the *test plan* in the *test case* decorator.

```
@sequence.testcase('Sleep Test 1')
def sleep_test_1(test):
    """Waits five seconds"""
    sleep(5)
```

```
@sequence.testcase('Sleep Test 2')
def sleep_test_2(test):
    """Waits ten seconds"""
    sleep(10)
```

This will create the following hierarchy

```
Test Plan
└── Sleep Sequence
    └── Sleep Test 1
    └── Sleep Test 2
```

#+++++++++++++++++++++++++++++

#Adding Levels to the Test Hierarchy

```
test plan
└── Sleep Sequence
    └── Sleep Sub Sequence 1
    |    └── Sleep Test 1A
    |    └── Sleep Test 1B
    └── Sleep Sub Sequence 2
        └── Sleep Test 2
```

#https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#adding-levels-to-the-test-hierarchy

#There is the source code at the end for you to download and try out to compare with your own code

#https://spintop-openhtf.readthedocs.io/en/latest/_downloads/46a07616fc8057ffbcaf475bdae34acd/main_subsequences.py

# Note: if test does not execute, download this file first and save in the same directory:

# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py

#+++++++++++++++++++++++++++++

#Managing the Test Flow Based on the Trigger Phase

# https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#managing-the-test-flow-based-on-the-trigger-phase

#There is the source code at the end for you to download and try out to compare with your own code

#https://spintop-openhtf.readthedocs.io/en/latest/_downloads/033cfbf2232186967f96f6ab6d04a150/main_dynamic_flow.py

# Note: if test does not execute, download this file first and save in the same directory:

# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py

As we have seen previously, the trigger phase is used to input dynamic configuration parameters at the beginning of the test bench. This test configuration can be used to manipulate the test flow. For example, a choice of test to execute in the form of a dropdown list or a scanned entry of the product version can lead to a different execution.

The test.state object is used to communicate the information through the test bench. Let's first define a new variable which will allow the execution of *Sleep Test 2* if the product entered in the trigger phase is "A"

```
def trigger(test, prompts):
    """Displays the configuration form"""
    response = prompts.prompt_form(FORM_LAYOUT)
    test.dut_id = response['dutid']
    test.state["operator"] = response['operator']
    test.state["product"] = response['product']
    if test.state["product"] == "A":
        test.state["run_sleep_2"] = True
    else:
        test.state["run_sleep_2"] = False
    pprint (response)
```

#+++++++++++++++++++++++++++++

#Using a Set Up and a Teardown or Cleanup Phase

It is a good practice to define a setup and a teardown phase within your sequences.

- The *setup phase* is used to initialize the test environment to execute the test cases in the sequence. Setup failure cancels the execution of the group, including the teardown. Define the setup phase using the *setup()* function.
- The *teardown phase* is usually used to reset the test environment to a known status and is always executed at the end of a sequence if at least one sequence's test phases is executed. It is executed even if one of the phase fails and the other intermediary phaes are not. Define the teardown phase using the *teardown()* function.

#https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#using-a-set-up-and-a-teardown-or-cleanup-phase

#Adding a setup and a teardown phase to a sub-sequence

# https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#adding-a-setup-and-a-teardown-phase-to-a-sub-sequence

#Final teardown

#https://spintop-openhtf.readthedocs.io/en/latest/docs/test-flow/ref.html#final-teardown

#There is the source code at the end for you to download and try out to compare with your own code

#https://spintop-openhtf.readthedocs.io/en/latest/_downloads/60e26f6b0fc601a0bc635a7998ad7100/main_setup_teardown.py
# Note: if test does not execute, download this file first and save in the same directory:
# https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1bcbeb9bb2b9a179689b399dccb03/static.py

```
sub_seq = sequence.sub_sequence('Sleep Sub Sequence 1')

@sub_seq.setup('Sub-sequence Setup')
def sub_setup(test):
    """Says Sub setup."""
    test.logger.info('Sub setup')

@sub_seq.testcase('Sleep Test 1A')
def sleep_test_1A(test):
    """Waits five seconds"""
    sleep(5)

@sub_seq.testcase('Sleep Test 1B')
def sleep_test_1B(test):
    """Waits five seconds"""
    sleep(5)

@sub_seq.teardown('Sub-sequence Cleanup')
def sub_cleanup(test):
    """Says Sub cleanup."""
    test.logger.info('Sub cleanup')

Test Plan
└─── Sleep Sequence
     └─── Sleep Sub Sequence 1
     │    └─── Sub-sequence Setup
     │    └─── Sleep Test 1A
     │    └─── Sleep Test 1B
     │    └─── Sub-sequence Cleanup
     └─── Sleep Sub Sequence 2
          └─── Sleep Test 2
```

==

#TUTORIAL 4: Test Bench Documentation Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/doc/ref.html#test-bench-documentation-tutorial

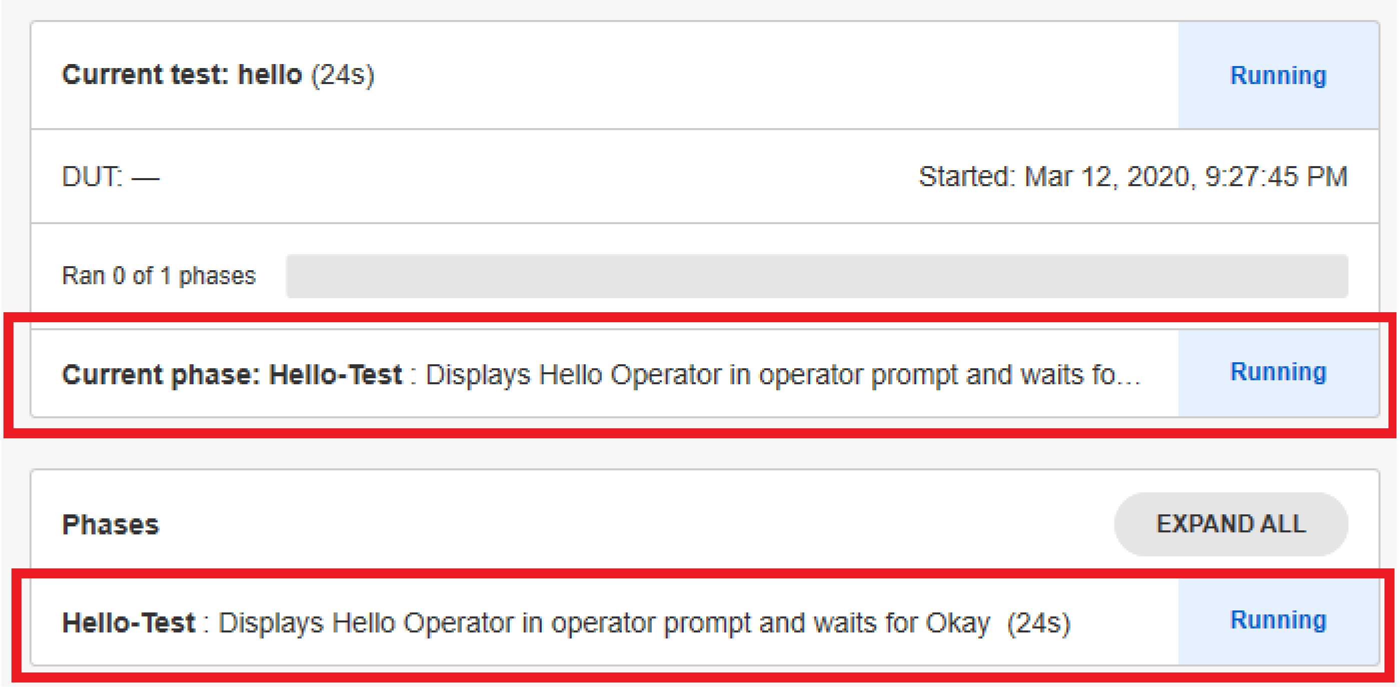
===

#The documentation process uses the python docstring feature. For more detail consult https://www.python.org/dev/peps/pep-0257/.

The following code snippet illustrates how to document a test case using the docstring feature.

```
@plan.testcase('Hello-Test')
@plan.plug(prompts=UserInput)
def hello_world(test, prompts):
    """Displays Hello Operator in operator prompt and waits for Okay"""
    prompts.prompt('Hello Operator!')
    test.dut_id = 'hello'
```

Add the docstring to the test bench implemented in the the Running a First Test Bench tutorial and run the test bench. As can be seen in the web interface, the docstring has been added to the test phase description in the current phase dialog and the executed phases dialog.



Current test: hello (24s) Running

DUT: — Started: Mar 12, 2020, 9:27:45 PM

Ran 0 of 1 phases

Current phase: Hello-Test : Displays Hello Operator in operator prompt and waits fo… Running

Phases EXPAND ALL

Hello-Test : Displays Hello Operator in operator prompt and waits for Okay (24s) Running

The docstring is also available throughout the test plan to be printed by the developper.

===

#TUTORIAL 5: Proposed Project Structure

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/project-structure/ref.html#proposed-project-structure

===

#Read the entire section and make sure you understand it.

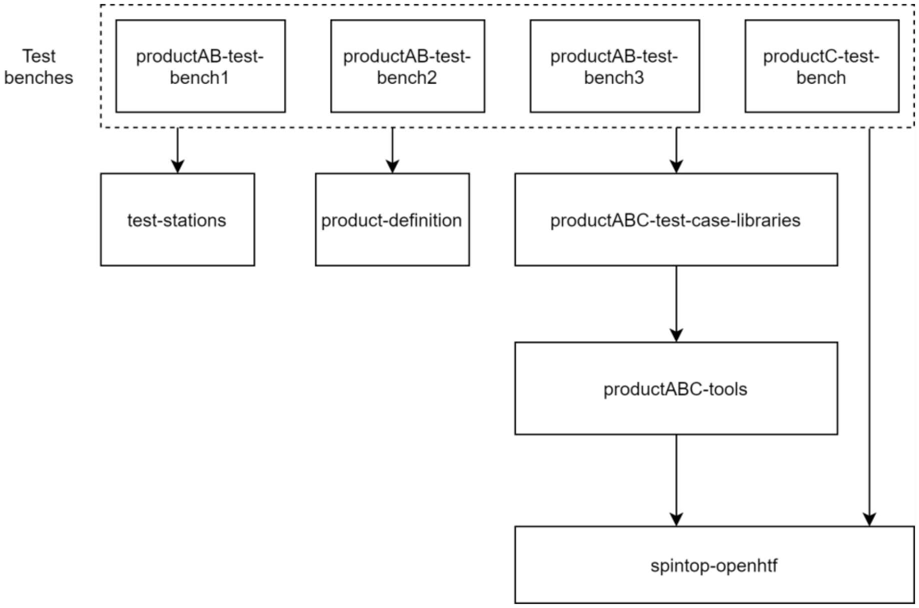
Proposed Single-Repository Structure

```
 repository
│   main.py: Calls and runs the test bench.
│
└───criteria: The criteria folder holds the global criteria for the test bench.
│   │         Sequence specific criteria can be defined at the sequence level.
│   │   global_criteria.py
│
└───products: Each python file defines a different product through its static parameters.
│   │   product_A.py
│   │   product_B.py
│   │   product_C.py
│
└───station_config: Each .yml file defines a different test station.
│   │   station_1.py
│   │   station_2.py
│
└───static_config: The static config folder holds the product-independent static configuration for
│   │              the test bench as a whole. Each sequence uses its own static configuration as well.
│   │   static_config.py
│
└───test_cases: The functions that implement the test cases are defined in test case libraries.
│   └───lib_A
│   │   │   cases_A1.py
│   │   │   cases_A2.py
│   │
│   └───lib_B
│   └───lib_C
│
└───test_sequences: The sequences are separated in folders which hold the sequence and test cases
│   │               declarations, the static configuration and the criterion specific to the sequence.
│   └───sequence_A
│   │   │   sequence_A.py
│   │   │   A_static_config.py
│   │   │   A_criteria.py
│   │
│   └───sequence_B
│   │   │   sequence_B.py
│   │   │   B_static_config.py
│   │   │   B_criteria.py
│
└───test_tools: The test tools that are used to implement the test cases are defined in tool libraries.
    └───lib_A
    │   │   tools_A1.py
    │   │   tools_A2.py
    │
    └───lib_B
    └───lib_C
```

===

Proposed Multiple-Repository Structure

The following example illustrates a set of test benches implementing the tests of a family of products. Product A and B are tested by the same set of 3 test benches which are run one after the other. Product C which is in the same product family as products A and B, is tested by a dedicated testb bench. It will however use the same test stations and test case libraries as the other products.



Test Bench Repository

At the top level of the repository architecture are the test bench repositories. The test case and tool libraries have been removed from the repository, as well as the test station configuration and product.

Each repository implements a specific test for a product or set of products.

```
repository
│   main.py: Calls and runs the test bench.
│
└───criteria: The criteria folder holds the global criteria for the test bench.
│   │         Sequence specific criteria can be defined at the sequence level.
│   │   global_criteria.py
│
└───static_config: The static config folder holds the product-independent static configuration
for
│   │              the test bench as a whole. Each sequence uses its own static configuration
as well.
│   │   static_config.py
│
└───test_sequences: The sequences are separated in folders which hold the sequence and test
cases
    │               declarations, the static configuration and the criterion specific to the
sequence.
    └───sequence_A
    │   │   sequence_A.py
    │   │   A_static_config.py
    │   │   A_criteria.py
    │
    └───sequence_B
        │   sequence_B.py
        │   B_static_config.py
        │   B_criteria.py
```

#TUTORIAL 6 : Test Bench Configuration Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/config/ref.html#test-bench-configuration-tutorial

#++++++++++++++++++++++++++++++

#Static configuration https://spintop-openhtf.readthedocs.io/en/latest/docs/config/ref.html#static-configuration

File static.py

```
class SleepConfig():
    SLEEP_TIME = 5
    SLEEP_ITERATIONS = 2
```

To access the configuration in the test bench code, simply import the class.

```
from static import SleepConfig
```

and access directly the instanciated parameters.

```
@plan.testcase('Sleep')
def sleep_test(test):
    """Waits five seconds"""

    for x in range(SleepConfig.SLEEP_ITERATIONS):
        print ("Sleep iteration {} - sleep time {}".format(x,
                                            SleepConfig.SLEEP_TIME))
        sleep(SleepConfig.SLEEP_TIME)
```

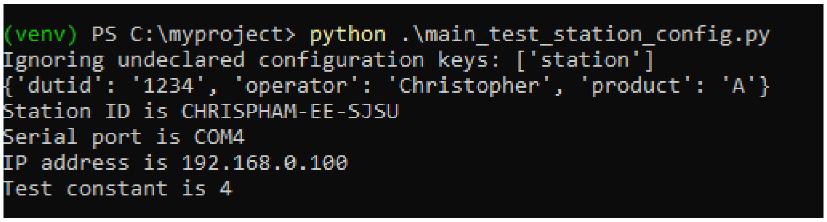
# Run code from https://spintop-openhtf.readthedocs.io/en/latest/_downloads/b1f1955ab9b01bd8f3f960c5db71969d/main_static_config.py

#++++++++++++++++++++++++++++++
#Test Station Configuration
#In the context of a test bench implementation on spintop-openhtf, the test station configuration
# is defined as a yaml file. As an example, the following yaml snippet defines the configuration of
# the serial port and ip address of a test bench.
# https://spintop-openhtf.readthedocs.io/en/latest/docs/config/ref.html#test-station-configuration

In the context of a test bench implementation on spintop-openhtf, the test station configuration is defined as a yaml file. As an example, the following yaml snippet defines the configuration of the serial port and ip address of a test bench.

```
serial  :
 comport: "COM4"
 baudrate : "115200"
ip_address : "192.168.0.100"
test_constant: 4
```

# Code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/6b83488b7f9e3c8e887cd9391cf4e697/main_test_station_config.py
# Code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/0956e639144a03e1f0ceaf33d8ac90c9/config.yml



```
(venv) PS C:\myproject> python .\main_test_station_config.py
Ignoring undeclared configuration keys: ['station']
{'dutid': '1234', 'operator': 'Christopher', 'product': 'A'}
Station ID is CHRISPHAM-EE-SJSU
Serial port is COM4
IP address is 192.168.0.100
Test constant is 4
```

#TUTORIAL 7: Plugs Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/plugs/ref.html#plugs-tutorial
==

A plug is a piece of code written to enable OpenHTF to interact with a particular type of hardware, whether that be a DUT itself or a piece of test equipment.

#Code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/82150495d0acc1414fae19fd4d6ea7cf/main_plug.py

#Save this code below as main_plug.py and run it to test whether you can
# copy file config.yml to the same directory as config_copied.yml
# You can learn more about shutil here: https://docs.python.org/3/library/shutil.html

```
# main_plug.py
import os

import openhtf as htf
from spintop_openhtf import TestPlan

from openhtf.plugs import BasePlug

import shutil as shutil

""" Plug Definition """

source = "config.yml"
destination = "C:\myproject\config_copied.yml"

class FileCopier(BasePlug):
    def copy_file(self, source_file, destination_folder):
        shutil.copy(source_file, destination_folder)


""" Test Plan """

# This defines the name of the testbench.
plan = TestPlan('File Copy Test Bench')

@plan.testcase('File Copy Test')
@plan.plug(copy_plug=FileCopier)
def file_copy_test(test, copy_plug):
    copy_plug.copy_file(source, destination)


if __name__ == '__main__':
    #plan.no_trigger()
    plan.run()
```

#TUTORIAL 8: Wrapping spintop-openhtf Plugs

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/plugs/ref.html#wrapping-spintop-openhtf-plugs

==

Read this section carefully and you will be able to work with the Linux products.

For example, a linux shell plug specific to a product can be created by wrapping the spintop-openhtf comport plug. The plug adds typical linux features to the comport plug such as login, file read, file copy, etc.

The plug is created

```
from spintop_openhtf.plugs.iointerface.comport import ComportInterface

class LinuxPlug(ComportInterface):

    def __init__(self, comport, baudrate=115200):
        super().__init__(comport, baudrate)

    def login(self, username):
        return self.com_target("{}".format(username), '{}@'.format(username), timeout=10,
keeplines=0)

    def file_read(self, file):
        return self.com_target("cat {}".format(file), '@', timeout=10, keeplines=0)

    def file_copy(self, source, destination):
        return self.com_target("cp {} {}".format(source, destination), '@', timeout=10,
keeplines=0)
```

and imported in a test case

```
linux_plug = LinuxPlug.as_plug( 'linux_plug', comport='COM5',  baudrate=115200)

@plan.testcase('Linux_Test')
@plan.plug(linux=linux_plug)
def LinuxTest(test, linux):
    try:
        linux.open_with_log()
        test.logger.info ("COM Port open")
        print(linux.file_read('file.txt'))
        linux.file_copy('file.txt',destination)
    except:
        test.logger.info ("COM Port open failed")
        return PhaseResult.STOP
```

#TUTORIAL 9: Test Criteria Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/criteria/ref.html#test-criteria-tutorial
==

#++++++++++++++++++++++++++++++
#Defining Test Criteria
# from https://spintop-openhtf.readthedocs.io/en/latest/docs/criteria/ref.html#defining-test-criteria

# code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/a6bb5d4265d808e5428db7f11a66af3f/main_criteria.py

### Defining Test Criteria

The criteria refer to the thresholds against which measures are compared to declare a test case PASS or FAIL. In the spintop-openhtf context, the measures module implements the criteria and the comparison against the selected values.

To define a test criterion, first define an openhtf measurement object.

```python
import openhtf as htf
criterion = htf.Measurement('test_criterion').in_range(18, 22)
```

Here the criterion defined will return a PASS if the value evaluated is between 18 and 22. The criterion name is “test_criterion” and it has been stored in the *criterion* variable.

Use the htf.measures decorator to use the defined criterion in a test case.

```python
@plan.testcase('Criteria test')
@htf.measures(criterion)
def criteria_test(test):
    value = 20
    test.measurements.test_criterion =  value
```

#++++++++++++++++++++++++++++++
# Execute the following code to learn about the criteria file:
# code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/11a87795b4588e0d6077febb5e285cbb/main_criteria_w_file.py
# criteria file: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/24b811903f3aeb968e8b41a88b49deea/criteria.py

#+++++++++++++++++++++++++++++++
#Dynamic Test Criteria
# https://spintop-openhtf.readthedocs.io/en/latest/docs/criteria/ref.html#dynamic-test-criteria
#Test criteria can be defined dynamically in the test logic. Two major reasons do so are:
# -To create criteria based on the product definition configuration
# -To create criteria based on previous results or measurements
# Execute the following code to learn about dynamic criteria:
# code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/064f55bafd6ae89dffb373f81c46bab4/main_criteria_dynamic.py
# product definition: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/5d30c0ca48b89bc419cf54167f9d3359/product.py

#+++++++++++++++++++++++++++++++
# Advanced reading (optional): OpenHTF Validators
# https://spintop-openhtf.readthedocs.io/en/latest/docs/criteria/ref.html#module-openhtf.util.validators

## Dynamic Test Criteria

Defining a Dynamic Test Criterion

To define a dynamic criterion, use the htf.Measurement function as in the static definitions, but instead as defining it as a function decorator, create a new entry in the *state.running_phase_state.measurements* dictionary. Also, the test case must have access to the *state* object. To do so,

- Add the *requires_state=True* attribute to the testcase defintion decorater
- Instead of passing the *test* variable to the test case function, the *state* variable must be passed.
- For the evaluation of the test criterion, the measurements dictionary must be accessed from the *state* object instead of the *test* objects (state.test_api.measurements)

```
@plan.testcase('Dynamic Criterion Test', requires_state=True)
def criteria_test(state):
    value = 12
    state.running_phase_state.measurements['VOLTAGE'] = htf.Measurement('VOLTAGE').
                        in_range(11.5,12.5)
    state.test_api.measurements.VOLTAGE = value
```

Test criteria can be defined dynamically in the test logic. Two major reasons do so are:

- To create criteria based on the product definition configuration
- To create criteria based on previous results or measurements

Based on a product definition

A good example of a criterion based on a product definition is a criterion defined around the input voltage of a device. For example, the product defintion states that the input voltage is 12V. A criterion defined around the voltage would state that a measure of 12V +/- 0.5 Volts would indicate a PASS.

First create a file name product.py, and implement the voltage definition within it.

```
class voltage():
    input_voltage = 12
```

Then define the dynamic criterion importing the value from the product definition.

```
from product import voltage

@plan.testcase('Dynamic Criterion Test from Product Definition', requires_state=True)
def product_definition_test(state):

    #value = measure_input_voltage()
    value = 12

    #definition of criterion
    state.running_phase_state.measurements['VOLTAGE'] = htf.Measurement('VOLTAGE').
                    in_range(voltage.input_voltage - 0.5, voltage.input_voltage + 0.5)


    #evaluation of criterion
    state.test_api.measurements.VOLTAGE = value
```

Based on a previous measurement

The same method can be used to define a criterion from a previous measurement. For example, testing a voltage divider, which has been set to produce a voltage 30% of the input voltage. The input voltage has been measured as 12.05 volts. To create the divided voltage criterion, the measured value must be used in its definition, not the nominal value.

```
@plan.testcase('Dynamic Criterion Test from Previous Measurement', requires_state=True)
def previous_measurement_test(state):

    #measured_input = measure_input_voltage()
    measured_input = 12.05

    #divider_voltage = measure_divider_voltage()
    divider_voltage = 3.55

    #definition of criterion as within +/- 5% of the divider voltage, which is 30% on the
measured input
    state.running_phase_state.measurements['DIVIDER_VOLTAGE'] =
htf.Measurement('DIVIDER_VOLTAGE').
                    in_range(measured_input * 0.3 * 0.95, measured_input * 0.3 * 1.05)

    #evaluation of criterion
```

# code: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/064f55bafd6ae89dffb373f81c46bab4/main_criteria_dynamic.py

#Run this code

(venv) PS C:\myproject> python .\main_criteria.py

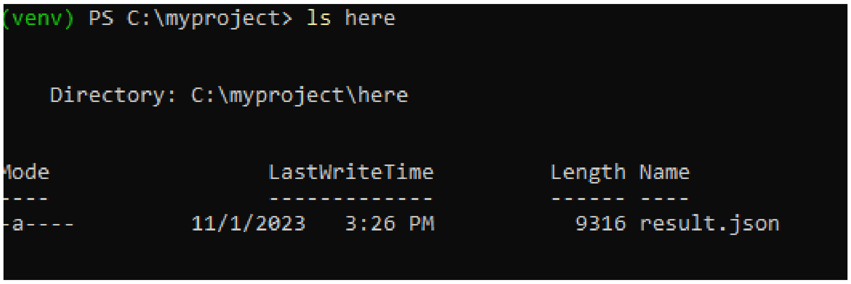
# product definition: https://spintop-openhtf.readthedocs.io/en/latest/_downloads/5d30c0ca48b89bc419cf54167f9d3359/product.py

```
    state.test_api.measurements.DIVIDER_VOLTAGE = divider_voltage
```

#TUTORIAL 10: Test Results Tutorial

# from https://spintop-openhtf.readthedocs.io/en/latest/docs/results/ref.html#test-results-tutorial
===
# Read this section carefully.

The result.json file is inside this folder:
C:\myproject\here



```
(venv) PS C:\myproject> ls here


    Directory: C:\myproject\here


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         11/1/2023   3:26 PM           9316 result.json
```

# My test results are saved in C:\Users\chris.pham\AppData\Local\tackv\spintop-openhtf\openhtf-history

PART 3: Design Your Own Tests

Now it is good time to modify the tutorials to add your own tests.

You will need:

1. Customized test menu
2. Your own tests to respond to each item in your customized test menu