Python Scripting API

- Gregory Jackson, RTDS Technologies

ATC
RTDS APPLICATIONS & TECHNOLOGY CONFERENCE

APPLICATIONS & TECHNOLOGY CONFERENCE 2025
CHICAGO, ILLINOIS, U.S.A.

RTDS Technologies
AMETEK

Outline

- Introduction
- Getting Started…
- Demonstration and Examples
- Using External Python Editors
- Learning Resources
- Summary

Introduction

python™

Scripting Overview

- Scripting allows users to automate the running of simulation cases and the collection of simulation results.
- Operation of RSCAD is controlled programmatically
- Users can do things like ....
  - Open, Save and Run Cases
  - Operate buttons, slider, dials etc.
  - Save plots
  - Change component parameters
  - Change case parameters

Application

- Scripting is a critically important tool when lots of simulations with small variations need to be run
- Prior to the introduction of our Python API we had a C-based scripting facility which has been labelled *legacy scripting.*
- Legacy scripting served our customer well but there was an increasing demand for us to support python scripting due to the language’s popularity and ease of use. We officially released the API in RSCAD FX version 2.4.

Python Scripting VS Legacy Scripting

- Control of more properties including case settings like timestep and rack
- Direct modification of component parameters without using ‘Draft’ variables
- Automated the placement of Draft components
- Syntax generally has an object-oriented structure
- RSCAD’s Python API can installed and run in **<u>any</u>** Python Interpreter with version >3.9. Therefore the launch and operation of RSCAD FX can be controlled from outside the application.
- Expandable functionality by incorporating 3rd Party Packages

python™

Support for 3rd Pary Packages

- Allows users to leverage the capability of these packages by incorporating them into their simulations.

pandas NumPy matplotlib XGBoost PyTorch scikit-image image processing in python

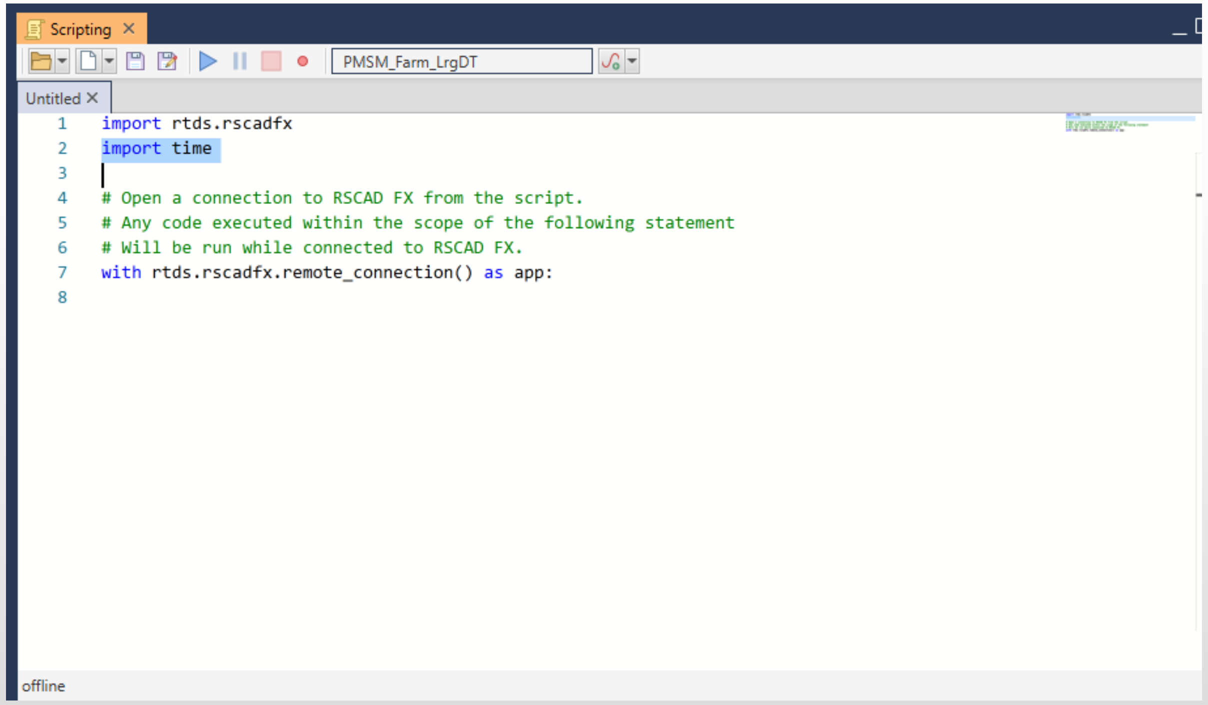
seaborn statsmodels SciPy biopython SimpleCV PyTorch pytest

mpmath TextBlob pyscreenshot OpenCV bokeh GENSIM topic modelling for humans

Getting Started...

Scripting Utility Tab

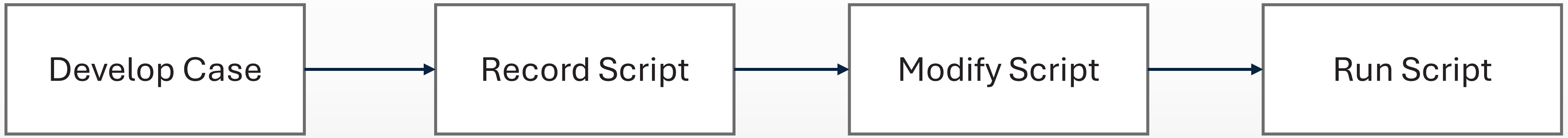
- Text editor that allows users to create, edit and run scripts within the RSCAD Environment
- Supports
  - Open Scripts
  - Save Scripts
  - Run Scripts
  - Pause Scripts
  - Stop Scripts
  - Record Scripts



```
import rtds.rscadfx
import time

# Open a connection to RSCAD FX from the script.
# Any code executed within the scope of the following statement
# Will be run while connected to RSCAD FX.
with rtds.rscadfx.remote_connection() as app:

```

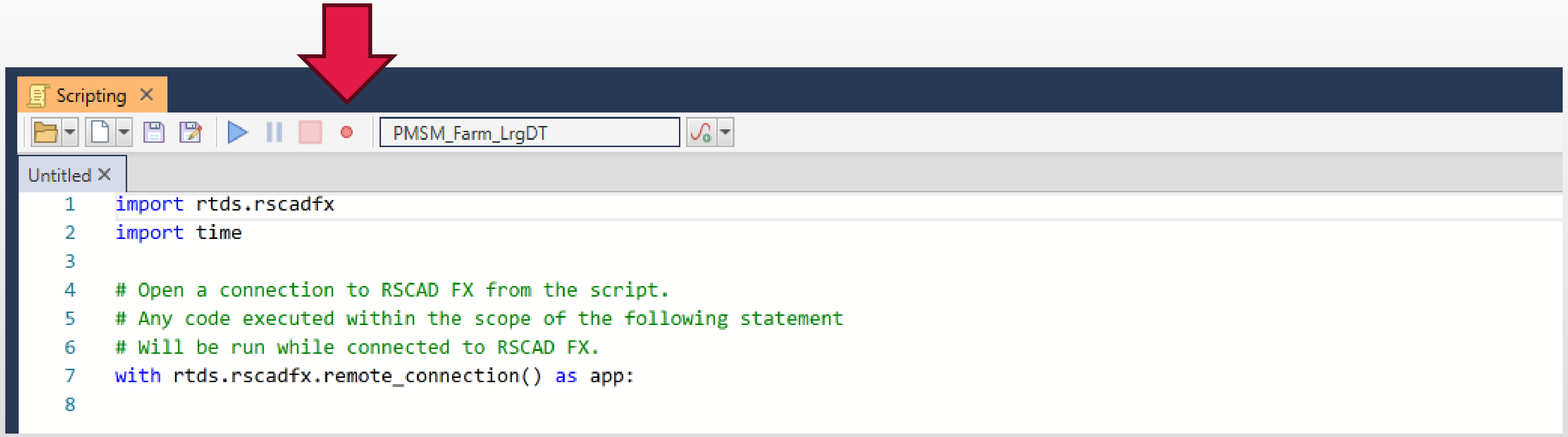
Typical Workflow

- Using Python scripting in RSCAD will generally follow this flow:



Recording a Script

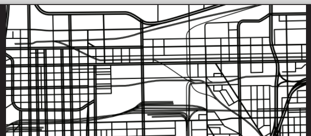
- Instead of writing code manually, users can simply record their actions and have them transcribed into code which can then be modified or run later
- Speeds up process of developing scripts



```python
import rtds.rscadfx
import time

# Open a connection to RSCAD FX from the script.
# Any code executed within the scope of the following statement
# Will be run while connected to RSCAD FX.
with rtds.rscadfx.remote_connection() as app:

```

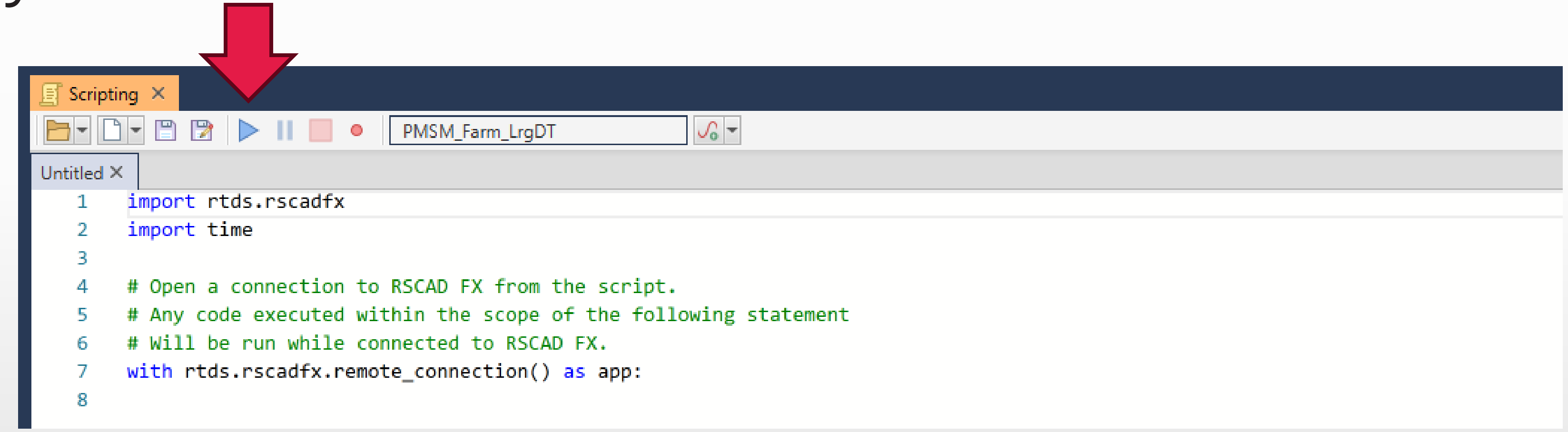
Modifying a Script

- Once a script is recorded users can add several coding structures such as *for loops*, *while* loops, and *if conditions* to customize them.
- Recorded commands can be duplicated or modified.
- New scripting commands from our Python API can be inserted into existing scripts. Users should reference the help documentation for details about syntax
- Other Python packages can be used along the RSCAD Python API



Running a Script

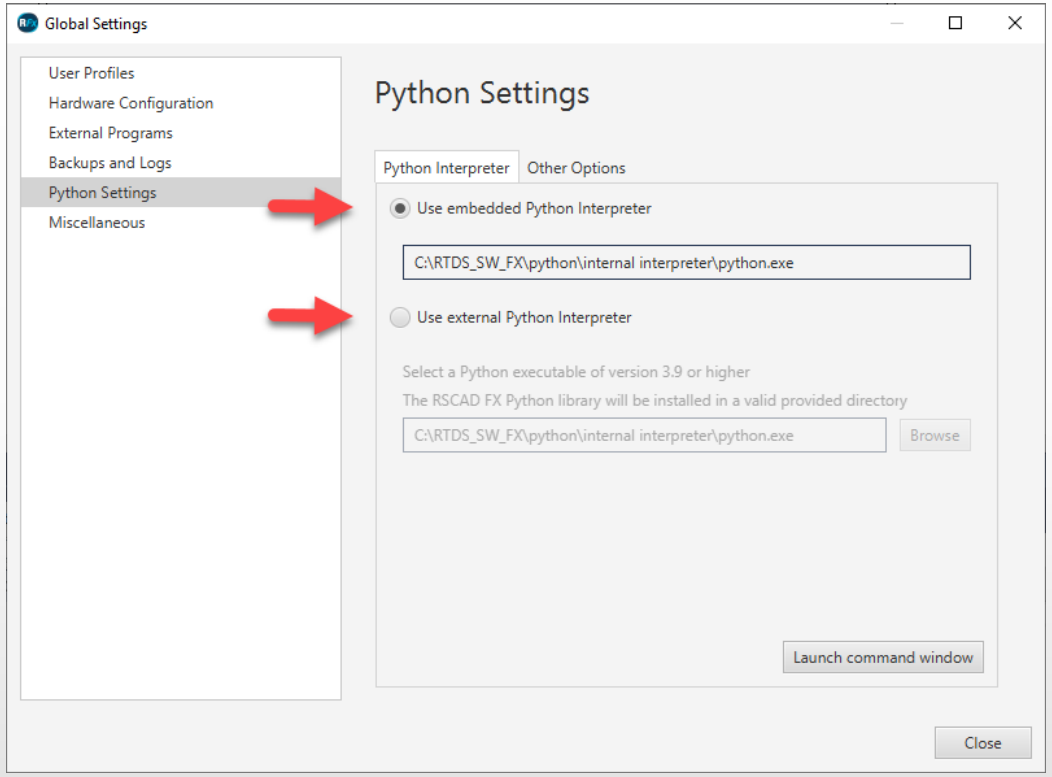
- After a script has been created or modified it is a simple to replay it



```
import rtds.rscadfx
import time

# Open a connection to RSCAD FX from the script.
# Any code executed within the scope of the following statement
# Will be run while connected to RSCAD FX.
with rtds.rscadfx.remote_connection() as app:

```

Python Scripting Settings

- Found under Global Settings
- Use embedded interpreter or use a python interpreter that exists outside of RSCAD
- Reference objects by their names or by their Unique ID #s
- Optionally track the delays between recorded actions.



Python Settings

(● Use embedded Python Interpreter)

C:\RTDS_SW_FX\python\internal interpreter\python.exe

○ Use external Python Interpreter

Select a Python executable of version 3.9 or higher

The RSCAD FX Python library will be installed in a valid provided directory

C:\RTDS_SW_FX\python\internal interpreter\python.exe

Demonstration and Examples

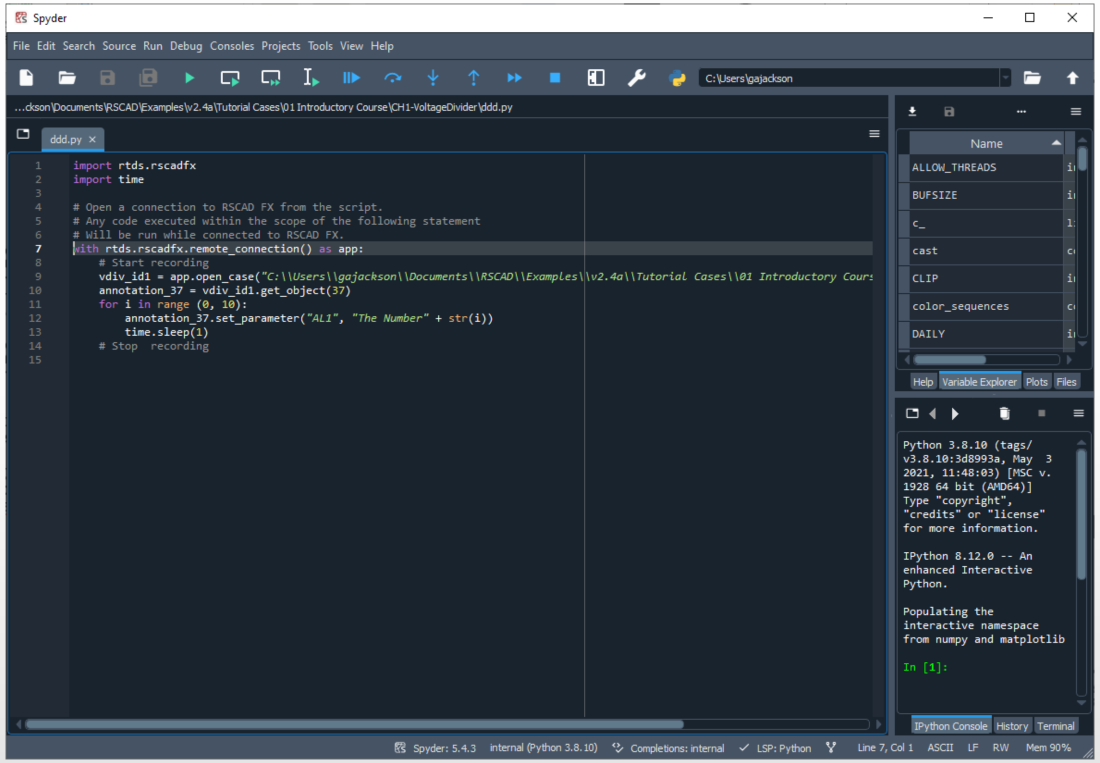
Using External Python Editors

Using Python API outside of RSCAD

- RSCAD's Python API can be used outside the RSCAD application using a text editor or IDE such as *Visual Studio Code* or *Spyder*
- RSCAD's Python API must be installed in any external Python interpreter that is used
    - Python Interpreter version >3.9
- Using an IDE like *Visual Studio Code* offers several advantages including debugging tools to step through running scripts as well as code completion tools like *intellisense*

Spyder

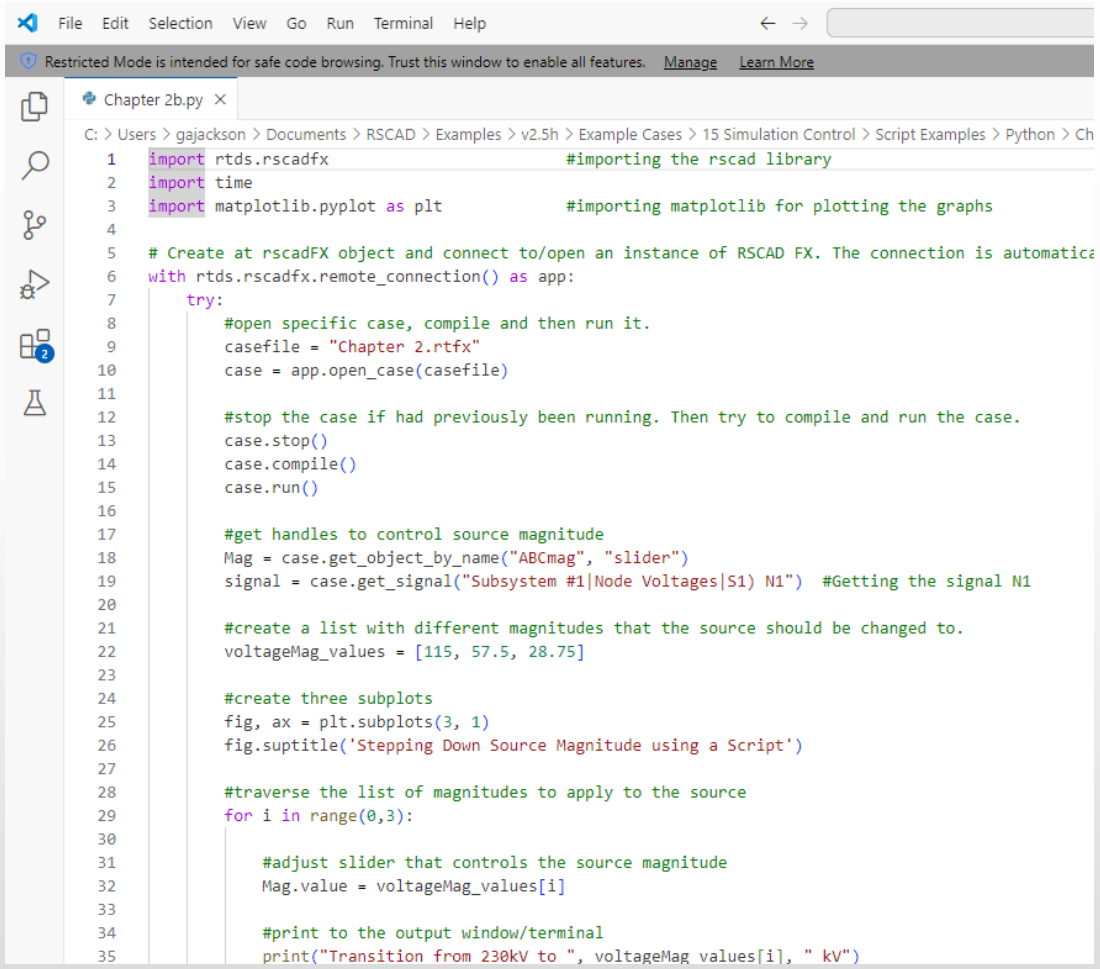
Visual Studio Code



Spyder
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\gajackson
...ckson\Documents\RSCAD\Examples\v2.4a\Tutorial Cases\01 Introductory Course\CH1-VoltageDivider\ddd.py
ddd.py
Name
ALLOW_THREADS
BUFSIZE
c_
cast
CLIP
color_sequences
DAILY
Help Variable Explorer Plots Files
IPython Console History Terminal
Spyder: 5.4.3 internal (Python 3.8.10) Completions: internal LSP: Python Line 7, Col 1 ASCII LF RW Mem 90%

```python
import rtds.rscadfx
import time

# Open a connection to RSCAD FX from the script.
# Any code executed within the scope of the following statement
# Will be run while connected to RSCAD FX.
with rtds.rscadfx.remote_connection() as app:
    # Start recording
    vdiv_id1 = app.open_case("C:\\Users\\gajackson\\Documents\\RSCAD\\Examples\\v2.4a\\Tutorial Cases\\01 Introductory Cours
    annotation_37 = vdiv_id1.get_object(37)
    for i in range (0, 10):
        annotation_37.set_parameter("AL1", "The Number" + str(i))
        time.sleep(1)
    # Stop  recording

```

```
Python 3.8.10 (tags/
v3.8.10:3d8993a, May  3
2021, 11:48:03) [MSC v.
1928 64 bit (AMD64)]
Type "copyright",
"credits" or "license"
for more information.

IPython 8.12.0 -- An
enhanced Interactive
Python.

Populating the
interactive namespace
from numpy and matplotlib

In [1]:
```



File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Chapter 2b.py
C: > Users > gajackson > Documents > RSCAD > Examples > v2.5h > Example Cases > 15 Simulation Control > Script Examples > Python > Ch

```python
import rtds.rscadfx                     #importing the rscad library
import time
import matplotlib.pyplot as plt         #importing matplotlib for plotting the graphs

# Create at rscadFX object and connect to/open an instance of RSCAD FX. The connection is automatica
with rtds.rscadfx.remote_connection() as app:
    try:
        #open specific case, compile and then run it.
        casefile = "Chapter 2.rtfx"
        case = app.open_case(casefile)

        #stop the case if had previously been running. Then try to compile and run the case.
        case.stop()
        case.compile()
        case.run()

        #get handles to control source magnitude
        Mag = case.get_object_by_name("ABCmag", "slider")
        signal = case.get_signal("Subsystem #1|Node Voltages|S1) N1")  #Getting the signal N1

        #create a list with different magnitudes that the source should be changed to.
        voltageMag_values = [115, 57.5, 28.75]

        #create three subplots
        fig, ax = plt.subplots(3, 1)
        fig.suptitle('Stepping Down Source Magnitude using a Script')

        #traverse the list of magnitudes to apply to the source
        for i in range(0,3):

            #adjust slider that controls the source magnitude
            Mag.value = voltageMag_values[i]

            #print to the output window/terminal
            print("Transition from 230kV to ", voltageMag_values[i], " kV")
```

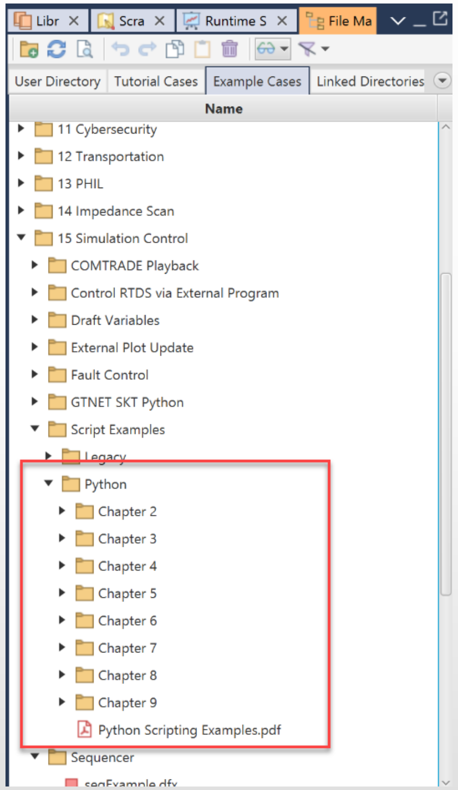
Learning Resources

Webinar

Your world in real time.
Webinar & Demo: New Python Scripting Capabilities for the RTDS® Simulator

- https://knowledge.rtds.com/hc/en-us/articles/30378458295703-Webinar-Demo-New-Python-Scripting-Capabilities-for-the-RTDS-Simulator

Example Cases

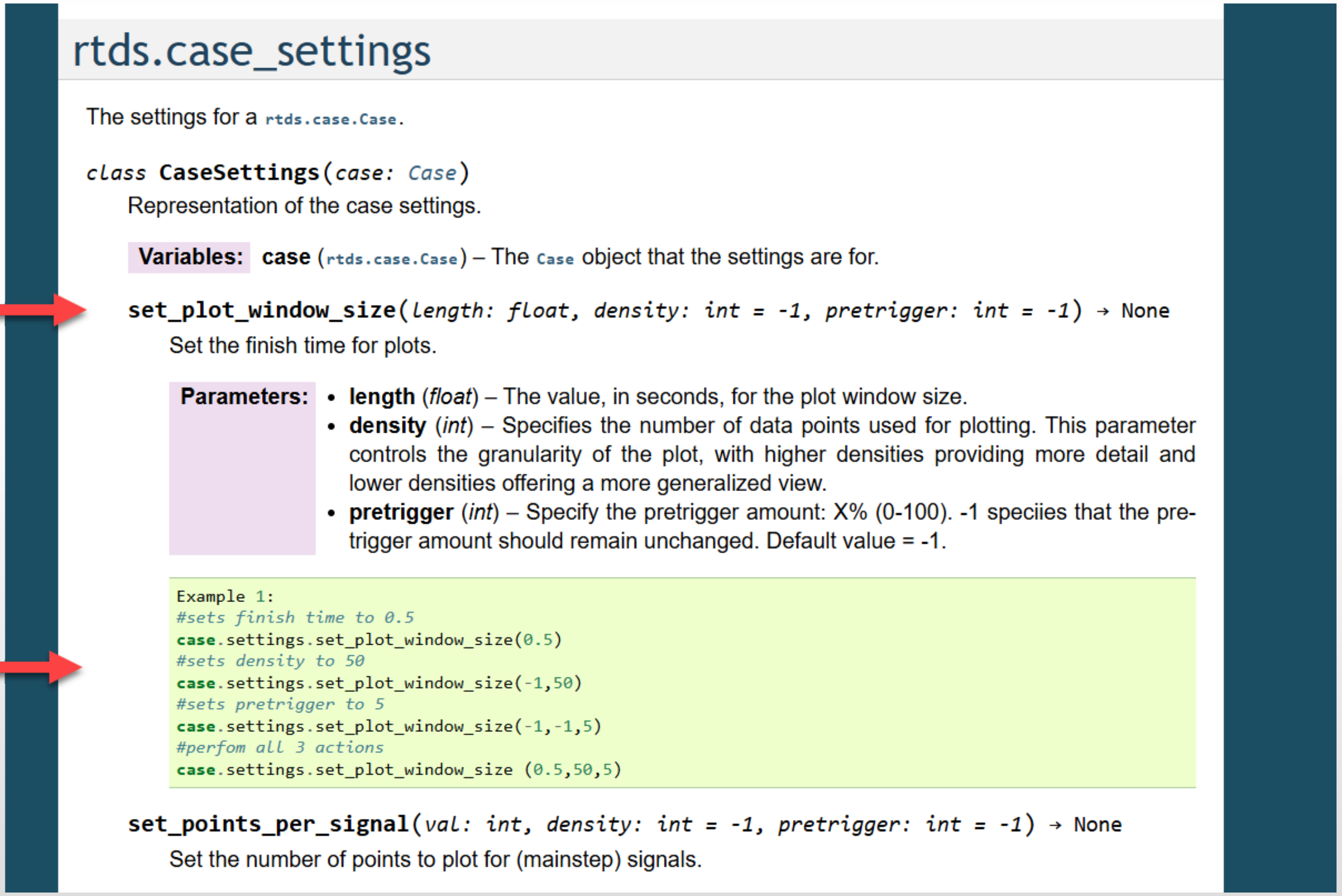


- 11 Cybersecurity
- 12 Transportation
- 13 PHIL
- 14 Impedance Scan
- 15 Simulation Control
  - COMTRADE Playback
  - Control RTDS via External Program
  - Draft Variables
  - External Plot Update
  - Fault Control
  - GTNET SKT Python
  - Script Examples
    - Legacy
    - Python
      - Chapter 2
      - Chapter 3
      - Chapter 4
      - Chapter 5
      - Chapter 6
      - Chapter 7
      - Chapter 8
      - Chapter 9
      - Python Scripting Examples.pdf
  - Sequencer
    - seqExample.dfx

- Several examples with documentation show users how to progressively build increasingly complex Python scripts to automate RSCAD. Including.....
  - Saving simulation results
  - Direct modification of component parameters
  - Using 3rd party Python packaged write MSWord Reports, plot simulation results, and do spectral analysis of simulation results

Documentation

- **There is no getting around this**….. users will have to consult the documentation to understand the syntax of RSCAD's python API.



rtds.case_settings

The settings for a `rtds.case.Case`.

class **CaseSettings**(*case: Case*)

Representation of the case settings.

Variables: **case** (`rtds.case.Case`) – The `Case` object that the settings are for.

set_plot_window_size(*length: float, density: int = -1, pretrigger: int = -1*) → None

Set the finish time for plots.

Parameters:
- **length** (*float*) – The value, in seconds, for the plot window size.
- **density** (*int*) – Specifies the number of data points used for plotting. This parameter controls the granularity of the plot, with higher densities providing more detail and lower densities offering a more generalized view.
- **pretrigger** (*int*) – Specify the pretrigger amount: X% (0-100). -1 speciies that the pre-trigger amount should remain unchanged. Default value = -1.

```
Example 1:
#sets finish time to 0.5
case.settings.set_plot_window_size(0.5)
#sets density to 50
case.settings.set_plot_window_size(-1,50)
#sets pretrigger to 5
case.settings.set_plot_window_size(-1,-1,5)
#perfom all 3 actions
case.settings.set_plot_window_size (0.5,50,5)
```

set_points_per_signal(*val: int, density: int = -1, pretrigger: int = -1*) → None

Set the number of points to plot for (mainstep) signals.

Summary

Summary

- New Python API allows for automated control of RSCAD FX
- API can be used with other 3rd Party Python Packages
- More cases parameters can be modified than with legacy scripting
- New functionality allows users to build cases programmatically
- Direct modification of component parameters is now possible without having to define *Draft Variables*
- It is possible to run scripts within RSCAD FX or other Python Development Environments

Thank you!