# REIXS Soft X-ray Spectroscopy Workshop

## On-the-fly data processing and visualization: An introduction to the python *reixs* package

# Outline

# What is the *reixs* python package



- Tool for data processing and visualization - based on the needs of the user community at the REIXS beamline

→ Disclaimer: dominant focus on the RIXS endstation

- Ships as python package and is distributed on *PyPi*
- Runs as stand-alone python code but works best when integrated in a Jupyter notebook environment for interactive computing

# Why develop the *reixs* python package

- Native integration for beamline data with commercial plotting software lacking
- Pre-Processing required before data can be visualized - multi-step process (e.g. strip the scan of interest out of data file, ...)
- Data reduction complex: combine information such as independent axis, detector scale, and matrix from multiple files (i.e. header file, auxiliary files)
- Mutual energy calibration of the monochromator and spectrometer difficult for new users

$\Rightarrow$ Users collect data but do not utilize it to the fullest extend

$\Rightarrow$ Focus on the data interpretation rather than the processing

# Why use the *reixs* python package



- On-the-fly data processing is important for decision making while measuring - especially when controlling the beamline remotely and scripting scans
- Analysis package provides real-time interaction with acquired spectra
- Directly compatible with the beamline data format
- Allows to quickly reduce higher-dimensional data to 1D
- Supports the export to commercial plotting software
- It is easy to use - no programming skills required and you will receive hands-on training today!

# Objectives for today



After this session, users will be able to

- run the *reixs* data analysis package;
- extract and export scans of interest;
- use reduction methods to exploit multi-channel analyzer data;
- apply mathematical operations to the collected data;
- plot and interact with visualized data.

# Nomenclature

## Variable

A *variable* is a name for a value. Variables are created on demand whenever a value is assigned to them and stored in memory.
e.g. `variable = "Hello"`

## Function

A block of code that performs a specific task.

## Class

A "blueprint" (constructor) for creating objects.

# Types of Data



- Strings
  - collection of characters
  - may include letters and numbers
  - string literals are written by enclosing them in single ('Hello') or double quotes ("Hello")
- Integer numbers
- Objects
  - A collection of data/properties (variables) and methods (functions)
  - Methods are functions that belong to the object
  - An object is a specific instance of a class

# Function calls



```
# Setup a variable "my_variable"
# Store values of function "my_function"
# Pass on arguments "arg1" and "arg2"
# Pass on key-word arguments "kwarg1" and "kwarg2"

my_variable =␣
 ↪my_function(arg1,arg2,kwarg1=True,kwarg2=False)

# Note the parentheses after "my_function"
# Need to specify arguments;
# Key-word arguments not required (default to pre-set␣
 ↪values)
```

# Calling a class method



```
# Similarly to functions: execute class method␣
 ↪associated with object

# Create object "my_object" of the "SomeObject" class
# Existing method "my_method"

my_object = SomeObject()
my_object.my_method(arg1,arg2,kwarg1=True,kwarg2=False)
```

# Quickstart: starting the jupyter environment



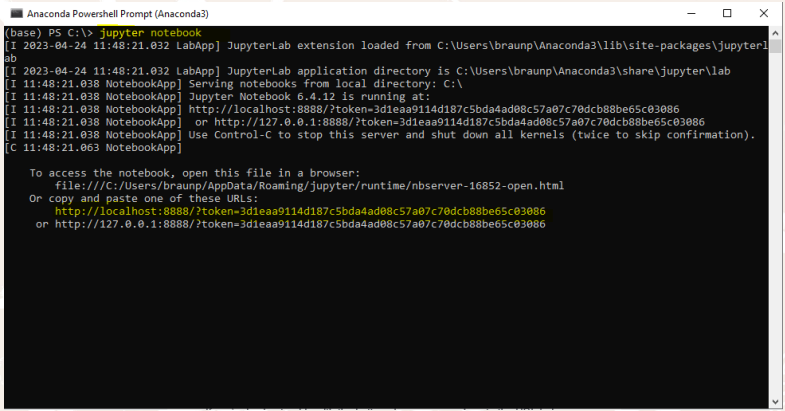



Figure: Starting Jupyter Notebook

# Quickstart: jupyter browser



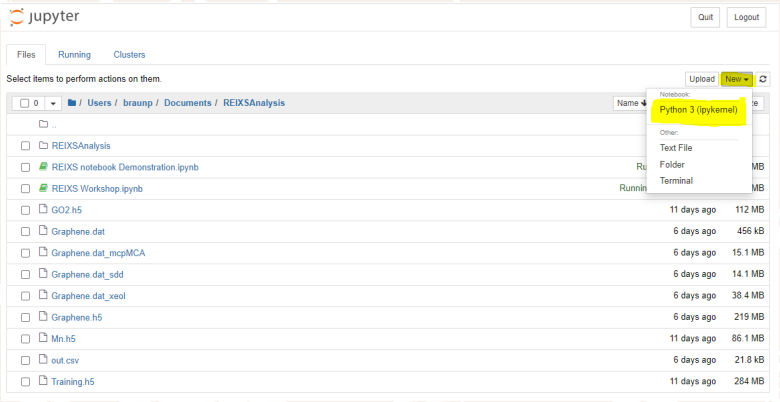



Figure: Integrated file browser allows to choose directory

# Quickstart: jupyter notebook



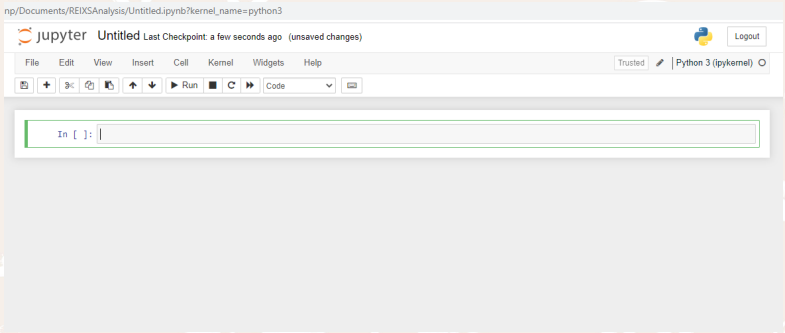



Figure: Empty Jupyter Notebook with python3 kernel

# Quickstart: using the example import



```
# Import module to load REIXS scans from package
from reixs.LoadData import *

# Import bokeh plotting module
from bokeh.io import output_notebook

# Enable bokeh plotting within the notebook environment
output_notebook(hide_banner=True)

# Select the base directory
# Can use absolute or relative path
basedir = '.' # We choose the current folder
```

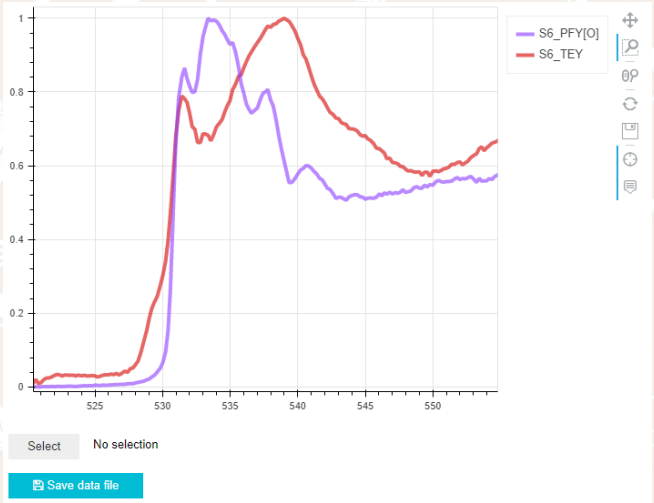
# Quickstart: pattern to load data



```
# Scan analysis always follows the same pattern
# (1) Create an object to load scans
# (2) Load scans as desired
# (3) Plot all loaded scans
# (4) Export the scans to ASCII

scans = Load1d()
scans.load(basedir,'Training.h5','Mono Energy','TEY',6)
scans.load(basedir,'Training.h5','Mono␣
 ↪Energy','PFY[0]',6)
scans.plot()
scans.exporter()
```

# Quickstart: output from the previous code





# Docstrings: Interactive Helptools are available by pressing Shift + Tab



```
In [7]: # Scan analysis always follows the same pattern
        # (1) Create an object to load scans
        # (2) Load scans as desired
        # (3) Plot all loaded scans
        # (4) Export the scans to ASCII

        scans = Loadid()
        scans.load(basedir,'Training.h5','Mono Energy','TEY',6)
        scans.plot()
```

```
Signature: scans.load(basedir, file, x_stream, y_stream, *args, **kwargs)
Docstring:
Load one or multiple specific scan(s) for selected streams.

Parameters
----------
basedir : string
    Specifiy the absolute or relative path to experimental data.
file : string
    Specify the file name (either ASCII or HDF5).
x_stream : string
    Specifiy the data for the horizontal axis.
    Use: "Mono Energy", "MCP Energy", "SDD Energy", "XEOL Energy", "Points", or any SCA scalar array.
y_stream : string
    Specifiy the data for the vertical axis.
    Use: "TEY", "TFY, "PFY", "iPFY", "XES", "rXES", "specPFY",
         "XRF", "rXRF", "XEOL", "rXEOL", "POY", "TOY", "EY", "Sample", "Mesh", "ET", or any SCA scalar array.
*args : int
    Separate scan numbers with comma.
**kwargs: multiple, optional
    Options:
        norm : boolean
            Norm the spectra to [0,1].
            default: True
        xoffset : list of tuples
            Offset the x-axis by applying a polynomial fit.
```

# Errors and Tracebacks: Intimidating but provide useful information



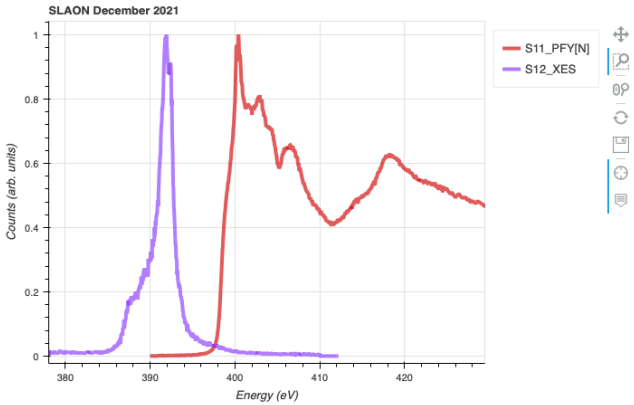
```
scans = Load1d()
scans.load(basedir,'Training.h5','Mono Energy','TEEY',6)
```

```
KeyError: 'TEEY'

UserWarning: Stream not defined. Only mnemonics␣
 ↪supported! Special Stream not defined.

      7 scans = Load1d()
----> 8 scans.load(basedir,'Training.h5','Mono␣
 ↪Energy','TEEY',6)
```

# Plot x-ray emission and absorption spectra on common energy scale

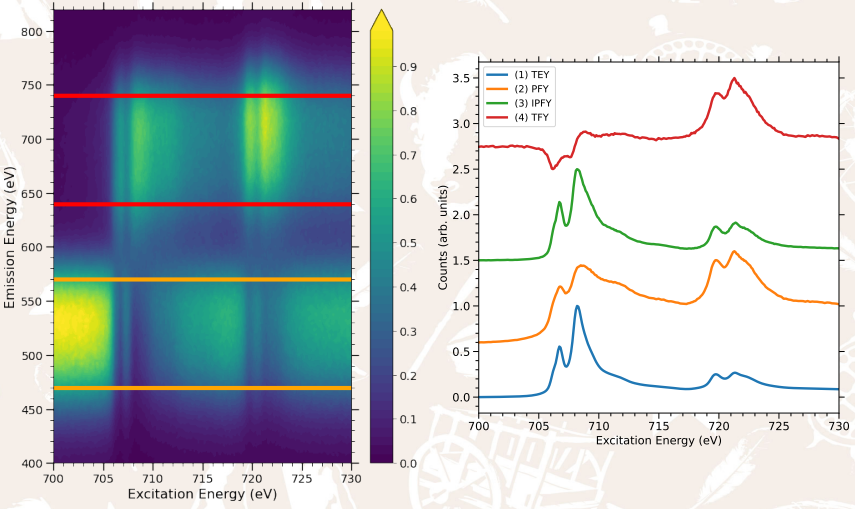


```
xas = Load1d()
xas.load(basedir,'Plate5_Ruhul.dat','Mono Energy','PFY[N]',11,norm=True)
xas.load(basedir,'Plate5_Ruhul.dat','MCP Energy','XES',12,norm=True,xoffset=[(377.9,380),(382.8,385),
xas.plot(xlabel='Energy (eV)',title='SLAON December 2021',ylabel='Counts (arb. units)')
```



# Silicon Drift Detector Data at your fingertips



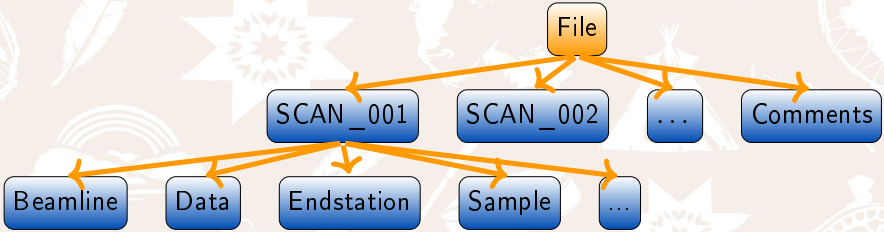


# Meta Data

- Descriptive information about a resource but does not include the actual experimental data
- Captures a snapshot of the beamline
- Useful to monitor beamline components and check beamline setup, i.e. time evolution of a specific PV
- Allows to log pertinent information for each scan and generate an automated experimental log, i.e. spreadsheet
- All information encapsulated in the HDF5 file

# HDF5 Data Structure





# Plot meta data as a function of scan number



```
beam = LoadBeamline()
beam.load(basedir,'Training.h5','Endstation/Motors/spa')
beam.legend_loc = 'top_left'
beam.plot(xlabel='Scan Number', ylabel='Degrees')
```

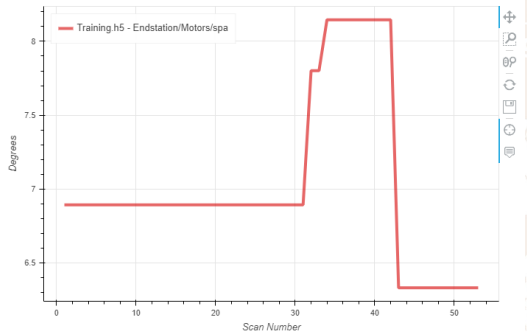



Figure: Variation of the spectrometer angle for the different scans in the data file.

# Under Development: Generate measurement log from HDF5 entries



| | Command | Sample Stage (ssh) | SS (ssv) | SS (ssd) | Spectrome (XES dist) | Spectromt (XES angl) | Flux 4-Jaw (mm) | Mono G | Mono M | Polarizati | Status |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | ascan engy 559.998 559.998 19 3( | 0.45 | -3.9 | 0 | 844.1715 | 6.893 | 8 | Ni LEG | NICKEL | LINEAR H | Scan successfully com |
| **4** | ascan engy 559.991 559.991 19 3( | 0.45 | -3.9 | 0 | 844.1715 | 6.893 | 7.999 | Ni LEG | NICKEL | LINEAR H | Scan successfully com |
| **5** | rscan engy 530 560 150 1 | 0.45 | -3.9 | 0 | 844.1715 | 6.893 | 8 | Ni LEG | NICKEL | LINEAR H | Scan successfully com |
| **6** | rscan engy 520 555 175 1 | 0.45 | -3.9 | 0 | 844.1715 | 6.893 | 8 | Ni LEG | NICKEL | LINEAR H | Scan successfully com |
| **7** | rscan engy 520 528 40 530 20 | 0.45 | -3.9 | 0 | 844.1715 | 6.893 | 7.999 | Ni LEG | NICKEL | LINEAR H | Scan successfully com |

# Summary



- Focus on the science rather than the technicalities of data plotting
- Make decisions on-the-fly
- Use all information contained in multi-channel analyzers

$\Rightarrow$ Learn how to use the program now!

# Questions & Resources



## Contact Information

Patrick Braun
patrick.braun@lightsource.ca
https://github.com/pmb399/REIXSAnalysis