# PSANA tutorial

Prerequisites

* SLAC unix account
* SSH and –X flag ready terminal
* Basic understanding of python

## The very first script

First steps are to get to the LCLS analysis servers from your computer and to source the environment.

ssh -Y USERNAME@pslogin.slac.stanford.edu

ssh -Y psana

source /reg/g/psdm/etc/ana\_env.csh

At this point it is useful to head to a working directory, e.g.

mkdir ~/psana/tutorial

cd ~/psana/tutorial

Copy the sample material into your working directory with

cp /reg/neh/home2/mbucher/public/tutorial/\* ~/psana/tutorial

Make yourself familiar with the file ‘sample.py’ by opening it in a viewer of your choice, e.g.

emacs ~/psana/tutorial/sample\_xcs.py &

Run the script like any python script with

python ~/psana/tutorial/sample\_xcs.py

Great! You just averaged and plotted a few Princeton camera frames with psana and MatPlotLib. Changing the EXP/RUN string runs the same code from different runs or experiments.

Best resource to read on:

<https://confluence.slac.stanford.edu/display/PSDM/psana+-+Python+Script+Analysis+Manual>

## Working with keys, the ‘TAB method’

Probably you want to do more than just average a TOF trace and most likely you have different detectors to read in. It’s good practice to add new detectors in the following way.

Head to iPython, an environment similar to Matlab with

ipython

and perform the following commands

from psana import \*

dataset\_name = "exp=XCS/xcstut13:run=15"

ds = DataSource(dataset\_name)

itr = ds.events()

evt = itr.next()

evt.keys()

You’ll find these lines combined as a script named ‘*detector-script.py*’ in the sample material you copied earlier into your working directory. It is a handy script that tells you all the detectors and data structures of the particular EXP/RUN string. Now that we figured out what detectors are in this run, we can ‘add’ that the detectors are read in by our analysis script with the following two commands.

fee\_src = Source("BldInfo(FEEGasDetEnergy)")

gas\_det = evt.get(Bld.BldDataFEEGasDetEnergy,fee\_src)

The LCLS gas detectors actually have 4 readings per shot. Two in front of the gas attenuator and two after the gas attenuator use the TAB method to figure your possibilities.

gas\_det.f\_21\_ENRC()

gas\_det.f\_22\_ENRC()

## MPI - Parallelization

The vast amount of data produced by LCLS will easily overwhelm a single core. A way out of this is parallelization. Let’s have another look at the ‘sample.py’ file and we find the necessary pieces of code already in the script. So let’s run it

`which mpirun` -n 2 python ~/psana/tutorial/sample\_time\_of\_flight.py

mpirun spreads the computation over all cores and is able to communicate between the cores. The EXP/RUN loop splits up all data from the run and core looks at a different range of the data. Each core is attributed a rank that one can work with by adding the following into the script at the last (careful with the indent).

if rank==0:

print ‘hello from the first core.’

if rank==1:

print ‘hello from the second core.’

## Batch jobs

After you created your script and debugged it, you’ll find yourself in a position that you’ll want to run it and basically your whole experiment. This becomes tedious in a shell and I recommend using the psana batch system. It is mostly likely compatible with your script, e.g. the LSF system supports parallel jobs, and the only thing you may want to change is the python output.

bsub -a mympi -n 24 -o %J –q psanaq python ~/psana/tutorial/sample\_time\_of\_flight.py

More about the batch system can be found here:

<https://confluence.slac.stanford.edu/display/PCDS/Submitting+Batch+Jobs>

## ‘Online’ processing

With all of this computational power available we can actually process the incoming data as it is taken, i.e. online. The scripts are almost completely reusable for offline and online analysis purposes by changing the run iterator to

ds = psana.DataSource('shmem=AMO.0:stop=no')

for run in ds.runs():

for evt in run.events():

To start a script for online analysis one has to use to monitoring nodes. At AMO this is typically server 2. So do

ssh -Y USERNAME@psdev.slac.stanford.edu (if from outside the AMO ctrl-room.)

ssh -Y amo-daq-mon02

source /reg/g/psdm/etc/ana\_env.csh

‘which mpirun’ -n 24 -s amo-daq-mon02, amo-daq-mon03, amo-daq-mon04 python sample.py

The monitoring nodes will see data when the DAQ is running and will automatically (it is programmed in the ‘.cnf’ file to do so) split the data-stream to all the running cores.