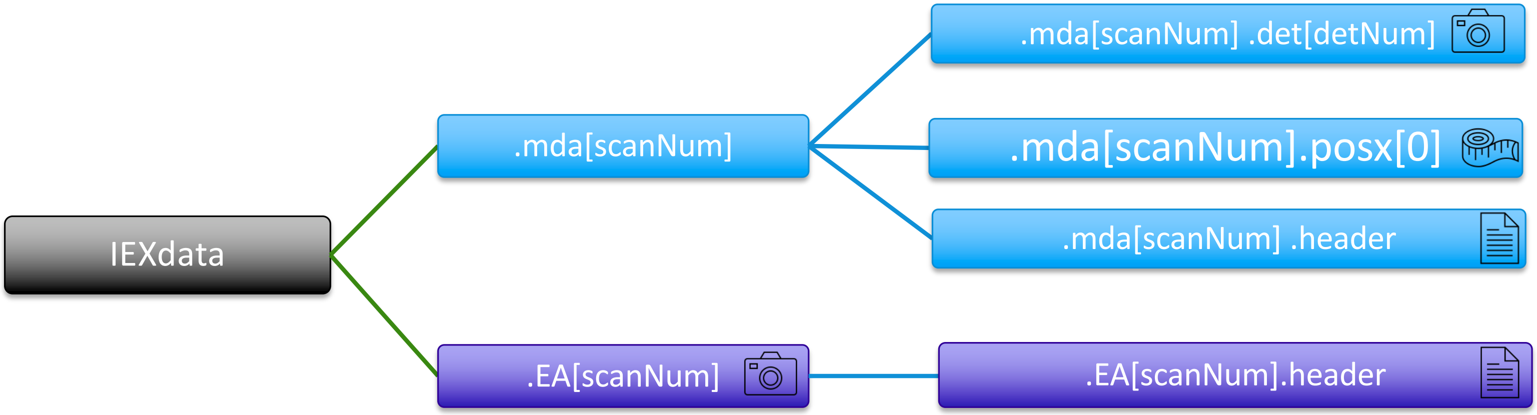
IEXdata structure



IEXdata

.mda[scanNum]

.mda[scanNum] .det[detNum]

.mda[scanNum].posx[0]

.mda[scanNum] .header

.EA[scanNum]

.EA[scanNum].header

IEXdata → data class to handle loading of mda and EA files

`Mydata=IEXdata(*scanNum,dtype='mda', **kwargs)`

`*scanNum`

scanNum → single scan
first, last → every scan from first to last inclusive
first, last, countby → every nth scan from first to last inclusive
last = inf → to load to the end of the directory
[scan1, scan2, scan3] → series of scan

`dtype` → mda by default (EA, EA_nc, nData)

**kwargs

path
prefix
suffix
nzeros
overwrite True/False; True → reloads the data, False → skips already loaded data

`mydata.update(*scanNum,**kwargs)` → loads additional scans to the dictionary (uses previous dtype and file info (path, prefix, suffix, nzeros)

`mydata.info()` → prints the loaded scans and current variable values

`remove(mydata.mda, * scanNum)` → remove scan from dictionary

`mydata.save(filename,filepath)` → to save a data set as hdf5
`reload_IEXdata(filename,filepath)` → to reload saved data

nmda → data class for mda files

`mydata= (scanNum*)`

`mydata.mda` → dictionary of mda scans

`mydata.mda[scanNum]` → dictionary of all detector data from scanNum

`myddata.mda[scanNum].det[detNum]` → pynData object

`mydata.mda[scanNum].header` → IEX header object

nEA → data class for EA files

`mydata=IEXdata(scanNum*,dtype='EA')`

`mydata.EA` → dictionary of EA scans

`mydata.EA[scanNum]` → pynData object with addition pynData_ARPES vars

`mydata.EA[scanNum].header` → IEX header object

Loading data

mda scans

Loading for the first time:

```
mydata = IEXdata(scanNum)  → single scan
mydata = IEXdata(first,last)  → every scan from first to last inclusive
mydata = IEXdata(first,last,countby)  → every nth scan from first to last inclusive inclusive
mydata = IEXdata([scan1, scan2, scan3]) → series of scan
```

Note: last = inf → to load to the end of the directory

Adding scans (uses the same scanNum syntax as above):

```
mydata.update(first,inf,overwrite=False)  → loads all unloaded scans (overwrite=True is the default and will reload the already loaded data)
```

EA scans

Loading for the first time (uses the same scanNum syntax as above):

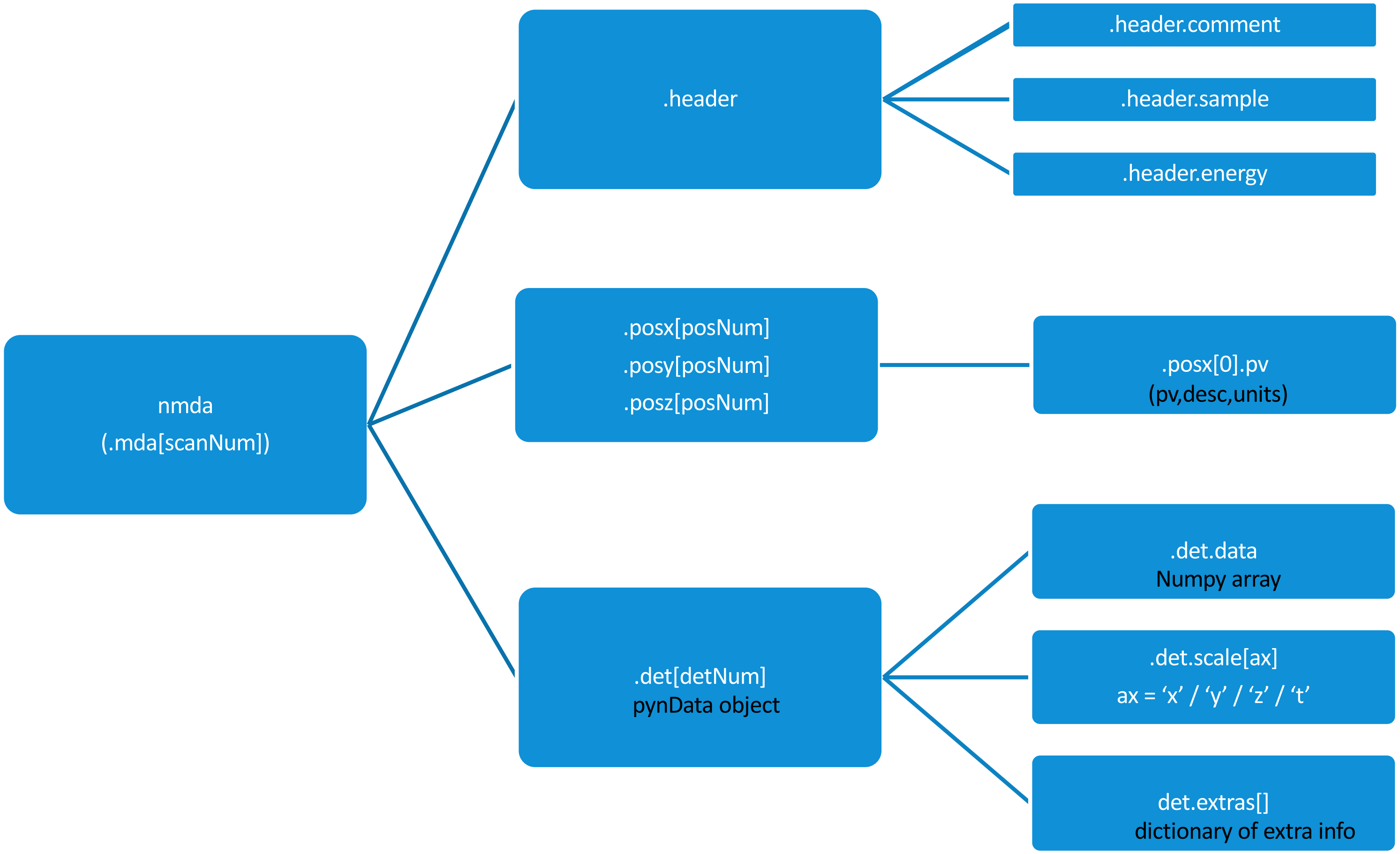
```
mydata = IEXdata(scanNum, dtype='EA')  → for new EA .h5 format
mydata = IEXdata(scanNum, dtype='EA_nc')  → for old EA .nc format
```

Adding scans (uses the same scanNum commands):

```
 mydata.update(first,inf,overwrite=False)→ loads all unloaded scans (overwrite=True is the default and will reload the already loaded data)
```

Note: update uses that last dtype for load unless otherwise specified

nmda structure



nmda
(.mda[scanNum])

.header

.header.comment

.header.sample

.header.energy

.posx[posNum]
.posy[posNum]
.posz[posNum]

.posx[0].pv
(pv,desc,units)

.det[detNum]
pynData object

.det.data
Numpy array

.det.scale[ax]
ax = ‘x’ / ‘y’ / ‘z’ / ‘t’

det.extras[]
dictionary of extra info

nmda

`nmda.fpath` → pnData_ARPES object

`nmda.header` ⟶ header object

`nmda.detAll` → print list of detectors detNum: (PV , description, units)

`nmda.det[detNum]` → nData object of detNum data (.data => data array)

`nmda.det[detNum].pv` → detector: (PV, description, units)

`nmda.posAll` → print list of positioners posNum: (PV , description, units)

`nmda.posx[posNum]` → nData object of posNum readback values (.data => data array)

`nmda.posx[posNum].pv` → positioner: (PV, description, units)

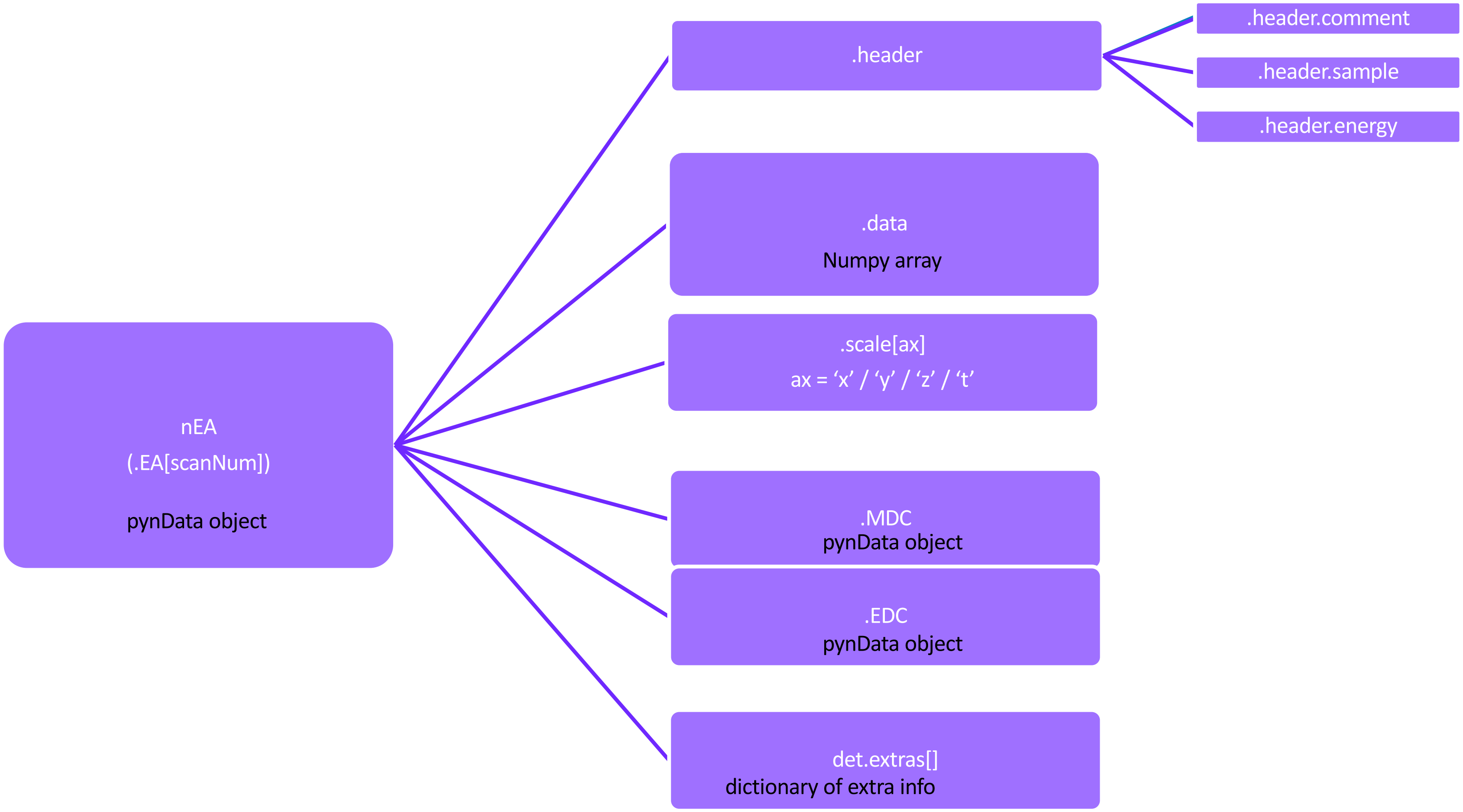
`nmda.posy[posNum], nmda.posz[posNum], nmda.post[posNum]` → for higher dimensional data

`nmda.setscalePos(‘x’,PosNum)` → sets the ’x’ / ’y’ / ’z’ scale for all detectors to the Positioner with index Pos

`nmda.setscalePos(‘x’,DetNum)` → sets the ’x’ / ’y’ / ’z’ scale for all detectors to scaling from values of DetNum

`nmda.setscaleIndex(‘x’)` → sets the ’x’ / ’y’ / ’z’ scale for all detectors index/point num

nEA structure



nEA
(.EA[scanNum])

pynData object

.header

.header.comment

.header.sample

.header.energy

.data
Numpy array

.scale[ax]
ax = ‘x’ / ‘y’ / ‘z’ / ‘t’

.MDC
pynData object

.EDC
pynData object

det.extras[]
dictionary of extra info

nEA / ndata_ARPES

nEA

`nEA` → pnData_ARPES object

`nEA.fpath` → full path to the original file

`nEA.mdafname` → mda file name

`nEA.header` → IEX header object

Ndata_ARPES

`nEA.EDC` → pnData object of angle integrated data

`nEA.MDC` → pnData object of energy integrated data

`nEA.hv` → photon energy (var or array)

`nEA.wk` → analyzer work function (var or array)

`nEA.thetaX` → polar angle (var or array)

`nEA.thetaY` → other angle (var or array)

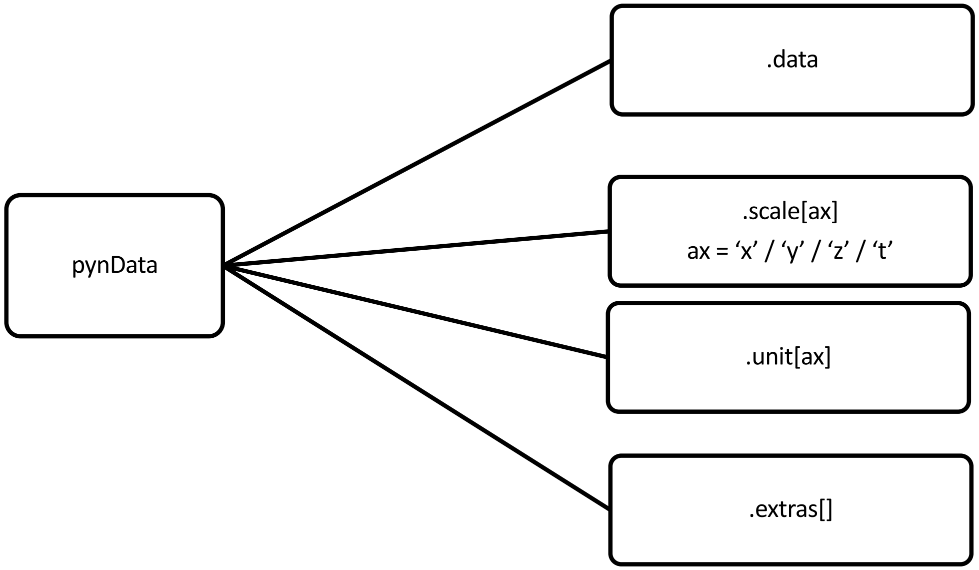
`nEA.KEscale` → array of original KE scaling

`nEA.angScale` → array of original detector angle scaling

`nEA.angOffset` → offset in original detector angle

`nEA.slitDir` → analyzer slit direction

pynData structure



pynData

.data

.scale[ax]

ax = ‘x’ / ‘y’ / ‘z’ / ‘t’

.unit[ax]

.extras[]

pynData

`nData.data` → data array

`nData.scale[‘x’]` → data array for ’x’ / ’y’ / ’z’ scale

`nData.unit[‘x’]` → units for ’x’ / ’y’ / ’z’ scale

`nData.extras` → dictionary for meta data

`nData.info()` → prints data array shape and axis info

`nData.updateAx(’x’,NewscaleArray, ‘Scale_units’)` → function for changing ’x’ / ’y’ / ’z’ scale

`nData.shiftScale(’x’,oldValue, newValue)` → function for changing ’x’ / ’y’ / ’z’ scale by calculating an offset from oldValue and newValue

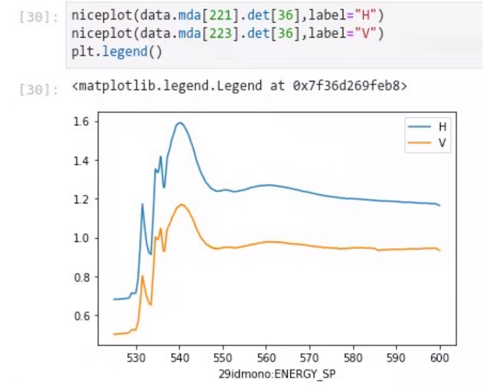
`nData.save()` → saves pynData object as an hdf5 file load_nData(fname, fdir=’’) **Note: only saves nData.data, nData.scale and nData.extras**

`load_nData(fname,fdir=‘’)` → load nData saved via .save()

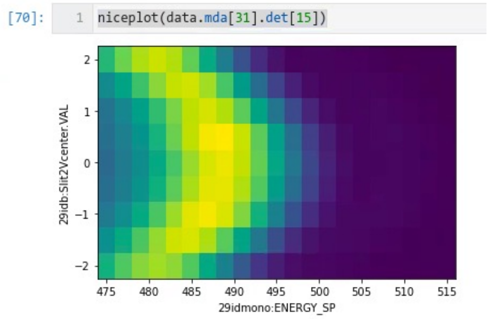
Simple plotting examples using pynData

Niceplot(pynData_object)

simple 1D and 2D plotting



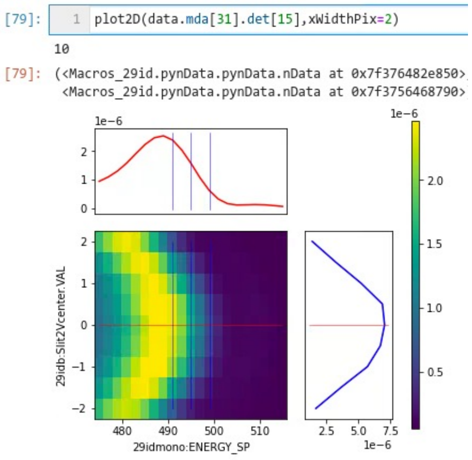
```
[30]: niceplot(data.mda[221].det[36],label="H")
      niceplot(data.mda[223].det[36],label="V")
      plt.legend()

[30]: <matplotlib.legend.Legend at 0x7f36d269feb8>
```



```
[70]: niceplot(data.mda[31].det[15])
```

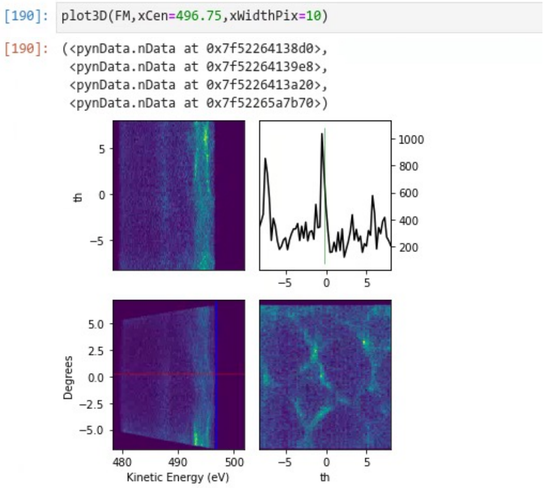
plot2D(pynData_object)

plotting with linecuts



```
[79]: plot2D(data.mda[31].det[15],xWidthPix=2)

      10

[79]: (<Macros_29id.pynData.pynData.nData at 0x7f376482e850>
       <Macros_29id.pynData.pynData.nData at 0x7f3756468790>
```

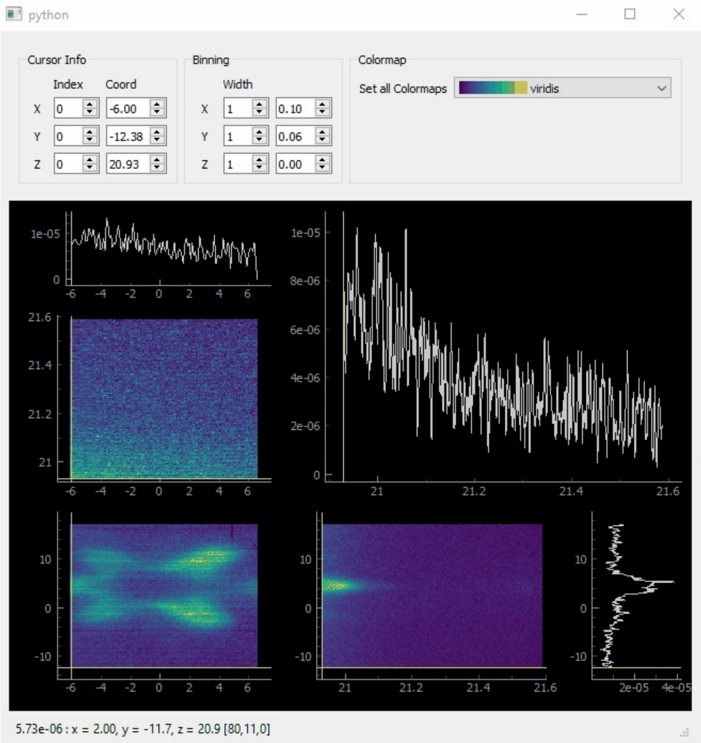
plot3D(pynData_object)



```
[190]: plot3D(FM,xCen=496.75,xWidthPix=10)

[190]: (<pynData.nData at 0x7f52264138d0>,
        <pynData.nData at 0x7f52264139e8>,
        <pynData.nData at 0x7f5226413a20>,
        <pynData.nData at 0x7f52265a7b70>)
```

IEX specific plotting

Interactive plotting



python

Cursor Info

	Index	Coord
X	0	-6.00
Y	0	-12.38
Z	0	20.93

Binning

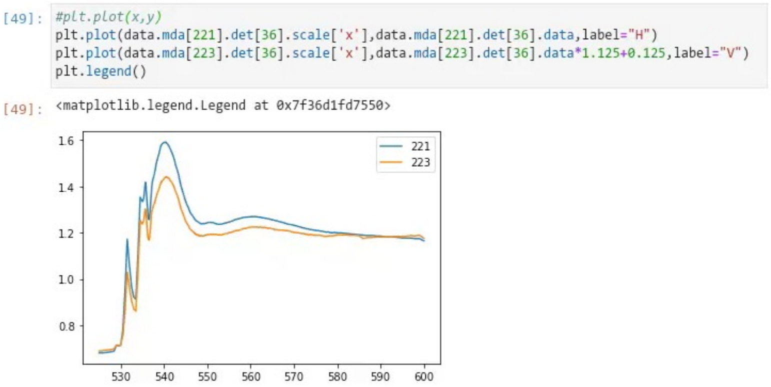
	Width	
X	1	0.10
Y	1	0.06
Z	1	0.00

Colormap

Set all Colormaps: viridis

5.73e-06 : x = 2.00, y = -11.7, z = 20.9 [80,11,0]

Python plotting



```
[49]: #plt.plot(x,y)
      plt.plot(data.mda[221].det[36].scale['x'],data.mda[221].det[36].data,label="H")
      plt.plot(data.mda[223].det[36].scale['x'],data.mda[223].det[36].data*1.125+0.125,label="V")
      plt.legend()

[49]: <matplotlib.legend.Legend at 0x7f36d1fd7550>
```

Saving experiment: IEX_nData hdf5 structure

```
h5 file
        group: mda => contains all the mda scans
           group: mda_scanNum
              . attrs -> fpath (original file path)
              group: header
                 .attrs for each header attribute
              group: det
                   group: det_1 …. -> nData with subgroups for each detector
              group: posx, posy …
                   group: posx_1 …. -> nData with subgroups for each positioner

         group: EA => contains all the EA scans
                   group: EA_scanNum -> nData with subgroups
                            group: EDC –> nData with subgroups
                            group: MDC –> nData with subgroups
                            .attrs: hv, wk, thetaX, thetaY, angOffset, slitDir
                            dataset:KEscale,angScale
                            .attrs fpath
                            .attrs mdafname
                             group: header
                            .attrs header info


    group: nDataObject
              subgroup: data
                        dataset -> data array
              subgroup: scale
                        dataset -> scale array
              subgroup: unit
              .attrs -> unit
```

Common command examples

```
f= h5py.File("/Users/jmcchesn/Documents/NoteBooks/Data/IEXmydata2.h5", 'r’)

for k in f.keys():
      print(k)

for k in f.attrs.keys():
      print('{} => {}'.format(k, f.attrs[k]))
```

View the data with python

ARPES data files

images: EA_0001.h5

motor scans: ARPES_0001.mda