```
#
# This script demonstrates the use of the TopSpin Python Interface (API)
# Copyright (c) 2022, Bruker BioSpin
#
# Usage requires Python 3.8+ environment
#
# - from shell (terminal):        python3 <script-name>
# - from TopSpin command line: xpy3 <script-name>
#

from bruker.api.topspin import Topspin
from bruker.data.nmr import *



#
# create TopSpin interface object
#

top = Topspin()
dp = top.getDataProvider()

# search the example data sets
data_list = dp.find([top.getInstallationDirectory() + '/examdata'], name="exam_CMCse_1",
expno = 1, dim=dp.FIND_1D)

# get the instance of the first data set in the list
proton = dp.getNMRData(data_list[0])

# print the data set info
print('Identifier                ',proton.getIdentifier())
print('Dataset dimension      ',proton.getDimension())
print('Processing dimension',proton.getProcDim())
print('Title',proton.getTitle())

print(proton.getInfo())

print('\nIdentifier :',proton.getIdentifier())
print('Next Expno :',dp.getNextEXPNO(proton.getIdentifier()))
print('Splitted identifier :',dp.splitIdentifier(proton.getIdentifier()))
print('Modify procno:'+dp.modifyIdentifier(proton.getIdentifier(), procno = 73))
print('Modify expno:'+dp.modifyIdentifier(proton.getIdentifier(), expno = 332, procno = 13))

# read the complete real part (1r)
print('\nReading the real part (1r)')
```

```
spec = proton.getSpecDataPoints()
print(spec)
if spec.get('exc'):
    print('Error :',spec.get('exc').details())

#
# read the imaginary part between 4 and 5 ppm
#
print("Imaginary part")
imag = proton.getSpecDataPoints(component=PROCDATA_IMAG,
physRange=[PhysicalRange(5, 4)])
print(imag)
#
# Error handling - exception key holds the error information
#
if imag.get(EXCEPTION):
    print('Error :',imag.get(EXCEPTION).details())

#
# read the first 16 points of the FID and print the result
#
print("FID")
fid = proton.getRawDataPoints(ir=[IndexRange(0, 16)])
for key in fid.keys():
    print(key + ":" + str(fid[key]))

# calculate the sum of all data points
dataPoints = fid.get(DATA_POINTS)
sum = 0
for d in dataPoints:
    sum += d

print("Calculated sum = {}".format(sum))

print("--- Finished")




#
# This script demonstrates the use of the TopSpin Python Interface (API)
# Copyright (c) 2022, Bruker BioSpin
#
```

```
# Usage requires Python 3.8+ environment
#
# - from shell (terminal):         python3 <script-name>
# - from TopSpin command line: xpy3 <script-name>
#
from bruker.api.topspin import Topspin
from bruker.data.nmr    import *


top = Topspin()
dp = top.getDataProvider()

# get the instance of NMR Data set
PROTON = top.getInstallationDirectory() + "/examdata/exam_CMCse_1/1/pdata/1/"
proton = dp.getNMRData(PROTON)

res = proton.getSpecDataPoints()

if res.get(EXCEPTION):
      print('Error :',res.get(EXCEPTION).details())
      exit(-1)
#
# read the integration regions
#
intRegions = proton.getIntegrationRegions()

if intRegions is None:
      print('Cannot read the integration region file')
      exit(-1)
#
# calculate the integrals
#
print("---- Integrals, ------")
for region in intRegions:
      start = region['start']
      end     = region['end']
      startIndex = proton.getIndexFromPhysical(start,0)
      endIndex = proton.getIndexFromPhysical(end,0)
      sum = 0
      data = res.get(DATA_POINTS)
      for i in range(startIndex,endIndex):
            sum = sum + data[i]

      print(" %14.2f    %6.3f    %6.3f" %(sum,start,end))
```

```
#
# calculate the integrals, read the data regions individually from server.
#
print("---- Integrals, data regions from server ------")
for region in intRegions:
        start = region['start']
        end     = region['end']
        data = proton.getSpecDataPoints(physRange = [PhysicalRange(start,end)])

        sum = 0
        for d in data['dataPoints']:
                sum = sum + d

        print(" %14.2f    %6.3f    %6.3f" %(sum,start,end))

print("--- Finished")



#
# This script demonstrates the use of the TopSpin Python Interface (API)
# Copyright (c) 2022, Bruker BioSpin
#
# Usage requires Python 3.8+ environment
#
# - from shell (terminal):         python3 <script-name>
# - from TopSpin command line: xpy3 <script-name>
#
from bruker.api.topspin import Topspin
from bruker.data.nmr    import *

from datetime import datetime

from matplotlib import pyplot as plt
import numpy
import sys



#
# Demonstrates the use of the Topspin Python API
# The spectra are plotted using Matplotlib
#
top = Topspin()
dp = top.getDataProvider()
```

```
try:
    PROTON = sys.argv[1]
except:
    PROTON = top.getInstallationDirectory() + "/examdata/exam_CMCse_1/1/pdata/1"

#
# get the data set info
#
startTime = datetime.now()

proton = dp.getNMRData(PROTON)

if proton == None:
    raise Exception('Dataset {} does not exists'.format(PROTON))

endTime = datetime.now()

print("Dataset    :" + proton.getIdentifier())
print("Dimension:" + str(proton.getDimension()))
print(proton.getInfo())
print("Elapsed Time (Data set object created):" + str(endTime - startTime))

#
#   get the complete FID and Spectrum (Real,Imaginary)
#
startTime = datetime.now()

rawData    = proton.getRawDataPoints()

specData = proton.getSpecDataPoints()
if specData.get(EXCEPTION):
    print('Error :',specData.get(EXCEPTION).details())
    exit(-1)

imagData = proton.getSpecDataPoints(component= PROCDATA_IMAG)

endTime = datetime.now()
print("Elapsed Time (3 component read):" + str(endTime - startTime))


# plot FID
# this avoids overlapping of subplots
```

```
plt.subplots(2,1, constrained_layout = True)
plt.subplot(2,1,1)
pr = rawData['physicalRanges'][0]
left = float(pr['start'])
right = float(pr['end'])
axis = numpy.linspace(left,right,len(rawData['dataPoints']))
# plot spectrum - both real and imaginary part
plt.plot(axis,rawData['dataPoints'])

# plot spectrum
plt.subplot(2,1,2)

pr = specData['physicalRanges'][0]
left = float(pr['start'])
right = float(pr['end'])
# necessary to inverse the x axis (NMR standard)
plt.xlim(left,right)

axis = numpy.linspace(left,right,len(specData['dataPoints']))
# plot spectrum - both real and imaginary part
plt.plot(axis,specData['dataPoints'])

# plot integration regions
yValue = -50000
region = proton.getIntegrationRegions()

if region is not None:
    for region in proton.getIntegrationRegions():
        plt.plot([region['start'],region['end']],[yValue,yValue],'C3')

# plot peak positions
for peak in proton.getPeakList():
    plt.plot([peak['position'],peak['position']],[0,yValue],'C4')


plt.show()
```