Interact with files

January 10, 2021

[1]:
```
# The following is to know when this notebook has been run and with which␣
↪python version.
import time, sys
print(time.ctime())
print(sys.version.split('|')[0])
```

```
Mon Oct 12 21:36:54 2020
3.7.6 (default, Jan  8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)]
```

1 C: How to read and write files (ASCII and FITS)

This is part of the Python lecture given by Christophe Morisset at IA-UNAM.

Some informations are here: http://www.tutorialspoint.com/python/python_files_io.htm

1.1 Reading a simple ascii file

[2]:
```
# numpy is needed in some part of the lecture
import numpy as np
```

First of all, we will have to have some files on the hard drive to read them The following notebook cell will write a file in the same directory where the notebook has been started.

[3]:
```
%%writefile data1.dat
1   2.3  6   8 star
2   3.5  7   9 galaxy
3  -4.2  5   7 cluster
```

```
Overwriting data1.dat
```

Now the goal is to read this file. The first way is to open the file, read it completely in a variable and close the file. Then we can play with the content of the file.

[4]:
```
datafile = open('data1.dat', 'r') # Open the file to read it
```

[5]:
```
data = datafile.readlines() # The variable data will receive the content of the␣
↪file.
```

[6]:
```
datafile.close() # Not need anymore of the file.
```

[7]:
```
print(type(data)) # The data file is stored in the form of a list, each element␣
↪of the list corresponding to a row of the list.
```

```
<class 'list'>
```

[8]:
```
print(data) # Each row is a string and terminates with \n, symbol of END OF␣
↪LINE.
```

```
['1   2.3  6   8 star\n', '2   3.5  7   9 galaxy\n', '3  -4.2  5   7 cluster\n']
```

[9]:
```
print(len(data)) # number of rows
```

```
3
```

[10]:
```
print(data[0], 'tralala')
```

```
1   2.3  6   8 star
 tralala
```

[11]:
```
for row in data:
    print(row)
```

```
1   2.3  6   8 star

2   3.5  7   9 galaxy

3  -4.2  5   7 cluster
```

[12]:
```
# In python 2:
for row in data:
    print(row),
```

```
1   2.3  6   8 star

2   3.5  7   9 galaxy

3  -4.2  5   7 cluster
```

[13]:
```
# In python 3:
for row in data:
    print(row, end='')
```

```
1   2.3  6   8 star
2   3.5  7   9 galaxy
```

```
3  -4.2  5   7 cluster
```

[14]:
```
print(type(data[0])) # Each element is a string
```

```
<class 'str'>
```

Now it is easy to separate each field with the split command:

[15]:
```
for row in data:
    print(row.split())
```

```
['1', '2.3', '6', '8', 'star']
['2', '3.5', '7', '9', 'galaxy']
['3', '-4.2', '5', '7', 'cluster']
```

[16]:
```
# One can also transform the data if the type is known:
for row in data:
    this_data = row.split()
    print('N = {0:2d} f = {1:5.2f} type = {2:>10s}'.format(int(this_data[0]),
                                                           float(this_data[1]),
                                                           this_data[4]))
```

```
N =  1 f =  2.30 type =       star
N =  2 f =  3.50 type =     galaxy
N =  3 f = -4.20 type =    cluster
```

[18]:
```
# One can even fill a list with the data, by column:
N = []
f = []
type_ = [] # take care, type is a python command, you can erase it if you use␣
 ↪it...
for row in data:
    this_data = row.split()
    N.append(int(this_data[0]))
    f.append(float(this_data[1]))
    type_.append(this_data[4])
print(N)
print(f)
print(type_)
N = np.array(N)
print(N)
```

```
[1, 2, 3]
[2.3, 3.5, -4.2]
['star', 'galaxy', 'cluster']
[1 2 3]
```

[19]:
```
# If the file number of rows is not too big, you can use list comprehension␣
↪(and even send the result to a numpy array)
N = np.array([int(row.split()[0]) for row in data])
f = np.array([float(row.split()[1]) for row in data])
# Each one of this command scans all the rows, don't use for huge files
print(N)
print(f)
```

```
[1 2 3]
[ 2.3  3.5 -4.2]
```

1.2 How to treat special rows (headers, comments)

[20]:
```
%%writefile data2.dat
# The following data are for test purpose
N    f   x   y type
1   2.3  6   8 star
2   3.5  7   9 galaxy
3  -4.2  5   7 cluster
#4  -10.5  5  7 test
```

```
Overwriting data2.dat
```

[21]:
```
!cat data2.dat # Just to check that the # comments are also in the file
```

```
# The following data are for test purpose
N    f   x   y type
1   2.3  6   8 star
2   3.5  7   9 galaxy
3  -4.2  5   7 cluster
#4  -10.5  5  7 test
```

The file has to be read row by row, to be sure that special cases are treated.

[22]:
```
datafile = open('data2.dat', 'r') # Open the file to read it

row = datafile.readline() # this reads only one line
first_comment = row
print(first_comment, end='')

row = datafile.readline() # this reads only one line
header = row
print(header, end='')

data = []
while True: # loops until exit by break command
    row = datafile.readline()
    if row == '':
```

```
        break
    if row[0] != '#' and row[0] != '\n': # comment lines are skipped
        data.append(row)
datafile.close()
print(data)
```

```
# The following data are for test purpose
N    f    x    y type
['1   2.3  6    8 star\n', '2   3.5  7    9 galaxy\n', '3  -4.2  5    7 cluster\n']
```

[23]:
```
datafile = open('data2.dat', 'r') # Open the file to read it
row = datafile.readline() # this reads only one line
first_comment = row
print(first_comment, end='')
row = datafile.readline() # this reads only one line
header = row
print(header, end='')
data = []
row = datafile.readline()
while row != '': # loops until exit by break command
    if row[0] != '#': # comment lines are skipped
        data.append(row)
    row = datafile.readline()
datafile.close()
print(data)
```

```
# The following data are for test purpose
N    f    x    y type
['1   2.3  6    8 star\n', '2   3.5  7    9 galaxy\n', '3  -4.2  5    7 cluster\n']
```

[24]:
```
# very shorter way to deal with the file. No need to look for the end of the␣
 ↪file.
datafile = open('data2.dat', 'r') # Open the file to read it
data = []
for row in datafile:
    if row[0] != '#': # comment lines are skipped
        data.append(row)
datafile.close()
print(data)
# This way will include the header in the data... Not what we want
```

```
['N    f    x    y type\n', '1   2.3  6    8 star\n', '2   3.5  7    9 galaxy\n', '3
-4.2  5    7 cluster\n']
```

[27]:
```
# very shorter way to deal with the file:
# we know that the header is the first no-comment line in the file.
datafile = open('data2.dat', 'r') # Open the file to read it
```

```
data = []
comments = [] # we can keep the comments for some usage
header_not_read = True # We will turn it to True once the header is read
for row in datafile:
    if row[0] != '#': # comment lines are skipped
        if header_not_read:
            header = row
            header_not_read = False # next time, data will be read
        else:
            data.append(row)
    else:
        comments.append(row)
datafile.close()
print(header, end='')
print('-----')
print(data)
print('-----')
print(comments)
```

```
N    f    x    y type
-----
['1   2.3  6   8 star\n', '2   3.5  7   9 galaxy\n', '3  -4.2  5   7 cluster\n']
-----
['# The following data are for test purpose\n', '#4  -10.5  5  7 test\n']
```

[28]:
```
# Alternative way using "with". No need to close the file, done when the "with"␣
 ↪block is terminated.
data = []
comments = []
header_read = False
def change_type(row_split):
    # This function change the type of the data read from the file from 5␣
 ↪strings into int, 3 floats and a string
    # It also return the result in form of a tuple
    return (int(row_split[0]),
            float(row_split[1]),
            float(row_split[2]),
            float(row_split[3]),
            row_split[4])
with open('data2.dat', 'r') as datafile:
    for row in datafile:
        if row[0] != '#' and row[0] != '\n': # comment lines are skipped
            if not header_read:
                header = row
                header_read = True
            else:
                data.append(change_type(row.split()))
```

```
        else:
            comments.append(row)
print(header)
print(data)
print(comments)
```

```
N    f   x   y type

[(1, 2.3, 6.0, 8.0, 'star'), (2, 3.5, 7.0, 9.0, 'galaxy'), (3, -4.2, 5.0, 7.0,
'cluster')]
['# The following data are for test purpose\n', '#4  -10.5  5  7 test\n']
```

[29]:
```
# We can define the result as a structured array
# We use the header to define the field names.
# data must be a list of tuples.
a = np.array(data, dtype={'names':header.split(),
                          'formats':['i4','f16', 'f16', 'f16', 'U10']})
```

[30]:
```
a
```

[30]:
```
array([(1,  2.3, 6., 8., 'star'), (2,  3.5, 7., 9., 'galaxy'),
       (3, -4.2, 5., 7., 'cluster')],
      dtype=[('N', '<i4'), ('f', '<f16'), ('x', '<f16'), ('y', '<f16'), ('type',
'<U10')])
```

[31]:
```
print(data[0])
```

```
(1, 2.3, 6.0, 8.0, 'star')
```

[32]:
```
print(a[0])
```

```
(1, 2.3, 6., 8., 'star')
```

[33]:
```
# Easy access to the columns, by their name
print(a['N'])
```

```
[1 2 3]
```

[34]:
```
print(a['type'])
```

```
['star' 'galaxy' 'cluster']
```

[36]:
```
# Easy combine the values of columns
print(np.sqrt(a['x']**2 + a['y']**2))
```

```
[10.         11.40175425  8.60232527]
```

1.2.1 Using numpy loadtxt

http://docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html

[37]:
```
# Fast way for reading the file
# One hace to tell to skip the 2 first rows
# skiprows
b = np.loadtxt('data2.dat', skiprows=2, dtype='i4,f, f, f, U10')
```

[38]:
```
print(b)
```

```
[(1,  2.3, 6., 8., 'star') (2,  3.5, 7., 9., 'galaxy')
 (3, -4.2, 5., 7., 'cluster')]
```

[39]:
```
type(b)
```

[39]:
```
numpy.ndarray
```

[40]:
```
# The names of the columns are f0, f1, f2, etc
b.dtype
```

[40]:
```
dtype([('f0', '<i4'), ('f1', '<f4'), ('f2', '<f4'), ('f3', '<f4'), ('f4',
'<U10')])
```

1.2.2 Using numpy genfromtxt

http://docs.scipy.org/doc/numpy/reference/generated/numpy.genfromtxt.html

[41]:
```
# Fast and versatile way to read the file
# the names are taken from the file
# The types are defined automatically when reading the columns
c = np.genfromtxt('data2.dat', names=True, dtype=None, skip_header=1)
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:4: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
  after removing the cwd from sys.path.
```

[42]:
```
print(c)
```

```
[(1,  2.3, 6, 8, b'star') (2,  3.5, 7, 9, b'galaxy')
 (3, -4.2, 5, 7, b'cluster')]
```

[43]:
```
type(c)
```

[43]:
```
numpy.ndarray
```

[44]:
```
c.dtype
```

[44]:
```
dtype([('N', '<i8'), ('f', '<f8'), ('x', '<i8'), ('y', '<i8'), ('type', 'S7')])
```

[45]:
```
c['f']
```

[45]:
```
array([ 2.3,  3.5, -4.2])
```

Now a value of x is missing (not possible with space separator, so we use "," as separator):

[47]:
```
%%writefile data3.dat
# The following data are for test purpose
N,    f,   x,   y, type
1,   2.3,  6,   8, star
2,   3000.5,   ,  9, galaxy
3,  -4.2,  5,   7, cluster
#4,  -10.5,  5,  7, test
```

```
Overwriting data3.dat
```

[48]:
```
d = np.genfromtxt('data3.dat', names=True, dtype=None, skip_header=1,
                  delimiter=',')
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
```

[49]:
```
# The missing value has been changed to -1
d
```

[49]:
```
array([(1,  2.3000e+00,  6, 8, b' star'),
       (2,  3.0005e+03, -1, 9, b' galaxy'),
       (3, -4.2000e+00,  5, 7, b' cluster')],
      dtype=[('N', '<i8'), ('f', '<f8'), ('x', '<i8'), ('y', '<i8'), ('type',
'S8')])
```

[52]:
```
# Th emissing value can be set to whatever you want (but non a NaN here, as the␣
 ↪typ eis integer, and NaN is a float...)
d = np.genfromtxt('data3.dat', names=True, dtype=None, skip_header=1,␣
 ↪delimiter=',',
                  filling_values=999)
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[53]:
```
d['x'][1]
```

[53]: 999

[54]:
```
# ons can select the columns to be store
e = np.genfromtxt('data3.dat', names=True, dtype=None, skip_header=1,
                  delimiter=',',usecols=(0,1,4))
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[55]:
```
print(e)
```

```
[(1,  2.3000e+00, b' star') (2,  3.0005e+03, b' galaxy')
 (3, -4.2000e+00, b' cluster')]
```

[60]:
```
# ons can select the columns to be store
N, f, typ = np.genfromtxt('data3.dat', skip_header=2,
                  delimiter=',',usecols=(0,1,4), unpack=True)
```

[61]:
```
# The resulting array now contains only the given columns
print(N)
print(f)
```

```
[1. 2. 3.]
[ 2.3000e+00  3.0005e+03 -4.2000e+00]
```

1.2.3 Using recfrom to obtain a record array

[62]:
```
# Uses the same keywords than genfromtxt
f = np.recfromtxt('data3.dat', names=True, dtype=None, skip_header=1,
                  delimiter=',',usecols=("N", "f", "type"))
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/numpy/lib/npyio.py:2336: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
  output = genfromtxt(fname, **kwargs)
```

[63]:
```
f
```

[63]:
```
rec.array([(1,  2.3000e+00, b' star'), (2,  3.0005e+03, b' galaxy'),
           (3, -4.2000e+00, b' cluster')],
          dtype=[('N', '<i8'), ('f', '<f8'), ('type', 'S8')])
```

[64]:
```
f.N
```

[64]:
```
array([1, 2, 3])
```

1.3 Fixed size ascii files

[65]:
```
%%writefile data4.dat
#  Line      Iobs    lambda  relat_error Obs_code
H  1  4861A 1.00000    4861. 0.08000  Anabel
H  1  6563A 2.8667     6563. 0.19467  Anabel
H  1  4340A 0.4933     4340. 0.03307  Anabel
H  1  4102A 0.2907     4102. 0.02229  Anabel
H  1  3970A 0.1800     3970. 0.01253  Anabel
N  2  6584A 2.1681     6584. 0.08686  Anabel
N  2 121.7m 0.0044621217000. 0.20000  Liu
O  1  6300A 0.0147     6300. 0.00325  Anabel
TOTL  2326A 0.07900    2326. 0.20000  Adams
C  2 157.6m 0.00856 1576000. 0.20000  Liu
O  1 63.17m 0.13647  631700. 0.10000  Liu
O  1 145.5m 0.00446 1455000. 0.200    Liu
TOTL  3727A 0.77609    3727. 0.200    Torres-Peimbert
S II  4070A 0.06174    4070. 0.200    Torres-Peimbert
S II  4078A 0.06174    4078. 0.200    Torres-Peimbert
```

```
Overwriting data4.dat
```

[76]:
```
# Here we cannot use SPACE as a separator, as some strings contains spaces.
# "delimiter" is used to specify the size (in characters in the file) of each␣
 ↪variables.
# The types must be clearly defined too.
obs  = np.genfromtxt('data4.dat',
                     dtype=["U11","float","float","float","U2"],
                     delimiter=[11,9,8,10,2],
                     names = True
                     )
```

[77]:
```
obs # The same delimiter (fixed sizes) is applied to the names. May not be what␣
 ↪you want:
```

[77]:
```
array([('H  1  4861A', 1.      ,    4861., 0.08   , 'An'),
       ('H  1  6563A', 2.8667  ,    6563., 0.19467, 'An'),
       ('H  1  4340A', 0.4933  ,    4340., 0.03307, 'An'),
       ('H  1  4102A', 0.2907  ,    4102., 0.02229, 'An'),
       ('H  1  3970A', 0.18    ,    3970., 0.01253, 'An'),
       ('N  2  6584A', 2.1681  ,    6584., 0.08686, 'An'),
       ('N  2 121.7m', 0.004462, 1217000., 0.2    , 'Li'),
       ('O  1  6300A', 0.0147  ,    6300., 0.00325, 'An'),
```

```
       ('TOTL  2326A', 0.079   ,    2326., 0.2    , 'Ad'),
       ('C  2 157.6m', 0.00856 , 1576000., 0.2    , 'Li'),
       ('O  1 63.17m', 0.13647 ,  631700., 0.1    , 'Li'),
       ('O  1 145.5m', 0.00446 , 1455000., 0.2    , 'Li'),
       ('TOTL  3727A', 0.77609 ,    3727., 0.2    , 'To'),
       ('S II  4070A', 0.06174 ,    4070., 0.2    , 'To'),
       ('S II  4078A', 0.06174 ,    4078., 0.2    , 'To')],
      dtype=[('Line', '<U11'), ('Iobs', '<f8'), ('lambda', '<f8'),
('relat_erro', '<f8'), ('r', '<U2')])
```

[78]:
```
# Defining the names:
obs2  = np.genfromtxt('data4.dat', skip_header=1,
                      dtype=None,
                      delimiter=[11,9,8,10,2],
                      names = ['label', 'i_obs', 'lambda', 'e_obs', 'observer']
                      )
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
  """
```

[79]:
```
obs2
```

[79]:
```
array([(b'H  1  4861A', 1.      ,    4861., 0.08   , b'An'),
       (b'H  1  6563A', 2.8667  ,    6563., 0.19467, b'An'),
       (b'H  1  4340A', 0.4933  ,    4340., 0.03307, b'An'),
       (b'H  1  4102A', 0.2907  ,    4102., 0.02229, b'An'),
       (b'H  1  3970A', 0.18    ,    3970., 0.01253, b'An'),
       (b'N  2  6584A', 2.1681  ,    6584., 0.08686, b'An'),
       (b'N  2 121.7m', 0.004462, 1217000., 0.2    , b'Li'),
       (b'O  1  6300A', 0.0147  ,    6300., 0.00325, b'An'),
       (b'TOTL  2326A', 0.079   ,    2326., 0.2    , b'Ad'),
       (b'C  2 157.6m', 0.00856 , 1576000., 0.2    , b'Li'),
       (b'O  1 63.17m', 0.13647 ,  631700., 0.1    , b'Li'),
       (b'O  1 145.5m', 0.00446 , 1455000., 0.2    , b'Li'),
       (b'TOTL  3727A', 0.77609 ,    3727., 0.2    , b'To'),
       (b'S II  4070A', 0.06174 ,    4070., 0.2    , b'To'),
       (b'S II  4078A', 0.06174 ,    4078., 0.2    , b'To')],
      dtype=[('label', 'S11'), ('i_obs', '<f8'), ('lambda', '<f8'), ('e_obs',
'<f8'), ('observer', 'S2')])
```

[80]:
```
%%writefile data5.dat
#  Line      Iobs    lambda  relat_error Obs_code
H  1  4861A 1.00000    4861. 0.08000 x Anabel
H  1  6563A 2.8667     6563. 0.19467 x Anabel
```

```
H  1  4340A 0.4933     4340. 0.03307 x Anabel
H  1  4102A 0.2907     4102. 0.02229 x Anabel
H  1  3970A 0.1800     3970. 0.01253 t Anabel
N  2  6584A 2.1681           0.08686 x Anabel
N  2 121.7m 0.00446 1217000. 0.20000 g Liu
O  1  6300A 0.0147     6300. 0.00325 t Anabel
TOTL  2326A 0.07900    2326. 0.20000 g Adams
C  2 157.6m 0.00856 1576000. 0.20000 t Liu
O  1 63.17m 0.13647  631700. 0.10000 g Liu
O  1 145.5m 0.00446 1455000. 0.200   g Liu
TOTL  3727A 0.77609    3727. 0.200   g Torres-Peimbert
S II  4070A 0.06174    4070. 0.200   g Torres-Peimbert
S II  4078A 0.06174    4078. 0.200   g Torres-Peimbert
```

```
Overwriting data5.dat
```

[81]:
```
# Here we want to skip one column:
obs3  = np.genfromtxt('data5.dat', skip_header=1,
                      dtype=None,
                      delimiter=[11, 8, 9, 9, 2, 2],
                      names = ['label', 'i_obs', 'lambda', 'e_obs', 'na',␣
↪'observer'],
                      usecols = (0, 1, 2, 3, 5)
                      )
```

```
/Users/christophemorisset/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: VisibleDeprecationWarning: Reading unicode
strings without specifying the encoding argument is deprecated. Set the
encoding, use None for the system default.
```

[82]:
```
obs3
```

[82]:
```
array([(b'H  1  4861A', 1.     ,    4861., 0.08   , b'An'),
       (b'H  1  6563A', 2.8667 ,    6563., 0.19467, b'An'),
       (b'H  1  4340A', 0.4933 ,    4340., 0.03307, b'An'),
       (b'H  1  4102A', 0.2907 ,    4102., 0.02229, b'An'),
       (b'H  1  3970A', 0.18   ,    3970., 0.01253, b'An'),
       (b'N  2  6584A', 2.1681 ,      nan, 0.08686, b'An'),
       (b'N  2 121.7m', 0.00446, 1217000., 0.2    , b'Li'),
       (b'O  1  6300A', 0.0147 ,    6300., 0.00325, b'An'),
       (b'TOTL  2326A', 0.079  ,    2326., 0.2    , b'Ad'),
       (b'C  2 157.6m', 0.00856, 1576000., 0.2    , b'Li'),
       (b'O  1 63.17m', 0.13647,  631700., 0.1    , b'Li'),
       (b'O  1 145.5m', 0.00446, 1455000., 0.2    , b'Li'),
       (b'TOTL  3727A', 0.77609,    3727., 0.2    , b'To'),
       (b'S II  4070A', 0.06174,    4070., 0.2    , b'To'),
```

```
       (b'S II  4078A', 0.06174,    4078., 0.2    , b'To')],
      dtype=[('label', 'S11'), ('i_obs', '<f8'), ('lambda', '<f8'), ('e_obs',
'<f8'), ('observer', 'S2')])
```

[83]:
```
obs3['lambda']
```

[83]:
```
array([   4861.,    6563.,    4340.,    4102.,    3970.,      nan,
       1217000.,    6300.,    2326., 1576000.,  631700., 1455000.,
          3727.,    4070.,    4078.])
```

[84]:
```
new_obs3 = obs3.view(np.recarray)
```

[85]:
```
new_obs3.label
```

[85]:
```
array([b'H  1  4861A', b'H  1  6563A', b'H  1  4340A', b'H  1  4102A',
       b'H  1  3970A', b'N  2  6584A', b'N  2 121.7m', b'O  1  6300A',
       b'TOTL  2326A', b'C  2 157.6m', b'O  1 63.17m', b'O  1 145.5m',
       b'TOTL  3727A', b'S II  4070A', b'S II  4078A'], dtype='|S11')
```

[86]:
```
new_obs3.lambda # lambda is reserved!!!
```

```
  File "<ipython-input-86-ed3c65ebb4cc>", line 1
    new_obs3.lambda # lambda is reserved!!!
                  ^
SyntaxError: invalid syntax
```

[87]:
```
new_obs3['lambda']
```

[87]:
```
array([   4861.,    6563.,    4340.,    4102.,    3970.,      nan,
       1217000.,    6300.,    2326., 1576000.,  631700., 1455000.,
          3727.,    4070.,    4078.])
```

Using masks on the structured array.

[88]:
```
mask_observer = (obs3['observer'] == b'An') & (np.isfinite(obs3['lambda']))
print(obs3[mask_observer])
```

```
[(b'H  1  4861A', 1.    , 4861., 0.08   , b'An')
 (b'H  1  6563A', 2.8667, 6563., 0.19467, b'An')
 (b'H  1  4340A', 0.4933, 4340., 0.03307, b'An')
 (b'H  1  4102A', 0.2907, 4102., 0.02229, b'An')
 (b'H  1  3970A', 0.18  , 3970., 0.01253, b'An')
 (b'O  1  6300A', 0.0147, 6300., 0.00325, b'An')]
```

[89]:
```
for o in obs3[mask_observer]:
    print('line {0[label]:4s}, wavelength={0[lambda]}A Intensity={0[i_obs]:5.
↪3f}+/-{1:4.1f}%)'.format(o, o['e_obs']*100))
```

```
line b'H  1  4861A', wavelength=4861.0A Intensity=1.000+/- 8.0%)
line b'H  1  6563A', wavelength=6563.0A Intensity=2.867+/-19.5%)
line b'H  1  4340A', wavelength=4340.0A Intensity=0.493+/- 3.3%)
line b'H  1  4102A', wavelength=4102.0A Intensity=0.291+/- 2.2%)
line b'H  1  3970A', wavelength=3970.0A Intensity=0.180+/- 1.3%)
line b'O  1  6300A', wavelength=6300.0A Intensity=0.015+/- 0.3%)
```

1.4 Writing files

1.4.1 Simple "write" method from "open" class

[94]:
```
f = open('data10.dat', 'w')
```

[95]:
```
f.write('tralala')
f.write('trololo')
```

[95]: 7

[97]:
```
!cat 'data10.dat' # the writing method put everything together.
```

[98]:
```
f.close()
```

[99]:
```
!cat 'data10.dat' # the writing method put everything together.
```

```
tralalatrololo
```

[100]:
```
f = open('data11.dat', 'w')
f.write('tralala\n') # \n to indicate end of line
f.write('trololo\n')
f.close()
!cat 'data11.dat'
```

```
tralala
trololo
```

[101]:
```
f = open('data11.dat', 'a') # Append to the edn of the file
f.write('trilili\n') # \n to indicate end of line
f.write('trululu\n')
f.close()
!cat 'data11.dat'
```

```
tralala
trololo
```

```
trilili
trululu
```

[102]:
```
a = 'Smith'
b = 3
with open('data12.dat', 'w') as datafile:
    datafile.write("""Hola Sr. {0}
This is a file
with a lot of lines.
It is easy to write it.
The value of your data is {1}.
""".format(a, b))
!cat "data12.dat"
```

```
Hola Sr. Smith
This is a file
with a lot of lines.
It is easy to write it.
The value of your data is 3.
```

1.4.2 Using pickle (and cpickle) python specific format

[103]:
```
# Let's define some stuffs we want to keep in a file (data and variable names)
a = 5
b = 'Hola'
c = np.array([1,2,3,4,5])
def d(x):
    """ Function mia"""
    return x**2
```

[104]:
```
import pickle # The module we will use for this
```

[105]:
```
pickle.dump((a,b,c,d), open('Demo.pickle','wb')) # Writing the variables
```

[106]:
```
res = pickle.load(open('Demo.pickle', 'rb'))
```

[107]:
```
type(res)
```

[107]:
```
tuple
```

[108]:
```
print(res[0])
print(res[1])
print(res[2])
```

```
5
Hola
[1 2 3 4 5]
```

[109]:
```
res[3](5)
```

[109]: 25

[110]:
```
a2,b2,c2,d2 = pickle.load(open('Demo.pickle', 'rb'))
```

[111]:
```
a2
```

[111]: 5

[112]:
```
d2(10)
```

[112]: 100

[113]:
```
help(d2)
```

```
Help on function d in module __main__:

d(x)
    Function mia
```

[114]:
```
%timeit res = pickle.load(open('Demo.pickle', 'rb'))
```

```
40.6 µs ± 2.11 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

[115]:
```
import gzip
pickle.dump((a,b,c,d), gzip.open('Demo.pklz','wb')) # Writing the variables
```

[116]:
```
f = gzip.open('Demo.pklz','rb')
a, b, c, d = pickle.load(f)
f.close()
```

1.4.3 FITS files

What is the FITS format? The FITS format is the most popular way to save and interchange astronomical data. The files are organized in units each of which contains a human readable header and a data. This structure is refereed as HDUs (Header/DATA Unit).

A FITS file can contain one or more HDUs, the first of which is called ”primary” and the rest are called ”extensions”. The primary HDU usually contains 1D spectrum, 2D image or 3D data cube, although any dimension from 0 to 999 are possible. The data are 1, 2 or 4 bytes integers or 4 or 8 bytes real numbers.

The extensions can contain or arrays as in the primary HDU or ascii tables or binary tables. If a FITS file contains only tables, it primary HDU does not contain data, but only header.

Both headers and data in a FITS file are organized in blocs of 2880 bytes. The header contain 80 bytes lines each of which consists of a keyword of 8 bytes followed in most of the cases by ’= ’ in

the position 9 and 10 and then the value of the keyword. The rest of the line is a comment string beginning with '/'. Each header begins with the following lines

SIMPLE = T / file conforms to FITS standard BITPIX = 16 / number of bits per data pixel NAXIS = 2 / number of data axes NAXIS1 = 440 / length of data axis 1 NAXIS2 = 300 / length of data axis 2

which defines the format of the file as standard FITS, the data format and the dimensions of the stored data.

One block of 2880 bytes contains 36 lines of 80 characters per line. The header can have several blocks of 36 lines. The last block is identified by the presence of the keyword 'END' The next 2880 bytes block contains the first part of the data. The empty lines after 'END' keyword are filled with blanks and the unused bytes from the end of the data to the end of the 2880 bytes block are filled with NULLs.

[117]:
```
import astropy
print(astropy.__version__)
```

```
4.0
```

[118]:
```
from astropy.io import fits
```

[ ]:
```
# All of the functionality of PyFITS is now available in Astropy
# from astropy.io import fits as pyfits
```

Manual here: https://pythonhosted.org/pyfits/

We will use one FITS files from San Pedro Martir echelle spectrograph. The file can be downloaded from: https://github.com/Morisset/Python-lectures-Notebooks/raw/master/Notebooks/n10017o.fits

[119]:
```
hdulist = fits.open('n10017o.fits')
```

[120]:
```
# The result hdulist is a list of HDU objects.
# In the case of a simple file, there is only one primary HDU so the list␣
 ↪contains only one element
len(hdulist)
```

[120]: 1

[121]:
```
# The information on what the file contains can be obtained by calling the␣
 ↪info() method:
hdulist.info()
# The table said that there is only a primary HDU which contains 2154 X 2048␣
 ↪image with data stored in 2 bytes (16 bits) integers.
```

```
Filename: n10017o.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
```

```
  0  PRIMARY       1 PrimaryHDU      62   (2154, 2048)   int16 (rescales to
uint16)
```

[122]:
```
# As described above, the HDU (header/data unit) contains header and data. The
 ↪header is a dictionary.
# To see what keywords were used in the header one can do:
list(hdulist[0].header.keys())
```

[122]:
```
['SIMPLE',
 'BITPIX',
 'NAXIS',
 'NAXIS1',
 'NAXIS2',
 'EXTEND',
 'COMMENT',
 'COMMENT',
 'BZERO',
 'BSCALE',
 'EXPTIME',
 'DETECTOR',
 'ORIGIN',
 'OBSERVAT',
 'TELESCOP',
 'LATITUDE',
 'LONGITUD',
 'ALTITUD',
 'SECONDAR',
 'TIMEZONE',
 'OBSERVER',
 'OBJECT',
 'INSTRUME',
 'GAINMODE',
 'FILTER',
 'IMGTYPE',
 'EQUINOX',
 'ST',
 'UT',
 'JD',
 'DATE-OBS',
 'CCDSUM',
 'RA',
 'DEC',
 'AH',
 'AIRMASS',
 'TMMIRROR',
 'TSMIRROR',
 'TAIR',
```

```
 'XTEMP',
 'HUMIDITY',
 'ATMOSBAR',
 'WIND',
 'WDATE',
 'DATE',
 'NAMPS',
 'CCDNAMPS',
 'AMPNAME',
 'CREATOR',
 'VERSION',
 'COMMENT',
 'COMMENT',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY',
 'HISTORY']
```

[123]:
```
# and to get the value of a given keyword :
hdulist[0].header['OBJECT']
```

[123]:
```
'107 Psc'
```

[124]:
```
hh = hdulist[0].header
hh?
```

```
Type:           Header
String form:    SIMPLE  =                    T / conforms to FITS standard      ␣
 ↪               BITPIX  =         <…>
Length:         62
File:           ~/anaconda3/lib/python3.7/site-packages/astropy/io/fits/header.py
Docstring:
FITS header class.  This class exposes both a dict-like interface and a
list-like interface to FITS headers.

The header may be indexed by keyword and, like a dict, the associated value
will be returned.  When the header contains cards with duplicate keywords,
only the value of the first card with the given keyword will be returned.
It is also possible to use a 2-tuple as the index in the form (keyword,
n)--this returns the n-th value with that keyword, in the case where there
are duplicate keywords.
```

```
For example::

    >>> header['NAXIS']
    0
    >>> header[('FOO', 1)]  # Return the value of the second FOO keyword
    'foo'

The header may also be indexed by card number::

    >>> header[0]  # Return the value of the first card in the header
    'T'

Commentary keywords such as HISTORY and COMMENT are special cases: When
indexing the Header object with either 'HISTORY' or 'COMMENT' a list of all
the HISTORY/COMMENT values is returned::

    >>> header['HISTORY']
    This is the first history entry in this header.
    This is the second history entry in this header.
    …

See the Astropy documentation for more details on working with headers.
Init docstring:
Construct a `Header` from an iterable and/or text file.

Parameters
----------
cards : A list of `Card` objects, optional
    The cards to initialize the header with. Also allowed are other
    `Header` (or `dict`-like) objects.

    .. versionchanged:: 1.2
        Allowed ``cards`` to be a `dict`-like object.

copy : bool, optional

    If ``True`` copies the ``cards`` if they were another `Header`
    instance.
    Default is ``False``.

    .. versionadded:: 1.3
```

[125]:
```
hdulist[0].header
```

```
[125]: SIMPLE  =                    T / conforms to FITS standard
       BITPIX  =                   16 / array data type
       NAXIS   =                    2 / number of array dimensions
       NAXIS1  =                 2154 / length of data axis 1
       NAXIS2  =                 2048 / length of data axis 2
       EXTEND  =                    T
       COMMENT FITS (Flexible Image Transport System) format is defined in 'Astronomy
       COMMENT and Astrophysics', volume 376, page 359; bibcode: 2001A&A…376..359H
       BZERO   =                32768 / BZERO
       BSCALE  =                    1 / BSCALE
       EXPTIME =                600.0 / Integration Time, sec.
       DETECTOR= 'e2vm2 E2V-4240'     / CCD Type
       ORIGIN  = 'UNAM    '           / OAN SPM, IA-UNAM
       OBSERVAT= 'SPM     '           / Observatory
       TELESCOP= '2.12m   '           / Telescope
       LATITUDE= '+31:02:39'          / Latitude
       LONGITUD= '115:27:49'          / Longitud
       ALTITUD =                 2800 / altitud
       SECONDAR=                   -1 / F/ Secondary type
       TIMEZONE=                    8 / Time Zone
       OBSERVER= 'Leonid  '           / Observer's Name
       OBJECT  = '107 Psc '           / Object
       INSTRUME= 'Echelle '           / Instrument
       GAINMODE=                    1 / Gain factor in the CCD
       FILTER  = 'None    '           / Filter
       IMGTYPE = 'object  '           / Image Type
       EQUINOX =               2011.7 / Equinox
       ST      = '01:25:41.3'         / Sideral Time
       UT      = '10:34:27'           / Universal Time
       JD      =            2455803.5 / Julian Date
       DATE-OBS= '2011-08-30'         / Observation Date UTM
       CCDSUM  = '1 1     '           / Binning [ Cols:Rows ]
       RA      = ' 01:43:10.8'        / Right Ascension
       DEC     = ' 20''19''43.0'      / Declination
       AH      = ' -00:17:29.1'       / Hour Angle
       AIRMASS =                 1.02 / Airmass
       TMMIRROR=                    0 / Primary Mirror Temperature (celsius degree)
       TSMIRROR=                    0 / Secundary Mirror Temperature (celsius degree)
       TAIR    =                    0 / Internal Telescope Air Temperature (celsius deg
       XTEMP   =                 14.7 / Exterior Temperature (celsius degree)
       HUMIDITY=                 46.0 / % external Humidity
       ATMOSBAR=                731.9 / Atmosferic Presure in mb
       WIND    = 'S at 30.6 km/h'     / Wind Direction
       WDATE   = '10:34:10, 08/30/11' / Weather Acquisition Date (Local time)
       DATE    = '2011-08-30T10:34:29' / file creation date (YYYY-MM-DDThh:mm:ss UT)
       NAMPS   =                    1 / Number of Amplifiers
       CCDNAMPS=                    1 / Number of amplifiers used
```

```
AMPNAME = '1 Channel'          / Amplifier name
CREATOR = 'Python Oan ccds'    / Name of the software task that created the file
VERSION = '4.12D   '           / Application Software Version
COMMENT Visit our weather site http://www.astrossp.unam.mx/weather15
COMMENT for complete meteorological data of your observation night
HISTORY bin2fits V1.0
HISTORY Programmer: Enrique Colorado [ colorado@astrosen.unam.mx ]
HISTORY Observatorio Astronomico Nacional -UNAM
HISTORY V1.00 By Arturo Nunez and Colorado >Ported to Python using pyfits
HISTORY V0.50 By E. Colorado >Added interior mirrors temperatures
HISTORY V0.49 By E. Colorado >Added BIASSEC parameter
HISTORY V0.48 By E. Colorado >Aditional info for autofocus calculations
HISTORY V0.4 By E. Colorado >Now we include timezone, and remove lat. sign
HISTORY V0.3 By E. Colorado >Now we include weather data
HISTORY V0.2 By E. Colorado >General OAN Working Release
```

[126]:
```
# The header can be printed as it appears in the file by
print(hdulist[0].header.cards)
```

```
('SIMPLE', True, 'conforms to FITS standard')
('BITPIX', 16, 'array data type')
('NAXIS', 2, 'number of array dimensions')
('NAXIS1', 2154, 'length of data axis 1')
('NAXIS2', 2048, 'length of data axis 2')
('EXTEND', True, '')
('COMMENT', "FITS (Flexible Image Transport System) format is defined in
'Astronomy", '')
('COMMENT', "and Astrophysics', volume 376, page 359; bibcode:
2001A&A…376..359H", '')
('BZERO', 32768, 'BZERO')
('BSCALE', 1, 'BSCALE')
('EXPTIME', 600.0, 'Integration Time, sec.')
('DETECTOR', 'e2vm2 E2V-4240', 'CCD Type')
('ORIGIN', 'UNAM', 'OAN SPM, IA-UNAM')
('OBSERVAT', 'SPM', 'Observatory')
('TELESCOP', '2.12m', 'Telescope')
('LATITUDE', '+31:02:39', 'Latitude')
('LONGITUD', '115:27:49', 'Longitud')
('ALTITUD', 2800, 'altitud')
('SECONDAR', -1, 'F/ Secondary type')
('TIMEZONE', 8, 'Time Zone')
('OBSERVER', 'Leonid', "Observer's Name")
('OBJECT', '107 Psc', 'Object')
('INSTRUME', 'Echelle', 'Instrument')
('GAINMODE', 1, 'Gain factor in the CCD')
('FILTER', 'None', 'Filter')
('IMGTYPE', 'object', 'Image Type')
```

```
('EQUINOX', 2011.7, 'Equinox')
('ST', '01:25:41.3', 'Sideral Time')
('UT', '10:34:27', 'Universal Time')
('JD', 2455803.5, 'Julian Date')
('DATE-OBS', '2011-08-30', 'Observation Date UTM')
('CCDSUM', '1 1', 'Binning [ Cols:Rows ]')
('RA', ' 01:43:10.8', 'Right Ascension')
('DEC', " 20'19'43.0", 'Declination')
('AH', ' -00:17:29.1', 'Hour Angle')
('AIRMASS', 1.02, 'Airmass')
('TMMIRROR', 0, 'Primary Mirror Temperature (celsius degree)')
('TSMIRROR', 0, 'Secundary Mirror Temperature (celsius degree)')
('TAIR', 0, 'Internal Telescope Air Temperature (celsius deg')
('XTEMP', 14.7, 'Exterior Temperature (celsius degree)')
('HUMIDITY', 46.0, '% external Humidity')
('ATMOSBAR', 731.9, 'Atmosferic Presure in mb')
('WIND', 'S at 30.6 km/h', 'Wind Direction')
('WDATE', '10:34:10, 08/30/11', 'Weather Acquisition Date (Local time)')
('DATE', '2011-08-30T10:34:29', 'file creation date (YYYY-MM-DDThh:mm:ss UT)')
('NAMPS', 1, 'Number of Amplifiers')
('CCDNAMPS', 1, 'Number of amplifiers used')
('AMPNAME', '1 Channel', 'Amplifier name')
('CREATOR', 'Python Oan ccds', 'Name of the software task that created the
file')
('VERSION', '4.12D', 'Application Software Version')
('COMMENT', 'Visit our weather site http://www.astrossp.unam.mx/weather15', '')
('COMMENT', 'for complete meteorological data of your observation night', '')
('HISTORY', 'bin2fits V1.0', '')
('HISTORY', 'Programmer: Enrique Colorado [ colorado@astrosen.unam.mx ]', '')
('HISTORY', 'Observatorio Astronomico Nacional -UNAM', '')
('HISTORY', 'V1.00 By Arturo Nunez and Colorado >Ported to Python using pyfits',
'')
('HISTORY', 'V0.50 By E. Colorado >Added interior mirrors temperatures', '')
('HISTORY', 'V0.49 By E. Colorado >Added BIASSEC parameter', '')
('HISTORY', 'V0.48 By E. Colorado >Aditional info for autofocus calculations',
'')
('HISTORY', 'V0.4 By E. Colorado >Now we include timezone, and remove lat.
sign', '')
('HISTORY', 'V0.3 By E. Colorado >Now we include weather data', '')
('HISTORY', 'V0.2 By E. Colorado >General OAN Working Release', '')
```

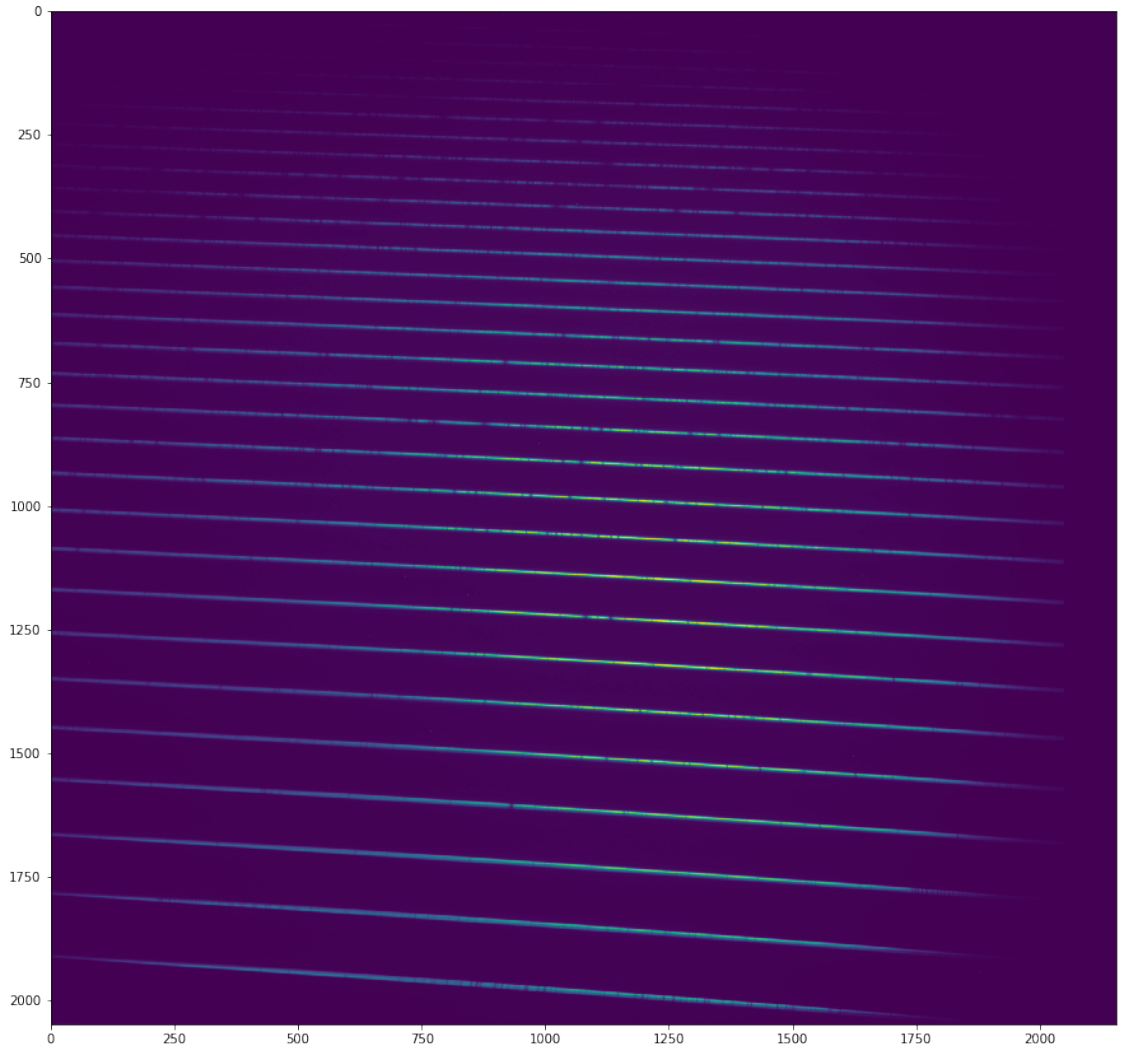
[127]:
```
# The data in the file are accessible with
data = hdulist[0].data
```

[129]:
```
data.shape
```

[129]:
```
(2048, 2154)
```

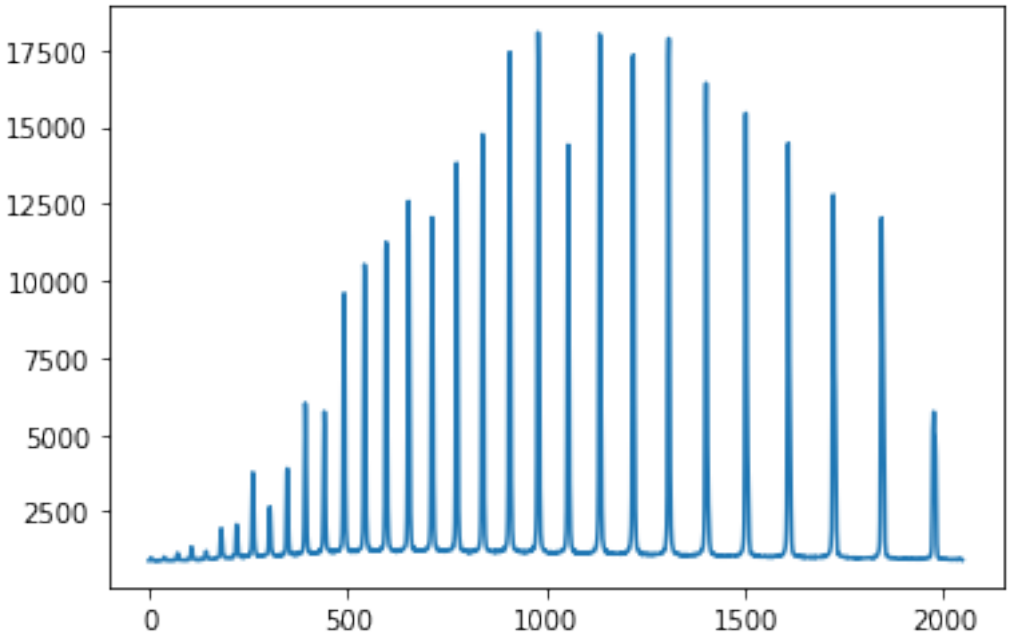
[128]:
```
# and can be seen with [we need to import matplotlib.pyplot as plt before␣
 ↪running this]:
%matplotlib inline
import matplotlib.pyplot as plt
f, ax = plt.subplots(figsize=(15,15))
ax.imshow(data)
```

[128]: <matplotlib.image.AxesImage at 0x7fd1d2b45950>



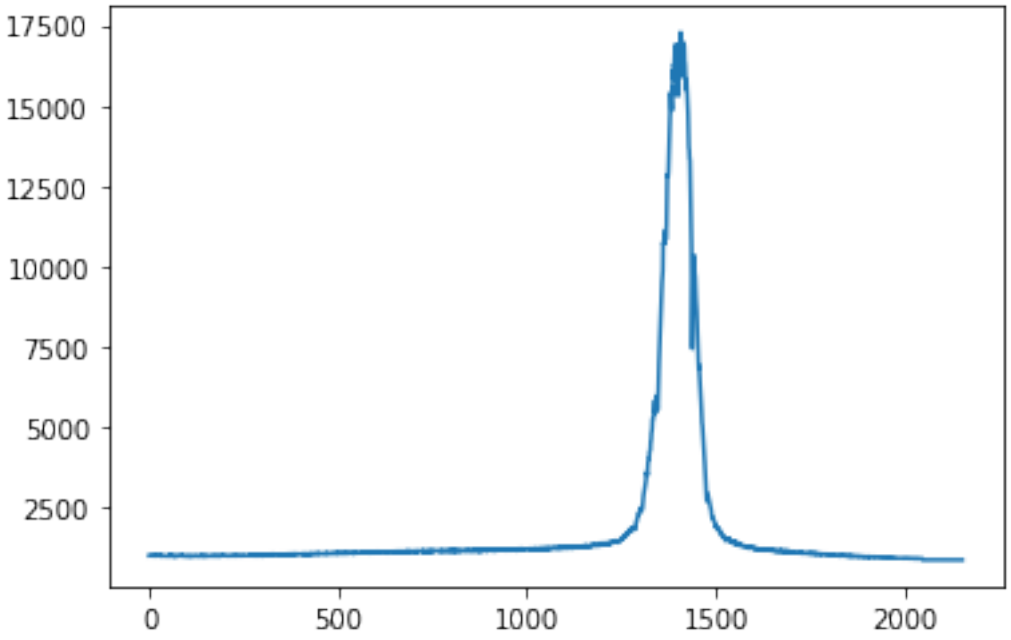
[130]:
```
# A column from the data can be plotted with
plt.plot(data[:,1000])
# where I am plotting the column number 1000.
```

[130]: [<matplotlib.lines.Line2D at 0x7fd1d10e4a50>]



[131]:
```
# In the same way a line from the data is plotted with:
plt.plot(data[1000,:])
```

[131]: [<matplotlib.lines.Line2D at 0x7fd1d1a812d0>]



[132]:
```
# For this example I'll use a spectrum obtain with the high dispersion camera␣
↪on board of IUE.
# The file is opened as usual:
hdulist = fits.open('swp04345.mxhi')
```

The file is there: https://github.com/Morisset/Python-lectures-Notebooks/raw/master/Notebooks/swp04345.mxhi

[133]:
```
#but now hdulist has 2 elements (2 header/data units):
len(hdulist)
```

[133]: 2

[134]:
```
# We can see that the primary header has dimension (), son does not contain any␣
↪data.
# The data are in the extension.
hdulist.info()
```

```
Filename: swp04345.mxhi
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     421   ()
  1  MEHI          1 BinTableHDU     61   60R x 17C   [1B, 1I, 1D, 1I, 1D, 1E,
1E, 768E, 768E, 768E, 768I, 768E, 768E, 1I, 1I, 1E, 7E]
```

[136]:
```
# The first header contains the minimal infirmation:
print(hdulist[0].header.cards)
```

```
('SIMPLE', True, 'Standard FITS Format')
('BITPIX', 8, 'Binary data')
('NAXIS', 0, 'Two-dimensional image')
('EXTEND', True, 'Extensions are present')
('TELESCOP', 'IUE', 'International Ultraviolet Explorer')
('DATE', '16/10/96', 'Date file written')
('ORIGIN', 'GSFC', 'Institution generating the file')
('COMMENT', '*', '')
('COMMENT', '* CORE DATA ITEMS - COMMON SET', '')
('COMMENT', '*', '')
('CAMERA', 'SWP', 'Camera')
('IMAGE', 4345, 'Sequential image number')
('DISPERSN', 'HIGH', 'Spectrograph dispersion mode')
('APERTURE', 'LARGE', 'Aperture')
('DISPTYPE', 'HIGH', 'Dispersion processing type')
('READMODE', 'FULL', 'Read mode')
('READGAIN', 'LOW', 'Read gain')
('EXPOGAIN', 'MAXIMUM', 'Exposure gain')
('UVC-VOLT', -5.0, 'UVC voltage')
('ABNNOSTD', 'NO', 'Non-standard image acquisition')
('ABNBADSC', 'NO', 'LWP bad scans')
```

```
('ABNHTRWU', 'NO', 'LWR heater warmup')
('ABNREAD', 'NO', 'Read at other than 20 KB')
('ABNUVC', 'NO', 'Non-standard UVC voltage')
('ABNHISTR', 'NO', 'History replay')
('ABNOTHER', 'NO', 'Other abnormality')
('THDAREAD', 8.8, 'THDA at read of image')
('EQUINOX', 1950.0, 'Epoch of coordinates')
('STATION', 'GSFC', 'Observing station')
('ORBEPOCH', '22/02/79', 'Orbital elements epoch')
('ORBSAXIS', 42170.5, 'Semi-major axis in kilometers')
('ORBECCEN', 0.2381567, 'Eccentricity')
('ORBINCLI', 28.379, 'Inclination in degrees')
('ORBASCEN', 199.759, 'Ascending node in degrees')
('ORBPERIG', 265.437, 'Argument of perigee in degrees')
('ORBANOMA', 337.433, 'Mean anomaly in degrees')
('POSANGLE', 30.32, 'Pos angle of the large aperture (deg)')
('LAMP', 'NONE', 'Lamp')
('PGM-ID', 'LABDS', 'Program identification')
('ABNMINFR', 'NO', 'Bad/missing minor frames')
('CC-PERCN', 82.7, 'Cross-correlation % successful')
('CC-WINDW', 29, 'Cross-correlation window size')
('CC-TEMPL', 23, 'Cross-correlation template size')
('CC-MEDN', 0.431, 'Median cross-correlation coefficient')
('CC-STDEV', 0.145, 'St dev of cross-corr coefficients')
('SHFTMEAN', 0.287, 'Mean shift between image and ITF')
('SHFTMAX', 1.425, 'Maximum shift between image and ITF')
('ITF', 'SWP85R92A', 'ITF identification')
('TILTCORR', 'NO', 'Tilt correction flag')
('MEANRAT', 0.987, 'SI vs LI mean')
('STDEVRAT', 0.9, 'SI vs LI standard deviation')
('COMMENT', 'BY RA:  EXP 1 APER L MAX DN = 165', '')
('COMMENT', 'BY RA:     0 MISSING MINOR FRAMES NOTED ON SCRIPT', '')
('COMMENT', 'BY RA:  EXP 1 TRACKED ON FES', '')
('COMMENT', 'BY RA:  S   PREP USED', '')
('COMMENT', '*', '')
('COMMENT', '* CORE DATA ITEMS - LARGE APERTURE SET', '')
('COMMENT', '*', '')
('LDATEOBS', '23/02/79', 'Observing date')
('LTIMEOBS', '02:49:03', 'Observing time')
('LJD-OBS', 2443927.6174, 'Julian Date start of obs.')
('LEXPTRMD', 'NO-TRAIL', 'Trail mode')
('LEXPMULT', 'NO', 'Multiple exposure mode')
('LEXPSEGM', 'NO', 'Segmented exposure code')
('LEXPTIME', 10799.793, 'Integration time in seconds')
('LTHDASTR', 8.2, 'THDA at start of exposure')
('LTHDAEND', 8.8, 'THDA at end of exposure')
('LRA', 14.425, 'Homogeneous R.A. in degrees')
('LDEC', -72.4333, 'Homogeneous Dec. in degrees')
```

```
('LLAPSTAT', 'OPEN', 'Large aperture status')
('LFES2MD', 'SO', 'FES(2) mode')
('LFES2CN', 363, 'FES(2) counts on target')
('LTARGET', 'HD 5980', 'Object as given by Guest Observer')
('LTARGRA', 14.4417, 'R.A. in degrees (given by GO)')
('LTARGDEC', -72.4347, 'Dec. in degrees (given by GO)')
('LOBJECT', 'HD  5980', 'Homogeneous Object ID')
('LIUECLAS', 11, 'Object class')
('LFOCUS', -1.5, 'Focus')
('LFPM', 0.08, 'Flux particle monitor')
('LGSTAR2X', 307, 'X coordinate of guide star in FES 2')
('LGSTAR2Y', -835, 'Y coordinate of guide star in FES 2')
('LJD-MID', 2443927.67989, 'Julian Date middle of obs.')
('LHELCORR', -0.00228, 'Heliocentric corr to midpoint (days)')
('LDATABKG', 67, 'Estimated mean background level (DNs)')
('LDATACNT', 150, 'Estimated maximum continuum level (DNs)')
('LCNTRAPR', 290.74, 'Predicted center line of spectrum')
('LXTRMODE', 'POINT', 'Extraction mode')
('LXTRCNTR', 290.7, 'Center line of extracted spectrum')
('LRADVELO', 2.67, 'Heliocentric velocity correction in km/s')
('COMMENT', '*', '')
('COMMENT', '* THE IUE VICAR HEADER', '')
('COMMENT', '*', '')
('COMMENT', 'IUE-VICAR HEADER START', '')
('', '                        0001000107680768   1 1 013004345           1  C',
'')
('', '      *    *        *   *   *10,800*      *   *  * * * * * * *     *  2  C',
'')
('', 'SWP4345,HD5980,180M,LGAP,HI,DAY54,1979,LABDS,SITKO                   3  C',
'')
('', '                                                                     4  C',
'')
('', '                        10799 SECS                                   5  C',
'')
('', '                                                                     6  C',
'')
('', '                                                                     7  C',
'')
('', '                                                                     8  C',
'')
('', '                                                                     9  C',
'')
('', '79 54055056* 8   * 215 *OPSTST23*133019 CAMINIT                   * 10  C',
'')
('', '                               *133019   1178 3830 4337 1149     * 11  C',
'')
('', '                               *                                 * 12  C',
'')
```

```
('', '                              *                               * 13  C',
'')
('', '                              *                               * 14  C',
'')
('', '200834 XPREP 2                *202631 SPREP 2                 * 15  C',
'')
('', '015359 EXPOBCM 2 12 0  MAXG NOL *                             * 16  C',
'')
('', '020605 EXPOSURE END TIME      *                               * 17  C',
'')
('', '                              *                               * 18  C',
'')
('', '025850 READPREP 2       NORMAL  *025923 SCAN 2  RDLO SS 1 G3 58  * 19  C',
'')
('', '025925 X 56 Y 72 G1 99 HR 106   *165808 SPREP 3                 * 20  C',
'')
('', '024907 EXPOBCM 3 180 0 MAXG NOL *                             * 21  C',
'')
('', '054910 EXPOSURE END TIME      *                               * 22  C',
'')
('', '                              *                               * 23  C',
'')
('', '054953 REREAD 3 3       NORMAL  *055055 SCAN 3  RDLO SS 1 G3 44  * 24  C',
'')
('', '055056 X 60 Y 76 G1 82 HR 105   *                             * 25  C',
'')
('', '                              *020838 MODE SWH                * 26  C',
'')
('', '023438 APERTURE OP            *                               * 27  C',
'')
('', '                              *                               * 28  C',
'')
('', '                              *                               * 29  C',
'')
('', '                              *024716TARGET IN SWLA            * 30  C',
'')
('', '015155TARGET FROM SWLA        *                               * 31  C',
'')
('', '                              *                               * 32  C',
'')
('', '                                                                33  C',
'')
('', '                                                                34  C',
'')
('', '                                                                35  C',
'')
('', 'LABDS* *  *SAVAGE          *   *   * *        * *0* *            36  C',
'')
```

```
('', '005745*-722603*   *OB* *11.8*+0.04*         *    *       *   *     37  C',
'')
('', '                                                                      38  C',
'')
('', '                                                                      39  C',
'')
('', '                                                                      40  C',
'')
('', '                                                                      41  C',
'')
('', '                                                                      42  C',
'')
('', '                                                                      43  C',
'')
('', '                                                                      44  C',
'')
('', '                                                                      45  C',
'')
('', '                                                                      46  C',
'')
('', '                                                                      47  C',
'')
('', '                                                                      48  C',
'')
('', '                                                                      49  C',
'')
('', '                                                                      50  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000 51  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000004040 51  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000 52  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000004040 52  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000 53  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000004040 53  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000 54  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000004040 54  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000 55  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000004040 55  C',
'')
```

```
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 56  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 56  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 57  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 57  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 58  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 58  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 59  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 59  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 60  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 60  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 61  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 61  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 62  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 62  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 63  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 63  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 64  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 64  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 65  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 65  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 66  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 66  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000000000 67  C',
'')
('', '00000000000000000000000000000000000000000000000000000000000000000000000000004040 67  C',
'')
```

```
('', '0000000000000000000000000000000000000000000000000000000000000000 68  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 68  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 69  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 69  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 70  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 70  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 71  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 71  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 72  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 72  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 73  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 73  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 74  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 74  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000000000 75  C',
'')
('', '0000000000000000000000000000000000000000000000000000000000004040 75  C',
'')
('', '0231F4FACC000000A6737B00C50E003AC80D31444B2E6827253D4540321E314B62 76  C',
'')
('', '27003D87F306060B6E70005E8085320D2C2A13282C1A3211151200A5ACAB000000 76  C',
'')
('', 'AC0400FF000436B9A5C100AF52B312341E2B33352C16182D2E1A1F181C202D2B34 77  C',
'')
('', '70717A247D247C227B1525A01E9320941F951E96219301B8E10000E02CE0F90C40 77  C',
'')
('', '7C007C0083007A7B7B7B7C7C3A8A0082453C3D314A7A93944EC989987D7D824300 78  C',
'')
('', '003A30333539302F2E3234322B2B000006C1F656828200010082824040404040 78  C',
'')
('', 'BBAB06C2F2558381000100808O3B3535333A343030353735323286B784B941B76D 79  C',
'')
('', 'B96BB72DB978B778B931B77AB97BB744B940404040404040404040404040404040 79  C',
'')
```

```
('', 'D0000080000000000000FF4040404040404040404040404040404040404040404040 80  C',
'')
('', '4040404040404040404040404040404040404040404040404040404040404040404040 80  C',
'')
('', '0FFBFFFF000000000203FE025046436BFB30000000000000D00000000000000080 81  C',
'')
('', '00000000000000FF000000880000000C0000000C0000000000000000000000004040 81  C',
'')
('', '4F25600000000000FF000000FF000000000000000000000000000000000000000000 82  C',
'')
('', '0000007F000000FF00000000000000FF000000FF0000000000000000000000004040 82  C',
'')
('', '                                                                      83  C',
'')
('', '                                                                      84  C',
'')
('', '                                                                      85  C',
'')
('', '03060000001C00000021879BE02CDFE1000004000016909E4ABAAB06C1F2558381 86  C',
'')
('', '00010080807BC978C931C900006141415100004040400040F0D0E404040404040 86  C',
'')
('', '030D0000001C0000002186A3E02CE0E1000004000016909E4ABAAB06C1F2558381 87  C',
'')
('', '00010080807AC977C931C900006141415100004040400401F0D0E404040404040 87  C',
'')
('', '030E0000001C00FFFFEF86A8E02CDFE1000004000016809E4ABAAB06C1F2558381 88  C',
'')
('', '00010080807BC978C931C900006141514200004040400040290D0E404040404040 88  C',
'')
('', '01360000001E00FFFFF68A9DE02CE0E1000004000016809E52000006C1F2558381 89  C',
'')
('', '0001007F80797F777A3179000061404141000040404000400B0C404040404040 89  C',
'')
('', '0205000000190000000338AA0E02CE0E1000004000016809E52000006C2F2558381 90  C',
'')
('', '0001007F807A7F787A3179000061404141000040404000400B0C404040404040 90  C',
'')
('', '023B00FFFFFE00FFFFF986A5E02DE0E1000004000010809E4ABAAB06C1F2558381 91  C',
'')
('', '000100807F7AC978CA31C900006141414A000040404000401900D0E404040404040 91  C',
'')
('', '023B00FFFFF40000000C86A7E02DE0E1000004000010809E4ABBAB06C1F2558381 92  C',
'')
('', '000100807F7AC978CA31C900006141514A000040404000402600D0E404040404040 92  C',
'')
('', '02110000002300FFFFED89A5E02CE0E1000004000006909E4C3B3535333A343030 93  C',
'')
```

```
('', '35373532327AC97BCA44C900006151414000004040400040 1D0D0E404040404040 93  C',
'')
('', '021A0000002300FFFFED89ACE02CE0E1000004000006909E4C3B3435333A34302F 94  C',
'')
('', '35363532327AC97BC944C900006151414000004040400040 160D0E404040404040 94  C',
'')
('', '021B0000002300FFFFED88A8DF2DE0E1000004000004809E4C3B3837353C353232 95  C',
'')
('', '36383735357AC97BC944C900007142414000004040400040 1F0D0E404040404040 95  C',
'')
('', '02310000002300FFFFED869CE02CE0E1000004000002809E4A3B3535333A343030 96  C',
'')
('', '35373532327AB97BB744B900006141414000004040400040 0C0B0C404040404040 96  C',
'')
('', '053100FFFFF40000000C87A3E02CE0E1000006000016809E4A3A3335333A342F2F 97  C',
'')
('', '35373531327AC77AC748C700006140414000004040400040 040D0E404040404040 97  C',
'')
('', '053300FFFFF40000000C87A9E02DE0E2000005000016809E4A3C3736343C363232 98  C',
'')
('', '36383735347AC97AC948C90000614A414000004040400040 060D0E404040404040 98  C',
'')
('', '053300FFFFF40000000C87A1E02CE0E100000E000016809E4A3C3837353C373332 99  C',
'')
('', '37383735357ACA7AC948C90000714A414000004040400040 250D0E404040404040 99  C',
'')
('', '4040404040404040404040404040404040404040404040404040404040404040100  C',
'')
('', '4040404040404040404040404040404040404040404040404040404040404040100  C',
'')
('', '****** SCHEME NAME: F3HLAC. ****                                      C',
'')
('', "*ARCHIVE   12:00Z MAR 18,'80                                        HL",
'')
('COMMENT', 'IUE-VICAR HEADER END', '')
('HISTORY', 'IUE-LOG STARTED                          16-OCT-1996 00:18:27',
'')
('HISTORY', 'PROCESSING SYSTEM: NEWSIPS VERSION 3.1_A', '')
('HISTORY', 'OPEN VMS VERSION', '')
('HISTORY', 'SWP04345', '')
('HISTORY', 'PROCESSED AT GODDARD SPACE FLIGHT CENTER', '')
('HISTORY', '************************************************************',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START RAW_SCREEN                         16-OCT-1996 00:18:37',
'')
('HISTORY', '  39 BRIGHT SPOTS DETECTED', '')
```

```
('HISTORY', '   0 MISSING MINOR FRAMES DETECTED', '')
('HISTORY', 'LARGE APERTURE SPECTRUM WILL BE EXTRACTED AS', '')
('HISTORY', '          POINT SOURCE', '')
('HISTORY', 'LARGE APERTURE CONTINUUM DN LEVEL = 150', '')
('HISTORY', 'BACKGROUND DN LEVEL =  67', '')
('HISTORY', 'ORDER REGISTRATION', '')
('HISTORY', '  GLOBAL OFFSET  0.78 PIXELS RELATIVE TO FIDUCIAL: SWP 13589', '')
('HISTORY', '  RELATIVE ORDER LOCATIONS DETERMINED FROM EMPIRICAL POSITIONS',
'')
('HISTORY', 'END   RAW_SCREEN                              16-OCT-1996 00:19:03',
'')
('HISTORY', '**************************************************************',
'')
('HISTORY', 'START TTDC                                    16-OCT-1996 00:19:07',
'')
('HISTORY', 'TEMPERATURE USED FOR CORRECTING DISPERSION CONSTANTS =  8.80', '')
('HISTORY', 'DATE OF OBSERVATION USED FOR CORRECTING', '')
('HISTORY', '          DISPERSION CONSTANTS = 23/ 2/79  02:49:03', '')
('HISTORY', 'ORDER  66 ZERO-POINT CORRECTION =   0.115 ANGSTROMS', '')
('HISTORY', 'ORDER  67 ZERO-POINT CORRECTION =   0.119 ANGSTROMS', '')
('HISTORY', 'ORDER  68 ZERO-POINT CORRECTION =   0.115 ANGSTROMS', '')
('HISTORY', 'ORDER  69 ZERO-POINT CORRECTION =   0.115 ANGSTROMS', '')
('HISTORY', 'ORDER  70 ZERO-POINT CORRECTION =   0.110 ANGSTROMS', '')
('HISTORY', 'ORDER  71 ZERO-POINT CORRECTION =   0.110 ANGSTROMS', '')
('HISTORY', 'ORDER  72 ZERO-POINT CORRECTION =   0.110 ANGSTROMS', '')
('HISTORY', 'ORDER  73 ZERO-POINT CORRECTION =   0.111 ANGSTROMS', '')
('HISTORY', 'ORDER  74 ZERO-POINT CORRECTION =   0.112 ANGSTROMS', '')
('HISTORY', 'ORDER  75 ZERO-POINT CORRECTION =   0.109 ANGSTROMS', '')
('HISTORY', 'ORDER  76 ZERO-POINT CORRECTION =   0.108 ANGSTROMS', '')
('HISTORY', 'ORDER  77 ZERO-POINT CORRECTION =   0.105 ANGSTROMS', '')
('HISTORY', 'ORDER  78 ZERO-POINT CORRECTION =   0.103 ANGSTROMS', '')
('HISTORY', 'ORDER  79 ZERO-POINT CORRECTION =   0.106 ANGSTROMS', '')
('HISTORY', 'ORDER  80 ZERO-POINT CORRECTION =   0.103 ANGSTROMS', '')
('HISTORY', 'ORDER  81 ZERO-POINT CORRECTION =   0.101 ANGSTROMS', '')
('HISTORY', 'ORDER  82 ZERO-POINT CORRECTION =   0.101 ANGSTROMS', '')
('HISTORY', 'ORDER  83 ZERO-POINT CORRECTION =   0.101 ANGSTROMS', '')
('HISTORY', 'ORDER  84 ZERO-POINT CORRECTION =   0.101 ANGSTROMS', '')
('HISTORY', 'ORDER  85 ZERO-POINT CORRECTION =   0.099 ANGSTROMS', '')
('HISTORY', 'ORDER  86 ZERO-POINT CORRECTION =   0.100 ANGSTROMS', '')
('HISTORY', 'ORDER  87 ZERO-POINT CORRECTION =   0.091 ANGSTROMS', '')
('HISTORY', 'ORDER  88 ZERO-POINT CORRECTION =   0.096 ANGSTROMS', '')
('HISTORY', 'ORDER  89 ZERO-POINT CORRECTION =   0.095 ANGSTROMS', '')
('HISTORY', 'ORDER  90 ZERO-POINT CORRECTION =   0.096 ANGSTROMS', '')
('HISTORY', 'ORDER  91 ZERO-POINT CORRECTION =   0.094 ANGSTROMS', '')
('HISTORY', 'ORDER  92 ZERO-POINT CORRECTION =   0.092 ANGSTROMS', '')
('HISTORY', 'ORDER  93 ZERO-POINT CORRECTION =   0.089 ANGSTROMS', '')
('HISTORY', 'ORDER  94 ZERO-POINT CORRECTION =   0.093 ANGSTROMS', '')
('HISTORY', 'ORDER  95 ZERO-POINT CORRECTION =   0.096 ANGSTROMS', '')
```

```
('HISTORY', 'ORDER  96 ZERO-POINT CORRECTION =   0.089 ANGSTROMS', '')
('HISTORY', 'ORDER  97 ZERO-POINT CORRECTION =   0.089 ANGSTROMS', '')
('HISTORY', 'ORDER  98 ZERO-POINT CORRECTION =   0.087 ANGSTROMS', '')
('HISTORY', 'ORDER  99 ZERO-POINT CORRECTION =   0.083 ANGSTROMS', '')
('HISTORY', 'ORDER 100 ZERO-POINT CORRECTION =   0.087 ANGSTROMS', '')
('HISTORY', 'ORDER 101 ZERO-POINT CORRECTION =   0.083 ANGSTROMS', '')
('HISTORY', 'ORDER 102 ZERO-POINT CORRECTION =   0.076 ANGSTROMS', '')
('HISTORY', 'ORDER 103 ZERO-POINT CORRECTION =   0.074 ANGSTROMS', '')
('HISTORY', 'ORDER 104 ZERO-POINT CORRECTION =   0.072 ANGSTROMS', '')
('HISTORY', 'ORDER 105 ZERO-POINT CORRECTION =   0.071 ANGSTROMS', '')
('HISTORY', 'ORDER 106 ZERO-POINT CORRECTION =   0.069 ANGSTROMS', '')
('HISTORY', 'ORDER 107 ZERO-POINT CORRECTION =   0.067 ANGSTROMS', '')
('HISTORY', 'ORDER 108 ZERO-POINT CORRECTION =   0.065 ANGSTROMS', '')
('HISTORY', 'ORDER 109 ZERO-POINT CORRECTION =   0.063 ANGSTROMS', '')
('HISTORY', 'ORDER 110 ZERO-POINT CORRECTION =   0.061 ANGSTROMS', '')
('HISTORY', 'ORDER 111 ZERO-POINT CORRECTION =   0.059 ANGSTROMS', '')
('HISTORY', 'ORDER 112 ZERO-POINT CORRECTION =   0.057 ANGSTROMS', '')
('HISTORY', 'ORDER 113 ZERO-POINT CORRECTION =   0.055 ANGSTROMS', '')
('HISTORY', 'ORDER 114 ZERO-POINT CORRECTION =   0.052 ANGSTROMS', '')
('HISTORY', 'ORDER 115 ZERO-POINT CORRECTION =   0.050 ANGSTROMS', '')
('HISTORY', 'ORDER 116 ZERO-POINT CORRECTION =   0.048 ANGSTROMS', '')
('HISTORY', 'ORDER 117 ZERO-POINT CORRECTION =   0.045 ANGSTROMS', '')
('HISTORY', 'ORDER 118 ZERO-POINT CORRECTION =   0.043 ANGSTROMS', '')
('HISTORY', 'ORDER 119 ZERO-POINT CORRECTION =   0.040 ANGSTROMS', '')
('HISTORY', 'ORDER 120 ZERO-POINT CORRECTION =   0.038 ANGSTROMS', '')
('HISTORY', 'ORDER 121 ZERO-POINT CORRECTION =   0.035 ANGSTROMS', '')
('HISTORY', 'ORDER 122 ZERO-POINT CORRECTION =   0.032 ANGSTROMS', '')
('HISTORY', 'ORDER 123 ZERO-POINT CORRECTION =   0.030 ANGSTROMS', '')
('HISTORY', 'ORDER 124 ZERO-POINT CORRECTION =   0.027 ANGSTROMS', '')
('HISTORY', 'ORDER 125 ZERO-POINT CORRECTION =   0.024 ANGSTROMS', '')
('HISTORY', '', '')
('HISTORY', 'SPACECRAFT VELOCITY:', '')
('HISTORY', 'X= -2.86     Y= -2.42     Z=  0.71', '')
('HISTORY', 'EARTH VELOCITY:', '')
('HISTORY', 'X=-13.59     Y=-24.64     Z=-10.69', '')
('HISTORY', 'NET CORRECTION VECTOR TO HELIOCENTRIC VELOCITY:', '')
('HISTORY', 'X=-16.46     Y=-27.06     Z= -9.98', '')
('HISTORY', 'HELIOCENTRIC VELOCITY CORRECTION:  +2.67  KM/S', '')
('HISTORY', 'END   TTDC                                  16-OCT-1996 00:19:10',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START CROSS-CORR                            16-OCT-1996 00:19:14',
'')
('HISTORY', 'WINDOW SIZE USED:    29 X  29 PIXELS', '')
('HISTORY', 'TEMPLATE SIZE USED:  23 X  23 PIXELS', '')
('HISTORY', 'ITF USED:  SWP85R92A', '')
('HISTORY', '  82.7 PERCENT SUCCESSFUL CORRELATIONS (422 OUT OF 510)', '')
```

```
('HISTORY', 'MEDIAN CORRELATION COEFFICIENT: 0.431', '')
('HISTORY', 'STANDARD DEVIATION OF CORRELATION COEFFICIENT: 0.145', '')
('HISTORY', 'MEAN SHIFT IN PIXELS: 0.287', '')
('HISTORY', 'MAXIMUM SHIFT IN PIXELS: 1.425', '')
('HISTORY', 'NUMBER OF SUCCESSFUL SHIFTS FILTERED AS UNRELIABLE IN', '')
('HISTORY', '    POST-FILTER ROUTINE:  11', '')
('HISTORY', 'END   CROSS-CORR                              16-OCT-1996 00:19:59',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START PHOTOM                                  16-OCT-1996 00:20:08',
'')
('HISTORY', 'ITF USED: SWP85R92A', '')
('HISTORY', 'MEAN TEMPERATURE OF ITF:  9.3 C', '')
('HISTORY', 'ITF UVC=-5.0 KV; UVFLOOD WAVELENGTH = 2536 A; ITF SEC =-6.1 KV',
'')
('HISTORY', 'ITF CONSTRUCTION: RAW SPACE, FOURIER FILTERED; JAN92', '')
('HISTORY', 'END   PHOTOM                                  16-OCT-1996 00:21:43',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START GEOM                                    16-OCT-1996 00:21:45',
'')
('HISTORY', ' EARLY EPOCH ORDER SPATIAL DEVIATION CORRECTION APPLIED', '')
('HISTORY', ' DE-SPLAYING ANGLE OF -0.29E-04 RADIANS', '')
('HISTORY', ' PREDICTED CENTER LINE OF ORDER 100 - LINE 290.74', '')
('HISTORY', 'END   GEOM                                    16-OCT-1996 00:30:51',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START COSMIC_RAY                              16-OCT-1996 00:31:00',
'')
('HISTORY', ' MEAN FN VALUE OF INTERORDER BACKGROUND =   46.907', '')
('HISTORY', ' 11968 PIXELS GREATER THAN 1.500   SIGMA FLAGGED IN', '')
('HISTORY', ' COSMIC_RAY IMAGE', '')
('HISTORY', 'END   COSMIC_RAY                              16-OCT-1996 00:31:13',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START BCKGRD                                  16-OCT-1996 00:31:15',
'')
('HISTORY', ' INTERORDER POINTS IDENTIFIED FOR POINT    SOURCE', '')
('HISTORY', ' GLOBAL BACKGROUND DETERMINATION SUCCESSFUL', '')
('HISTORY', '    NORMAL GRID INTERPOLATION', '')
('HISTORY', 'END   BCKGRD                                  16-OCT-1996 00:32:11',
'')
('HISTORY', '************************************************************',
'')
```

```
('HISTORY', 'START EXTRACT                                16-OCT-1996 00:32:14',
'')
('HISTORY', 'BOXCAR EXTRACTION', '')
('HISTORY', 'NOISE MODEL USED: SWP VERSION 1.0', '')
('HISTORY', '', '')
('HISTORY', '**********************LARGE APERTURE DATA**********************',
'')
('HISTORY', '', '')
('HISTORY', 'MEAN SLIT HEIGHT FOR LARGE APERTURE POINT SOURCE USED FOR EACH
ORDER', '')
('HISTORY', ' ORDER 100 FOUND AT LINE 290.68', '')
('HISTORY', '*** WARNING: ORDER 113 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 114 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 119 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 120 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 121 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 122 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 123 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 124 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '*** WARNING: ORDER 125 EXPLICIT CENTROID DETERMINATION INVALID.',
'')
('HISTORY', '             FIDUCIAL CENTROID USED.', '')
('HISTORY', '', '')
('HISTORY', ' SWP RIPPLE CORRECTION VERSION 2.0 APPLIED.', '')
('HISTORY', ' ABSOLUTE FLUX CALIBRATION DERIVED FROM LOW DISPERSION FLUX', '')
('HISTORY', ' CALIBRATION.', '')
('HISTORY', ' ABSOLUTE FLUX CALIBRATION SWP VERSION 1.2 APPLIED USING:', '')
('HISTORY', '      MODE = LARGE APERTURE POINT SOURCE', '')
('HISTORY', '      CALIBRATION EPOCH = 1985.00', '')
('HISTORY', '      CAMERA RISE TIME = 0.130 SECONDS', '')
('HISTORY', '      EFFECTIVE EXPOSURE TIME = 10799.793 SECONDS', '')
('HISTORY', ' TEMPERATURE-DEPENDENT SENSITIVITY CORRECTION APPLIED USING:', '')
```

```
('HISTORY', '      THDA OF IMAGE =  8.80', '')
('HISTORY', '      REFERENCE THDA =  9.40', '')
('HISTORY', '      TEMPERATURE COEFFICIENT = -0.0046', '')
('HISTORY', '      TEMPERATURE CORRECTION FACTOR =   0.997', '')
('HISTORY', ' SENSITIVITY DEGRADATION CORRECTION SWP VERSION 1.0 APPLIED
USING:', '')
('HISTORY', '      MODE = LARGE APERTURE POINT SOURCE', '')
('HISTORY', '      CALIBRATION EPOCH = 1985.00', '')
('HISTORY', '      OBSERVATION DATE = 1979.148', '')
('HISTORY', 'END   EXTRACT                                16-OCT-1996 00:32:36',
'')
('HISTORY', '************************************************************',
'')
('HISTORY', 'START FITSCOPY                               16-OCT-1996 00:32:39',
'')
```

[137]:
```
# The number of axis is 0 which means there is no data block in the primary HDU.
 ↪
# The header of the second HDU begins with the keyword XTENSION and with the␣
 ↪specification of the data
print(hdulist[1].header.cards[:5])
```

```
('XTENSION', 'BINTABLE', 'Binary table extension')
('BITPIX', 8, 'Binary data')
('NAXIS', 2, 'Two-dimensional table array')
('NAXIS1', 16961, 'Width of row in bytes')
('NAXIS2', 60, 'Number of orders')
```

[138]:
```
# To progress further we need to know what is in the table.
# As usual, the columns have names and type of the stored data.
# These information can be obtained using the column attribute of hdulist:
cols = hdulist[1].columns
```

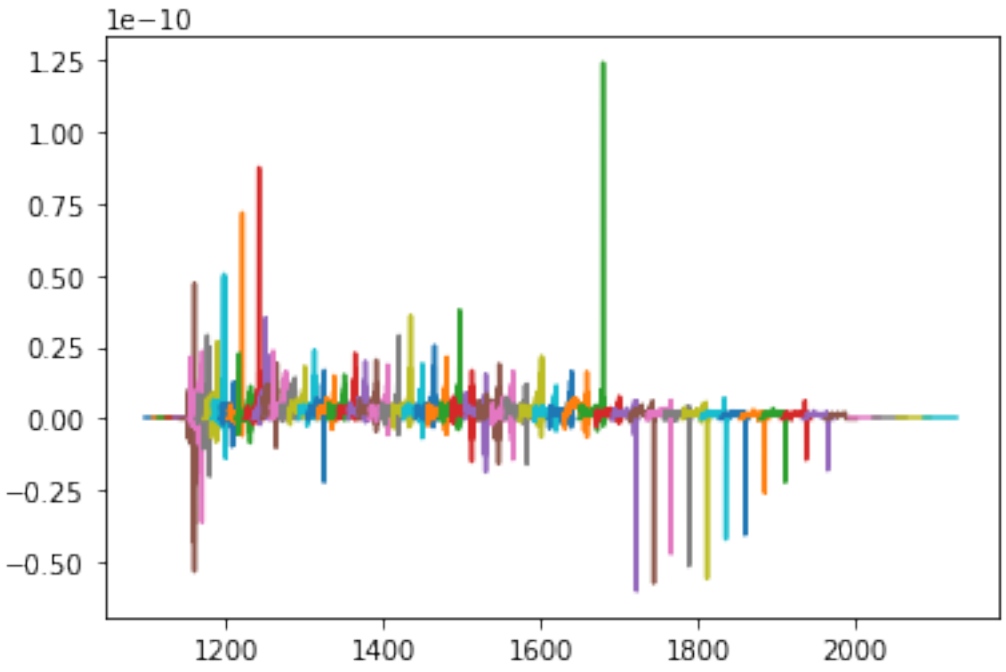
[139]:
```
# the cols.info returns the names of the columns and the information of their␣
 ↪format and units.
cols.info
```

[139]:
```
<bound method ColDefs.info of ColDefs(
    name = 'ORDER'; format = '1B'
    name = 'NPOINTS'; format = '1I'
    name = 'WAVELENGTH'; format = '1D'; unit = 'ANGSTROM'
    name = 'STARTPIX'; format = '1I'; unit = 'PIXEL'
    name = 'DELTAW'; format = '1D'; unit = 'ANGSTROM'
    name = 'SLIT HEIGHT'; format = '1E'; unit = 'PIXEL'
    name = 'LINE_FOUND'; format = '1E'; unit = 'PIXEL'
    name = 'NET'; format = '768E'; unit = 'FN'
    name = 'BACKGROUND'; format = '768E'; unit = 'FN'
```

```
    name = 'NOISE'; format = '768E'; unit = 'FN'
    name = 'QUALITY'; format = '768I'
    name = 'RIPPLE'; format = '768E'; unit = 'FN'
    name = 'ABS_CAL'; format = '768E'; unit = 'ERG/CM2/S/A'
    name = 'START-BKG'; format = '1I'; unit = 'PIXEL'
    name = 'END-BKG'; format = '1I'; unit = 'PIXEL'
    name = 'SCALE_BKG'; format = '1E'
    name = 'COEFF'; format = '7E'
)>
```

[140]:
```
# The data are available using (this example is NOT the right way of plotting␣
 ↪the data, it's just an example)
# and don't forget to import numpy as np to have np.arange working]:

data1 = hdulist[1].data
DTs =  data1.ABS_CAL
WLs = data1.WAVELENGTH
DWs = data1.DELTAW
for WL, DW, DT in zip(WLs, DWs, DTs):
    plt.plot(WL + np.arange(len(DT)) * DW, DT)
```



1.4.4 Writing FITS files

[141]:
```
# Creation of numpy array with the data.
x = np.arange(100)
```

[142]:
```
# Creation of the HDU from the data.
hdu = fits.PrimaryHDU(x)
print(hdu.header.cards)
```

```
('SIMPLE', True, 'conforms to FITS standard')
('BITPIX', 64, 'array data type')
('NAXIS', 1, 'number of array dimensions')
('NAXIS1', 100, '')
('EXTEND', True, '')
```

[143]:
```
#Adding additional keywords to the header.
# The automatically created header contains only the required minimum of␣
 ↪keywords.
# If additional keywords are needed they are added with:
hdu.header['testkey'] = (0.001,'some test value')
```

[144]:
```
print(hdu.header.cards)
```

```
('SIMPLE', True, 'conforms to FITS standard')
('BITPIX', 64, 'array data type')
('NAXIS', 1, 'number of array dimensions')
('NAXIS1', 100, '')
('EXTEND', True, '')
('TESTKEY', 0.001, 'some test value')
```

[145]:
```
hdulist = fits.HDUList([hdu])
hdulist.writeto('new.fits', overwrite=True)
hdulist.close()
```

Another way to deal with FITS tables is to use the ATpy library, we'll see this later