Advanced Astronomical Observations 2022
Session 02: Manipulating FITS Files

Kinoshita Daisuke

25 February 2022
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2022 to June 2022.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we deal with FITS files. FITS (Flexible Image Transport System) is a standard file format for astronomy. Almost all the data produced at research-oriented astronomical observatories are FITS files. For astronomical data reduction and analysis, we need to read and write FITS files. For today's session, we download a set of FITS files, and try basic operations of FITS files.

1 Downloading data

A set of FITS files is placed at following location. Download the file.

- `https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s02.tar.xz`

If you prefer to use a command-line tool, such as `curl`, then try following command.

```
% curl -k -o data_s02.tar.xz \
? https://s3b.astro.ncu.edu.tw/advobs_202202/data/data_s02.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  409M  100  409M    0     0  5427k      0  0:01:17  0:01:17 --:--:-- 5443k
% ls
data_s02.tar.xz
% ls -lF data_s02.tar.xz
-rw-r--r--  1 daisuke  taiwan  429546208 Feb 24 13:11 data_s02.tar.xz
```

If you prefer to use a web browser, such as Firefox, then start a web browser and download the file.
Use the command `file` to check the file type of the file `data_s02.tar.xz`.

```
% file data_s02.tar.xz
data_s02.tar.xz: XZ compressed data, checksum CRC64
```

It is XZ compressed data. If you would like to learn about `file` command, try following.

```
% man 1 file
```

Then, you can read the manual page of `file` command. (Fig. 1)

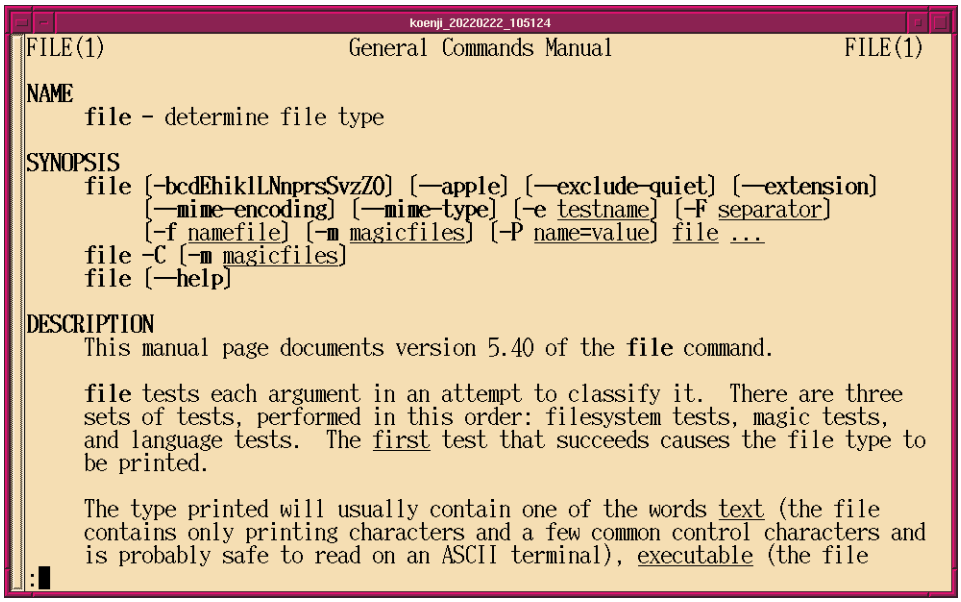

Figure 1: The manual page of the command `file`.

2 Extracting data

The file you have downloaded is a compressed TAR archive file. To extract FITS files, try following command. 120 FITS files should be extracted from the archive file.

```
% ls -lF
total 410
-rw-r--r--  1 daisuke  taiwan  429546208 Feb 24 13:11 data_s02.tar.xz
% tar xJvf data_s02.tar.xz
x data_s02/
x data_s02/lot_20210214_0245.fits
x data_s02/lot_20210214_0246.fits
x data_s02/lot_20210214_0247.fits
x data_s02/lot_20210214_0248.fits
x data_s02/lot_20210214_0249.fits

.....

x data_s02/lot_20210214_0534.fits
x data_s02/lot_20210214_0535.fits
x data_s02/lot_20210214_0536.fits
x data_s02/lot_20210214_0537.fits
x data_s02/lot_20210214_0538.fits
```

```
% ls -lF
total 410
drwxr-xr-x  2 daisuke  taiwan        4096 Feb 25  2021 data_s02/
-rw-r--r--  1 daisuke  taiwan  429546208 Feb 24 13:11 data_s02.tar.xz
% ls -lF data_s02 | head
total 968
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13  2021 lot_20210214_0245.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13  2021 lot_20210214_0246.fits
-rw-r--r--  1 daisuke  taiwan  8395200 Feb 13  2021 lot_20210214_0247.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0248.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0249.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0250.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0251.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0252.fits
-rw-r--r--  1 daisuke  taiwan  8400960 Feb 13  2021 lot_20210214_0253.fits
% ls data_s02/*.fits | wc
     120     120    3840
```

If above command does not work on your computer, then try following.

```
% unxz -c data_s02.tar.xz | tar xvf -
x data_s02/
x data_s02/lot_20210214_0245.fits
x data_s02/lot_20210214_0246.fits
x data_s02/lot_20210214_0247.fits
x data_s02/lot_20210214_0248.fits
x data_s02/lot_20210214_0249.fits

.....

x data_s02/lot_20210214_0534.fits
x data_s02/lot_20210214_0535.fits
x data_s02/lot_20210214_0536.fits
x data_s02/lot_20210214_0537.fits
x data_s02/lot_20210214_0538.fits
```

If above command fails, you probably do not have XZ Utils. Try command `which` to check whether you have command `xz` on your computer. If you see following, then you have `xz` installed on your computer.

```
% which xz
/usr/bin/xz
```

If you see following, then you do not have `xz` on your computer.

```
% which xz
xz: Command not found.
```

If you do not have XZ Utils, visit following website (Fig. 2) and install XZ Utils.

- `https://tukaani.org/xz/`

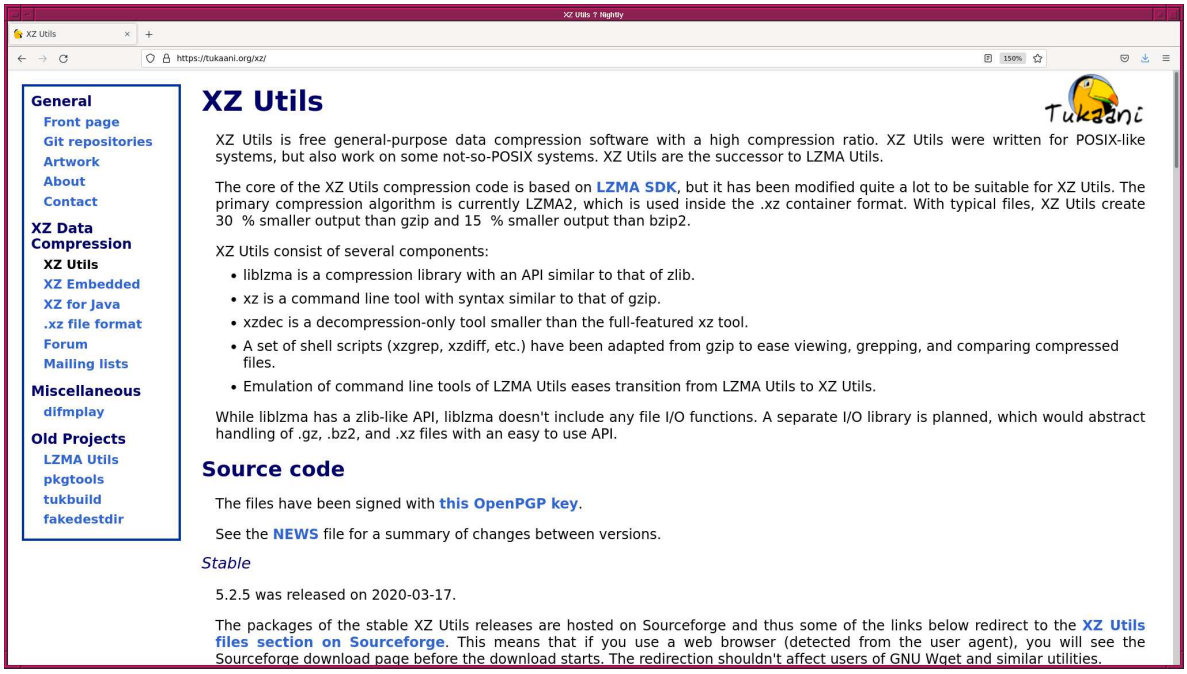

Figure 2: The website of XZ Utils.

3 Importing Astropy module

3.1 Importing Astropy module in interactive mode

For today's session, we need to use Astropy module. Check whether or not you have Astropy on your computer. If Astropy is properly installed on your computer, you can successfully import Astropy by typing `import astropy`. (Fig. 3)

```
% python3.9
Python 3.9.10 (main, Feb  6 2022, 14:59:28)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astropy
>>> exit ()
```

If you see error message after typing `import astropy`, then you do not have Astropy on your computer.

```
>>> import astropy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'astropy'
>>> exit ()
```

Visit the official website of Astropy (Fig. 4), and install Astropy on your computer.

3.2 Making a simple Python script using Astropy

Make some simple Python scripts using Astropy.

Python Code 1: advobs202202_s02_01.py

```
#!/usr/pkg/bin/python3.9
```

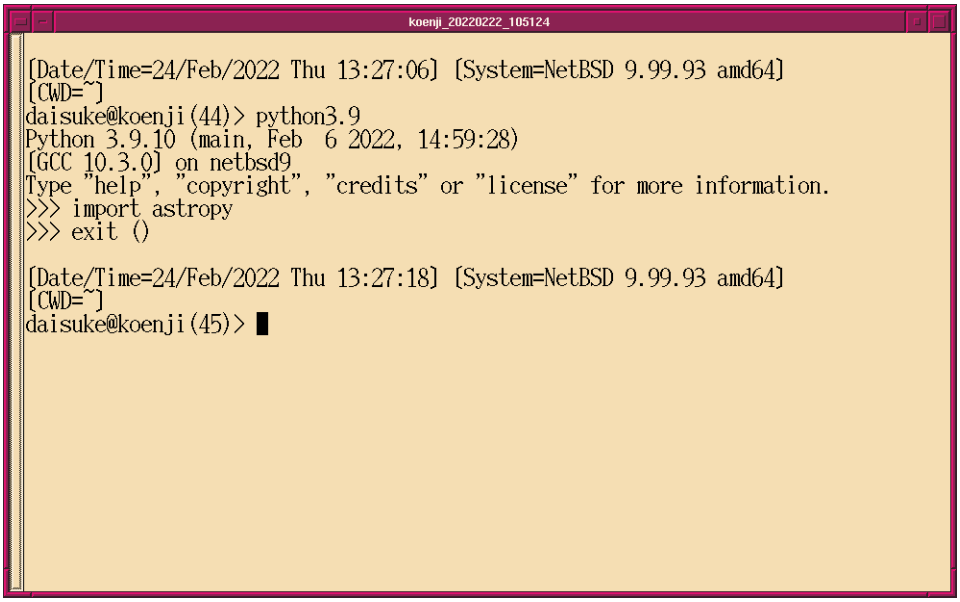

Figure 3: Importing Astropy module in interactive mode of Python.

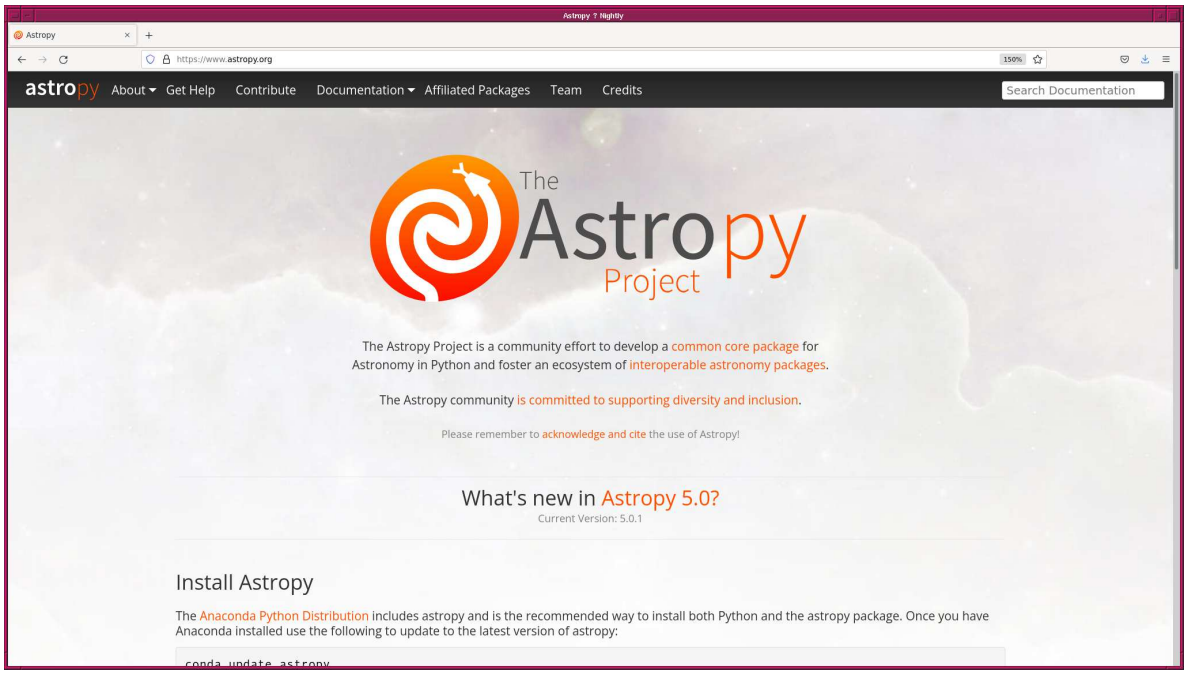

Figure 4: The official website of Astropy.

```
# importing Astropy module
import astropy.constants

# gravitational constant
G = astropy.constants.G

# printing gravitational constant G
print (G)
```

Execute the script.

```
% chmod a+x advobs202202_s02_01.py
% ./advobs202202_s02_01.py
  Name   = Gravitational constant
  Value  = 6.6743e-11
  Uncertainty  = 1.5e-15
  Unit  = m3 / (kg s2)
  Reference = CODATA 2018
```

Astropy is successfully imported, and the information about the gravitational constant is shown.
Try unit conversion.

Python Code 2: advobs202202_s02_02.py

```
#!/usr/pkg/bin/python3.9

# importing astropy module
import astropy.units

# units
u_m  = astropy.units.m
u_pc = astropy.units.pc

# distance in pc
d_pc = 1.32 * u_pc

# converting the distance in pc into metre
d_m = d_pc.to (u_m)

# printing result
print (d_pc, "=", d_m)
```

Execute the script.

```
% chmod a+x advobs202202_s02_02.py
% ./advobs202202_s02_02.py
1.32 pc = 4.073094407568605e+16 m
```

Try time calculation.

Python Code 3: advobs202202_s02_03.py

```
#!/usr/pkg/bin/python3.9

# importing astropy module
```

```
import astropy
import astropy.time

# time "t1"
t1 = astropy.time.Time ('2022-02-25T12:00:00.000', format='isot', scale='utc')

# printing the time "t1"
print ("t1 =", t1)

# JD
t1_jd = t1.jd

# printing JD corresponding to the time "t1"
print ("t1 in JD =", t1_jd)

# time "t2"
t2 = astropy.time.Time ('2000-01-01T12:00:00.000', format='isot', scale='utc')

# printing the time "t2"
print ("t2 =", t2)

# calculating the time difference between "t1" and "t2"
dt = t1 - t2

# printing "dt"
print ("dt = t1 - t2 = ", dt, "day =", dt.sec, "sec")
```

Execute the script.

```
% chmod a+x advobs202202_s02_03.py
% ./advobs202202_s02_03.py
t1 = 2022-02-25T12:00:00.000
t1 in JD = 2459636.0
t2 = 2000-01-01T12:00:00.000
dt = t1 - t2 =  8091.00005787037 day = 699062405.0 sec
```

Try coordinate conversion.

Python Code 4: advobs202202_s02_04.py

```
#!/usr/pkg/bin/python3.9

# importing astropy module
import astropy
import astropy.coordinates
import astropy.units

# equatorial coordinate of Regulus (alpha Leo)
coo_regulus = astropy.coordinates.SkyCoord ('10h08m22.31s', '+11d58m02.0s', \
                                            frame='icrs')

# printing coordinate
print ("Regulus:")
print ("  (RA, Dec) = (%s, %s) = (%8.4f deg, %8.4f deg)" \
       % (coo_regulus.ra, coo_regulus.dec, \
          coo_regulus.ra.deg, coo_regulus.dec.deg) )

# transformation into ecliptic coordinate
```

```
ecl_regulus = coo_regulus.transform_to ('geocentricmeanecliptic')

# printing results
print ("  (lambda, beta) = (%f deg, %f deg)" \
       % (ecl_regulus.lon.deg, ecl_regulus.lat.deg) )

# units
u_hourangle = astropy.units.hourangle
u_deg       = astropy.units.deg

# equatorial coordinate of Antares (alpha Sco)
coo_antares = astropy.coordinates.SkyCoord ('16 29 24.46', '-26 25 55.2', \
                                            frame='icrs', \
                                            unit=(u_hourangle, u_deg) )

# printing coordinate
print ("Antares:")
print ("  (RA, Dec) = (%s, %s) = (%8.4f deg, %8.4f deg)" \
       % (coo_antares.ra, coo_antares.dec, \
          coo_antares.ra.deg, coo_antares.dec.deg) )

# transformation into galactic coordinate
gal_antares = coo_antares.transform_to ('galactic')

# printing results
print ("  (l, b) = (%f deg, %f deg)" % (gal_antares.l.deg, gal_antares.b.deg) )
```

Execute the script.

```
% chmod a+x advobs202202_s02_04.py
% ./advobs202202_s02_04.py
Regulus:
  (RA, Dec) = (152d05m34.65s, 11d58m02s) = (152.0930 deg,  11.9672 deg)
  (lambda, beta) = (149.832894 deg, 0.464824 deg)
Antares:
  (RA, Dec) = (247d21m06.9s, -26d25m55.2s) = (247.3519 deg, -26.4320 deg)
  (l, b) = (351.947140 deg, 15.064323 deg)
```

Try to plot blackbody radiation curves.

Python Code 5: advobs202202_s02_05.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy
import astropy.modeling.models
import astropy.units

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg
```

```
# construction of parser object
desc = 'plotting blackbody models'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-o', help='output file name')
parser.add_argument ('T', nargs='+', type=float, \
                     help='list of blackbody temperatures in K')

# command-line argument analysis
args = parser.parse_args ()

# parameters
file_output = args.o
list_T      = args.T

# number of blackbody models
n_bb = len (list_T)

# units
unit_micron = astropy.units.micron
unit_K      = astropy.units.K

# wavelength
wl_min = -8.0
wl_max = -4.0
n_wl   = 10**4
wl     = numpy.logspace (wl_min, wl_max, num=n_wl) * 10**6 * unit_micron

# making an empty list for blackbody data
bb_data = []

# blackbody radiation
for T in list_T:
    # temperature
    T = T * unit_K
    # making a blackbody model
    bb = astropy.modeling.models.BlackBody (temperature=T)
    # generating blackbody curve data
    bb_data.extend ([bb (wl)])

# making objects "fig" and "ax" for plotting
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# labels
label_x = 'Wavelength [micron]'
label_y = 'Spectral Radiance [erg sec^-1 cm^-2, sr^-1 Hz^-1]'
ax.set_xlabel (label_x)
ax.set_ylabel (label_y)

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.01, 100)
ax.set_ylim (10**-10, 10**0)
ax.grid ()
```

```
#ax.ticklabel_format (axis='y', style='sci', scilimits=(0,0))

# plotting data
for i in range (n_bb):
    # temperature
    T_str = "%d K blackbody" % list_T[i]
    # plotting data
    ax.plot (wl, bb_data[i], '-', linewidth=3, label=T_str)
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute the script and generate a plot.

```
% chmod a+x advobs202202_s02_05.py
% ./advobs202202_s02_05.py -h
usage: advobs202202_s02_05.py [-h] [-o O] T [T ...]

plotting blackbody models

positional arguments:
  T           list of blackbody temperatures in K

optional arguments:
  -h, --help  show this help message and exit
  -o O        output file name

% ./advobs202202_s02_05.py -o blackbody.pdf 3000 6000 30000
% ls -lF blackbody.pdf
-rw-r--r--  1 daisuke  taiwan  17276 Feb 24 15:31 blackbody.pdf
```

Now, you have a file named `blackbody.pdf`. Use your favourite PDF viewer software to show the plot. (Fig. 5)

```
% okular blackbody.pdf
```

To know more about Astropy, read following document.

- Astropy Documentation: `https://docs.astropy.org/` (Fig. 6)

4 About Numpy and Matplotlib

If you are new to Numpy, download following material and read it.

- "Session 03: Using Numpy" of the course "Astroinformatics" offered at 2nd semester of academic year 2020
  - `https://s3b.astro.ncu.edu.tw/advobs_202202/pdf/astroinformatics_2021_03e.pdf`

You may also read the official document of Numpy.

- Numpy User Guide: `https://numpy.org/doc/stable/user/` (Fig. 7)

If you are new to Matplotlib, download following material and read it.

- "Session 04: Using Matplotlib" of the course "Astroinformatics" offered at 2nd semester of academic year 2020
  - `https://s3b.astro.ncu.edu.tw/advobs_202202/pdf/astroinformatics_2021_04e.pdf`

You may also read the official document of Matplotlib.

- Matplotlib User Guide: `https://matplotlib.org/stable/users/index` (Fig. 8)

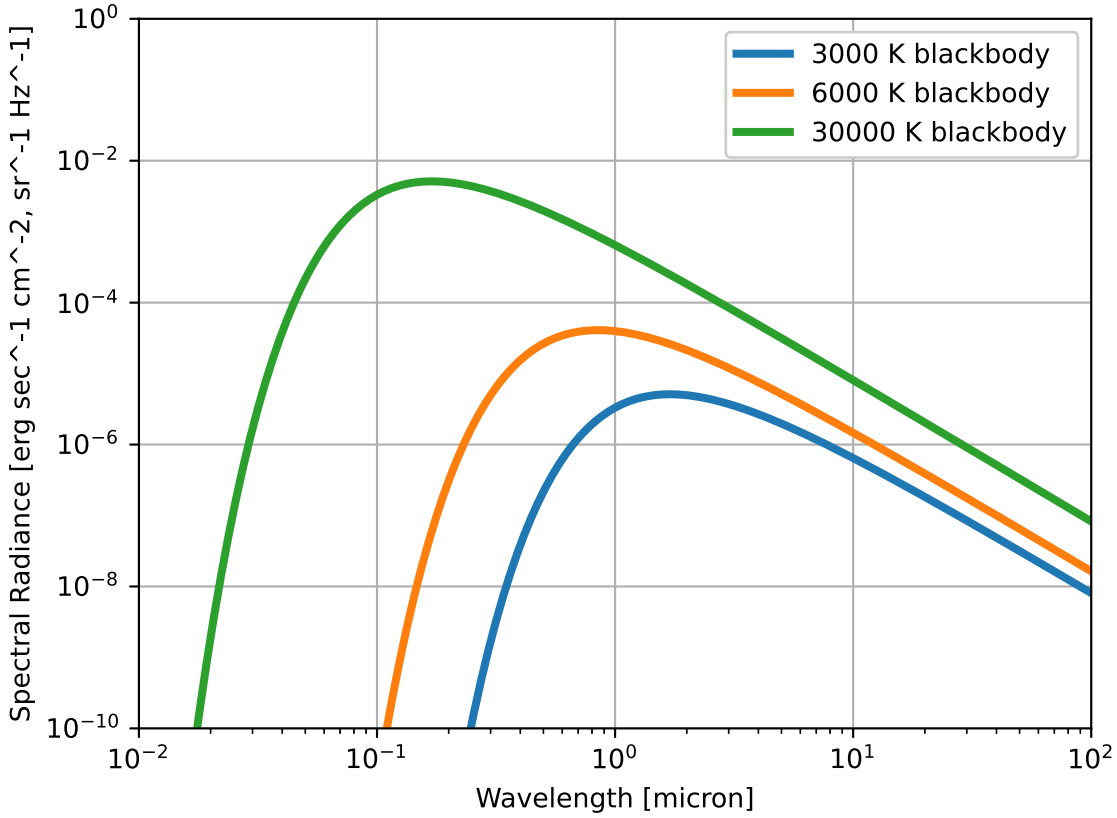

Figure 5: Blackbody radiation of $T = 3000, 6000$, and 30000 K.

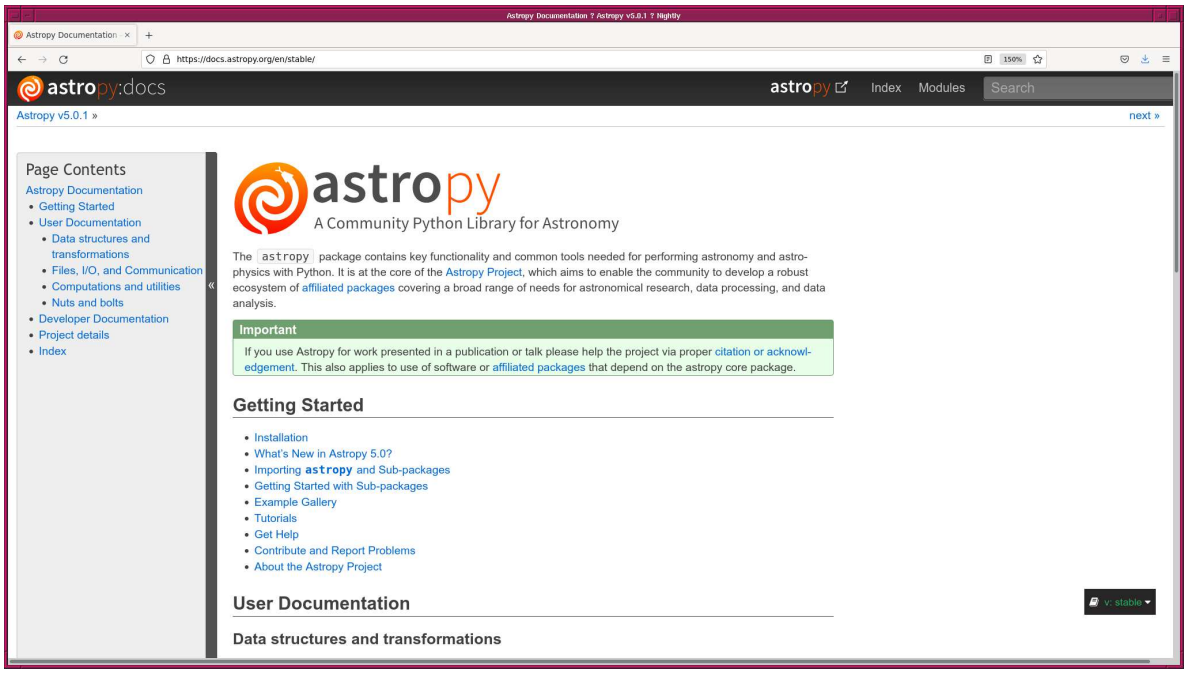

Figure 6: The official documentation of Astropy.

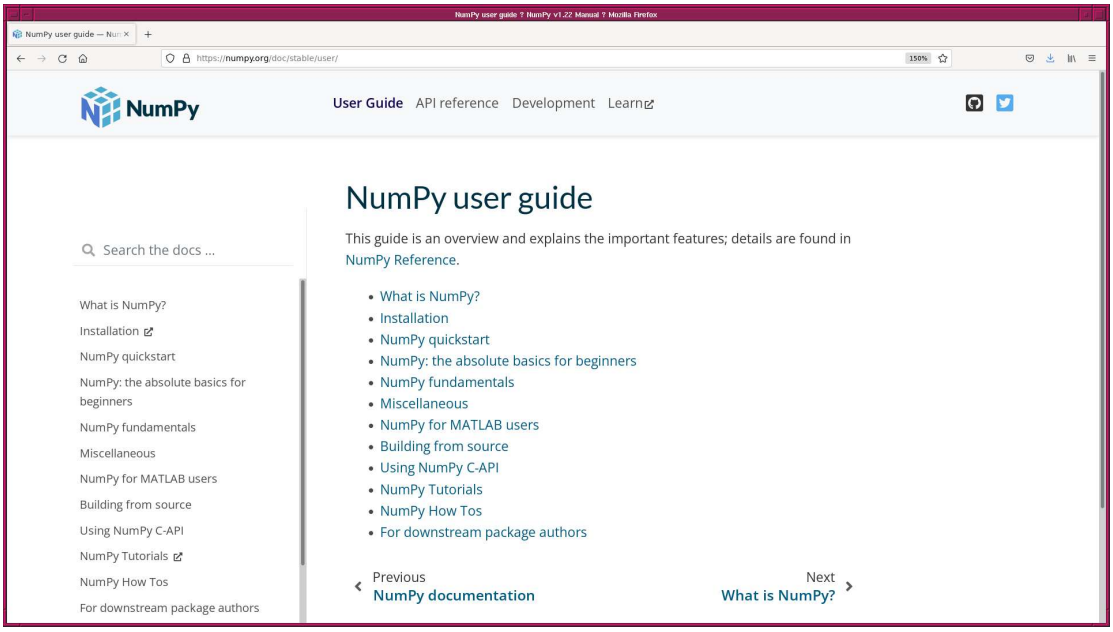

Figure 7: the web page of "Numpy User Guide".

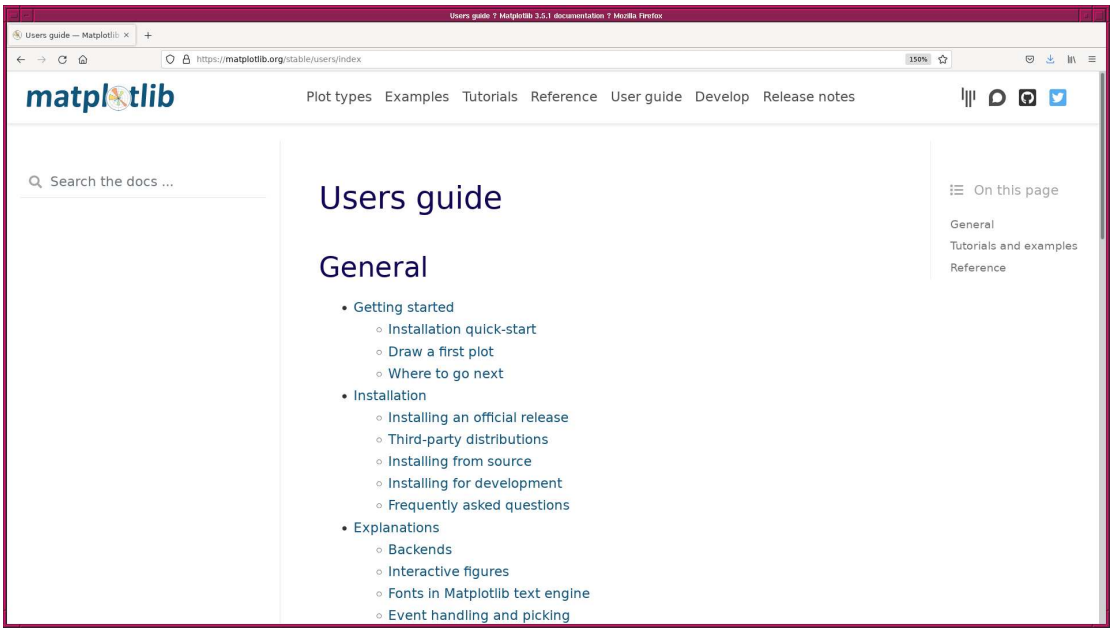

Figure 8: the web page of "Matplotlib User Guide".

5 Opening a FITS file

Make a Python script to open a FITS file and print HDU list information. Here is an example.

Python Code 6: advobs202202_s02_06.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy
import astropy.io.fits

# construction of parser object
desc = 'opening FITS files and printing HDU information'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='list of FITS files')

# command-line argument analysis
args = parser.parse_args ()

# parameters
list_fits = args.files

# processing FITS files one-by-one
for file_fits in list_fits:
    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # printing HDU list information
    print (hdu_list.info () )

    # closing FITS file
    hdu_list.close ()
```

Execute the script.

```
% chmod a+x advobs202202_s02_06.py
% ./advobs202202_s02_06.py data_s02/lot_20210214_0245.fits
Filename: data_s02/lot_20210214_0245.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      61   (2048, 2048)   int16 (rescales to uint16)
None
% ./advobs202202_s02_06.py data_s02/lot_20210214_0250.fits \
? data_s02/lot_20210214_0260.fits data_s02/lot_20210214_0270.fits
Filename: data_s02/lot_20210214_0250.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     111   (2048, 2048)   int16 (rescales to uint16)
None
Filename: data_s02/lot_20210214_0260.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      78   (2048, 2048)   int16 (rescales to uint16)
None
Filename: data_s02/lot_20210214_0270.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
```

```
  0  PRIMARY       1 PrimaryHDU      61   (2048, 2048)   int16 (rescales to uint16)
None
```

One HDU (Header Data Unit) is found in the file. The dimensions of the image data is 2048×2048, and the image data is stored as 16-bit integer.

6 Printing FITS header

Make a Python script to read the header part of a FITS file and print it.

Python Code 7: advobs202202_s02_07.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy
import astropy.io.fits

# construction of parser object
desc = 'opening FITS files and printing header information'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('file', help='name of FITS file')

# command-line argument analysis
args = parser.parse_args ()

# parameters
file_fits = args.file

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# closing FITS file
hdu_list.close ()

# printing FITS header
print (repr (header0))
```

Execute the script. You will see the header of a FITS file.

```
% chmod a+x advobs202202_s02_07.py
% ./advobs202202_s02_07.py -h
usage: advobs202202_s02_07.py [-h] file

opening FITS files and printing header information
```

```
positional arguments:
  file        name of FITS file

optional arguments:
  -h, --help  show this help message and exit

% ./advobs202202_s02_07.py data_s02/lot_20210214_0245.fits
SIMPLE  =                    T
BITPIX  =                   16 /8 unsigned int, 16 & 32 int, -32 & -64 real
NAXIS   =                    2 /number of axes
NAXIS1  =                 2048 /fastest changing axis
NAXIS2  =                 2048 /next to fastest changing axis
BSCALE  =   1.0000000000000000 /physical = BZERO + BSCALE*array_value
BZERO   =   32768.000000000000 /physical = BZERO + BSCALE*array_value
DATE-OBS= '2021-02-12' /        YYYY-MM-DDThh:mm:ss observation start, UT
TIME-OBS= '18:16:20' /          HH:MM:SS observation start time, UT
EXPTIME =  0.00000000000000000 /Exposure time in seconds
EXPOSURE=  0.00000000000000000 /Exposure time in seconds
SET-TEMP=  -80.000000000000000 /CCD temperature setpoint in C
CCD-TEMP=  -80.000000000000000 /CCD temperature at start of exposure in C
XPIXSZ  =   15.000000000000000 /Pixel Width in microns (after binning)
YPIXSZ  =   15.000000000000000 /Pixel Height in microns (after binning)
XBINNING=                    1 /Binning factor in width
YBINNING=                    1 /Binning factor in height
XORGSUBF=                    0 /Subframe X position in binned pixels
YORGSUBF=                    0 /Subframe Y position in binned pixels
IMAGETYP= 'BIAS    ' /          Type of image
OBJCTRA = '12 32 04' /          Nominal Right Ascension of center of image
OBJCTDEC= '-02 06 03' /         Nominal Declination of center of image
OBJCTALT= ' 62.4274' /          Nominal altitude of center of image
OBJCTAZ = '157.3035' /          Nominal azimuth of center of image
OBJCTHA = ' -0.6864' /          Nominal hour angle of center of image
SITELAT = '23 28 07' /          Latitude of the imaging location
SITELONG= '120 52 25' /         Longitude of the imaging location
JD      =   2459258.2613425925 /Julian Date at start of exposure
JD-HELIO=   2459258.2654279913 /Heliocentric Julian Date at exposure midpoint
AIRMASS =   1.1277189765535098 /Relative optical path length through atmosphere
FOCALLEN=   8000.0000000000000 /Focal length of telescope in mm
APTDIA  =   1000.0000000000000 /Aperture diameter of telescope in mm
APTAREA =   772124.95592236519 /Aperture area of telescope in mm^2
SWCREATE= 'MaxIm DL Version 5.24 130419 0CYVP' /Name of software that created
         the image
OBJECT  = 'dark    '
TELESCOP= 'LOT     ' /          telescope used to acquire this image
INSTRUME= 'Driver for Princeton Instruments cameras'
OBSERVER= 'lulin   '
NOTES   = '        '
DETECTOR= '        '
OWNER   = 'Institute of Astronomy, NCU, Taiwan'
TIMESYS = 'UTC     '
EQUINX  = '2000    '
EPOCH   = '2000    '
RADECSYS= 'FK5     '
CAMERA  = 'SOPHIA  '
GAIN    = '2       '
SITEELEV=                 2862
RMSNOISE= '8.5     '
OPERATOR= 'lulin   '
CTYPE1  = 'RA---TAN' /          fastest changing axis name
```

```
CTYPE2  = 'DEC--TAN' /           slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '         '
SWOWNER = 'Ming-Hsin Chang' /    Licensed owner of software
CSTRETCH= 'Medium   ' /          Initial display stretch mode
CBLACK  =                  594 /Initial display black level in ADUs
CWHITE  =                  611 /Initial display white level in ADUs
PEDESTAL=                    0 /Correction to add for zero-based ADU
```

We can find that the FITS file "lot_20210214_0245.fits" is a bias frame taken at 18:16:20 on 2021-02-12.

7 Printing specific keywords in the header

Make a Python script to print a specific keyword and its value in the FITS header.

Python Code 8: advobs202202_s02_08.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy
import astropy.io.fits

# construction of parser object
desc = 'opening FITS files and printing values of selected header keywords'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-k', '--keywords', \
                     help='list of keywords (e.g. "NAXIS1,NAXIS2,IMAGETYP")')
parser.add_argument ('files', nargs='+', help='list of FITS files')

# command-line argument analysis
args = parser.parse_args ()

# parameters
list_fits = args.files
keywords  = args.keywords

# keywords
list_keywords = keywords.split (',')

# processing files one-by-one
for file_fits in list_fits:
    # opening FITS file
    hdu_list = astropy.io.fits.open (file_fits)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # closing FITS file
```

```
    hdu_list.close ()

    # printing specific keywords and their values in FITS header
    print ("file = %s" % file_fits)
    for keyword in list_keywords:
        print ("  %-8s ==> %s" % (keyword, header0[keyword]) )
```

Run the script.

```
% chmod a+x advobs202202_s02_08.py
% ./advobs202202_s02_08.py -h
usage: advobs202202_s02_08.py [-h] [-k KEYWORDS] files [files ...]

opening FITS files and printing values of selected header keywords

positional arguments:
  files                 list of FITS files

optional arguments:
  -h, --help            show this help message and exit
  -k KEYWORDS, --keywords KEYWORDS
                        list of keywords (e.g. "NAXIS1,NAXIS2,IMAGETYP")

% ./advobs202202_s02_08.py -k TIME-OBS,IMAGETYP,EXPTIME \
? data_s02/lot_20210214_0245.fits
file = data_s02/lot_20210214_0245.fits
  TIME-OBS ==> 18:16:20
  IMAGETYP ==> BIAS
  EXPTIME  ==> 0.0
```

Multiple files can be handled.

```
% ./advobs202202_s02_08.py -k TIME-OBS,IMAGETYP,EXPTIME \
? data_s02/lot_20210214_0245.fits data_s02/lot_20210214_0260.fits
file = data_s02/lot_20210214_0245.fits
  TIME-OBS ==> 18:16:20
  IMAGETYP ==> BIAS
  EXPTIME  ==> 0.0
file = data_s02/lot_20210214_0260.fits
  TIME-OBS ==> 18:48:02
  IMAGETYP ==> LIGHT
  EXPTIME  ==> 60.0
```

8 Generating a simple observing log

Make a Python script to generate a simple observing log. Here is an example.

Python Code 9: advobs202202_s02_09.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy
```

```
import astropy.io.fits

# construction of parser object
desc = 'Generating a simple observing log'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
default_keyword = 'TIME-OBS,IMAGETYP,OBJECT,EXPTIME,FILTER'
parser.add_argument ('-k', '--keyword', default=default_keyword, \
                     help='a comma-separated list of FITS keywords')
parser.add_argument ('files', nargs='+', help='list of FITS files')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
keyword = args.keyword
files   = args.files

# a list of keywords
list_keyword = keyword.split (',')

# processing files
for file in files:
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # gathering information from FITS header
    record = filename
    for key in list_keyword:
        if key in header0:
            value = str (header0[key])
        else:
            value = "__NONE__"
        record += " %8s" % value

    # closing FITS file
    hdu_list.close ()

    # printing information
    print (record)
```

Execute the script as follows, and generate a simple observing log. The file name, time of the observation in UT, data type, target object name, exposure time, and filter name are shown.

```
% chmod a+x advobs202202_s02_09.py
% ./advobs202202_s02_09.py -k time-obs,imagetyp,object,exptime,filter \
? data_s02/*.fits
lot_20210214_0245.fits 18:16:20     BIAS     dark      0.0 __NONE__
lot_20210214_0246.fits 18:16:27     BIAS     dark      0.0 __NONE__
lot_20210214_0247.fits 18:16:34     BIAS     dark      0.0 __NONE__
lot_20210214_0248.fits 18:18:33    LIGHT   PG1047    180.0 gp_Astrodon_2019
lot_20210214_0249.fits 18:21:50    LIGHT   PG1047    180.0 gp_Astrodon_2019
lot_20210214_0250.fits 18:25:16    LIGHT   PG1047    180.0 rp_Astrodon_2019
lot_20210214_0251.fits 18:28:33    LIGHT   PG1047    180.0 rp_Astrodon_2019
lot_20210214_0252.fits 18:31:58    LIGHT   PG1047    180.0 ip_Astrodon_2019
lot_20210214_0253.fits 18:35:15    LIGHT   PG1047    180.0 ip_Astrodon_2019
lot_20210214_0254.fits 18:38:55     BIAS     dark      0.0 __NONE__
lot_20210214_0255.fits 18:39:02     BIAS     dark      0.0 __NONE__
lot_20210214_0256.fits 18:39:09     BIAS     dark      0.0 __NONE__
lot_20210214_0257.fits 18:44:18    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0258.fits 18:45:34    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0259.fits 18:46:49    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0260.fits 18:48:02    LIGHT    HCG57     60.0 gp_Astrodon_2019
lot_20210214_0261.fits 18:49:25    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0262.fits 18:50:40    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0263.fits 18:51:55    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0264.fits 18:53:10    LIGHT    HCG57     60.0 rp_Astrodon_2019
lot_20210214_0265.fits 18:54:34    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0266.fits 18:55:50    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0267.fits 18:57:05    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0268.fits 18:58:18    LIGHT    HCG57     60.0 ip_Astrodon_2019
lot_20210214_0269.fits 18:59:54     BIAS     dark      0.0 __NONE__
lot_20210214_0270.fits 19:00:01     BIAS     dark      0.0 __NONE__
lot_20210214_0271.fits 19:00:08     BIAS     dark      0.0 __NONE__
lot_20210214_0272.fits 19:02:27    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0273.fits 19:03:43    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0274.fits 19:04:57    LIGHT V0678VIR     60.0 rp_Astrodon_2019
lot_20210214_0275.fits 19:11:50     BIAS     dark      0.0 __NONE__
lot_20210214_0276.fits 19:11:57     BIAS     dark      0.0 __NONE__
lot_20210214_0277.fits 19:12:04     BIAS     dark      0.0 __NONE__
lot_20210214_0452.fits 22:03:45     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0453.fits 22:03:57     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0454.fits 22:04:09     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0455.fits 22:04:26     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0456.fits 22:04:37     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0457.fits 22:04:49     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0458.fits 22:05:05     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0459.fits 22:05:17     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0460.fits 22:05:29     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0461.fits 22:05:45     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0462.fits 22:05:57     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0463.fits 22:06:08     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0464.fits 22:06:24     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0465.fits 22:06:36     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0466.fits 22:06:48     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0467.fits 22:07:03     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0468.fits 22:07:15     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0469.fits 22:07:26     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0470.fits 22:07:42     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0471.fits 22:07:53     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0472.fits 22:08:05     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0473.fits 22:08:22     FLAT     dark      5.0 ip_Astrodon_2019
```

```
lot_20210214_0474.fits 22:08:34     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0475.fits 22:08:45     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0476.fits 22:09:01     FLAT     dark      5.0 ip_Astrodon_2019
lot_20210214_0477.fits 22:09:13     FLAT     dark      5.0 rp_Astrodon_2019
lot_20210214_0478.fits 22:09:24     FLAT     dark      5.0 gp_Astrodon_2019
lot_20210214_0479.fits 22:14:37     BIAS     dark      0.0 __NONE__
lot_20210214_0480.fits 22:14:42     BIAS     dark      0.0 __NONE__
lot_20210214_0481.fits 22:14:47     BIAS     dark      0.0 __NONE__
lot_20210214_0482.fits 22:14:52     BIAS     dark      0.0 __NONE__
lot_20210214_0483.fits 22:14:58     BIAS     dark      0.0 __NONE__
lot_20210214_0484.fits 22:15:03     BIAS     dark      0.0 __NONE__
lot_20210214_0485.fits 22:15:08     BIAS     dark      0.0 __NONE__
lot_20210214_0486.fits 22:15:13     BIAS     dark      0.0 __NONE__
lot_20210214_0487.fits 22:15:18     BIAS     dark      0.0 __NONE__
lot_20210214_0488.fits 22:15:23     BIAS     dark      0.0 __NONE__
lot_20210214_0489.fits 22:15:28     BIAS     dark      0.0 __NONE__
lot_20210214_0490.fits 22:15:33     BIAS     dark      0.0 __NONE__
lot_20210214_0491.fits 22:15:39     BIAS     dark      0.0 __NONE__
lot_20210214_0492.fits 22:15:44     BIAS     dark      0.0 __NONE__
lot_20210214_0493.fits 22:15:49     BIAS     dark      0.0 __NONE__
lot_20210214_0494.fits 22:15:54     BIAS     dark      0.0 __NONE__
lot_20210214_0495.fits 22:15:59     BIAS     dark      0.0 __NONE__
lot_20210214_0496.fits 22:16:04     BIAS     dark      0.0 __NONE__
lot_20210214_0497.fits 22:16:09     BIAS     dark      0.0 __NONE__
lot_20210214_0498.fits 22:16:14     BIAS     dark      0.0 __NONE__
lot_20210214_0499.fits 22:16:20     DARK     dark      5.0 __NONE__
lot_20210214_0500.fits 22:16:30     DARK     dark      5.0 __NONE__
lot_20210214_0501.fits 22:16:40     DARK     dark      5.0 __NONE__
lot_20210214_0502.fits 22:16:50     DARK     dark      5.0 __NONE__
lot_20210214_0503.fits 22:17:00     DARK     dark      5.0 __NONE__
lot_20210214_0504.fits 22:17:10     DARK     dark      5.0 __NONE__
lot_20210214_0505.fits 22:17:20     DARK     dark      5.0 __NONE__
lot_20210214_0506.fits 22:17:30     DARK     dark      5.0 __NONE__
lot_20210214_0507.fits 22:17:40     DARK     dark      5.0 __NONE__
lot_20210214_0508.fits 22:17:50     DARK     dark      5.0 __NONE__
lot_20210214_0509.fits 22:17:59     DARK     dark      5.0 __NONE__
lot_20210214_0510.fits 22:18:10     DARK     dark      5.0 __NONE__
lot_20210214_0511.fits 22:18:20     DARK     dark      5.0 __NONE__
lot_20210214_0512.fits 22:18:30     DARK     dark      5.0 __NONE__
lot_20210214_0513.fits 22:18:40     DARK     dark      5.0 __NONE__
lot_20210214_0514.fits 22:18:50     DARK     dark      5.0 __NONE__
lot_20210214_0515.fits 22:19:00     DARK     dark      5.0 __NONE__
lot_20210214_0516.fits 22:19:10     DARK     dark      5.0 __NONE__
lot_20210214_0517.fits 22:19:20     DARK     dark      5.0 __NONE__
lot_20210214_0518.fits 22:19:30     DARK     dark      5.0 __NONE__
lot_20210214_0519.fits 22:19:40     BIAS     dark      0.0 __NONE__
lot_20210214_0520.fits 22:19:45     BIAS     dark      0.0 __NONE__
lot_20210214_0521.fits 22:19:50     BIAS     dark      0.0 __NONE__
lot_20210214_0522.fits 22:19:55     BIAS     dark      0.0 __NONE__
lot_20210214_0523.fits 22:20:01     BIAS     dark      0.0 __NONE__
lot_20210214_0524.fits 22:20:06     BIAS     dark      0.0 __NONE__
lot_20210214_0525.fits 22:20:11     BIAS     dark      0.0 __NONE__
lot_20210214_0526.fits 22:20:16     BIAS     dark      0.0 __NONE__
lot_20210214_0527.fits 22:20:21     BIAS     dark      0.0 __NONE__
lot_20210214_0528.fits 22:20:26     BIAS     dark      0.0 __NONE__
lot_20210214_0529.fits 22:20:31     BIAS     dark      0.0 __NONE__
lot_20210214_0530.fits 22:20:36     BIAS     dark      0.0 __NONE__
lot_20210214_0531.fits 22:20:41     BIAS     dark      0.0 __NONE__
lot_20210214_0532.fits 22:20:46     BIAS     dark      0.0 __NONE__
```

```
lot_20210214_0533.fits 22:20:51     BIAS     dark      0.0 __NONE__
lot_20210214_0534.fits 22:20:56     BIAS     dark      0.0 __NONE__
lot_20210214_0535.fits 22:21:01     BIAS     dark      0.0 __NONE__
lot_20210214_0536.fits 22:21:06     BIAS     dark      0.0 __NONE__
lot_20210214_0537.fits 22:21:11     BIAS     dark      0.0 __NONE__
lot_20210214_0538.fits 22:21:16     BIAS     dark      0.0 __NONE__
```

Now we know better about what we have.

9 Reading image data from a FITS file

Make a Python script to read image data from a FITS file.

Python Code 10: advobs202202_s02_10.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing astropy module
import astropy
import astropy.io.fits

# construction of parser object
desc = 'opening FITS files and reading image data'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('file', help='name of FITS file')

# command-line argument analysis
args = parser.parse_args ()

# parameters
file_fits = args.file

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# closing FITS file
hdu_list.close ()

# printing a value of a pixel [1024,1024]
print ("image[1024,1024] =", data0[1024,1024])

# printing values of 10x10 subframe of near the centre of the image
print ("image[1024:1034,1024:1034] =")
print (data0[1024:1034,1024:1034])
```

Run the script.

```
% chmod a+x advobs202202_s02_10.py
% ./advobs202202_s02_10.py -h
usage: advobs202202_s02_10.py [-h] file

opening FITS files and reading image data

positional arguments:
  file        name of FITS file

optional arguments:
  -h, --help  show this help message and exit

% ./advobs202202_s02_10.py data_s02/lot_20210214_0245.fits
image[1024,1024] = 586
image[1024:1034,1024:1034] =
[[586 585 604 589 600 594 586 604 597 606]
 [608 587 605 584 584 585 597 611 590 597]
 [593 590 599 615 595 609 598 601 591 589]
 [594 593 589 600 595 595 592 584 598 598]
 [605 590 586 602 595 603 594 594 599 598]
 [593 589 585 592 600 594 594 590 606 601]
 [593 584 586 579 588 587 602 599 595 591]
 [591 605 604 598 604 601 583 599 606 606]
 [581 592 588 584 583 606 585 579 598 589]
 [594 593 595 592 588 599 602 593 592 578]]
```

The pixel values are shown. It is a bias frame, and pixels have values around 600.
Read image data of the other file.

```
% ./advobs202202_s02_10.py data_s02/lot_20210214_0246.fits
image[1024,1024] = 596
image[1024:1034,1024:1034] =
[[596 601 600 609 588 604 599 578 594 600]
 [583 594 600 592 596 585 592 589 603 600]
 [606 590 597 605 595 615 609 604 593 578]
 [583 579 590 585 599 604 591 599 589 586]
 [606 588 599 601 590 598 595 593 601 585]
 [595 584 591 590 594 604 608 602 576 595]
 [586 595 614 590 596 585 594 593 594 611]
 [600 589 588 599 599 590 588 604 601 606]
 [599 585 600 595 572 610 584 606 594 590]
 [586 599 595 576 583 595 604 602 588 592]]
```

Values are slightly different.

10 Average value of pixel values of an image

Make a Python script to calculate the average value of pixel values of an image of a FITS file.

Python Code 11: advobs202202_s02_11.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
```

```
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Calculating average value of pixel values of images'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='FITS files')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
list_files = args.files

# printing header
print ("%-32s %8s %8s %8s %8s %8s" \
       % ('file', 'max', 'min', 'mean', 'median', 'stddev') )
print ("%s" % '-' * 77)

# processing files
for file in list_files:
    # if the extension of the file is not '.fits', then skip
    if (file[-5:] != '.fits'):
        continue

    # file name
    path = file.split ('/')
    filename = path[-1]

    # opening FITS file
    hdu_list = astropy.io.fits.open (file)

    # primary HDU
    hdu0 = hdu_list[0]

    # header of primary HDU
    header0 = hdu0.header

    # data of primary HDU
    data0 = hdu0.data

    # closing FITS file
    hdu_list.close ()

    # calculations of statistical values using numpy
    data_max      = numpy.amax (data0)
    data_min      = numpy.amin (data0)
    data_mean     = numpy.mean (data0)
    data_median   = numpy.median (data0)
    data_variance = numpy.var (data0)
    data_stddev   = numpy.std (data0)
```

```
    # printing results
    print ("%-32s %8.2f %8.2f %8.2f %8.2f %8.2f" \
           % (filename, data_max, data_min, \
              data_mean, data_median, data_stddev) )
```

Use above script to examine bias frames.

```
% chmod a+x advobs202202_s02_11.py
% ./advobs202202_s02_11.py -h
usage: advobs202202_s02_11.py [-h] files [files ...]

Calculating average value of pixel values of images

positional arguments:
  files       FITS files

optional arguments:
  -h, --help  show this help message and exit

% ./advobs202202_s02_11.py data_s02/lot_20210214_0245.fits
file                                    max      min     mean   median   stddev
------------------------------------------------------------------------------
lot_20210214_0245.fits              1363.00   556.00   595.62   596.00     7.88
% ./advobs202202_s02_11.py data_s02/lot_20210214_02[45]?.fits
file                                    max      min     mean   median   stddev
------------------------------------------------------------------------------
lot_20210214_0245.fits              1363.00   556.00   595.62   596.00     7.88
lot_20210214_0246.fits              1590.00   551.00   594.80   595.00     7.90
lot_20210214_0247.fits               891.00   554.00   595.48   595.00     7.88
lot_20210214_0248.fits             65535.00   624.00   770.90   765.00   417.41
lot_20210214_0249.fits             65535.00   599.00   771.53   766.00   415.90
lot_20210214_0250.fits             65535.00   627.00   916.15   908.00   532.35
lot_20210214_0251.fits             65535.00   621.00   916.61   908.00   529.25
lot_20210214_0252.fits             65535.00   630.00   912.30   906.00   402.82
lot_20210214_0253.fits             65535.00   635.00   911.45   906.00   405.97
lot_20210214_0254.fits               635.00   558.00   596.27   596.00     7.92
lot_20210214_0255.fits              1883.00   556.00   595.85   596.00     7.95
lot_20210214_0256.fits               638.00   556.00   596.13   596.00     7.92
lot_20210214_0257.fits             65535.00   594.00   657.94   656.00   169.75
lot_20210214_0258.fits             65535.00   604.00   658.21   656.00   178.49
lot_20210214_0259.fits             65535.00   596.00   658.33   657.00   174.33
```

The mean count of bias frames is around 595.

```
% ./advobs202202_s02_11.py data_s02/lot_20210214_045?.fits
file                                    max      min     mean   median   stddev
------------------------------------------------------------------------------
lot_20210214_0452.fits             20811.00  3808.00  7421.30  7425.00   247.37
lot_20210214_0453.fits             47063.00  4564.00  8972.19  8988.00   262.94
lot_20210214_0454.fits             36558.00  4873.00  9471.12  9501.00   288.62
lot_20210214_0455.fits             26382.00  4386.00  8709.72  8714.00   289.10
lot_20210214_0456.fits             49441.00  5253.00 10564.77 10583.00   305.15
lot_20210214_0457.fits             32915.00  5757.00 11303.81 11343.00   345.61
lot_20210214_0458.fits             31871.00  5127.00 10235.88 10241.00   341.95
lot_20210214_0459.fits             52127.00  6100.00 12433.64 12456.00   353.98
```

The mean count of flatfield frames is around 10,000.

11 Writing a new FITS file

Make a Python script to read a FITS file, normalise the image, and write a new FITS file.

Python Code 12: advobs202202_s02_12.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Normalising an image'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('fits', help='input FITS file name')
parser.add_argument ('-o', '--output', default='new.fits', \
                     help='output FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# calculations of mean value
data_mean = numpy.mean (data0)

# printing mean value
print ("mean value of %s = %f" % (file_input, data_mean) )

# normalisation
data_new = data0 / data_mean

# calculation of mean value after normalisation
new_mean = numpy.mean (data_new)
```

```
# printing mean value after normalisation
print ("mean value after normalisation = %f" % new_mean)

# printing status
print ("Now, writing a file \"%s\"..." % file_output)

# writing normalised image into a file
hdu_new = astropy.io.fits.PrimaryHDU (data=data_new, header=header0)
hdu_new.writeto (file_output)

# closing FITS file
hdu_list.close ()

# printing status
print ("Finished writing a file \"%s\"!" % file_output)
```

Execute the script.

```
% chmod a+x advobs202202_s02_12.py
% ./advobs202202_s02_12.py -h
usage: advobs202202_s02_12.py [-h] [-o OUTPUT] fits

Normalising an image

positional arguments:
  fits                  input FITS file name

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output FITS file name

% ./advobs202202_s02_12.py -o 0456n.fits data_s02/lot_20210214_0456.fits
input file  = data_s02/lot_20210214_0456.fits
output file = 0456n.fits
mean value of data_s02/lot_20210214_0456.fits = 10564.770122
mean value after normalisation = 1.000000
Now, writing a file "0456n.fits"...
Finished writing a file "0456n.fits"!
% ls -lF 0456n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:19 0456n.fits
```

A new file "`0456n.fits`" is created. Examine the mean count of newly created file "`0456n.fits`".

```
% ./advobs202202_s02_11.py 0456n.fits
file                                    max      min      mean    median    stddev
-------------------------------------------------------------------------------
0456n.fits                             4.68     0.50      1.00      1.00      0.03
```

The mean value of the file "`0456n.fits`" is a unity.

12 The other way to create a new FITS file

Here is the other way to create a new FITS file. The function `astropy.io.fits.writeto` can also be used to make a new FITS file.

Python Code 13: ao2021_s02_09.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Normalising an image'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('fits', help='input FITS file name')
parser.add_argument ('-o', '--output', default='new.fits', \
                     help='output FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# calculations of mean value
data_mean = numpy.mean (data0)

# printing mean value
print ("mean value of %s = %f" % (file_input, data_mean) )

# normalisation
data_new = data0 / data_mean

# calculation of mean value after normalisation
new_mean = numpy.mean (data_new)

# printing mean value after normalisation
print ("mean value after normalisation = %f" % new_mean)

# printing status
```

```
print ("Now, writing a file \"%s\"..." % file_output)

# writing normalised image into a file
astropy.io.fits.writeto (file_output, data_new, header=header0)

# closing FITS file
hdu_list.close ()

# printing status
print ("Finished writing a file \"%s\"!" % file_output)
```

Try this script.

```
% chmod a+x advobs202202_s02_13.py
% ./advobs202202_s02_13.py -h
usage: advobs202202_s02_13.py [-h] [-o OUTPUT] fits

Normalising an image

positional arguments:
  fits                  input FITS file name

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output FITS file name

% ./advobs202202_s02_13.py -o 0456n2.fits data_s02/lot_20210214_0456.fits
input file  = data_s02/lot_20210214_0456.fits
output file = 0456n2.fits
mean value of data_s02/lot_20210214_0456.fits = 10564.770122
mean value after normalisation = 1.000000
Now, writing a file "0456n2.fits"...
Finished writing a file "0456n2.fits"!
% ls -lF 0456n*.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:19 0456n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:23 0456n2.fits
```

Check whether or not two files "`0456n.fits`" and "`0456n2.fits`" are identical. Use the command `diff`.

```
% diff 0456n*.fits
```

The file "`0456n2.fits`" is found to be identical to the file "`0456n.fits`".
Calculate statistical values of two files "`0456n.fits`" and "`0456n2.fits`".

```
% ./advobs202202_s02_11.py 0456n*.fits
file                                    max       min      mean    median    stddev
-------------------------------------------------------------------------------
0456n.fits                             4.68      0.50      1.00      1.00      0.03
0456n2.fits                            4.68      0.50      1.00      1.00      0.03
```

13 Adding new comments to a FITS file

After a processing of FITS files, we should leave some comments in the header of newly created FITS files. Here is a sample script to add new comments to a FITS file.

Python Code 14: advobs202202_s02_14.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse

# importing datetime module
import datetime

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# construction of parser object
desc = 'Normalising an image and adding comments in new file'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('fits', help='FITS file')
parser.add_argument ('-o', '--output', default='new.fits', \
                     help='new FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
file_input  = args.fits
file_output = args.output

print ("input file  =", file_input)
print ("output file =", file_output)

# date/time of now
datetime_now = datetime.datetime.utcnow ()
datetime_str = "%04d-%02d-%02dT%02d:%02d:%06.3f" \
    % (datetime_now.year, datetime_now.month, datetime_now.day, \
       datetime_now.hour, datetime_now.minute, \
       datetime_now.second + datetime_now.microsecond * 10**-6)

# opening FITS file
hdu_list = astropy.io.fits.open (file_input)

# primary HDU
hdu0 = hdu_list[0]

# header of primary HDU
header0 = hdu0.header

# data of primary HDU
data0 = hdu0.data

# calculations of statistical values
data_mean = numpy.mean (data0)

# printing mean value
print ("mean value of %s = %f" % (file_input, data_mean) )
```

```
# normalisation
data_new = data0 / data_mean

# calculation of mean value after normalisation
new_mean = numpy.mean (data_new)

# printing mean value after normalisation
print ("mean value after normalisation = %f" % new_mean)

# adding new comments to header
header0['history'] = "updated on %s" % datetime_str
header0['comment'] = "image was normalised"
header0['comment'] = "mean value before normalisation was %f" % data_mean

# printing status
print ("Now, writing a file \"%s\"..." % file_output)

# writing normalised image into a file
astropy.io.fits.writeto (file_output, data_new, header=header0)

# closing FITS file
hdu_list.close ()

# printing status
print ("Finished writing a file \"%s\"!" % file_output)
```

Try this script.

```
% chmod a+x advobs202202_s02_14.py
% ./advobs202202_s02_14.py -h
usage: advobs202202_s02_14.py [-h] [-o OUTPUT] fits

Normalising an image and adding comments in new file

positional arguments:
  fits                  FITS file

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        new FITS file name
% ./advobs202202_s02_14.py -o 0456n3.fits data_s02/lot_20210214_0456.fits
input file  = data_s02/lot_20210214_0456.fits
output file = 0456n3.fits
mean value of data_s02/lot_20210214_0456.fits = 10564.770122
mean value after normalisation = 1.000000
Now, writing a file "0456n3.fits"...
Finished writing a file "0456n3.fits"!
% ls -lF 0456n*.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:19 0456n.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:23 0456n2.fits
-rw-r--r--  1 daisuke  taiwan  33560640 Feb 24 21:41 0456n3.fits
```

Check the difference of "0456n2.fits" and "0456n3.fits".

```
% ./advobs202202_s02_11.py 0456n?.fits
file                                max       min      mean    median    stddev
```

```
-------------------------------------------------------------------------------
0456n2.fits                              4.68      0.50      1.00      1.00      0.03
0456n3.fits                              4.68      0.50      1.00      1.00      0.03
% diff 0456n?.fits
Binary files 0456n2.fits and 0456n3.fits differ
```

Statistical values of pixels are the same, but the file contents are different.
Use the script "advobs202202_s02_07.py" to print the header of files "0456n2.fits" and "0456n3.fits".

```
% ./advobs202202_s02_07.py 0456n2.fits > 0456n2.header
% ./advobs202202_s02_07.py 0456n3.fits > 0456n3.header
% ls -lF 0456n*.header
-rw-r--r--  1 daisuke  taiwan  4536 Feb 24 21:47 0456n2.header
-rw-r--r--  1 daisuke  taiwan  4779 Feb 24 21:47 0456n3.header
```

Compare the header of files "0452n2.fits" and "0452n3.fits".

```
% tail 0456n2.header
GAIN    = '2       '
SITEELEV=                 2862
RMSNOISE= '8.5     '
OPERATOR= 'lulin   '
CTYPE1  = 'RA---TAN' /           fastest changing axis name
CTYPE2  = 'DEC--TAN' /           slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '        '
SWOWNER = 'Ming-Hsin Chang' /   Licensed owner of software
% tail 0456n3.header
OPERATOR= 'lulin   '
CTYPE1  = 'RA---TAN' /           fastest changing axis name
CTYPE2  = 'DEC--TAN' /           slowest changing axis name
FOV     = '13'' 08" x 13'' 08"'
PIXSIZE =  0.39000000000000001
FLIPSTAT= '        '
SWOWNER = 'Ming-Hsin Chang' /   Licensed owner of software
HISTORY updated on 2022-02-24T13:41:34.699
COMMENT image was normalised
COMMENT mean value before normalisation was 10564.770122
% diff 0456n*.header
56a57,59
> HISTORY updated on 2022-02-24T13:41:34.699
> COMMENT image was normalised
> COMMENT mean value before normalisation was 10564.770122
```

14 Generating a synthetic image

Use Astropy to generate a synthetic image simulating sky background.

Python Code 15: advobs202202_s02_15.py

```
#!/usr/pkg/bin/python3.9

# importing argparse module
import argparse
```

```
# importing sys module
import sys

# importing datetime module
import datetime

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits

# constructing parser object
desc = 'generating a synthetic image'
parser = argparse.ArgumentParser (description=desc)

# adding command-line arguments
parser.add_argument ('-b', '--background', type=float, default=1000.0, \
                     help='background level (default: 1000)')
parser.add_argument ('-s', '--sigma', type=float, default=10.0, \
                     help='noise level (default: 10)')
parser.add_argument ('-x', '--xsize', type=int, default=512, \
                     help='image size in x-axis (default: 512)')
parser.add_argument ('-y', '--ysize', type=int, default=512, \
                     help='image size in y-axis (default: 512)')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
sky_background = args.background
noise          = args.sigma
image_size_x   = args.xsize
image_size_y   = args.ysize
file_output    = args.output

# if file name is not given, then stop the script
if (file_output == ''):
    print ("File name is not given.")
    print ("Use -o option to specify output file name.")
    print ("Exiting...")
    sys.exit ()

# image size
image_size = (image_size_x, image_size_y)

# date/time of now
datetime_now = datetime.datetime.utcnow ()
datetime_str = "%04d-%02d-%02dT%02d:%02d:%06.3f" \
    % (datetime_now.year, datetime_now.month, datetime_now.day, \
       datetime_now.hour, datetime_now.minute, \
       datetime_now.second + datetime_now.microsecond * 10**-6)

# initialisation of random number generator
rng = numpy.random.default_rng ()
# generation of numpy array of size (image_size_x, image_size_y)
array_image = rng.normal (loc=sky_background, scale=noise, size=image_size)
```

```
# making a header
header = astropy.io.fits.PrimaryHDU ().header
header['history'] = "generated on %s" % datetime_str
header['comment'] = "synthetic image generated by command \"%s\"" % sys.argv[0]
header['comment'] = "image generation parameters:"
header['comment'] = "  image size in X      = %d" % image_size_x
header['comment'] = "  image size in Y      = %d" % image_size_y
header['comment'] = "  mean sky background  = %f" % sky_background
header['comment'] = "  stddev of background = %f" % noise

# writing a FITS file
astropy.io.fits.writeto (file_output, array_image, header=header)
```

Execute the script.

```
% chmod a+x advobs202202_s02_15.py
% ./advobs202202_s02_15.py -h
usage: advobs202202_s02_15.py [-h] [-b BACKGROUND] [-s SIGMA] [-x XSIZE]
                              [-y YSIZE] [-o OUTPUT]

generating a synthetic image

optional arguments:
  -h, --help            show this help message and exit
  -b BACKGROUND, --background BACKGROUND
                        background level (default: 1000)
  -s SIGMA, --sigma SIGMA
                        noise level (default: 10)
  -x XSIZE, --xsize XSIZE
                        image size in x-axis (default: 512)
  -y YSIZE, --ysize YSIZE
                        image size in y-axis (default: 512)
  -o OUTPUT, --output OUTPUT
                        output file name

% ./advobs202202_s02_15.py -o skybg.fits -x 2048 -y 2048 -b 2000 -s 15
% ls -lF skybg.fits
-rw-r--r--  1 daisuke  taiwan  33557760 Feb 24 23:24 skybg.fits
```

Calculate statistical values of newly created file "`skybg.fits`".

```
% ./advobs202202_s02_11.py skybg.fits
file                                  max      min     mean   median   stddev
-------------------------------------------------------------------------------
skybg.fits                        2075.56  1924.47  2000.00  2000.00    15.00
```

Use your favourite FITS browser to visualise the file. Here is an example of using Ginga. (Fig. 9)

```
% ginga skybg.fits
```

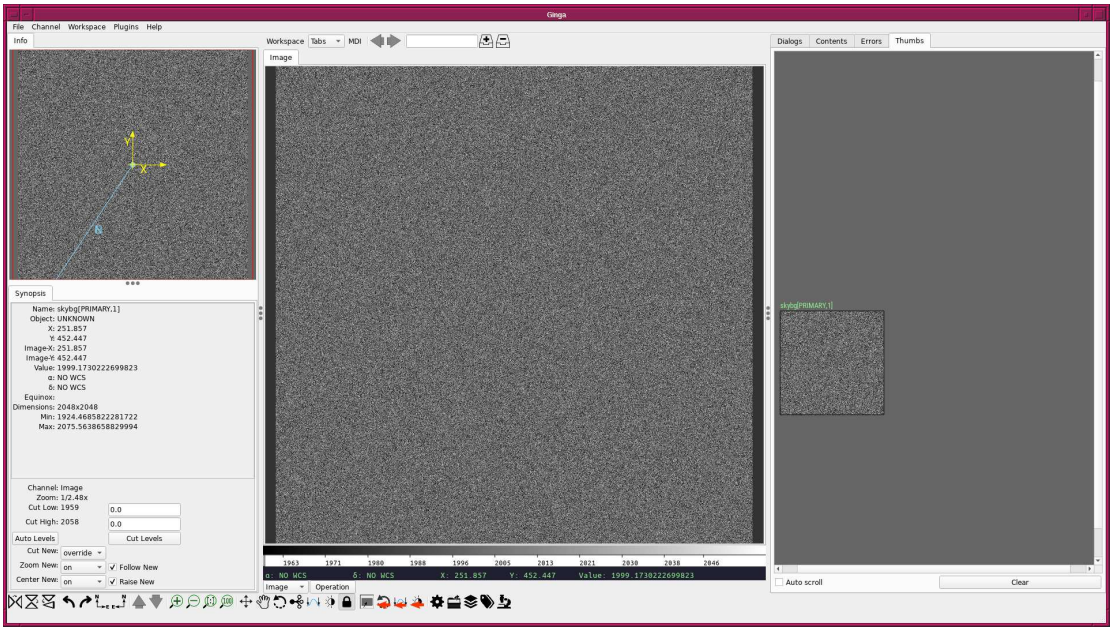

Figure 9: Visualisation of the file "`skybg.fits`" using FITS browser Ginga.

15 More about FITS

To know more about FITS, read following documentation.

- "FITS Documentation" page (Fig. 10)
  - `https://fits.gsfc.nasa.gov/fits_documentation.html`
- the section "FITS File Handling" of Astropy official document (Fig. 11)
  - `https://docs.astropy.org/en/stable/io/fits/index.html`

16 Exercises

1. Learn about FITS. Describe FITS. What is the overall design of FITS? What is the structure of a FITS file? What is an advantage of using FITS? How widely FITS is used in astronomy? Study about FITS, and give a write-up about FITS.
2. Make 3 Python scripts which use `astropy.io.fits` module. Describe the design of your Python scripts. Show the source code of your Python scripts. Take screenshots of your computer display to show the result of the execution of your Python scripts.
3. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.
   (a) What is the focal length of the telescope written in the header?
   (b) What is the pixel size of the CCD imager written in the header?
   (c) Make a Python script to calculate pixel scale.
   (d) What is the pixel scale written in the header? Compare the pixel scale written in the header to the one you calculated.
   (e) What is the pixel number of the CCD imager?
   (f) Make a Python script to calculate the field-of-view of the CCD imager.
4. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.

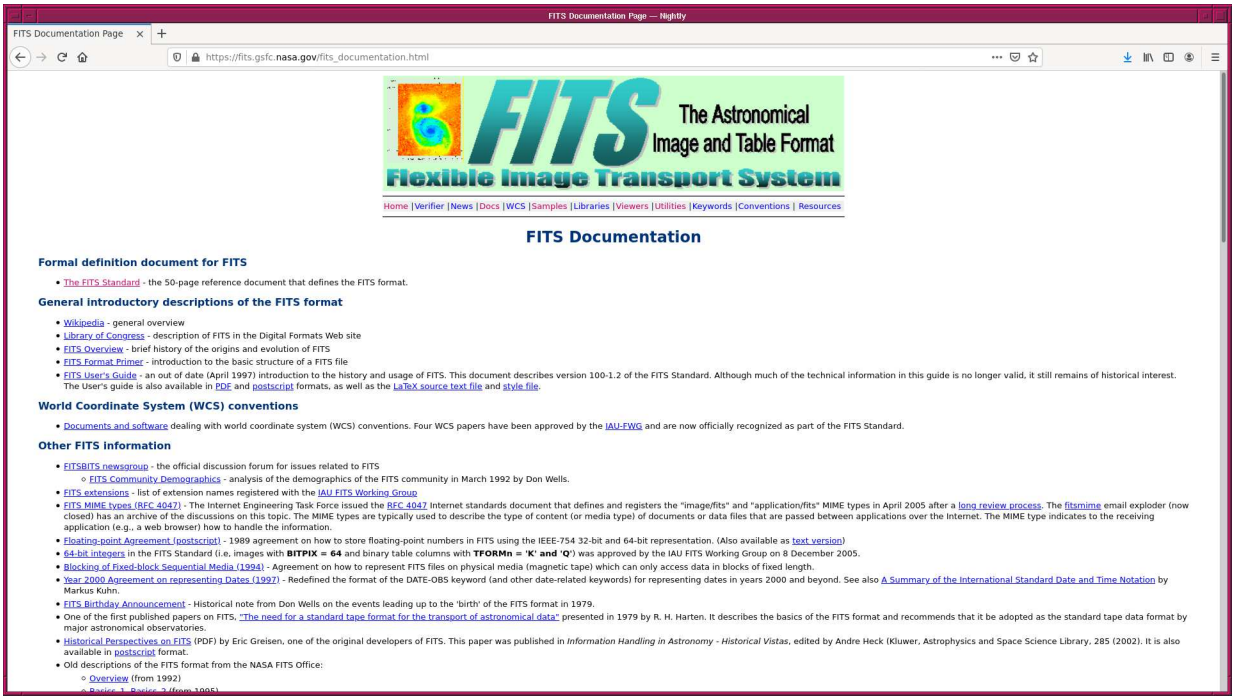

Figure 10: The FITS documentation web page.

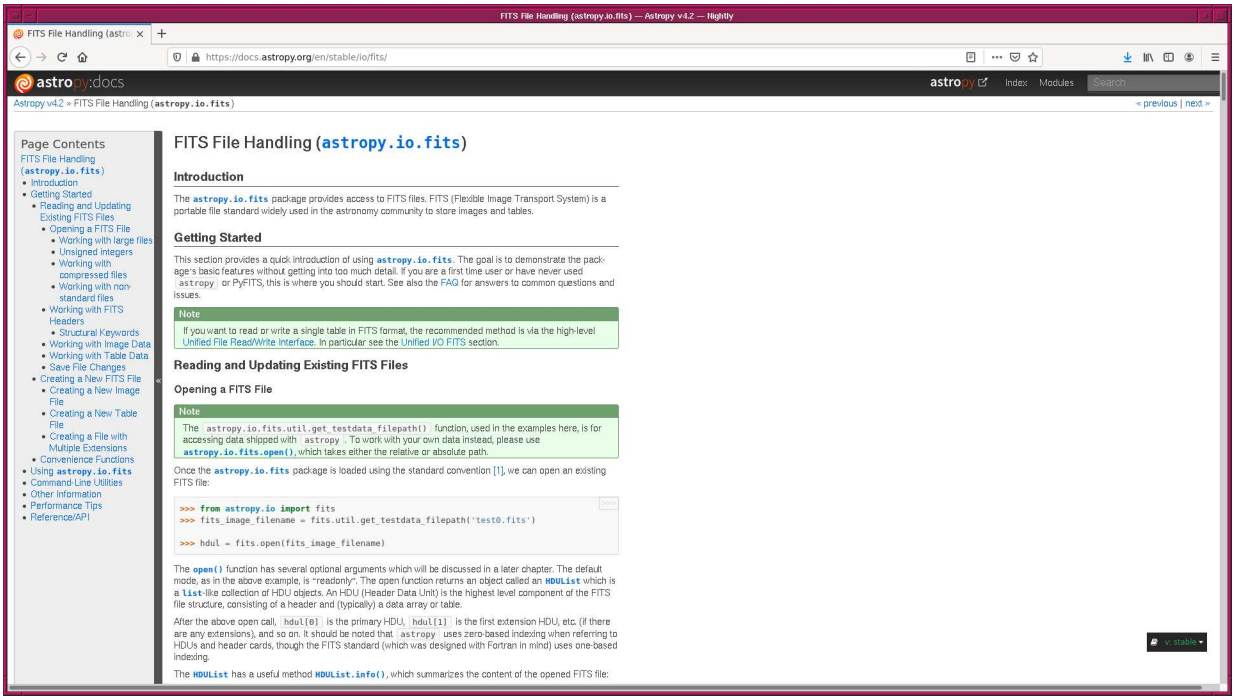

Figure 11: The FITS File Handling documentation of Astropy.

(a) What is the date/time of the middle of the exposure?

(b) Make a Python script to calculate JD corresponding the middle of the exposure (without using Astropy functions).

(c) What is the JD written in the header? Compare the JD written in the header to the one you calculated.

5. Choose one object frame from the data for this session. Read the header of the FITS file using `astropy.io.fits` module. Make Python scripts to do followings. Show source code of your Python scripts.

   (a) What is the altitude written in the header?

   (b) Make a Python script to calculate the airmass corresponding to the altitude written in the header.

   (c) What is the airmass written in the header? Compare the airmass written in the header to the one you calculated.

6. Choose one dark frame from the data for this session. Read the data using `astropy.io.fits` module. Calculate mean, median, and standard deviation of the image. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

7. Choose one bias frame from the data for this session. Read the data using `astropy.io.fits` module. Define four regions of 256×256 pixels in the image. Visualise locations of those four regions you selected. Make a Python script to calculate mean, median, and standard deviation of four regions you defined. Compare the values, and discuss the uniformity of the bias frame. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

8. Make a Python script to produce nicely formatted observing log. Think about which information should be included in the observing log, and design your own Python script. Execute the script, and generate the log. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

9. Visit the official website of WCSTools (Fig. 12). Install WCSTools on your computer. Summarise the installation process.

   - `http://tdc-www.harvard.edu/wcstools/`

   Read the description of "`imhead`" command which is included in WCSTools. Play with it. Make a Python script which works like `imhead`. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

   - `http://tdc-www.harvard.edu/software/wcstools/imhead.html`

10. Read the description of "`gethead`" command which is included in WCSTools. Play with it. Make a Python script which works like `gethead`. Describe the design of your Python script. Show the source code of your Python script. Take a screenshot of your computer display to show the result of the execution of your Python script.

    - `http://tdc-www.harvard.edu/software/wcstools/gethead/`

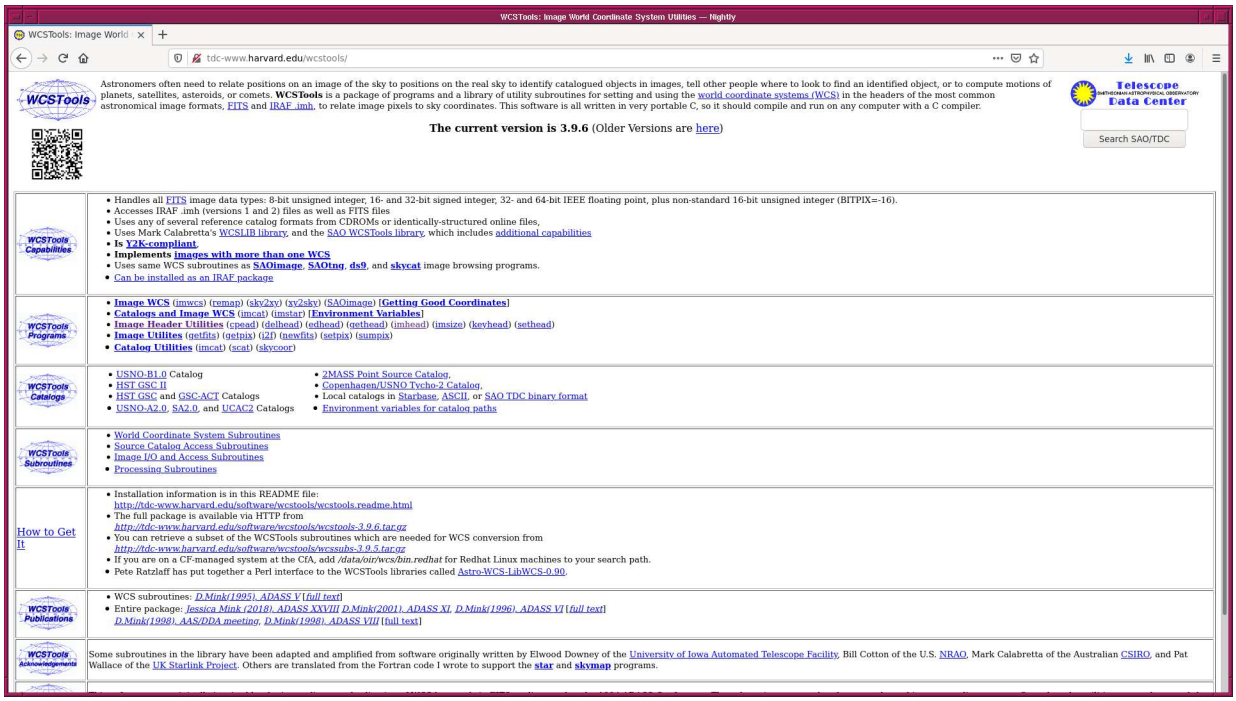

Figure 12: The official website of WCSTools.