Advanced Astronomical Observations 2022
Session 18: Preparation for an Observing Run

Kinoshita Daisuke

30 June 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 do some calculations needed for the preparation for an observing run and make plots and charts.

1 Astroquery package

The `astroquery` package provides a set of tools to query astronomical databases. It is a very useful package. Name resolvers, such as SIMBAD or NED, can be used via astroquery. Astronomical images can be downloaded from data archive servers, such as DSS and SDSS image archives, using `astroquery` package. Visit the official websites of `astroquery` package to learn about it.

- astroquery
  - `https://astroquery.readthedocs.io/` (Fig. 1)
  - `https://pypi.org/project/astroquery/` (Fig. 2)
  - `https://github.com/astropy/astroquery` (Fig. 3)

1.1 Installation

`Astroquery` package depends on following packages.

- numpy
- astropy
- pyVO
- requests

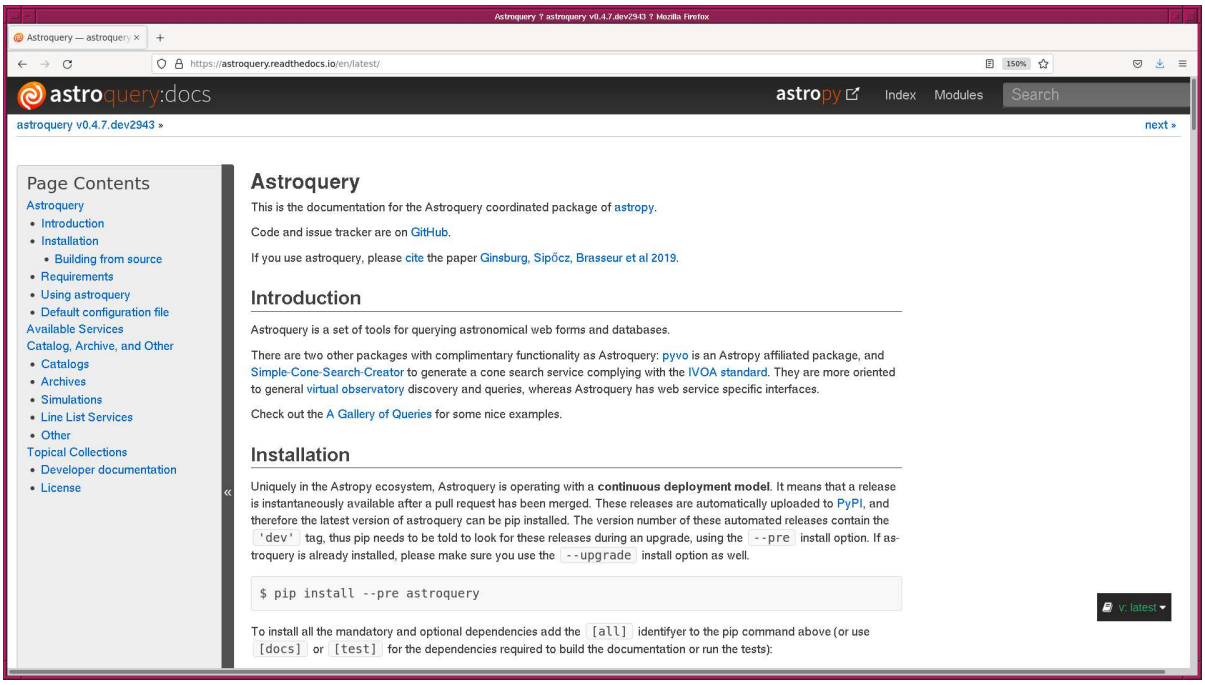

Figure 1: The official website of `astroquery` package at `readthedocs`.

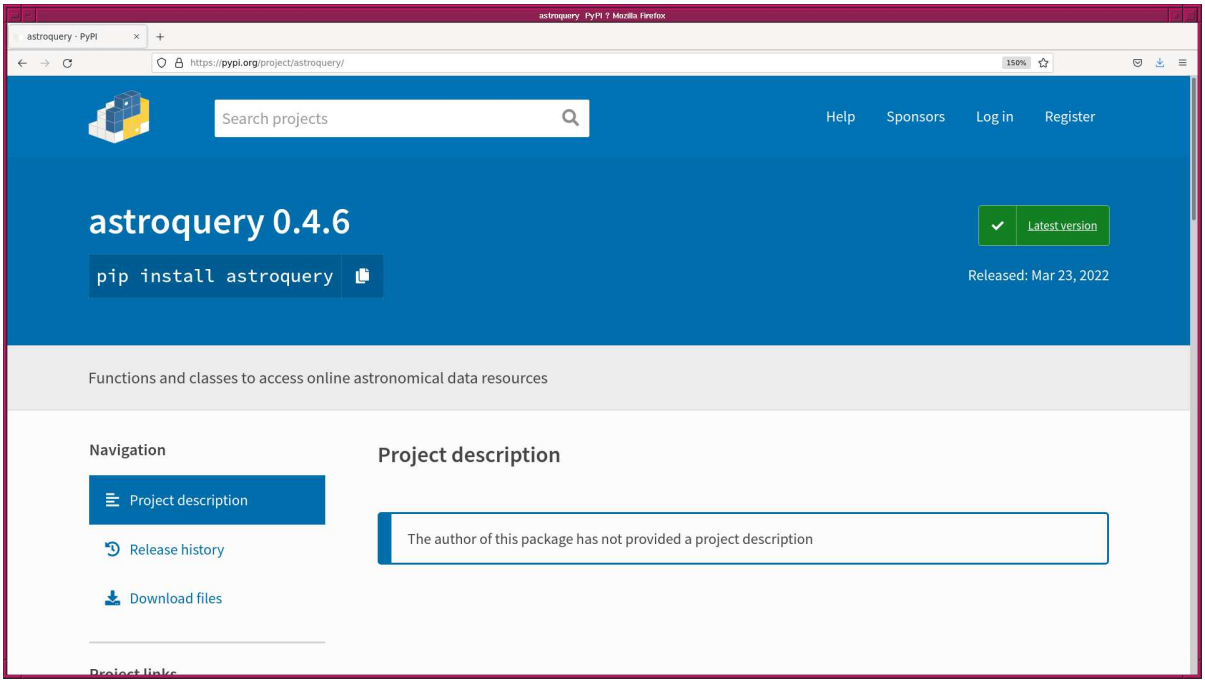

Figure 2: The official website of `astroquery` package at `pypi`.

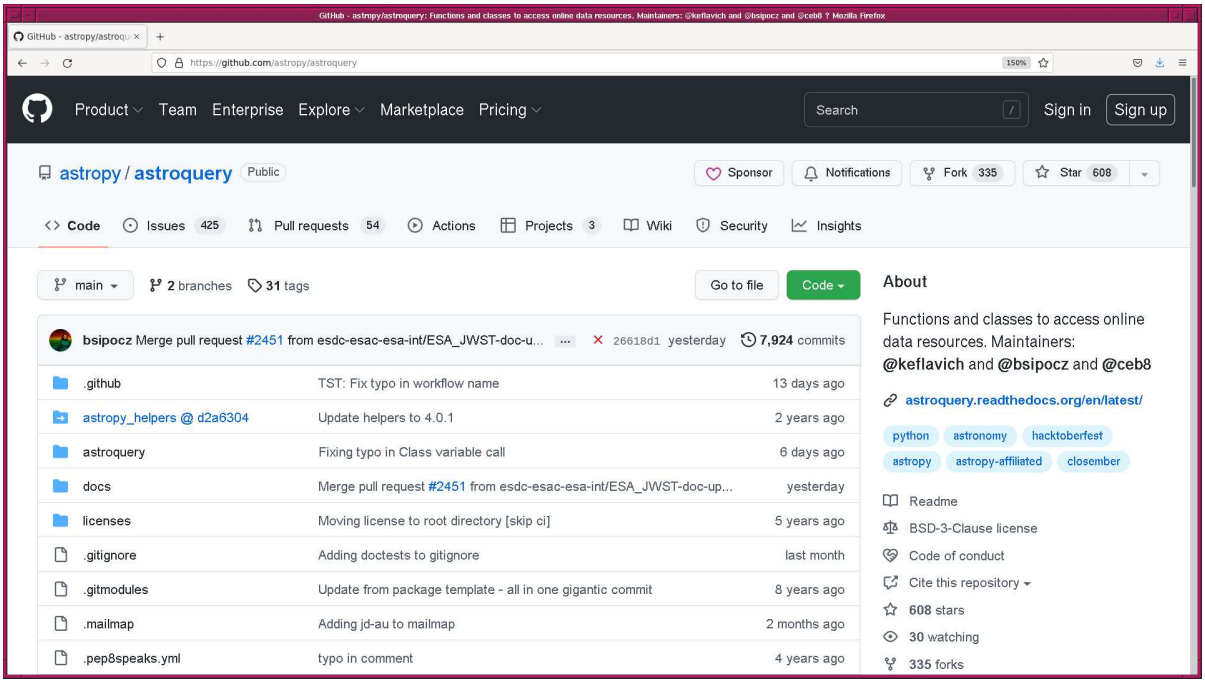

Figure 3: The official website of `astroquery` package at `github`.

- keyring
- Beautiful Soup
- html5lib
- six

Before starting to install `astroquery` package, make sure to install these packages on your computer. Then, read the installation instruction on the official website, and install `astroquery` package.

1.2 Importing astroquery package

Try to import `astroquery` package. If you have `astroquery` package properly installed on your computer, you can successfully import `astroquery` module. (Fig. 4)

```
% python3.9
Python 3.9.12 (main, Apr  4 2022, 09:16:48)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroquery
>>> exit ()
```

If you have not yet installed `astroquery` package, then you see an error message when trying to import `astroquery` package. (Fig. 5)

```
% python3.9
Python 3.9.12 (main, Apr  6 2022, 23:48:29)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroquery
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'astroquery'
>>> exit ()
```

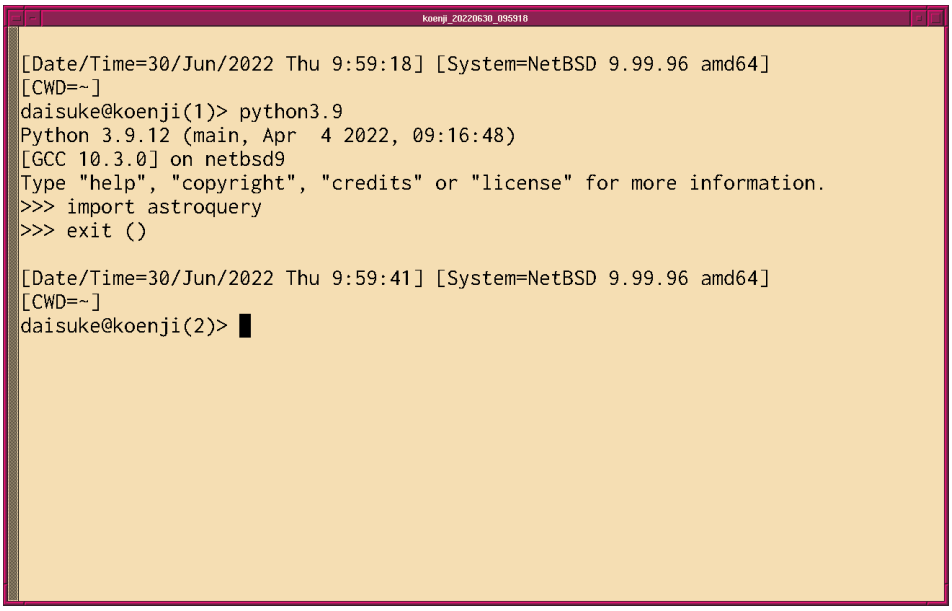

Figure 4: A successful import of `astroquery` module.

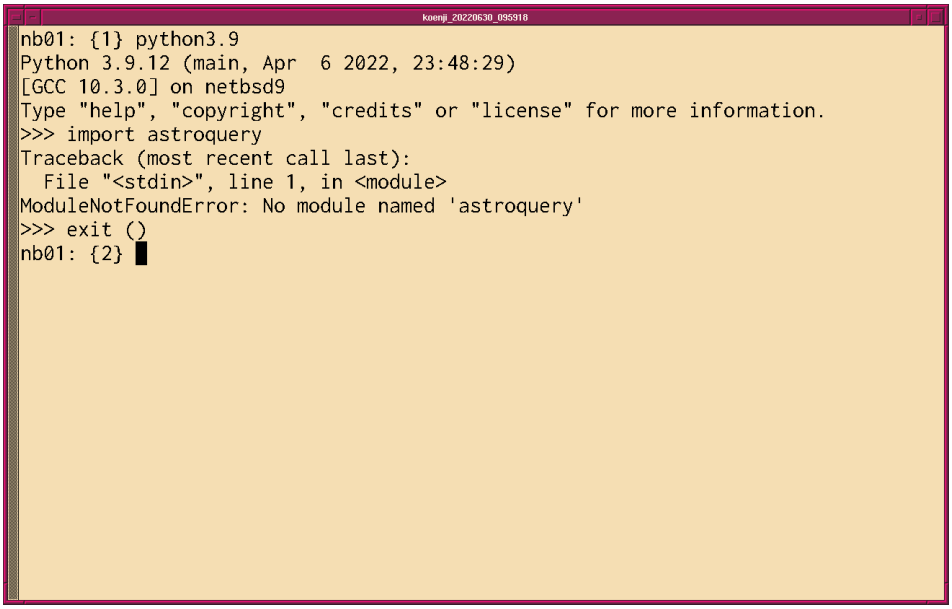

Figure 5: An unsuccessful import of `astroquery` module.

2 Astroplan package

The `astroplan` package is an useful Python package for planning observations. It is an `astropy` affiliated package. Visit the official websites of `astroplan` package and learn about it.

- astroplan
  - `https://astroplan.readthedocs.io/` (Fig. 6)
  - `https://pypi.org/project/astroplan/` (Fig. 7)
  - `https://github.com/astropy/astroplan` (Fig. 8)

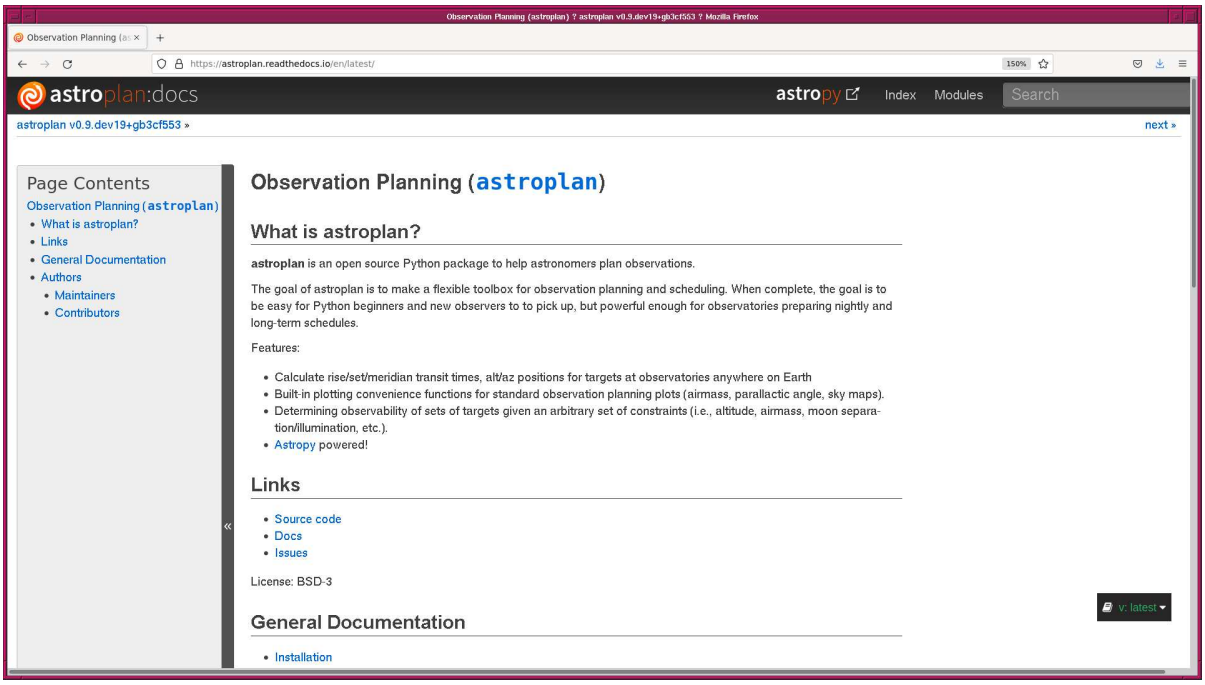

Figure 6: The official document of `astroplan` package at `readthedocs`.

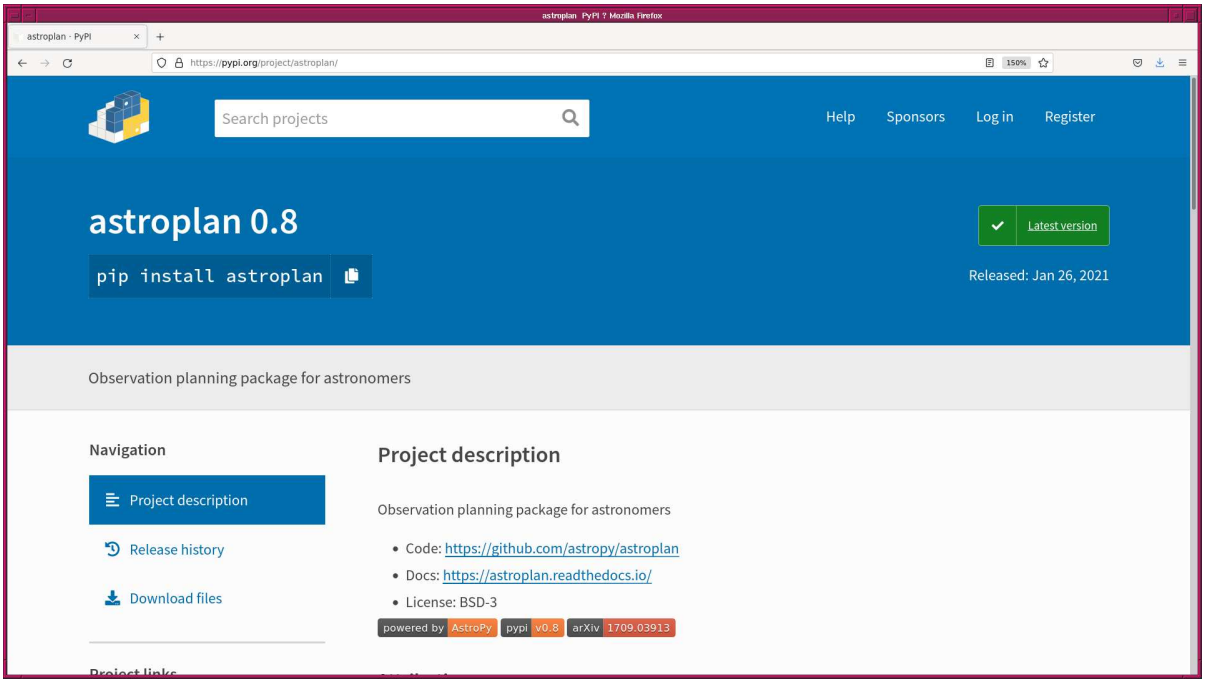

Figure 7: The official document of `astroplan` package at `pypi`.

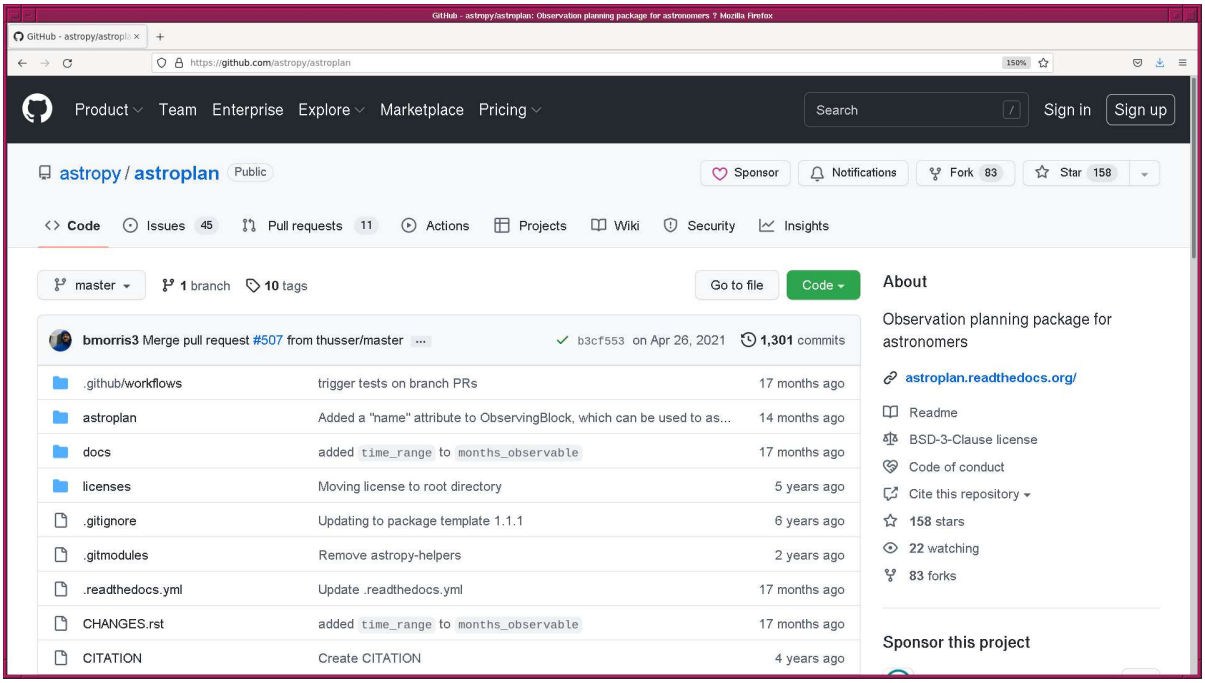

Figure 8: The official document of `astroplan` package at `github`.

2.1 Requirements

At the time of this writing (June 2022), here is a list of requirements for the installation of `astroplan`.

- Python 3.5 or newer
- Numpy 1.10 or newer
- astropy 1.3 or newer
- pytz

Also, following packages are recommended to be installed.

- Matplotlib
- astroquery

If your system does not fulfil these requirements, make sure to install required packages before starting the installation of `astroplan`.

2.2 Installation

Visit the following web page, read the document, and install `astroplan` package.

- `https://astroplan.readthedocs.io/en/latest/installation.html`

2.3 Importing astroplan package

After the installation, try to import `astroplan` package. Here is a successful example of importing `astroplan` package. (Fig. 9)

```
% python3.9
Python 3.9.12 (main, Apr  4 2022, 09:16:48)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroplan
>>> exit ()
```

If you have not yet installed **astroplan** package, you see an error message like below. (Fig. 10)

```
% python3.9
Python 3.9.12 (main, Apr  6 2022, 23:48:29)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroplan
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'astroplan'
>>> exit ()
```

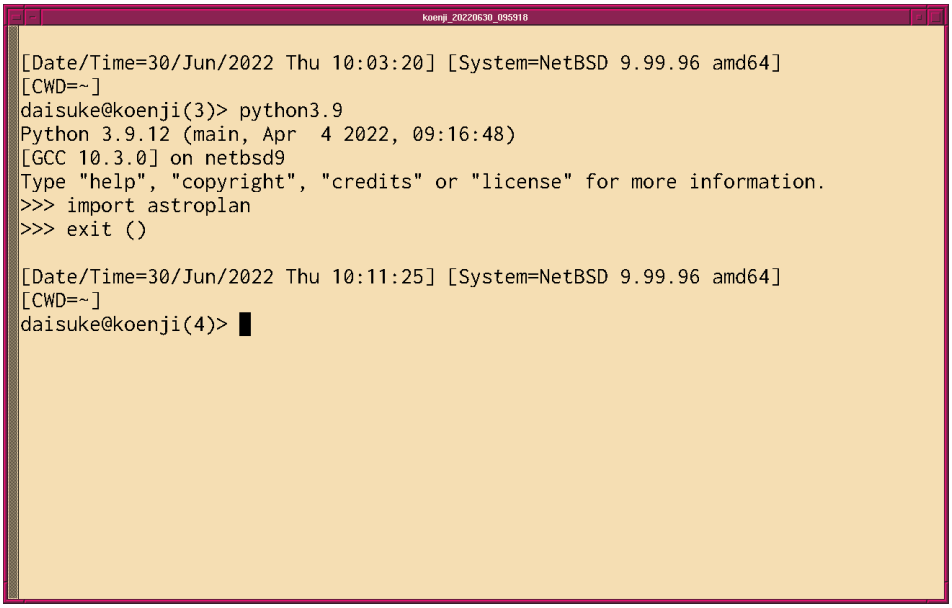

Figure 9: A successful import of **astroplan** module.

3 Using astroplan package

Make a simple Python script using **astroplan** package. Here is an example to examine whether or not it is night at given observing site and date/time.

Python Code 1: advobs202202_s18_01.py

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

#
# Time-stamp: <2022/06/30 10:43:27 (CST) daisuke>
#

# importing argparse module
import argparse

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

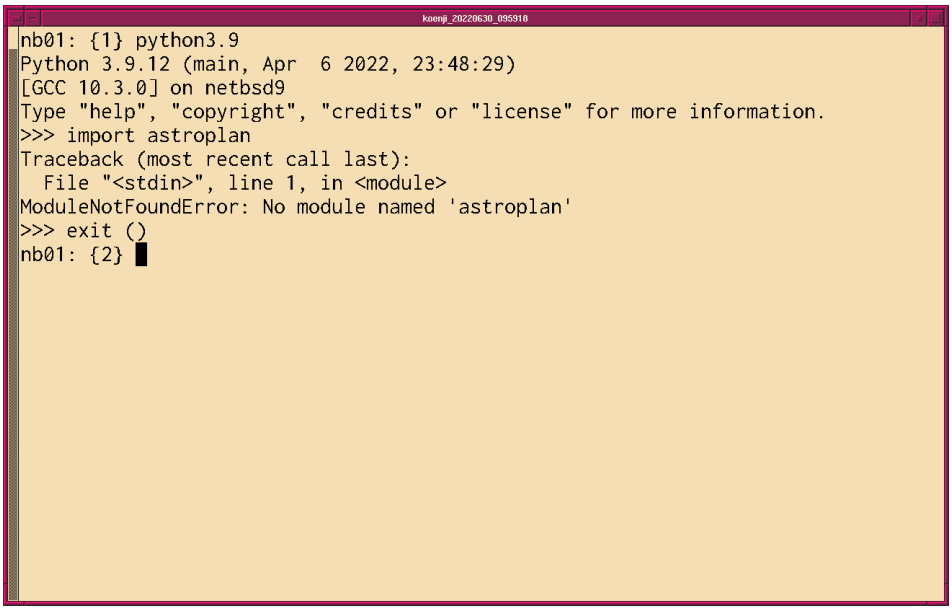

Figure 10: An unsuccessful import of `astroplan` module.

```
# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "daytime or nighttime at given location"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#    longitude =", site_lon)
print ("#    latitude  =", site_lat)
```

```
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# is night?
result_is_night = observer.is_night (datetime)

# printing results
print ("Is night?")
print ("  %s ==> %s" % (datetime, result_is_night) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_01.py
% ./advobs202202_s18_01.py -h
usage: advobs202202_s18_01.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

daytime or nighttime at given location

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_01.py
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2000-01-01T12:00:00.000
Downloading https://maia.usno.navy.mil/ser7/finals2000A.all
|=========================================| 3.4M/3.4M (100.00%)         8s
Is night?
  2000-01-01T12:00:00.000 ==> True
```

Is 2022-07-09T09:00:00 nighttime at Lulin Observatory?

```
% ./advobs202202_s18_01.py -t 2022-07-09T09:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T09:00:00
Is night?
  2022-07-09T09:00:00.000 ==> False
```

Is 2022-07-09T10:00:00 nighttime at Lulin Observatory?

```
% ./advobs202202_s18_01.py -t 2022-07-09T10:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T10:00:00
Is night?
  2022-07-09T10:00:00.000 ==> False
```

Is 2022-07-09T11:00:00 nighttime at Lulin Observatory?

```
% ./advobs202202_s18_01.py -t 2022-07-09T11:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T11:00:00
Is night?
  2022-07-09T11:00:00.000 ==> True
```

Is 2022-07-09T12:00:00 nighttime at Lulin Observatory?

```
% ./advobs202202_s18_01.py -t 2022-07-09T12:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T12:00:00
Is night?
  2022-07-09T12:00:00.000 ==> True
```

Next, check the visibility of targets.

Python Code 2: advobs202202_s18_02.py

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

#
# Time-stamp: <2022/06/30 10:57:34 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "visibility check at given location"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')
parser.add_argument ('target', nargs='+', default='Vega', \
                     help='names of targets')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime
list_target  = args.target

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#    longitude =", site_lon)
print ("#    latitude  =", site_lat)
print ("#    altitude  =", site_alt)
print ("#  Date/Time:")
print ("#    date/time (UT) =", datetime_str)
```

```
print ("#  Targets:")
for target in list_target:
    print ("#   %s" % target)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# processing target one-by-one
print ("Is visible?")
for target in list_target:
    # getting object from name
    obj = astroplan.FixedTarget.from_name (target)
    # checking visibility of the object
    visibility = observer.target_is_up (datetime, obj)
    # printing result
    print ("%-16s ==> %s" % (target, visibility) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_02.py
% ./advobs202202_s18_02.py -h
usage: advobs202202_s18_02.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]
                              target [target ...]

visibility check at given location

positional arguments:
  target                names of targets

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_02.py -t 2022-07-09T16:00:00 Vega Antares Fomalhaut \
? Canopus Sirius Spica
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
```

```
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
#  Targets:
#   Vega
#   Antares
#   Fomalhaut
#   Canopus
#   Sirius
#   Spica
Is visible?
Vega              ==> True
Antares           ==> True
Fomalhaut         ==> True
Canopus           ==> False
Sirius            ==> False
Spica             ==> False
```

4 The Sun

4.1 Position of the Sun

Show the position of the Sun for given location and date/time.

Python Code 3: advobs202202_s18_03.py

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

#
# Time-stamp: <2022/06/30 11:06:18 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "position of the Sun in (RA, Dec) at given location and time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
```

```
                    help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                    help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# using DE430
astropy.coordinates.solar_system_ephemeris.set ('de430')

# position of the Sun
sun = astropy.coordinates.get_body ('sun', datetime, location)
(sun_ra, sun_dec) = sun.to_string ('hmsdms').split ()

# printing position of the Sun
print ("Sun:")
print ("  RA:  %s" % sun_ra)
print ("  Dec: %s" % sun_dec)
```

Execute the script. It may take a while for downloading DE430.

```
% chmod a+x advobs202202_s18_03.py
% ./advobs202202_s18_03.py -h
usage: advobs202202_s18_03.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

position of the Sun in (RA, Dec) at given location and time

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
```

```
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_03.py -t 2022-07-09T04:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T04:00:00
Downloading https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/de430.bsp
|=========================================| 119M/119M (100.00%)      1m25s
Downloading https://maia.usno.navy.mil/ser7/finals2000A.all
|=========================================| 3.4M/3.4M (100.00%)         7s
Sun:
  RA:  07h12m12.83979624s
  Dec: +22d23m56.96617608s
```

4.2 Coordinate conversion

Carry out coordinate conversions for the position of the Sun.

Python Code 4: advobs202202_s18_04.py

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

#
# Time-stamp: <2022/06/30 11:10:57 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "position of the Sun in (az, alt) at given location and time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
```

```
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# using DE430
astropy.coordinates.solar_system_ephemeris.set ('de430')

# position of the Sun
sun = astropy.coordinates.get_body ('sun', datetime, location)
(sun_ra, sun_dec) = sun.to_string ('hmsdms').split ()

# conversion from equatorial into ecliptic
sun_ecliptic = sun.transform_to \
    (astropy.coordinates.GeocentricMeanEcliptic (obstime=datetime) )
sun_lambda = sun_ecliptic.lon
sun_beta   = sun_ecliptic.lat

# conversion from equatorial into horizontal
sun_altaz = sun.transform_to \
    (astropy.coordinates.AltAz (obstime=datetime, location=location) )
sun_alt = sun_altaz.alt
sun_az  = sun_altaz.az

# printing position of the Sun
print ("Sun:")
print ("  Equatorial: RA=%s, Dec=%s" % (sun_ra, sun_dec) )
print ("  Ecliptic:   lambda=%s, beta=%s" % (sun_lambda, sun_beta) )
print ("  AltAz:      Az=%s, Alt=%s" % (sun_az, sun_alt) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_04.py
% ./advobs202202_s18_04.py -h
usage: advobs202202_s18_04.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

position of the Sun in (az, alt) at given location and time

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_04.py -t 2022-07-09T04:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T04:00:00
Sun:
  Equatorial: RA=07h12m12.83979624s, Dec=+22d23m56.96617608s
  Ecliptic:   lambda=106d38m59.31890045s, beta=-0d00m09.50597037s
  AltAz:      Az=159d58m57.58735649s, Alt=88d49m22.34686277s
% ./advobs202202_s18_04.py -t 2022-07-09T10:42:50
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T10:42:50
Sun:
  Equatorial: RA=07h13m20.92371341s, Dec=+22d21m54.23595161s
  Ecliptic:   lambda=106d54m59.28856789s, beta=-0d00m09.52482535s
  AltAz:      Az=294d27m49.19995488s, Alt=0d00m18.05136864s
```

4.3 Sunset and sunrise

Calculate sunset and sunrise time.

Python Code 5: advobs202202_s18_05.py

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

#
```

```
# Time-stamp: <2022/06/30 11:21:19 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding sunset/sunrise time at given location nearest to given time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")
```

```
# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# sunset
sunset = observer.sun_set_time (datetime, which="nearest")

# sunrise
sunrise = observer.sun_rise_time (datetime, which="nearest")

# printing results
print ("sunset time (UT) nearest to %s (UT)" % datetime)
print ("  JD = %s" % sunset)
print ("     = %s" % sunset.isot)
print ("sunrise time (UT) nearest to %s (UT)" % datetime)
print ("  JD = %s" % sunrise)
print ("     = %s" % sunrise.isot)
```

Execute the script. Find sunset and sunrise time.

```
% chmod a+x advobs202202_s18_05.py
% ./advobs202202_s18_05.py -h
usage: advobs202202_s18_05.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding sunset/sunrise time at given location nearest to given time

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_05.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#    longitude = +120d52m25s
#    latitude  = +23d28m07s
#    altitude  = 2862.0 m
#  Date/Time:
#    date/time (UT) = 2022-07-09T16:00:00
sunset time (UT) nearest to 2022-07-09T16:00:00.000 (UT)
  JD = 2459769.946432798
     = 2022-07-09T10:42:51.794
sunrise time (UT) nearest to 2022-07-09T16:00:00.000 (UT)
  JD = 2459770.3894996163
     = 2022-07-09T21:20:52.767
```

Show the nighttime length at given location and date/time.

Python Code 6: advobs202202_s18_06.py

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

#
# Time-stamp: <2022/06/30 11:34:47 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding length of night"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)
```

```
# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# sunset
sunset = observer.sun_set_time (datetime, which="nearest")

# sunrise
sunrise = observer.sun_rise_time (datetime, which="nearest")

# length of night
length_night = sunrise.mjd - sunset.mjd

# printing results
print ("sunset time (UT) nearest to %s (UT)" % datetime)
print ("  JD = %s" % sunset)
print ("     = %s" % sunset.isot)
print ("sunrise time (UT) nearest to %s (UT)" % datetime)
print ("  JD = %s" % sunrise)
print ("     = %s" % sunrise.isot)
print ("length of night on %s (UT)" % datetime)
print ("  length = %6.4f [day]" % length_night)
print ("         = %5.2f [hour]" % (length_night * 24.0) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_06.py
% ./advobs202202_s18_06.py -h
usage: advobs202202_s18_06.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding length of night

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_06.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
sunset time (UT) nearest to 2022-07-09T16:00:00.000 (UT)
```

```
  JD = 2459769.946432798
     = 2022-07-09T10:42:51.794
sunrise time (UT) nearest to 2022-07-09T16:00:00.000 (UT)
  JD = 2459770.3894996163
     = 2022-07-09T21:20:52.767
length of night on 2022-07-09T16:00:00.000 (UT)
  length = 0.4431 [day]
         = 10.63 [hour]
```

Do the same thing, but for The Royal Observatory Greenwich in UK.

```
% ./advobs202202_s18_06.py -t 2022-07-09T23:00:00 -l +00d00m02s -b +51d28m37s -a 47
#
# input parameters
#
#  Location:
#   longitude = +00d00m02s
#   latitude  = +51d28m37s
#   altitude  = 47.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T23:00:00
sunset time (UT) nearest to 2022-07-09T23:00:00.000 (UT)
  JD = 2459770.339749962
     = 2022-07-09T20:09:14.397
sunrise time (UT) nearest to 2022-07-09T23:00:00.000 (UT)
  JD = 2459770.6678866604
     = 2022-07-10T04:01:45.407
length of night on 2022-07-09T23:00:00.000 (UT)
  length = 0.3281 [day]
         =  7.88 [hour]
```

4.4 Twilight

Calculate the time of end of evening twilight and start of morning twilight.

Python Code 7: advobs202202_s18_08.py

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

#
# Time-stamp: <2022/06/30 11:36:40 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m
```

```
# constructing parser object
desc   = "finding time of twilight"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#    longitude =", site_lon)
print ("#    latitude  =", site_lat)
print ("#    altitude  =", site_alt)
print ("#  Date/Time:")
print ("#    date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# end of twilight
twilight_civil_end        = observer.twilight_evening_civil \
    (datetime, which="nearest")
twilight_nautical_end     = observer.twilight_evening_nautical \
    (datetime, which="nearest")
twilight_astronomical_end = observer.twilight_evening_astronomical \
    (datetime, which="nearest")

# start of twilight
twilight_civil_start        = observer.twilight_morning_civil \
    (datetime, which="nearest")
twilight_nautical_start     = observer.twilight_morning_nautical \
    (datetime, which="nearest")
twilight_astronomical_start = observer.twilight_morning_astronomical \
```

```
    (datetime, which="nearest")

# printing results
print ("end of evening twilight nearest to %s" % datetime)
print ("  civil twilight:        %s" % twilight_civil_end.isot)
print ("  nautical twilight:     %s" % twilight_nautical_end.isot)
print ("  astronomical twilight: %s" % twilight_astronomical_end.isot)
print ("start of morning twilight nearest to %s" % datetime)
print ("  civil twilight:        %s" % twilight_civil_start.isot)
print ("  nautical twilight:     %s" % twilight_nautical_start.isot)
print ("  astronomical twilight: %s" % twilight_astronomical_start.isot)
```

Execute the script.

```
% chmod a+x advobs202202_s18_07.py
% ./advobs202202_s18_07.py -h
usage: advobs202202_s18_07.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding time of twilight

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_07.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
end of evening twilight nearest to 2022-07-09T16:00:00.000
  civil twilight:        2022-07-09T11:11:58.073
  nautical twilight:     2022-07-09T11:41:57.633
  astronomical twilight: 2022-07-09T12:13:11.581
start of morning twilight nearest to 2022-07-09T16:00:00.000
  civil twilight:        2022-07-09T20:51:46.733
  nautical twilight:     2022-07-09T20:21:47.831
  astronomical twilight: 2022-07-09T19:50:34.184
```

Show the length of observable time.

Python Code 8: advobs202202_s18_08.py

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

#
```

```
# Time-stamp: <2022/06/30 11:41:21 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding length of observable time on a night"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")
```

```
# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# end of twilight
twilight_civil_end        = observer.twilight_evening_civil \
    (datetime, which="nearest")
twilight_nautical_end     = observer.twilight_evening_nautical \
    (datetime, which="nearest")
twilight_astronomical_end = observer.twilight_evening_astronomical \
    (datetime, which="nearest")

# start of twilight
twilight_civil_start        = observer.twilight_morning_civil \
    (datetime, which="nearest")
twilight_nautical_start     = observer.twilight_morning_nautical \
    (datetime, which="nearest")
twilight_astronomical_start = observer.twilight_morning_astronomical \
    (datetime, which="nearest")

# printing results
print ("end of evening twilight nearest to %s" % datetime)
print ("  civil twilight:        %s" % twilight_civil_end.isot)
print ("  nautical twilight:     %s" % twilight_nautical_end.isot)
print ("  astronomical twilight: %s" % twilight_astronomical_end.isot)
print ("start of morning twilight nearest to %s" % datetime)
print ("  civil twilight:        %s" % twilight_civil_start.isot)
print ("  nautical twilight:     %s" % twilight_nautical_start.isot)
print ("  astronomical twilight: %s" % twilight_astronomical_start.isot)

# calculating observable time
obs_time = twilight_astronomical_start.mjd - twilight_astronomical_end.mjd

print ("length of observable time on %s" % datetime)
print ("  observable time = %6.4f [day]" % obs_time)
print ("                  = %5.2f [hour]" % (obs_time * 24.0) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_08.py
% ./advobs202202_s18_08.py -h
usage: advobs202202_s18_08.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding length of observable time on a night

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_08.py -t 2022-07-09T16:00:00
#
```

```
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
end of evening twilight nearest to 2022-07-09T16:00:00.000
  civil twilight:        2022-07-09T11:11:58.073
  nautical twilight:     2022-07-09T11:41:57.633
  astronomical twilight: 2022-07-09T12:13:11.581
start of morning twilight nearest to 2022-07-09T16:00:00.000
  civil twilight:        2022-07-09T20:51:46.733
  nautical twilight:     2022-07-09T20:21:47.831
  astronomical twilight: 2022-07-09T19:50:34.184
length of observable time on 2022-07-09T16:00:00.000
  observable time = 0.3176 [day]
                  =  7.62 [hour]
```

Do the same thing, but for the Royal Observatory Greenwich in UK.

```
% ./advobs202202_s18_08.py -t 2022-07-09T23:00:00 -l +00d00m02s -b +51d28m37s -a 47
#
# input parameters
#
#  Location:
#   longitude = +00d00m02s
#   latitude  = +51d28m37s
#   altitude  = 47.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T23:00:00
WARNING: TargetAlwaysUpWarning: Target with index 0 does not cross horizon=-18.0
 deg within 24 hours [astroplan.observer]
/usr/pkg/lib/python3.9/site-packages/erfa/core.py:154: ErfaWarning: ERFA functio
n "utctai" yielded 1 of "dubious year (Note 3)"
  warnings.warn('ERFA function "{}" yielded {}'.format(func_name, wmsg),
end of evening twilight nearest to 2022-07-09T23:00:00.000
  civil twilight:        2022-07-09T21:01:44.860
  nautical twilight:     2022-07-09T22:09:36.522
/usr/pkg/lib/python3.9/site-packages/erfa/core.py:154: ErfaWarning: ERFA functio
n "d2dtf" yielded 1 of "dubious year (Note 5)"
  warnings.warn('ERFA function "{}" yielded {}'.format(func_name, wmsg),
  astronomical twilight: --
start of morning twilight nearest to 2022-07-09T23:00:00.000
  civil twilight:        2022-07-10T03:09:14.281
  nautical twilight:     2022-07-10T02:01:21.896
  astronomical twilight: --
length of observable time on 2022-07-09T23:00:00.000
/amd/ogikubo/root/volume1/nas0/daisuke/backup/daisuke/tex/astro/NCU/Lecture/AdvO
bs_202202/s18/script_18/./advobs202202_s18_08.py:96: UserWarning: Warning: conve
rting a masked element to nan.
  print ("  observable time = %6.4f [day]" % obs_time)
  observable time =    nan [day]
/amd/ogikubo/root/volume1/nas0/daisuke/backup/daisuke/tex/astro/NCU/Lecture/AdvO
bs_202202/s18/script_18/./advobs202202_s18_08.py:97: UserWarning: Warning: conve
rting a masked element to nan.
  print ("                  = %5.2f [hour]" % (obs_time * 24.0) )
```

```
                  =   nan [hour]
```

At the time around summer solstice in London, the Sun never goes to elevation below -18 degrees.
Find the length of observable time at Greenwich Observatory on 01/Aug/2022.

```
% ./advobs202202_s18_08.py -t 2022-08-01T23:00:00 -l +00d00m02s -b +51d28m37s -a 47
#
# input parameters
#
#  Location:
#   longitude = +00d00m02s
#   latitude  = +51d28m37s
#   altitude  = 47.0 m
#  Date/Time:
#   date/time (UT) = 2022-08-01T23:00:00
end of evening twilight nearest to 2022-08-01T23:00:00.000
  civil twilight:        2022-08-01T20:28:25.551
  nautical twilight:     2022-08-01T21:22:28.167
  astronomical twilight: 2022-08-01T22:37:14.421
start of morning twilight nearest to 2022-08-01T23:00:00.000
  civil twilight:        2022-08-02T03:44:47.999
  nautical twilight:     2022-08-02T02:50:44.266
  astronomical twilight: 2022-08-02T01:36:00.653
length of observable time on 2022-08-01T23:00:00.000
  observable time = 0.1241 [day]
                  =  2.98 [hour]
```

5 The Moon

5.1 Position of the Moon

Calculate the position of the Moon.

Python Code 9: advobs202202_s18_09.py

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

#
# Time-stamp: <2022/06/30 11:53:41 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding position of the Moon"
```

```
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# using DE430
astropy.coordinates.solar_system_ephemeris.set ('de430')

# position of the Moon
moon = astropy.coordinates.get_body ('moon', datetime, location)
(moon_ra, moon_dec) = moon.to_string ('hmsdms').split ()

# printing position of the Moon
print ("Moon:")
print ("  RA:  %s" % moon_ra)
print ("  Dec: %s" % moon_dec)
```

Execute the script.

```
% chmod a+x advobs202202_s18_09.py
```

```
% ./advobs202202_s18_09.py -h
usage: advobs202202_s18_09.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding position of the Moon

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_09.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
Moon:
  RA:  15h06m04.55471244s
  Dec: -18d06m52.05949874s
```

Try coordinate conversion.

Python Code 10: advobs202202_s18_10.py

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

#
# Time-stamp: <2022/06/30 11:54:29 (CST) daisuke>
#

# importing argparse module
import argparse

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding position of the Moon in equatorial, ecliptic, and altaz"
parser = argparse.ArgumentParser (description=desc)
```

```
# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# using DE430
astropy.coordinates.solar_system_ephemeris.set ('de430')

# position of the Moon
moon = astropy.coordinates.get_body ('moon', datetime, location)
(moon_ra, moon_dec) = moon.to_string ('hmsdms').split ()

# conversion from equatorial into ecliptic
moon_ecliptic = moon.transform_to \
    (astropy.coordinates.GeocentricMeanEcliptic (obstime=datetime) )
moon_lambda = moon_ecliptic.lon
moon_beta   = moon_ecliptic.lat

# conversion from equatorial into horizontal
moon_altaz = moon.transform_to \
    (astropy.coordinates.AltAz (obstime=datetime, location=location) )
moon_alt = moon_altaz.alt
moon_az  = moon_altaz.az
```

```
# printing position of the Moon
print ("Moon:")
print ("  Equatorial: RA=%s, Dec=%s" % (moon_ra, moon_dec) )
print ("  Ecliptic:   lambda=%s, beta=%s" % (moon_lambda, moon_beta) )
print ("  AltAz:      Az=%s, Alt=%s" % (moon_az, moon_alt) )
```

Run the script.

```
% chmod a+x advobs202202_s18_10.py
% ./advobs202202_s18_10.py -h
usage: advobs202202_s18_10.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding position of the Moon in equatorial, ecliptic, and altaz

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_10.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
Moon:
  Equatorial: RA=15h06m04.55471244s, Dec=-18d06m52.05949874s
  Ecliptic:   lambda=229d47m03.51385819s, beta=0d04m47.20774424s
  AltAz:      Az=240d50m09.82846017s, Alt=16d51m10.13271681s
```

5.2 Moonrise and moonset

Calculate the time of moonrise and moonset.

Python Code 11: advobs202202_s18_11.py

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

#
# Time-stamp: <2022/06/30 12:05:32 (CST) daisuke>
#

# importing argparse module
import argparse

# importing astropy module
```

```
import astropy.units
import astropy.time
import astropy.coordinates

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m

# constructing parser object
desc   = "finding moonrise/moonset time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# using DE430
astropy.coordinates.solar_system_ephemeris.set ('de430')

# moonrise
```

```
moonrise = observer.moon_rise_time (datetime, which="nearest")

# moonset
moonset = observer.moon_set_time (datetime, which="nearest")

# printing results
print ("moonrise time nearest to %s" % datetime)
print ("   JD = %s" % moonrise)
print ("      = %s" % moonrise.isot)
print ("moonset time nearest to %s" % datetime)
print ("   JD = %s" % moonset)
print ("      = %s" % moonset.isot)
```

Execute the script.

```
% chmod a+x advobs202202_s18_11.py
% ./advobs202202_s18_11.py -h
usage: advobs202202_s18_11.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding moonrise/moonset time

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_11.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
moonrise time nearest to 2022-07-09T16:00:00.000
  JD = 2459769.755694353
     = 2022-07-09T06:08:11.992
moonset time nearest to 2022-07-09T16:00:00.000
  JD = 2459770.2242506677
     = 2022-07-09T17:22:55.258
```

5.3 Moon phase

Calculate the moon phase.

Python Code 12: advobs202202_s18_12.py

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

```
#
# Time-stamp: <2022/06/30 12:15:47 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m
unit_rad = astropy.units.rad
unit_deg = astropy.units.deg

# constructing parser object
desc   = "finding moonrise/moonset time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#    longitude =", site_lon)
print ("#    latitude  =", site_lat)
print ("#    altitude  =", site_alt)
print ("#  Date/Time:")
print ("#    date/time (UT) =", datetime_str)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
```

```
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# illumination
illumination = observer.moon_illumination (datetime)

# moon phase
phase = observer.moon_phase (datetime)

print ("Moon on %s" % datetime)
print ("  illumination = %f" % illumination)
print ("  moon phase = %s = %s" \
       % (phase, phase / numpy.pi * 180.0 / unit_rad * unit_deg) )
```

Execute the script.

```
% chmod a+x advobs202202_s18_12.py
% ./advobs202202_s18_12.py -h
usage: advobs202202_s18_12.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]

finding moonrise/moonset time

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_12.py -t 2022-07-09T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
Moon on 2022-07-09T16:00:00.000
  illumination = 0.770666
  moon phase = 0.9987759166624955 rad = 57.22564470407103 deg
% ./advobs202202_s18_12.py -t 2022-07-11T16:00:00
#
# input parameters
#
#  Location:
```

```
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-11T16:00:00
Moon on 2022-07-11T16:00:00.000
  illumination = 0.932781
  moon phase = 0.5245234117641752 rad = 30.052977749889873 deg
% ./advobs202202_s18_12.py -t 2022-07-13T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-13T16:00:00
Moon on 2022-07-13T16:00:00.000
  illumination = 0.998382
  moon phase = 0.08048014233055935 rad = 4.611172490153212 deg
% ./advobs202202_s18_12.py -t 2022-07-28T16:00:00
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-28T16:00:00
Moon on 2022-07-28T16:00:00.000
  illumination = 0.001855
  moon phase = 3.055421720605432 rad = 175.06276922329147 deg
```

6 Stars

6.1 Altitude and azimuth

Calculate altitude and azimuth of a target for a given location and time.

Python Code 13: advobs202202_s18_13.py

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

#
# Time-stamp: <2022/06/30 12:18:36 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.time
```

```
import astropy.coordinates

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m
unit_rad = astropy.units.rad
unit_deg = astropy.units.deg

# constructing parser object
desc   = "finding (Az, Alt) of targets at given location and date/time"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')
parser.add_argument ('target', nargs='+', default='Vega', \
                     help='names of targets')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime
list_target  = args.target

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)
print ("#  Targets:")
for target in list_target:
    print ("#   %s" % target)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
```

```
datetime = astropy.time.Time (datetime_str, scale='utc')

# processing each target
for target in list_target:
    # object
    obj = astroplan.FixedTarget.from_name (target)
    # alt and az of target
    obj_altaz = observer.altaz (datetime, obj)
    # printing results
    print ("Object = %s" % target)
    print ("  Az:  %s" % obj_altaz.az)
    print ("  Alt: %s" % obj_altaz.alt)
    print ("  airmass: %s" % obj_altaz.secz)
```

Execute the script.

```
% chmod a+x advobs202202_s18_13.py
% ./advobs202202_s18_13.py -h
usage: advobs202202_s18_13.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]
                              target [target ...]

finding (Az, Alt) of targets at given location and date/time

positional arguments:
  target                names of targets

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_13.py -t 2022-07-09T16:00:00 Vega Antares Fomalhaut
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
#  Targets:
#   Vega
#   Antares
#   Fomalhaut
Object = Vega
  Az:  335d31m31.64119429s
  Alt: 72d51m20.42889438s
  airmass: 1.0465010079665695
Object = Antares
  Az:  220d42m19.95193029s
```

```
  Alt: 26d23m16.11846553s
  airmass: 2.2499991512143653
Object = Fomalhaut
  Az:  131d39m46.77083376s
  Alt: 14d18m28.53496692s
  airmass: 4.0464030925395615
```

6.2 Time of rise and set of a target

Calculate the time of rise and set of a target.

Python Code 14: advobs202202_s18_14.py

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

#
# Time-stamp: <2022/06/30 12:22:39 (CST) daisuke>
#

# importing argparse module
import argparse

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
unit_m = astropy.units.m
unit_rad = astropy.units.rad
unit_deg = astropy.units.deg

# constructing parser object
desc   = "finding rise and set time of targets at given location"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')
parser.add_argument ('target', nargs='+', default='Vega', \
                     help='names of targets')

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

# input parameters
site_lon     = args.longitude
```

```
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime
list_target  = args.target

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)
print ("#  Targets:")
for target in list_target:
    print ("#   %s" % target)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')

# processing each target
for target in list_target:
    # object
    obj = astroplan.FixedTarget.from_name (target)
    # rise time
    time_rise = observer.target_rise_time (datetime, obj, which="nearest")
    # set time
    time_set  = observer.target_set_time (datetime, obj, which="nearest")
    # printing results
    print ("rise and set of %s on %s:" % (target, datetime) )
    print ("  rise time: %s" % time_rise.isot)
    print ("  set time:  %s" % time_set.isot)
```

Execute the script.

```
% chmod a+x advobs202202_s18_14.py
% ./advobs202202_s18_14.py -h
usage: advobs202202_s18_14.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME]
                              target [target ...]

finding rise and set time of targets at given location

positional arguments:
  target                names of targets

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

```
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format

% ./advobs202202_s18_14.py -t 2022-07-09T16:00 Vega Antares Fomalhaut
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00
#  Targets:
#   Vega
#   Antares
#   Fomalhaut
rise and set of Vega on 2022-07-09T16:00:00.000:
  rise time: 2022-07-09T08:03:29.530
  set time:  2022-07-09T22:44:34.155
rise and set of Antares on 2022-07-09T16:00:00.000:
  rise time: 2022-07-09T08:08:16.824
  set time:  2022-07-09T18:26:38.624
rise and set of Fomalhaut on 2022-07-09T16:00:00.000:
  rise time: 2022-07-09T14:42:11.902
  set time:  2022-07-10T00:46:46.322
```

7 Making an airmass plot

Make a time vs. airmass plot.

Python Code 15: advobs202202_s18_15.py

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

#
# Time-stamp: <2022/06/30 12:43:07 (CST) daisuke>
#

# importing argparse module
import argparse

# importing pathlib module
import pathlib

# importing sys module
import sys

# importing numpy module
import numpy

# importing astropy module
```

```
import astropy.units
import astropy.time
import astropy.coordinates

# importing astroplan module
import astroplan
import astroplan.plots

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

# units
unit_m    = astropy.units.m
unit_rad  = astropy.units.rad
unit_deg  = astropy.units.deg
unit_hour = astropy.units.hour

# constructing parser object
desc   = "making an airmass plot"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')
parser.add_argument ('-o', '--output', default='airmass.png', \
                     help='output file name (EPS, PDF, PNG, or PS file)')
parser.add_argument ('target', nargs='+', default='Vega', \
                     help='names of targets')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime
file_output  = args.output
list_target  = args.target

# making pathlib object
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing message
    print ("ERROR: output file '%s' exists." % file_output)
    # exit
    sys.exit ()
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("ERROR: output file must be either EPS, PDF, PNG, or PS.")
    # exit
```

```
    sys.exit ()

# printing input parameters
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)
print ("#  Targets:")
for target in list_target:
    print ("#   %s" % target)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')
datetime = datetime + numpy.linspace (-8, +8, 100) * unit_hour

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

# plotting
box = ax.get_position ()
ax.set_position ([box.x0, box.y0, box.width * 0.83, box.height])

# processing each target
for i in range ( len (list_target) ):
    # object
    target = list_target[i]
    obj = astroplan.FixedTarget.from_name (target)
    if (i == len (list_target) - 1):
        astroplan.plots.plot_airmass (obj, observer, datetime, ax=ax, \
                                      brightness_shading=True, max_airmass=2.5)
    else:
        astroplan.plots.plot_airmass (obj, observer, datetime, ax=ax)

# plotting
ax.grid ()
ax.legend (bbox_to_anchor=(1.05, 1.00), loc='upper left', shadow=True)

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

Execute the script, and show the plot. (Fig. 11)

```
% ./advobs202202_s18_15.py -h
```

```
usage: advobs202202_s18_15.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME] [-o OUTPUT]
                              target [target ...]

making an airmass plot

positional arguments:
  target                names of targets

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format
  -o OUTPUT, --output OUTPUT
                        output file name (EPS, PDF, PNG, or PS file)

% ./advobs202202_s18_15.py -t 2022-07-09T16:00:00 -o airmass_20220709.pdf \
? Vega Antares Fomalhaut
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
#  Targets:
#   Vega
#   Antares
#   Fomalhaut
/usr/pkg/lib/python3.9/site-packages/astroplan/plots/time_dependent.py:194: User
Warning: linestyle is redundantly defined by the 'linestyle' keyword argument an
d the fmt string "-" (-> linestyle='-'). The keyword argument will take preceden
ce.
  ax.plot_date(timetoplot.plot_date, masked_airmass, label=target_name, **style_
kwargs)
/usr/pkg/lib/python3.9/site-packages/astroplan/plots/time_dependent.py:194: User
Warning: linestyle is redundantly defined by the 'linestyle' keyword argument an
d the fmt string "-" (-> linestyle='-'). The keyword argument will take preceden
ce.
  ax.plot_date(timetoplot.plot_date, masked_airmass, label=target_name, **style_
kwargs)
/usr/pkg/lib/python3.9/site-packages/astroplan/plots/time_dependent.py:194: User
Warning: linestyle is redundantly defined by the 'linestyle' keyword argument an
d the fmt string "-" (-> linestyle='-'). The keyword argument will take preceden
ce.
  ax.plot_date(timetoplot.plot_date, masked_airmass, label=target_name, **style_
kwargs)
% ls -l airmass_20220709.pdf
-rw-r--r--  1 daisuke  taiwan  15530 Jun 30 12:47 airmass_20220709.pdf
% okular airmass_20220709.pdf
```

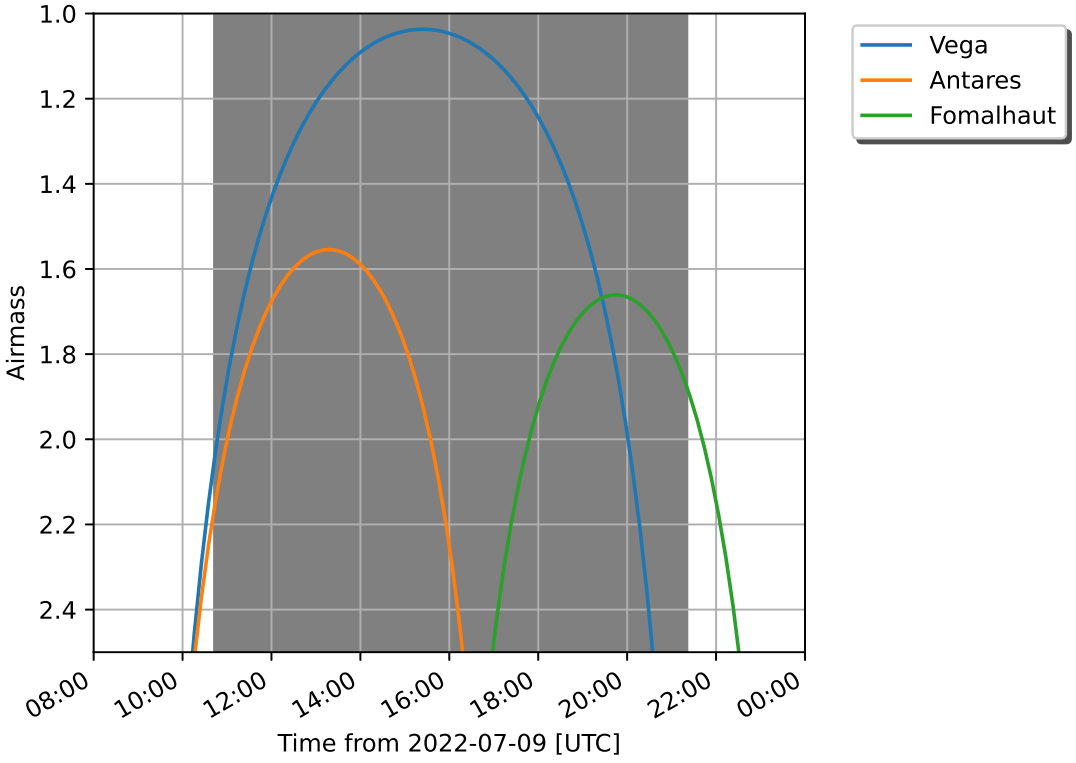

Figure 11: The time vs. airmass plot of bright stars.

You may need a package `fonttools`. If you see an error message `ModuleNotFoundError: No module named 'fontTools'`, then install the package `fonttools` and re-run the script.

8 Making a sky chart

Make a sky chart.

Python Code 16: advobs202202_s18_16.py

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

#
# Time-stamp: <2022/06/30 12:55:29 (CST) daisuke>
#

# importing argparse module
import argparse

# importing pathlib module
import pathlib

# importing sys module
import sys

# importing numpy module
import numpy

# importing astropy module
import astropy.units
import astropy.time
```

```
import astropy.coordinates

# importing astroplan module
import astroplan
import astroplan.plots

# importing matplotlib module
import matplotlib.pyplot

# units
unit_m    = astropy.units.m
unit_rad  = astropy.units.rad
unit_deg  = astropy.units.deg
unit_hour = astropy.units.hour

# constructing parser object
desc   = "making a sky chart"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-l', '--longitude', default='+120d52m25s', \
                     help='longitude of observing site in format "+121d11m12s"')
parser.add_argument ('-b', '--latitude', default='+23d28m07s', \
                     help='latitude of observing site in format "+24d58m12s"')
parser.add_argument ('-a', '--altitude', type=float, default=2862.0, \
                     help='altitude above sea-level in metre')
parser.add_argument ('-t', '--datetime', default='2000-01-01T12:00:00.000', \
                     help='date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format')
parser.add_argument ('-o', '--output', default='airmass.png', \
                     help='output file name (EPS, PDF, PNG, or PS file)')
parser.add_argument ('target', nargs='+', default='Vega', \
                     help='names of targets')

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

# input parameters
site_lon     = args.longitude
site_lat     = args.latitude
site_alt     = args.altitude * unit_m
datetime_str = args.datetime
file_output  = args.output
list_target  = args.target

# making pathlib object
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing message
    print ("ERROR: output file '%s' exists." % file_output)
    # exit
    sys.exit ()
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("ERROR: output file must be either EPS, PDF, PNG, or PS.")
    # exit
    sys.exit ()

# printing input parameters
```

```
print ("#")
print ("# input parameters")
print ("#")
print ("#  Location:")
print ("#   longitude =", site_lon)
print ("#   latitude  =", site_lat)
print ("#   altitude  =", site_alt)
print ("#  Date/Time:")
print ("#   date/time (UT) =", datetime_str)
print ("#  Targets:")
for target in list_target:
    print ("#   %s" % target)

# location object
location = astropy.coordinates.EarthLocation.from_geodetic \
    (site_lon, site_lat, site_alt)

# observer object
observer = astroplan.Observer (location=location, name="observer", \
                               timezone="UTC")

# time object
datetime = astropy.time.Time (datetime_str, scale='utc')
datetime = datetime + numpy.linspace (-8, +8, 100) * unit_hour

# processing each target
for i in range ( len (list_target) ):
    # object
    target = list_target[i]
    obj = astroplan.FixedTarget.from_name (target)
    astroplan.plots.plot_sky (obj, observer, datetime)

# plotting
matplotlib.pyplot.legend (loc='upper left', bbox_to_anchor=(1.05, 1.0) )
matplotlib.pyplot.title ("sky chart")

# saving the plot into a file
matplotlib.pyplot.savefig (file_output, bbox_inches="tight", dpi=225)
```

Execute the script, and show the plot. (Fig. 12)

```
% chmod a+x advobs202202_s18_16.py
% ./advobs202202_s18_16.py -h
usage: advobs202202_s18_16.py [-h] [-l LONGITUDE] [-b LATITUDE] [-a ALTITUDE]
                              [-t DATETIME] [-o OUTPUT]
                              target [target ...]

making a sky chart

positional arguments:
  target                names of targets

optional arguments:
  -h, --help            show this help message and exit
  -l LONGITUDE, --longitude LONGITUDE
                        longitude of observing site in format "+121d11m12s"
  -b LATITUDE, --latitude LATITUDE
                        latitude of observing site in format "+24d58m12s"
```

```
  -a ALTITUDE, --altitude ALTITUDE
                        altitude above sea-level in metre
  -t DATETIME, --datetime DATETIME
                        date/time in UT in "YYYY-MM-DDThh:mm:ss.sss" format
  -o OUTPUT, --output OUTPUT
                        output file name (EPS, PDF, PNG, or PS file)

% ./advobs202202_s18_16.py -t 2022-07-09T16:00:00 -o sky_20220709.pdf \
? Vega Antares Fomalhaut
#
# input parameters
#
#  Location:
#   longitude = +120d52m25s
#   latitude  = +23d28m07s
#   altitude  = 2862.0 m
#  Date/Time:
#   date/time (UT) = 2022-07-09T16:00:00
#  Targets:
#   Vega
#   Antares
#   Fomalhaut
% ls -l sky_20220709.pdf
-rw-r--r--  1 daisuke  taiwan  21464 Jun 30 12:57 sky_20220709.pdf
% okular sky_20220709.pdf
```

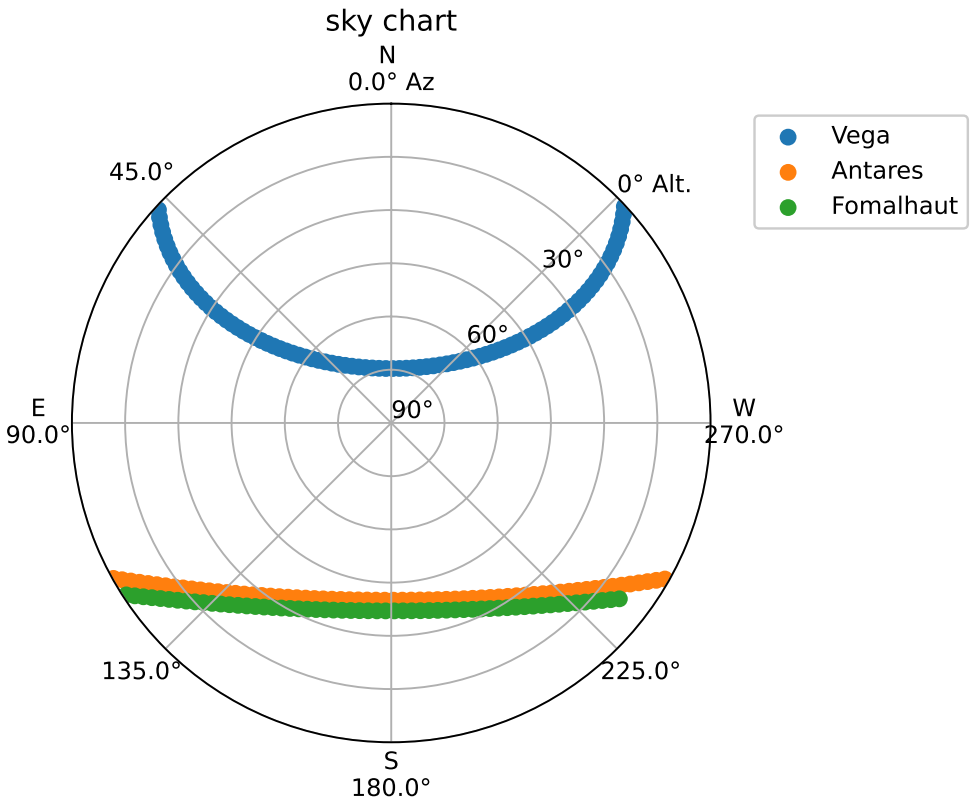

Figure 12: An example of sky chart.

9 Making a finding chart

Make a finding chart.

Python Code 17: advobs202202_s18_20.py

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

#
# Time-stamp: <2022/06/30 13:05:32 (CST) daisuke>
#

# importing argparse module
import argparse

# importing pathlib module
import pathlib

# importing sys module
import sys

# importing astropy module
import astropy.units

# importing astroplan module
import astroplan
import astroplan.plots

# importing matplotlib module
import matplotlib.pyplot

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# units
unit_arcmin = astropy.units.arcmin

# constructing parser object
desc   = "making a finding chart"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-t', '--target', default='M1', help='target name')
parser.add_argument ('-f', '--fov', type=float, default=13.0, \
                     help='field-of-view in arcmin (default: 13 arcmin)')
parser.add_argument ('-o', '--output', default='chart.png', \
                     help='output file name (EPS, PDF, PNG, or PS file)')

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

# input parameters
target      = args.target
fov         = args.fov * unit_arcmin
file_output = args.output

# making pathlib object
```

```
path_output = pathlib.Path (file_output)
if (path_output.exists ()):
    # printing message
    print ("ERROR: output file '%s' exists." % file_output)
    # exit
    sys.exit ()
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("ERROR: output file must be either EPS, PDF, PNG, or PS.")
    # exit
    sys.exit ()

# target
obj = astroplan.FixedTarget.from_name (target)

# image
ax, hdu = astroplan.plots.plot_finder_image (obj, fov_radius=fov)

# saving the image to file
matplotlib.pyplot.savefig (file_output)
```

Execute the script, and show the finding chart. (Fig. **??**)

```
% chmod a+x advobs202202_s18_17.py
% ./advobs202202_s18_17.py -h
usage: advobs202202_s18_17.py [-h] [-t TARGET] [-f FOV] [-o OUTPUT]

making a finding chart

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET, --target TARGET
                        target name
  -f FOV, --fov FOV     field-of-view in arcmin (default: 13 arcmin)
  -o OUTPUT, --output OUTPUT
                        output file name (EPS, PDF, PNG, or PS file)

% ./advobs202202_s18_17.py -t M8 -o chart_m8.pdf
% Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv10828150129697.fits
|==========================================| 374k/374k (100.00%)         3s
% ls -l chart_m8.pdf
-rw-r--r--  1 daisuke  taiwan  75402 Jun 30 13:08 chart_m8.pdf
% okular chart_m8.pdf
```

10 Exercises

1. astroquery package
   (a) Install astroquery package on your computer. Record the procedure of installation process.
   (b) Make 3 Python scripts using astroquery package. Show the source code of your Python scripts.
2. astroplan package
   (a) Install astroplan package on your computer. Record the procedure of installation process.
   (b) Make 3 Python scripts using astroplan package. Show the source code of your Python scripts.

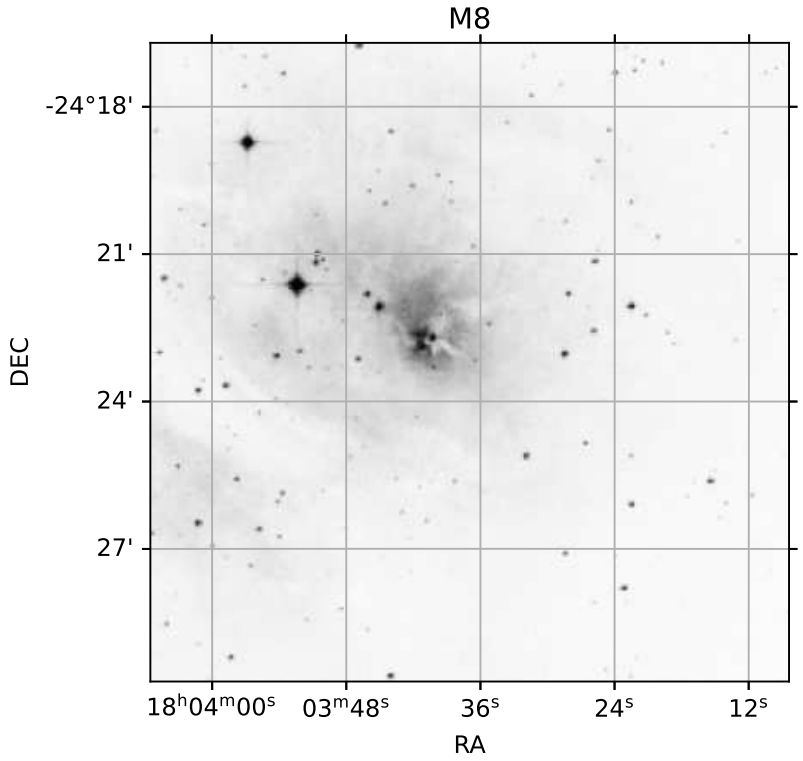

Figure 13: An example of findig chart.

3. Coordinate conversions
   (a) Describe the coordinate conversion from equatorial coordinate system into ecliptic coordinate system.
   (b) Describe the coordinate conversion from equatorial coordinate system into horizontal coordinate system.
   (c) Describe the coordinate conversion from equatorial coordinate system into galactic coordinate system.
4. Position of the Sun on the sky
   (a) Guess a rough position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Mar 2021 as observed at NCU main campus. Describe your answer.
   (b) Make a Python script to calculate the position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Mar 2021 as observed at NCU main campus. Run the script and show the result. Show the Python script you made.
   (c) Guess a rough position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Jun 2021 as observed at NCU main campus. Describe your answer.
   (d) Make a Python script to calculate the position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Jun 2021 as observed at NCU main campus. Run the script and show the result. Show the Python script you made.
   (e) Guess a rough position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Sep 2021 as observed at NCU main campus. Describe your answer.
   (f) Make a Python script to calculate the position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Sep 2021 as observed at NCU main campus. Run the script and show the result. Show the Python script you made.
   (g) Guess a rough position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Dec 2021 as observed at NCU main campus. Describe your answer.
   (h) Make a Python script to calculate the position of the Sun on the sky on ecliptic coordinate at 04:00:00 (UT) on 21 Dec 2021 as observed at NCU main campus. Run the script and show the result. Show the Python script you made.

5. Sunrise and sunset
   (a) Describe the method to calculate sunset and sunrise time for a given location and date.
   (b) Calculate the time of sunset at NCU main campus on 20 May 2021. Show the Python script you made.
   (c) Calculate the time of sunset in Chiayi city on 21 Jun 2021. Show the Python script you made.
   (d) Calculate the time of sunrise at Alishan on 10 Aug 2021. Show the Python script you made.
   (e) Calculate the time of sunrise at Kenting on 25 Aug 2021. Show the Python script you made.
6. Twilight
   (a) What is the definition of civil twilight?
   (b) What is the definition of nautical twilight?
   (c) What is the definition of astronomical twilight?
   (d) How important is it to know the time of twilight when doing astronomical observations?
7. Time for your observation
   (a) Suppose you have one night of observing time of VLT UT1 on 5 Jun 2021. You begin the observation at the end of evening twilight and finish the observation at the start of morning twilight. How many hours can you do your observation? Show the Python script you made.
8. Development Ephemerides
   (a) What is DE430?
9. The Moon
   (a) Calculate the position of the Moon on the sky as seen from NCU main campus at 10:00:00 (UT) on 14 June 2021. Show (RA, Dec) and (Az, Alt) of the Moon. Show the Python script you made.
   (b) Which one is your favourite planetarium type software?
   (c) Use your favourite planetarium type software to check the position of the Moon on the sky as seen from NCU main campus at 10:00:00 (UT) on 14 June 2021. Is the value same as the result of your calculation using Python and Astropy? If you see a discrepancy between two results, what is a possible reason for its discrepancy? Explain your answer.
10. The Sun and the Moon
    (a) Calculate RA and Dec of the Sun at 11:00:00 (UT) on 26 May 2021 as observed at Lulin Observatory. Show the source code of your Python script.
    (b) What is the Moon phase at 11:00:00 (UT) on 26 May 2021?
    (c) Guess where is the Moon on the sky.
    (d) Calculate RA and Dec of the Moon at 11:00:00 (UT) on 26 May 2021 as observed at Lulin Observatory. Show the source code of your Python script.
11. Positions of planets
    (a) Show RA and Dec of Jupiter at 15:00:00 (UT) on 15 Jul 2021. Show the Python script you made.
    (b) Show Az and Alt of Jupiter at Lulin at 15:00:00 (UT) on 15 Jul 2021. Show the Python script you made.
    (c) Show RA and Dec of Uranus at 12:00:00 (UT) on 1 Aug 2021. Show the Python script you made.
    (d) Show Az and Alt of Uranus at Mauna Kea at 12:00:00 (UT) on 1 Aug 2021. Show the Python script you made.
12. Airmass
    (a) What is airmass?
    (b) How useful is it?
13. Kepler-13
    (a) Suppose you are going to carry out photometric observations of Kepler-13 to detect transits of the exoplanet Kepler-13b at Lulin Observatory. Make time vs. airmass plots and sky charts, and discuss suitable months to observe this target. Describe your answer. Show the Python script you made.

14. Finding chart

    (a) Make a finding chart of a photometric standard star F11 for Lulin One-meter telescope. Show the Python script you made.

    (b) Make a finding chart of a photometric standard star PG1323-085A for Lulin One-meter telescope. Show the Python script you made.

    (c) Make a finding chart of Kepler-13 for Lulin One-meter telescope. Show the Python script you made.

15. Planning your observation

    (a) Suppose you have 5 nights of observing run starting on 15 Jul 2021. Plan your observation. Show the target list. Describe the night-time schedule of your observation. Show time vs. airmass plots of your targets. Show sky charts. Make finding charts. Show all the Python scripts you made.