Astroinformatics 2022
Session 08: Sun, Moon, and observation planning

Kinoshita Daisuke

7 November 2022
publicly accessible version

The "`astroplan`" package provides useful functionalities for observation planning. For this session, we try some functionalities of "`astroplan`" package.

1 Sample Python scripts for this session

Sample Python scripts for this session can be downloaded from GitHub repository. Visit following GitHub repository.

- `https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209`

1.1 Executing sample Python scripts on a terminal emulator

If you prefer to execute sample Python scripts for this session on a terminal emulator, download `.py` files from GitHub repository.

1.2 Executing sample Python scripts on JupyterLab

If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

1.3 Executing sample Python scripts using Binder

If you prefer to execute sample Python scripts for this session on Binder, visit following web page.

- `https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD`

Start your favourite web browser and go to above web page. (Fig. 1) In a minute or two, you see JupyterLab working on your web browser. (Fig. 2) Go to the directory (folder) "`s08`". (Fig. 3) Choose the file "`ai202209_s08.ipynb`" (Fig. 4 and 5) and open it (Fig. 6).

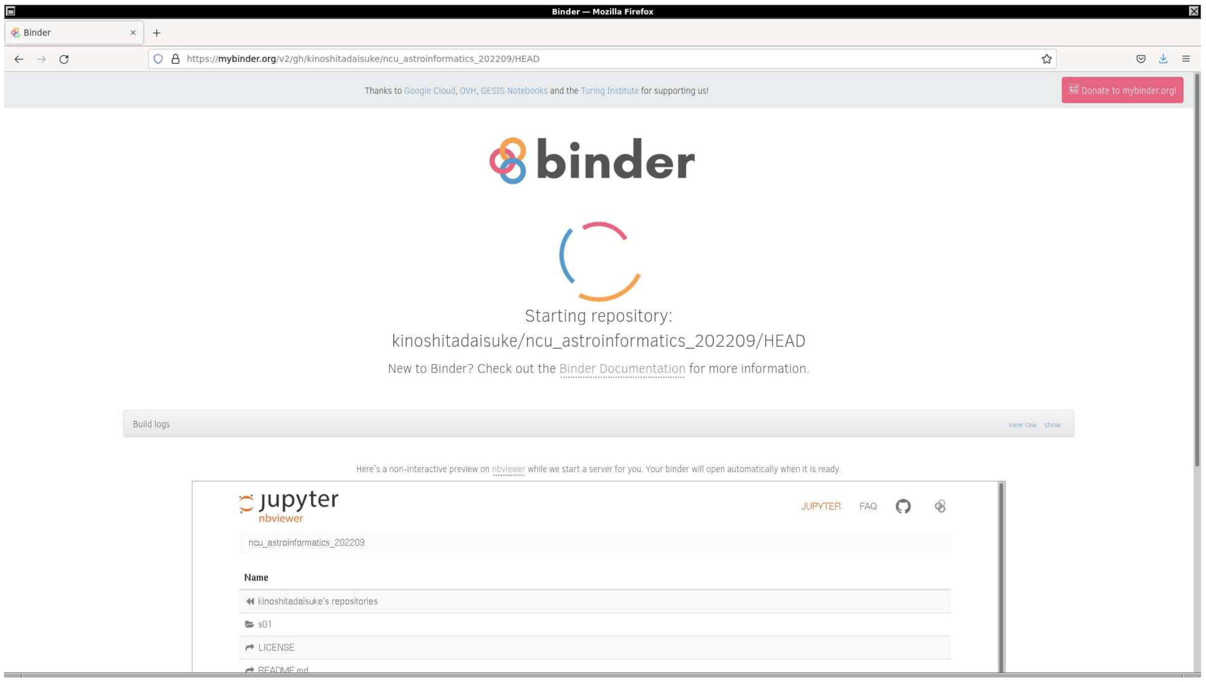

Binder — Mozilla Firefox
https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD
Thanks to Google Cloud, OVH, GESIS Notebooks and the Turing Institute for supporting us!
Donate to mybinder.org!
binder
Starting repository:
kinoshitadaisuke/ncu_astroinformatics_202209/HEAD
New to Binder? Check out the Binder Documentation for more information.
Build logs
Here's a non-interactive preview on nbviewer while we start a server for you. Your binder will open automatically when it is ready.
jupyter nbviewer
ncu_astroinformatics_202209
Name
kinoshitadaisuke's repositories
s01
LICENSE

Figure 1: Using Binder to execute sample Python scripts for this session.

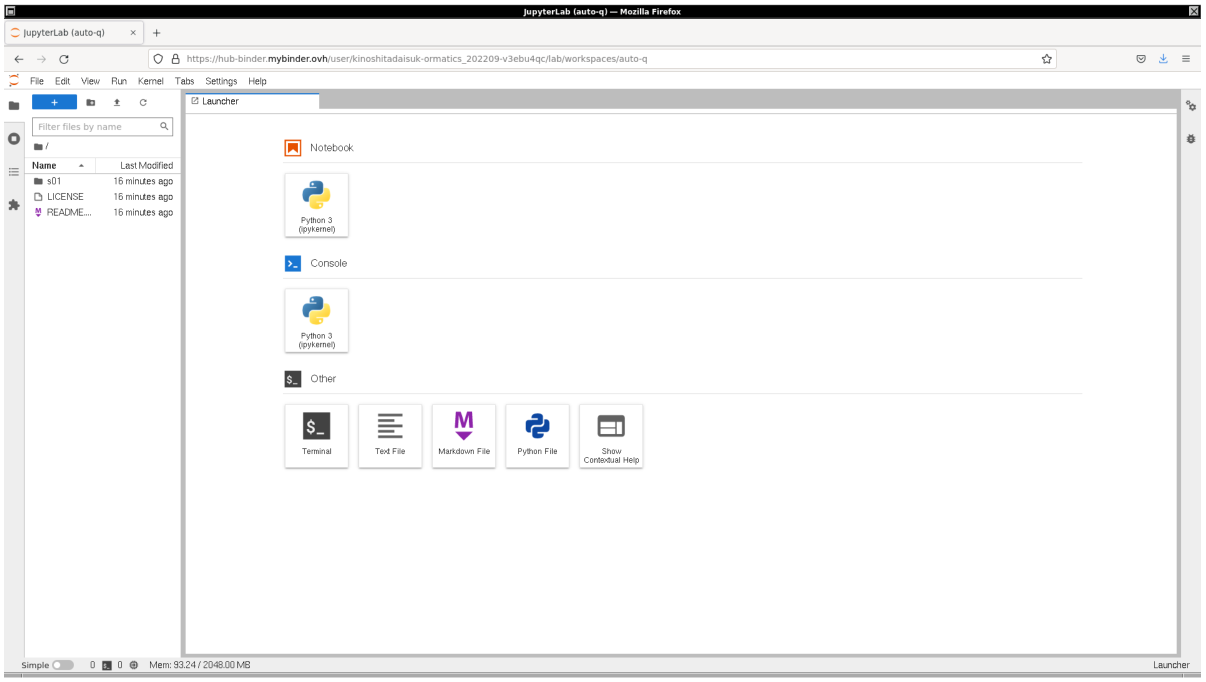
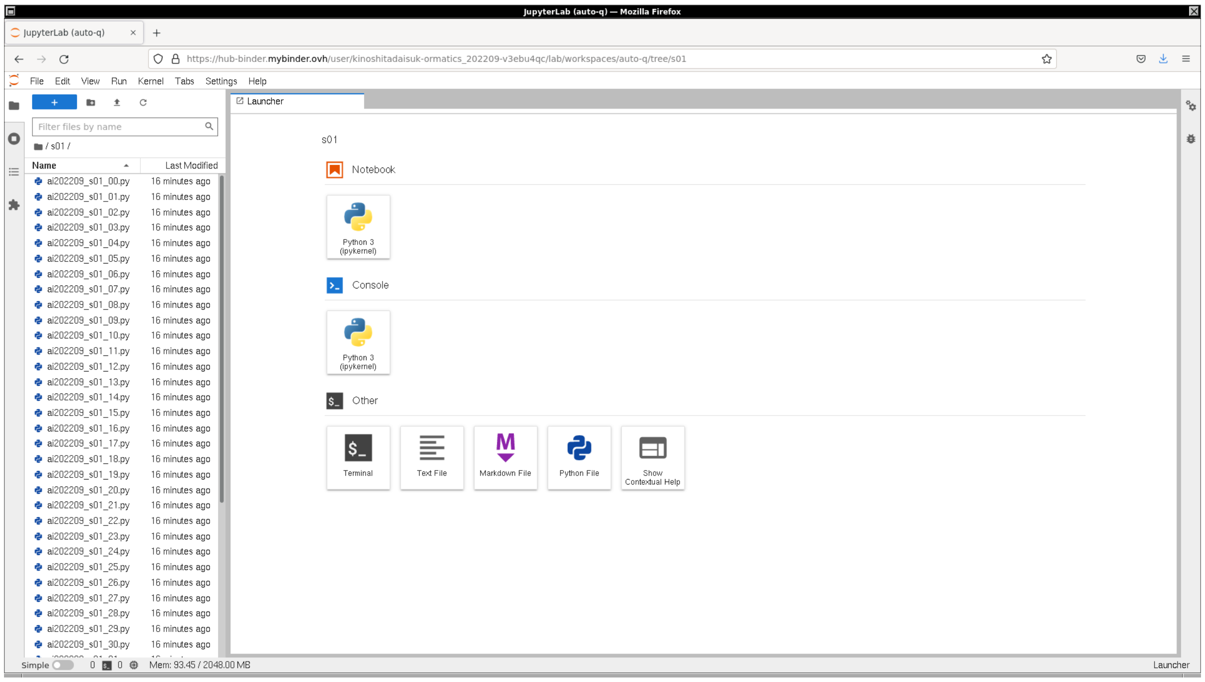
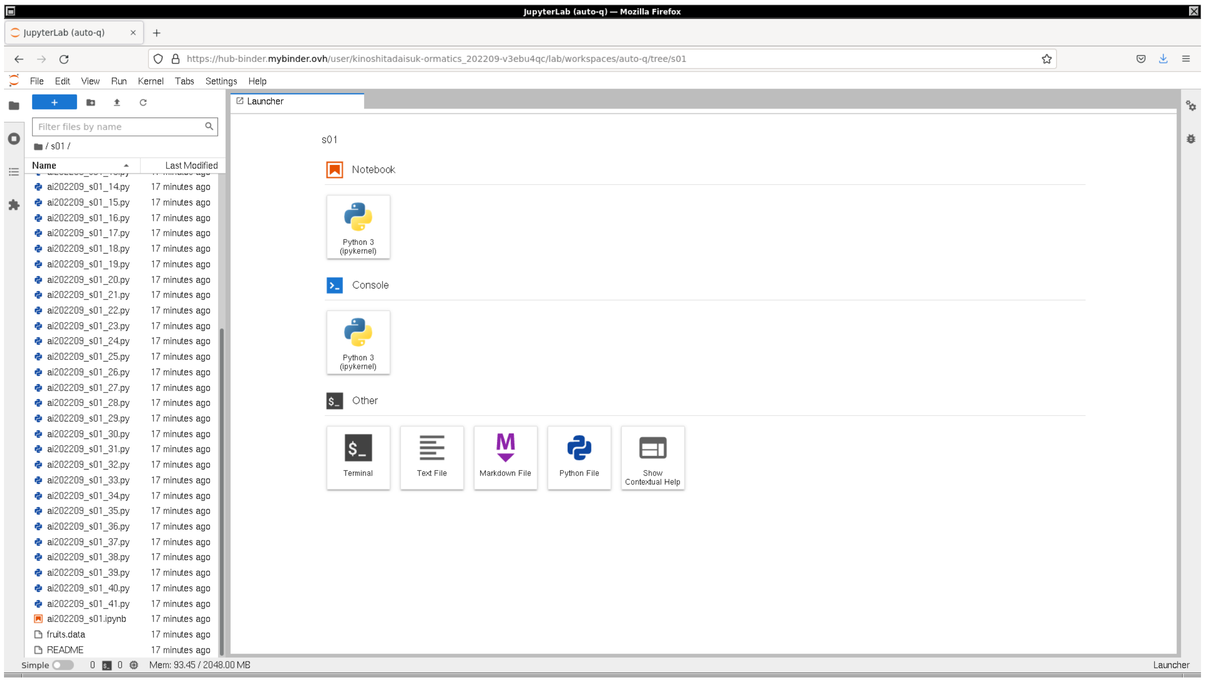


Figure 2: Using Binder to execute sample Python scripts for this session.



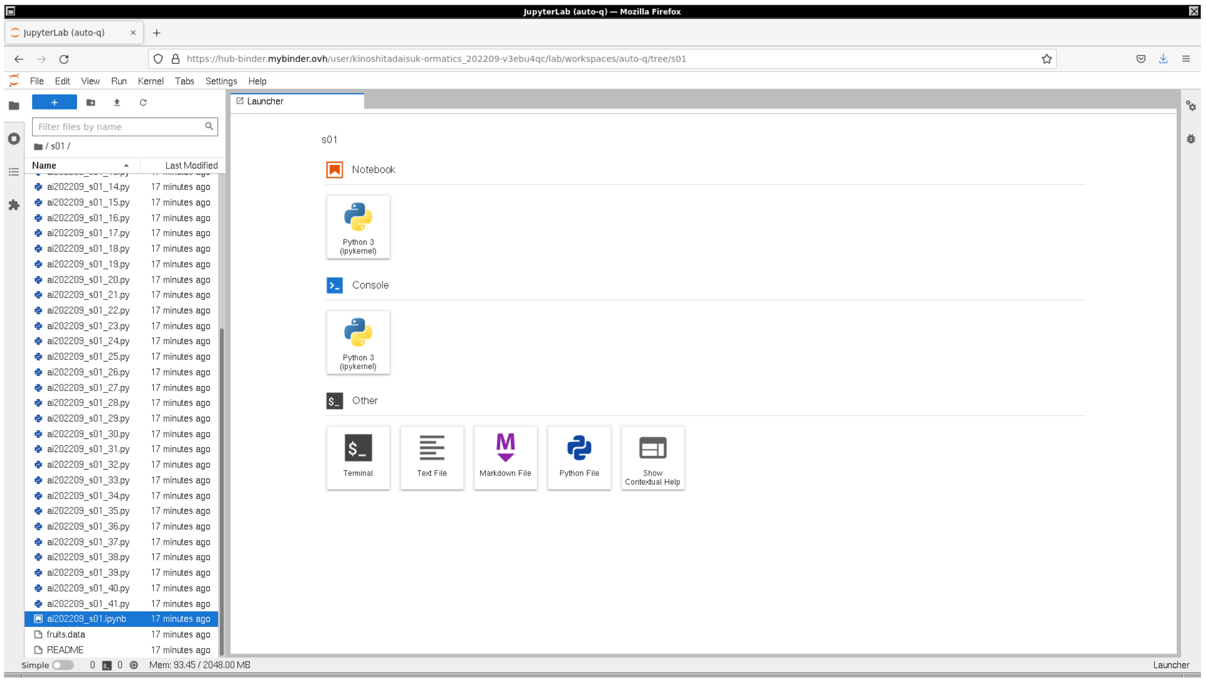
Name	Last Modified
ai202209_s01_00.py	16 minutes ago
ai202209_s01_01.py	16 minutes ago
ai202209_s01_02.py	16 minutes ago
ai202209_s01_03.py	16 minutes ago
ai202209_s01_04.py	16 minutes ago
ai202209_s01_05.py	16 minutes ago
ai202209_s01_06.py	16 minutes ago
ai202209_s01_07.py	16 minutes ago
ai202209_s01_08.py	16 minutes ago
ai202209_s01_09.py	16 minutes ago
ai202209_s01_10.py	16 minutes ago
ai202209_s01_11.py	16 minutes ago
ai202209_s01_12.py	16 minutes ago
ai202209_s01_13.py	16 minutes ago
ai202209_s01_14.py	16 minutes ago
ai202209_s01_15.py	16 minutes ago
ai202209_s01_16.py	16 minutes ago
ai202209_s01_17.py	16 minutes ago
ai202209_s01_18.py	16 minutes ago
ai202209_s01_19.py	16 minutes ago
ai202209_s01_20.py	16 minutes ago
ai202209_s01_21.py	16 minutes ago
ai202209_s01_22.py	16 minutes ago
ai202209_s01_23.py	16 minutes ago
ai202209_s01_24.py	16 minutes ago
ai202209_s01_25.py	16 minutes ago
ai202209_s01_26.py	16 minutes ago
ai202209_s01_27.py	16 minutes ago
ai202209_s01_28.py	16 minutes ago
ai202209_s01_29.py	16 minutes ago
ai202209_s01_30.py	16 minutes ago

Figure 3: Using Binder to execute sample Python scripts for this session.



Name	Last Modified
ai202209_s01_14.py	17 minutes ago
ai202209_s01_15.py	17 minutes ago
ai202209_s01_16.py	17 minutes ago
ai202209_s01_17.py	17 minutes ago
ai202209_s01_18.py	17 minutes ago
ai202209_s01_19.py	17 minutes ago
ai202209_s01_20.py	17 minutes ago
ai202209_s01_21.py	17 minutes ago
ai202209_s01_22.py	17 minutes ago
ai202209_s01_23.py	17 minutes ago
ai202209_s01_24.py	17 minutes ago
ai202209_s01_25.py	17 minutes ago
ai202209_s01_26.py	17 minutes ago
ai202209_s01_27.py	17 minutes ago
ai202209_s01_28.py	17 minutes ago
ai202209_s01_29.py	17 minutes ago
ai202209_s01_30.py	17 minutes ago
ai202209_s01_31.py	17 minutes ago
ai202209_s01_32.py	17 minutes ago
ai202209_s01_33.py	17 minutes ago
ai202209_s01_34.py	17 minutes ago
ai202209_s01_35.py	17 minutes ago
ai202209_s01_36.py	17 minutes ago
ai202209_s01_37.py	17 minutes ago
ai202209_s01_38.py	17 minutes ago
ai202209_s01_39.py	17 minutes ago
ai202209_s01_40.py	17 minutes ago
ai202209_s01_41.py	17 minutes ago
ai202209_s01.ipynb	17 minutes ago
fruits.data	17 minutes ago
README	17 minutes ago

Figure 4: Using Binder to execute sample Python scripts for this session.



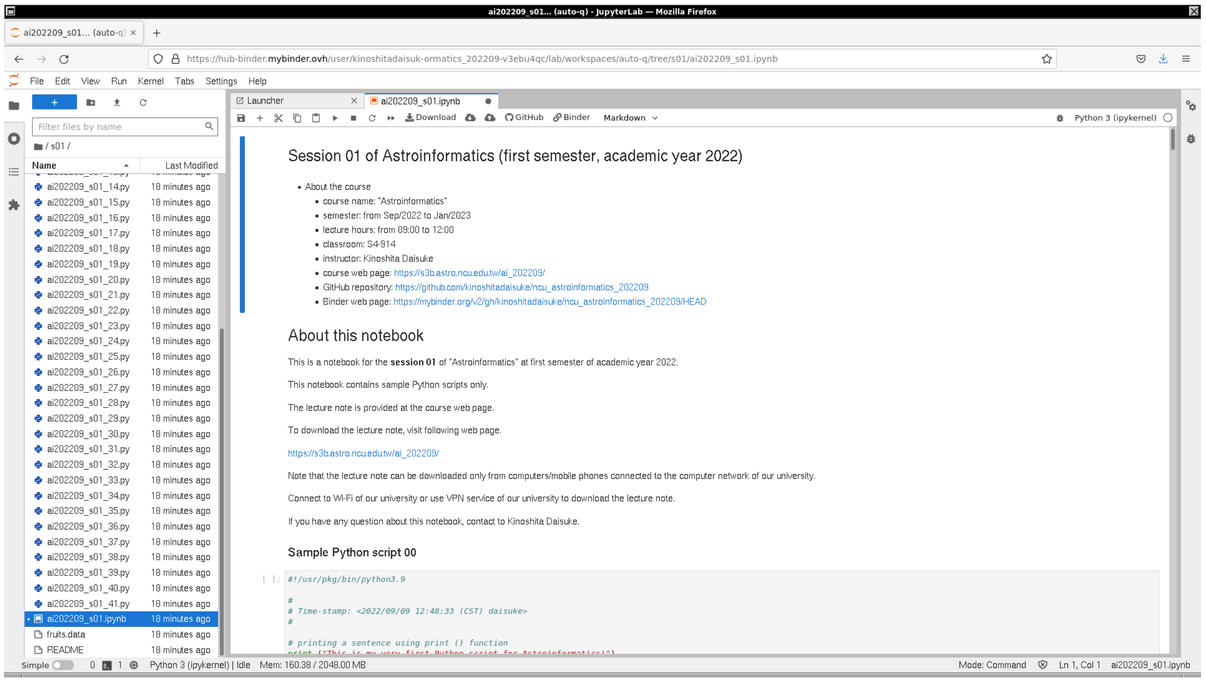
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01

/ s01 /

Name	Last Modified
ai202209_s01_14.py	17 minutes ago
ai202209_s01_15.py	17 minutes ago
ai202209_s01_16.py	17 minutes ago
ai202209_s01_17.py	17 minutes ago
ai202209_s01_18.py	17 minutes ago
ai202209_s01_19.py	17 minutes ago
ai202209_s01_20.py	17 minutes ago
ai202209_s01_21.py	17 minutes ago
ai202209_s01_22.py	17 minutes ago
ai202209_s01_23.py	17 minutes ago
ai202209_s01_24.py	17 minutes ago
ai202209_s01_25.py	17 minutes ago
ai202209_s01_26.py	17 minutes ago
ai202209_s01_27.py	17 minutes ago
ai202209_s01_28.py	17 minutes ago
ai202209_s01_29.py	17 minutes ago
ai202209_s01_30.py	17 minutes ago
ai202209_s01_31.py	17 minutes ago
ai202209_s01_32.py	17 minutes ago
ai202209_s01_33.py	17 minutes ago
ai202209_s01_34.py	17 minutes ago
ai202209_s01_35.py	17 minutes ago
ai202209_s01_36.py	17 minutes ago
ai202209_s01_37.py	17 minutes ago
ai202209_s01_38.py	17 minutes ago
ai202209_s01_39.py	17 minutes ago
ai202209_s01_40.py	17 minutes ago
ai202209_s01_41.py	17 minutes ago
ai202209_s01.ipynb	17 minutes ago
fruits.data	17 minutes ago
README	17 minutes ago

```
0  0   Mem: 93.45 / 2048.00 MB
```

Figure 5: Using Binder to execute sample Python scripts for this session.



https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01/ai202209_s01.ipynb

/ s01 /

Name	Last Modified
ai202209_s01_14.py	18 minutes ago
ai202209_s01_15.py	18 minutes ago
ai202209_s01_16.py	18 minutes ago
ai202209_s01_17.py	18 minutes ago
ai202209_s01_18.py	18 minutes ago
ai202209_s01_19.py	18 minutes ago
ai202209_s01_20.py	18 minutes ago
ai202209_s01_21.py	18 minutes ago
ai202209_s01_22.py	18 minutes ago
ai202209_s01_23.py	18 minutes ago
ai202209_s01_24.py	18 minutes ago
ai202209_s01_25.py	18 minutes ago
ai202209_s01_26.py	18 minutes ago
ai202209_s01_27.py	18 minutes ago
ai202209_s01_28.py	18 minutes ago
ai202209_s01_29.py	18 minutes ago
ai202209_s01_30.py	18 minutes ago
ai202209_s01_31.py	18 minutes ago
ai202209_s01_32.py	18 minutes ago
ai202209_s01_33.py	18 minutes ago
ai202209_s01_34.py	18 minutes ago
ai202209_s01_35.py	18 minutes ago
ai202209_s01_36.py	18 minutes ago
ai202209_s01_37.py	18 minutes ago
ai202209_s01_38.py	18 minutes ago
ai202209_s01_39.py	18 minutes ago
ai202209_s01_40.py	18 minutes ago
ai202209_s01_41.py	18 minutes ago
ai202209_s01.ipynb	18 minutes ago
fruits.data	18 minutes ago
README	18 minutes ago

Session 01 of Astroinformatics (first semester, academic year 2022)

- About the course
  - course name: "Astroinformatics"
  - semester: from Sep/2022 to Jan/2023
  - lecture hours: from 09:00 to 12:00
  - classroom: S4-914
  - instructor: Kinoshita Daisuke
  - course web page: https://s3b.astro.ncu.edu.tw/ai_202209/
  - GitHub repository: https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209
  - Binder web page: https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD

About this notebook

This is a notebook for the **session 01** of "Astroinformatics" at first semester of academic year 2022.

This notebook contains sample Python scripts only.

The lecture note is provided at the course web page.

To download the lecture note, visit following web page.

https://s3b.astro.ncu.edu.tw/ai_202209/

Note that the lecture note can be downloaded only from computers/mobile phones connected to the computer network of our university.

Connect to Wi-Fi of our university or use VPN service of our university to download the lecture note.

If you have any question about this notebook, contact to Kinoshita Daisuke.

Sample Python script 00

```
[ ]: #!/usr/pkg/bin/python3.9

     #
     # Time-stamp: <2022/09/09 12:48:33 (CST) daisuke>
     #

     # printing a sentence using print () function
```

```
0  1   Python 3 (ipykernel) | Idle   Mem: 160.38 / 2048.00 MB        Mode: Command   Ln 1, Col 1   ai202209_s01.ipynb
```

Figure 6: Using Binder to execute sample Python scripts for this session.

2 The "astroplan" package

2.1 Checking whether you have astroplan package

Try following to check whether or not you have "astroplan" package on your computer.

```
% python3.9
Python 3.9.13 (main, Jul 27 2022, 21:06:59)
[GCC 10.4.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import astroplan
>>> exit ()
```

If the "astroplan" package is successfully imported without any error or warning, you have "astroplan" package properly installed on your computer.

If you see an error message like below, you do not have "astroplan" package on your computer.

```
% python3.9
Python 3.9.13 (main, Jul 27 2022, 21:06:59)
[GCC 10.4.0] 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 ()
```

If you do not have "astroplan" package on your computer, (1) install "astroplan" package on your computer, or (2) use Binder to execute sample scripts for this session.

2.2 Resources of astroplan package

Here are some resources of "astroplan" package.

- Astroplan documentation: `https://astroplan.readthedocs.io/` (Fig. 7)
- GitHub repository of Astroplan: `https://github.com/astropy/astroplan`
- Astroplan on PyPI: `https://pypi.org/project/astroplan/`
- A paper about Astroplan: `https://doi.org/10.3847/1538-3881/aaa47e`

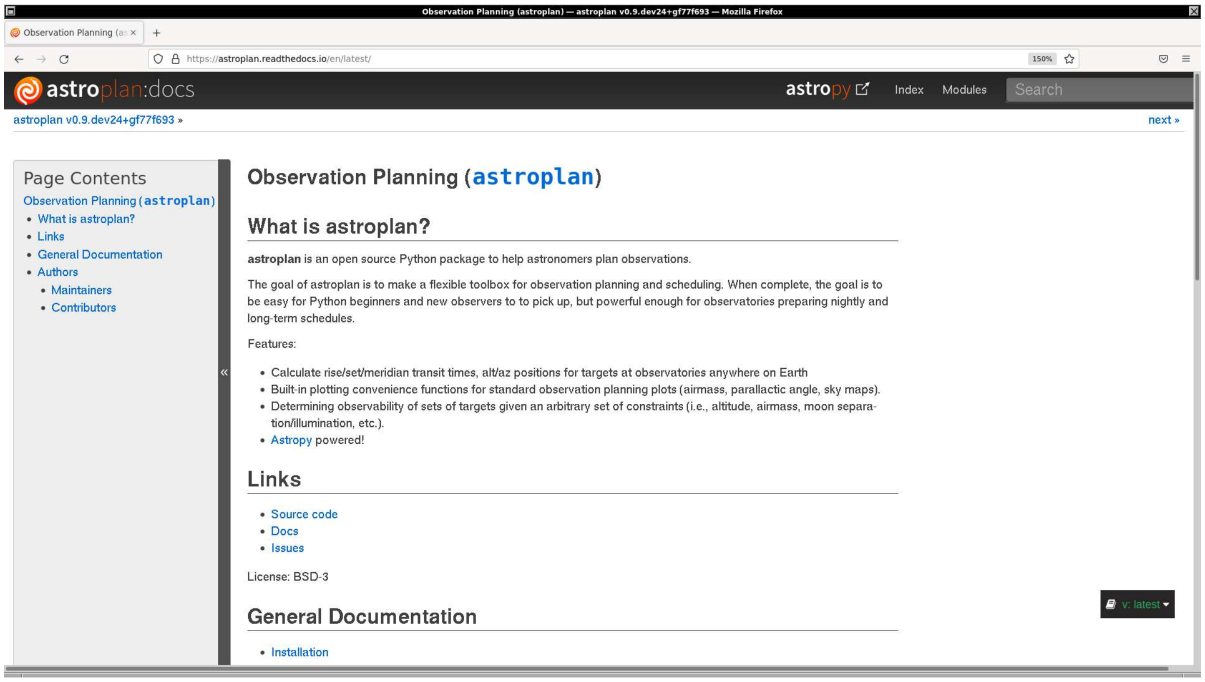
2.3 The "jplephem" package

The installation of "jplephem" package is recommended.

- jplephem on PyPI: `https://pypi.org/project/jplephem/`
- GitHub repository of jplephem: `https://github.com/brandon-rhodes/python-jplephem`

3 Constructing an observer object

We need to define the location of the observer for observation planning. Here is a way to construct an observer object of "astroplan" package.


Observation Planning (astroplan) — astroplan v0.9.dev24+gf77f693 — Mozilla Firefox

https://astroplan.readthedocs.io/en/latest/

astroplan v0.9.dev24+gf77f693 » next »

Page Contents

- Observation Planning (astroplan)
  - What is astroplan?
  - Links
  - General Documentation
  - Authors
    - Maintainers
    - Contributors

Observation Planning (astroplan)

What is astroplan?

astroplan is an open source Python package to help astronomers plan observations.

The goal of astroplan is to make a flexible toolbox for observation planning and scheduling. When complete, the goal is to be easy for Python beginners and new observers to to pick up, but powerful enough for observatories preparing nightly and long-term schedules.

Features:

- Calculate rise/set/meridian transit times, alt/az positions for targets at observatories anywhere on Earth
- Built-in plotting convenience functions for standard observation planning plots (airmass, parallactic angle, sky maps).
- Determining observability of sets of targets given an arbitrary set of constraints (i.e., altitude, airmass, moon separation/illumination, etc.).
- Astropy powered!

Links

- Source code
- Docs
- Issues

License: BSD-3

General Documentation

- Installation

Figure 7: The official documentation of Astroplan.

Python Code 1: ai202209_s08_00_00.py

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

#
# Time-stamp: <2022/11/04 15:29:44 (CST) daisuke>
#

# importing astropy module
import astropy.units

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making an observer object
observer = astroplan.Observer (longitude=longitude, latitude=latitude, \
                               elevation=height, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)
```

Execute above script to construct an observer object.

```
% chmod a+x ai202209_s08_00_00.py
% ./ai202209_s08_00_00.py
```

```
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Try following practice.

Practice 08-01

Check the longitude, latitude, and elevation of Lulin Observatory and construct an observer object of Astroplan for Lulin Observatory using the function "`astroplan.Observer`".

Here is the other way to construct an observer object.

Python Code 2: ai202209_s08_00_01.py

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

#
# Time-stamp: <2022/11/04 15:37:50 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)
```

Execute above script to construct an observer object.

```
% chmod a+x ai202209_s08_00_01.py
% ./ai202209_s08_00_01.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Try following practice.

Practice 08-02

Check the longitude, latitude, and elevation of Lulin Observatory, construct a location object using Astropy, and then construct an observer object of Astroplan for Lulin Observatory.

For famous astronomical observatories, we are able to get the location object easily by using “`of_site`” of Astropy.

Python Code 3: ai202209_s08_00_02.py

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

#
# Time-stamp: <2022/11/04 15:46:44 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.of_site ('Palomar')

# printing created location object
print (location.geodetic)
```

Execute above script to construct a location object of Astropy.

```
% chmod a+x ai202209_s08_00_02.py
% ./ai202209_s08_00_02.py
Downloading http://data.astropy.org/coordinates/sites.json
|=========================================|  28k/ 28k (100.00%)         0s
GeodeticLocation(lon=<Longitude -116.863 deg>, lat=<Latitude 33.356 deg>, height
=<Quantity 1706. m>)
```

Try following practice.

Practice 08-03

Use the function “`astropy.coordinates.EarthLocation.of_site`” to construct a location object for Paranal Observatory.

Create a location object using Astropy, and then create an observer object of Astroplan.

Python Code 4: ai202209_s08_00_03.py

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

#
```

```
# Time-stamp: <2022/11/04 15:49:37 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.of_site ('Palomar')

# making an observer object
observer = astroplan.Observer (location=location, name='Palomar', \
                               timezone='US/Pacific')

# printing created observer object
print (observer)
```

Execute above script to construct a location object of Astropy and then construct an observer object of Astroplan.

```
% chmod a+x ai202209_s08_00_03.py
% ./ai202209_s08_00_03.py
<Observer: name='Palomar',
    location (lon, lat, el)=(-116.86300000000003 deg, 33.35600000000001 deg, 170
5.999999999058 m),
    timezone=<DstTzInfo 'US/Pacific' LMT-1 day, 16:07:00 STD>>
```

Try following practice.

Practice 08-04

Use the function "`astropy.coordinates.EarthLocation.of_site`" to construct a location object for Paranal Observatory, and then construct an observer object of Astroplan.

To obtain a list of available observatory site names, try following.

Python Code 5: ai202209_s08_00_04.py

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

#
# Time-stamp: <2022/11/04 15:53:02 (CST) daisuke>
#

# importing astropy module
import astropy.coordinates

# getting site names
site_names = astropy.coordinates.EarthLocation.get_site_names ()
```

```
# printing site names
print (f'Built-in site names of Astropy:')
for site_name in site_names:
    print (f'  {site_name}')
```

Execute above script to print a list of available observatory site names.

```
% chmod a+x ai202209_s08_00_04.py
% ./ai202209_s08_00_04.py
Built-in site names of Astropy:
  ALMA
  ATST
  Anglo-Australian Observatory
  Apache Point
  Apache Point Observatory
  Atacama Large Millimeter Array
  BAO
  BBSO
  Beijing XingLong Observatory
  Big Bear Solar Observatory
  Black Moshannon Observatory
  CHARA
  CHIME
  Canada-France-Hawaii Telescope
  Canadian Hydrogen Intensity Mapping Experiment
  Catalina Observatory
  Catalina Observatory: 61 inch telescope
  Cerro Pachon
  Cerro Paranal
  Cerro Tololo
  Cerro Tololo Interamerican Observatory
  Cima Ekar 182 cm Telescope
  Cima Ekar Observing Station
  DCT
  DKIST
  DRAO
  DRAO 26m Telescope
  Daniel K. Inouye Solar Telescope
  Discovery Channel Telescope
  Dominion Astrophysical Observatory
  Dominion Radio Astrophysical Observatory
  G1
  GBT
  GEO
  GEO600 Gravitational Wave Detector
  GEO_600
  Gemini North
  Gemini South
  Green Bank Telescope
  H1
  Hale Telescope
  Haleakala Observatories
  Happy Jack
  IAO
  Indian Astronomical Observatory
  JCMT
  James Clerk Maxwell Telescope
  Jansky Very Large Array
  John Galt Telescope
```

```
K1
KAGRA
Kamioka Gravitational Wave Detector
Keck Observatory
Kitt Peak
Kitt Peak National Observatory
L1
LHO
LHO_4k
LIGO Hanford Observatory
LIGO Livingston Observatory
LLO
LLO_4k
La Silla Observatory
La Silla Observatory (ESO)
Large Binocular Telescope
Las Campanas Observatory
Lick Observatory
Lowell Observatory
MDM Observatory
MWA
Manastash Ridge Observatory
McDonald Observatory
Medicina
Medicina Dish
Medicina Radio Telescope
Michigan-Dartmouth-MIT Observatory
Mount Graham International Observatory
Mount Wilson Observatory
Mt Graham
Mt. Ekar 182 cm Telescope
Mt. Stromlo Observatory
Multiple Mirror Telescope
Murchison Widefield Array
NASA Infrared Telescope Facility
NOV
NST
National Observatory of Venezuela
Noto
Noto Radio Telescope
Observatoire SIRENE
Observatoire de Haute Provence
Observatorio Astronomico Nacional, San Pedro Martir
Observatorio Astronomico Nacional, Tonantzintla
Owens Valley Radio Observatory
Palomar
Paranal Observatory
Paranal Observatory (ESO)
Roque de los Muchachos
Roque de los Muchachos, La Palma
Royal Observatory Greenwich
SAAO
SALT
SPO
SRT
Sac Peak
Sacramento Peak
Sacramento Peak Observatory
Sardinia Radio Telescope
```

```
  Siding Spring Observatory
  Southern African Large Telescope
  Subaru
  Subaru Telescope
  Sunspot
  Sutherland
  TUBITAK National Observatory
  TUG
  The Hale Telescope
  UKIRT
  United Kingdom Infrared Telescope
  V1
  VIRGO
  Vainu Bappu Observatory
  Very Large Array
  Virgo
  Virgo Observatory
  W. M. Keck Observatory
  Whipple
  Whipple Observatory
  aao
  alma
  apo
  bbso
  bmo
  cfht
  chime
  ctio
  dao
  dct
  dkist
  drao
  ekar
  example_site
  flwo
  gbt
  gemini_north
  gemini_south
  gemn
  gems
  geo_600
  greenwich
  haleakala
  iao
  irtf
  jcmt
  kagra
  keck
  kpno
  lapalma
  lasilla
  lbt
  lco
  lho_4k
  lick
  llo_4k
  lowell
  mcdonald
  mdm
  medicina
  mma
  mmt
  mro
  mso
  mtbigelow
  mwa
  mwo
  noto
  ohp
  ovro
  paranal
  salt
  sirene
  spm
  spo
  srt
  sso
  tona
  tug
  ukirt
  vbo
  virgo
  vla
```

There is an even more simple way to construct an observer object for well known observatories.

Python Code 6: ai202209_s08_00_05.py

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

#
# Time-stamp: <2022/11/04 15:58:51 (CST) daisuke>
#

# importing astroplan module
import astroplan

# making an observer object
observer = astroplan.Observer.at_site ('Palomar', timezone='US/Pacific')

# printing created observer object
print (observer)
```

Execute above script to construct an observer object of Astroplan.

```
% chmod a+x ai202209_s08_00_05.py
% ./ai202209_s08_00_05.py
<Observer: name='Palomar',
    location (lon, lat, el)=(-116.86300000000003 deg, 33.35600000000001 deg, 170
5.999999999058 m),
    timezone=<DstTzInfo 'US/Pacific' LMT-1 day, 16:07:00 STD>>
```

Try following practice.

Practice 08-05

Use the function "`astroplan.Observer.at_site`" to construct an observer object of Astroplan for Paranal Observatory.

4 Day and night

4.1 Local midnight

Find the time of local apparent solar midnight at NCU main campus nearest to 16:00:00 (UT) on 07/Nov/2022.

Python Code 7: ai202209_s08_01_00.py

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

#
# Time-stamp: <2022/11/04 21:20:58 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# local apparent solar midnight
midnight_guess = astropy.time.Time ('2022-11-07 16:00:00', \
                                    format='iso', scale='utc')
midnight = observer.midnight (midnight_guess, which='nearest', \
                              n_grid_points=500)

# printing local apparent solar midnight
print (f'local midnight = JD {midnight.jd} = {midnight.iso}')
```

Execute above script to find the time of local apparent solar midnight at NCU main campus nearest to 16:00:00 (UT) on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_00.py
% ./ai202209_s08_01_00.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
local midnight = JD 2459891.1520589865 = 2022-11-07 15:38:57.896
```

Try following practice.

Practice 08-06

Find the time of local apparent solar midnight at Lulin Observatory nearest to 16:00:00 (UT) on 01/Dec/2022.

At local midnight, the Sun is on the lower meridian. Therefore, azimuth is equal to zero and altitude is negative. Find the position of the Sun in horizontal coordinate system to check about it.

Python Code 8: ai202209_s08_01_01.py

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

#
# Time-stamp: <2022/11/04 21:21:08 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# local apparent solar midnight
midnight_guess = astropy.time.Time ('2022-11-07 16:00:00', \
                                    format='iso', scale='utc')
midnight = observer.midnight (midnight_guess, which='nearest', \
                              n_grid_points=500)

# printing local apparent solar midnight
```

```
print (f'local midnight = JD {midnight.jd} = {midnight.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', midnight, location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at local midnight:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the time of local apparent solar midnight at NCU main campus nearest to 16:00:00 (UT) on 07/Nov/2022 and find the altitude and azimuth of the Sun at local midnight.

```
% chmod a+x ai202209_s08_01_01.py
% ./ai202209_s08_01_01.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
local midnight = JD 2459891.1520589865 = 2022-11-07 15:38:57.896
position of the Sun at local midnight:
  alt = -81.42558478902902 deg
  az  = 0.0528237819280577 deg
```

Try following practice.

Practice 08-07

Find the time of local apparent solar midnight at Lulin Observatory nearest to 16:00:00 (UT) on 01/Dec/2022 and find altitude and azimuth of the Sun at local midnight.

4.2 Local noon

Find the time of local apparent solar noon at NCU main campus nearest to 04:00:00 (UT) on 07/Nov/2022.

Python Code 9: ai202209_s08_01_02.py

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

#
# Time-stamp: <2022/11/04 21:25:14 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m
```

```
# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# local apparent solar noon
noon_guess = astropy.time.Time ('2022-11-07 04:00:00', \
                                format='iso', scale='utc')
noon = observer.noon (noon_guess, which='nearest', n_grid_points=500)

# printing local apparent solar midnight
print (f'local noon = JD {noon.jd} = {noon.iso}')
```

Execute above script to find the time of local apparent solar noon at NCU main campus nearest to 04:00:00 (UT) on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_02.py
% ./ai202209_s08_01_02.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
local noon = JD 2459890.651919136 = 2022-11-07 03:38:45.813
```

Try following practice.

Practice 08-08

Find the time of local apparent solar noon at Keck Observatory nearest to 22:00:00 (UT) on 01/Jan/2023.

At local noon, the Sun is on the upper meridian. Therefore, azimuth is equal to 180° and altitude is positive. Find the position of the Sun in horizontal coordinate system to check about it.

Python Code 10: ai202209_s08_01_03.py

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

#
# Time-stamp: <2022/11/04 21:25:25 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# local apparent solar noon
noon_guess = astropy.time.Time ('2022-11-07 04:00:00', \
                                format='iso', scale='utc')
noon = observer.noon (noon_guess, which='nearest', n_grid_points=500)

# printing local apparent solar midnight
print (f'local noon = JD {noon.jd} = {noon.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', noon, location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at local noon:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the time of local apparent solar noon at NCU main campus nearest to 04:00:00 (UT) on 07/Nov/2022 and find the position of the Sun in horizontal coordinate system at local noon.

```
% chmod a+x ai202209_s08_01_03.py
% ./ai202209_s08_01_03.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

```
local noon = JD 2459890.651919136 = 2022-11-07 03:38:45.813
position of the Sun at local noon:
  alt = 48.78039319316061 deg
  az  = 179.9481676136466 deg
```

Try following practice.

Practice 08-09

Find the time of local apparent solar noon at Keck Observatory nearest to 22:00:00 (UT) on 01/Jan/2023 and find the position of the Sun in horizontal coordinate system at local noon.

Find the change of the altitude angle of the Sun at local apparent solar noon at NCU main campus in the course of a year.

Python Code 11: ai202209_s08_01_04.py

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

#
# Time-stamp: <2022/11/05 13:20:13 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# initial guess of local noon (501 days from 01/Nov/2022)
time_guess = astropy.time.Time ('2022-11-01 05:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day
```

```
# calculation of altitude angle at local noon
for i in range (len (time_guess)):
    # local apparent solar noon
    noon = observer.noon (time_guess[i], which='nearest', n_grid_points=500)

    # getting position of the Sun
    sun = astropy.coordinates.get_body ('sun', noon, location=location)

    # conversion from equatorial into horizontal
    altaz     = astropy.coordinates.AltAz (obstime=noon, location=location)
    sun_altaz = sun.transform_to (altaz)

    # printing result
    print (f'{noon.iso} {sun_altaz.alt}')
```

Execute above script to find the change of the altitude angle of the Sun at local apparent solar noon at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_04.py
% ./ai202209_s08_01_04.py
2022-11-01 03:38:42.029 50.62991922493444 deg
2022-11-02 03:38:40.648 50.311293291523505 deg
2022-11-03 03:38:40.059 49.996672428077666 deg
2022-11-04 03:38:40.272 49.68616634184924 deg
2022-11-05 03:38:41.296 49.3798817623621 deg
2022-11-06 03:38:43.141 49.077923065263626 deg
2022-11-07 03:38:45.816 48.78039319314837 deg
2022-11-08 03:38:49.331 48.48739465214193 deg
2022-11-09 03:38:53.692 48.1990303898877 deg
2022-11-10 03:38:58.907 47.91540440992845 deg

.....

2024-03-06 04:06:34.529 59.535245641854246 deg
2024-03-07 04:06:20.256 59.92423103526876 deg
2024-03-08 04:06:05.611 60.314365233867036 deg
2024-03-09 04:05:50.608 60.70553399325686 deg
2024-03-10 04:05:35.264 61.09762328928378 deg
2024-03-11 04:05:19.592 61.49051989449704 deg
2024-03-12 04:05:03.609 61.88411128850944 deg
2024-03-13 04:04:47.329 62.278285266885874 deg
2024-03-14 04:04:30.768 62.6729298838904 deg
2024-03-15 04:04:13.944 63.067933981610075 deg
```

Try following practice.

Practice 08-10

Find the change of the altitude angle of the Sun at local apparent solar noon at Siding Spring Observatory in Australia in the course of a year.

Find the change of the altitude angle of the Sun at local apparent solar noon at NCU main campus in the course of a year and make a plot of the altitude angle change in the course of a year.

Python Code 12: ai202209_s08_01_05.py

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

#
# Time-stamp: <2022/11/05 22:47:58 (CST) daisuke>
```

```
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

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

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'sun_maxalt_ncu.png'

# empty numpy arrays for plotting
dates = numpy.array ([])
alts  = numpy.array ([])

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# initial guess of local noon (501 days from 01/Nov/2022)
time_guess = astropy.time.Time ('2022-11-01 05:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# calculation of altitude angle at local noon
for i in range (len (time_guess)):
    # local apparent solar noon
    noon = observer.noon (time_guess[i], which='nearest', n_grid_points=500)

    # getting position of the Sun
    sun = astropy.coordinates.get_body ('sun', noon, location=location)
    # conversion from equatorial into horizontal
    altaz     = astropy.coordinates.AltAz (obstime=noon, location=location)
    sun_altaz = sun.transform_to (altaz)

    # printing result
    print (f'{noon.iso} {sun_altaz.alt}')

    # appending data to numpy arrays
    dates = numpy.append (dates, noon.plot_date)
    alts  = numpy.append (alts, sun_altaz.alt.value)

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Altitude Angle [deg]')
ax.set_title ('Change of Maximum Altitude of the Sun')

# plotting data
ax.plot (dates, alts, linestyle='-', linewidth=3.0, color='red', \
         label='NCU, Taiwan')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the altitude angle of the Sun at local apparent solar noon at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_05.py
% ./ai202209_s08_01_05.py
2022-11-01 03:38:42.029 50.62991922493444 deg
2022-11-02 03:38:40.648 50.311293291523505 deg
2022-11-03 03:38:40.059 49.996672428077666 deg
2022-11-04 03:38:40.272 49.68616634184924 deg
2022-11-05 03:38:41.296 49.3798817623621 deg
2022-11-06 03:38:43.141 49.07792306526362 6 deg
2022-11-07 03:38:45.816 48.78039319314837 deg
2022-11-08 03:38:49.331 48.48739465214193 deg
2022-11-09 03:38:53.692 48.1990303898877 deg
2022-11-10 03:38:58.907 47.91540440992845 deg
```

```
.....

2024-03-06 04:06:34.529 59.535245641854246 deg
2024-03-07 04:06:20.256 59.92423103526876 deg
2024-03-08 04:06:05.611 60.314365233867036 deg
2024-03-09 04:05:50.608 60.70553399325686 deg
2024-03-10 04:05:35.264 61.09762328928378 deg
2024-03-11 04:05:19.592 61.49051989449704 deg
2024-03-12 04:05:03.609 61.88411128850944 deg
2024-03-13 04:04:47.329 62.278285266885874 deg
2024-03-14 04:04:30.768 62.6729298838904 deg
2024-03-15 04:04:13.944 63.067933981610075 deg
```

Display created PNG file. (Fig. 8)

```
% feh -dF sun_maxalt_ncu.png
```

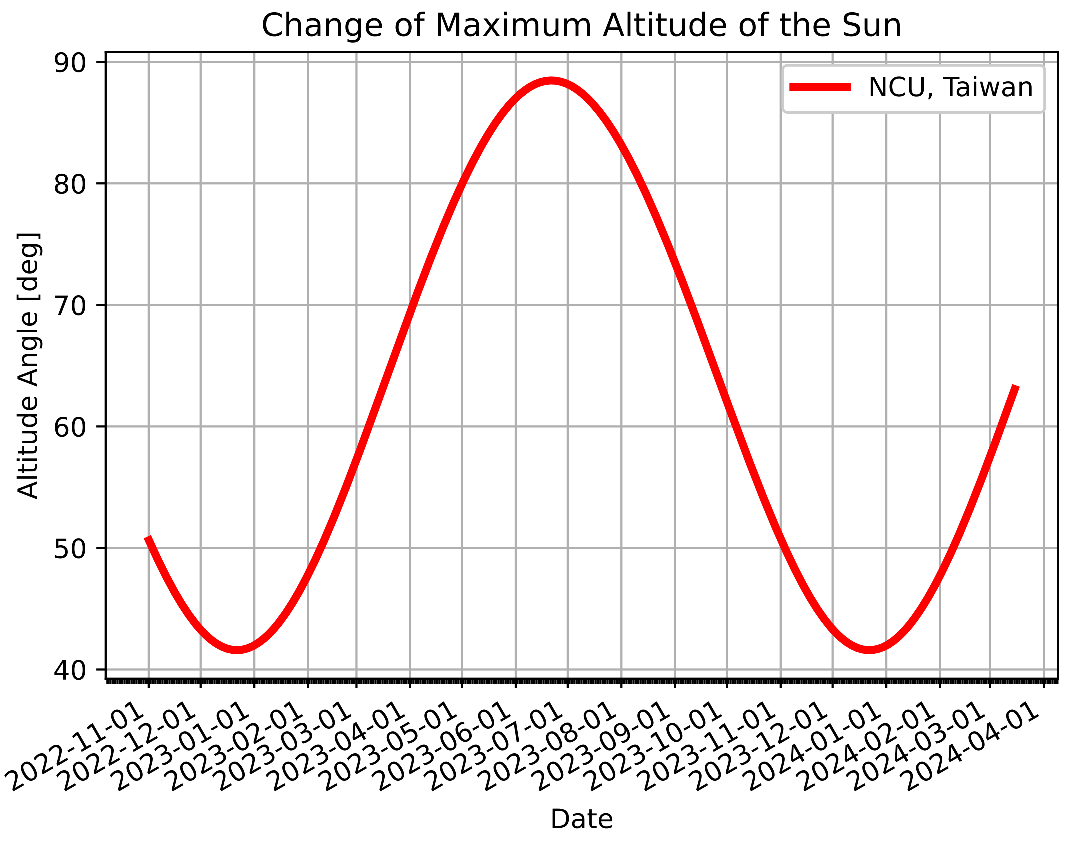


Figure 8: The change of the altitude angle of the Sun as observed at NCU main campus at the local noon in the course of a year.

Try following practice.

Practice 08-11

Plot the change of the altitude angle of the Sun at local apparent solar noon at La Silla Observatory in Chile in the course of a year.

Find the change of the altitude angle of the Sun at local apparent solar noon at some astronomical observatories in the course of a year and make a plot of the altitude angle change in the course of a year.

Python Code 13: ai202209_s08_01_06.py

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

#
# Time-stamp: <2022/11/05 19:43:20 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

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

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'sun_maxalt_world.png'

# locations
observatories = {
    'uppsala': {
        'longitude': '+17d38m13.1s',
        'latitude' : '+59d51m35.5s',
        'height'   : 37.9 * u_m,
        'name'     : 'Uppsala',
    },
    'greenwich': {
        'longitude': '-0d00m05s',
        'latitude' : '+51d28m40s',
        'height'   : 45.0 * u_m,
        'name'     : 'Greenwich',
    },
    'vatican': {
        'longitude': '+12d39m02s',
        'latitude' : '+41d44m50s',
        'height'   : 430.0 * u_m,
        'name'     : 'Vatican',
    },
    'palomar': {
        'longitude': '-116d51m54s',
        'latitude' : '+33d21m23s',
        'height'   : 1712.0 * u_m,
        'name'     : 'Palomar',
    },
    'chiangmai': {
        'longitude': '+98d29m12s',
        'latitude' : '+18d35m26s',
        'height'   : 2457.0 * u_m,
        'name'     : 'Chiang-Mai',
    },
    'paranal': {
        'longitude': '-70d24m15s',
        'latitude' : '-24d37m38s',
        'height'   : 2635.0 * u_m,
        'name'     : 'Paranal',
    },
    'sidingspring': {
        'longitude': '+149d03m52s',
        'latitude' : '-31d16m24s',
        'height'   : 1165.0 * u_m,
        'name'     : 'Siding Spring',
    },
}

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Altitude Angle [deg]')
ax.set_title ('Change of Maximum Altitude of the Sun')

# calculations for each observatory site
for site in observatories.keys ():
    # printing status
    print (f'Now, calculating for {observatories[site]["name"]}...')

    # empty numpy arrays for plotting
    dates = numpy.array ([])
    alts  = numpy.array ([])

    # making a location object using Astropy
    location \
        = astropy.coordinates.EarthLocation.from_geodetic \
        (lon=observatories[site]['longitude'], \
         lat=observatories[site]['latitude'], \
         height=observatories[site]['height'])

    # making an observer object
    observer = astroplan.Observer (location=location, name=site)

    # initial guess of local noon (501 days from 01/Nov/2022)
    time_guess = astropy.time.Time ('2022-11-01 05:00:00', \
                                    format='iso', scale='utc') \
                                    + numpy.linspace (0.0, 500.0, 501) \
                                    * u_day

    # calculation of altitude angle at local noon
    for i in range (len (time_guess)):
        # local apparent solar noon
        noon = observer.noon (time_guess[i], which='nearest', \
                              n_grid_points=200)

        # getting position of the Sun
        sun = astropy.coordinates.get_body ('sun', noon, location=location)

        # conversion from equatorial into horizontal
        altaz = astropy.coordinates.AltAz (obstime=noon, location=location)
        sun_altaz = sun.transform_to (altaz)

        # appending data to numpy arrays
        dates = numpy.append (dates, noon.plot_date)
        alts  = numpy.append (alts, sun_altaz.alt.value)


    # plotting data
    ax.plot (dates, alts, linestyle='-', linewidth=3.0, \
             label=observatories[site]['name'])

# legend
ax.legend (loc='upper right')

# grid
ax.grid ()

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the altitude angle of the Sun at local apparent solar noon at some astronomical observatories in the course of a year.

```
% chmod a+x ai202209_s08_01_06.py
% ./ai202209_s08_01_06.py
Now, calculating for Uppsala...
Now, calculating for Greenwich...
Now, calculating for Vatican...
Now, calculating for Palomar...
Now, calculating for Chiang-Mai...
Now, calculating for Paranal...
Now, calculating for Siding Spring...
```

Display created PNG file. (Fig. 9)

```
% feh -dF sun_maxalt_world.png
```

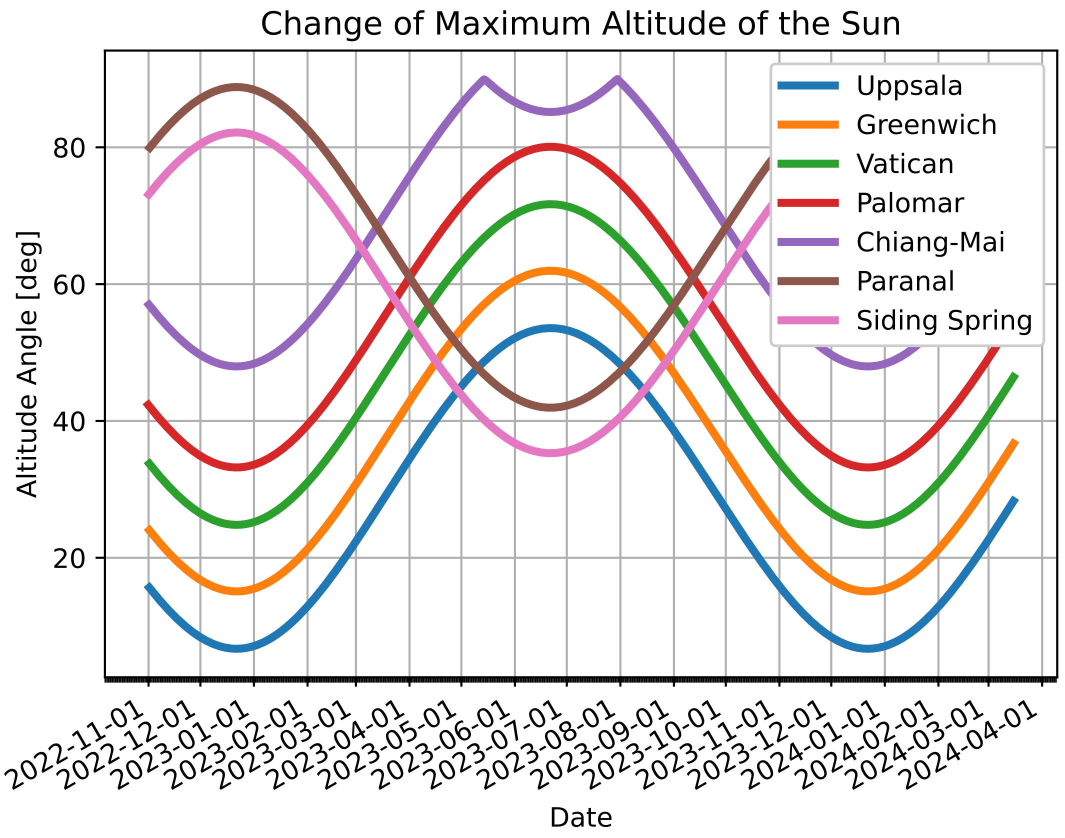


Figure 9: The change of the altitude angle of the Sun as observed at some astronomical observatories in the world at the local noon in the course of a year.

Try following practice.

Practice 08-12

Plot the change of the altitude angle of the Sun at local apparent solar noon at Paris Observatory, Kitt Peak National Observatory, Keck Observatory, Sutherland Observatory in the course of a year.

4.3 Sunset time

Find the sunset time at NCU main campus on 07/Nov/2022. Here is an example Python script.

Python Code 14: ai202209_s08_01_07.py

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

#
# Time-stamp: <2022/11/05 19:51:43 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# sunset time
sunset_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                  format='iso', scale='utc')
sunset = observer.sun_set_time (sunset_guess, which='nearest', \
                                n_grid_points=500)

# printing sunset time
print (f'sunset = JD {sunset.jd} = {sunset.iso}')
```

Execute above script to find the sunset time at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_07.py
% ./ai202209_s08_01_07.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
sunset = JD 2459890.8802561956 = 2022-11-07 09:07:34.135
```

Try following practice.

Practice 08-13

Find the sunset time at Bosscha Observatory in Indonesia on 01/Jan/2023.

Find the position of the Sun as observed at NCU main campus at the time of the sunset on 07/Nov/2022. At the sunset, the altitude of the Sun should be zero.

Python Code 15: ai202209_s08_01_08.py

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

#
# Time-stamp: <2022/11/05 19:51:52 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# sunset time
sunset_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                  format='iso', scale='utc')
```

```
sunset = observer.sun_set_time (sunset_guess, which='nearest', \
                                n_grid_points=500)

# printing sunset time
print (f'sunset = JD {sunset.jd} = {sunset.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', sunset, location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at sunset:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the sunset time at NCU main campus on 07/Nov/2022 and find the position of the Sun on the sky.

```
% chmod a+x ai202209_s08_01_08.py
% ./ai202209_s08_01_08.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
sunset = JD 2459890.8802561956 = 2022-11-07 09:07:34.135
position of the Sun at sunset:
  alt = 6.11685759113656e-05 deg
  az  = 251.94616273888153 deg
```

Try following practice.

Practice 08-14

Find the sunset time at Bosscha Observatory in Indonesia on 01/Jan/2023 and find the position of the Sun on the sky at the time of the sunset.

Find the change of azimuth angle of the Sun as observed at NCU main campus at the time of the sunset in the course of a year.

Python Code 16: ai202209_s08_01_09.py

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

#
# Time-stamp: <2022/11/05 20:15:06 (CST) daisuke>
#

# importing numpy module
import numpy

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

```
# importing astroplan module
import astroplan

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

# output file name
file_output = 'sun_sunset_az_ncu.png'

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# empty arrays for plotting
dates = numpy.array ([])
azs   = numpy.array ([])

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# sunset time
time_guess = astropy.time.Time ('2022-11-01 09:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# calculation of azimuth angle at sunset
for i in range (len (time_guess)):
    # printing status
    if (i % 50 == 0):
        print (f'Now, calculating sunset on {time_guess[i]}...')

    # sunset time
    sunset = observer.sun_set_time (time_guess[i], which='nearest', \
                                    n_grid_points=500)

    # getting position of the Sun
    sun = astropy.coordinates.get_body ('sun', sunset, location=location)

    # conversion from equatorial into horizontal
    altaz     = astropy.coordinates.AltAz (obstime=sunset, location=location)
    sun_altaz = sun.transform_to (altaz)

    # appending data to numpy arrays
    dates = numpy.append (dates, sunset.plot_date)
    azs   = numpy.append (azs, sun_altaz.az.value)

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Azimuth Angle [deg]')
ax.set_title ('Change of azimuth of the Sun at sunset')

# plotting data
ax.plot (dates, azs, linestyle='-', linewidth=3.0, color='red', \
         label='NCU, Taiwan')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot the change of azimuth angle of the Sun at the time of the sunset in the course of a year.

```
% chmod a+x ai202209_s08_01_09.py
% ./ai202209_s08_01_09.py
Now, calculating sunset on 2022-11-01 09:00:00.000...
Now, calculating sunset on 2022-12-21 09:00:00.000...
Now, calculating sunset on 2023-02-09 09:00:00.000...
Now, calculating sunset on 2023-03-31 09:00:00.000...
Now, calculating sunset on 2023-05-20 09:00:00.000...
Now, calculating sunset on 2023-07-09 09:00:00.000...
Now, calculating sunset on 2023-08-28 09:00:00.000...
Now, calculating sunset on 2023-10-17 09:00:00.000...
Now, calculating sunset on 2023-12-06 09:00:00.000...
Now, calculating sunset on 2024-01-25 09:00:00.000...
Now, calculating sunset on 2024-03-15 09:00:00.000...
```

Display created PNG file. (Fig. 10)

```
% feh -dF sun_sunset_az_ncu.png
```

Try following practice.

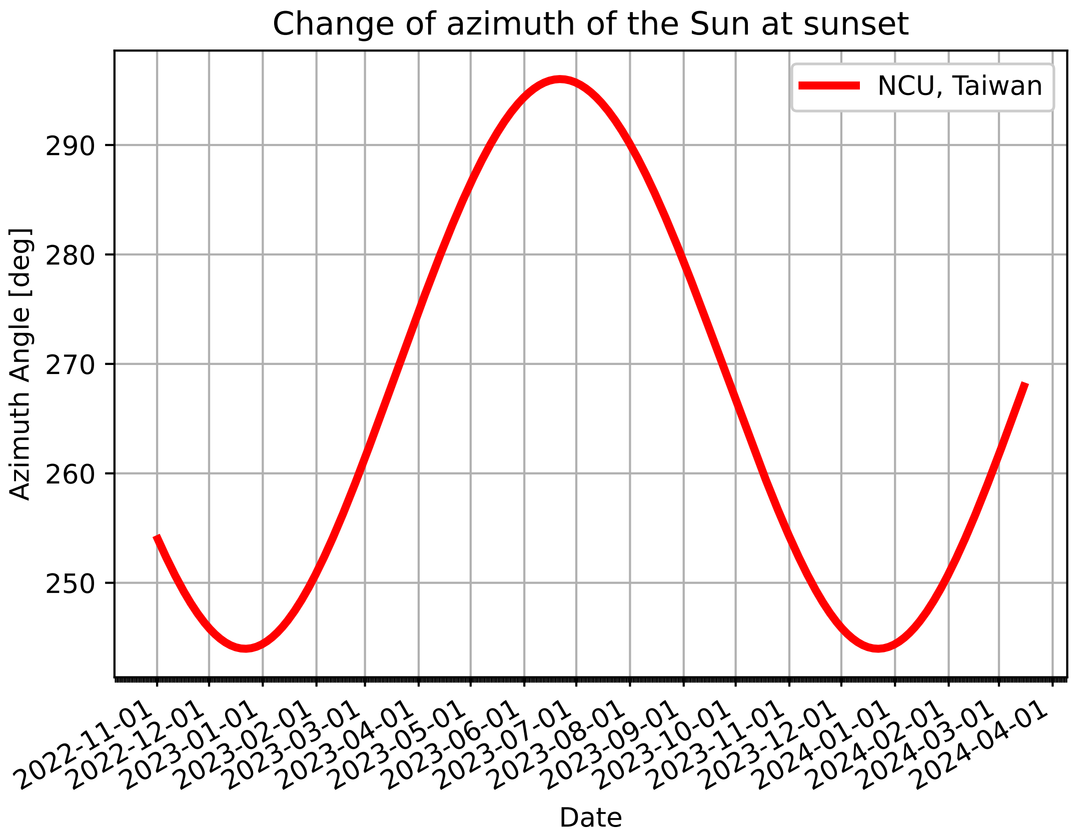


Figure 10: The change of the azimuth angle of the Sun as observed at NCU main campus at sunset in the course of a year.

Practice 08-15

Find the change of azimuth angle of the Sun as observed at Mauna Kea at the time of the sunset in the course of a year.

4.4 Sunrise time

Find the sunrise time at NCU main campus on 08/Nov/2022.

Python Code 17: ai202209_s08_01_10.py

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

#
# Time-stamp: <2022/11/05 21:58:42 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# sunrise time
sunrise_guess = astropy.time.Time ('2022-11-08 00:00:00', \
                                   format='iso', scale='utc')
sunrise = observer.sun_rise_time (sunrise_guess, which='nearest', \
                                  n_grid_points=500)

# printing sunrise time
print (f'sunrise = JD {sunrise.jd} = {sunrise.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', sunrise, location=location)
```

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

# printing altitude and azimuth
print (f'position of the Sun at sunrise:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the sunrise time at NCU main campus on 08/Nov/2022.

```
% chmod a+x ai202209_s08_01_10.py
% ./ai202209_s08_01_10.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
sunrise = JD 2459891.424043474 = 2022-11-07 22:10:37.356
position of the Sun at sunrise:
  alt = 8.228366847482508e-05 deg
  az  = 108.23117842782295 deg
```

Try following practice.

Practice 08-16

Find the sunrise time at Greenwich Observatory on 15/Dec/2022.

Plot the change of azimuth angle of the Sun at the sunrise as observed in Uppsala Observatory in Sweden in the course of a year.

Python Code 18: ai202209_s08_01_11.py

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

#
# Time-stamp: <2022/11/05 22:51:24 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

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

# output file name
file_output = 'sun_sunrise_az_uppsala.png'
```

```
# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# empty arrays for plotting
dates = numpy.array ([])
azs   = numpy.array ([])

# location of observer: Uppsala Observatory
longitude = '+17d38m13.1s'
latitude  = '+59d51m35.5s'
height    = 37.9 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='Uppsala')

# sunset time
time_guess = astropy.time.Time ('2022-11-01 09:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# calculation of azimuth angle at sunrise
for i in range (len (time_guess)):
    # printing status
    if (i % 50 == 0):
        print (f'Now, calculating sunrise on {time_guess[i]}...')

    # sunrise time
    sunrise = observer.sun_rise_time (time_guess[i], which='nearest', \
                                      n_grid_points=500)

    # getting position of the Sun
    sun = astropy.coordinates.get_body ('sun', sunrise, location=location)

    # conversion from equatorial into horizontal
    altaz = astropy.coordinates.AltAz (obstime=sunrise, location=location)
    sun_altaz = sun.transform_to (altaz)

    # appending data to numpy arrays
    dates = numpy.append (dates, sunrise.plot_date)
    azs   = numpy.append (azs, sun_altaz.az.value)

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Azimuth Angle [deg]')
ax.set_title ('Change of azimuth of the Sun at sunrise')
```

```
# plotting data
ax.plot (dates, azs, linestyle='-', linewidth=3.0, color='red', \
         label='Uppsala')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot the change of azimuth angle of the Sun at the sunrise as observed in Uppsala Observatory in Sweden in the course of a year.

```
% chmod a+x ai202209_s08_01_11.py
% ./ai202209_s08_01_11.py
Now, calculating sunrise on 2022-11-01 09:00:00.000...
Now, calculating sunrise on 2022-12-21 09:00:00.000...
Now, calculating sunrise on 2023-02-09 09:00:00.000...
Now, calculating sunrise on 2023-03-31 09:00:00.000...
Now, calculating sunrise on 2023-05-20 09:00:00.000...
Now, calculating sunrise on 2023-07-09 09:00:00.000...
Now, calculating sunrise on 2023-08-28 09:00:00.000...
Now, calculating sunrise on 2023-10-17 09:00:00.000...
Now, calculating sunrise on 2023-12-06 09:00:00.000...
Now, calculating sunrise on 2024-01-25 09:00:00.000...
Now, calculating sunrise on 2024-03-15 09:00:00.000...
```

Display created PNG file. (Fig. 11)

```
% feh -dF sun_sunrise_az_uppsala.png
```

Try following practice.

Practice 08-17

Plot the change of azimuth angle of the Sun at the sunrise as observed in Las Campanas Observatory in Chile in the course of a year.

4.5 Nighttime length

Find the nighttime length at NCU main campus on 07/Nov/2022.

Python Code 19: ai202209_s08_01_12.py

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

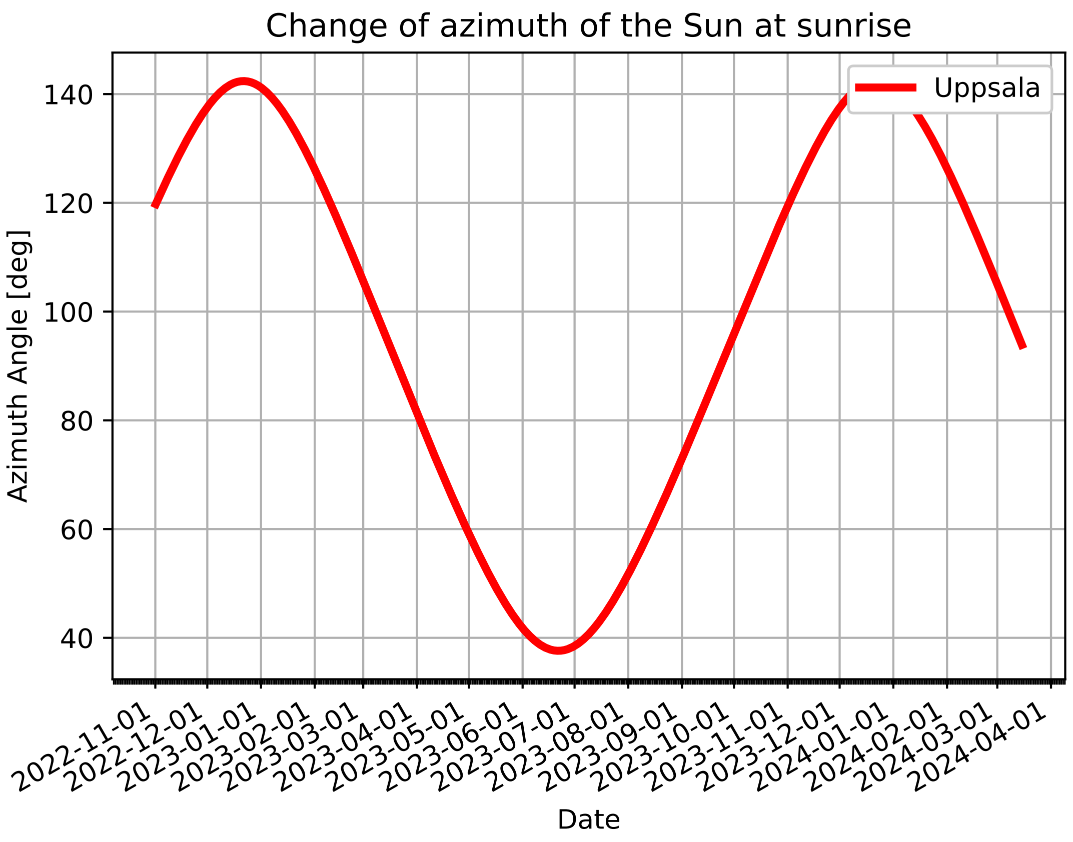


Figure 11: The change of the azimuth angle of the Sun as observed at Uppsala Observatory at sunrise in the course of a year.

```
#
# Time-stamp: <2022/11/05 23:12:42 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# sunset time
sunset_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                  format='iso', scale='utc')
sunset = observer.sun_set_time (sunset_guess, which='nearest', \
                                n_grid_points=500)

# sunrise time
sunrise_guess = astropy.time.Time ('2022-11-08 00:00:00', \
                                   format='iso', scale='utc')
sunrise = observer.sun_rise_time (sunrise_guess, which='nearest', \
                                  n_grid_points=500)

# nighttime length
night_length_day = sunrise.jd - sunset.jd
night_length_hr  = night_length_day * 24.0

# printing sunset time, sunrise time, and length of nighttime
print (f'sunset  = JD {sunset.jd:14.6f} = {sunset.iso}')
print (f'sunrise = JD {sunrise.jd:14.6f} = {sunrise.iso}')
print (f'nighttime length = {night_length_day} day = {night_length_hr} hr')
```

Execute above script to find the nighttime length at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_12.py
% ./ai202209_s08_01_12.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
sunset  = JD 2459890.880256 = 2022-11-07 09:07:34.135
sunrise = JD 2459891.424043 = 2022-11-07 22:10:37.356
nighttime length = 0.5437872782349586 day = 13.050894677639008 hr
```

Try following practice.

Practice 08-18

Find the nighttime length at Mount Stromlo Observatory on 01/Dec/2022.

Find the change of the nighttime length at NCU main campus in the course of a year.

Python Code 20: ai202209_s08_01_13.py

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

#
# Time-stamp: <2022/11/06 10:58:35 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

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

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
```

```
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# sunset time and sunrise time
time_guess = astropy.time.Time ('2022-11-01 16:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day
sunset = observer.sun_set_time (time_guess, which='nearest', \
                                n_grid_points=500)
sunrise = observer.sun_rise_time (time_guess, which='nearest', \
                                  n_grid_points=500)

# nighttime length
night_length_hr = (sunrise.jd - sunset.jd) * 24.0

# printing date/time and length of nighttime
for i in range (len (time_guess)):
    print (f'{time_guess[i]} {night_length_hr[i]:8.5f} hr')
```

Execute above script to find the change of the nighttime length at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_13.py
% ./ai202209_s08_01_13.py
2022-11-01 16:00:00.000 12.92631 hr
2022-11-02 16:00:00.000 12.94761 hr
2022-11-03 16:00:00.000 12.96870 hr
2022-11-04 16:00:00.000 12.98958 hr
2022-11-05 16:00:00.000 13.01024 hr
2022-11-06 16:00:00.000 13.03068 hr
2022-11-07 16:00:00.000 13.05088 hr
2022-11-08 16:00:00.000 13.07084 hr
2022-11-09 16:00:00.000 13.09055 hr
2022-11-10 16:00:00.000 13.11000 hr

.....

2024-03-06 16:00:00.000 12.32842 hr
2024-03-07 16:00:00.000 12.30397 hr
2024-03-08 16:00:00.000 12.27949 hr
2024-03-09 16:00:00.000 12.25498 hr
2024-03-10 16:00:00.000 12.23044 hr
2024-03-11 16:00:00.000 12.20588 hr
2024-03-12 16:00:00.000 12.18130 hr
2024-03-13 16:00:00.000 12.15671 hr
2024-03-14 16:00:00.000 12.13212 hr
2024-03-15 16:00:00.000 12.10753 hr
```

Try following practice.

Practice 08-19

Find the change of the nighttime length at Mount Stromlo Observatory in the course of a year.

Plot the change of the nighttime length at NCU main campus in the course of a year.

Python Code 21: ai202209_s08_01_14.py

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

#
```

```
# Time-stamp: <2022/11/06 10:59:09 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

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

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'nighttime_length_ncu.png'

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# sunset time and sunrise time
time_guess = astropy.time.Time ('2022-11-01 16:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day
sunset = observer.sun_set_time (time_guess, which='nearest', \
                                n_grid_points=500)
sunrise = observer.sun_rise_time (time_guess, which='nearest', \
                                  n_grid_points=500)

# nighttime length
night_length_hr = (sunrise.jd - sunset.jd) * 24.0

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

```
# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Nighttime length [hr]')
ax.set_title ('Change of nighttime length')

# plotting data
ax.plot (time_guess.plot_date, night_length_hr, \
         linestyle='-', linewidth=3.0, color='red', label='NCU, Taiwan')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the nighttime length at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_14.py
% ./ai202209_s08_01_14.py
```

Display created PNG file. (Fig. 12)

```
% feh -dF nighttime_length_ncu.png
```

Try following practice.

Practice 08-20

Make a plot of the change of the nighttime length at Mount Stromlo Observatory in the course of a year.

Plot the change of the nighttime length at Greenwich Observatory in the course of a year.

Python Code 22: ai202209_s08_01_15.py

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

#
# Time-stamp: <2022/11/06 16:11:14 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
```

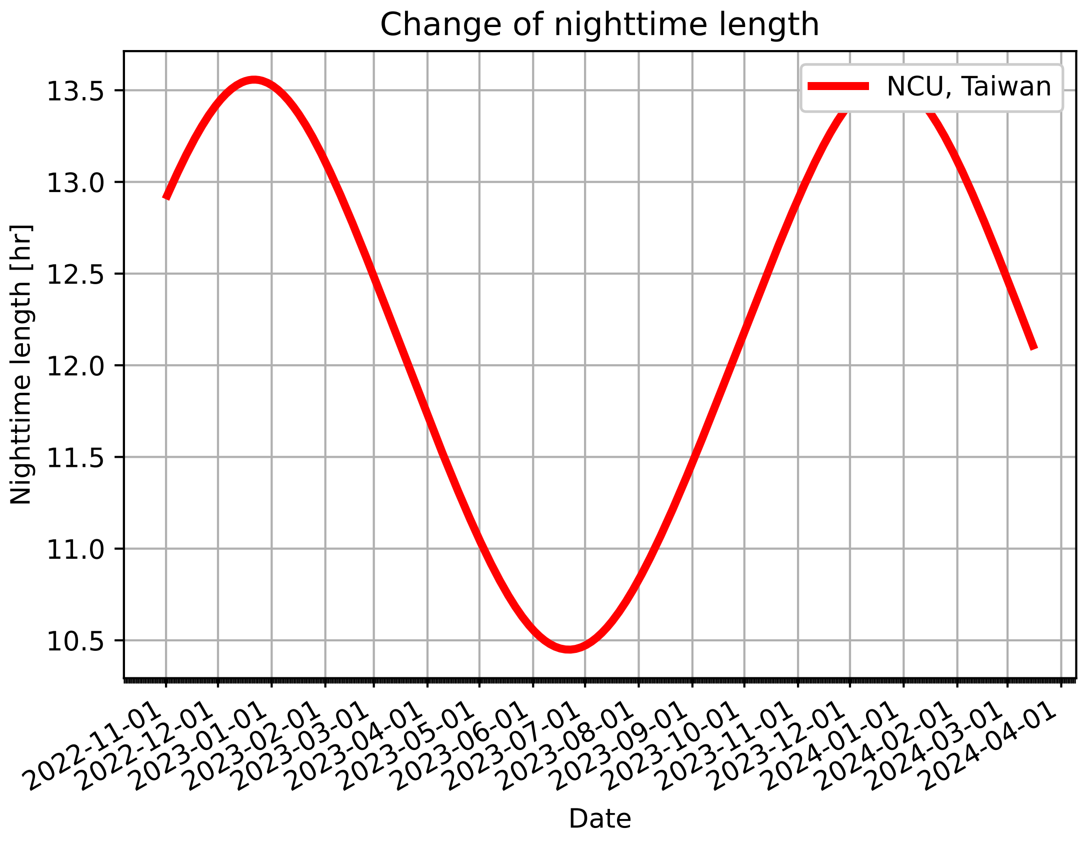


Figure 12: The change of the nighttime length at NCU main campus in the course of a year.

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

# importing astroplan module
import astroplan

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

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'nighttime_length_greenwich.png'

# location of observer: Greenwich Observatory
longitude = '-0d00m05s'
latitude  = '+51d28m40s'
height    = 45.0 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='Greenwich', \
                               timezone='UTC')

# sunset time and sunrise time
time_guess = astropy.time.Time ('2022-11-01 23:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day
sunset = observer.sun_set_time (time_guess, which='nearest', \
                                n_grid_points=500)
sunrise = observer.sun_rise_time (time_guess, which='nearest', \
                                  n_grid_points=500)

# nighttime length
night_length_hr = (sunrise.jd - sunset.jd) * 24.0

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Nighttime length [hr]')
ax.set_title ('Change of nighttime length')

# plotting data
```

```
ax.plot (time_guess.plot_date, night_length_hr, \
         linestyle='-', linewidth=3.0, color='red', label='Greenwich')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the nighttime length at Greenwich Observatory in the course of a year.

```
% chmod a+x ai202209_s08_01_15.py
% ./ai202209_s08_01_15.py
```

Display created PNG file. (Fig. 13)

```
% feh -dF nighttime_length_greenwich.png
```

Try following practice.

Practice 08-21

Make a plot of the change of the nighttime length at Bosscha Observatory in the course of a year.

4.6 Twilight

Find the end of evening astronomical twilight at NCU main campus on 07/Nov/2022.

Python Code 23: ai202209_s08_01_16.py

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

#
# Time-stamp: <2022/11/06 16:02:55 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
```

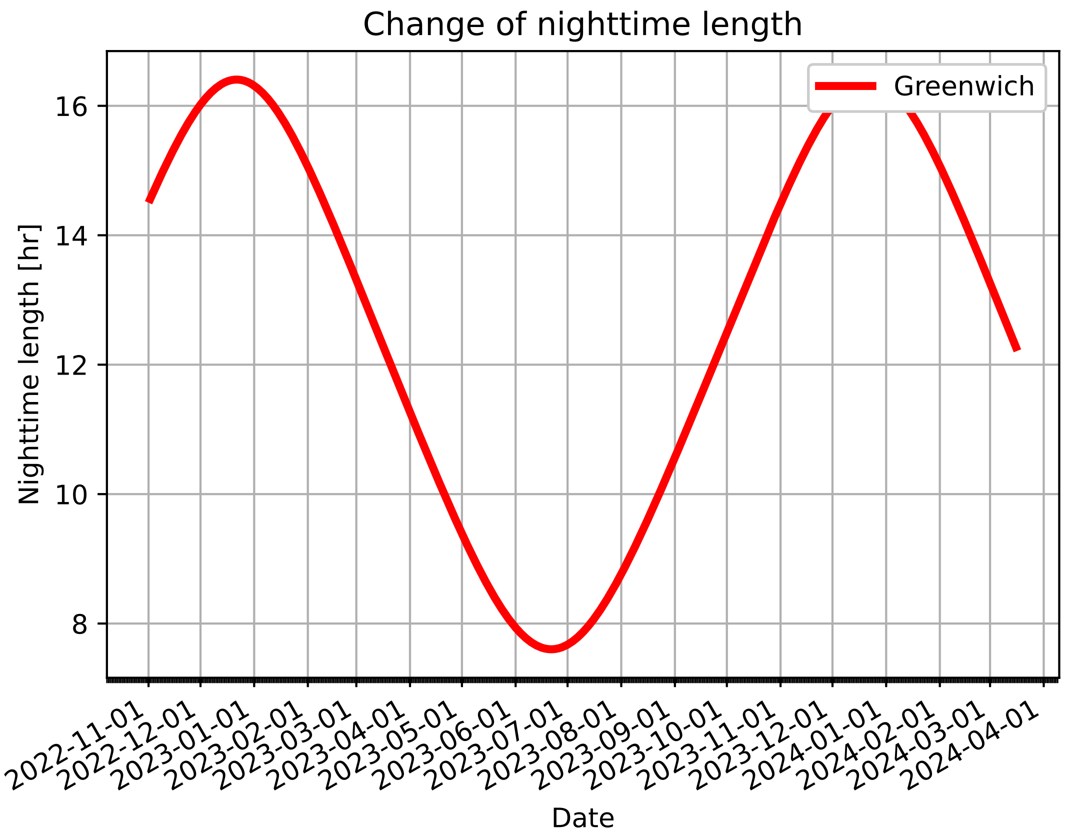


Figure 13: The change of the nighttime length at Greenwich Observatory in the course of a year.

```
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# end of evening twilight
twilight_evening_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                            format='iso', scale='utc')
twilight_evening = observer.twilight_evening_astronomical \
    (twilight_evening_guess, which='nearest', n_grid_points=500)

# printing end of evening twilight
print (f'end of evening twilight = JD {twilight_evening.jd}', \
       f'= {twilight_evening.iso}')
```

Execute above script to find the end of evening astronomical twilight at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_16.py
% ./ai202209_s08_01_16.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
end of evening twilight = JD 2459890.9371209065 = 2022-11-07 10:29:27.246
```

Try following practice.

Practice 08-22

Find the end of evening astronomical twilight at Lulin Observatory on 07/Nov/2022.

The end of evening astronomical twilight is defined to be the time that the Sun is at altitude of $-18°$. Find the altitude of the Sun at the end of evening astronomical twilight.

Python Code 24: ai202209_s08_01_17.py

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

#
# Time-stamp: <2022/11/06 16:05:18 (CST) daisuke>
#

# importing astropy module
```

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                              lat=latitude, \
                                                              height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# end of evening twilight
twilight_evening_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                            format='iso', scale='utc')
twilight_evening = observer.twilight_evening_astronomical \
    (twilight_evening_guess, which='nearest', n_grid_points=500)

# printing end of evening twilight
print (f'end of evening twilight = JD {twilight_evening.jd}', \
       f'= {twilight_evening.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', twilight_evening, \
                                    location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at the end of evening twilight:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the end of evening astronomical twilight at NCU main campus on 07/Nov/2022 and find the altitude of the Sun at the end of evening astronomical twilight.

```
% chmod a+x ai202209_s08_01_17.py
% ./ai202209_s08_01_17.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
end of evening twilight = JD 2459890.9371209065 = 2022-11-07 10:29:27.246
position of the Sun at the end of evening twilight:
  alt = -17.999952887023777 deg
  az  = 259.92863934669657 deg
```

Try following practice.

Practice 08-23

Find the end of evening astronomical twilight at Lulin Observatory on 07/Nov/2022, and find the altitude of the Sun at the end of evening astronomical twilight.

Find the end of evening astronomical twilight at Greenwich Observatory on 07/Nov/2022, and find the altitude of the Sun at the end of evening astronomical twilight.

Python Code 25: ai202209_s08_01_18.py

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

#
# Time-stamp: <2022/11/06 16:12:25 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: Greenwich Observatory
longitude = '-0d00m05s'
latitude  = '+51d28m40s'
height    = 45.0 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='Greenwich', \
                               timezone='UTC')

# printing created observer object
print (observer)
```

```
# end of evening twilight
twilight_evening_guess = astropy.time.Time ('2022-11-07 18:00:00', \
                                            format='iso', scale='utc')
twilight_evening = observer.twilight_evening_astronomical \
    (twilight_evening_guess, which='nearest', n_grid_points=500)

# printing end of evening twilight
print (f'end of evening twilight = JD {twilight_evening.jd}', \
       f'= {twilight_evening.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', twilight_evening, \
                                    location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at the end of evening twilight:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to find the end of evening astronomical twilight at Greenwich Observatory on 07/Nov/2022, and find the altitude of the Sun at the end of evening astronomical twilight.

```
% chmod a+x ai202209_s08_01_18.py
% ./ai202209_s08_01_18.py
<Observer: name='Greenwich',
    location (lon, lat, el)=(-0.0013888888868678 deg, 51.47777777777779 deg, 4
5.00000000011866 m),
    timezone=<UTC>>
end of evening twilight = JD 2459891.2621301706 = 2022-11-07 18:17:28.047
position of the Sun at the end of evening twilight:
  alt = -17.999992442938236 deg
  az  = 266.02454946165534 deg
```

Try following practice.

Practice 08-24

Find the end of evening astronomical twilight at ALMA Observatory on 20/Dec/2022, and find the altitude of the Sun at the end of evening astronomical twilight.

Find the start of morning astronomical twilight at NCU main campus on 08/Nov/2022.

Python Code 26: ai202209_s08_01_19.py

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

#
# Time-stamp: <2022/11/06 16:28:40 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# start of morning twilight
twilight_morning_guess = astropy.time.Time ('2022-11-07 22:00:00', \
                                            format='iso', scale='utc')
twilight_morning = observer.twilight_morning_astronomical \
    (twilight_morning_guess, which='nearest', n_grid_points=500)

# printing start of morning twilight
print (f'start of morning twilight = JD {twilight_morning.jd}', \
       f'= {twilight_morning.iso}')
```

Execute above script to find the start of morning astronomical twilight at NCU main campus on 08/Nov/2022.

```
% chmod a+x ai202209_s08_01_19.py
% ./ai202209_s08_01_19.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
start of morning twilight = JD 2459891.36709625 = 2022-11-07 20:48:37.116
```

Try following practice.

Practice 08-25

Find the start of morning astronomical twilight at Lulin Observatory on 08/Nov/2022.

Check the altitude of the Sun at the start of morning twilight.

Python Code 27: ai202209_s08_01_20.py

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

```
#
# Time-stamp: <2022/11/06 16:28:31 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing created observer object
print (observer)

# start of morning twilight
twilight_morning_guess = astropy.time.Time ('2022-11-07 22:00:00', \
                                            format='iso', scale='utc')
twilight_morning = observer.twilight_morning_astronomical \
    (twilight_morning_guess, which='nearest', n_grid_points=500)

# printing start of morning twilight
print (f'start of morning twilight = JD {twilight_morning.jd}', \
       f'= {twilight_morning.iso}')

# getting position of the Sun
sun = astropy.coordinates.get_body ('sun', twilight_morning, \
                                    location=location)

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

# printing altitude and azimuth
print (f'position of the Sun at the start of morning twilight:')
print (f'  alt = {sun_alt}')
print (f'  az  = {sun_az}')
```

Execute above script to check the altitude of the Sun at the start of morning twilight.

```
% chmod a+x ai202209_s08_01_20.py
% ./ai202209_s08_01_20.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
start of morning twilight = JD 2459891.36709625 = 2022-11-07 20:48:37.116
position of the Sun at the start of morning twilight:
  alt = -17.999922798937916 deg
  az  = 100.2136806292227 deg
```

Try following practice.

Practice 08-26

Find the altitude of the Sun at Kitt Peak Observatory at the start of morning astronomical twilight on 15/Jan/2023.

Find the end of evening twilight and start of morning twilight at NCU main campus, and find the length of observable time on 07/Nov/2022.

Python Code 28: ai202209_s08_01_21.py

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

#
# Time-stamp: <2022/11/06 16:36:41 (CST) daisuke>
#

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

# importing astroplan module
import astroplan

# units
u_m = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')
```

```
# printing created observer object
print (observer)

# end of evening twilight
twilight_evening_guess = astropy.time.Time ('2022-11-07 10:00:00', \
                                             format='iso', scale='utc')
twilight_evening = observer.twilight_evening_astronomical \
    (twilight_evening_guess, which='nearest', n_grid_points=500)

# start of morning twilight
twilight_morning_guess = astropy.time.Time ('2022-11-07 22:00:00', \
                                             format='iso', scale='utc')
twilight_morning = observer.twilight_morning_astronomical \
    (twilight_morning_guess, which='nearest', n_grid_points=500)

# length of observable time
obs_time = (twilight_morning.jd - twilight_evening.jd) * 24.0

# printing result
print (f'end of evening twilight = JD {twilight_evening.jd}', \
       f'= {twilight_evening.iso}')
print (f'start of morning twilight = JD {twilight_morning.jd}', \
       f'= {twilight_morning.iso}')
print (f'length of observable time = {obs_time} hr')
```

Execute above script to find the end of evening twilight and start of morning twilight at NCU main campus, and find the length of observable time on 07/Nov/2022.

```
% chmod a+x ai202209_s08_01_21.py
% ./ai202209_s08_01_21.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
end of evening twilight = JD 2459890.9371209065 = 2022-11-07 10:29:27.246
start of morning twilight = JD 2459891.36709625 = 2022-11-07 20:48:37.116
length of observable time = 10.319408241659403 hr
```

Try following practice.

Practice 08-27

Find the length of observable time at Paris Observatory on 01/Dec/2022.

Find the change of the length of observable time at NCU main campus in the course of a year.

Python Code 29: ai202209_s08_01_22.py

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

#
# Time-stamp: <2022/11/06 16:44:48 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_day = astropy.units.day

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# time
time_guess = astropy.time.Time ('2022-11-01 16:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# end of evening twilight
twilight_evening = observer.twilight_evening_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# start of morning twilight
twilight_morning = observer.twilight_morning_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# length of observable time
obs_time = (twilight_morning.jd - twilight_evening.jd) * 24.0

# printing result
for i in range (len (time_guess)):
    print (f'{time_guess[i]} {obs_time[i]:8.5f} hr')
```

Execute above script to find the change of the length of observable time at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_22.py
% ./ai202209_s08_01_22.py
2022-11-01 16:00:00.000 10.21741 hr
2022-11-02 16:00:00.000 10.23508 hr
2022-11-03 16:00:00.000 10.25249 hr
2022-11-04 16:00:00.000 10.26963 hr
2022-11-05 16:00:00.000 10.28649 hr
2022-11-06 16:00:00.000 10.30308 hr
2022-11-07 16:00:00.000 10.31939 hr
2022-11-08 16:00:00.000 10.33543 hr
2022-11-09 16:00:00.000 10.35117 hr
```

```
2022-11-10 16:00:00.000 10.36662 hr

.....

2024-03-06 16:00:00.000  9.67789 hr
2024-03-07 16:00:00.000  9.65395 hr
2024-03-08 16:00:00.000  9.62983 hr
2024-03-09 16:00:00.000  9.60552 hr
2024-03-10 16:00:00.000  9.58103 hr
2024-03-11 16:00:00.000  9.55636 hr
2024-03-12 16:00:00.000  9.53152 hr
2024-03-13 16:00:00.000  9.50650 hr
2024-03-14 16:00:00.000  9.48133 hr
2024-03-15 16:00:00.000  9.45598 hr
```

Try following practice.

Practice 08-28

Find the change of the length of observable time at Okayama Astrophysical Observatory in Japan in the course of a year.

Make a plot of the change of the length of observable time at NCU main campus in the course of a year.

Python Code 30: ai202209_s08_01_23.py

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

#
# Time-stamp: <2022/11/06 17:38:07 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_day = astropy.units.day

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

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'obstime_length_ncu.png'

# location of observer: NCU main campus
longitude = '121d11m12s'
```

```
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# time
time_guess = astropy.time.Time ('2022-11-01 16:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# end of evening twilight
twilight_evening = observer.twilight_evening_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# start of morning twilight
twilight_morning = observer.twilight_morning_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# length of observable time
obs_time = (twilight_morning.jd - twilight_evening.jd) * 24.0

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Nighttime length [hr]')
ax.set_title ('Change of observable time length')

# plotting data
ax.plot (time_guess.plot_date, obs_time, \
         linestyle='-', linewidth=3.0, color='red', label='NCU, Taiwan')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the length of observable time at NCU main campus in the course of a year.

```
% chmod a+x ai202209_s08_01_23.py
% ./ai202209_s08_01_23.py
```

Display created PNG file. (Fig. 14)

```
% feh -dF obstime_length_ncu.png
```

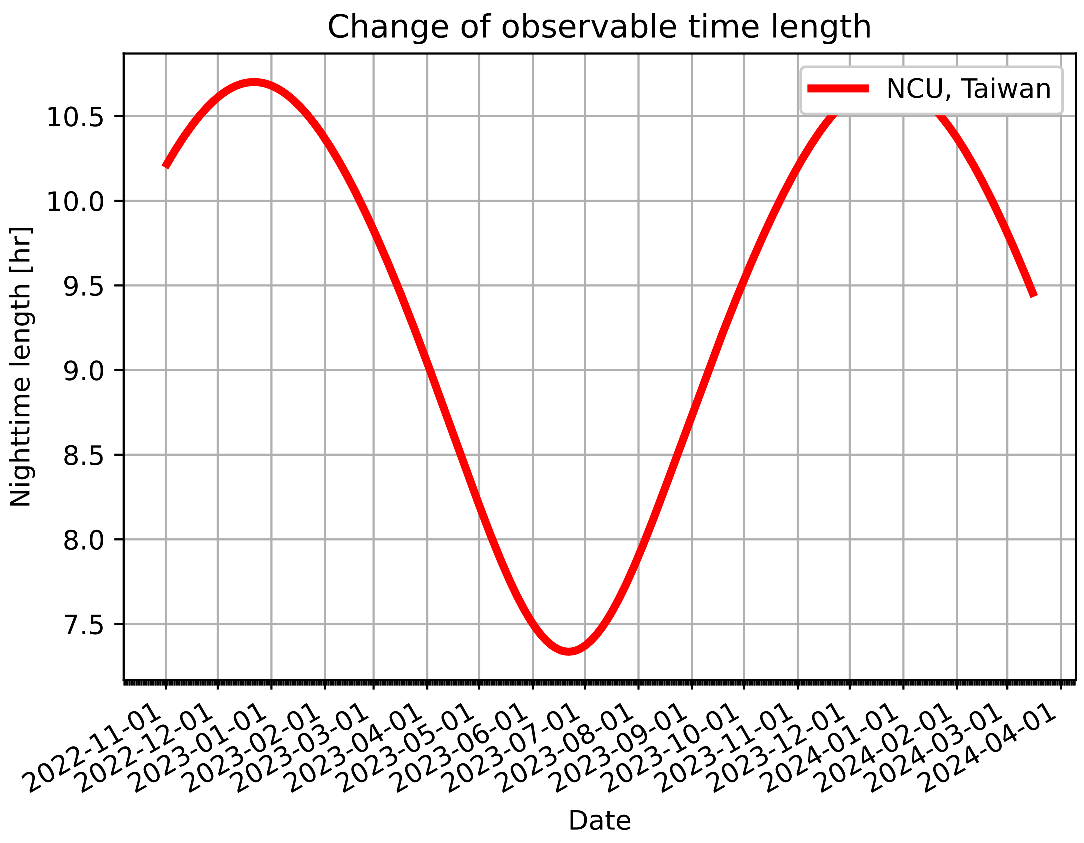


Figure 14: The change of the nighttime length at NCU main campus in the course of a year.

Try following practice.

Practice 08-29

Make a plot of the change of the length of observable time at Okayama Astrophysical Observatory in Japan in the course of a year.

Make a plot of the change of the length of observable time at Pic du Midi Observatory in the course of a year.

Python Code 31: ai202209_s08_01_24.py

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

#
```

```
# Time-stamp: <2022/11/06 20:17:26 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_day = astropy.units.day

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

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# output file name
file_output = 'obstime_length_picdumidi.png'

# location of observer: Pic du Midi Observatory
longitude = '+0d08m34s'
latitude  = '+42d56m11s'
height    = 2877.0 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='Pic du Midi', \
                               timezone='CET')

# time
time_guess = astropy.time.Time ('2022-11-01 01:00:00', \
                                format='iso', scale='utc') \
                                + numpy.linspace (0.0, 500.0, 501) * u_day

# end of evening twilight
twilight_evening = observer.twilight_evening_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# start of morning twilight
twilight_morning = observer.twilight_morning_astronomical \
    (time_guess, which='nearest', n_grid_points=500)

# length of observable time
obs_time = (twilight_morning.jd - twilight_evening.jd) * 24.0
```

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

# labels
ax.set_xlabel ('Date')
ax.set_ylabel ('Nighttime length [hr]')
ax.set_title ('Change of observable time length')

# plotting data
ax.plot (time_guess.plot_date, obs_time, \
         linestyle='-', linewidth=3.0, color='red', label='Pic du Midi')
ax.grid ()

# legend
ax.legend (loc='upper right')

# ticks
months     = matplotlib.dates.MonthLocator ()
days       = matplotlib.dates.DayLocator ()
months_fmt = matplotlib.dates.DateFormatter ('%Y-%m-%d')
ax.xaxis.set_major_locator (months)
ax.xaxis.set_major_formatter (months_fmt)
ax.xaxis.set_minor_locator (days)

# formatting labels
fig.autofmt_xdate()

# saving file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a plot of the change of the length of observable time at Pic du Midi Observatory in the course of a year.

```
% chmod a+x ai202209_s08_01_24.py
% ./ai202209_s08_01_24.py
```

Display created PNG file. (Fig. 15)

```
% feh -dF obstime_length_picdumidi.png
```

Try following practice.

Practice 08-30

Make a plot of the change of the length of observable time at Leiden Observatory in the course of a year.

Check whether it is night time at given location and date/time.

Python Code 32: ai202209_s08_01_25.py

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

#
# Time-stamp: <2022/11/06 20:27:23 (CST) daisuke>
#
```

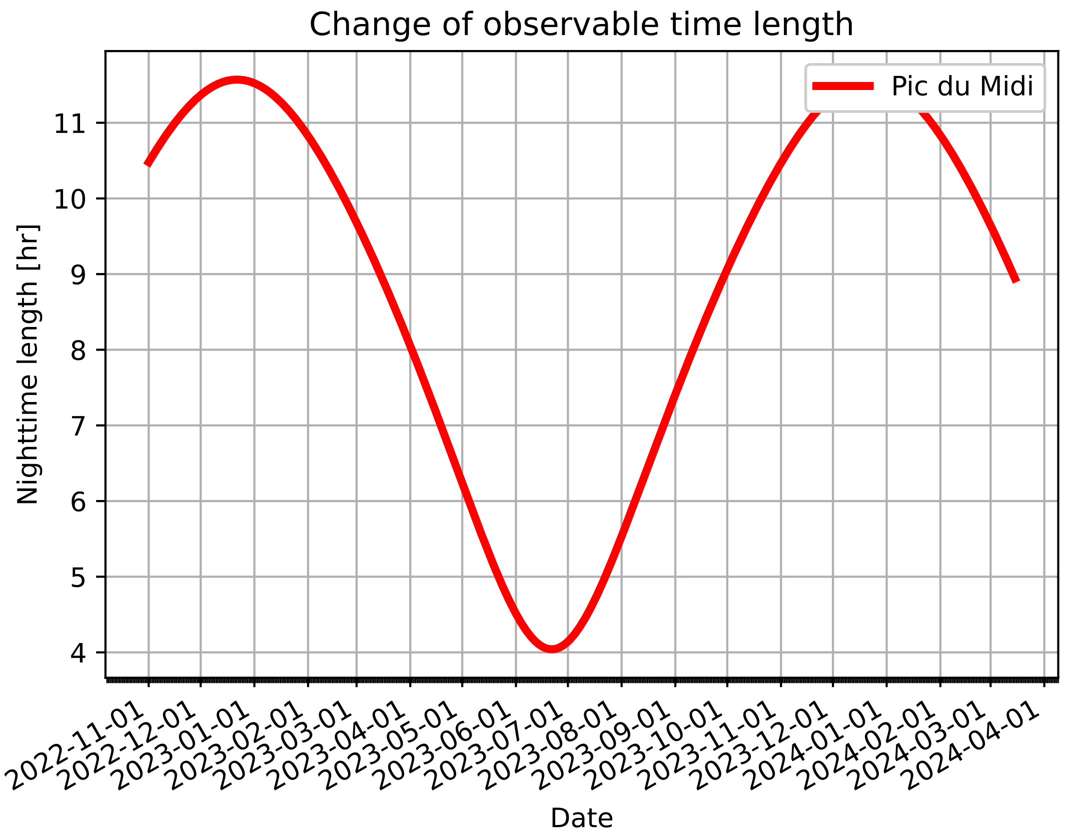


Figure 15: The change of the nighttime length at Pic du Midi Observatory in the course of a year.

```
# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_hr  = astropy.units.hour

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# date/time
times = astropy.time.Time ('2022-11-07 04:00:00', format='iso', \
                           scale='utc') + numpy.linspace (0, 20, 21) * u_hr

# is it night time?
night = observer.is_night (times)

# printing results
print (f'# night time at NCU?')
for i in range (len (times)):
    print (f'{times[i]} ==> {night[i]}')
```

Execute above script to check whether it is night time at given location and date/time.

```
% chmod a+x ai202209_s08_01_25.py
% ./ai202209_s08_01_25.py
# night time at NCU?
2022-11-07 04:00:00.000 ==> False
2022-11-07 05:00:00.000 ==> False
2022-11-07 06:00:00.000 ==> False
2022-11-07 07:00:00.000 ==> False
2022-11-07 08:00:00.000 ==> False
2022-11-07 09:00:00.000 ==> False
2022-11-07 10:00:00.000 ==> True
2022-11-07 11:00:00.000 ==> True
2022-11-07 12:00:00.000 ==> True
2022-11-07 13:00:00.000 ==> True
```

```
2022-11-07 14:00:00.000 ==> True
2022-11-07 15:00:00.000 ==> True
2022-11-07 16:00:00.000 ==> True
2022-11-07 17:00:00.000 ==> True
2022-11-07 18:00:00.000 ==> True
2022-11-07 19:00:00.000 ==> True
2022-11-07 20:00:00.000 ==> True
2022-11-07 21:00:00.000 ==> True
2022-11-07 22:00:00.000 ==> True
2022-11-07 23:00:00.000 ==> False
2022-11-08 00:00:00.000 ==> False
```

Try following practice.

Practice 08-31

Check whether it is at night at Mauna Kea at 05:00:00 (UT) on 10/Jan/2023.

5 Moon

5.1 Moon rise time

Find the Moon rise time at NCU main campus on 07/Nov/2022.

Python Code 33: ai202209_s08_02_00.py

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

#
# Time-stamp: <2022/11/06 20:39:07 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m    = astropy.units.m

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)
```

```
# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing observer object
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')

# moon rise time
moonrise = observer.moon_rise_time (time, which='nearest', \
                                    n_grid_points=500)

# printing result
print (f'moon rise time = {moonrise.iso}')
```

Execute above script to find the Moon rise time at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_02_00.py
% ./ai202209_s08_02_00.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
moon rise time = 2022-11-07 08:32:26.628
```

Try following practice.

Practice 08-32

Find the Moon rise time at Lowell Observatory on 01/Feb/2023.

5.2 Moon set time

Find the Moon set time at NCU main campus on 08/Nov/2022.

Python Code 34: ai202209_s08_02_01.py

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

#
# Time-stamp: <2022/11/06 20:47:01 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
```

```
# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing observer object
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 20:00:00', format='iso', scale='utc')

# moon set time
moonset = observer.moon_set_time (time, which='nearest', \
                                  n_grid_points=500)

# printing result
print (f'moon set time = {moonset.iso}')
```

Execute above script to find the Moon set time at NCU main campus on 08/Nov/2022.

```
% chmod a+x ai202209_s08_02_01.py
% ./ai202209_s08_02_01.py
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
moon set time = 2022-11-07 21:35:48.652
```

Try following practice.

Practice 08-33

Find the Moon set time at Roque de los Muchachos Observatory on 01/Mar/2023.

5.3 Moon phase

Find the moon phase at 11:00:00 (UT) on 08/Nov/2022 as observed at NCU main campus.

Python Code 35: ai202209_s08_02_02.py

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

#
# Time-stamp: <2022/11/06 21:02:33 (CST) daisuke>
#

# importing numpy module
```

```
import numpy

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

# importing astroplan module
import astroplan

# units
u_m   = astropy.units.m
u_deg = astropy.units.degree

# using "DE440" for solar system ephemeris
astropy.coordinates.solar_system_ephemeris.set ('de440')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# date/time
time = astropy.time.Time ('2022-11-08 11:00:00', format='iso', scale='utc')

# moon illumination
illumination = observer.moon_illumination (time)

# moon phase
phase_rad = observer.moon_phase (time)
phase_deg = phase_rad.to (u_deg)

# printing result
print (f'time: {time}')
print (f'  illumination fraction = {illumination:5.3f}')
print (f'  moon phase            = {phase_deg:6.2f}')
```

Execute above script to find the Moon phase at 11:00:00 (UT) on 08/Nov/2022.

```
% chmod a+x ai202209_s08_02_02.py
% ./ai202209_s08_02_02.py
time: 2022-11-08 11:00:00.000
  illumination fraction = 1.000
  moon phase            =   0.24 deg
```

Moon phase of 0 deg corresponds to full moon, and moon phase of 180 deg corresponds to new moon. Try following practice.

Practice 08-34

Find the moon phase at 00:00:00 (UT) on 24/Nov/2022 as observed at NCU main campus.

6 Stars

6.1 Getting coordinate of a fixed target

Make a Python script to get the coordinate of a fixed target. Here is an example for getting the coordinate of Canopus.

Python Code 36: ai202209_s08_03_00.py

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

#
# Time-stamp: <2022/11/06 21:27:46 (CST) daisuke>
#

# importing astroplan module
import astroplan

# getting coordinate of a fixed target
canopus = astroplan.FixedTarget.from_name ('Canopus')

# printing coordinate
print (f'{canopus}')
```

Execute above script to get the coordinate of Canopus.

```
% chmod a+x ai202209_s08_03_00.py
% ./ai202209_s08_03_00.py
<FixedTarget "Canopus" at SkyCoord (ICRS): (ra, dec) in deg (95.98795783, -52.69
566138)>
```

Try following practice.

Practice 08-35

Get the coordinate of Arcturus.

Get the coordinate of Crab Nebula.

Python Code 37: ai202209_s08_03_01.py

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

#
# Time-stamp: <2022/11/06 21:34:51 (CST) daisuke>
#

# importing astroplan module
import astroplan

# getting coordinate of a fixed target
crab_nebula = astroplan.FixedTarget.from_name ('M1')

# printing coordinate
print (f'{crab_nebula}')
```

Execute above script to get the coordinate of Crab Nebula (M1).

```
% chmod a+x ai202209_s08_03_01.py
% ./ai202209_s08_03_01.py
<FixedTarget "M1" at SkyCoord (ICRS): (ra, dec) in deg (83.6333, 22.0133)>
```

Try following practice.

Practice 08-36

Get the coordinate of Cassiopeia A.

Get the coordinate of Omega Centauri.

Python Code 38: ai202209_s08_03_02.py

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

#
# Time-stamp: <2022/11/06 21:38:40 (CST) daisuke>
#

# importing astroplan module
import astroplan

# getting coordinate of a fixed target
omega_centauri = astroplan.FixedTarget.from_name ('NGC 5139')

# printing coordinate
print (f'{omega_centauri}')
```

Execute above script to get the coordinate of Omega Centauri (NGC 5139).

```
% chmod a+x ai202209_s08_03_02.py
% ./ai202209_s08_03_02.py
<FixedTarget "NGC 5139" at SkyCoord (ICRS): (ra, dec) in deg (201.697, -47.47947
2)>
```

Try following practice.

Practice 08-37

Get the coordinate of the open cluster Praesepe (M44).

Get the coordinate of Andromeda Galaxy (M31).

Python Code 39: ai202209_s08_03_03.py

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

#
# Time-stamp: <2022/11/06 21:45:42 (CST) daisuke>
#

# importing astroplan module
import astroplan

# getting coordinate of a fixed target
m31 = astroplan.FixedTarget.from_name ('Andromeda Galaxy')

# printing coordinate
print (f'{m31}')
```

Execute above script to get the coordinate of Andromeda Galaxy (M31).

```
% chmod a+x ai202209_s08_03_03.py
% ./ai202209_s08_03_03.py
<FixedTarget "Andromeda Galaxy" at SkyCoord (ICRS): (ra, dec) in deg (10.6847083
, 41.26875)>
```

Try following practice.

Practice 08-37

Get the coordinate of Sombrero Galaxy (M104).

6.2 Visibility of a target

Make a Python script to check whether a target is above the horizon or not.

Python Code 40: ai202209_s08_03_04.py

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

#
# Time-stamp: <2022/11/06 22:04:01 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# getting coordinate of a fixed target
canopus = astroplan.FixedTarget.from_name ('Canopus')

# printing coordinate
print (f'{canopus}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')
```

```
# printing location
print (f'{observer}')

# date/time
times = astropy.time.Time ('2022-11-07 08:00:00', format='iso', \
                           scale='utc') + numpy.linspace (0, 18, 19) * u_hr

# is above the horizon?
up = observer.target_is_up (times, canopus)

# printing results
for i in range (len (times)):
    print (f'{times[i]} ==> {up[i]}')
```

Execute above script to check whether Canopus is above the horizon or not at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_03_04.py
% ./ai202209_s08_03_04.py
<FixedTarget "Canopus" at SkyCoord (ICRS): (ra, dec) in deg (95.98795783, -52.69
566138)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
2022-11-07 08:00:00.000 ==> False
2022-11-07 09:00:00.000 ==> False
2022-11-07 10:00:00.000 ==> False
2022-11-07 11:00:00.000 ==> False
2022-11-07 12:00:00.000 ==> False
2022-11-07 13:00:00.000 ==> False
2022-11-07 14:00:00.000 ==> False
2022-11-07 15:00:00.000 ==> False
2022-11-07 16:00:00.000 ==> True
2022-11-07 17:00:00.000 ==> True
2022-11-07 18:00:00.000 ==> True
2022-11-07 19:00:00.000 ==> True
2022-11-07 20:00:00.000 ==> True
2022-11-07 21:00:00.000 ==> True
2022-11-07 22:00:00.000 ==> True
2022-11-07 23:00:00.000 ==> False
2022-11-08 00:00:00.000 ==> False
2022-11-08 01:00:00.000 ==> False
2022-11-08 02:00:00.000 ==> False
```

Try following practice.

Practice 08-38

Check whether Regulus is above the horizon or not at Mauna Kea on 25/Nov/2022.

Find the rise time of a target.

Python Code 41: ai202209_s08_03_05.py

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

#
# Time-stamp: <2022/11/06 22:05:03 (CST) daisuke>
#
```

```
# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# getting coordinate of a fixed target
canopus = astroplan.FixedTarget.from_name ('Canopus')

# printing coordinate
print (f'{canopus}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
times = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')

# rise time
risetime = observer.target_rise_time (times, canopus, which='nearest', \
                                      n_grid_points=500)

# printing results
print (f'Rise time of Canopus = {risetime.iso}')
```

Execute above script to find the rise time of Canopus at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_03_05.py
% ./ai202209_s08_03_05.py
<FixedTarget "Canopus" at SkyCoord (ICRS): (ra, dec) in deg (95.98795783, -52.69
566138)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

```
Rise time of Canopus = 2022-11-07 15:43:11.168
```

Try following practice.

Practice 08-39

Find the rise time of Regulus at Lulin Observatory on 01/Jan/2023.

Find the set time of a target.

Python Code 42: ai202209_s08_03_06.py

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

#
# Time-stamp: <2022/11/06 22:10:04 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# getting coordinate of a fixed target
canopus = astroplan.FixedTarget.from_name ('Canopus')

# printing coordinate
print (f'{canopus}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
times = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')

# rise time
```

```
settime = observer.target_set_time (times, canopus, which='nearest', \
                                    n_grid_points=500)

# printing results
print (f'Set time of Canopus = {settime.iso}')
```

Execute above script to find the set time of Canopus at NCU main campus on 08/Nov/2022.

```
% chmod a+x ai202209_s08_03_06.py
% ./ai202209_s08_03_06.py
<FixedTarget "Canopus" at SkyCoord (ICRS): (ra, dec) in deg (95.98795783, -52.69
566138)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
Set time of Canopus = 2022-11-07 22:40:32.828
```

Try following practice.

Practice 08-40

Find the set time of Regulus at Lulin Observatory on 02/Jan/2023.

Find the meridian transit time of Canopus at NCU main campus on 07/Nov/2022.

Python Code 43: ai202209_s08_03_07.py

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

#
# Time-stamp: <2022/11/06 22:14:35 (CST) daisuke>
#

# importing numpy module
import numpy

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

# importing astroplan module
import astroplan

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# getting coordinate of a fixed target
canopus = astroplan.FixedTarget.from_name ('Canopus')

# printing coordinate
print (f'{canopus}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
```

```
# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
times = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')

# meridian transit time
transittime = observer.target_meridian_transit_time (times, canopus, \
                                                     which='nearest', \
                                                     n_grid_points=500)

# printing results
print (f'Meridian transit time of Canopus = {transittime.iso}')
```

Execute above script to find the meridian transit time of Canopus at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_03_07.py
% ./ai202209_s08_03_07.py
<FixedTarget "Canopus" at SkyCoord (ICRS): (ra, dec) in deg (95.98795783, -52.69
566138)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
Meridian transit time of Canopus = 2022-11-07 19:11:52.009
```

Try following practice.

Practice 08-41

Find the meridian time of Regulus at Lulin Observatory on 15/Jan/2023.

7 Airmass

Make a plot of airmass changes of targets.

Python Code 44: ai202209_s08_04_00.py

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

#
# Time-stamp: <2022/11/06 23:32:38 (CST) daisuke>
#

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

```
# importing astroplan module
import astroplan
import astroplan.plots

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

# units
u_m  = astropy.units.m

# output file name
file_output = 'airmass_plot_00.png'

# getting coordinate of a fixed target
fomalhaut = astroplan.FixedTarget.from_name ('Fomalhaut')
sirius    = astroplan.FixedTarget.from_name ('Sirius')
m31       = astroplan.FixedTarget.from_name ('M31')

# printing coordinate
print (f'{fomalhaut}')
print (f'{sirius}')
print (f'{m31}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                             lat=latitude, \
                                                             height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')

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

# plotting airmass change
astroplan.plots.plot_airmass (fomalhaut, observer, time, ax=ax)
astroplan.plots.plot_airmass (sirius, observer, time, ax=ax)
astroplan.plots.plot_airmass (m31, observer, time, ax=ax)

# legend
ax.legend ()

# saving file
```

```
fig.savefig (file_output, dpi=225, bbox_inches='tight')
```

Execute above script to make a plot of airmass changes of Fomalhaut, Sirius, and M31 as observed at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_04_00.py
% ./ai202209_s08_04_00.py
<FixedTarget "Fomalhaut" at SkyCoord (ICRS): (ra, dec) in deg (344.41269272, -29
.62223703)>
<FixedTarget "Sirius" at SkyCoord (ICRS): (ra, dec) in deg (101.28715533, -16.71
611586)>
<FixedTarget "M31" at SkyCoord (ICRS): (ra, dec) in deg (10.68470833, 41.26875)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Display created PNG file. (Fig. 16)

```
% feh -dF airmass_plot_00.png
```

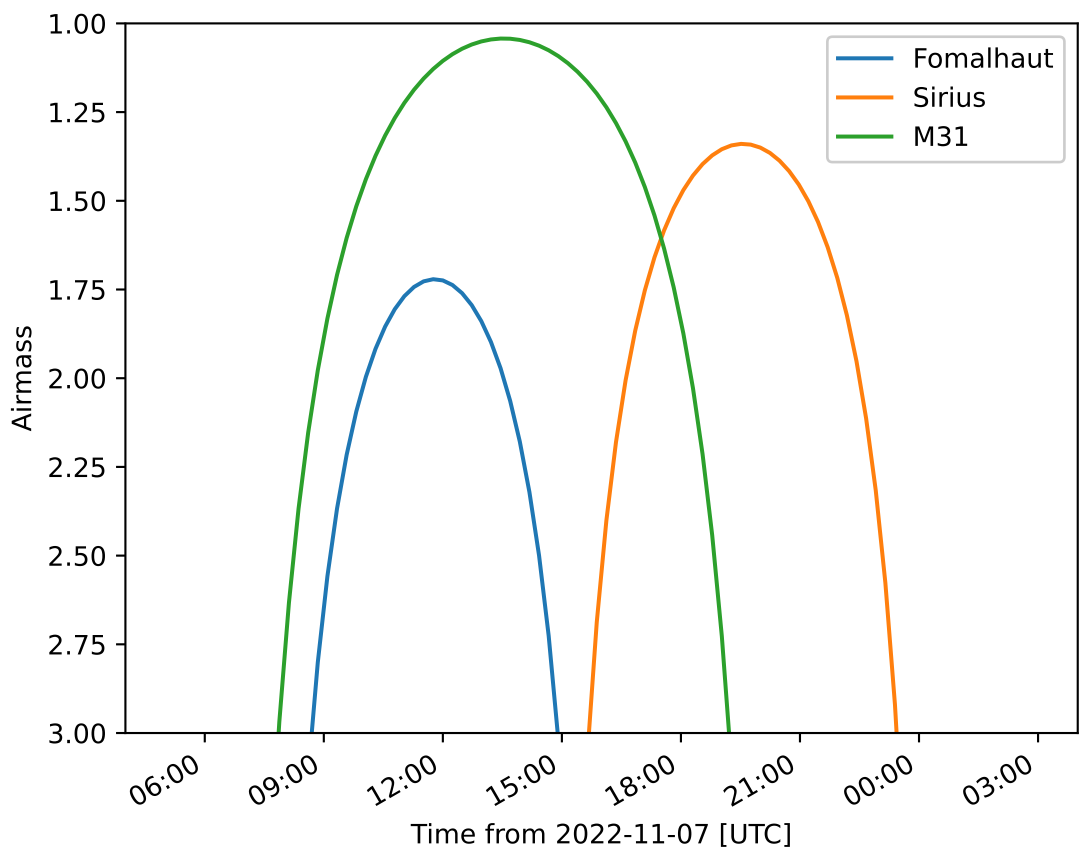


Figure 16: The change of the airmass of Fomalhaut, Sirius, and M31 at NCU main campus on 07/Nov/2022.

Try following practice.

Practice 08-42

Make a plot of airmass change of Deneb, Capella, and Betelgeuse as observed at Lulin Observatory on 15/Nov/2022.

Change some settings, and generate a plot again.

Python Code 45: ai202209_s08_04_01.py

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

#
# Time-stamp: <2022/11/06 23:46:44 (CST) daisuke>
#

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

# importing astroplan module
import astroplan
import astroplan.plots

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

# units
u_m  = astropy.units.m

# output file name
file_output = 'airmass_plot_01.png'

# getting coordinate of a fixed target
fomalhaut = astroplan.FixedTarget.from_name ('Fomalhaut')
sirius    = astroplan.FixedTarget.from_name ('Sirius')
m31       = astroplan.FixedTarget.from_name ('M31')

# printing coordinate
print (f'{fomalhaut}')
print (f'{sirius}')
print (f'{m31}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 16:00:00', format='iso', scale='utc')
```

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

# plotting airmass change
box = ax.get_position ()
ax.set_position ([box.x0, box.y0, box.width * 0.8, box.height])
astroplan.plots.plot_airmass (fomalhaut, observer, time, ax=ax)
astroplan.plots.plot_airmass (sirius, observer, time, ax=ax)
astroplan.plots.plot_airmass (m31, observer, time, ax=ax, \
                              brightness_shading=True, max_airmass=2.5)

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

# grid
ax.grid ()

# saving file
fig.savefig (file_output, dpi=225, bbox_inches='tight')
```

Execute above script to make a plot of airmass changes of Fomalhaut, Sirius, and M31 as observed at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_04_01.py
% ./ai202209_s08_04_01.py
<FixedTarget "Fomalhaut" at SkyCoord (ICRS): (ra, dec) in deg (344.41269272, -29
.62223703)>
<FixedTarget "Sirius" at SkyCoord (ICRS): (ra, dec) in deg (101.28715533, -16.71
611586)>
<FixedTarget "M31" at SkyCoord (ICRS): (ra, dec) in deg (10.68470833, 41.26875)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Display created PNG file. (Fig. 17)

```
% feh -dF airmass_plot_01.png
```

Try following practice.

Practice 08-43

Make a plot of airmass change of M45, Algol, and Rigel as observed at Lulin Observatory on 15/Dec/2022.

Set time range and generate the plot again.

Python Code 46: ai202209_s08_04_02.py

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

#
# Time-stamp: <2022/11/06 23:48:45 (CST) daisuke>
#

# importing numpy module
```

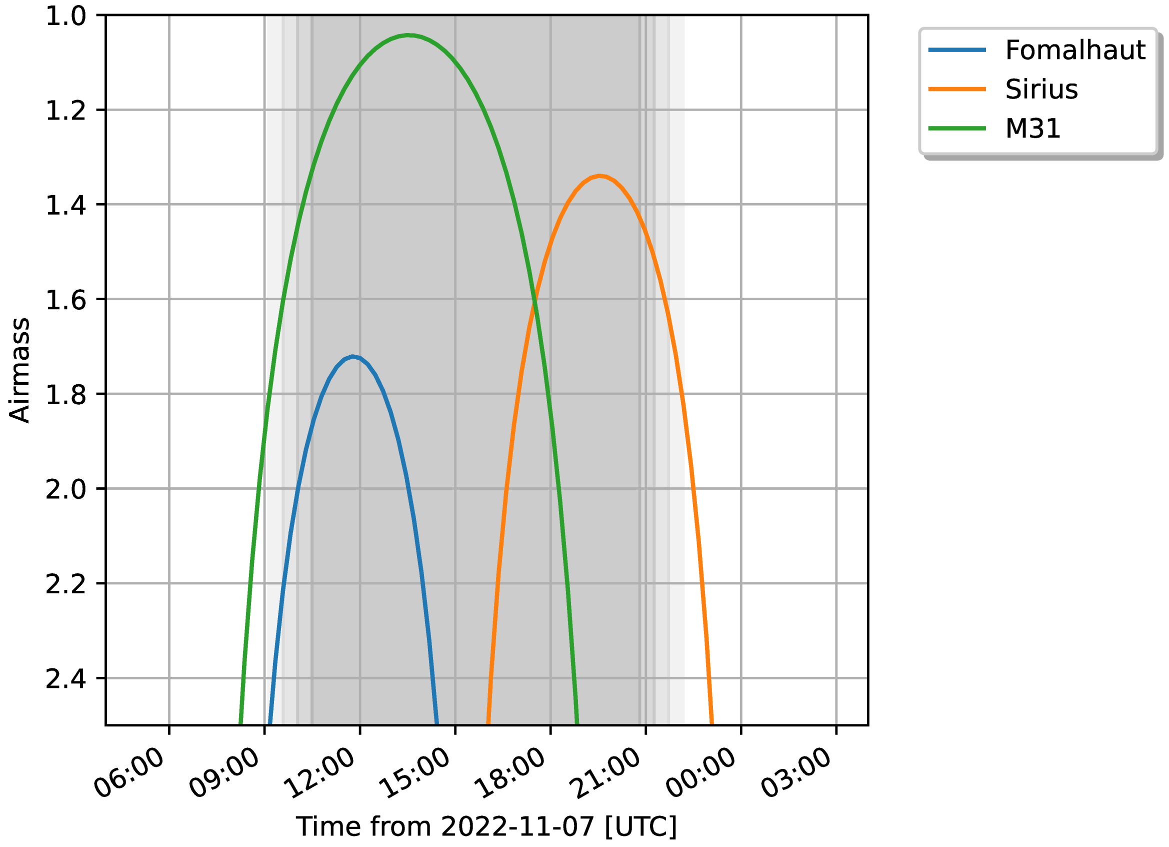


Figure 17: The change of the airmass of Fomalhaut, Sirius, and M31 at NCU main campus on 07/Nov/2022.

```
import numpy

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

# importing astroplan module
import astroplan
import astroplan.plots

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

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# output file name
file_output = 'airmass_plot_02.png'

# getting coordinate of a fixed target
fomalhaut = astroplan.FixedTarget.from_name ('Fomalhaut')
sirius    = astroplan.FixedTarget.from_name ('Sirius')
m31       = astroplan.FixedTarget.from_name ('M31')

# printing coordinate
print (f'{fomalhaut}')
print (f'{sirius}')
print (f'{m31}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 16:00:00', format='iso', \
                          scale='utc') \
                          + numpy.linspace (-7.5, +7.5, 1000) * u_hr

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

```
# plotting airmass change
box = ax.get_position ()
ax.set_position ([box.x0, box.y0, box.width * 0.8, box.height])
astroplan.plots.plot_airmass (fomalhaut, observer, time, ax=ax)
astroplan.plots.plot_airmass (sirius, observer, time, ax=ax)
astroplan.plots.plot_airmass (m31, observer, time, ax=ax, \
                              brightness_shading=True, max_airmass=2.5)

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

# grid
ax.grid ()

# saving file
fig.savefig (file_output, dpi=225, bbox_inches='tight')
```

Execute above script to make a plot of airmass changes of Fomalhaut, Sirius, and M31 as observed at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_04_02.py
% ./ai202209_s08_04_02.py
<FixedTarget "Fomalhaut" at SkyCoord (ICRS): (ra, dec) in deg (344.41269272, -29
.62223703)>
<FixedTarget "Sirius" at SkyCoord (ICRS): (ra, dec) in deg (101.28715533, -16.71
611586)>
<FixedTarget "M31" at SkyCoord (ICRS): (ra, dec) in deg (10.68470833, 41.26875)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Display created PNG file. (Fig. 18)

```
% feh -dF airmass_plot_02.png
```

Try following practice.

Practice 08-44

Make a plot of airmass change of Aldebaran, M35, and Procyon as observed at Lulin Observatory on 15/Dec/2022.

8 Sky chart

Make a sky chart for Fomalhaut, M31, and M45 as observed at NCU main campus on 07/Nov/2022.

Python Code 47: ai202209_s08_05_00.py

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

#
# Time-stamp: <2022/11/07 00:37:08 (CST) daisuke>
#

# importing numpy module
import numpy
```

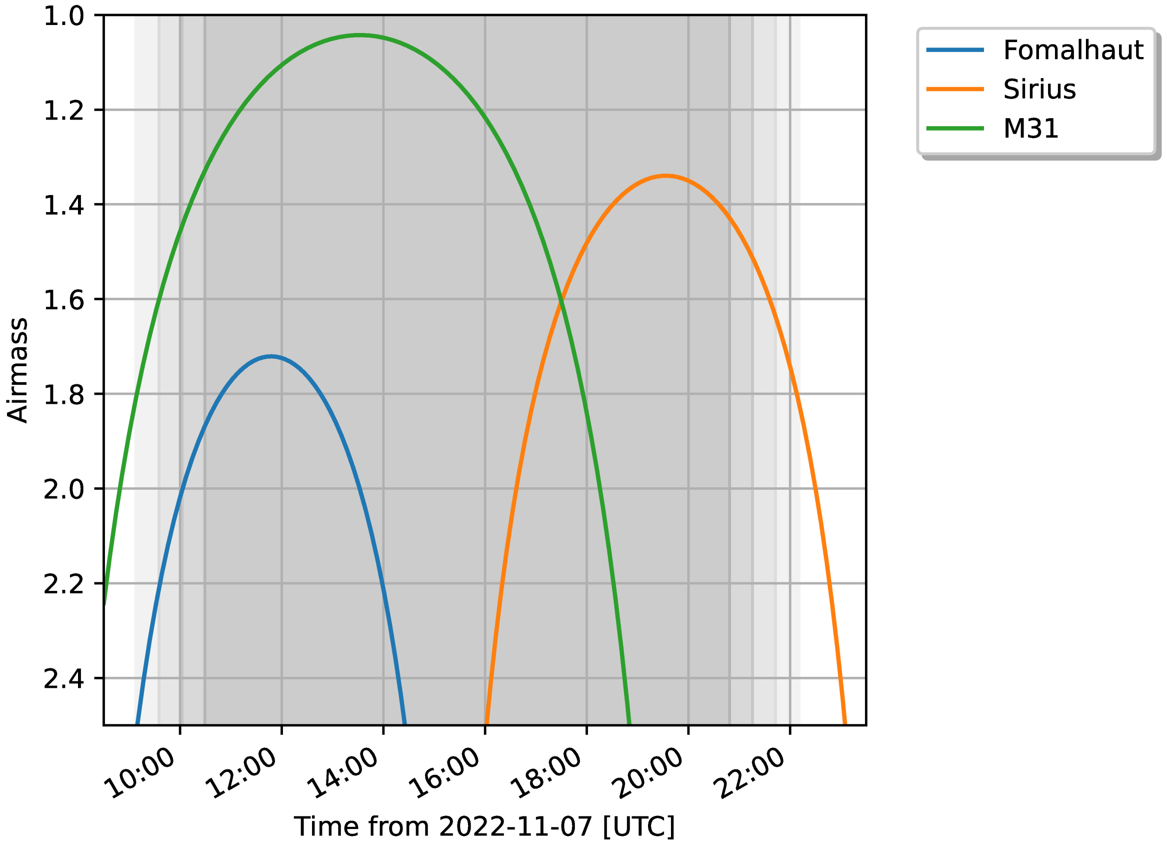


Figure 18: The change of the airmass of Fomalhaut, Sirius, and M31 at NCU main campus on 07/Nov/2022.

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

# importing astroplan module
import astroplan
import astroplan.plots

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

# units
u_m  = astropy.units.m
u_hr = astropy.units.hour

# output file name
file_output = 'sky_plot_00.png'

# getting coordinate of a fixed target
fomalhaut = astroplan.FixedTarget.from_name ('Fomalhaut')
m31       = astroplan.FixedTarget.from_name ('M31')
m45       = astroplan.FixedTarget.from_name ('M45')

# printing coordinate
print (f'{fomalhaut}')
print (f'{m31}')
print (f'{m45}')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m

# making a location object using Astropy
location = astropy.coordinates.EarthLocation.from_geodetic (lon=longitude, \
                                                            lat=latitude, \
                                                            height=height)

# making an observer object
observer = astroplan.Observer (location=location, name='NCU', \
                               timezone='Asia/Taipei')

# printing location
print (f'{observer}')

# date/time
time = astropy.time.Time ('2022-11-07 16:00:00', format='iso', \
                          scale='utc') \
                          + numpy.linspace (-6, +6, 13) * u_hr

# making fig, canvas, and ax objects
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111, projection='polar')
```

```
# grid
ax.grid ()

# plotting airmass change
astroplan.plots.plot_sky (fomalhaut, observer, time, ax=ax)
astroplan.plots.plot_sky (m31, observer, time, ax=ax)
astroplan.plots.plot_sky (m45, observer, time, ax=ax)

# title
ax.set_title ('Sky chart on 07/Nov/2022')

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

# saving file
fig.savefig (file_output, dpi=225, bbox_inches='tight')
```

Execute above script to make a sky chart for Fomalhaut, M31, and M45 as observed at NCU main campus on 07/Nov/2022.

```
% chmod a+x ai202209_s08_05_00.py
% ./ai202209_s08_05_00.py
<FixedTarget "Fomalhaut" at SkyCoord (ICRS): (ra, dec) in deg (344.41269272, -29
.62223703)>
<FixedTarget "M31" at SkyCoord (ICRS): (ra, dec) in deg (10.68470833, 41.26875)>
<FixedTarget "M45" at SkyCoord (ICRS): (ra, dec) in deg (56.601, 24.114)>
<Observer: name='NCU',
    location (lon, lat, el)=(121.18666666666667 deg, 24.970000000000002 deg, 151
.60000000097773 m),
    timezone=<DstTzInfo 'Asia/Taipei' LMT+8:06:00 STD>>
```

Display created PNG file. (Fig. 19)

```
% feh -dF sky_plot_00.png
```

Try following practice.

Practice 08-44

Make a sky chart of Aldebaran, Capella, and Sirius as observed at Lulin Observatory on 15/Jan/2023.

9 Finding chart

Make a finding chart for M101.

Python Code 48: ai202209_s08_06_00.py

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

#
# Time-stamp: <2022/11/07 00:55:47 (CST) daisuke>
#

# importing astropy module
import astropy.units

# importing astroplan module
```

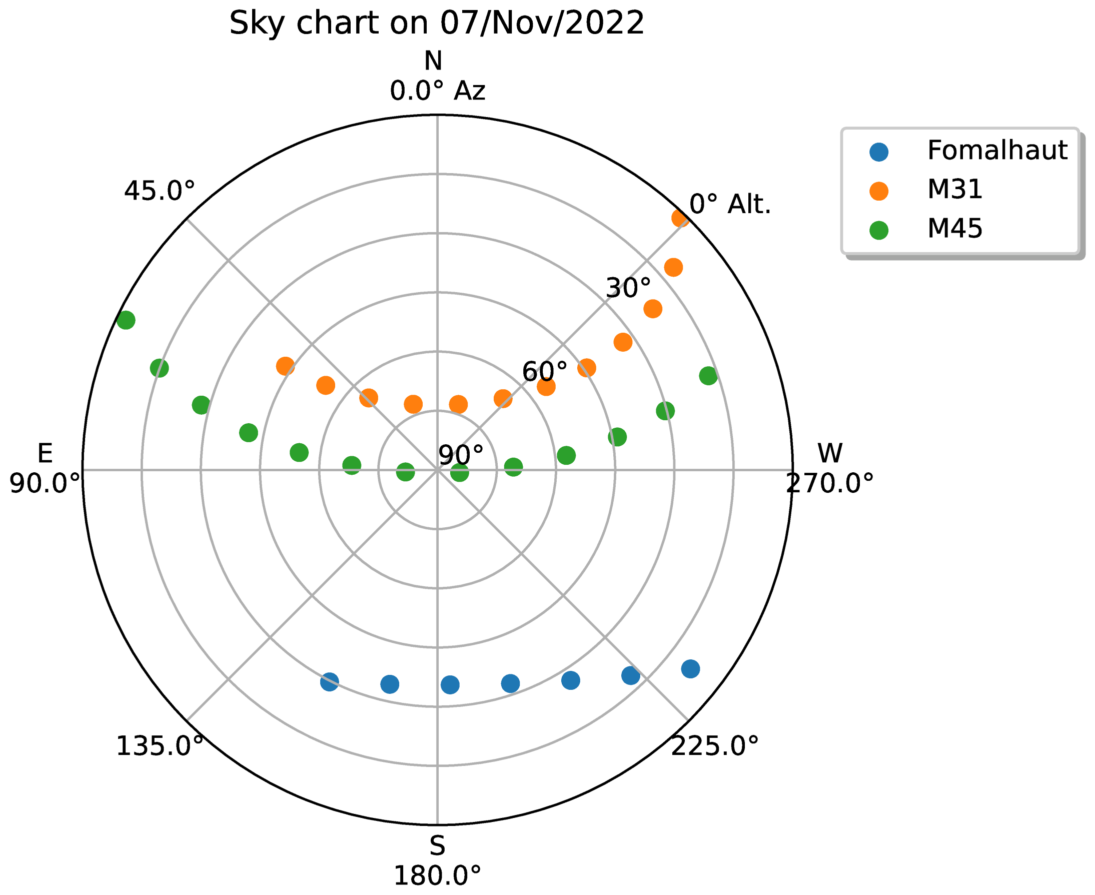


Figure 19: The change of the airmass of Fomalhaut, M31, and M45 at NCU main campus on 07/Nov/2022.

```
import astroplan
import astroplan.plots

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

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# output file name
file_output = 'findingchart_00.png'

# units
unit_arcmin = astropy.units.arcmin

# field-of-view
fov = 20.0 * unit_arcmin

# target object
m101 = astroplan.FixedTarget.from_name ('M101')

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

# image
ax, hdu = astroplan.plots.plot_finder_image (m101, fov_radius=fov, ax=ax)

# saving the image to file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a finding chart of M101.

```
% chmod a+x ai202209_s08_06_00.py
% ./ai202209_s08_06_00.py
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv22016059932944.fits
|==========================================| 374k/374k (100.00%)         3s
```

Display created PNG file. (Fig. 20)

```
% feh -dF findingchart_00.png
```

Try following practice.

Practice 08-45

Make a finding chart for your favourite object.

10 For your further reading

Read following document to learn more about Astroplan.

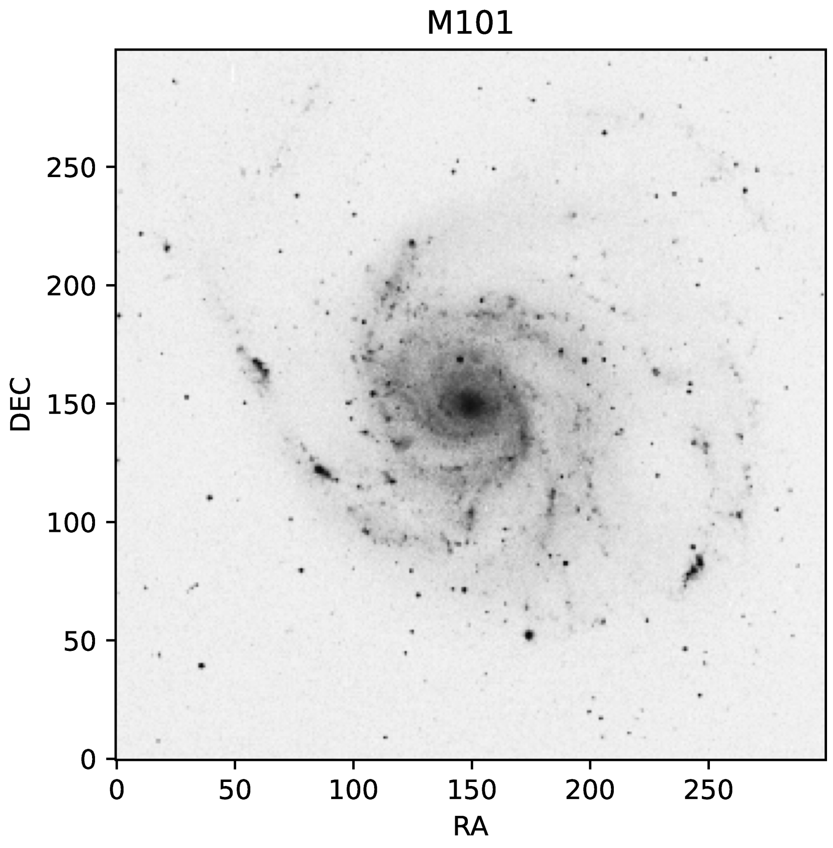


Figure 20: A finding chart for M101.

- Astroplan documentation: `https://astroplan.readthedocs.io/`
  - Tutorials: `https://astroplan.readthedocs.io/en/latest/tutorials/`
  - Reference/API: `https://astroplan.readthedocs.io/en/latest/api.html`

11 Assignment

1. Sunrise and sunset
   - (a) Calculate the time of sunset and sunrise at Lulin Observatory on 20 December 2022.
   - (b) Calculate the time of sunset and sunrise at Lulin Observatory on 20 June 2023.
   - (c) Calculate the time of sunset and sunrise at Paris Observatory on 20 December 2022.
   - (d) Calculate the time of sunset and sunrise at Paris Observatory on 20 June 2023.
   - (e) Calculate the time of sunset and sunrise at Bosscha Observatory on 20 December 2022.
   - (f) Calculate the time of sunset and sunrise at Bosscha Observatory on 20 June 2023.
2. 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?
3. Airmass
   - (a) What is airmass?
   - (b) How important is it for astronomical observations?
4. 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.