Astroinformatics 2022
Session 07: Using Astropy

Kinoshita Daisuke

31 October 2022
publicly accessible version

The Astropy is a Python package for astronomical calculations and data analysis. The Astropy is designed to be a tool for professional astronomers and astrophysicists, and is very useful for our research activities. For this session, we try some functionalities of Astropy.

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

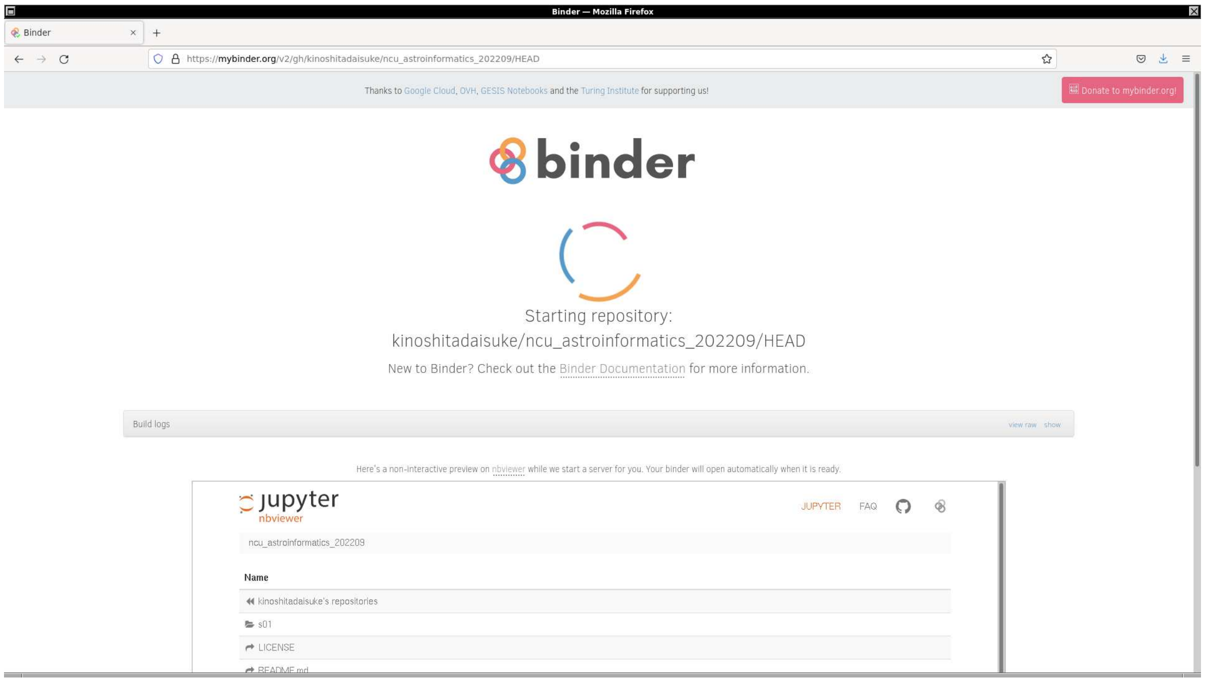
If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

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) "`s06`". (Fig. 3) Choose the file "`ai202209_s06.ipynb`" (Fig. 4 and 5) and open it (Fig. 6).


Binder — Mozilla Firefox

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

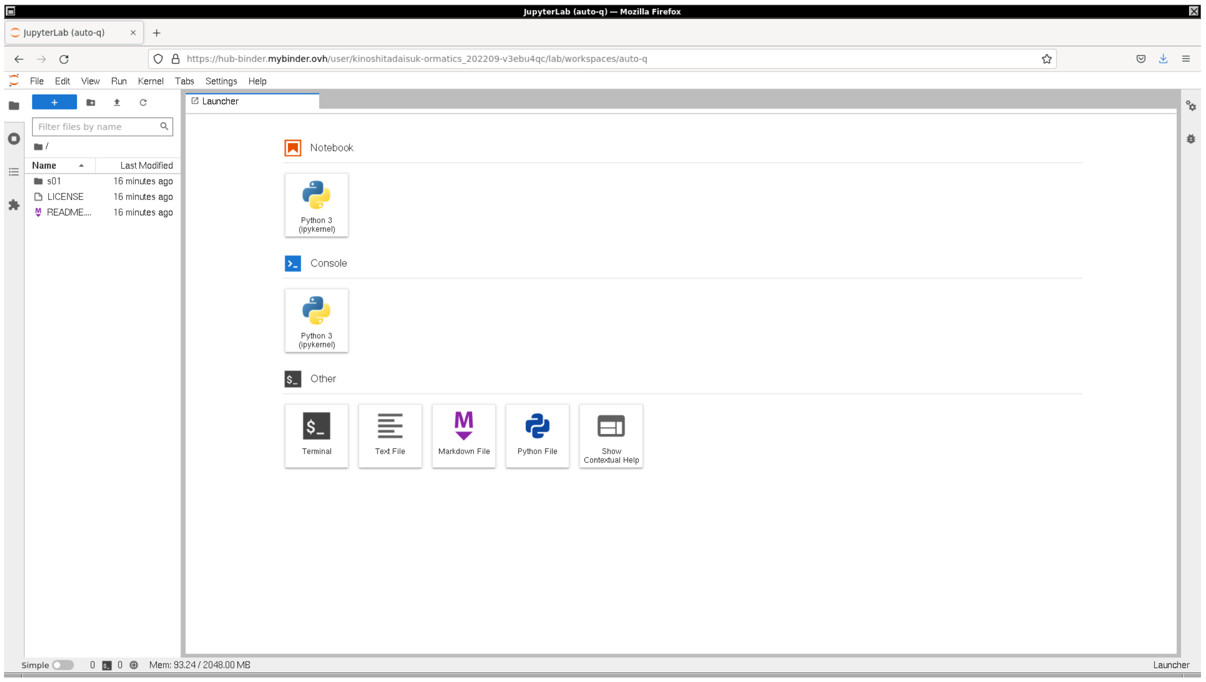
Thanks to Google Cloud, OVH, GESIS Notebooks and the Turing Institute for supporting us!

Starting repository: kinoshitadaisuke/ncu_astroinformatics_202209/HEAD

New to Binder? Check out the Binder Documentation for more information.

Here's a non-interactive preview on nbviewer while we start a server for you. Your binder will open automatically when it is ready.

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

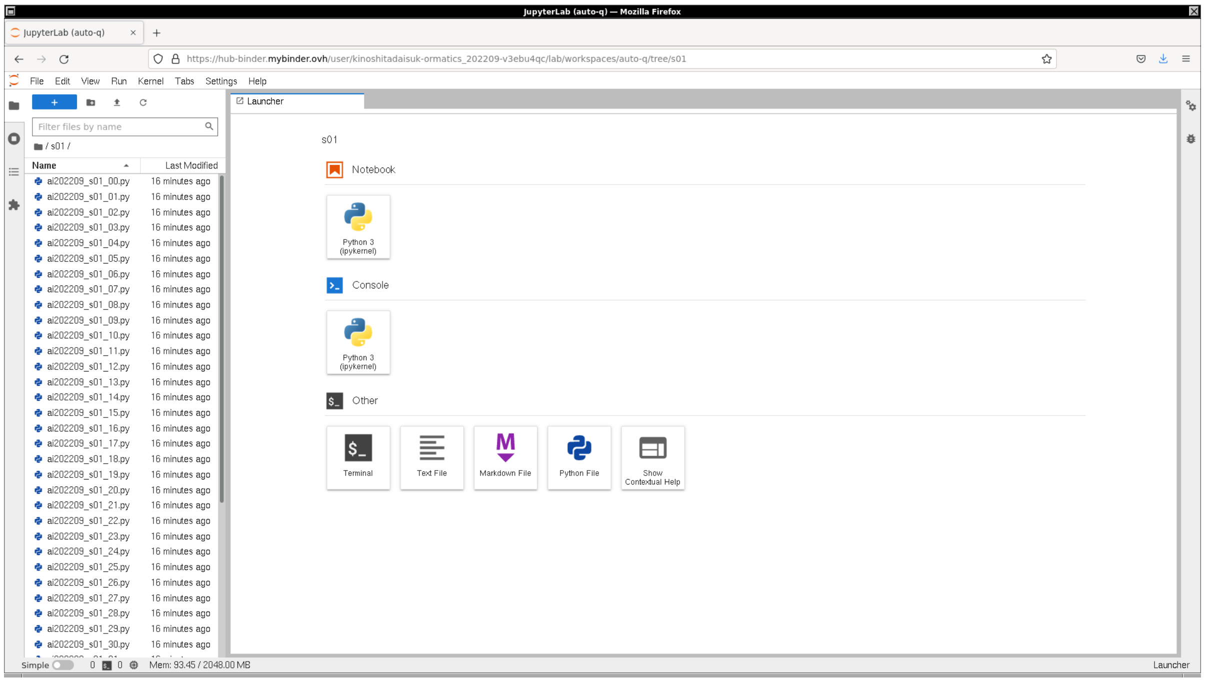

JupyterLab (auto-q) — Mozilla Firefox

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

Name	Last Modified
s01	16 minutes ago
LICENSE	16 minutes ago
README....	16 minutes ago

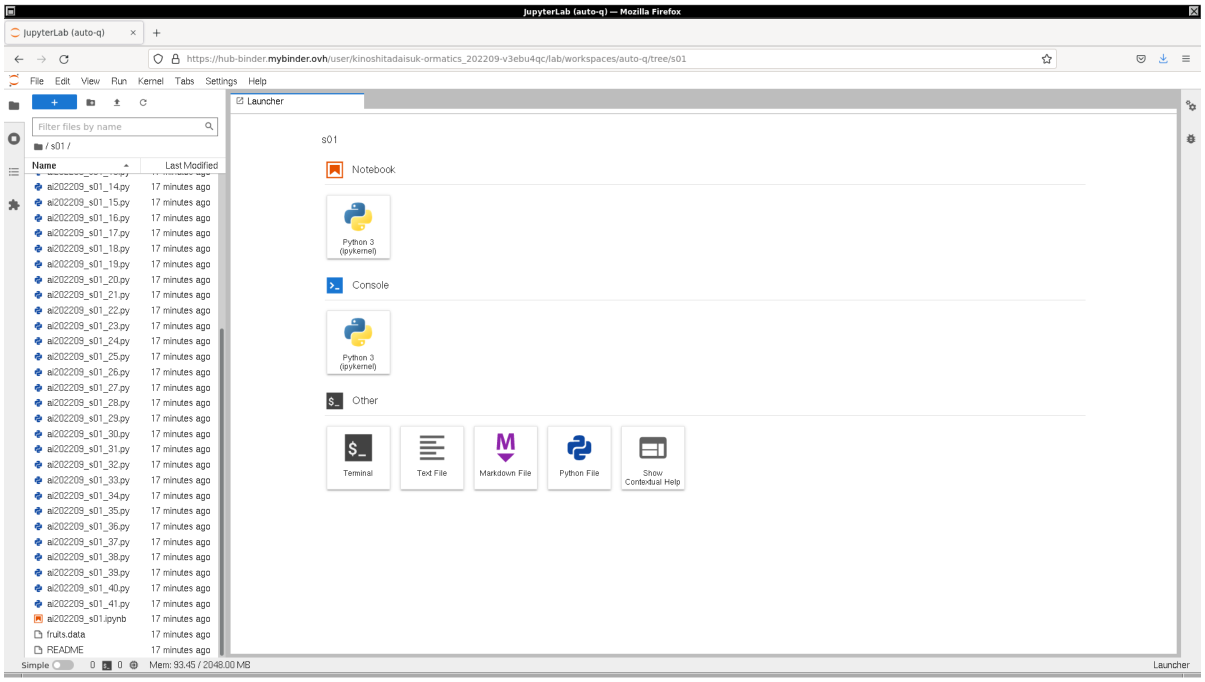
Mem: 93.24 / 2048.00 MB

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


JupyterLab (auto-q) — Mozilla Firefox
JupyterLab (auto-q)
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ s01 /
Launcher
s01
Notebook
Python 3 (ipykernel)
Console
Python 3 (ipykernel)
Other
Terminal
Text File
Markdown File
Python File
Show Contextual Help
Simple 0 0 Mem: 93.45 / 2048.00 MB
Launcher

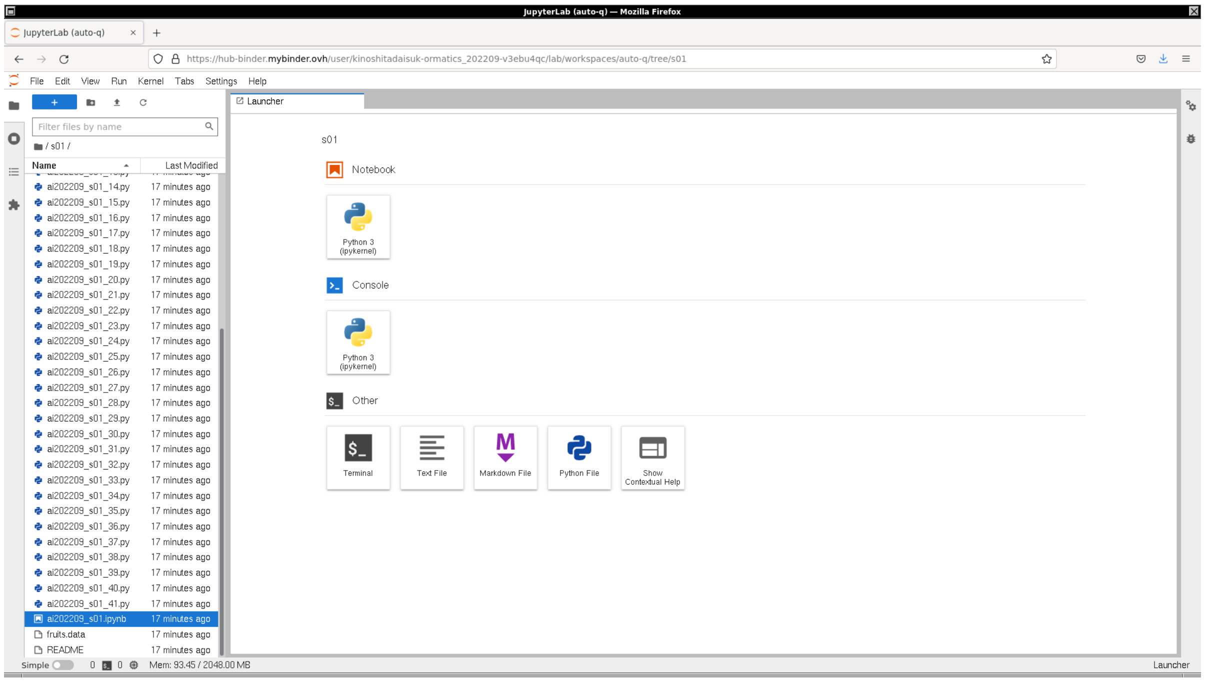
Name	Last Modified
ai202209_s01_00.py	16 minutes ago
ai202209_s01_01.py	16 minutes ago
ai202209_s01_02.py	16 minutes ago
ai202209_s01_03.py	16 minutes ago
ai202209_s01_04.py	16 minutes ago
ai202209_s01_05.py	16 minutes ago
ai202209_s01_06.py	16 minutes ago
ai202209_s01_07.py	16 minutes ago
ai202209_s01_08.py	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.


JupyterLab (auto-q) — Mozilla Firefox
JupyterLab (auto-q)
https://hub-binder.mybinder.ovh/user/kinoshitadaisuk-ormatics_202209-v3ebu4qc/lab/workspaces/auto-q/tree/s01
File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ s01 /
Launcher
s01
Notebook
Python 3 (ipykernel)
Console
Python 3 (ipykernel)
Other
Terminal
Text File
Markdown File
Python File
Show Contextual Help
Simple 0 0 Mem: 93.45 / 2048.00 MB
Launcher

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

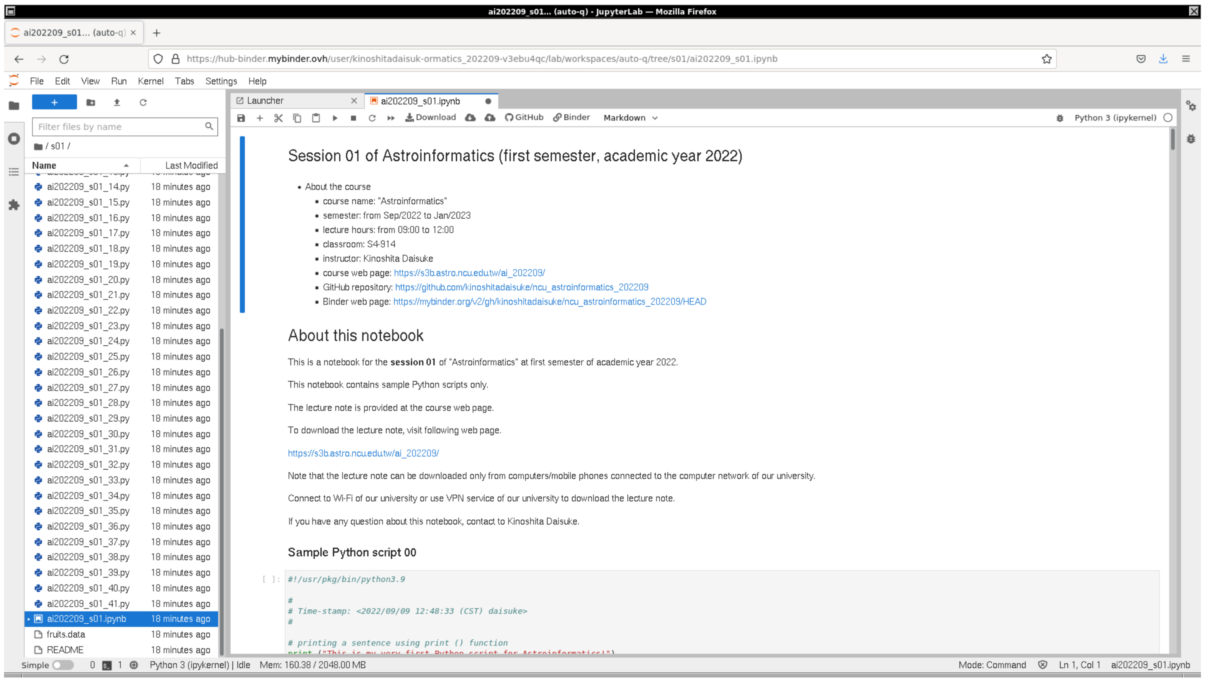
Figure 4: Using Binder to execute sample Python scripts for this session.


JupyterLab (auto-q) — Mozilla Firefox

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

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


ai202209_s01... (auto-q) - JupyterLab — Mozilla Firefox

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

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
```

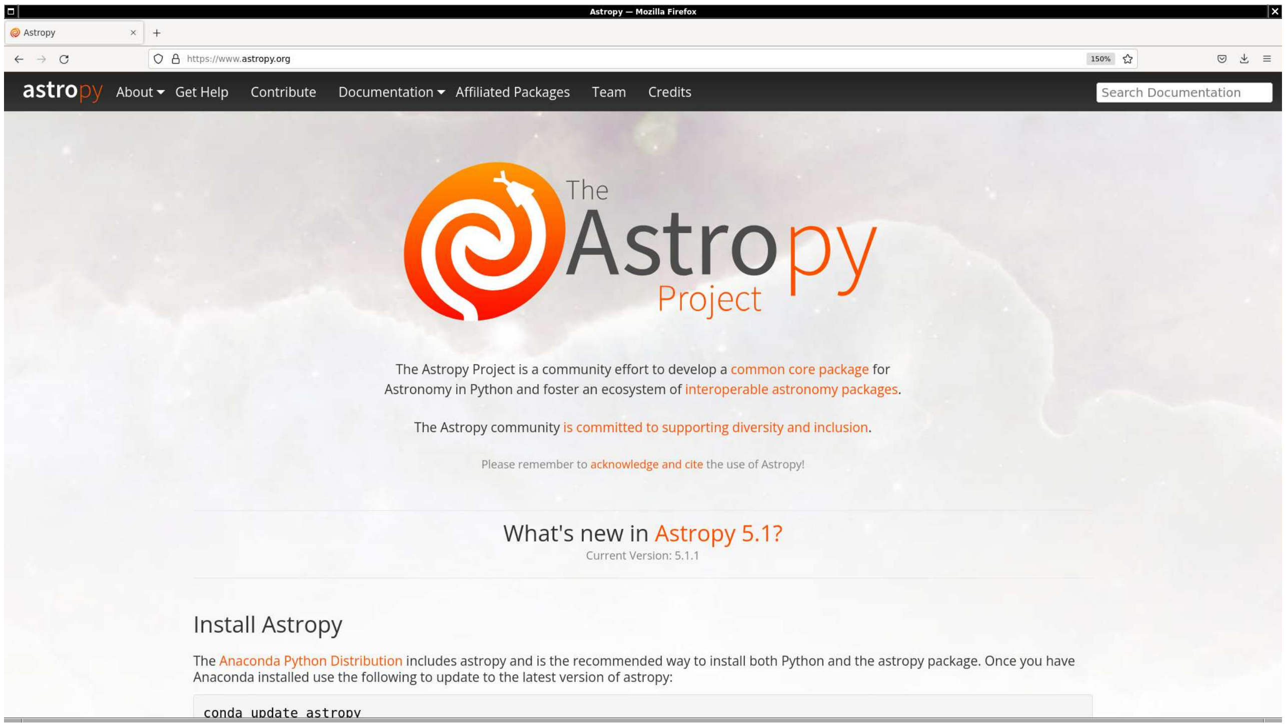
Figure 6: Using Binder to execute sample Python scripts for this session.

2 About Astropy

2.1 Online resources of Astropy

To learn about Astropy, visit the official website of the Astropy Project.

- Astropy: `https://www.astropy.org/` (Fig. 7)
    - Astropy documentation: `https://docs.astropy.org/` (Fig. 8)
    - Astropy on PyPI: `https://pypi.org/project/astropy/` (Fig. 9)
    - GitHub repository: `https://github.com/astropy/astropy` (Fig. 10)
    - paper about Astropy: `https://doi.org/10.1051/0004-6361/201322068` (Fig. 11)



The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

The Astropy community is committed to supporting diversity and inclusion.

Please remember to acknowledge and cite the use of Astropy!

What's new in Astropy 5.1?
Current Version: 5.1.1

Install Astropy

The Anaconda Python Distribution includes astropy and is the recommended way to install both Python and the astropy package. Once you have Anaconda installed use the following to update to the latest version of astropy:

```
conda update astropy
```

Figure 7: The official website of the Astropy project.

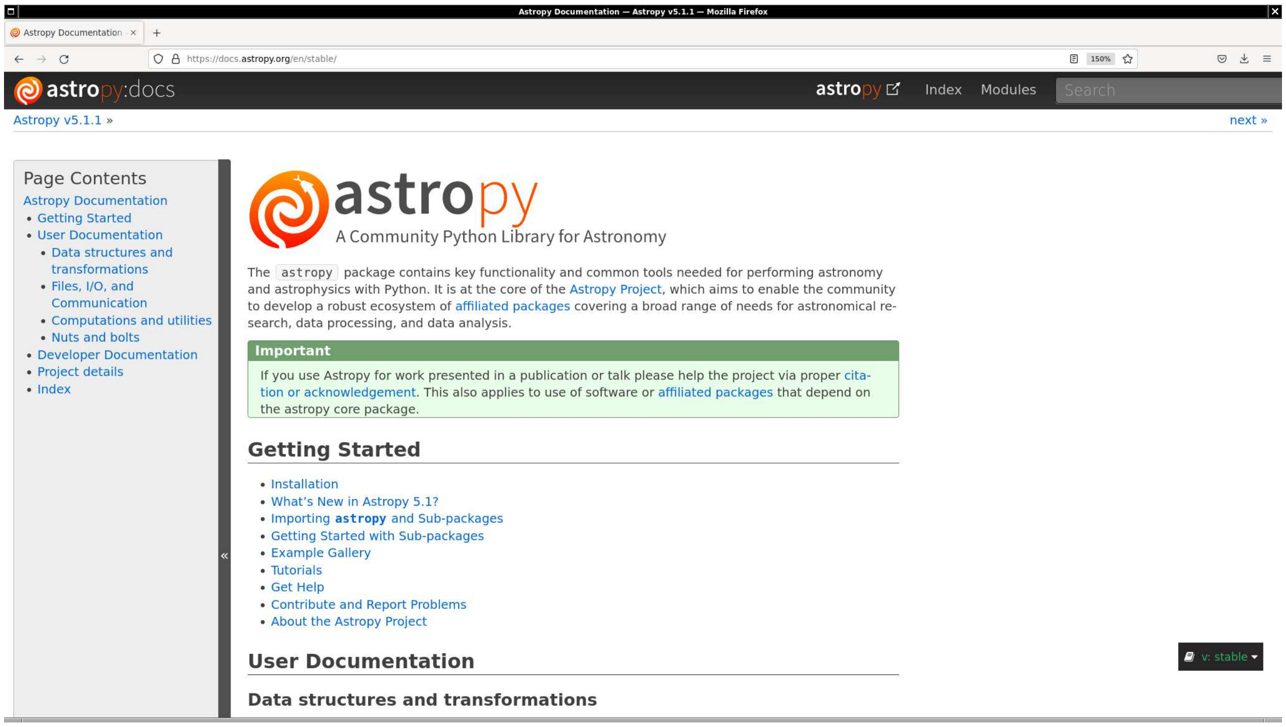
2.2 Importing Astropy

To check whether or not you have Astropy on your computer, try following.

```
% 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 astropy
>>> exit ()
```

If you do not have Astropy installed on your computer, you see following error message.

```
% python3.9
Python 3.9.13 (main, Sep 13 2022, 10:07:35)
[Clang 13.0.0 (git@github.com:llvm/llvm-project.git llvmorg-13.0.0-0-gd7b669b3a
on freebsd13
```



Page Contents

- Astropy Documentation
  - Getting Started
  - User Documentation
    - Data structures and transformations
    - Files, I/O, and Communication
    - Computations and utilities
    - Nuts and bolts
  - Developer Documentation
  - Project details
  - Index

astropy
A Community Python Library for Astronomy

The `astropy` package contains key functionality and common tools needed for performing astronomy and astrophysics with Python. It is at the core of the Astropy Project, which aims to enable the community to develop a robust ecosystem of affiliated packages covering a broad range of needs for astronomical research, data processing, and data analysis.

Important

If you use Astropy for work presented in a publication or talk please help the project via proper citation or acknowledgement. This also applies to use of software or affiliated packages that depend on the astropy core package.

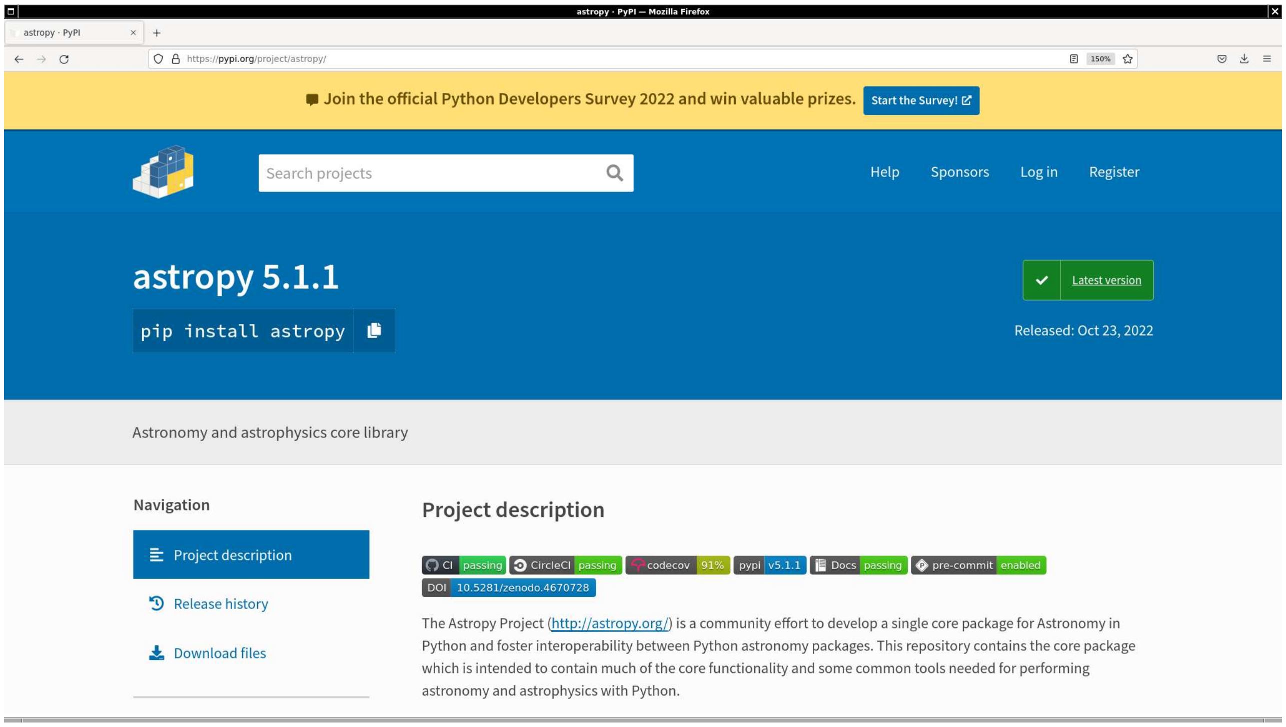
Getting Started

- Installation
- What's New in Astropy 5.1?
- Importing **`astropy`** and Sub-packages
- Getting Started with Sub-packages
- Example Gallery
- Tutorials
- Get Help
- Contribute and Report Problems
- About the Astropy Project

User Documentation

Data structures and transformations

Figure 8: The web page of the Astropy documentation.



Join the official Python Developers Survey 2022 and win valuable prizes.

astropy 5.1.1

```
pip install astropy
```

Released: Oct 23, 2022

Astronomy and astrophysics core library

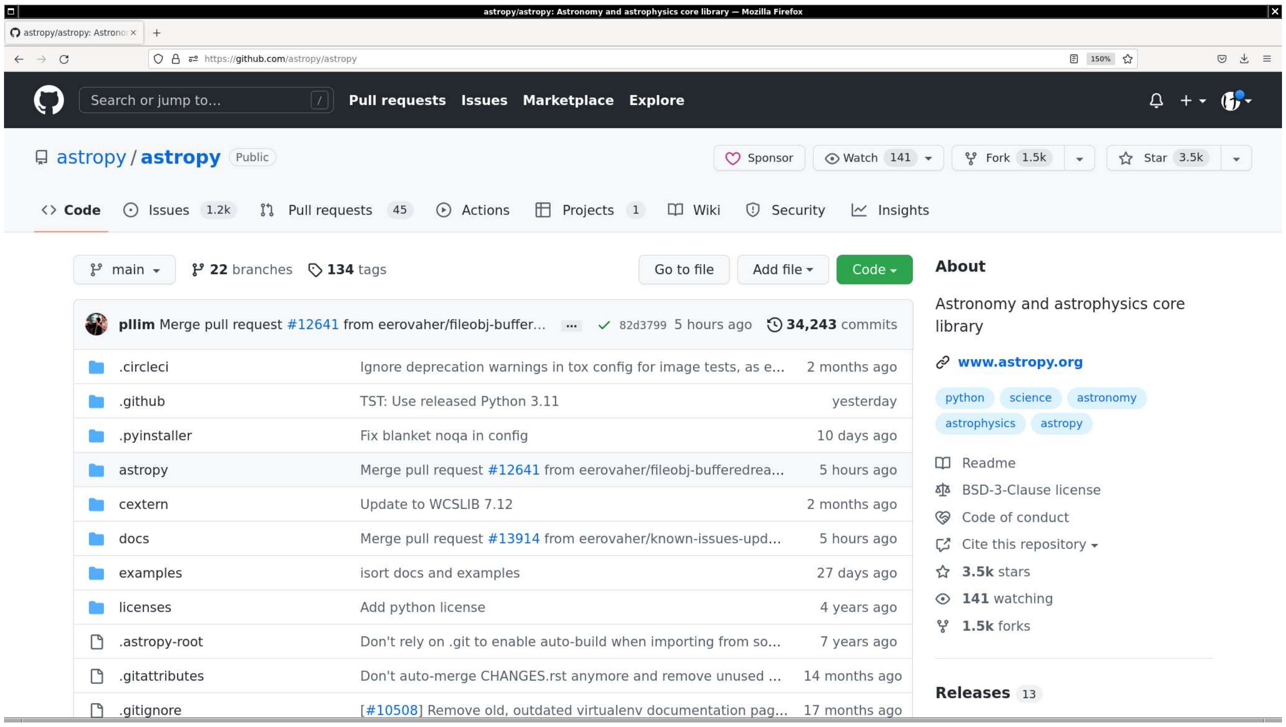
Navigation

- Project description
- Release history
- Download files

Project description

The Astropy Project (http://astropy.org/) is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages. This repository contains the core package which is intended to contain much of the core functionality and some common tools needed for performing astronomy and astrophysics with Python.

Figure 9: The Astropy web page on PyPI website.


astropy/astropy: Astronomy and astrophysics core library — Mozilla Firefox

https://github.com/astropy/astropy

Search or jump to... Pull requests Issues Marketplace Explore

astropy / astropy Public

Sponsor | Watch 141 | Fork 1.5k | Star 3.5k

Code | Issues 1.2k | Pull requests 45 | Actions | Projects 1 | Wiki | Security | Insights

main | 22 branches | 134 tags | Go to file | Add file | Code

pllim Merge pull request #12641 from eerovaher/fileobj-buffer... 82d3799 5 hours ago 34,243 commits

.circleci	Ignore deprecation warnings in tox config for image tests, as e...	2 months ago
.github	TST: Use released Python 3.11	yesterday
.pyinstaller	Fix blanket noqa in config	10 days ago
astropy	Merge pull request #12641 from eerovaher/fileobj-bufferedrea...	5 hours ago
cextern	Update to WCSLIB 7.12	2 months ago
docs	Merge pull request #13914 from eerovaher/known-issues-upd...	5 hours ago
examples	isort docs and examples	27 days ago
licenses	Add python license	4 years ago
.astropy-root	Don't rely on .git to enable auto-build when importing from so...	7 years ago
.gitattributes	Don't auto-merge CHANGES.rst anymore and remove unused ...	14 months ago
.gitignore	[#10508] Remove old, outdated virtualenv documentation pag...	17 months ago

About

Astronomy and astrophysics core library

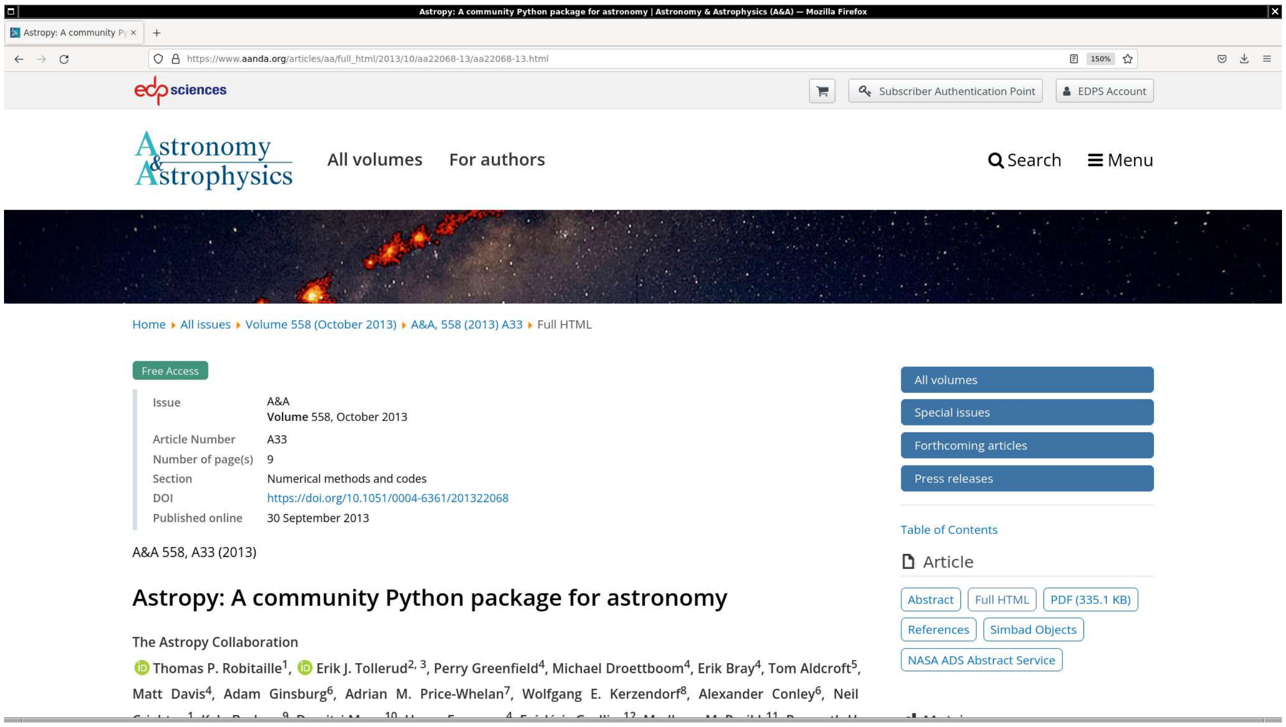
www.astropy.org

python science astronomy astrophysics astropy

Readme
BSD-3-Clause license
Code of conduct
Cite this repository
3.5k stars
141 watching
1.5k forks

Releases 13

Figure 10: The Astropy repository on GitHub.


Astropy: A community Python package for astronomy | Astronomy & Astrophysics (A&A) — Mozilla Firefox

https://www.aanda.org/articles/aa/full_html/2013/10/aa22068-13/aa22068-13.html

edp sciences | Subscriber Authentication Point | EDPS Account

Astronomy & Astrophysics | All volumes | For authors | Search | Menu

Home ▸ All issues ▸ Volume 558 (October 2013) ▸ A&A, 558 (2013) A33 ▸ Full HTML

Free Access

Issue	A&A **Volume** 558, October 2013
Article Number	A33
Number of page(s)	9
Section	Numerical methods and codes
DOI	https://doi.org/10.1051/0004-6361/201322068
Published online	30 September 2013

A&A 558, A33 (2013)

Astropy: A community Python package for astronomy

The Astropy Collaboration

Thomas P. Robitaille[1], Erik J. Tollerud[2, 3], Perry Greenfield[4], Michael Droettboom[4], Erik Bray[4], Tom Aldcroft[5], Matt Davis[4], Adam Ginsburg[6], Adrian M. Price-Whelan[7], Wolfgang E. Kerzendorf[8], Alexander Conley[6], Neil

All volumes | Special issues | Forthcoming articles | Press releases

Table of Contents

Article

Abstract | Full HTML | PDF (335.1 KB) | References | Simbad Objects | NASA ADS Abstract Service

Figure 11: The paper about Astropy on the journal “Astronomy and Astrophysics”.

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import astropy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'astropy'
>>> exit ()
```

If you do not have Astropy on your computer, (1) install Astropy on your computer, or (2) use Binder.

3 Constants

Astropy contains number of physical and astronomical constants.

3.1 Speed of light

Show the value of speed of light in vacuum. Here is an example Python script.

Python Code 1: ai202209_s07_00.py

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

#
# Time-stamp: <2022/10/27 14:42:30 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# speeed of light in vacuum
c = astropy.constants.c

# printing c
print (c)
```

Execute above script to print the value of speed of light in vacuum.

```
% chmod a+x ai202209_s07_00.py
% ./ai202209_s07_00.py
  Name   = Speed of light in vacuum
  Value  = 299792458.0
  Uncertainty  = 0.0
  Unit  = m / s
  Reference = CODATA 2018
```

Try following practice.

Practice 07-01

Print the value of Planck constant using `astropy.constants`.

The constant can be used for calculations. Here is an example.

Python Code 2: ai202209_s07_01.py

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

#
# Time-stamp: <2022/10/27 14:56:29 (CST) daisuke>
#
```

```python

# importing astropy module
import astropy.constants

# speeed of light in vacuum
c = astropy.constants.c

# calculation
v = 0.1 * c

# printing c and v
print (f'c = {c}')
print (f'v = 0.1 * {c}\n  = {v}')
```

Execute above script to print $v = 0.1c$.

```
% chmod a+x ai202209_s07_01.py
% ./ai202209_s07_01.py
c = 299792458.0 m / s
v = 0.1 * 299792458.0 m / s
  = 29979245.8 m / s
```

The variable v is a "`Quantity`" object which is a number with a unit. To know about "`Quantity`" object, visit following web page.

- "`Quantity`" object of Astropy: `https://docs.astropy.org/en/stable/api/astropy.units.Quantity.html`

Try following practice.

Practice 07-01

Print the value of $0.5c$ using `astropy.constants`.

To convert a unit to the other, try following.

Python Code 3: ai202209_s07_02.py

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

#
# Time-stamp: <2022/10/27 14:59:09 (CST) daisuke>
#

# importing astropy module
import astropy.constants
import astropy.units

# speeed of light in vacuum
c = astropy.constants.c

# calculation
v = 0.1 * c

# km/s
unit_km          = astropy.units.km
unit_sec         = astropy.units.s
unit_km_per_sec = unit_km / unit_sec

# conversion of unit
v2 = v.to (unit_km_per_sec)
```

```
# printing c, v, and v2
print (f'c = {c}')
print (f'v = 0.1 * {c}\n  = {v}\n  = {v2}')
```

Execute above script to print $v = 0.1c$ in km/s.

```
% chmod a+x ai202209_s07_02.py
% ./ai202209_s07_02.py
c = 299792458.0 m / s
v = 0.1 * 299792458.0 m / s
  = 29979245.8 m / s
  = 29979.2458 km / s
```

Try following practice.

Practice 07-01

Print the value of $0.5c$ in km/s using `astropy.constants`.

3.2 Some more physical constants

Try some more physical constants.

Python Code 4: ai202209_s07_03.py

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

#
# Time-stamp: <2022/10/27 15:06:07 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# speeed of light in vacuum
c = astropy.constants.c

# printing c
print (c)
print ()

# gravitational constant
G = astropy.constants.G

# printing G
print (G)
print ()

# Planck constant
h = astropy.constants.h

# printing h
print (h)
print ()

# Boltzmann constant
k = astropy.constants.k_B
```

```
# printing k
print (k)
print ()

# Stafan-Boltzmann constant
sigma = astropy.constants.sigma_sb

# printing sigma
print (sigma)
print ()
```

Execute above script to print c, G, h, k, and σ.

```
% chmod a+x ai202209_s07_03.py
% ./ai202209_s07_03.py
  Name   = Speed of light in vacuum
  Value  = 299792458.0
  Uncertainty  = 0.0
  Unit  = m / s
  Reference = CODATA 2018

  Name   = Gravitational constant
  Value  = 6.6743e-11
  Uncertainty  = 1.5e-15
  Unit  = m3 / (kg s2)
  Reference = CODATA 2018

  Name   = Planck constant
  Value  = 6.62607015e-34
  Uncertainty  = 0.0
  Unit  = J s
  Reference = CODATA 2018

  Name   = Boltzmann constant
  Value  = 1.380649e-23
  Uncertainty  = 0.0
  Unit  = J / K
  Reference = CODATA 2018

  Name   = Stefan-Boltzmann constant
  Value  = 5.6703744191844314e-08
  Uncertainty  = 0.0
  Unit  = W / (K4 m2)
  Reference = CODATA 2018
```

Try following practice.

Practice 07-01

Print the values of proton mass and electron mass using `astropy.constants`.

3.3 Astronomical unit and parsec

Try astronomical unit and parsec. Here is an example.

Python Code 5: ai202209_s07_04.py

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

```
#
# Time-stamp: <2022/10/27 15:15:42 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# astronomical unit
au = astropy.constants.au

# printing au
print (au)
print ()

# parsec
pc = astropy.constants.pc

# printing pc
print (pc)
print ()

# 1 au
print (f'1 au = {au:g}')

# 1 pc
print (f'1 pc = {pc:g}\n      = {pc / au} au')
```

Execute above script to print the values of 1 au and 1 pc.

```
% chmod a+x ai202209_s07_04.py
% ./ai202209_s07_04.py
  Name   = Astronomical Unit
  Value  = 149597870700.0
  Uncertainty  = 0.0
  Unit  = m
  Reference = IAU 2012 Resolution B2

  Name   = Parsec
  Value  = 3.085677581491367e+16
  Uncertainty  = 0.0
  Unit  = m
  Reference = Derived from au + IAU 2015 Resolution B 2 note [4]

1 au = 1.49598e+11 m
1 pc = 3.08568e+16 m
     = 206264.80624709636 au
```

Try following practice.

Practice 07-01

Print the distance from the Sun to Jupiter in km using `astropy.constants`.

3.4 Solar mass, Jupiter mass, and Earth mass

Print Solar mass, Jupiter mass, and Earth mass. Here is an example.

Python Code 6: ai202209_s07_05.py

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

#
# Time-stamp: <2022/10/27 15:25:07 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# Solar mass
M_S = astropy.constants.M_sun

# Jupiter mass
M_J = astropy.constants.M_jup

# Earth mass
M_E = astropy.constants.M_earth

# printing Solar mass, Jupiter mass, and Earth mass
print (M_S)
print ()
print (M_J)
print ()
print (M_E)
print ()

# amount of Jupiter mass in the unit of Solar mass
print (f'1 M_J = {M_J}\n      = {M_J / M_S:g} M_S')
```

Execute above script to print the values of 1 Solar mass, 1 Jupiter mass, and 1 Earth mass.

```
% chmod a+x ai202209_s07_05.py
% ./ai202209_s07_05.py
  Name   = Solar mass
  Value  = 1.988409870698051e+30
  Uncertainty  = 4.468805426856864e+25
  Unit  = kg
  Reference = IAU 2015 Resolution B 3 + CODATA 2018

  Name   = Jupiter mass
  Value  = 1.8981245973360505e+27
  Uncertainty  = 4.26589589320839e+22
  Unit  = kg
  Reference = IAU 2015 Resolution B 3 + CODATA 2018

  Name   = Earth mass
  Value  = 5.972167867791379e+24
  Uncertainty  = 1.3422009501651213e+20
  Unit  = kg
  Reference = IAU 2015 Resolution B 3 + CODATA 2018

1 M_J = 1.8981245973360505e+27 kg
      = 0.000954594 M_S
```

Try following practice.

Practice 07-01

Print the value of 1 solar mass in Jupiter mass using `astropy.constants`.

3.5 Solar radius, Jupiter radius, and Earth radius

Print Solar radius, Jupiter radius, and Earth radius. Here is an example.

Python Code 7: ai202209_s07_06.py

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

#
# Time-stamp: <2022/10/27 15:31:10 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# Solar radius
R_S = astropy.constants.R_sun

# Jupiter radius
R_J = astropy.constants.R_jup

# Earth radius
R_E = astropy.constants.R_earth

# printing Solar radius, Jupiter radius, and Earth radius
print (R_S)
print ()
print (R_J)
print ()
print (R_E)
print ()

# amount of 1 Earth radius in the unit of Solar radius
print (f'1 R_E = {R_E:g}\n      = {R_E / R_S:g} R_S')
```

Execute above script to print the values of 1 Solar radius, 1 Jupiter radius, and 1 Earth radius.

```
% chmod a+x ai202209_s07_06.py
% ./ai202209_s07_06.py
  Name   = Nominal solar radius
  Value  = 695700000.0
  Uncertainty  = 0.0
  Unit  = m
  Reference = IAU 2015 Resolution B 3

  Name   = Nominal Jupiter equatorial radius
  Value  = 71492000.0
  Uncertainty  = 0.0
  Unit  = m
  Reference = IAU 2015 Resolution B 3

  Name   = Nominal Earth equatorial radius
  Value  = 6378100.0
  Uncertainty  = 0.0
  Unit  = m
  Reference = IAU 2015 Resolution B 3

1 R_E = 6.3781e+06 m
      = 0.00916789 R_S
```

Try following practice.

Practice 07-01

Print the value of 1 Jupiter radius in Earth radius using `astropy.constants`.

3.6 Solar luminosity

Print the value of Solar luminosity. Here is an example.

Python Code 8: ai202209_s07_07.py

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

#
# Time-stamp: <2022/10/27 15:35:55 (CST) daisuke>
#

# importing astropy module
import astropy.constants

# Solar luminosity
L_S = astropy.constants.L_sun

# printing Solar radius, Jupiter radius, and Earth radius
print (L_S)
print ()

# amount of 10,000 Solar luminosity
print (f'    1 L_S = {L_S:g}')
print (f'10000 L_S = {10000 * L_S:g}')
```

Execute above script to print the values of 1 Solar luminosity.

```
% chmod a+x ai202209_s07_07.py
% ./ai202209_s07_07.py
  Name   = Nominal solar luminosity
  Value  = 3.828e+26
  Uncertainty  = 0.0
  Unit  = W
  Reference = IAU 2015 Resolution B 3

    1 L_S = 3.828e+26 W
10000 L_S = 3.828e+30 W
```

Try following practice.

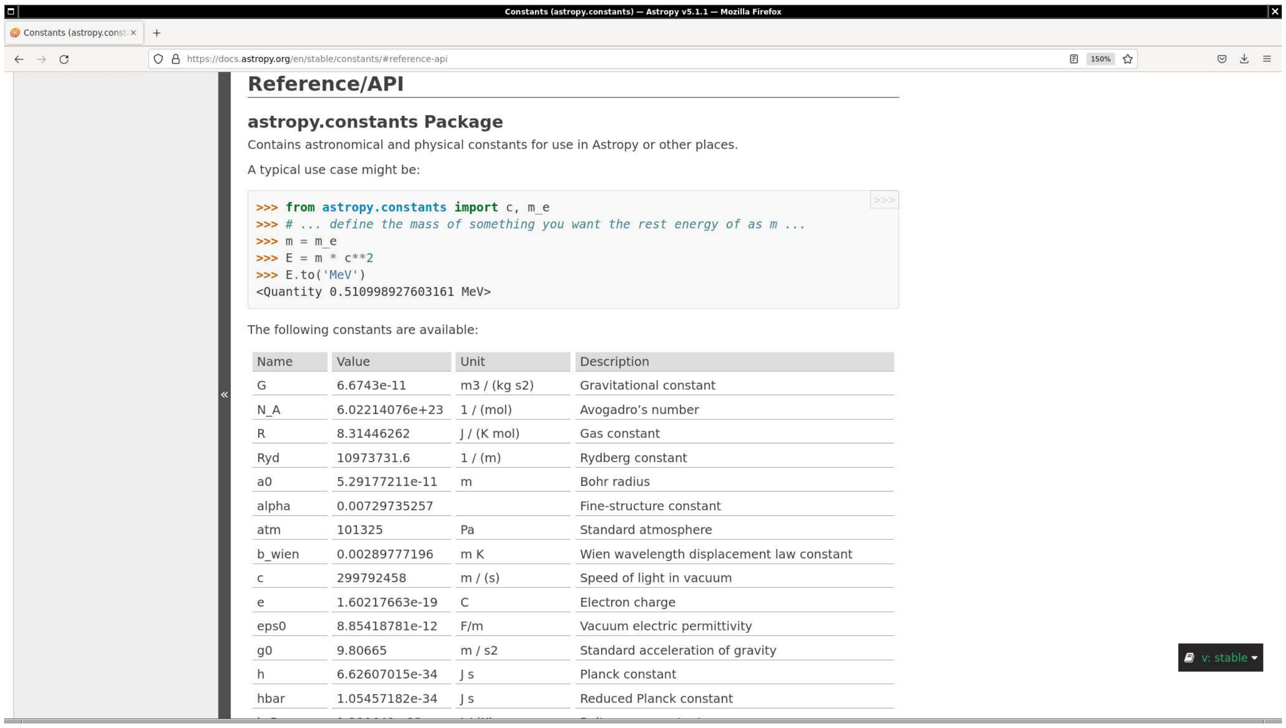
Practice 07-01

The luminosity of Betelgeuse is 126,000 $L_{\odot}$. Show the value of the luminosity of Betelgeuse in W.

3.7 A list of available constants

A list of available constants can be obtained at following web page. (Fig. 12)

- Constants (astropy.constants): `https://docs.astropy.org/en/stable/constants/#reference-api`


Constants (astropy.constants) — Astropy v5.1.1 — Mozilla Firefox
https://docs.astropy.org/en/stable/constants/#reference-api
v: stable

Reference/API

astropy.constants Package

Contains astronomical and physical constants for use in Astropy or other places.

A typical use case might be:

```
>>> from astropy.constants import c, m_e
>>> # ... define the mass of something you want the rest energy of as m ...
>>> m = m_e
>>> E = m * c**2
>>> E.to('MeV')
<Quantity 0.510998927603161 MeV>
```

The following constants are available:

Name	Value	Unit	Description
G	6.6743e-11	m3 / (kg s2)	Gravitational constant
N_A	6.02214076e+23	1 / (mol)	Avogadro's number
R	8.31446262	J / (K mol)	Gas constant
Ryd	10973731.6	1 / (m)	Rydberg constant
a0	5.29177211e-11	m	Bohr radius
alpha	0.00729735257		Fine-structure constant
atm	101325	Pa	Standard atmosphere
b_wien	0.00289777196	m K	Wien wavelength displacement law constant
c	299792458	m / (s)	Speed of light in vacuum
e	1.60217663e-19	C	Electron charge
eps0	8.85418781e-12	F/m	Vacuum electric permittivity
g0	9.80665	m / s2	Standard acceleration of gravity
h	6.62607015e-34	J s	Planck constant
hbar	1.05457182e-34	J s	Reduced Planck constant

Figure 12: The Reference/API of astropy.constants. A list of available constants is shown.

4 Units

By using "`Quantity`" object of Astropy, we are able to handle values with units. Calculations can be carried out conveniently with "`Quantity`" objects.

4.1 Playing with "`Quantity`" object

An easy way to construct a "`Quantity`" object is to multiply a number by a unit (or divide a number by a unit). Make a "`Quantity`" object of Astropy by multiplying a value with a unit. Here is an example.

Python Code 9: ai202209_s07_08.py

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

#
# Time-stamp: <2022/10/27 16:02:08 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_sec = astropy.units.s

# 900 sec
t = 900.0 * u_sec

# printing t
print (f't = {t}')
```

Execute above script to make a `Quantity` of "900 sec".

```
% chmod a+x ai202209_s07_08.py
% ./ai202209_s07_08.py
t = 900.0 s
```

Try following practice.

Practice 07-01

Make a `Quantity` object of 6.96×10^8 m.

Get the value and the unit from a "`Quantity`" object using `.value`.attribute and `.unit` attribute.

Python Code 10: ai202209_s07_09.py

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

#
# Time-stamp: <2022/10/27 16:07:42 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_sec = astropy.units.s

# 900 sec
t = 900.0 * u_sec

# printing t
print (f't = {t}')

# value and unit of t
print (f'value of t = {t.value}')
print (f'unit of t  = {t.unit}')
```

Execute above script to carry out an arithmetic calculation of "`Quantity`" objects.

```
% chmod a+x ai202209_s07_09.py
% ./ai202209_s07_09.py
t = 900.0 s
value of t = 900.0
unit of t  = s
```

Try following practice.

Practice 07-01

Make a `Quantity` object of 6400 km, and get the value and the unit from it.

Try an arithmetic calculation of "`Quantity`" objects. Here is an example.

Python Code 11: ai202209_s07_10.py

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

#
# Time-stamp: <2022/10/27 16:06:58 (CST) daisuke>
#

# importing astropy module
```

```
import astropy.units

# units
u_sec = astropy.units.s

# t1 = 3600 sec
t1 = 3600.0 * u_sec

# t2 = 900 sec
t2 = 900.0 * u_sec

# calculation of t3 = t1 - t2
t3 = t1 - t2

# printing t1, t2, and t3
print (f't1 = {t1}')
print (f't2 = {t2}')
print (f't3 = t1 - t2 = {t3}')
```

Execute above script to carry out an arithmetic calculation of "`Quantity`" objects.

```
% chmod a+x ai202209_s07_10.py
% ./ai202209_s07_10.py
t1 = 3600.0 s
t2 = 900.0 s
t3 = t1 - t2 = 2700.0 s
```

Try following practice.

Practice 07-01

Make `Quantity` objects of 100 m and 30 m, and calculate 100 m × 30 m.

Try a conversion of unit using `.to ()` method. Here is an example.

Python Code 12: ai202209_s07_11.py

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

#
# Time-stamp: <2022/10/27 16:11:50 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_sec = astropy.units.s
u_min = astropy.units.min
u_hr  = astropy.units.h

# t1 = 3600 sec
t1 = 3600.0 * u_sec

# t2 = 900 sec
t2 = 900.0 * u_sec

# calculation of t3 = t1 - t2
t3 = t1 - t2
```

```
# conversion of unit
t4 = t3.to (u_min)
t5 = t3.to (u_hr)

# printing t1, t2, and t3
print (f't1 = {t1}')
print (f't2 = {t2}')
print (f't3 = t1 - t2 = {t3} = {t4} = {t5}')
```

Execute above script to convert units.

```
% chmod a+x ai202209_s07_11.py
% ./ai202209_s07_11.py
t1 = 3600.0 s
t2 = 900.0 s
t3 = t1 - t2 = 2700.0 s = 45.0 min = 0.75 h
```

Try following practice.

Practice 07-01

Make a `Quantity` object of 3000 m, and convert the unit into km.

Combine two or more units to make a custom unit of your own. Here is an example.

Python Code 13: ai202209_s07_12.py

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

#
# Time-stamp: <2022/10/27 16:30:42 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_sec = astropy.units.s
u_min = astropy.units.min
u_hr  = astropy.units.h
u_m   = astropy.units.m
u_km  = astropy.units.km

# distance travelled
d = 180.0 * u_km

# time elapsed
t = 2.0 * u_hr

# velocity
v = d / t

# unit of metre per sec
u_m_per_sec = u_m / u_sec

# printing distance, time, and velocity
print (f'distance travelled = {d}')
print (f'time elapsed       = {t}')
print (f'average velocity   = {v} = {v.to (u_m_per_sec)}')
```

Execute above script to deal with the unit of velocity.

```
% chmod a+x ai202209_s07_12.py
% ./ai202209_s07_12.py
distance travelled = 180.0 km
time elapsed       = 2.0 h
average velocity   = 90.0 km / h = 25.0 m / s
```

Try following practice.

Practice 07-01

Make a `Quantity` object of 9.8 m/s^2.

Try a conversion from SI unit system into CGS unit system.

Python Code 14: ai202209_s07_13.py

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

#
# Time-stamp: <2022/10/27 16:39:45 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_m          = astropy.units.m
u_kg         = astropy.units.kg
u_kg_per_m3 = u_kg / u_m**3

# density in kg/m^3
rho1 = 3000.0 * u_kg_per_m3

# density in g/cm^3
rho2 = rho1.cgs

# printing density in SI and in CGS
print (f'rho1 = {rho1}')
print (f'rho2 = {rho2}')
```

Execute above script to convert a quantity in SI unit system into CGS unit system.

```
% chmod a+x ai202209_s07_13.py
% ./ai202209_s07_13.py
rho1 = 3000.0 kg / m3
rho2 = 3.0000000000000001 g / cm3
```

Try following practice.

Practice 07-01

Make a `Quantity` object of 9.8 m/s^2, and then convert it to CGS unit system.

Try a conversion from CGS unit system into SI unit system.

Python Code 15: ai202209_s07_14.py

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

#
# Time-stamp: <2022/10/27 16:43:09 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_cm         = astropy.units.cm
u_g          = astropy.units.g
u_g_per_cm3 = u_g / u_cm**3

# density in g/cm^3
rho1 = 3.0 * u_g_per_cm3

# density in kg/m^3
rho2 = rho1.si

# printing density in SI and in CGS
print (f'rho1 = {rho1}')
print (f'rho2 = {rho2}')
```

Execute above script to convert a quantity in CGS unit system into SI unit system.

```
% chmod a+x ai202209_s07_14.py
% ./ai202209_s07_14.py
rho1 = 3.0 g / cm3
rho2 = 2999.9999999999995 kg / m3
```

Try following practice.

Practice 07-01

Convert 1 erg into SI unit system.

To learn more about units available with Astropy, visit following web page. (Fig. 13)

- `https://docs.astropy.org/en/stable/units/index.html#module-astropy.units.si`

4.2 Conversion from wavelength into frequency

Try to calculate the frequency corresponding to the electromagnetic wave having the wavelength of $\lambda = 850\,\mu m$. Here is an example.

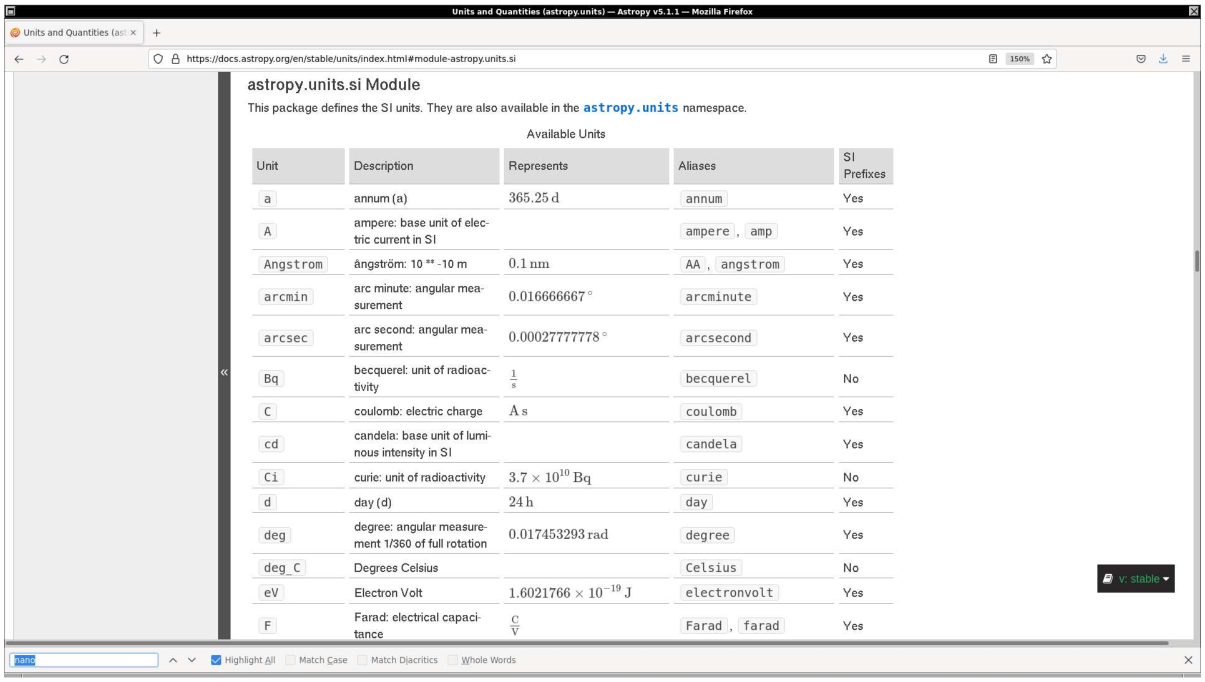
Python Code 16: ai202209_s07_15.py

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

#
# Time-stamp: <2022/10/28 13:33:55 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
```



Units and Quantities (astropy.units) — Astropy v5.1.1 — Mozilla Firefox

https://docs.astropy.org/en/stable/units/index.html#module-astropy.units.si

astropy.units.si Module

This package defines the SI units. They are also available in the `astropy.units` namespace.

Available Units

Unit	Description	Represents	Aliases	SI Prefixes
`a`	annum (a)	$365.25\,\mathrm{d}$	`annum`	Yes
`A`	ampere: base unit of electric current in SI		`ampere`, `amp`	Yes
`Angstrom`	ångström: 10 ** -10 m	$0.1\,\mathrm{nm}$	`AA`, `angstrom`	Yes
`arcmin`	arc minute: angular measurement	$0.016666667\,^{\circ}$	`arcminute`	Yes
`arcsec`	arc second: angular measurement	$0.00027777778\,^{\circ}$	`arcsecond`	Yes
`Bq`	becquerel: unit of radioactivity	$\frac{1}{\mathrm{s}}$	`becquerel`	No
`C`	coulomb: electric charge	$\mathrm{A\,s}$	`coulomb`	Yes
`cd`	candela: base unit of luminous intensity in SI		`candela`	Yes
`Ci`	curie: unit of radioactivity	$3.7 \times 10^{10}\,\mathrm{Bq}$	`curie`	No
`d`	day (d)	$24\,\mathrm{h}$	`day`	Yes
`deg`	degree: angular measurement 1/360 of full rotation	$0.017453293\,\mathrm{rad}$	`degree`	Yes
`deg_C`	Degrees Celsius		`Celsius`	No
`eV`	Electron Volt	$1.6021766 \times 10^{-19}\,\mathrm{J}$	`electronvolt`	Yes
`F`	Farad: electrical capacitance	$\frac{\mathrm{C}}{\mathrm{V}}$	`Farad`, `farad`	Yes

Figure 13: A list of available units with Astropy.

```
u_micron    = astropy.units.micron
u_Hz        = astropy.units.Hz
u_GHz       = astropy.units.GHz
u_spectral = astropy.units.spectral ()

# wavelength
wl = 850 * u_micron

# frequency corresponding to EM wave of wavelength 850 micron
freq     = wl.to (u_Hz, equivalencies=u_spectral)
freq_GHz = wl.to (u_GHz, equivalencies=u_spectral)

# printing result
print (f'wavelength = {wl:g}  ==>  frequency = {freq:g} = {freq_GHz:g}')
```

Execute above script to convert from wavelength into frequency. Note that you need to specify "`equivalencies`" when using `.to ()` method.

```
% chmod a+x ai202209_s07_15.py
% ./ai202209_s07_15.py
wavelength = 850 micron  ==>  frequency = 3.52697e+11 Hz = 352.697 GHz
```

Try following practice.

Practice 07-01

Calculate the frequency corresponding to the electromagnetic wave having the wavelength of $\lambda = 2.2\mu m$.

4.3 Conversion from frequency into wavelength

Try to calculate the wavelength corresponding to the electromagnetic wave having the frequency of $\nu = 115\,GHz$. Here is an example.

Python Code 17: ai202209_s07_16.py

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

#
# Time-stamp: <2022/10/28 13:40:14 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_m        = astropy.units.m
u_mm       = astropy.units.mm
u_GHz      = astropy.units.GHz
u_spectral = astropy.units.spectral ()

# frequency
freq = 115 * u_GHz

# wavelength corresponding to EM wave of frequency 115 GHz
wl    = freq.to (u_m, equivalencies=u_spectral)
wl_mm = freq.to (u_mm, equivalencies=u_spectral)

# printing result
print (f'frequency = {freq:g}  ==>  wavelength = {wl:g} = {wl_mm}')
```

Execute above script to convert from frequency into wavelength.

```
% chmod a+x ai202209_s07_16.py
% ./ai202209_s07_16.py
frequency = 115 GHz  ==>  wavelength = 0.00260689 m = 2.6068909391304347 mm
```

Try following practice.

Practice 07-01

Calculate the wavelength corresponding to the electromagnetic wave having the frequency of $\nu = 345$ Hz.

4.4 Calculating energy of a photon for a given wavelength

Try to calculate the energy of a photon for wavelength of $\lambda = 1\mathring{A}$. Here is an example.

Python Code 18: ai202209_s07_17.py

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

#
# Time-stamp: <2022/10/28 13:48:49 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_AA       = astropy.units.AA
u_eV       = astropy.units.eV
u_keV      = astropy.units.keV
u_spectral = astropy.units.spectral ()
```

```
# wavelength
wl = 1.0 * u_AA

# energy of a photon corresponding to wavelength of 10 AA
energy_eV  = wl.to (u_eV, equivalencies=u_spectral)
energy_keV = wl.to (u_keV, equivalencies=u_spectral)

# printing result
print (f'wavelength = {wl:g}  ==>  energy = {energy_eV:g} = {energy_keV:g}')
```

Execute above script to calculate the energy of a photon.

```
% chmod a+x ai202209_s07_17.py
% ./ai202209_s07_17.py
wavelength = 1 Angstrom  ==>  energy = 12398.4 eV = 12.3984 keV
```

Try following practice.

Practice 07-01

Calculate the energy of a photon having the wavelength of $\lambda = 1 \mu m$.

4.5 Calculation of velocity from Doppler shift

Try a calculation of velocity from the amount of Doppler shift. The rest frame wavelength of Hα emission line is $\lambda_0 = 656.28$ nm. Suppose the observed wavelength of Hα emission line is $\lambda_{obs} = 656.12$ nm. Calculate the velocity corresponding to this Doppler shift. Here is an example Python script.

Python Code 19: ai202209_s07_18.py

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

#
# Time-stamp: <2022/10/28 16:08:10 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_nm          = astropy.units.nm
u_km          = astropy.units.km
u_sec         = astropy.units.s
u_km_per_sec = u_km / u_sec
u_spectral    = astropy.units.spectral ()

# wavelength of H-alpha at rest frame
wl_rest = 656.28 * u_nm

# observed wavelength of H-alpha
wl_obs = 656.12 * u_nm

# calculation of velocity
H_alpha  = astropy.units.doppler_optical (wl_rest)
velocity = wl_obs.to (u_km_per_sec, equivalencies=H_alpha)

# printing result
print (f'rest frame wavelength  = {wl_rest}')
print (f'observed wavelength    = {wl_obs}')
print (f'line-of-sight velocity = {velocity}')
```

Execute above script to calculate the velocity.

```
% chmod a+x ai202209_s07_18.py
% ./ai202209_s07_18.py
rest frame wavelength  = 656.28 nm
observed wavelength    = 656.12 nm
line-of-sight velocity = -73.08891521904629 km / s
```

Try following practice.

Practice 07-01

The doubly ionised oxygen has a spectral line at $\lambda_0 = 495.89$ nm. Suppose the observed wavelength is $\lambda = 495.95$ nm. Calculate corresponding line-of-sight velocity.

4.6 Calculation of equivalent temperature

Suppose X-ray photons have an average energy of 1 keV. Calculate the equivalent temperature. Here is an example Python script.

Python Code 20: ai202209_s07_19.py

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

#
# Time-stamp: <2022/10/28 16:15:35 (CST) daisuke>
#

# importing astropy module
import astropy.units

# units
u_K        = astropy.units.K
u_keV      = astropy.units.keV
u_T_energy = astropy.units.temperature_energy ()

# energy
energy = 1.0 * u_keV

# calculation of equivalent temperature for energy of 100 eV
T = energy.to (u_K, equivalencies=u_T_energy)

# printing result
print (f'energy = {energy:g}  ==>  temperature = {T:g}')
```

Execute above script to calculate the equivalent temperature.

```
% chmod a+x ai202209_s07_19.py
% ./ai202209_s07_19.py
energy = 1 keV  ==>  temperature = 1.16045e+07 K
```

Try following practice.

Practice 07-01

Calculate the equivalent temperature for 100 eV.

4.7 Conversion between mass and energy

Calculate the amount of energy corresponding to the mass difference between 4 protons and 1 helium-4 nucleus. Here is an example.

Python Code 21: ai202209_s07_20.py

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

#
# Time-stamp: <2022/10/28 16:54:34 (CST) daisuke>
#

# importing astropy module
import astropy.constants
import astropy.units

# units
u_kg          = astropy.units.kg
u_MeV         = astropy.units.MeV
u_J           = astropy.units.J
u_J_per_kg    = u_J / u_kg
u_mass_energy = astropy.units.mass_energy ()

# proton mass
m_p = astropy.constants.m_p

# helium-4 nucleus mass
amu = astropy.constants.u
m_He4 = 4.002603254 * amu

# mass difference between 4 protons and 1 He-4 nucleus
m_diff = m_p * 4 - m_He4

# conversion from mass to energy
energy_MeV = m_diff.to (u_MeV, equivalencies=u_mass_energy)
energy_J   = m_diff.to (u_J, equivalencies=u_mass_energy)
specific_energy = energy_J / (m_p * 4)

# printing result
print (f'Delta mass = {m_diff:g}  ==>  energy = {energy_MeV:g}')
print (f'specific energy = {specific_energy:g}')
```

Execute above script to calculate the equivalent energy.

```
% chmod a+x ai202209_s07_20.py
% ./ai202209_s07_20.py
Delta mass = 4.40086e-29 kg  ==>  energy = 24.687 MeV
specific energy = 5.91182e+14 J / kg
```

Try following practice.

Practice 07-01

Calculate the equivalent energy for 2 electron mass.

4.8 Conversion between magnitude and flux density

Calculate flux densities corresponding to V-band magnitude $m_V = 0.0$ and K-band magnitude $m_K = 20.0$. Here is an example.

Python Code 22: ai202209_s07_21.py

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

#
# Time-stamp: <2022/10/28 17:47:13 (CST) daisuke>
#

# importing astropy module
import astropy.constants
import astropy.units

# units
u_ABmag = astropy.units.ABmag
u_Jy    = astropy.units.Jy
u_uJy   = astropy.units.uJy

# V-band magnitude of 0 in AB magnitude system
m_V = 0.0 * u_ABmag

# K-band magnitude of 20 in AB magnitude system
m_K = 20.0 * u_ABmag

# conversion of m_K = 20.0 into Jy
flux_density_V = m_V.to (u_Jy)
flux_density_K = m_K.to (u_uJy)

# printing result
print (f'm_V = {m_V:4.1f}  ==>  flux density = {flux_density_V:g}')
print (f'm_K = {m_K:4.1f}  ==>  flux density = {flux_density_K:g}')
```

Execute above script to calculate the flux densities.

```
% chmod a+x ai202209_s07_21.py
% ./ai202209_s07_21.py
m_V =  0.0 mag(AB)  ==>  flux density = 3630.78 Jy
m_K = 20.0 mag(AB)  ==>  flux density = 36.3078 uJy
```

Try following practice.

Practice 07-01

Calculate the flux density for a source with R-band AB magnitude of $m_R = 22.50$.

5 Astropy's data table

Astropy offers a data structure named `astropy.table`.

5.1 Making an Astropy table

Make a Python script to construct an Astropy table.

Python Code 23: ai202209_s07_22.py

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

#
# Time-stamp: <2022/11/01 08:27:08 (CST) daisuke>
#
```

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

# units
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# data for making Astropy Table
hr       = [2491, 2326, 5340, 5459, 7001, \
            1708, 1713, 2943, 472, 2061]
name     = ['Sirius', 'Canopus', 'Arcturus', 'Alpha Centauri', 'Vega', \
            'Capella', 'Rigel', 'Procyon', 'Achernar', 'Betelgeuse']
vmag     = [-1.46, -0.72, -0.04, -0.01, 0.03, \
            0.08, 0.12, 0.38, 0.46, 0.50] * u_mag
bv       = [0.00, 0.15, 1.23, 0.71, 0.00, \
            0.80, -0.03, 0.42, -0.16, 1.85]
parallax = [0.375, 0.028, 0.090, 0.751, 0.123, \
            0.073, 0.013, 0.288, 0.026, 0.005] * u_arcsec
sptype   = ['A1V', 'F0II', 'K1.5III', 'G2V', 'A0V', \
            'G5III', 'B8I', 'F5IV', 'B3V', 'M2I']

# making Astropy Table
stars = astropy.table.QTable ([hr, name, vmag, bv, parallax, sptype], \
                              names=('HR number', 'Name', 'V-band mag', \
                                     'B-V colour index', 'Parallax', \
                                     'Spectral Type'), \
                              meta={'hr': 'HR number of star', \
                                    'name': 'name of star', \
                                    'vmag': 'V-band apparent magnitude', \
                                    'bv': '(B-V) colour index', \
                                    'parallax': 'parallax in arcsec', \
                                    'sptype': 'spectral type'} )

# printing table
print (stars)
```

Execute above script to construct an Astropy table and print constructed table.

```
% chmod a+x ai202209_s07_22.py
% ./ai202209_s07_22.py
HR number      Name      V-band mag B-V colour index Parallax Spectral Type
                            mag                       arcsec
--------- -------------- ---------- ---------------- -------- -------------
     2491         Sirius      -1.46              0.0    0.375           A1V
     2326        Canopus      -0.72             0.15    0.028          F0II
     5340       Arcturus      -0.04             1.23     0.09       K1.5III
     5459 Alpha Centauri      -0.01             0.71    0.751           G2V
     7001           Vega       0.03              0.0    0.123           A0V
     1708        Capella       0.08              0.8    0.073         G5III
     1713          Rigel       0.12            -0.03    0.013           B8I
     2943        Procyon       0.38             0.42    0.288          F5IV
      472       Achernar       0.46            -0.16    0.026           B3V
     2061     Betelgeuse        0.5             1.85    0.005           M2I
```

Try following practice.

Practice 07-01

Make an Astropy table for some properties of planets in solar system.

Make a Python script to construct an Astropy table and print the information of the table.

Python Code 24: ai202209_s07_23.py

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

#
# Time-stamp: <2022/11/01 08:27:28 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# data for making Astropy Table
hr       = [2491, 2326, 5340, 5459, 7001, \
            1708, 1713, 2943, 472, 2061]
name     = ['Sirius', 'Canopus', 'Arcturus', 'Alpha Centauri', 'Vega', \
            'Capella', 'Rigel', 'Procyon', 'Achernar', 'Betelgeuse']
vmag     = [-1.46, -0.72, -0.04, -0.01, 0.03, \
            0.08, 0.12, 0.38, 0.46, 0.50] * u_mag
bv       = [0.00, 0.15, 1.23, 0.71, 0.00, \
            0.80, -0.03, 0.42, -0.16, 1.85]
parallax = [0.375, 0.028, 0.090, 0.751, 0.123, \
            0.073, 0.013, 0.288, 0.026, 0.005] * u_arcsec
sptype   = ['A1V', 'F0II', 'K1.5III', 'G2V', 'A0V', \
            'G5III', 'B8I', 'F5IV', 'B3V', 'M2I']

# making Astropy Table
stars = astropy.table.QTable ([hr, name, vmag, bv, parallax, sptype], \
                              names=('HR number', 'Name', 'V-band mag', \
                                     'B-V colour index', 'Parallax', \
                                     'Spectral Type'), \
                              meta={'hr': 'HR number of star', \
                                    'name': 'name of star', \
                                    'vmag': 'V-band apparent magnitude', \
                                    'bv': '(B-V) colour index', \
                                    'parallax': 'parallax in arcsec', \
                                    'sptype': 'spectral type'} )

# printing table information
print (stars.info)
```

Execute above script to construct an Astropy table and print the information of the table.

```
% chmod a+x ai202209_s07_23.py
% ./ai202209_s07_23.py
<QTable length=10>
      name        dtype   unit   class
---------------- ------- ------ --------
       HR number   int64          Column
            Name   str14          Column
      V-band mag float64    mag Quantity
B-V colour index float64          Column
        Parallax float64 arcsec Quantity
   Spectral Type    str7          Column
```

Try following practice.

Practice 07-01

Make an Astropy table for some properties of planets in solar system and print the information of the table.

Make a Python script to construct an Astropy table and print the column names of the table.

Python Code 25: ai202209_s07_24.py

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

#
# Time-stamp: <2022/11/01 08:27:38 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# data for making Astropy Table
hr       = [2491, 2326, 5340, 5459, 7001, \
            1708, 1713, 2943, 472, 2061]
name     = ['Sirius', 'Canopus', 'Arcturus', 'Alpha Centauri', 'Vega', \
            'Capella', 'Rigel', 'Procyon', 'Achernar', 'Betelgeuse']
vmag     = [-1.46, -0.72, -0.04, -0.01, 0.03, \
            0.08, 0.12, 0.38, 0.46, 0.50] * u_mag
bv       = [0.00, 0.15, 1.23, 0.71, 0.00, \
            0.80, -0.03, 0.42, -0.16, 1.85]
parallax = [0.375, 0.028, 0.090, 0.751, 0.123, \
            0.073, 0.013, 0.288, 0.026, 0.005] * u_arcsec
sptype   = ['A1V', 'F0II', 'K1.5III', 'G2V', 'A0V', \
            'G5III', 'B8I', 'F5IV', 'B3V', 'M2I']

# making Astropy Table
stars = astropy.table.QTable ([hr, name, vmag, bv, parallax, sptype], \
                              names=('HR number', 'Name', 'V-band mag', \
                                     'B-V colour index', 'Parallax', \
                                     'Spectral Type'), \
                              meta={'hr': 'HR number of star', \
                                    'name': 'name of star', \
                                    'vmag': 'V-band apparent magnitude', \
                                    'bv': '(B-V) colour index', \
                                    'parallax': 'parallax in arcsec', \
                                    'sptype': 'spectral type'} )

# printing column names of table
print (f'stars.colnames = {stars.colnames}')
print ()
print (f'stars.columns = {stars.columns}')
```

Execute above script to construct an Astropy table and print the column names of the table.

```
% chmod a+x ai202209_s07_24.py
% ./ai202209_s07_24.py
stars.colnames = ['HR number', 'Name', 'V-band mag', 'B-V colour index', 'Parall
ax', 'Spectral Type']

stars.columns = <TableColumns names=('HR number','Name','V-band mag','B-V colour
 index','Parallax','Spectral Type')>
```

Try following practice.

Practice 07-01

Make an Astropy table for some properties of planets in solar system and print the column names of the table.

Make a Python script to construct an Astropy table and print the metadata of the table.

Python Code 26: ai202209_s07_25.py

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

#
# Time-stamp: <2022/11/01 08:28:01 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# data for making Astropy Table
hr       = [2491, 2326, 5340, 5459, 7001, \
            1708, 1713, 2943, 472, 2061]
name     = ['Sirius', 'Canopus', 'Arcturus', 'Alpha Centauri', 'Vega', \
            'Capella', 'Rigel', 'Procyon', 'Achernar', 'Betelgeuse']
vmag     = [-1.46, -0.72, -0.04, -0.01, 0.03, \
            0.08, 0.12, 0.38, 0.46, 0.50] * u_mag
bv       = [0.00, 0.15, 1.23, 0.71, 0.00, \
            0.80, -0.03, 0.42, -0.16, 1.85]
parallax = [0.375, 0.028, 0.090, 0.751, 0.123, \
            0.073, 0.013, 0.288, 0.026, 0.005] * u_arcsec
sptype   = ['A1V', 'F0II', 'K1.5III', 'G2V', 'A0V', \
            'G5III', 'B8I', 'F5IV', 'B3V', 'M2I']

# making Astropy Table
stars = astropy.table.QTable ([hr, name, vmag, bv, parallax, sptype], \
                              names=('HR number', 'Name', 'V-band mag', \
                                     'B-V colour index', 'Parallax', \
                                     'Spectral Type'), \
                              meta={'hr': 'HR number of star', \
                                    'name': 'name of star', \
                                    'vmag': 'V-band apparent magnitude', \
                                    'bv': '(B-V) colour index', \
                                    'parallax': 'parallax in arcsec', \
                                    'sptype': 'spectral type'} )
```

```
# printing metadata of table
for key in stars.meta.keys ():
    print (f'{key:8s} : {stars.meta[key]}')
```

Execute above script to construct an Astropy table and print the metadata of the table.

```
% chmod a+x ai202209_s07_25.py
% ./ai202209_s07_25.py
hr       : HR number of star
name     : name of star
vmag     : V-band apparent magnitude
bv       : (B-V) colour index
parallax : parallax in arcsec
sptype   : spectral type
```

Try following practice.

Practice 07-01

Make an Astropy table for some properties of planets in solar system and print the metadata of the table.

5.2 The other ways to make an Astropy table

Here is the other way to construct an Astropy table. Make an empty Astropy table first, and then add data to the table. Here is an example.

Python Code 27: ai202209_s07_26.py

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

#
# Time-stamp: <2022/11/01 08:28:17 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# making an empty Astropy table
stars = astropy.table.QTable ()

# adding data to table
stars['HR number'] = [2491, 2326, 5340, 5459, 7001, \
                      1708, 1713, 2943, 472, 2061]
stars['Name'] = ['Sirius', 'Canopus', 'Arcturus', 'Alpha Centauri', 'Vega', \
                 'Capella', 'Rigel', 'Procyon', 'Achernar', 'Betelgeuse']
stars['V-band mag'] = [-1.46, -0.72, -0.04, -0.01, 0.03, \
                       0.08, 0.12, 0.38, 0.46, 0.50] * u_mag
stars['(B-V) colour index'] = [0.00, 0.15, 1.23, 0.71, 0.00, \
                               0.80, -0.03, 0.42, -0.16, 1.85]
stars['Parallax'] = [0.375, 0.028, 0.090, 0.751, 0.123, \
                     0.073, 0.013, 0.288, 0.026, 0.005] * u_arcsec
stars['Spectral type'] = ['A1V', 'F0II', 'K1.5III', 'G2V', 'A0V', \
                          'G5III', 'B8I', 'F5IV', 'B3V', 'M2I']
```

```
# printing table
print (stars)

# printing information of table
print (stars.info)
```

Execute above script to construct an Astropy table and print the information of the table.

```
% chmod a+x ai202209_s07_26.py
% ./ai202209_s07_26.py
HR number      Name      V-band mag (B-V) colour index Parallax Spectral type
                            mag                         arcsec
--------- -------------- ---------- ------------------ -------- -------------
     2491         Sirius      -1.46                0.0    0.375           A1V
     2326        Canopus      -0.72               0.15    0.028          F0II
     5340       Arcturus      -0.04               1.23     0.09       K1.5III
     5459 Alpha Centauri      -0.01               0.71    0.751           G2V
     7001           Vega       0.03                0.0    0.123           A0V
     1708        Capella       0.08                0.8    0.073         G5III
     1713          Rigel       0.12              -0.03    0.013           B8I
     2943        Procyon       0.38               0.42    0.288          F5IV
      472       Achernar       0.46              -0.16    0.026           B3V
     2061     Betelgeuse        0.5               1.85    0.005           M2I
<QTable length=10>
       name         dtype   unit   class
------------------ ------- ------ --------
         HR number   int64          Column
              Name   str14          Column
        V-band mag float64    mag Quantity
(B-V) colour index float64          Column
          Parallax float64 arcsec Quantity
     Spectral type    str7          Column
```

Try following practice.

Practice 07-01

Make an empty Astropy table, and add data to the table.

Here is one more way to construct an Astropy table. Make an empty Astropy table first, and then add rows to the table. Here is an example.

Python Code 28: ai202209_s07_27.py

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

#
# Time-stamp: <2022/10/30 05:15:06 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_ct     = astropy.units.ct
u_sec    = astropy.units.s
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec
```

```
# making an empty Astropy table
stars \
    = astropy.table.QTable (names=('HR number', 'Name', 'V-band mag', \
                                   'B-V colour index', 'Parallax', \
                                   'Spectral Type'), \
                            dtype=('i4', 'U32', 'f8', 'f8', 'f8', 'U16'), \
                            units=(None, None, u_mag, None, u_arcsec, None))

# adding rows to table
stars.add_row ( (2491, 'Sirius',          -1.46 * u_mag, +0.00, \
                 0.375 * u_arcsec, 'A1V') )
stars.add_row ( (2326, 'Canopus',         -0.72 * u_mag, +0.15, \
                 0.028 * u_arcsec, 'F0II') )
stars.add_row ( (5340, 'Arcturus',        -0.04 * u_mag, +1.23, \
                 0.090 * u_arcsec, 'K1.5III') )
stars.add_row ( (5459, 'Alpha Centauri', -0.01 * u_mag, +0.71, \
                 0.751 * u_arcsec, 'G2V') )
stars.add_row ( (7001, 'Vega',            +0.03 * u_mag, +0.00, \
                 0.123 * u_arcsec, 'A0V') )
stars.add_row ( (1708, 'Capella',         +0.08 * u_mag, +0.80, \
                 0.073 * u_arcsec, 'G5III') )
stars.add_row ( (1713, 'Rigel',           +0.12 * u_mag, -0.03, \
                 0.013 * u_arcsec, 'B8I') )
stars.add_row ( (2943, 'Procyon',         +0.38 * u_mag, +0.42, \
                 0.288 * u_arcsec, 'F5IV') )
stars.add_row ( ( 472, 'Achernar',        +0.46 * u_mag, -0.16, \
                  0.026 * u_arcsec, 'B3V') )
stars.add_row ( (2061, 'Betelgeuse',      +0.50 * u_mag, +1.85, \
                 0.005 * u_arcsec, 'M2I') )

# printing table
print (stars)

# printing information of table
print (stars.info)
```

Execute above script to construct an Astropy table and print the information of the table.

```
% chmod a+x ai202209_s07_27.py
% ./ai202209_s07_27.py
HR number      Name      V-band mag B-V colour index Parallax Spectral Type
                            mag                        arcsec
--------- -------------- ---------- ---------------- -------- -------------
     2491         Sirius      -1.46              0.0    0.375           A1V
     2326        Canopus      -0.72             0.15    0.028          F0II
     5340       Arcturus      -0.04             1.23     0.09       K1.5III
     5459 Alpha Centauri      -0.01             0.71    0.751           G2V
     7001           Vega       0.03              0.0    0.123           A0V
     1708        Capella       0.08              0.8    0.073         G5III
     1713          Rigel       0.12            -0.03    0.013           B8I
     2943        Procyon       0.38             0.42    0.288          F5IV
      472       Achernar       0.46            -0.16    0.026           B3V
     2061     Betelgeuse        0.5             1.85    0.005           M2I
<QTable length=10>
      name        dtype   unit   class
---------------- ------- ------ --------
       HR number   int32          Column
            Name   str32          Column
      V-band mag float64    mag Quantity
B-V colour index float64          Column
        Parallax float64 arcsec Quantity
    Spectral Type   str16          Column
```

Try following practice.

Practice 07-01

Make an empty Astropy table, and add rows to the table.

5.3 Accessing to elements of Astropy table

Here is an example of accessing to a specific column of Astropy table.

Python Code 29: ai202209_s07_28.py

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

#
# Time-stamp: <2022/10/30 09:33:51 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_ct     = astropy.units.ct
u_sec    = astropy.units.s
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# making an empty Astropy table
stars \
    = astropy.table.QTable (names=('HR number', 'Name', 'V-band mag', \
                                   'B-V colour index', 'Parallax', \
                                   'Spectral Type'), \
                            dtype=('i4', 'U32', 'f8', 'f8', 'f8', 'U16'), \
                            units=(None, None, u_mag, None, u_arcsec, None))

# adding rows to table
stars.add_row ( (2491, 'Sirius',         -1.46 * u_mag, +0.00, \
                 0.375 * u_arcsec, 'A1V') )
stars.add_row ( (2326, 'Canopus',        -0.72 * u_mag, +0.15, \
                 0.028 * u_arcsec, 'F0II') )
stars.add_row ( (5340, 'Arcturus',       -0.04 * u_mag, +1.23, \
                 0.090 * u_arcsec, 'K1.5III') )
stars.add_row ( (5459, 'Alpha Centauri', -0.01 * u_mag, +0.71, \
                 0.751 * u_arcsec, 'G2V') )
stars.add_row ( (7001, 'Vega',           +0.03 * u_mag, +0.00, \
                 0.123 * u_arcsec, 'A0V') )
stars.add_row ( (1708, 'Capella',        +0.08 * u_mag, +0.80, \
                 0.073 * u_arcsec, 'G5III') )
stars.add_row ( (1713, 'Rigel',          +0.12 * u_mag, -0.03, \
                 0.013 * u_arcsec, 'B8I') )
stars.add_row ( (2943, 'Procyon',        +0.38 * u_mag, +0.42, \
                 0.288 * u_arcsec, 'F5IV') )
stars.add_row ( ( 472, 'Achernar',       +0.46 * u_mag, -0.16, \
                  0.026 * u_arcsec, 'B3V') )
stars.add_row ( (2061, 'Betelgeuse',     +0.50 * u_mag, +1.85, \
                 0.005 * u_arcsec, 'M2I') )

# accessing to a column of table
print (f'printing the column "Name":')
print (stars["Name"])
print ()
print (f'printing the column "Parallax":')
print (stars["Parallax"])
```

Execute above script to print elements of Astropy table.

```
% chmod a+x ai202209_s07_28.py
% ./ai202209_s07_28.py
printing the column "Name":
     Name
--------------
        Sirius
       Canopus
      Arcturus
Alpha Centauri
          Vega
       Capella
         Rigel
       Procyon
      Achernar
    Betelgeuse

printing the column "Parallax":
[0.375 0.028 0.09  0.751 0.123 0.073 0.013 0.288 0.026 0.005] arcsec
```

Try following practice.

Practice 07-01

Print the column “Spectral Type” of the Astropy table of bright stars.

Here is an example of accessing to a specific row of Astropy table.

Python Code 30: ai202209_s07_29.py

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

#
# Time-stamp: <2022/10/30 09:40:00 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_ct     = astropy.units.ct
u_sec    = astropy.units.s
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# making an empty Astropy table
stars \
    = astropy.table.QTable (names=('HR number', 'Name', 'V-band mag', \
                                   'B-V colour index', 'Parallax', \
                                   'Spectral Type'), \
                            dtype=('i4', 'U32', 'f8', 'f8', 'f8', 'U16'), \
                            units=(None, None, u_mag, None, u_arcsec, None))

# adding rows to table
stars.add_row ( (2491, 'Sirius',         -1.46 * u_mag, +0.00, \
                 0.375 * u_arcsec, 'A1V') )
stars.add_row ( (2326, 'Canopus',        -0.72 * u_mag, +0.15, \
                 0.028 * u_arcsec, 'F0II') )
stars.add_row ( (5340, 'Arcturus',       -0.04 * u_mag, +1.23, \
                 0.090 * u_arcsec, 'K1.5III') )
stars.add_row ( (5459, 'Alpha Centauri', -0.01 * u_mag, +0.71, \
                 0.751 * u_arcsec, 'G2V') )
stars.add_row ( (7001, 'Vega',           +0.03 * u_mag, +0.00, \
                 0.123 * u_arcsec, 'A0V') )
stars.add_row ( (1708, 'Capella',        +0.08 * u_mag, +0.80, \
                 0.073 * u_arcsec, 'G5III') )
stars.add_row ( (1713, 'Rigel',          +0.12 * u_mag, -0.03, \
                 0.013 * u_arcsec, 'B8I') )
stars.add_row ( (2943, 'Procyon',        +0.38 * u_mag, +0.42, \
                 0.288 * u_arcsec, 'F5IV') )
stars.add_row ( ( 472, 'Achernar',       +0.46 * u_mag, -0.16, \
                  0.026 * u_arcsec, 'B3V') )
stars.add_row ( (2061, 'Betelgeuse',     +0.50 * u_mag, +1.85, \
                 0.005 * u_arcsec, 'M2I') )

# accessing to a row / rows of table
print (f'printing data of first column:')
print (stars[0])
print ()
print (f'printing data of 5th column:')
print (stars[4])
print ()
print (f'printing data of 6th, 7th, and 8th columns:')
print (stars[5:8])
```

Execute above script to print elements of Astropy table.

```
% chmod a+x ai202209_s07_29.py
% ./ai202209_s07_29.py
printing data of first column:
HR number  Name  V-band mag B-V colour index Parallax Spectral Type
                    mag                       arcsec
--------- ------ ---------- ---------------- -------- -------------
     2491 Sirius      -1.46              0.0    0.375           A1V

printing data of 5th column:
HR number Name V-band mag B-V colour index Parallax Spectral Type
                  mag                       arcsec
--------- ---- ---------- ---------------- -------- -------------
     7001 Vega       0.03              0.0    0.123           A0V

printing data of 6th, 7th, and 8th columns:
HR number   Name  V-band mag B-V colour index Parallax Spectral Type
                     mag                       arcsec
--------- ------- ---------- ---------------- -------- -------------
     1708 Capella       0.08              0.8    0.073         G5III
     1713   Rigel        0.12               -0.03    0.013            B8I
     2943  Procyon       0.38                0.42    0.288           F5IV
```

Try following practice.

Practice 07-01

Print the third row of the Astropy table of bright stars.

Here is an example of accessing to elements of Astropy table.

Python Code 31: ai202209_s07_30.py

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

#
# Time-stamp: <2022/10/30 09:25:08 (CST) daisuke>
#

# importing astropy module
import astropy.table
import astropy.units

# units
u_ct     = astropy.units.ct
u_sec    = astropy.units.s
u_mag    = astropy.units.mag
u_arcsec = astropy.units.arcsec

# making an empty Astropy table
stars \
    = astropy.table.QTable (names=('HR number', 'Name', 'V-band mag', \
                                   'B-V colour index', 'Parallax', \
                                   'Spectral Type'), \
                            dtype=('i4', 'U32', 'f8', 'f8', 'f8', 'U16'), \
                            units=(None, None, u_mag, None, u_arcsec, None))

# adding rows to table
stars.add_row ( (2491, 'Sirius',         -1.46 * u_mag, +0.00, \
                 0.375 * u_arcsec, 'A1V') )
stars.add_row ( (2326, 'Canopus',        -0.72 * u_mag, +0.15, \
                 0.028 * u_arcsec, 'F0II') )
stars.add_row ( (5340, 'Arcturus',       -0.04 * u_mag, +1.23, \
                 0.090 * u_arcsec, 'K1.5III') )
stars.add_row ( (5459, 'Alpha Centauri', -0.01 * u_mag, +0.71, \
                 0.751 * u_arcsec, 'G2V') )
stars.add_row ( (7001, 'Vega',           +0.03 * u_mag, +0.00, \
                 0.123 * u_arcsec, 'A0V') )
stars.add_row ( (1708, 'Capella',        +0.08 * u_mag, +0.80, \
                 0.073 * u_arcsec, 'G5III') )
stars.add_row ( (1713, 'Rigel',          +0.12 * u_mag, -0.03, \
                 0.013 * u_arcsec, 'B8I') )
stars.add_row ( (2943, 'Procyon',        +0.38 * u_mag, +0.42, \
                 0.288 * u_arcsec, 'F5IV') )
stars.add_row ( ( 472, 'Achernar',       +0.46 * u_mag, -0.16, \
                  0.026 * u_arcsec, 'B3V') )
stars.add_row ( (2061, 'Betelgeuse',     +0.50 * u_mag, +1.85, \
                 0.005 * u_arcsec, 'M2I') )

# accessing to elements of table
```

```
print (f'Name of star for the first row:')
print (f'  stars[0]["Name"]              = {stars[0]["Name"]}')
print (f'  stars["Name"][0]              = {stars["Name"][0]}')
print (f'V-band mag of star for the second row:')
print (f'  stars[1]["V-band mag"]        = {stars[1]["V-band mag"]}')
print (f'  stars["V-band mag"][1]        = {stars["V-band mag"][1]}')
print (f'(B-V) colour of star for the 9th row:')
print (f'  stars[9]["B-V colour index"] = {stars[9]["B-V colour index"]}')
print (f'  stars["B-V colour index"][9] = {stars["B-V colour index"][9]}')
print (f'Parallax of star for the 7th row:')
print (f'  stars[7]["Parallax"]          = {stars[7]["Parallax"]}')
print (f'  stars["Parallax"][7]          = {stars["Parallax"][7]}')
```

Execute above script to print elements of Astropy table.

```
% chmod a+x ai202209_s07_30.py
% ./ai202209_s07_30.py
Name of star for the first row:
  stars[0]["Name"]              = Sirius
  stars["Name"][0]              = Sirius
V-band mag of star for the second row:
  stars[1]["V-band mag"]        = -0.72 mag
  stars["V-band mag"][1]        = -0.72 mag
(B-V) colour of star for the 9th row:
  stars[9]["B-V colour index"] = 1.85
  stars["B-V colour index"][9] = 1.85
Parallax of star for the 7th row:
  stars[7]["Parallax"]          = 0.288 arcsec
  stars["Parallax"][7]          = 0.288 arcsec
```

Try following practice.

Practice 07-01

Print the spectral type of Canopus.

5.4 Reading VOTable file as an Astropy table

First, download a VOTable file.

Python Code 32: ai202209_s07_31.py

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

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

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'http://exoplanet.eu/catalog/votable/'
```

```
# output file name
file_output = 'exoplanet.vot'

# printing status
print (f'Now, fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Finished fetching {url_data}!')

# converting raw byte data into string
data_str = data_byte.decode ('utf-8')

# printing status
print (f'Now, writing data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'w') as fh_write:
    # writing data
    fh_write.write (data_str)

# printing status
print (f'Finished writing data into file "{file_output}"!')
```

Execute above script to download a VOTable file.

```
% chmod a+x ai202209_s07_31.py
% ./ai202209_s07_31.py
Now, fetching http://exoplanet.eu/catalog/votable/...
Finished fetching http://exoplanet.eu/catalog/votable/!
Now, writing data into file "exoplanet.vot"...
Finished writing data into file "exoplanet.vot"!
```

Check downloaded file.

```
% file exoplanet.vot
exoplanet.vot: XML 1.0 document, ASCII text
% ls -lF exoplanet.vot
-rw-r--r--  1 daisuke  taiwan  11092826 Oct 30 10:07 exoplanet.vot
% head exoplanet.vot | cut -b 1-80
<?xml version="1.0" encoding="utf-8"?>
<!-- Produced with astropy.io.votable version 2.0.14
     http://www.astropy.org/ -->
<VOTABLE version="1.3" xmlns="http://www.ivoa.net/xml/VOTable/v1.3" xmlns:xsi="h
 <RESOURCE description="Encyclopedia of extrasolar planet" type="results">
  <COOSYS ID="J2000" equinox="J2000" system="eq_FK5"/>
  <TABLE>
   <FIELD ID="name" arraysize="*" datatype="unicodeChar" name="name" ucd="meta.i
    <DESCRIPTION>
     Name of a planet
```

Try following practice.

Practice 07-01

Go to NASA Exoplanet Archive, and download exoplanet data in VOTable format.

Make a Python script to read a VOTable file and construct an Astropy table.

Python Code 33: ai202209_s07_32.py

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

#
# Time-stamp: <2022/10/30 10:27:53 (CST) daisuke>
#

# importing astropy module
import astropy.table

# VOTable file
file_vot = 'exoplanet.vot'

# reading VOTable file and making an Astropy table
table_exoplanet = astropy.table.Table.read (file_vot)

# printing name, mass, semimajor axis, orbital period, detection type,
# and year of discovery of exoplanets
print (table_exoplanet["name", "mass", "semi_major_axis", \
                       "orbital_period", "detection_type", "discovered"])
```

Execute above script to read a VOTable file and make an Astropy table.

```
% chmod a+x ai202209_s07_32.py
% ./ai202209_s07_32.py
   name     mass   semi_major_axis orbital_period  detection_type discovered
          jovMass        AU              d                            yr
--------- ------- --------------- -------------- --------------- ----------
 11 Com b      --            1.29         326.03 Radial Velocity     2008.0
 11 Oph b    21.0           243.0       730000.0         Imaging     2007.0
 11 UMi b      --            1.54         516.22 Radial Velocity     2009.0
 14 And b      --            0.83         185.84 Radial Velocity     2008.0
 14 Her b     9.1           2.845         1763.3 Radial Velocity     2002.0
 14 Her c     6.9            27.4        52596.0 Radial Velocity     2006.0
      ...     ...             ...            ...             ...        ...
tau Gem b      --            1.17          305.5 Radial Velocity     2004.0
ups And b    0.62           0.059        4.61711 Radial Velocity     1996.0
ups And c     9.1           0.861        240.937 Radial Velocity     1999.0
ups And d   23.58            2.55       1281.439 Radial Velocity     1999.0
ups And e      --          5.2456        3848.86 Radial Velocity     2010.0
ups Leo b      --            1.18          385.2 Radial Velocity     2021.0
zet Del B    40.0           907.0             --         Imaging     2014.0
Length = 5225 rows
```

Try following practice.

Practice 07-01

Read the VOTable file of exoplanet data from NASA Exoplanet Archive.

Make a Python script to read a VOTable file, construct an Astropy table, and print the information of the planet "51 Peg b".

Python Code 34: ai202209_s07_33.py

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

#
# Time-stamp: <2022/10/30 10:43:44 (CST) daisuke>
#

# importing astropy module
import astropy.table

# VOTable file
file_vot = 'exoplanet.vot'

# reading VOTable file and making an Astropy table
table_exoplanet = astropy.table.Table.read (file_vot)

# printing information of planet "51 Peg b"
for i in range (len (table_exoplanet)):
    if (table_exoplanet[i]["name"] == "51 Peg b"):
        print (table_exoplanet[i]["name", "mass", "semi_major_axis", \
                                  "orbital_period", "detection_type", \
                                  "discovered"])
```

Execute above script to read a VOTable file and print the information of the planet “51 Peg b”.

```
% chmod a+x ai202209_s07_33.py
% ./ai202209_s07_33.py
  name     mass   semi_major_axis orbital_period  detection_type discovered
         jovMass        AU              d                            yr
-------- ------- --------------- -------------- --------------- ----------
51 Peg b    0.47           0.052         4.2308 Radial Velocity     1995.0
```

Try following practice.

Practice 07-01

Read the VOTable file of exoplanet data from NASA Exoplanet Archive and print information of the planet 51 Peg b.

The same thing can be carried out also by following Python script.

Python Code 35: ai202209_s07_34.py

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

#
# Time-stamp: <2022/10/30 10:45:29 (CST) daisuke>
#

# importing astropy module
import astropy.table

# VOTable file
file_vot = 'exoplanet.vot'

# reading VOTable file and making an Astropy table
table_exoplanet = astropy.table.Table.read (file_vot)

# printing information of planet "51 Peg b"
```

```
mask = (table_exoplanet["name"] == "51 Peg b")
print (table_exoplanet[mask]["name", "mass", "semi_major_axis", \
                             "orbital_period", "detection_type", \
                             "discovered"])
```

Execute above script to read a VOTable file and print the information of the planet "51 Peg b".

```
% chmod a+x ai202209_s07_34.py
% ./ai202209_s07_34.py
  name     mass   semi_major_axis orbital_period  detection_type discovered
         jovMass         AU             d                            yr
-------- ------- ---------------- -------------- ---------------- ----------
51 Peg b    0.47            0.052         4.2308 Radial Velocity      1995.0
```

Try following practice.

Practice 07-01

Read the VOTable file of exoplanet data from NASA Exoplanet Archive and print information of the planet 51 Peg b by using a mask.

Make a Python script to find exoplanets discovered from 1995 to 1997. Here is an example.

Python Code 36: ai202209_s07_35.py

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

#
# Time-stamp: <2022/10/30 10:50:12 (CST) daisuke>
#

# importing astropy module
import astropy.table

# VOTable file
file_vot = 'exoplanet.vot'

# reading VOTable file and making an Astropy table
table_exoplanet = astropy.table.Table.read (file_vot)

# printing information of planet discovered in 1995, 1996, and 1997
mask = (table_exoplanet["discovered"] >= 1995) \
    & (table_exoplanet["discovered"] <= 1997)
print (table_exoplanet[mask]["name", "mass", "semi_major_axis", \
                             "orbital_period", "detection_type", \
                             "discovered"])
```

Execute above script to read a VOTable file and print the information of planets discovered from 1995 to 1997.

```
% chmod a+x ai202209_s07_35.py
% ./ai202209_s07_35.py
      name          mass   semi_major_axis ...  detection_type discovered
                  jovMass         AU       ...                     yr
---------------- ------- ---------------- ... --------------- ----------
      16 Cyg B b      --             1.68 ... Radial Velocity     1996.0
        47 Uma b      --              2.1 ... Radial Velocity     1996.0
        51 Peg b    0.47            0.052 ... Radial Velocity     1995.0
        55 Cnc b    0.84          0.11339 ... Radial Velocity     1996.0
        70 Vir b      --             0.48 ... Radial Velocity     1996.0
```

```
         GJ 229 B     35.0            19.433 ...         Imaging      1995.0
PSR J2051-0827 b     28.3                -- ...          Timing      1996.0
          Teide 1     54.0                -- ...         Imaging      1995.0
      tau Boo A b     5.84             0.046 ... Radial Velocity      1996.0
        ups And b     0.62             0.059 ... Radial Velocity      1996.0
```

Try following practice.

Practice 07-01

Read the VOTable file of exoplanet data from NASA Exoplanet Archive and print information of planets discovered from 2000 and 2002.

Make a Python script to find exoplanets discovered by direct imaging in 2020. Here is an example.

Python Code 37: ai202209_s07_36.py

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

#
# Time-stamp: <2022/10/30 10:56:43 (CST) daisuke>
#

# importing astropy module
import astropy.table

# VOTable file
file_vot = 'exoplanet.vot'

# reading VOTable file and making an Astropy table
table_exoplanet = astropy.table.Table.read (file_vot)

# printing information of planet discovered by direct imaging in 2020
mask = (table_exoplanet["detection_type"] == "Imaging") \
    & (table_exoplanet["discovered"] == 2020)
print (table_exoplanet[mask]["name", "mass", "semi_major_axis", \
                             "orbital_period", "detection_type", \
                             "discovered"])
```

Execute above script to read a VOTable file and print the information of planets discovered by direct imaging in 2020.

```
% chmod a+x ai202209_s07_36.py
% ./ai202209_s07_36.py
           name              mass  semi_major_axis ... detection_type discovered
                           jovMass        AU       ...                    yr
-------------------------- ------- --------------- ... -------------- ----------
       2MASS J1155-7919 b     10.0           582.0 ...        Imaging     2020.0
             HD 33632 A b     50.0            23.6 ...        Imaging     2020.0
              HIP 75056 b     25.0            30.0 ...        Imaging     2020.0
                 Oph 98 b      7.8           200.0 ...        Imaging     2020.0
         TYC 8998-760-1 b     14.0           162.0 ...        Imaging     2020.0
         TYC 8998-760-1 c      6.0           320.0 ...        Imaging     2020.0
WISEA J083011.95+283716.0      8.5              -- ...        Imaging     2020.0
```

Try following practice.

Practice 07-01

Read the VOTable file of exoplanet data from NASA Exoplanet Archive and print information of planets discovered by transit observation in 2015.

6 Date and time

Astropy offers "`Time`" object for handling of date/time.

6.1 Constructing Astropy's Time object

Make a Python script to construct a "`Time`" object. Here is an example.

Python Code 38: ai202209_s07_37.py

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

#
# Time-stamp: <2022/10/30 11:09:48 (CST) daisuke>
#

# importing astropy module
import astropy.time

# date/time in UT as a string
time_str = '2022-10-31T12:00:00'

# printing "time_str"
print (f'type of "time_str"  = {type (time_str)}')
print (f'value of "time_str" = "{time_str}"')

# constructing Astropy's Time object from a string
time = astropy.time.Time (time_str, format='isot', scale='utc')

# printing "time"
print (f'type of "time"      = {type (time)}')
print (f'value of "time"     = "{time}"')
```

Execute above script to construct an Astropy's "`Time`" object.

```
% chmod a+x ai202209_s07_37.py
% ./ai202209_s07_37.py
type of "time_str"  = <class 'str'>
value of "time_str" = "2022-10-31T12:00:00"
type of "time"      = <class 'astropy.time.core.Time'>
value of "time"     = "2022-10-31T12:00:00.000"
```

Try following practice.

Practice 07-01

Make a Time object for 12:00:00 on 01/Jan/2023, and print it.

6.2 Calculating JD and MJD for a given date/time

Make a Python script to calculate JD and MJD for a given date/time. Here is an example.

Python Code 39: ai202209_s07_38.py

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

#
# Time-stamp: <2022/10/30 11:15:09 (CST) daisuke>
#
```

```
# importing astropy module
import astropy.time

# date/time in UT as a string
time_str = '2022-10-31T12:00:00'

# constructing Astropy's Time object from a string
time = astropy.time.Time (time_str, format='isot', scale='utc')

# calculating JD and MJD
time_jd  = time.jd
time_mjd = time.mjd

# printing JD and MJD
print (f'{time} (UT) = JD {time_jd} = MJD {time_mjd}')
```

Execute above script to calculate JD and MJD for a given date/time.

```
% chmod a+x ai202209_s07_38.py
% ./ai202209_s07_38.py
2022-10-31T12:00:00.000 (UT) = JD 2459884.0 = MJD 59883.5
```

Try following practice.

Practice 07-01

Make a Time object for 12:00:00 on 01/Jan/2023, and print JD and MJD corresponding to it.

6.3 Knowing current date/time

To get current date/time, try following.

Python Code 40: ai202209_s07_39.py

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

#
# Time-stamp: <2022/10/30 16:38:51 (CST) daisuke>
#

# importing datetime module
import datetime

# importing astropy module
import astropy.time

# getting current date/time using datetime module
now_datetime = datetime.datetime.utcnow ()

# constructing Astropy's Time object from a string
now = astropy.time.Time (now_datetime, scale='utc')

# printing date/time
print (f'now  = {now}')
print (f'     = JD  {now.jd:14.6f}')
print (f'     = MJD {now.mjd:14.6f}')
```

Execute above script to get current date/time.

```
% chmod a+x ai202209_s07_39.py
% ./ai202209_s07_39.py
now   = 2022-10-30 08:38:53.263584
      = JD   2459882.860339
      = MJD    59882.360339
```

Here is the other way to get current date/time, try following.

Python Code 41: ai202209_s07_40.py

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

#
# Time-stamp: <2022/10/30 17:44:47 (CST) daisuke>
#

# importing astropy module
import astropy.time

# getting current date/time
now = astropy.time.Time.now ()

# printing date/time
print (f'now   = {now}')
print (f'      = JD  {now.jd:14.6f}')
print (f'      = MJD {now.mjd:14.6f}')
```

Execute above script to get current date/time.

```
% chmod a+x ai202209_s07_40.py
% ./ai202209_s07_40.py
now   = 2022-10-30 08:40:39.232545
      = JD   2459882.861565
      = MJD    59882.361565
```

Try following practice.

Practice 07-01

Get the current date/time and print it.

6.4 Calculation of date/time

Make a Python script to calculate time difference between two given dates. Here is an example.

Python Code 42: ai202209_s07_41.py

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

#
# Time-stamp: <2022/10/30 17:44:41 (CST) daisuke>
#

# importing astropy module
import astropy.time

# getting current date/time
now = astropy.time.Time.now ()
```

```
# date/time of J2000
j2000 = astropy.time.Time ('2000-01-01 12:00:00', format='iso', scale='utc')

# time between J2000 and now
delta_t = now - j2000

# printing date/time
print (f'j2000       = {j2000}')
print (f'now         = {now}')
print (f'now - j2000 = {delta_t} day = {delta_t.sec:g} sec')
```

Execute above script to calculate time between J2000 and now.

```
% chmod a+x ai202209_s07_41.py
% ./ai202209_s07_41.py
j2000       = 2000-01-01 12:00:00.000
now         = 2022-10-30 08:51:04.860369
now - j2000 = 8337.86886412464 day = 7.20392e+08 sec
```

Try following practice.

Practice 07-01

Calculate number of days from the date of your birth to today.

Check number of days in year 2020.

Python Code 43: ai202209_s07_42.py

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

#
# Time-stamp: <2022/10/30 17:44:32 (CST) daisuke>
#

# importing astropy module
import astropy.time

# time t1
t1 = astropy.time.Time ('2020-01-01 00:00:00', format='iso', scale='utc')

# time t2
t2 = astropy.time.Time ('2021-01-01 00:00:00', format='iso', scale='utc')

# time between t1 and t2
delta_t = t2 - t1

# printing date/time
print (f't1      = {t1}')
print (f't2      = {t2}')
print (f't2 - t1 = {delta_t} day = {delta_t.sec:g} sec')
```

Execute above script to show number of days in year 2020.

```
% chmod a+x ai202209_s07_42.py
% ./ai202209_s07_42.py
t1      = 2020-01-01 00:00:00.000
t2      = 2021-01-01 00:00:00.000
t2 - t1 = 366.0 day = 3.16224e+07 sec
```

Try following practice.

Practice 07-01

Count the number of days in year 2400.

Check number of days in year 2021.

Python Code 44: ai202209_s07_43.py

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

#
# Time-stamp: <2022/10/30 17:44:24 (CST) daisuke>
#

# importing astropy module
import astropy.time

# time t1
t1 = astropy.time.Time ('2021-01-01 00:00:00', format='iso', scale='utc')

# time t2
t2 = astropy.time.Time ('2022-01-01 00:00:00', format='iso', scale='utc')

# time between t1 and t2
delta_t = t2 - t1

# printing date/time
print (f't1      = {t1}')
print (f't2      = {t2}')
print (f't2 - t1 = {delta_t} day = {delta_t.sec:g} sec')
```

Execute above script to show number of days in year 2021.

```
% chmod a+x ai202209_s07_43.py
% ./ai202209_s07_43.py
t1      = 2021-01-01 00:00:00.000
t2      = 2022-01-01 00:00:00.000
t2 - t1 = 365.0 day = 3.1536e+07 sec
```

Try following practice.

Practice 07-01

Count the number of days in year 2500.

Make a Python script to calculate the date/time of given date/time `t1` plus 12 hours. Here is an example.

Python Code 45: ai202209_s07_44.py

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

#
# Time-stamp: <2022/10/30 17:44:17 (CST) daisuke>
#

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

```
# hour
u_hr = astropy.units.hr

# time t1
t1 = astropy.time.Time ('2022-10-31 18:00:00', format='iso', scale='utc')

# calculation of t1 + 12-hr
t2 = t1 + 12.0 * u_hr

# printing date/time
print (f't1              = {t1}')
print (f't2 = t1 + 12-hr = {t2}')
```

Execute above script to calculate the date/time of given date/time `t1` plus 12 hours.

```
% chmod a+x ai202209_s07_44.py
% ./ai202209_s07_44.py
t1              = 2022-10-31 18:00:00.000
t2 = t1 + 12-hr = 2022-11-01 06:00:00.000
```

Try following practice.

Practice 07-01

Show the date of 01/Nov/2022 + 100 days.

6.5 Calculation of local sidereal time

Make a Python script to calculate the local sidereal time of given date/time and location.

Python Code 46: ai202209_s07_45.py

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

#
# Time-stamp: <2022/10/30 17:30:47 (CST) daisuke>
#

# importing numpy module
import numpy

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

# units
u_hr = astropy.units.hr

# location of observer
longitude = '120d52m25s'
latitude  = '+23d28m07s'

# t0
t0 = astropy.time.Time ('2022-10-31 10:00:00', format='iso', scale='utc', \
                        location=(longitude, latitude) )

# times
delta_t = numpy.linspace (0.0, 12.0, 13) * u_hr
times   = t0 + delta_t
```

```
# calculation of local sidereal time
lsts = times.sidereal_time ('apparent')

# printing results of calculations
print (f'local sidereal time at Lulin ({longitude}, {latitude})')
for i in range (len (times)):
    print (f'UT: {times[i]}  ==>  ', \
           f'LST: {int (lsts[i].hms.h):02d}:{int (lsts[i].hms.m):02d}', \
           f':{lsts[i].hms.s:06.3f}', sep='')
```

Execute above script to calculate the local sidereal time at Lulin Observatory.

```
% chmod a+x ai202209_s07_45.py
% ./ai202209_s07_45.py
Downloading https://datacenter.iers.org/data/9/finals2000A.all
|=========================================| 3.5M/3.5M (100.00%)        18s
local sidereal time at Lulin (120d52m25s, +23d28m07s)
UT: 2022-10-31 10:00:00.000  ==>  LST: 20:42:14.921
UT: 2022-10-31 11:00:00.000  ==>  LST: 21:42:24.778
UT: 2022-10-31 12:00:00.000  ==>  LST: 22:42:34.635
UT: 2022-10-31 13:00:00.000  ==>  LST: 23:42:44.491
UT: 2022-10-31 14:00:00.000  ==>  LST: 00:42:54.348
UT: 2022-10-31 15:00:00.000  ==>  LST: 01:43:04.205
UT: 2022-10-31 16:00:00.000  ==>  LST: 02:43:14.062
UT: 2022-10-31 17:00:00.000  ==>  LST: 03:43:23.918
UT: 2022-10-31 18:00:00.000  ==>  LST: 04:43:33.775
UT: 2022-10-31 19:00:00.000  ==>  LST: 05:43:43.632
UT: 2022-10-31 20:00:00.000  ==>  LST: 06:43:53.488
UT: 2022-10-31 21:00:00.000  ==>  LST: 07:44:03.345
UT: 2022-10-31 22:00:00.000  ==>  LST: 08:44:13.202
```

Try following practice.

Practice 07-01

Calculate the local sidereal time at Mauna Kea at 07:00:00 (UT) on 01/Jan/2023.

6.6 Timezone

Make a Python script to calculate a local time in a given timezone. Here is an example.

Python Code 47: ai202209_s07_46.py

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

#
# Time-stamp: <2022/10/30 17:41:13 (CST) daisuke>
#

# importing numpy module
import numpy

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

# units
u_hr = astropy.units.hr
```

```
# t_utc
t_utc = astropy.time.Time ('2022-10-31 09:00:00', format='iso', scale='utc')

# timezone
timezone_taiwan = astropy.time.TimezoneInfo (utc_offset=+8.0 * u_hr)
timezone_hawaii = astropy.time.TimezoneInfo (utc_offset=-10.0 * u_hr)

# date/time in Taiwan
t_taiwan = t_utc.to_datetime (timezone=timezone_taiwan)

# date/time in Hawaii
t_hawaii = t_utc.to_datetime (timezone=timezone_hawaii)

# printing results
print (f'date/time UTC       = {t_utc}')
print (f'date/time in Taiwan = {t_taiwan}')
print (f'date/time in Hawaii = {t_hawaii}')
```

Execute above script to calculate a local time in a given timezone.

```
% chmod a+x ai202209_s07_46.py
% ./ai202209_s07_46.py
date/time UTC       = 2022-10-31 09:00:00.000
date/time in Taiwan = 2022-10-31 17:00:00+08:00
date/time in Hawaii = 2022-10-30 23:00:00-10:00
```

Try following practice.

Practice 07-01

Show the local time in New York corresponding to 00:00:00 (UT) on 01/Jan/2023.

7 Astronomical coordinates

Handling of astronomical coordinates can be carried out by using “`SkyCoord`” object of Astropy.

7.1 Constructing “`SkyCoord`” object

Make a Python script to construct a “`SkyCoord`” object of Astropy. Here is an example.

Python Code 48: ai202209_s07_47.py

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

#
# Time-stamp: <2022/10/30 18:04:55 (CST) daisuke>
#

# importing astropy module
import astropy.coordinates

# coordinate of Sirius
sirius_ra  = '06h45m08.9s'
sirius_dec = '-16d42m58s'

# making a SkyCoord object
coord = astropy.coordinates.SkyCoord (ra=sirius_ra, dec=sirius_dec, \
                                      frame='icrs')
```

```
# printing coordinate
print (f'Coordinate of Sirius:')
print (f'  (RA, Dec) = ({sirius_ra}, {sirius_dec})')
print (f'            = ({coord.ra}, {coord.dec})')
print (f'            = ({int (coord.ra.hms.h):02d}', \
       f':{int (coord.ra.hms.m):02d}:{coord.ra.hms.s:06.3f}, ', \
       f'{int (coord.dec.dms.d):02d}:{abs (int (coord.dec.dms.m)):02d}:', \
       f'{abs (coord.dec.dms.s):06.3f})', sep='')
print (f'            = ({coord.to_string ("decimal")} deg)')
print (f'            = ({coord.to_string ("hmsdms")})')
```

Execute above script to construct a “SkyCoord” object.

```
% chmod a+x ai202209_s07_47.py
% ./ai202209_s07_47.py
Coordinate of Sirius:
  (RA, Dec) = (06h45m08.9s, -16d42m58s)
            = (101.28708333333331 deg, -16.716111111111111 deg)
            = (06:45:08.900, -16:42:58.000)
            = (101.287 -16.7161 deg)
            = (06h45m08.9s -16d42m58s)
```

Try following practice.

Practice 07-01

Make a SkyCoord object for (RA, Dec) of Vega.

“SkyCoord” object can also be created by following way.

Python Code 49: ai202209_s07_48.py

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

#
# Time-stamp: <2022/10/30 18:09:14 (CST) daisuke>
#

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

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.degree

# coordinate of Sirius
sirius_ra  = '06:45:08.9'
sirius_dec = '-16:42:58'

# making a SkyCoord object
coord = astropy.coordinates.SkyCoord (ra=sirius_ra, dec=sirius_dec, \
                                      unit=(u_ha, u_deg), frame='icrs')

# printing coordinate
print (f'Coordinate of Sirius:')
print (f'  (RA, Dec) = ({sirius_ra}, {sirius_dec})')
print (f'            = ({coord.ra}, {coord.dec})')
print (f'            = ({int (coord.ra.hms.h):02d}', \
       f':{int (coord.ra.hms.m):02d}:{coord.ra.hms.s:06.3f}, ', \
       f'{int (coord.dec.dms.d):02d}:{abs (int (coord.dec.dms.m)):02d}:', \
       f'{abs (coord.dec.dms.s):06.3f})', sep='')
print (f'             = ({coord.to_string ("decimal")} deg)')
print (f'             = ({coord.to_string ("hmsdms")})')
```

Execute above script to construct a "`SkyCoord`" object.

```
% chmod a+x ai202209_s07_48.py
% ./ai202209_s07_48.py
Coordinate of Sirius:
  (RA, Dec) = (06h45m08.9s, -16d42m58s)
            = (101.28708333333331 deg, -16.716111111111111 deg)
            = (06:45:08.900, -16:42:58.000)
            = (101.287 -16.7161 deg)
            = (06h45m08.9s -16d42m58s)
```

Try following practice.

Practice 07-01

Make a SkyCoord object for (RA, Dec) of Antares.

7.2 Getting the position of the Sun

Make a Python script to get the position of the Sun at given date/time.

Python Code 50: ai202209_s07_49.py

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

#
# Time-stamp: <2022/10/30 19:19:15 (CST) daisuke>
#

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-10-31 04:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

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

```
# printing position of the Sun
print (f'position of the Sun as observed at NCU main campus at {t_utc}:')
print (f'  RA:  {int (sun.ra.hms.h):02d}:{int (sun.ra.hms.m):02d}', \
       f':{sun.ra.hms.s:06.3f}', sep='')
print (f'  Dec: {int (sun.dec.dms.d):02d}', \
       f':{abs (int (sun.dec.dms.m)):02d}:{abs (sun.dec.dms.s):06.3f}', \
       sep='')
```

Execute above script to get the position of the Sun. It may take a while to download the ephemeris "DE440".

```
% chmod a+x ai202209_s07_49.py
% ./ai202209_s07_49.py
Downloading https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/de440
.bsp
|==========================================| 119M/119M (100.00%)       1m26s
position of the Sun as observed at NCU main campus at 2022-10-31 04:00:00.000:
  RA:  14:20:10.220
  Dec: -13:58:43.119
```

Try following practice.

Practice 07-01

Get the position of the Sun as observed at Mauna Kea at 20:00:00 (UT) on 01/Jan/2023.

7.3 Getting the position of the Moon

Make a Python script to get the position of the Moon at given date/time.

Python Code 51: ai202209_s07_50.py

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

#
# Time-stamp: <2022/10/30 19:22:16 (CST) daisuke>
#

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-10-31 04:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

# getting position of the Sun
```

```
moon = astropy.coordinates.get_body ('moon', t_utc, location=observer)

# printing position of the Sun
print (f'position of the Moon as observed at NCU main campus at {t_utc}:')
print (f'  RA:  {int (moon.ra.hms.h):02d}:{int (moon.ra.hms.m):02d}', \
       f':{moon.ra.hms.s:06.3f}', sep='')
print (f'  Dec: {int (moon.dec.dms.d):02d}', \
       f':{abs (int (moon.dec.dms.m)):02d}:{abs (moon.dec.dms.s):06.3f}', \
       sep='')
```

Execute above script to get the position of the Moon.

```
% chmod a+x ai202209_s07_50.py
% ./ai202209_s07_50.py
position of the Moon as observed at NCU main campus at 2022-10-31 04:00:00.000:
  RA:  19:45:42.152
  Dec: -26:52:38.652
```

Try following practice.

Practice 07-01

Get the position of the Moon as observed at Mauna Kea at 05:00:00 (UT) on 01/Jan/2023.

7.4 Conversion between coordinate system

Make a Python script to calculate the position of the Sun in equatorial, ecliptic, and alt-az coordinate systems. Here is an example.

Python Code 52: ai202209_s07_51.py

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

#
# Time-stamp: <2022/10/30 20:12:51 (CST) daisuke>
#

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-10-31 04:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

# getting position of the Sun
```

```
sun = astropy.coordinates.get_body ('sun', t_utc, location=observer)

# RA and Dec
(sun_ra, sun_dec) = sun.to_string ('hmsdms').split ()

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

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


# printing position of the Sun
print (f'position of the Sun as observed at NCU main campus at {t_utc}:')
print (f'  (RA, Dec)       = ({sun_ra}, {sun_dec})')
print (f'  (lambda, beta) = ({sun_lambda}, {sun_beta})')
print (f'  (az, alt)       = ({sun_az}, {sun_alt})')
```

Execute above script to get the position of the Sun in equatorial, ecliptic, and alt-az systems.

```
% chmod a+x ai202209_s07_51.py
% ./ai202209_s07_51.py
position of the Sun as observed at NCU main campus at 2022-10-31 04:00:00.000:
  (RA, Dec)       = (14h20m10.21953419s, -13d58m43.11895792s)
  (lambda, beta) = (217.39348258674687 deg, 0.0018846702164613761 deg)
  (az, alt)       = (188.08442472545045 deg, 50.60968844448877 deg)
```

Try following practice.

Practice 07-01

Show the position of the Sun in ecliptic coordinate system at 04:00:00 (UT) on 21/Mar/2023.

Show the position of the Sun at NCU main campus at noon on the day of summer solstice.

Python Code 53: ai202209_s07_52.py

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

#
# Time-stamp: <2022/10/30 20:24:12 (CST) daisuke>
#

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

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

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

```
# date/time in UTC
t_str = '2022-06-21 04:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

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

# RA and Dec
(sun_ra, sun_dec) = sun.to_string ('hmsdms').split ()

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

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


# printing position of the Sun
print (f'position of the Sun as observed at NCU main campus at {t_utc}:')
print (f'  (RA, Dec)      = ({sun_ra}, {sun_dec})')
print (f'  (lambda, beta) = ({sun_lambda}, {sun_beta})')
print (f'  (az, alt)      = ({sun_az}, {sun_alt})')
```

Execute above script to get the position of the Sun in equatorial, ecliptic, and alt-az systems on the day of summer solstice.

```
% chmod a+x ai202209_s07_52.py
% ./ai202209_s07_52.py
position of the Sun as observed at NCU main campus at 2022-06-21 04:00:00.000:
  (RA, Dec)      = (05h57m44.44005144s, +23d26m06.76348586s)
  (lambda, beta) = (89.48171072884502 deg, -0.0029953968577791116 deg)
  (az, alt)      = (204.20357891788979 deg, 88.32207846744066 deg)
```

Try following practice.

Practice 07-01

Show the position of the Sun in alt-az coordinate system at 04:00:00 (UT) on 23/Sep/2023.

Show the position of the Sun at NCU main campus at noon on the day of winter solstice.

Python Code 54: ai202209_s07_53.py

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

#
# Time-stamp: <2022/10/30 20:24:23 (CST) daisuke>
#
```

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-12-22 04:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

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

# RA and Dec
(sun_ra, sun_dec) = sun.to_string ('hmsdms').split ()

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

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


# printing position of the Sun
print (f'position of the Sun as observed at NCU main campus at {t_utc}:')
print (f'  (RA, Dec)      = ({sun_ra}, {sun_dec})')
print (f'  (lambda, beta) = ({sun_lambda}, {sun_beta})')
print (f'  (az, alt)      = ({sun_az}, {sun_alt})')
```

Execute above script to get the position of the Sun in equatorial, ecliptic, and alt-az systems on the day of winter solstice.

```
% chmod a+x ai202209_s07_53.py
% ./ai202209_s07_53.py
position of the Sun as observed at NCU main campus at 2022-12-22 04:00:00.000:
  (RA, Dec)      = (17h59m45.6788732s, -23d26m17.67464344s)
  (lambda, beta) = (269.94523971905323 deg, 0.0028794689754975855 deg)
  (az, alt)      = (181.95742204718792 deg, 41.56560753673294 deg)
```

Try following practice.

Practice 07-01

Show the position of the Moon in alt-az coordinate system at 12:00:00 (UT) on 23/Sep/2023.

Here is an example of coordinate conversion between equatorial system and galactic system.

Python Code 55: ai202209_s07_54.py

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

#
# Time-stamp: <2022/10/30 20:41:22 (CST) daisuke>
#

# importing astropy module
import astropy.coordinates

# RA and Dec of Betelgeuse
ra_str  = '05h55m10.3s'
dec_str = '+07d24m25s'

# making a SkyCoord object
betelgeuse = astropy.coordinates.SkyCoord (ra=ra_str, dec=dec_str, \
                                           frame='icrs', equinox='J2000')
(betelgeuse_ra, betelgeuse_dec) = betelgeuse.to_string ('hmsdms').split ()

# conversion between equatorial system and galactic system
betelgeuse_gal = betelgeuse.galactic
betelgeuse_l   = betelgeuse_gal.l
betelgeuse_b   = betelgeuse_gal.b

# printing coordinate of Betelgeuse
print (f'Coordinate of Betelgeuse:')
print (f'  (RA, Dec) = ({betelgeuse_ra}, {betelgeuse_dec})')
print (f'  (l, b)    = ({betelgeuse_l}, {betelgeuse_b})')
```

Execute above script to show the coordinates of Betelgeuse.

```
% chmod a+x ai202209_s07_54.py
% ./ai202209_s07_54.py
Coordinate of Betelgeuse:
  (RA, Dec) = (05h55m10.3s, +07d24m25s)
  (l, b)    = (199.78733106792814 deg, -8.958680734520657 deg)
```

Try following practice.

Practice 07-01

Show the position of Antares in galactic coordinate system.

Show the coordinate of the Galactic centre, Galactic anti-centre, north Galactic pole, and south galactic pole.

Python Code 56: ai202209_s07_55.py

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

#
# Time-stamp: <2022/10/30 20:54:29 (CST) daisuke>
#
```

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

# unit
u_deg = astropy.units.degree

# Galactic centre
centre_l = 0.0 * u_deg
centre_b = 0.0 * u_deg

# Galactic anti-centre
anti_l = 180.0 * u_deg
anti_b = 0.0 * u_deg

# north Galactic pole
npole_l = 0.0 * u_deg
npole_b = +90.0 * u_deg

# south Galactic pole
spole_l = 0.0 * u_deg
spole_b = -90.0 * u_deg

# making SkyCoord objects
centre = astropy.coordinates.SkyCoord (l=centre_l, b=centre_b, \
                                       frame='galactic')
anti   = astropy.coordinates.SkyCoord (l=anti_l, b=anti_b, \
                                       frame='galactic')
npole  = astropy.coordinates.SkyCoord (l=npole_l, b=npole_b, \
                                       frame='galactic')
spole  = astropy.coordinates.SkyCoord (l=spole_l, b=spole_b, \
                                       frame='galactic')

# conversion between galactic system and equatorial system
centre_radec = centre.transform_to ('icrs')
(centre_ra, centre_dec) = centre_radec.to_string ('hmsdms').split ()
anti_radec = anti.transform_to ('icrs')
(anti_ra, anti_dec) = anti_radec.to_string ('hmsdms').split ()
npole_radec = npole.transform_to ('icrs')
(npole_ra, npole_dec) = npole_radec.to_string ('hmsdms').split ()
spole_radec = spole.transform_to ('icrs')
(spole_ra, spole_dec) = spole_radec.to_string ('hmsdms').split ()

# printing coordinate of Betelgeuse
print (f'Coordinate of Galactic centre:')
print (f'  (l, b)    = ({centre_l}, {centre_b})')
print (f'  (RA, Dec) = ({centre_ra}, {centre_dec})')
print (f'Coordinate of Galactic anti-centre:')
print (f'  (l, b)    = ({anti_l}, {anti_b})')
print (f'  (RA, Dec) = ({anti_ra}, {anti_dec})')
print (f'Coordinate of north Galactic pole:')
print (f'  (l, b)    = ({npole_l}, {npole_b})')
print (f'  (RA, Dec) = ({npole_ra}, {npole_dec})')
print (f'Coordinate of south Galactic pole:')
print (f'  (l, b)    = ({spole_l}, {spole_b})')
print (f'  (RA, Dec) = ({spole_ra}, {spole_dec})')
```

Execute above script to show the Galactic centre, Galactic anti-centre, north Galactic pole, and south galactic pole.

```
% chmod a+x ai202209_s07_55.py
% ./ai202209_s07_55.py
Coordinate of Galactic centre:
  (l, b)    = (0.0 deg, 0.0 deg)
  (RA, Dec) = (17h45m37.19718877s, -28d56m10.23994245s)
Coordinate of Galactic anti-centre:
  (l, b)    = (180.0 deg, 0.0 deg)
  (RA, Dec) = (05h45m37.19718877s, +28d56m10.23994245s)
Coordinate of north Galactic pole:
  (l, b)    = (0.0 deg, 90.0 deg)
  (RA, Dec) = (12h51m26.27469475s, +27d07m41.70869388s)
Coordinate of south Galactic pole:
  (l, b)    = (0.0 deg, -90.0 deg)
  (RA, Dec) = (00h51m26.27469475s, -27d07m41.70869388s)
```

Try following practice.

Practice 07-01

Calculate the coordinate of north ecliptic pole in (RA, Dec).

7.5 Angular distance between two celestial objects on the sky

Calculate the angular distance between the Sun and the Moon on the day of a total lunar eclipse.

Python Code 57: ai202209_s07_56.py

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

#
# Time-stamp: <2022/10/30 21:12:00 (CST) daisuke>
#

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-11-08 10:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

# location of observer: NCU main campus
longitude = '121d11m12s'
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

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

# getting position of the Moon
moon = astropy.coordinates.get_body ('moon', t_utc, location=observer)
```

```
# RA and Dec
(sun_ra, sun_dec)   = sun.to_string ('hmsdms').split ()
(moon_ra, moon_dec) = moon.to_string ('hmsdms').split ()

# calculation of angular distance
separation = sun.separation (moon)

# printing result of calculation
print (f'Position of the Sun:')
print (f'  (RA, Dec) = ({sun_ra}, {sun_dec})')
print (f'Position of the Moon:')
print (f'  (RA, Dec) = ({moon_ra}, {moon_dec})')
print (f'Angular distance between the Sun and the Moon on {t_utc}')
print (f'  separation = {separation}')
```

Execute above script to calculate the angular distance between the Sun and the Moon on the day of total lunar eclipse.

```
% chmod a+x ai202209_s07_56.py
% ./ai202209_s07_56.py
Position of the Sun:
  (RA, Dec) = (14h52m45.20360699s, -16d31m33.73836547s)
Position of the Moon:
  (RA, Dec) = (02h54m02.21882658s, +16d12m23.33916472s)
Angular distance between the Sun and the Moon on 2022-11-08 10:00:00.000
  separation = 179.556250049813 deg
```

Try following practice.

Practice 07-01

Calculate the angular distance between Vega and Altair.

Calculate the angular distance between the Sun and the Moon on the day of a partial solar eclipse.

Python Code 58: ai202209_s07_57.py

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

#
# Time-stamp: <2022/10/30 21:17:53 (CST) daisuke>
#

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

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

# units
u_m = astropy.units.m

# date/time in UTC
t_str = '2022-10-25 11:00:00'
t_utc = astropy.time.Time (t_str, format='iso', scale='utc')

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

```
latitude  = '+24d58m12s'
height    = 151.6 * u_m
observer = astropy.coordinates.EarthLocation (longitude, latitude, height)

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

# getting position of the Moon
moon = astropy.coordinates.get_body ('moon', t_utc, location=observer)

# RA and Dec
(sun_ra, sun_dec)   = sun.to_string ('hmsdms').split ()
(moon_ra, moon_dec) = moon.to_string ('hmsdms').split ()

# calculation of angular distance
separation = sun.separation (moon)

# printing result of calculation
print (f'Position of the Sun:')
print (f'  (RA, Dec) = ({sun_ra}, {sun_dec})')
print (f'Position of the Moon:')
print (f'  (RA, Dec) = ({moon_ra}, {moon_dec})')
print (f'Angular distance between the Sun and the Moon on {t_utc}')
print (f'  separation = {separation}')
```

Execute above script to calculate the angular distance between the Sun and the Moon on the day of total lunar eclipse.

```
% chmod a+x ai202209_s07_57.py
% ./ai202209_s07_57.py
Position of the Sun:
  (RA, Dec) = (13h58m07.49504035s, -12d03m43.92701164s)
Position of the Moon:
  (RA, Dec) = (13h56m36.42502237s, -11d27m53.87879819s)
Angular distance between the Sun and the Moon on 2022-10-25 11:00:00.000
  separation = 0.703344016030799 deg
```

Try following practice.

Practice 07-01

Calculate the angular distance between the Sun and Venus on 12:00:00 (UT) on 01/Mar/2023.

8 Sigma clipping

The sigma clipping algorithm is useful for rejecting outliers in the data.

8.1 Generating a set of random numbers

Make a Python script to generate a set of random numbers of Gaussian distribution.

Python Code 59: ai202209_s07_58.py

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

#
# Time-stamp: <2022/10/30 21:51:27 (CST) daisuke>
#

# importing numpy module
```

```
import numpy

# importing astropy module
import astropy.stats

# parameters for random number generation
mean   = 1000.0
stddev = 30.0
n      = 10**4

# generation of a set of random number of Gaussian distribution
rng  = numpy.random.default_rng ()
data = rng.normal (loc=mean, scale=stddev, size=n)

# printing generated data
print (f'Genearated random numbers:')
print (f'{data}')
print (f'Number of random numbers generated: {len (data)}')
```

Execute above script to generate a set of random numbers.

```
% chmod a+x ai202209_s07_58.py
% ./ai202209_s07_58.py
Genearated random numbers:
[ 997.72938754  973.4101684   971.62134693 ...  995.2542037  1007.53210821
  972.20801465]
Number of random numbers generated: 10000
```

Try following practice.

Practice 07-01

Generate 10^6 random numbers of Gaussian distribution of mean 10,000 and standard deviation of 500.

Make a Python script to generate a set of random numbers of Gaussian distribution and add some outliers.

Python Code 60: ai202209_s07_59.py

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

#
# Time-stamp: <2022/10/30 21:53:31 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.stats

# parameters for random number generation
mean   = 1000.0
stddev = 30.0
n      = 10**4

# generation of a set of random number of Gaussian distribution
rng  = numpy.random.default_rng ()
data = rng.normal (loc=mean, scale=stddev, size=n)

# adding some outliers
```

```
data[0]    +=  2000.0
data[1000] +=  3000.0
data[2000] +=  4000.0
data[3000] +=  5000.0
data[4000] +=  6000.0
data[5000] +=  7000.0
data[6000] +=  8000.0
data[7000] +=  9000.0
data[8000] += 10000.0
data[9000] += 11000.0

# printing generated data
print (f'Genearated random numbers:')
print (f'{data}')
print (f'Number of random numbers generated: {len (data)}')
```

Execute above script to generate a set of random numbers.

```
% chmod a+x ai202209_s07_59.py
% ./ai202209_s07_59.py
[3019.63962971 1031.60054577 1061.77251573 ...  946.35474959  966.63361208
 1006.29443201]
Number of random numbers generated: 10000
```

Try following practice.

Practice 07-01

Generate 10^6 random numbers of Gaussian distribution of mean 10,000 and standard deviation of 500 and add some outliers.

Make a Python script to generate a set of random numbers of Gaussian distribution, add some outliers, calculate mean and standard deviation.

Python Code 61: ai202209_s07_60.py

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

#
# Time-stamp: <2022/10/30 21:56:25 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.stats

# parameters for random number generation
mean   = 1000.0
stddev = 30.0
n      = 10**4

# generation of a set of random number of Gaussian distribution
rng  = numpy.random.default_rng ()
data = rng.normal (loc=mean, scale=stddev, size=n)

# adding some outliers
data[0]    +=  2000.0
data[1000] +=  3000.0
```

```
data[2000] +=  4000.0
data[3000] +=  5000.0
data[4000] +=  6000.0
data[5000] +=  7000.0
data[6000] +=  8000.0
data[7000] +=  9000.0
data[8000] += 10000.0
data[9000] += 11000.0

# mean and stddev
mean1   = numpy.mean (data)
stddev1 = numpy.std (data)

# printing results
print (f'Genearated random numbers:')
print (f'{data}')
print (f'Number of random numbers generated: {len (data)}')
print (f'Simple mean and stddev:')
print (f'  mean1   = {mean1}')
print (f'  stddev1 = {stddev1}')
```

Execute above script to calculate mean and standard deviation.

```
% chmod a+x ai202209_s07_60.py
% ./ai202209_s07_60.py
Genearated random numbers:
[2997.5224849   970.1271292   990.18107422 ... 1042.91909738  990.77787037
  998.7662445 ]
Number of random numbers generated: 10000
Simple mean and stddev:
  mean1   = 1006.2131050608324
  stddev1 = 226.57121630836608
```

Try following practice.

Practice 07-01

Generate 10^6 random numbers of Gaussian distribution of mean 10,000 and standard deviation of 500, add some outliers, and calculate mean, median, and standard deviation.

Make a Python script to generate a set of random numbers of Gaussian distribution, add some outliers, carry out sigma-clipping, and calculate mean and standard deviation.

Python Code 62: ai202209_s07_61.py

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

#
# Time-stamp: <2022/10/30 21:56:14 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.stats

# parameters for random number generation
mean   = 1000.0
stddev = 30.0
```

```
n       = 10**4

# generation of a set of random number of Gaussian distribution
rng  = numpy.random.default_rng ()
data = rng.normal (loc=mean, scale=stddev, size=n)

# adding some outliers
data[0]    +=  2000.0
data[1000] +=  3000.0
data[2000] +=  4000.0
data[3000] +=  5000.0
data[4000] +=  6000.0
data[5000] +=  7000.0
data[6000] +=  8000.0
data[7000] +=  9000.0
data[8000] += 10000.0
data[9000] += 11000.0

# mean and stddev
mean1   = numpy.mean (data)
stddev1 = numpy.std (data)

# sigma clipping
data_clipped = astropy.stats.sigma_clip (data, sigma=3.0, maxiters=10)

# sigma-clipped mean and stddev
mean2   = numpy.mean (data_clipped)
stddev2 = numpy.std (data_clipped)

# printing results
print (f'Genearated random numbers:')
print (f'{data}')
print (f'Number of random numbers generated: {len (data)}')
print (f'Simple mean and stddev:')
print (f'  mean1   = {mean1}')
print (f'  stddev1 = {stddev1}')
print (f'Sigma-clipped data:')
print (f'{data_clipped}')
print (f'Sigma-clipped mean and stddev:')
print (f'  mean2   = {mean2}')
print (f'  stddev2 = {stddev2}')
```

Execute above script to calculate sigma-clipped mean and standard deviation.

```
% chmod a+x ai202209_s07_61.py
% ./ai202209_s07_61.py
Genearated random numbers:
[2979.52410634  989.56727057 1054.85029487 ... 1022.66143474  995.94981993
 1008.41247268]
Number of random numbers generated: 10000
Simple mean and stddev:
  mean1   = 1006.6291328900762
  stddev1 = 226.0470195433362
Sigma-clipped data:
[-- 989.5672705652785 1054.8502948748057 ... 1022.6614347407533
 995.9498199254178 1008.4124726802813]
Sigma-clipped mean and stddev:
  mean2   = 1000.2693501997537
  stddev2 = 29.316184609051852
```

Try following practice.

Practice 07-01

Generate 10^6 random numbers of Gaussian distribution of mean 10,000 and standard deviation of 500, add some outliers, apply sigma-clipping, and calculate mean, median, and standard deviation.

If only mean, median, and standard deviation values are needed, the function "`sigma_clipped_stats`" may be used.

Python Code 63: ai202209_s07_62.py

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

#
# Time-stamp: <2022/10/30 22:03:19 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.stats

# parameters for random number generation
mean   = 1000.0
stddev = 30.0
n      = 10**4

# generation of a set of random number of Gaussian distribution
rng  = numpy.random.default_rng ()
data = rng.normal (loc=mean, scale=stddev, size=n)

# adding some outliers
data[0]    +=  2000.0
data[1000] +=  3000.0
data[2000] +=  4000.0
data[3000] +=  5000.0
data[4000] +=  6000.0
data[5000] +=  7000.0
data[6000] +=  8000.0
data[7000] +=  9000.0
data[8000] += 10000.0
data[9000] += 11000.0

# mean and stddev
mean1   = numpy.mean (data)
median1 = numpy.median (data)
stddev1 = numpy.std (data)

# sigma clipping
mean2, median2, stddev2 = astropy.stats.sigma_clipped_stats (data, \
                                                              sigma=3.0, \
                                                              maxiters=10)

# printing results
print (f'Genearated random numbers:')
print (f'{data}')
print (f'Number of random numbers generated: {len (data)}')
print (f'Simple mean and stddev:')
print (f'  mean1   = {mean1}')
print (f'  median1 = {median1}')
print (f'  stddev1 = {stddev1}')
print (f'Sigma-clipped mean and stddev:')
print (f'  mean2   = {mean2}')
print (f'  median2 = {median2}')
print (f'  stddev2 = {stddev2}')
```

Execute above script to calculate sigma-clipped mean, median, and standard deviation.

```
% chmod a+x ai202209_s07_62.py
% ./ai202209_s07_62.py
Genearated random numbers:
[2993.84266692 1073.67931704  996.18737623 ... 1015.2494613  1026.28273835
 1008.4201166 ]
Number of random numbers generated: 10000
Simple mean and stddev:
  mean1   = 1006.5885006544981
  median1 = 999.9140589227344
  stddev1 = 226.8229179054472
Sigma-clipped mean and stddev:
  mean2   = 1000.0180985155611
  median2 = 999.8663366421304
  stddev2 = 29.601990708350197
```

Try following practice.

Practice 07-01

Generate 10^6 random numbers of Gaussian distribution of mean 10,000 and standard deviation of 500, add some outliers, apply sigma-clipping, and calculate mean, median, and standard deviation using the function `sigma_clipped_stats`.

9 FITS file I/O

FITS (Flexible Image Transport System) is the de-facto standard file format for astronomy.

9.1 Using name resolver

Make a Python script to carry out name resolving using Simbad. Here is an example of getting the coordinate of globular cluster M3 using name resolver.

Python Code 64: ai202209_s07_63.py

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

#
# Time-stamp: <2022/10/30 22:26:18 (CST) daisuke>
#

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

# importing astroquery module
import astroquery.simbad

# object name
object_name = 'M3'

```

```
# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')
```

Execute above script to carry out name resolving.

```
% chmod a+x ai202209_s07_63.py
% ./ai202209_s07_63.py
Target name: "M3"
  RA  = 13h42m11.62s
  Dec = +28d22m38.2s
```

Try following practice.

Practice 07-01

Get the coordinate of M51 by using name resolver.

Here is an example of getting the coordinate of Antares.

Python Code 65: ai202209_s07_64.py

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

#
# Time-stamp: <2022/10/30 22:29:29 (CST) daisuke>
#

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

# importing astroquery module
import astroquery.simbad

# object name
object_name = 'Antares'

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

# name resolver
```

```
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')
```

Execute above script to carry out name resolving.

```
% chmod a+x ai202209_s07_64.py
% ./ai202209_s07_64.py
Target name: "Antares"
  RA  = 16h29m24.4597s
  Dec = -26d25m55.209s
```

Try following practice.

Practice 07-01

Get the coordinate of Deneb by using name resolver.

Here is an example of getting the coordinate of NGC 1234.

Python Code 66: ai202209_s07_65.py

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

#
# Time-stamp: <2022/10/30 22:33:47 (CST) daisuke>
#

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

# importing astroquery module
import astroquery.simbad

# object name
object_name = 'NGC 1234'

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

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]
```

```
# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')
```

Execute above script to carry out name resolving.

```
% chmod a+x ai202209_s07_65.py
% ./ai202209_s07_65.py
Target name: "NGC 1234"
  RA  = 03h09m39s
  Dec = -07d50m43.39s
```

Try following practice.

Practice 07-01

Get the coordinate of NGC 4321 by using name resolver.

9.2 Downloading FITS image

Make a Python script to download DSS (Digitized Sky Survey) image and save a FITS file. Here is an example.

Python Code 67: ai202209_s07_66.py

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

#
# Time-stamp: <2022/10/30 22:43:46 (CST) daisuke>
#

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

# importing astroquery module
import astroquery.simbad
import astroquery.skyview

# object name
object_name = 'M3'

# survey name
survey = 'DSS2 Red'

# field-of-view
fov_arcmin = 30.0
fov_arcsec = fov_arcmin * 60.0
npixel     = int (fov_arcsec)

# output file name
file_output = 'm3.fits'
```

```
# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')

# getting a list of images
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                        survey=survey)

print ("images =", list_image)

# getting images
images = astroquery.skyview.SkyView.get_images (position=coord, \
                                                survey=survey, \
                                                pixels=npixel)

# image
image  = images[0]
header = image[0].header
data   = image[0].data
print (image.info ())

# writing FITS file
print (f'Writing a FITS file "{file_output}"...')
hdu = astropy.io.fits.PrimaryHDU (data=data, header=header)
hdu.writeto (file_output)
print (f'Done!')
```

Execute above script to download a FITS file.

```
% chmod a+x ai202209_s07_66.py
% ./ai202209_s07_66.py
Target name: "M3"
  RA  = 13h42m11.62s
  Dec = +28d22m38.2s
images = ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv21403476037890.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv21403492811425.fits
|=========================================|  12M/ 12M (100.00%)        29s
Filename: (No file associated with this HDUList)
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (1800, 1800)   float32
None
```

```
Writing a FITS file "m3.fits"...
Done!
```

Check downloaded file.

```
% ls -lF *.fits
-rw-r--r--  1 daisuke  taiwan  12971520 Oct 30 22:44 m3.fits
% file m3.fits
m3.fits: FITS image data, 32-bit, floating point, single precision
```

Try following practice.

Practice 07-01

Download the image of M20.

9.3 Reading a FITS file

Make a Python script to read a FITS file and create a PNG file. Here is an example.

Python Code 68: ai202209_s07_67.py

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

#
# Time-stamp: <2022/10/30 22:54:50 (CST) daisuke>
#

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

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

# input file name
file_input = 'm3.fits'

# output file name
file_output = 'm3.png'

# object name
object_name = 'Globular Cluster M3'

# colour map
cmap = 'gray'

# opening FITS file
with astropy.io.fits.open (file_input) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data
```

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap)
fig.colorbar (im)

# saving file
print (f'{file_input} ==> {file_output}')
fig.savefig (file_output, dpi=450)
```

Execute above script to create a PNG file.

```
% chmod a+x ai202209_s07_67.py
% ./ai202209_s07_67.py
Filename: m3.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (1800, 1800)   float32
None
m3.fits ==> m3.png
```

Display PNG file. (Fig. 14)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
% feh -dF m3.png
```

Try following practice.

Practice 07-01

Download the image of M20, and create a PNG file from FITS file.

Choose different colour map to generate a PNG file.

Python Code 69: ai202209_s07_68.py

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

#
# Time-stamp: <2022/10/31 00:27:41 (CST) daisuke>
#

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

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

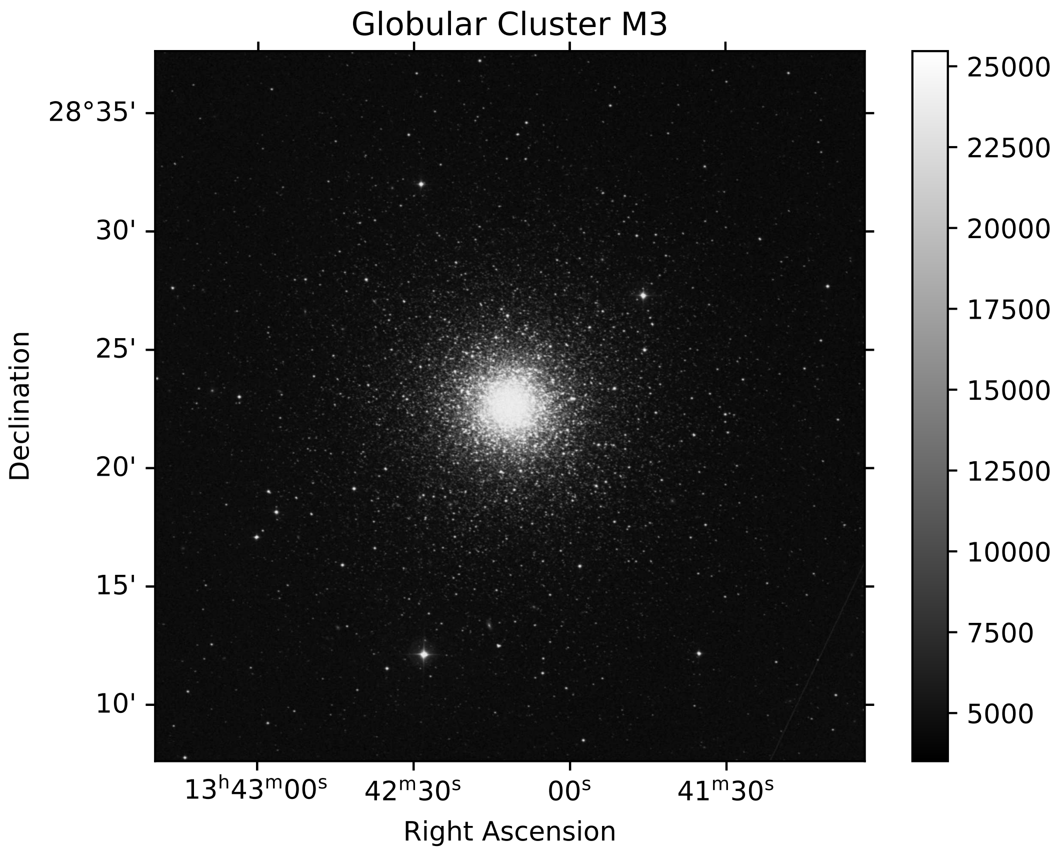


Figure 14: DSS image of globular cluster M3.

```
# input file name
file_input = 'm3.fits'

# output file name
file_output = 'm3_inferno.png'

# object name
object_name = 'Globular Cluster M3'

# colour map
cmap = 'inferno'

# opening FITS file
with astropy.io.fits.open (file_input) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap)
fig.colorbar (im)

# saving file
print (f'{file_input} ==> {file_output}')
fig.savefig (file_output, dpi=450)
```

Execute above script to create a PNG file.

```
% chmod a+x ai202209_s07_68.py
% ./ai202209_s07_68.py
Filename: m3.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (1800, 1800)   float32
None
m3.fits ==> m3_inferno.png
```

Display PNG file. (Fig. 15)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
-rw-r--r--  1 daisuke  taiwan  3580188 Oct 30 23:00 m3_inferno.png
% feh -dF m3_inferno.png
```

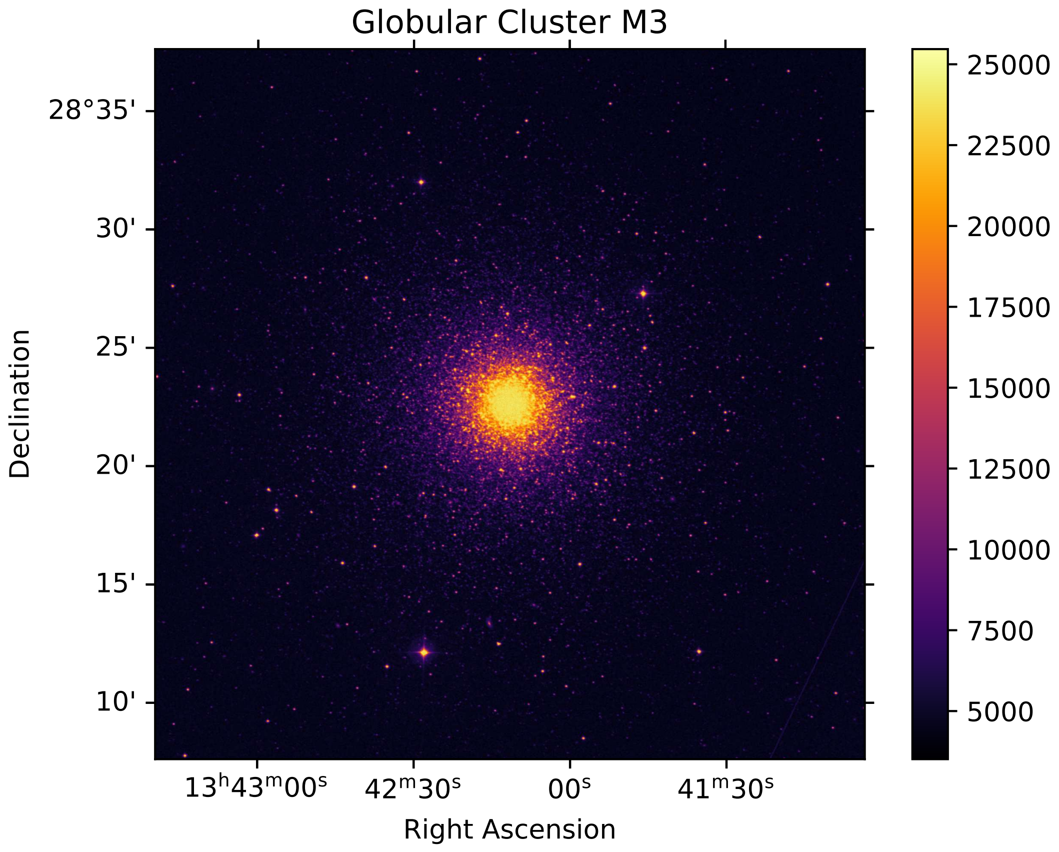


Figure 15: DSS image of globular cluster M3. The colour map of "`inferno`" is used.

Try following practice.

Practice 07-01

Download the image of M20, and create a PNG file from FITS file using colour mas other than "`gray`".

9.4 Downloading a FITS file of M66 and creating a PNG image

Make a Python script to download a FITS file of M66 and creating a PNG image. Here is an example.

Python Code 70: ai202209_s07_69.py

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

#
# Time-stamp: <2022/10/31 00:27:19 (CST) daisuke>
#

# importing astropy module
import astropy.io.fits
import astropy.units
import astropy.coordinates
import astropy.wcs

# importing astroquery module
import astroquery.simbad
import astroquery.skyview

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

# object name
object_name = 'M66'

# survey name
survey = 'DSS2 Red'

# field-of-view
fov_arcmin = 15.0
fov_arcsec = fov_arcmin * 60.0
npixel     = int (fov_arcsec)

# FITS file name
file_fits = 'm66.fits'

# PNG file name
file_png = 'm66.png'

# colour map
cmap = 'bone'

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

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
```

```
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')

# getting a list of images
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                        survey=survey)

print ("images =", list_image)

# getting images
images = astroquery.skyview.SkyView.get_images (position=coord, \
                                                survey=survey, \
                                                pixels=npixel)

# image
image  = images[0]
header = image[0].header
data   = image[0].data
print (image.info ())

# writing FITS file
print (f'Writing a FITS file "{file_fits}"...')
hdu = astropy.io.fits.PrimaryHDU (data=data, header=header)
hdu.writeto (file_fits)
print (f'Done!')

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap)
fig.colorbar (im)
```

```
# saving file
print (f'{file_fits} ==> {file_png}')
fig.savefig (file_png, dpi=450)
```

Execute above script to create a PNG file of M66.

```
% chmod a+x ai202209_s07_69.py
% ./ai202209_s07_69.py
Target name: "M66"
  RA  = 11h20m15.026s
  Dec = +12d59m28.64s
images = ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv21408275567600.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv21408158425577.fits
|========================================| 3.2M/3.2M (100.00%)        11s
Filename: (No file associated with this HDUList)
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (900, 900)   float32
None
Writing a FITS file "m66.fits"...
Done!
Filename: m66.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (900, 900)   float32
None
m66.fits ==> m66.png
```

Display PNG file. (Fig. 16)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
-rw-r--r--  1 daisuke  taiwan  3580188 Oct 30 23:00 m3_inferno.png
-rw-r--r--  1 daisuke  taiwan  1845744 Oct 31 00:02 m66.png
% feh -dF m66.png
```

Try following practice.

Practice 07-01

Download the image of M51, and create a PNG file from FITS file. Use your favourite colour map.

9.5 Downloading a FITS file of M57 and creating a PNG image

Make a Python script to download a FITS file of M57 and creating a PNG image. Here is an example.

Python Code 71: ai202209_s07_70.py

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

#
# Time-stamp: <2022/10/31 00:26:46 (CST) daisuke>
#

# importing astropy module
import astropy.io.fits
import astropy.units
import astropy.coordinates
import astropy.wcs
```

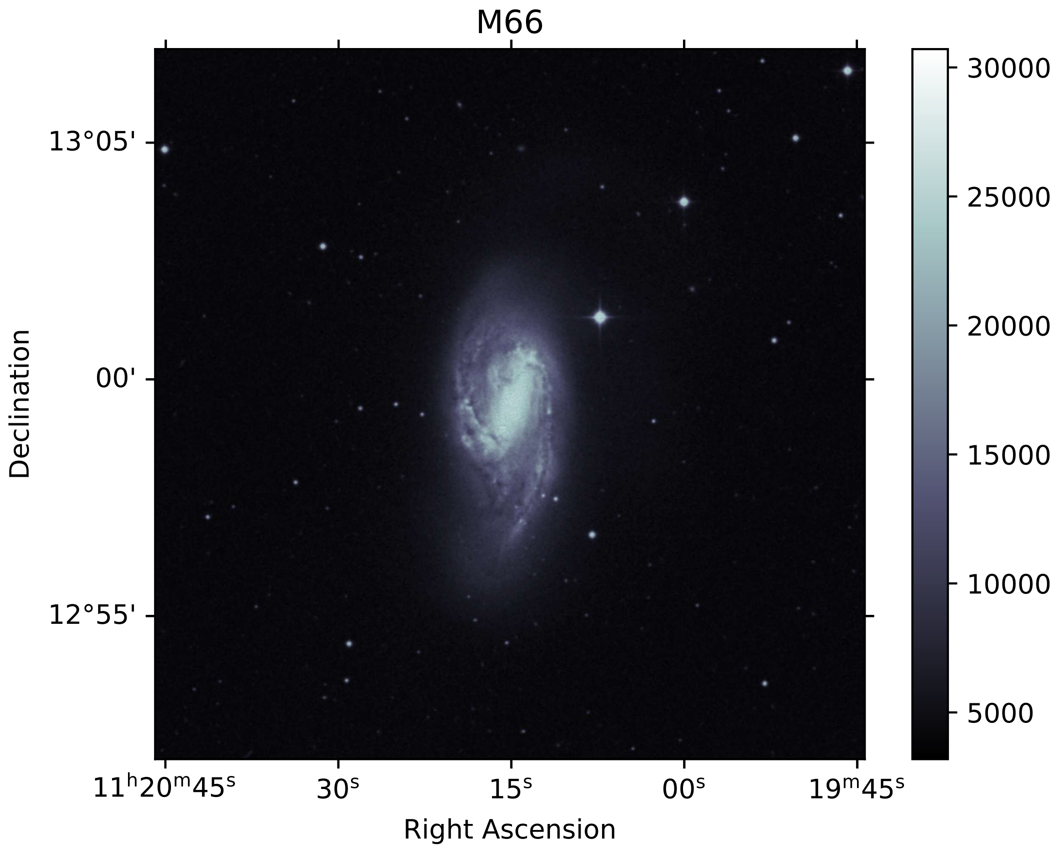


Figure 16: DSS image of spiral galaxy M66. The colour map of “`bone`” is used.

```
# importing astroquery module
import astroquery.simbad
import astroquery.skyview

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

# object name
object_name = 'M57'

# survey name
survey = 'DSS2 Red'

# field-of-view
fov_arcmin = 10.0
fov_arcsec = fov_arcmin * 60.0
npixel     = int (fov_arcsec)

# FITS file name
file_fits = 'm57.fits'

# PNG file name
file_png = 'm57.png'

# colour map
cmap = 'viridis'

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

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')

# getting a list of images
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                        survey=survey)

print ("images =", list_image)

# getting images
images = astroquery.skyview.SkyView.get_images (position=coord, \
                                                survey=survey, \
                                                         pixels=npixel)

# image
image  = images[0]
header = image[0].header
data   = image[0].data
print (image.info ())

# writing FITS file
print (f'Writing a FITS file "{file_fits}"...')
hdu = astropy.io.fits.PrimaryHDU (data=data, header=header)
hdu.writeto (file_fits)
print (f'Done!')

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap)
fig.colorbar (im)

# saving file
print (f'{file_fits} ==> {file_png}')
fig.savefig (file_png, dpi=450)
```

Execute above script to create a PNG file of M57.

```
% chmod a+x ai202209_s07_70.py
% ./ai202209_s07_70.py
Target name: "M57"
  RA  = 18h53m35.0967s
  Dec = +33d01m44.883s
images = ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv21408660273238.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv21408518871515.fits
|=========================================| 1.4M/1.4M (100.00%)         3s
Filename: (No file associated with this HDUList)
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (600, 600)   float32
None
Writing a FITS file "m57.fits"...
Done!
Filename: m57.fits
```

```
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (600, 600)   float32
None
m57.fits ==> m57.png
```

Display PNG file. (Fig. 17)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
-rw-r--r--  1 daisuke  taiwan  3580188 Oct 30 23:00 m3_inferno.png
-rw-r--r--  1 daisuke  taiwan  1998686 Oct 31 00:07 m57.png
-rw-r--r--  1 daisuke  taiwan  1845744 Oct 31 00:02 m66.png
% feh -dF m57.png
```

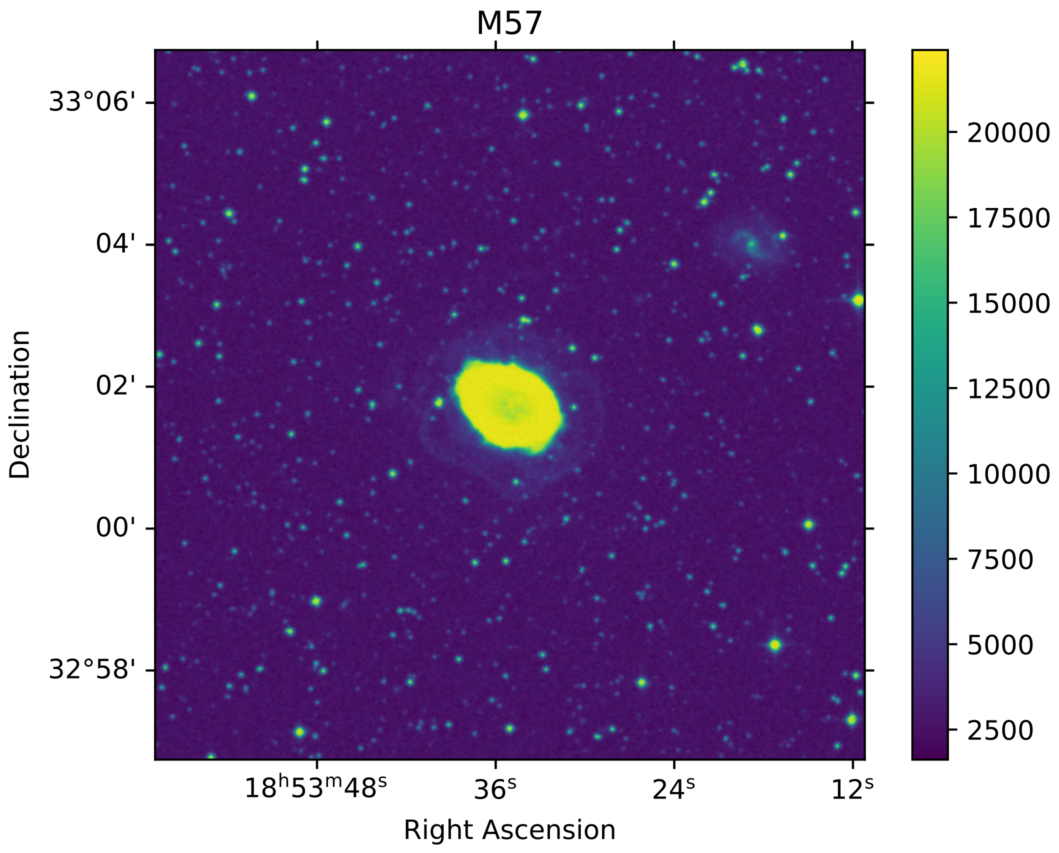


Figure 17: DSS image of the ring nebula M57. The colour map of "`viridis`" is used.

Try following practice.

Practice 07-01

Download the image of M51, and create a PNG file from FITS file. Use your favourite colour map.

Use different image stretching to make a PNG file.

Python Code 72: ai202209_s07_71.py

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

```
#
# Time-stamp: <2022/10/31 00:26:18 (CST) daisuke>
#

# importing astropy module
import astropy.io.fits
import astropy.units
import astropy.coordinates
import astropy.visualization
import astropy.wcs

# importing astroquery module
import astroquery.simbad
import astroquery.skyview

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

# object name
object_name = 'M57'

# FITS file name
file_fits = 'm57.fits'

# PNG file name
file_png = 'm57_norm1.png'

# colour map
cmap = 'viridis'

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# normalisation
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.SinhStretch (0.15) )

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap, norm=norm)
```

```
fig.colorbar (im)

# saving file
print (f'{file_fits} ==> {file_png}')
fig.savefig (file_png, dpi=450)
```

Execute above script to create a PNG file of M57.

```
% chmod a+x ai202209_s07_71.py
% ./ai202209_s07_71.py
Filename: m57.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     123   (600, 600)   float32
None
m57.fits ==> m57_norm1.png
```

Display PNG file. (Fig. 18)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
-rw-r--r--  1 daisuke  taiwan  3580188 Oct 30 23:00 m3_inferno.png
-rw-r--r--  1 daisuke  taiwan  1998686 Oct 31 00:07 m57.png
-rw-r--r--  1 daisuke  taiwan   309471 Oct 31 00:24 m57_norm1.png
-rw-r--r--  1 daisuke  taiwan  1845744 Oct 31 00:02 m66.png
% feh -dF m57_norm1.png
```

Try following practice.

Practice 07-01

Download the image of M51, and create a PNG file from FITS file. Use your favourite colour map. Use image stretching of `AsinhStretch`.

9.6 Downloading a FITS file of Horsehead Nebula

Make a Python script to download a FITS file of Horsehead Nebula and creating a PNG image. Here is an example.

Python Code 73: ai202209_s07_72.py

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

#
# Time-stamp: <2022/10/31 00:40:53 (CST) daisuke>
#

# importing astropy module
import astropy.io.fits
import astropy.units
import astropy.coordinates
import astropy.wcs

# importing astroquery module
import astroquery.simbad
import astroquery.skyview

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

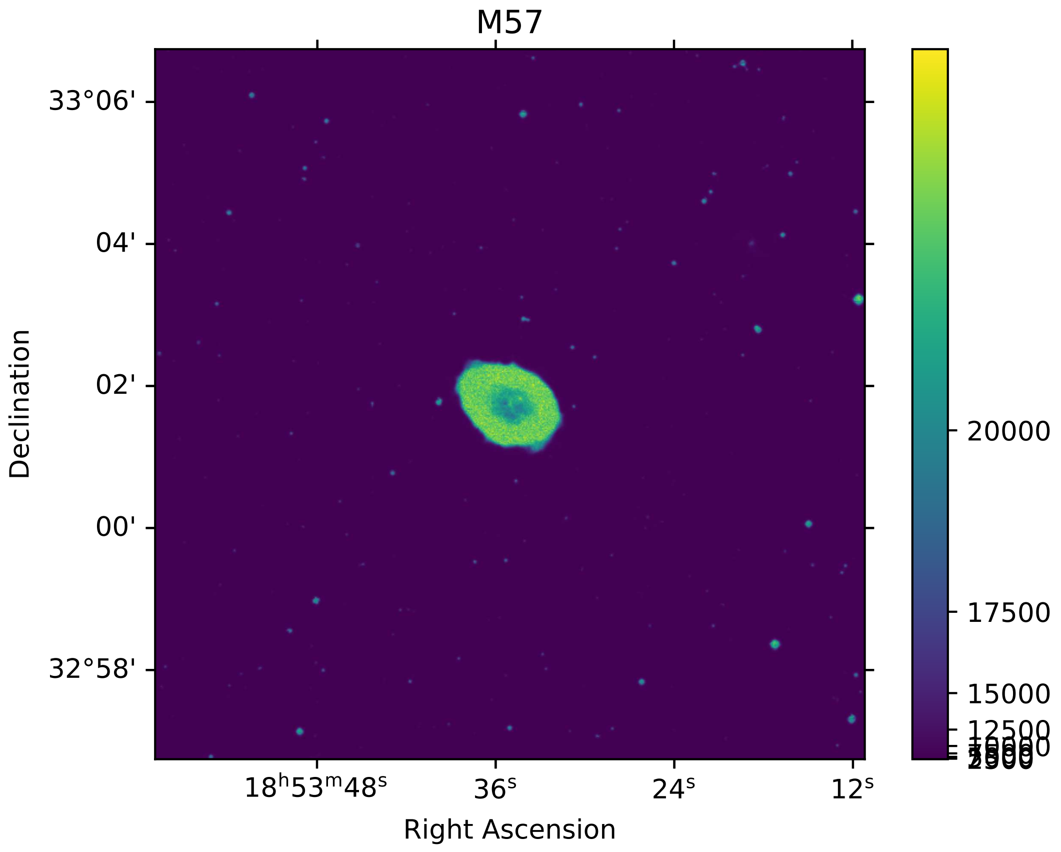


Figure 18: DSS image of the ring nebula M57. The colour map of "`viridis`" is used. The image stretch of `SinhStretch` is used.

```
# object name
object_name = 'Horsehead Nebula'

# survey name
survey = 'DSS2 Red'

# field-of-view
fov_arcmin = 20.0
fov_arcsec = fov_arcmin * 60.0
npixel     = int (fov_arcsec)

# FITS file name
file_fits = 'horsehead.fits'

# PNG file name
file_png = 'horsehead.png'

# colour map
cmap = 'magma'

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

# name resolver
query_result = astroquery.simbad.Simbad.query_object (object_name)

# coordinate from Simbad
ra_str  = query_result['RA'][0]
dec_str = query_result['DEC'][0]

# making SkyCoord object of astropy
coord = astropy.coordinates.SkyCoord (ra_str, dec_str, frame='icrs', \
                                      unit=(u_ha, u_deg) )
(ra, dec) = coord.to_string (style='hmsdms').split ()

# printing result
print (f'Target name: "{object_name}"')
print (f'  RA  = {ra}')
print (f'  Dec = {dec}')

# getting a list of images
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                         survey=survey)

print ("images =", list_image)

# getting images
images = astroquery.skyview.SkyView.get_images (position=coord, \
                                                survey=survey, \
                                                pixels=npixel)

# image
image  = images[0]
header = image[0].header
data   = image[0].data
print (image.info ())
```

```
# writing FITS file
print (f'Writing a FITS file "{file_fits}"...')
hdu = astropy.io.fits.PrimaryHDU (data=data, header=header)
hdu.writeto (file_fits)
print (f'Done!')

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # printing HDU information
    print (hdu_list.info ())

    # reading FITS header, WCS information, and image data
    header = hdu_list[0].header
    wcs    = astropy.wcs.WCS (header)
    image  = hdu_list[0].data

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

# axes
ax.set_title (object_name)
ax.set_xlabel ('Right Ascension')
ax.set_ylabel ('Declination')

# normalisation
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.AsinhStretch () )

# plotting image
im = ax.imshow (image, origin='lower', cmap=cmap, norm=norm)
fig.colorbar (im)

# saving file
print (f'{file_fits} ==> {file_png}')
fig.savefig (file_png, dpi=450)
```

Execute above script to create a PNG file of M57.

```
% chmod a+x ai202209_s07_72.py
% ./ai202209_s07_72.py
Target name: "Horsehead Nebula"
  RA  = 05h40m59s
  Dec = -02d27m30s
images = ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv21410297499461.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv21410608776559.fits
|========================================| 5.7M/5.7M (100.00%)        15s
Filename: (No file associated with this HDUList)
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     130   (1200, 1200)   float32
None
Writing a FITS file "horsehead.fits"...
Done!
Filename: horsehead.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     130   (1200, 1200)   float32
None
```

```
horsehead.fits ==> horsehead.png
```

Display PNG file. (Fig. 19)

```
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  3871365 Oct 31 00:38 horsehead.png
-rw-r--r--  1 daisuke  taiwan  2188783 Oct 30 22:54 m3.png
-rw-r--r--  1 daisuke  taiwan  3580188 Oct 30 23:00 m3_inferno.png
-rw-r--r--  1 daisuke  taiwan  1998686 Oct 31 00:07 m57.png
-rw-r--r--  1 daisuke  taiwan   309471 Oct 31 00:24 m57_norm1.png
-rw-r--r--  1 daisuke  taiwan  1845744 Oct 31 00:02 m66.png
% feh -dF horsehead.png
```

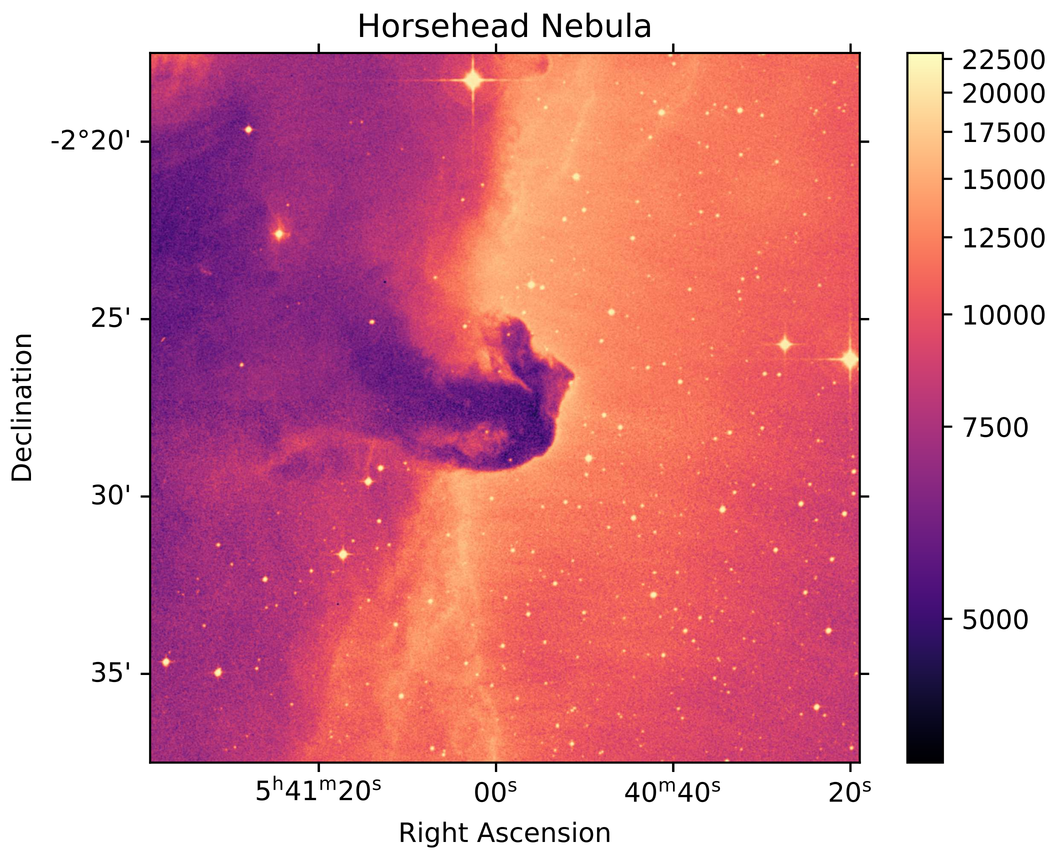


Figure 19: DSS image of the Horsehead Nebula. The colour map of "`magma`" is used. The image stretch `AshinhStretch` is used.

Try following practice.

Practice 07-01

Download the image of your favourite object, and create a PNG file from FITS file. Use your favourite colour map. Use image stretching of `AsinhStretch`.

10 For your further reading

Read following document to learn more about Astropy.

- Astropy documentation: `https://docs.astropy.org/`
  - `astropy.constants`: `https://docs.astropy.org/en/stable/constants/`
  - `astropy.units`: `https://docs.astropy.org/en/stable/units/`
  - `astropy.table`: `https://docs.astropy.org/en/stable/table/`
  - `astropy.time`: `https://docs.astropy.org/en/stable/time/`
  - `astropy.coordinates`: `https://docs.astropy.org/en/stable/coordinates/`
  - `astropy.io.fits`: `https://docs.astropy.org/en/stable/io/fits/`
  - `astropy.visualization`: `https://docs.astropy.org/en/stable/visualization/`
  - `astropy.stats`: `https://docs.astropy.org/en/stable/stats/`

11 Assignment

1. Read the official documentation of Astropy. Summarise what you have learnt.
2. Date and time
   (a) Calculate JD corresponding to 12:00:00 (UT) on 01/Jan/2000.
   (b) Calculate JD corresponding to 12:00:00 (UT) on 01/Jan/2023.
   (c) Calculate MJD corresponding to 18:00:00 (UT) on 01/Aug/2023.
   (d) Calculate the local sidereal time at Paranal Observatory at 23:00:00 (UT) on 01/May/2023.
   (e) Calculate the local sidereal time at Palomar Observatory at 03:00:00 (UT) on 01/Jun/2023.
3. Astronomical coordinates
   (a) Calculate the ecliptic coordinate of Jupiter at 12:00:00 (UT) on 15/Dec/2022.
   (b) Calculate the galactic coordinate of Spica.
   (c) Calculate the alt-az coordinate of Mars at 16:00:00 (UT) on 01/Feb/2023.
   (d) Calculate the equatorial coordinate of south ecliptic pole at 12:00:00 (UT) on 01/Jul/2023.
4. FITS file
   - Choose your favourite astronomical object.
   - Give a brief description about the object you have chosen.
   - Download DSS image of your favourite object.
   - Read the FITS file and create a PNG file.