Advanced Astronomical Observations 2022
Session 01: Basic Python Programming

Kinoshita Daisuke

18 February 2022
publicly accessible version

This course, "Advanced Astronomical Observations", aims to provide an opportunity to learn Python programming for data reduction and analysis of observational data. Topics for this course include, but not limited to,

- manipulation of FITS files,
  - reading and modifying keywords in header,
  - reading and writing image data,
- characterisation of properties of CCD imager using bias, dark, and flatfield images,
  - stability of bias,
  - dark current generation rate,
  - gain (A/D conversion factor) and readout noise,
- standard CCD data reduction,
  - dark subtraction,
  - flatfielding,
- improvement of WCS-related keywords in FITS header,
- aperture photometry
  - construction of PSF profile,
  - photometry of photometric standard stars,
  - atmospheric extinction coefficients,
  - solving transformation equation,

- estimate of sky background brightness,
- estimate of limiting magnitude (detection limit),
- differential photometry and construction of photometric lightcurve,
  - construction of intra-night lightcurve,
  - construction of inter-night lightcurve,
  - period search and construction of folded lightcurve,
- three-colour composite of multi-band data,

1 Suggested operating systems

Many astronomical software is developed on Unix-like operating systems. For this course, use of BSD or Linux operating system is highly encouraged. Here is a list of suggested operating systems.

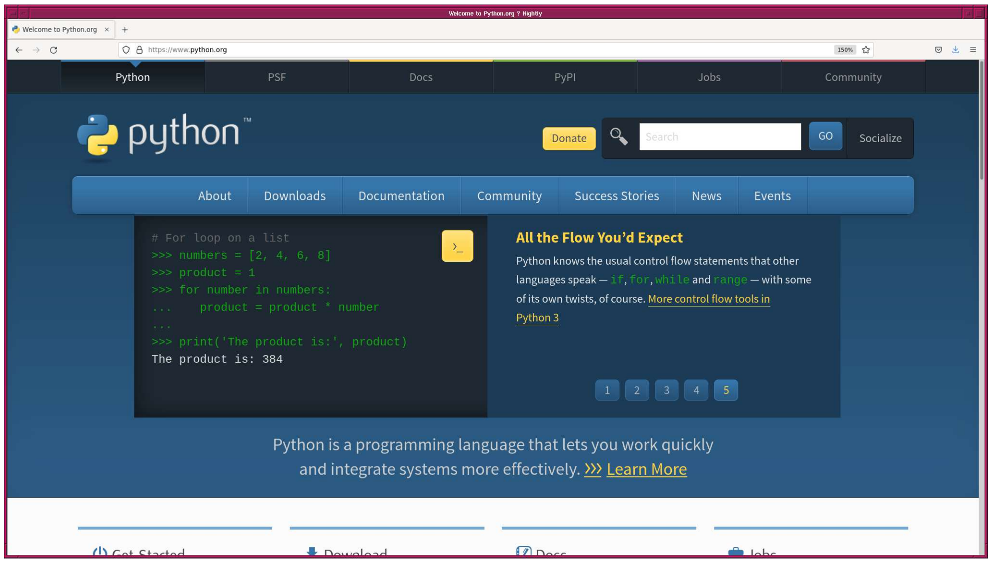
- NetBSD: `https://www.netbsd.org/`
- FreeBSD: `https://www.freebsd.org/`
- Debian GNU/Linux: `https://www.debian.org/`
- Debian based Linux distributions (such as Ubuntu, Linux Mint)

If you are not using a Unix-like operating system, and you are not willing to destroy your favourite operating system, you may consider to use a hypervisor to install BSD operating system or Linux distribution on a virtual machine.

2 Programming language

The official programming language for this course is Python. To learn about Python programming language, visit the official website of Python. (Fig. 1)

- Python: `https://www.python.org/`



```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

All the Flow You'd Expect

Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. More control flow tools in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 1: The official website of Python.

We use Python 3 for our programming for this course. Install Python 3 on your computer. For the installation of Python 3, you may consider to use

- pkgsrc system on NetBSD,

- ports collection on FreeBSD,
- APT package management system on Debian GNU/Linux and its variants.

If you are using NetBSD operating system, try following to install Python 3.

```
% su -
# cd /usr/pkgsrc/lang/python39
# make install
=> Bootstrap dependency digest>=20211023: found digest-20211023
=> Checksum BLAKE2s OK for Python-3.9.10.tar.xz
=> Checksum SHA512 OK for Python-3.9.10.tar.xz
===> Installing dependencies for python39-3.9.10

.....

# make clean
===> Cleaning for python39-3.9.10
# exit
```

If you are using FreeBSD operating system, try following to install Python 3.

```
% su -
# cd /usr/ports/lang/python39
# make -DBATCH install
# make clean
# exit
```

If you are using Debian GNU/Linux or its variant, try following to install Python 3.

```
% su -
# apt install python3
# exit
```

The official documents for Python 3 can be found at following website.

- `https://docs.python.org/3/` (Fig. 2)

If you are new to Python programming, “The Python Tutorial” is highly recommended.

- `https://docs.python.org/3/tutorial/` (Fig. 3)
- `https://docs.python.org/3/download.html` (Fig. 4)

For this course, the use of Python 3.9 is recommended. All the sample Python scripts are developed and tested with Python 3.9. The latest version of Python 3.9 series is 3.9.10 as of February 2022. The installation of Python 3.9.10 on your computer is highly encouraged.

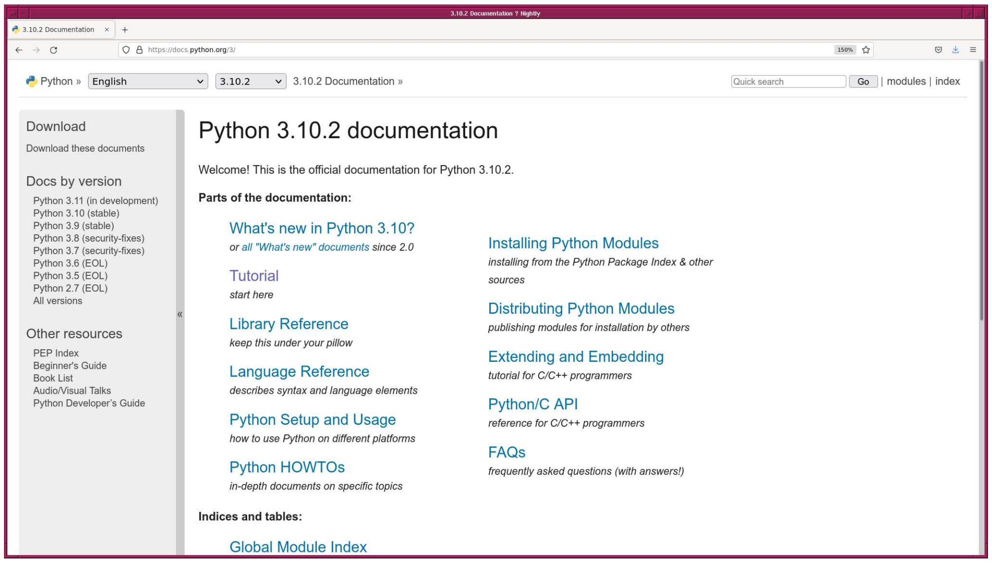
If you prefer to use version 3.10 series of Python, you may go ahead with it. But, make sure to check whether astronomy related external modules of Python, such as Astropy, Astroquery, Photutils, Ginga, etc. work well with Python 3.10.

2.1 Text editor

For this course, you are encouraged to write Python scripts using a text editor. The text editor “GNU Emacs” is highly recommended.

- GNU Emacs: `https://www.gnu.org/software/emacs/` (Fig. 5)

The GNU Emacs has strong supporting functions for writing computer programs.


Python » English 3.10.2 3.10.2 Documentation » Quick search Go | modules | index

Download

Download these documents

Docs by version

Python 3.11 (in development)
Python 3.10 (stable)
Python 3.9 (stable)
Python 3.8 (security-fixes)
Python 3.7 (security-fixes)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 2.7 (EOL)
All versions

Other resources

PEP Index
Beginner's Guide
Book List
Audio/Visual Talks
Python Developer's Guide

Python 3.10.2 documentation

Welcome! This is the official documentation for Python 3.10.2.

Parts of the documentation:

What's new in Python 3.10?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage
how to use Python on different platforms

Python HOWTOs
in-depth documents on specific topics

Installing Python Modules
installing from the Python Package Index & other sources

Distributing Python Modules
publishing modules for installation by others

Extending and Embedding
tutorial for C/C++ programmers

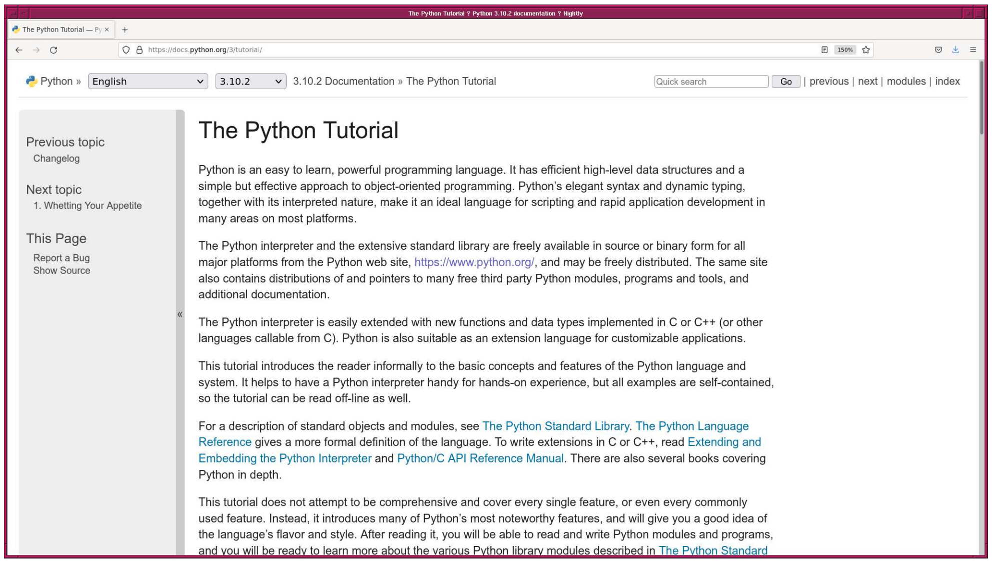
Python/C API
reference for C/C++ programmers

FAQs
frequently asked questions (with answers!)

Indices and tables:

Global Module Index

Figure 2: The official web page of Python 3 documents.


Python » English 3.10.2 3.10.2 Documentation » The Python Tutorial Quick search Go | previous | next | modules | index

Previous topic
Changelog

Next topic
1. Whetting Your Appetite

This Page
Report a Bug
Show Source

The Python Tutorial

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

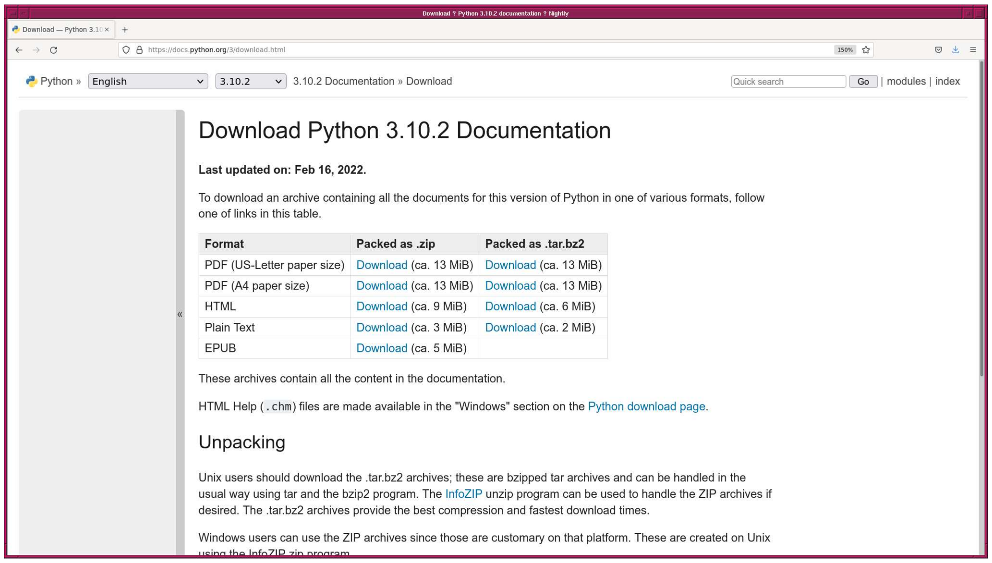
The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see The Python Standard Library. The Python Language Reference gives a more formal definition of the language. To write extensions in C or C++, read Extending and Embedding the Python Interpreter and Python/C API Reference Manual. There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in The Python Standard

Figure 3: "The Python Tutorial" on the official website of Python.


Download — Python 3.10.2 documentation
https://docs.python.org/3/download.html
Python » English 3.10.2 3.10.2 Documentation » Download
Quick search Go | modules | index

Download Python 3.10.2 Documentation

Last updated on: Feb 16, 2022.

To download an archive containing all the documents for this version of Python in one of various formats, follow one of links in this table.

Format	Packed as .zip	Packed as .tar.bz2
PDF (US-Letter paper size)	Download (ca. 13 MiB)	Download (ca. 13 MiB)
PDF (A4 paper size)	Download (ca. 13 MiB)	Download (ca. 13 MiB)
HTML	Download (ca. 9 MiB)	Download (ca. 6 MiB)
Plain Text	Download (ca. 3 MiB)	Download (ca. 2 MiB)
EPUB	Download (ca. 5 MiB)	

These archives contain all the content in the documentation.

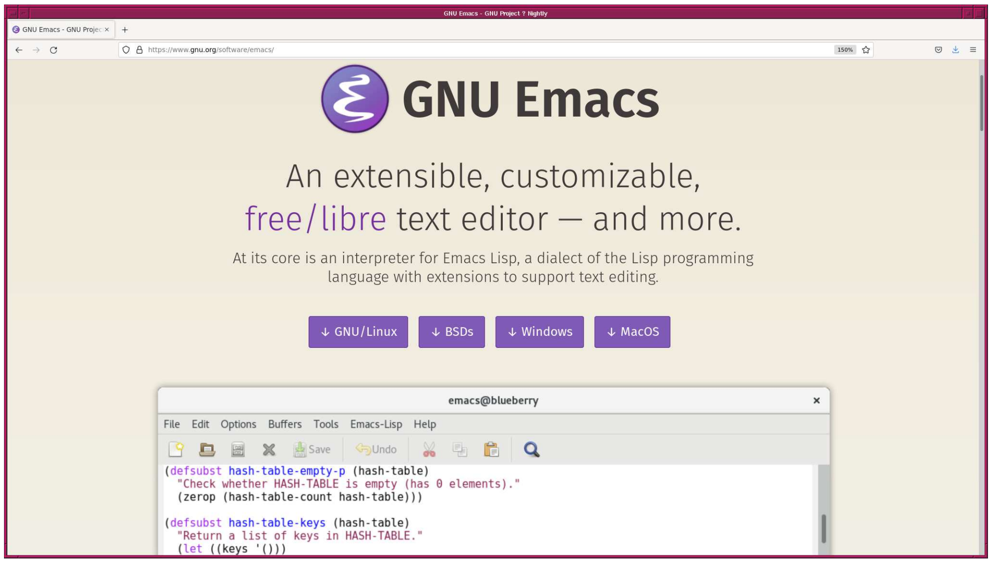
HTML Help (`.chm`) files are made available in the "Windows" section on the Python download page.

Unpacking

Unix users should download the .tar.bz2 archives; these are bzipped tar archives and can be handled in the usual way using tar and the bzip2 program. The InfoZIP unzip program can be used to handle the ZIP archives if desired. The .tar.bz2 archives provide the best compression and fastest download times.

Windows users can use the ZIP archives since those are customary on that platform. These are created on Unix

Figure 4: The Python official web page for downloading documents.


GNU Emacs - GNU Project
https://www.gnu.org/software/emacs/
GNU Emacs
An extensible, customizable,
free/libre text editor — and more.
At its core is an interpreter for Emacs Lisp, a dialect of the Lisp programming language with extensions to support text editing.
↓ GNU/Linux
↓ BSDs
↓ Windows
↓ MacOS
emacs@blueberry
File Edit Options Buffers Tools Emacs-Lisp Help
Save Undo

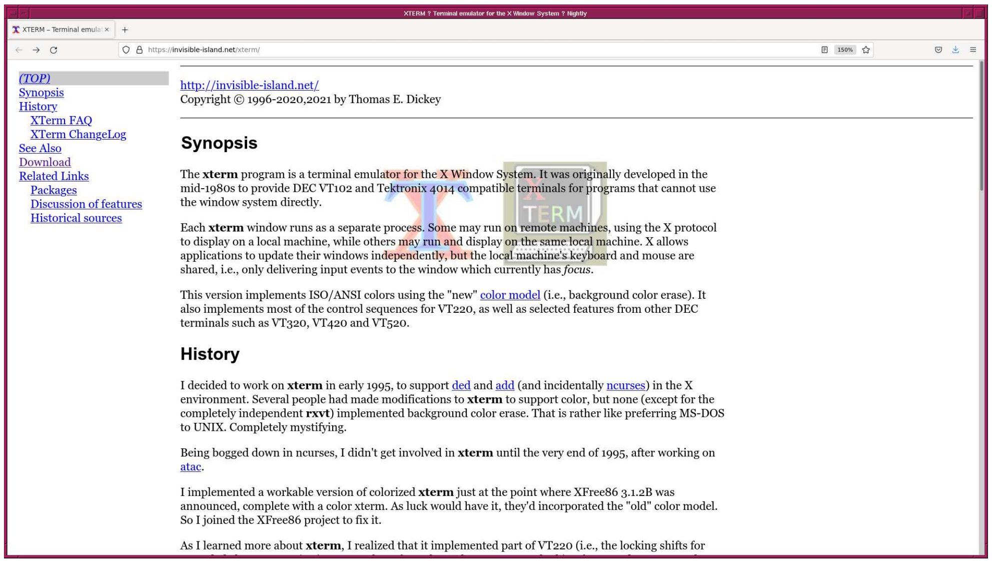
```
(defsubst hash-table-empty-p (hash-table)
  "Check whether HASH-TABLE is empty (has 0 elements)."
  (zerop (hash-table-count hash-table)))

(defsubst hash-table-keys (hash-table)
  "Return a list of keys in HASH-TABLE."
  (let ((keys '()))
```

Figure 5: The official website of GNU Emacs.

2.2 Terminal emulator

When you finish writing a Python script, you execute the script on a terminal emulator. The use of "xterm" is suggested.

- XTerm: `https://invisible-island.net/xterm/` (Fig. 6)


XTERM ? Terminal emulator for the X Window System ? Nightly
XTERM – Terminal emula
https://invisible-island.net/xterm/
150%

(TOP)
Synopsis
History
XTerm FAQ
XTerm ChangeLog
See Also
Download
Related Links
Packages
Discussion of features
Historical sources

http://invisible-island.net/
Copyright © 1996-2020,2021 by Thomas E. Dickey

Synopsis

The **xterm** program is a terminal emulator for the X Window System. It was originally developed in the mid-1980s to provide DEC VT102 and Tektronix 4014 compatible terminals for programs that cannot use the window system directly.

Each **xterm** window runs as a separate process. Some may run on remote machines, using the X protocol to display on a local machine, while others may run and display on the same local machine. X allows applications to update their windows independently, but the local machine's keyboard and mouse are shared, i.e., only delivering input events to the window which currently has *focus*.

This version implements ISO/ANSI colors using the "new" color model (i.e., background color erase). It also implements most of the control sequences for VT220, as well as selected features from other DEC terminals such as VT320, VT420 and VT520.

History

I decided to work on **xterm** in early 1995, to support ded and add (and incidentally ncurses) in the X environment. Several people had made modifications to **xterm** to support color, but none (except for the completely independent **rxvt**) implemented background color erase. That is rather like preferring MS-DOS to UNIX. Completely mystifying.

Being bogged down in ncurses, I didn't get involved in **xterm** until the very end of 1995, after working on atac.

I implemented a workable version of colorized **xterm** just at the point where XFree86 3.1.2B was announced, complete with a color xterm. As luck would have it, they'd incorporated the "old" color model. So I joined the XFree86 project to fix it.

As I learned more about **xterm**, I realized that it implemented part of VT220 (i.e., the locking shifts for

Figure 6: The official website of XTerm.

2.3 Why text editor and terminal emulator?

For this course, use of a text editor and a terminal emulator is encouraged. The reason is that it is more efficient to use a text editor to make a Python script and executing it on a terminal emulator for astronomical data analysis. Use of command-line argument analysis enables you to make a flexible and versatile Python program, and such a single Python program is able to process observational data taken on different nights. It may be able to handle data from different observatories.

3 Suggested readings

Here is a list of suggested readings. If you are new to Python programming, you are encouraged to read these books.

- "Python Tutorial"
  - `https://docs.python.org/3/tutorial/`
- "Learning Python" 3rd Edition, 2008, Mark Lutz, O'Reilly
  - `http://shop.oreilly.com/product/9780596513986.do`
- "Programming Python" 4th Edition, 2010, Mark Lutz, O'Reilly
  - `http://shop.oreilly.com/product/9780596158118.do`
- "Python Cookbook" 2nd Edition, 2008, David Ascher, Alex Martelli, Anna Ravenscroft, O'Reilly
  - `http://shop.oreilly.com/product/9780596007973.do`

"Python Tutorial" is highly recommended, if you have just started to learn Python programming. It is available at Python official website. Many more books about Python can be found at the Library of our University. Visit the Library and try to find your favourite Python textbook.

"The Python Standard Library" of Python can be found at following web page, and it is very useful for learning built-in functions of Python.

- "The Python Standard Library": `https://docs.python.org/3/library/`

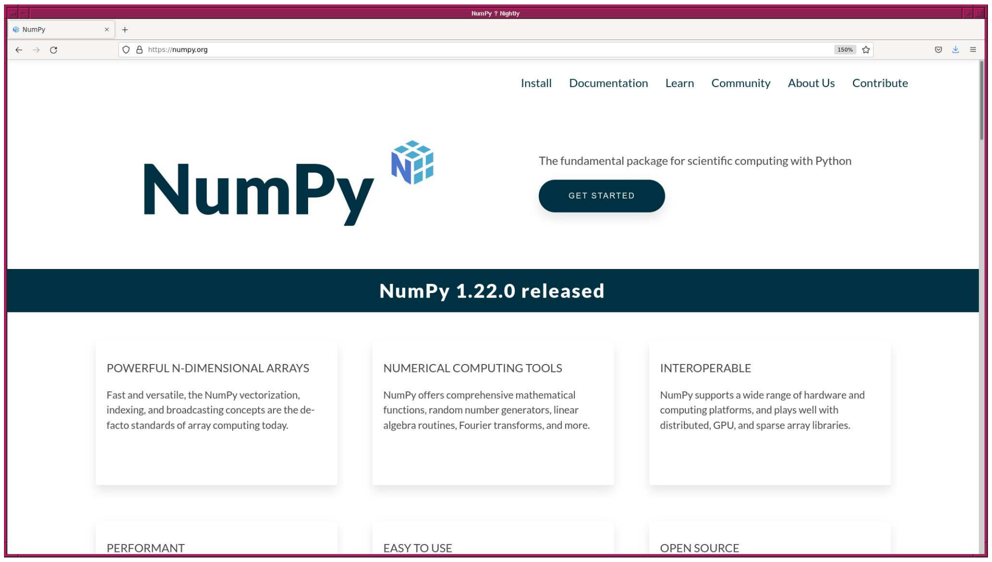
4 Python modules used for this course

Following Python modules will be used for this course. Visit the official websites of these modules, and install them on your computer.

- NumPy
  - `https://numpy.org/` (Fig. 7)
- SciPy
  - `https://scipy.org/` (Fig. 8)
- Matplotlib
  - `https://matplotlib.org/` (Fig. 9)
- Astropy
  - `https://www.astropy.org/` (Fig. 10)
- Astroquery
  - `https://astroquery.readthedocs.io/` (Fig. 11)
- ccdproc
  - `https://ccdproc.readthedocs.io/` (Fig. 12)
- photutils
  - `https://photutils.readthedocs.io/` (Fig. 13)
- gwcs
  - `https://gwcs.readthedocs.io/` (Fig. 14)
- ginga
  - `https://ginga.readthedocs.io/` (Fig. 15)

The documentation of above modules are available on the official website. Read official documents and learn about them.

If you find any difficulty for the installation of these Python modules, come and talk to me.



The fundamental package for scientific computing with Python

NumPy 1.22.0 released

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

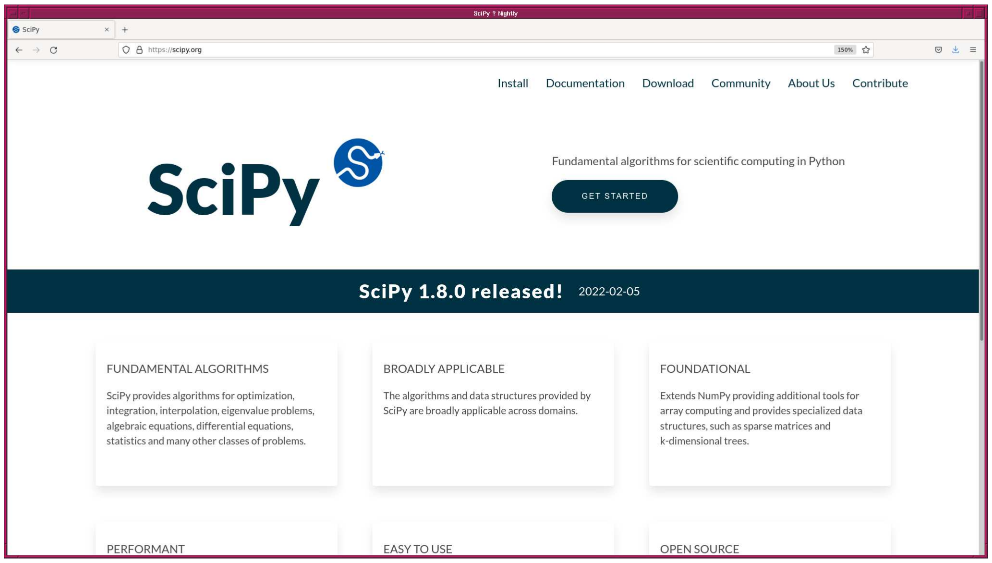
NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

EASY TO USE

OPEN SOURCE

Figure 7: The official website of Numpy.



Fundamental algorithms for scientific computing in Python

SciPy 1.8.0 released! 2022-02-05

FUNDAMENTAL ALGORITHMS

SciPy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems.

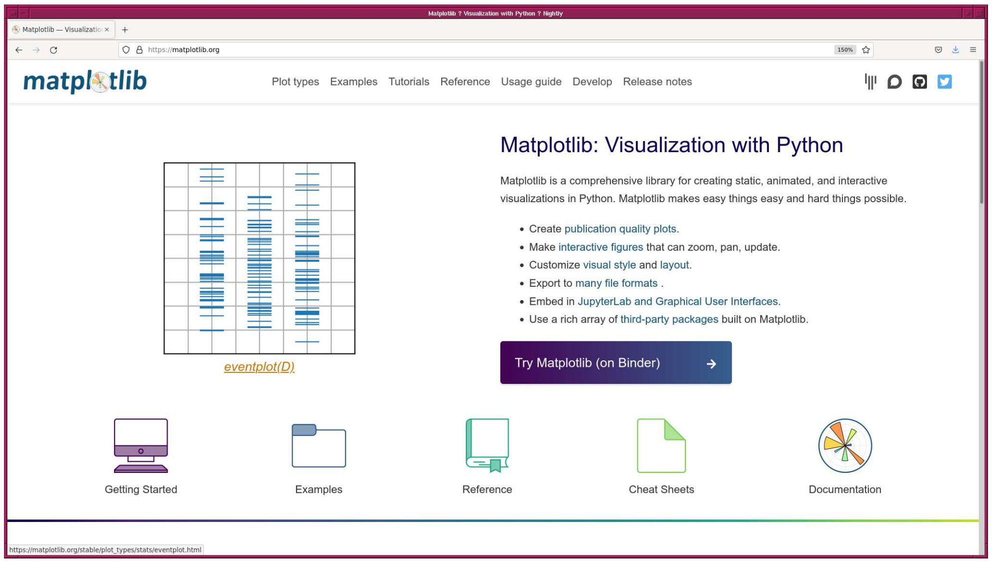
BROADLY APPLICABLE

The algorithms and data structures provided by SciPy are broadly applicable across domains.

FOUNDATIONAL

Extends NumPy providing additional tools for array computing and provides specialized data structures, such as sparse matrices and k-dimensional trees.

Figure 8: The official website of Scipy.

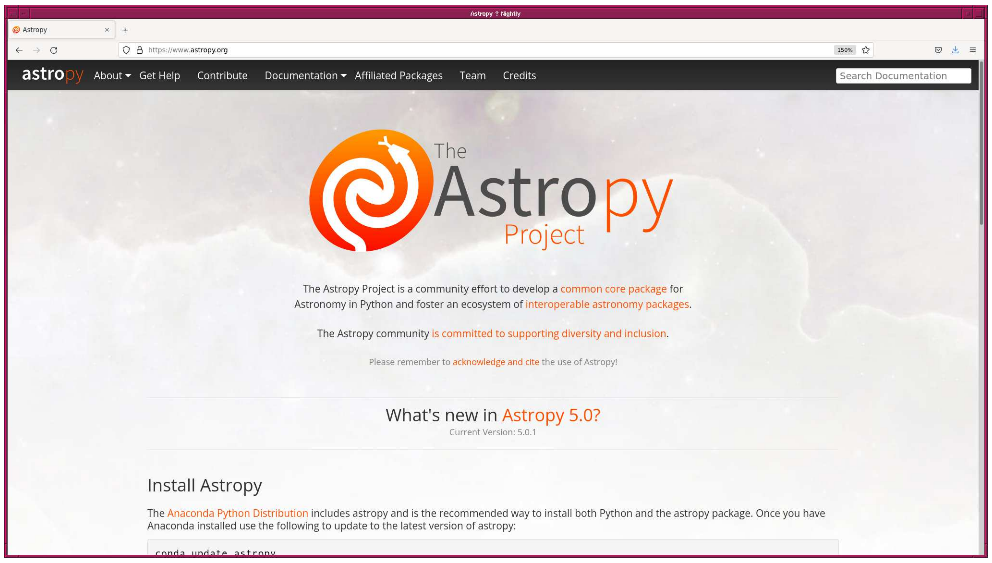


Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats .
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Figure 9: The official website of Matplotlib.



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!

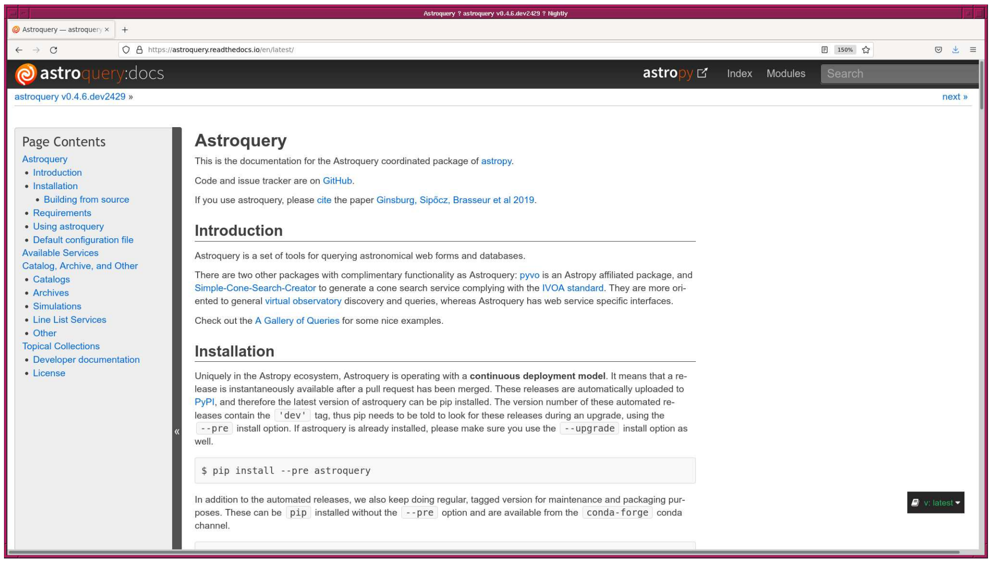
What's new in Astropy 5.0?
Current Version: 5.0.1

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 10: The official website of Astropy.


Astroquery ? astroquery v0.4.6.dev2429 ? Nightly
Astroquery — astroquery
https://astroquery.readthedocs.io/en/latest/
150%
astroquery:docs
astropy
Index
Modules
Search
astroquery v0.4.6.dev2429 »
next »
v: latest

Page Contents
- Astroquery
  - Introduction
  - Installation
    - Building from source
  - Requirements
  - Using astroquery
  - Default configuration file
- Available Services
- Catalog, Archive, and Other
  - Catalogs
  - Archives
  - Simulations
  - Line List Services
  - Other
- Topical Collections
  - Developer documentation
  - License

Astroquery

This is the documentation for the Astroquery coordinated package of astropy.

Code and issue tracker are on GitHub.

If you use astroquery, please cite the paper Ginsburg, Sipőcz, Brasseur et al 2019.

Introduction

Astroquery is a set of tools for querying astronomical web forms and databases.

There are two other packages with complimentary functionality as Astroquery: pyvo is an Astropy affiliated package, and Simple-Cone-Search-Creator to generate a cone search service complying with the IVOA standard. They are more oriented to general virtual observatory discovery and queries, whereas Astroquery has web service specific interfaces.

Check out the A Gallery of Queries for some nice examples.

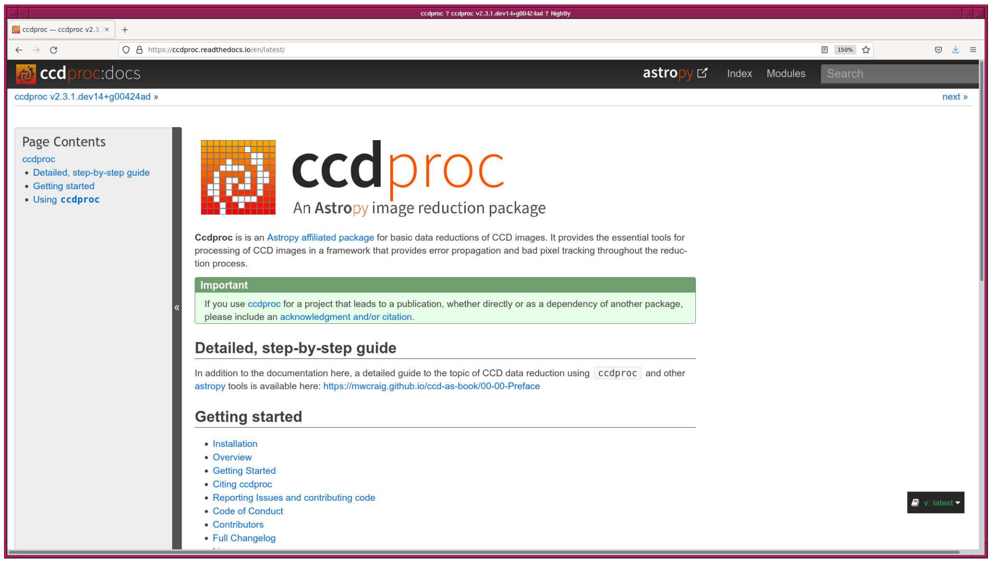
Installation

Uniquely in the Astropy ecosystem, Astroquery is operating with a **continuous deployment model**. It means that a release is instantaneously available after a pull request has been merged. These releases are automatically uploaded to PyPI, and therefore the latest version of astroquery can be pip installed. The version number of these automated releases contain the `'dev'` tag, thus pip needs to be told to look for these releases during an upgrade, using the `--pre` install option. If astroquery is already installed, please make sure you use the `--upgrade` install option as well.

```
$ pip install --pre astroquery
```

In addition to the automated releases, we also keep doing regular, tagged version for maintenance and packaging purposes. These can be `pip` installed without the `--pre` option and are available from the `conda-forge` conda channel.

Figure 11: The official website of Astroquery.


ccdproc ? ccdproc v2.3.1.dev14+g00424ad ? Nightly
ccdproc — ccdproc v2.3
https://ccdproc.readthedocs.io/en/latest/
150%
ccdproc:docs
astropy
Index
Modules
Search
ccdproc v2.3.1.dev14+g00424ad »
next »
v: latest

Page Contents

- ccdproc
  - Detailed, step-by-step guide
  - Getting started
  - Using `ccdproc`

ccdproc
An Astropy image reduction package

Ccdproc is is an Astropy affiliated package for basic data reductions of CCD images. It provides the essential tools for processing of CCD images in a framework that provides error propagation and bad pixel tracking throughout the reduction process.

Important

If you use ccdproc for a project that leads to a publication, whether directly or as a dependency of another package, please include an acknowledgment and/or citation.

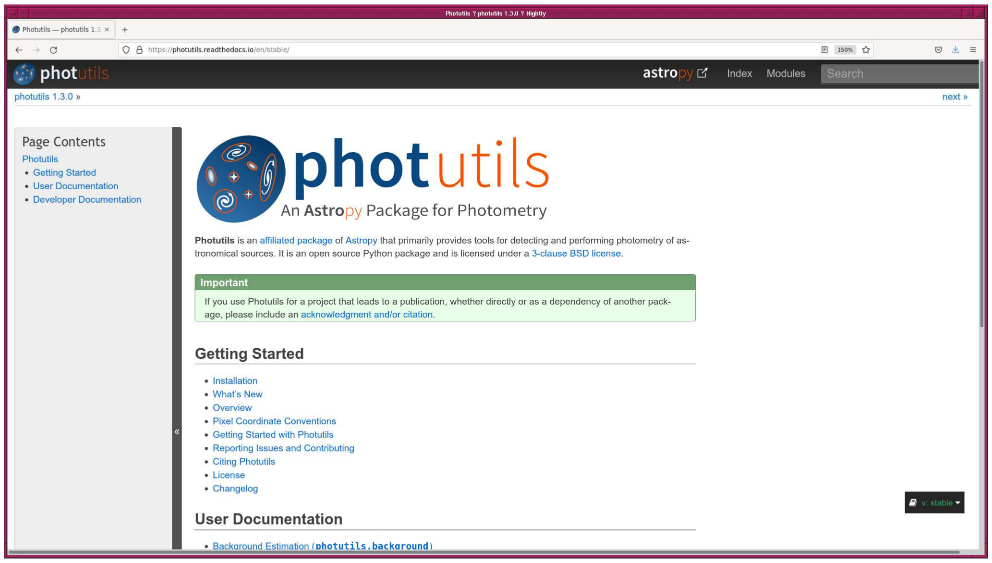
Detailed, step-by-step guide

In addition to the documentation here, a detailed guide to the topic of CCD data reduction using `ccdproc` and other astropy tools is available here: https://mwcraig.github.io/ccd-as-book/00-00-Preface

Getting started

- Installation
- Overview
- Getting Started
- Citing ccdproc
- Reporting Issues and contributing code
- Code of Conduct
- Contributors
- Full Changelog

Figure 12: The official website of ccdproc.


Photutils ? photutils 1.3.0 ? Nightly
Photutils — photutils 1.3
https://photutils.readthedocs.io/en/stable/
150%
photutils
astropy
Index
Modules
Search
photutils 1.3.0 »
next »
v: stable

Page Contents

- Photutils
  - Getting Started
  - User Documentation
  - Developer Documentation

photutils
An Astropy Package for Photometry

Photutils is an affiliated package of Astropy that primarily provides tools for detecting and performing photometry of astronomical sources. It is an open source Python package and is licensed under a 3-clause BSD license.

Important

If you use Photutils for a project that leads to a publication, whether directly or as a dependency of another package, please include an acknowledgment and/or citation.

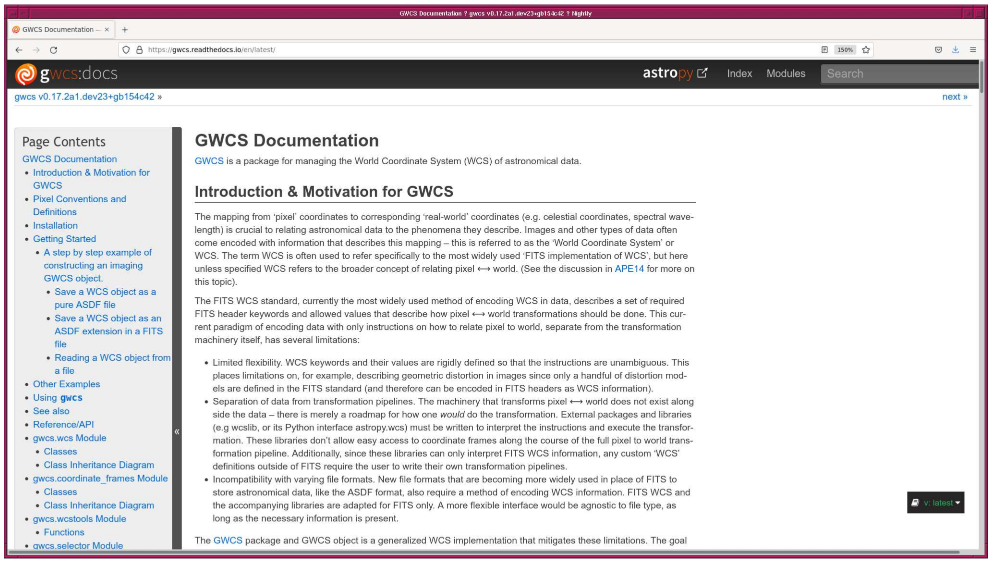
Getting Started

- Installation
- What's New
- Overview
- Pixel Coordinate Conventions
- Getting Started with Photutils
- Reporting Issues and Contributing
- Citing Photutils
- License
- Changelog

User Documentation

- Background Estimation (`photutils.background`)

Figure 13: The official website of Photutils.


GWCS Documentation ? gwcs v0.17.2a1.dev23+gb154c42 ? Nightly
GWCS Documentation —
https://gwcs.readthedocs.io/en/latest/
150%
gwcs:docs
astropy
Index
Modules
Search
gwcs v0.17.2a1.dev23+gb154c42 »
next »
v: latest

Page Contents

- GWCS Documentation
  - Introduction & Motivation for GWCS
  - Pixel Conventions and Definitions
  - Installation
  - Getting Started
    - A step by step example of constructing an imaging GWCS object.
      - Save a WCS object as a pure ASDF file
      - Save a WCS object as an ASDF extension in a FITS file
      - Reading a WCS object from a file
  - Other Examples
  - Using **gwcs**
  - See also
  - Reference/API
  - gwcs.wcs Module
    - Classes
    - Class Inheritance Diagram
  - gwcs.coordinate_frames Module
    - Classes
    - Class Inheritance Diagram
  - gwcs.wcstools Module
    - Functions
  - gwcs.selector Module

GWCS Documentation

GWCS is a package for managing the World Coordinate System (WCS) of astronomical data.

Introduction & Motivation for GWCS

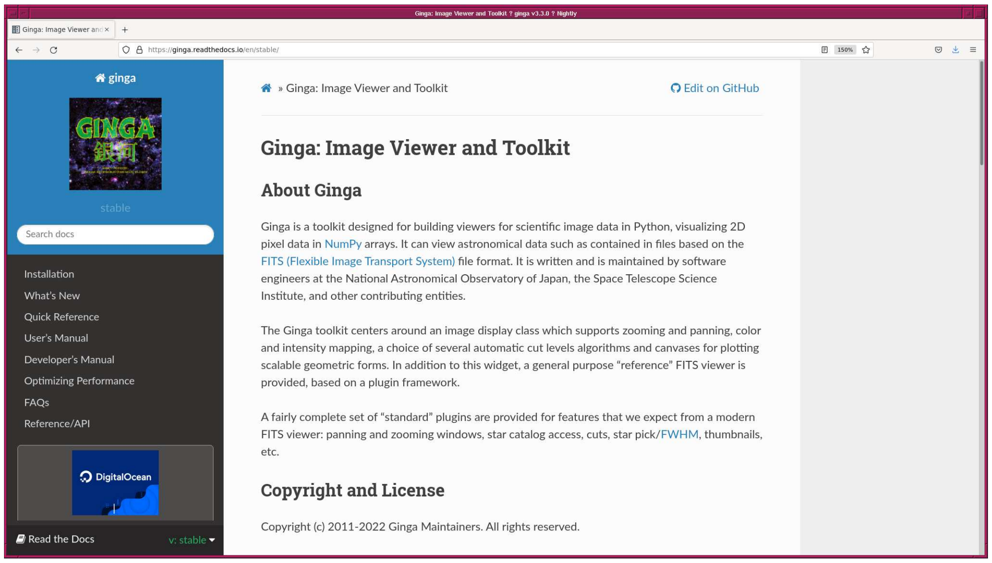
The mapping from 'pixel' coordinates to corresponding 'real-world' coordinates (e.g. celestial coordinates, spectral wavelength) is crucial to relating astronomical data to the phenomena they describe. Images and other types of data often come encoded with information that describes this mapping – this is referred to as the 'World Coordinate System' or WCS. The term WCS is often used to refer specifically to the most widely used 'FITS implementation of WCS', but here unless specified WCS refers to the broader concept of relating pixel ⟷ world. (See the discussion in APE14 for more on this topic).

The FITS WCS standard, currently the most widely used method of encoding WCS in data, describes a set of required FITS header keywords and allowed values that describe how pixel ⟷ world transformations should be done. This current paradigm of encoding data with only instructions on how to relate pixel to world, separate from the transformation machinery itself, has several limitations:

- Limited flexibility. WCS keywords and their values are rigidly defined so that the instructions are unambiguous. This places limitations on, for example, describing geometric distortion in images since only a handful of distortion models are defined in the FITS standard (and therefore can be encoded in FITS headers as WCS information).
- Separation of data from transformation pipelines. The machinery that transforms pixel ⟷ world does not exist along side the data – there is merely a roadmap for how one *would* do the transformation. External packages and libraries (e.g wcslib, or its Python interface astropy.wcs) must be written to interpret the instructions and execute the transformation. These libraries don't allow easy access to coordinate frames along the course of the full pixel to world transformation pipeline. Additionally, since these libraries can only interpret FITS WCS information, any custom 'WCS' definitions outside of FITS require the user to write their own transformation pipelines.
- Incompatibility with varying file formats. New file formats that are becoming more widely used in place of FITS to store astronomical data, like the ASDF format, also require a method of encoding WCS information. FITS WCS and the accompanying libraries are adapted for FITS only. A more flexible interface would be agnostic to file type, as long as the necessary information is present.

The GWCS package and GWCS object is a generalized WCS implementation that mitigates these limitations. The goal

Figure 14: The official website of gwcs.



Ginga: Image Viewer and Toolkit

About Ginga

Ginga is a toolkit designed for building viewers for scientific image data in Python, visualizing 2D pixel data in NumPy arrays. It can view astronomical data such as contained in files based on the FITS (Flexible Image Transport System) file format. It is written and is maintained by software engineers at the National Astronomical Observatory of Japan, the Space Telescope Science Institute, and other contributing entities.

The Ginga toolkit centers around an image display class which supports zooming and panning, color and intensity mapping, a choice of several automatic cut levels algorithms and canvases for plotting scalable geometric forms. In addition to this widget, a general purpose "reference" FITS viewer is provided, based on a plugin framework.

A fairly complete set of "standard" plugins are provided for features that we expect from a modern FITS viewer: panning and zooming windows, star catalog access, cuts, star pick/FWHM, thumbnails, etc.

Copyright and License

Copyright (c) 2011-2022 Ginga Maintainers. All rights reserved.

Figure 15: The official website of Ginga.

5 Playing with Unix commands

If you are not familiar with Unix commands, download the course material for the session #01 of "New Students' Camp 2021". (file name: `camp2021_s01e.pdf`)

6 Making a directory to store files for this session

Make a directory to store files for this session. Here is an example. Remember to replace "`/SOMEWHERE/IN/THE/DISK/advobs_2`
with the path name you choose.

```
% mkdir -p /SOMEWHERE/IN/THE/DISK/advobs_202202/s01
```

Then, move to the directory you have made.

```
% cd /SOMEWHERE/IN/THE/DISK/advobs_202202/s01
```

7 Making and executing a Python script

Now, we start to write and run Python scripts. Here is a simple example of making and executing a Python script.

7.1 Finding your Python executable

First of all, you need to know where the Python executable is located.

If you are using Python 3.9, then try following command to find the location of Python executable. Here is an example on NetBSD operating system.

```
% which python3.9
/usr/pkg/bin/python3.9
```

The command "`which`" successfully finds the location of Python executable. We now know that Python executable is located at "`/usr/pkg/bin/python3.9`" on NetBSD operating system.

Here is an example on FreeBSD operating system.

```
% which python3.9
/usr/local/bin/python3.9
```

On FreeBSD operating system, Python 3.9 executable is located at "`/usr/local/bin/python3.9`".

Here is one more example on Debian GNU/Linux.

```
% which python3.9
/usr/bin/python3.9
```

On Debian GNU/Linux, Python 3.9 executable is located at "`/usr/bin/python3.9`".

If you are using Python 3.8, then try following.

```
% which python3.8
/usr/pkg/bin/python3.8
```

If you do not have an executable of given name, then you see following message.

```
% which python3.7
python3.7: Command not found.
```

On some operating system, there may be a file named "`python3`".

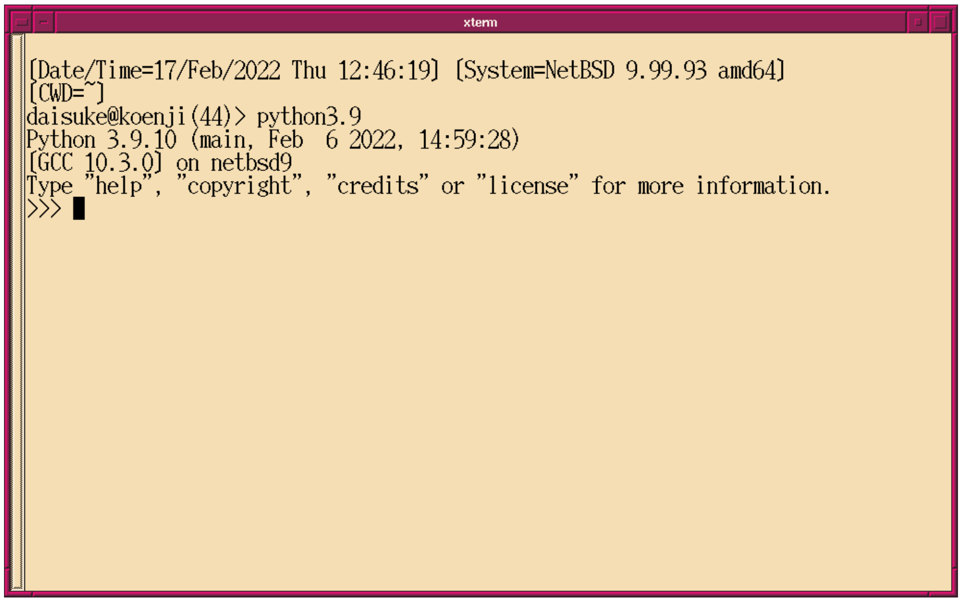
```
% which python3
/usr/bin/python3
```

The name and location of Python executable depends on the operating system. It may be `/usr/bin/python3` or `/usr/local/bin/python3.9`. Make sure to know where do you find Python executable on your computer.

7.2 Starting Python in interactive mode

Once you find the location of Python executable, start Python in interactive mode. Python interpreter is started by executing Python executable on a terminal emulator. Try following if your Python executable is `python3.9`. (Fig. 16)

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

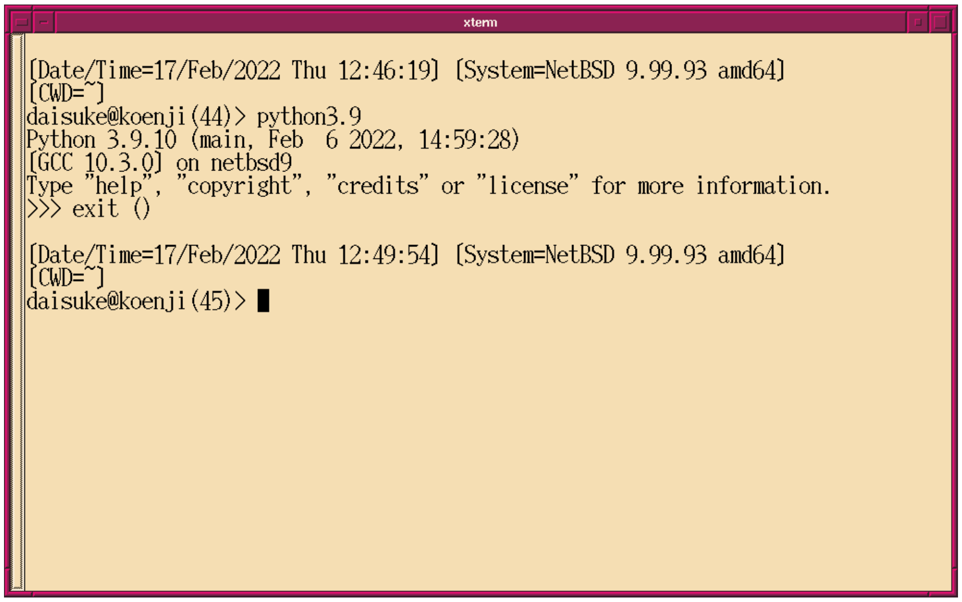


```
[Date/Time=17/Feb/2022 Thu 12:46:19] [System=NetBSD 9.99.93 amd64]
[CWD=~]
daisuke@koenji(44)> python3.9
Python 3.9.10 (main, Feb  6 2022, 14:59:28)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 16: Python in interactive mode.

We also need to know how to close Python interactive session. To leave from Python interactive mode, type `exit ()` or Ctrl-D. (Fig. 17)

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



```
[Date/Time=17/Feb/2022 Thu 12:46:19] [System=NetBSD 9.99.93 amd64]
[CWD=~]
daisuke@koenji(44)> python3.9
Python 3.9.10 (main, Feb  6 2022, 14:59:28)
[GCC 10.3.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> exit ()

[Date/Time=17/Feb/2022 Thu 12:49:54] [System=NetBSD 9.99.93 amd64]
[CWD=~]
daisuke@koenji(45)>
```

Figure 17: A way to close Python interactive session.

7.3 Making a simple Python script

Use your favourite text editor to make a simple Python script of following content. (Fig. 18)

Python Code 1: advobs202202_s01_01.py

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

print ("This is a very simple Python 3 script.")
```

Here I clarify differences between shadowed square box and round box. Shadowed square box shows command(s) you type on a terminal emulator. Round box shows Python script you type on a text editor.

Save this script as the file "`advobs202202_s01_01.py`".

```
% ls -l
total 1
-rw-r--r--  1 daisuke  taiwan  75 Feb 17 12:57 advobs202202_s01_01.py
% cat advobs202202_s01_01.py
#!/usr/pkg/bin/python3.9

print ("This is a very simple Python 3 script.")
```

If you are not familiar to the usage of the text editor Emacs, download and read the course material for the session #01 of "New Students' Camp 2021" (file name: `camp2021_s01e.pdf`).

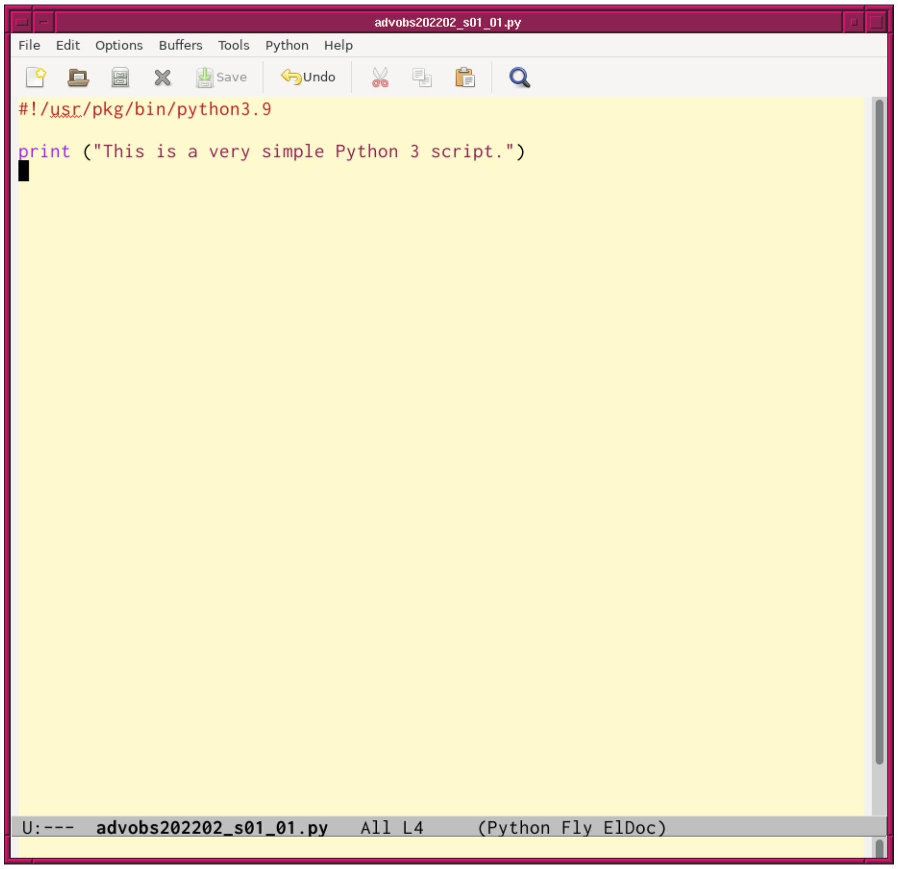
7.4 Executing a Python script

For executing a Python script, check the current file mode of the Python script first.

```
% ls -lF
total 1
-rw-r--r--  1 daisuke  taiwan  75 Feb 17 12:57 advobs202202_s01_01.py
```

For this file, the owner of the file has read and write permissions. Other users have read permission only.

Now, we try to add execute permission to this file by using the command "`chmod`". Try following.



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

print ("This is a very simple Python 3 script.")
```

```
U:---  advobs202202_s01_01.py   All L4     (Python Fly ElDoc)
```

Figure 18: A sample Python script typed on the text editor Emacs.

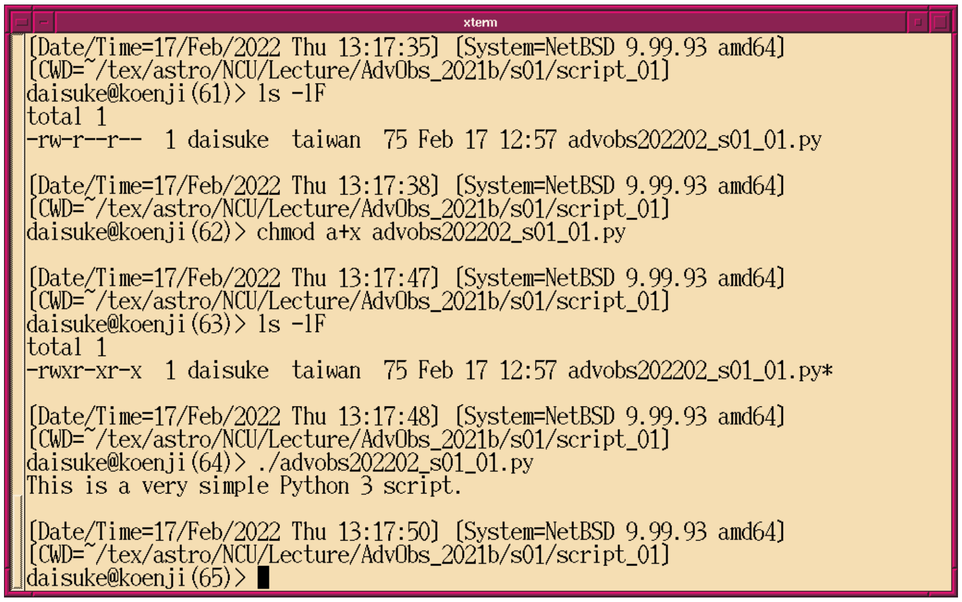
```
% chmod a+x advobs202202_s01_01.py
```

Check the file mode again.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  75 Feb 17 12:57 advobs202202_s01_01.py*
```

Now, the file can be executed by everyone.
Execute the Python script. (Fig. 19)

```
% ./advobs202202_s01_01.py
This is a very simple Python 3 script.
```



```
[Date/Time=17/Feb/2022 Thu 13:17:35] [System=NetBSD 9.99.93 amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvObs_2021b/s01/script_01]
daisuke@koenji(61)> ls -lF
total 1
-rw-r--r--  1 daisuke  taiwan  75 Feb 17 12:57 advobs202202_s01_01.py

[Date/Time=17/Feb/2022 Thu 13:17:38] [System=NetBSD 9.99.93 amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvObs_2021b/s01/script_01]
daisuke@koenji(62)> chmod a+x advobs202202_s01_01.py

[Date/Time=17/Feb/2022 Thu 13:17:47] [System=NetBSD 9.99.93 amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvObs_2021b/s01/script_01]
daisuke@koenji(63)> ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  75 Feb 17 12:57 advobs202202_s01_01.py*

[Date/Time=17/Feb/2022 Thu 13:17:48] [System=NetBSD 9.99.93 amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvObs_2021b/s01/script_01]
daisuke@koenji(64)> ./advobs202202_s01_01.py
This is a very simple Python 3 script.

[Date/Time=17/Feb/2022 Thu 13:17:50] [System=NetBSD 9.99.93 amd64]
[CWD=~/tex/astro/NCU/Lecture/AdvObs_2021b/s01/script_01]
daisuke@koenji(65)> 
```

Figure 19: The way to execute a Python script on a terminal emulator.

8 Using argparse module

To handle command-line arguments, `argparse` module is extremely helpful. Try `argparse` module.
The official document about `argparse` module can be found at following web page.

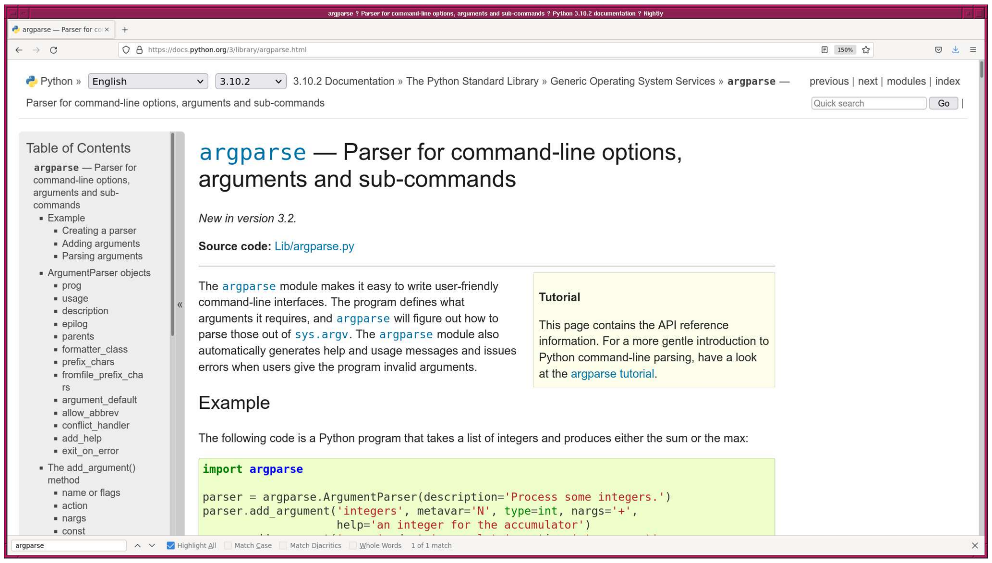
- argparse: `https://docs.python.org/3/library/argparse.html` (Fig. 20)

8.1 Adding two integers

Make a Python script to add two integers and return the result. Here is an example.

Python Code 2: advobs202202_s01_02.py

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

# importing argparse module
import argparse
```


argparse ? Parser for command-line options, arguments and sub-commands ? Python 3.10.2 documentation ? Nightly

https://docs.python.org/3/library/argparse.html

Python » English 3.10.2 3.10.2 Documentation » The Python Standard Library » Generic Operating System Services » **argparse** — Parser for command-line options, arguments and sub-commands

Table of Contents

- **argparse** — Parser for command-line options, arguments and sub-commands
  - Example
    - Creating a parser
    - Adding arguments
    - Parsing arguments
  - ArgumentParser objects
    - prog
    - usage
    - description
    - epilog
    - parents
    - formatter_class
    - prefix_chars
    - fromfile_prefix_chars
    - argument_default
    - allow_abbrev
    - conflict_handler
    - add_help
    - exit_on_error
  - The add_argument() method
    - name or flags
    - action
    - nargs
    - const

argparse — Parser for command-line options, arguments and sub-commands

New in version 3.2.

Source code: Lib/argparse.py

The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

Tutorial

This page contains the API reference information. For a more gentle introduction to Python command-line parsing, have a look at the argparse tutorial.

Example

The following code is a Python program that takes a list of integers and produces either the sum or the max:

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
```

Figure 20: The official document of argparse module.

```
# construction of parser object
parser = argparse.ArgumentParser (description='Adding two numbers')

# adding arguments
parser.add_argument ('-a', type=int, default=1, help='number 1')
parser.add_argument ('-b', type=int, default=1, help='number 2')

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

# two numbers
a = args.a
b = args.b

# adding two numbers
c = a + b

# printing result
print (a, '+', b, '=', c)
```

Ask question if any line of above script is not clear to you.
Execute the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rw-r--r--  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py
% chmod a+x advobs202202_s01_02.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
% ./advobs202202_s01_02.py -h
usage: advobs202202_s01_02.py [-h] [-a A] [-b B]

Adding two numbers
```

```
optional arguments:
  -h, --help  show this help message and exit
  -a A        number 1
  -b B        number 2

% ./advobs202202_s01_02.py -a 2 -b 3
2 + 3 = 5
% ./advobs202202_s01_02.py -a 5 -b 8
5 + 8 = 13
% ./advobs202202_s01_02.py -a 1024 -b 4096
1024 + 4096 = 5120
% ./advobs202202_s01_02.py -a 1 -b -3
1 + -3 = -2
% ./advobs202202_s01_02.py
1 + 1 = 2
```

8.2 Adding two floating point numbers

If you specify a floating point number for `advobs202202_s01_02.py`, you get an error.

```
% ./advobs202202_s01_02.py -a 1.2 -b 3
usage: advobs202202_s01_02.py [-h] [-a A] [-b B]
advobs202202_s01_02.py: error: argument -a: invalid int value: '1.2'
% ./advobs202202_s01_02.py -a 1 -b 2.3
usage: advobs202202_s01_02.py [-h] [-a A] [-b B]
advobs202202_s01_02.py: error: argument -b: invalid int value: '2.3'
% ./advobs202202_s01_02.py -a 1.2 -b 3.4
usage: advobs202202_s01_02.py [-h] [-a A] [-b B]
advobs202202_s01_02.py: error: argument -a: invalid int value: '1.2'
```

Make a new Python script to add two floating point numbers, try following.

Python Code 3: advobs202202_s01_03.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'Adding two floating point numbers'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-a', type=float, default=1.0, help='number 1')
parser.add_argument ('-b', type=float, default=1.0, help='number 2')

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

# two numbers
a = args.a
b = args.b

# calculation to add two numbers
c = a + b
```

```
# printing result
print ("%f + %f = %f" % (a, b, c) )
```

Execute the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rw-r--r--  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py
% chmod a+x advobs202202_s01_03.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py*
% ./advobs202202_s01_03.py -h
usage: advobs202202_s01_03.py [-h] [-a A] [-b B]

Adding two floating point numbers

optional arguments:
  -h, --help  show this help message and exit
  -a A        number 1
  -b B        number 2

% ./advobs202202_s01_03.py -a 1.2 -b 3.4
1.200000 + 3.400000 = 4.600000
% ./advobs202202_s01_03.py -a 123.456 -b 234.567
123.456000 + 234.567000 = 358.023000
% ./advobs202202_s01_03.py -a 0.001 -b 0.0001
0.001000 + 0.000100 = 0.001100
% ./advobs202202_s01_03.py -a 1.2 -b -3.4
1.200000 + -3.400000 = -2.200000
```

8.3 Arithmetic operations

Make a Python script to carry out arbitrary arithmetic operation of two numbers.

Python Code 4: advobs202202_s01_04.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'Arithmetic operation of two numbers'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('n1', type=float, default=1.0, help='number 1')
parser.add_argument ('operator', choices=['+', '-', 'x', '/'], \
                     default='+', help='operator (one of [+, -, x, /])')
parser.add_argument ('n2', type=float, default=1.0, help='number 2')

# command-line argument analysis
```

```
args = parser.parse_args()

# two numbers
n1       = args.n1
operator = args.operator
n2       = args.n2

# calculation
if (operator == '+'):
    n3 = n1 + n2
elif (operator == '-'):
    n3 = n1 - n2
elif (operator == 'x'):
    n3 = n1 * n2
elif (operator == '/'):
    n3 = n1 / n2

# printing result
print ("%f %s %f = %f" % (n1, operator, n2, n3) )
```

Try the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py*
-rw-r--r--  1 daisuke  taiwan  932 Feb 17 13:50 advobs202202_s01_04.py
% chmod a+x advobs202202_s01_04.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan  932 Feb 17 13:50 advobs202202_s01_04.py*
% ./advobs202202_s01_04.py -h
usage: advobs202202_s01_04.py [-h] n1 {+,-,x,/} n2

Arithmetic operation of two numbers

positional arguments:
  n1          number 1
  {+,-,x,/}   operator (one of [+, -, x, /])
  n2          number 2

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

% ./advobs202202_s01_04.py 12.3 + 45.6
12.300000 + 45.600000 = 57.900000
% ./advobs202202_s01_04.py 123.4 - 56.7
123.400000 - 56.700000 = 66.700000
% ./advobs202202_s01_04.py 1.2 x 3.4
1.200000 x 3.400000 = 4.080000
% ./advobs202202_s01_04.py 9.87 / 6.54
9.870000 / 6.540000 = 1.509174
% ./advobs202202_s01_04.py 1.23 x -4.56
1.230000 x -4.560000 = -5.608800
```

8.4 Calculation of a mean

Make a Python script to calculate a mean for a given set of numbers.

Python Code 5: advobs202202_s01_05.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'Calculation of a mean'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('numbers', type=float, nargs='+', help='a set of numbers')

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

# numbers
numbers = args.numbers

# number of data
n = len (numbers)

# initialisation of a variable 'total'
total = 0.0

# adding numbers using "for"
for number in numbers:
    total += number

# calculation of a mean
mean = total / n

# printing result
print ("input data set =", numbers)
print ("number of data = %d" % n)
print ("mean value     = %f / %d = %f" % (total, n, mean) )
```

Execute the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan  932 Feb 17 13:50 advobs202202_s01_04.py*
-rw-r--r--  1 daisuke  taiwan  784 Feb 17 14:04 advobs202202_s01_05.py
% chmod a+x advobs202202_s01_05.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan  126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan  559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan  604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan  932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan  784 Feb 17 14:04 advobs202202_s01_05.py*
```

```
% ./advobs202202_s01_05.py -h
usage: advobs202202_s01_05.py [-h] numbers [numbers ...]

Calculation of a mean

positional arguments:
  numbers     a set of numbers

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

% ./advobs202202_s01_05.py 1 2 3 4 5 6 7 8 9 10
input data set = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
number of data = 10
mean value     = 55.000000 / 10 = 5.500000
% ./advobs202202_s01_05.py 10.0 9.0 11.0 9.5 10.5 8.5 11.5 8.0 12.0 7.5 12.5
input data set = [10.0, 9.0, 11.0, 9.5, 10.5, 8.5, 11.5, 8.0, 12.0, 7.5, 12.5]
number of data = 11
mean value     = 110.000000 / 11 = 10.000000
```

Modify above script to allow the calculation of a median.

Python Code 6: advobs202202_s01_06.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'Calculation of average'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-a', choices=['mean', 'median'], \
                     default='mean', help='algorithm (mean or median)')
parser.add_argument ('numbers', type=float, nargs='+', help='a set of numbers')

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

# numbers
algorithm = args.a
numbers   = args.numbers

# number of data
n = len (numbers)

#
# calculation of average
#

# mean calculation
if (algorithm == 'mean'):
    # initialisation of a variable 'total'
    total = 0.0
    # adding numbers
    for number in numbers:
        total += number
    # calculation of a mean
    mean = total / n
# median calculation
elif (algorithm == 'median'):
    # sorting data
    sorted_numbers = sorted (numbers)
    # calculation of a median
    if (n % 2 == 0):
        # if number of data is even number,
        # adding two numbers in the middle, then divide by two
        median = ( sorted_numbers[int (n / 2) - 1] \
                   + sorted_numbers[int (n / 2)] ) / 2.0
    elif (n % 2 == 1):
        # if number of data is odd number,
        # simply taking the number in the middle
        median = sorted_numbers[int (n / 2)]

# printing result
print ("input data     =", numbers)
print ("number of data = %d" % n)
if (algorithm == 'mean'):
    print ("mean           = %f / %d = %f" % (total, n, mean) )
elif (algorithm == 'median'):
    print ("median         = %f" % median)
```

Try the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rw-r--r--  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py
% chmod a+x advobs202202_s01_06.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
% ./advobs202202_s01_06.py -h
usage: advobs202202_s01_06.py [-h] [-a {mean,median}] numbers [numbers ...]

Calculation of average

positional arguments:
  numbers            a set of numbers

optional arguments:
  -h, --help         show this help message and exit
  -a {mean,median}   algorithm (mean or median)

% ./advobs202202_s01_06.py -a mean 100 101 99 102 98 103 97 104 96
input data     = [100.0, 101.0, 99.0, 102.0, 98.0, 103.0, 97.0, 104.0, 96.0]
```

```
number of data = 9
mean           = 900.000000 / 9 = 100.000000
% ./advobs202202_s01_06.py -a median 1 2 3 4 5
input data     = [1.0, 2.0, 3.0, 4.0, 5.0]
number of data = 5
median         = 3.000000
% ./advobs202202_s01_06.py -a median 1 2 3 4 5 6
input data     = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
number of data = 6
median         = 3.500000
% ./advobs202202_s01_06.py -a mean 10 10.1 9.9 10.2 9.8 10.3 9.7 50
input data     = [10.0, 10.1, 9.9, 10.2, 9.8, 10.3, 9.7, 50.0]
number of data = 8
mean           = 120.000000 / 8 = 15.000000
% ./advobs202202_s01_06.py -a median 10 10.1 9.9 10.2 9.8 10.3 9.7 50
input data     = [10.0, 10.1, 9.9, 10.2, 9.8, 10.3, 9.7, 50.0]
number of data = 8
median         = 10.050000
```

9 Reading and writing files

We do some practices for reading and writing files.

9.1 Opening a file and reading it

First, download the data file. Use the command "`curl`" to download the file.

```
% curl -k -o fruits.data \
? https://s3b.astro.ncu.edu.tw/advobs_202202/data/fruits.data
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   246  100   246    0     0   8048      0 --:--:-- --:--:-- --:--:--  8200
% ls -lF fruits.data
-rw-r--r--  1 daisuke  taiwan  246 Feb 17 14:56 fruits.data
% cat fruits.data
#
# data format
#
#   first column:  fruit name
#   second column: unit price
#   third column:  quantity
#
apple         100    5
orange         20   10
strawberry    200    2
grape         150    3
banana         10   10
watermelon     80    4
```

If you prefer to use web browser to download the file, start your favourite web browser and type following address on the address bar of the web browser. Then, save the file on your computer.

- location of data file: `https://s3b.astro.ncu.edu.tw/advobs_202202/data/fruits.data`

Make a Python script to open a file and read the contents of it, and calculate the total price of fruits.

Python Code 7: advobs202202_s01_07.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'Reading a file'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-i', '--input', default='', \
                     help='input file name')

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

# parameters
file_input = args.input

# initialisation of the parameter "total_price"
total_price = 0

# printing header
print ("%s" % '-' * 46)
print ("%-10s  %10s  %10s  %10s" \
       % ("fruit name", "unit price", "quantity", "sub-total") )
print ("%s" % '=' * 46)

# opening input file with read mode
with open (file_input, 'r') as fh:
    # reading the file line-by-line
    for line in fh:
        # removing line feed at the end of the line
        line = line.strip ()
        # if the line starts with "#", then skip
        if (line[0] == '#'):
            continue
        # splitting data
        (fruit_name, unit_price_str, quantity_str) = line.split ()
        # conversion from string into integer
        unit_price = int (unit_price_str)
        quantity   = int (quantity_str)
        # calculation of sub-total
        subtotal = unit_price * quantity
        # adding prices
        total_price += subtotal
        # printing sub-total
        print ("%-10s  %10d  %10d  %10d" \
               % (fruit_name, unit_price, quantity, subtotal) )

# printing result
print ("%s" % '-' * 46)
print ("")
print ("Total price: %d" % total_price)
```

Execute the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rw-r--r--  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
% chmod a+x advobs202202_s01_07.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
% ./advobs202202_s01_07.py -h
usage: advobs202202_s01_07.py [-h] [-i INPUT]

Reading a file

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input file name

% ./advobs202202_s01_07.py -i fruits.data
----------------------------------------------
fruit name  unit price    quantity   sub-total
==============================================
apple              100           5         500
orange              20          10         200
strawberry         200           2         400
grape              150           3         450
banana              10          10         100
watermelon          80           4         320
----------------------------------------------

Total price: 1970
```

The total price of fruits is 1970.

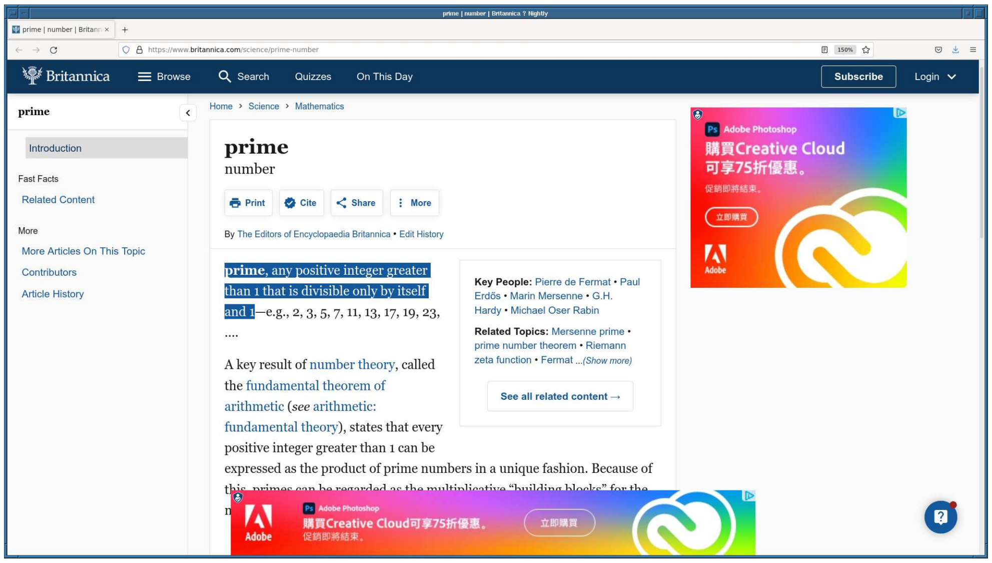
9.2 Finding prime numbers and writing those numbers to a file

Make a Python script to find prime numbers and write those numbers into a file.
If you are not sure about prime numbers, check out following web page.

- `https://www.britannica.com/science/prime-number` (Fig. 21)

Python Code 8: advobs202202_s01_08.py

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


prime | number | Britannica ? Nightly
prime | number | Britann ×
https://www.britannica.com/science/prime-number
150%
Britannica
Browse
Search
Quizzes
On This Day
Subscribe
Login
prime
Introduction
Fast Facts
Related Content
More
More Articles On This Topic
Contributors
Article History
Home › Science › Mathematics
Print
Cite
Share
More
See all related content →
Adobe Photoshop
購買Creative Cloud可享75折優惠。
促銷即將結束。
立即購買
Adobe

prime
number

By The Editors of Encyclopaedia Britannica • Edit History

prime, any positive integer greater than 1 that is divisible only by itself and 1—e.g., 2, 3, 5, 7, 11, 13, 17, 19, 23, ...

A key result of number theory, called the fundamental theorem of arithmetic (*see* arithmetic: fundamental theory), states that every positive integer greater than 1 can be expressed as the product of prime numbers in a unique fashion. Because of this, primes can be regarded as the multiplicative "building blocks" for the

Key People: Pierre de Fermat • Paul Erdős • Marin Mersenne • G.H. Hardy • Michael Oser Rabin

Related Topics: Mersenne prime • prime number theorem • Riemann zeta function • Fermat ...*(Show more)*

Figure 21: The description about prime numbers on Encyclopaedia Britannica.

```
# importing argparse module
import argparse

# construction of parser object
desc = 'Finding prime numbers'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('-o', '--output', default='primenumbers.data', \
                     help='output file name')
parser.add_argument ('-s', '--start', type=int, default=2, \
                     help='number to start')
parser.add_argument ('-e', '--end', type=int, default=100, \
                     help='number to end')

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

# parameters
file_output = args.output
n_start     = args.start
n_end       = args.end

# initialisation of a list to store results
primenumbers = []

# checking numbers from n_start to n_end
for i in range (n_start, n_end + 1):
    # resetting the parameter "count"
    count = 0
    # examining if the number is divisible by numbers between 2 and (i-1)
    for j in range (2, i):
        # if the number is divisible, then adding 1 to "count"
        if (i % j == 0):
            count += 1
            # if count is >= 1, the number is not a prime number
            break
    # if the number is a prime number, add it to the list "primenumbers"
    if (count == 0):
        primenumbers.append (i)

# writing result into a file
with open (file_output, 'w') as fh:
    # for each prime number
    for i in primenumbers:
        # writing the number to the file
        fh.write ("%d\n" % i)
```

Try the script.

```
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rw-r--r--  1 daisuke  taiwan  1539 Feb 17 15:23 advobs202202_s01_08.py
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
% chmod a+x advobs202202_s01_08.py
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rwxr-xr-x  1 daisuke  taiwan  1539 Feb 17 15:23 advobs202202_s01_08.py*
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
% ./advobs202202_s01_08.py -h
usage: advobs202202_s01_08.py [-h] [-o OUTPUT] [-s START] [-e END]

Finding prime numbers

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        output file name
  -s START, --start START
                        number to start
  -e END, --end END     number to end

% ./advobs202202_s01_08.py -s 2 -e 10 -o prime_00010.data
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
```

```
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rwxr-xr-x  1 daisuke  taiwan  1539 Feb 17 15:23 advobs202202_s01_08.py*
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
-rw-r--r--  1 daisuke  taiwan     8 Feb 17 15:25 prime_00010.data
% cat prime_00010.data
2
3
5
7
% ./advobs202202_s01_08.py --start 2 --end 100 -o prime_00100.data
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rwxr-xr-x  1 daisuke  taiwan  1539 Feb 17 15:23 advobs202202_s01_08.py*
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
-rw-r--r--  1 daisuke  taiwan     8 Feb 17 15:25 prime_00010.data
-rw-r--r--  1 daisuke  taiwan    71 Feb 17 15:26 prime_00100.data
% tail prime_00100.data
53
59
61
67
71
73
79
83
89
97
% ./advobs202202_s01_08.py --start 2 --end 10000 -o prime_10000.data
% ls -lF
total 1
-rwxr-xr-x  1 daisuke  taiwan   126 Feb 17 13:32 advobs202202_s01_01.py*
-rwxr-xr-x  1 daisuke  taiwan   559 Feb 17 13:33 advobs202202_s01_02.py*
-rwxr-xr-x  1 daisuke  taiwan   604 Feb 17 13:42 advobs202202_s01_03.py*
-rwxr-xr-x  1 daisuke  taiwan   932 Feb 17 13:50 advobs202202_s01_04.py*
-rwxr-xr-x  1 daisuke  taiwan   784 Feb 17 14:04 advobs202202_s01_05.py*
-rwxr-xr-x  1 daisuke  taiwan  1663 Feb 17 14:15 advobs202202_s01_06.py*
-rwxr-xr-x  1 daisuke  taiwan  1566 Feb 17 15:18 advobs202202_s01_07.py*
-rwxr-xr-x  1 daisuke  taiwan  1539 Feb 17 15:23 advobs202202_s01_08.py*
-rw-r--r--  1 daisuke  taiwan   246 Feb 17 14:56 fruits.data
-rw-r--r--  1 daisuke  taiwan     8 Feb 17 15:25 prime_00010.data
-rw-r--r--  1 daisuke  taiwan    71 Feb 17 15:26 prime_00100.data
-rw-r--r--  1 daisuke  taiwan  5948 Feb 17 15:26 prime_10000.data
% tail prime_10000.data
9887
9901
9907
9923
9929
9931
9941
9949
9967
```

```
9973
```

10 Doing simple calculations using Python

10.1 Wavelength of a photon

Question: What wavelength photon would you need to ionize a helium atom? (ionization energy = 24.59 eV)

Python Code 9: advobs202202_s01_09.py

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

# Planck constant
h = 6.63 * 10**-34

# speed of light in vacuum
c = 3.00 * 10**8

# 1 eV in J
eV = 1.60 * 10**-19

# energy of a photon in eV
E_eV = 24.59

# calculation of energy of a photon in J
E_J = E_eV * eV

# calculation of wavelength of a photon
wavelength = h * c / E_J

# printing result
print ("energy of a photon = %f eV = %g J" % (E_eV, E_J) )
print ("wavelength = %g m = %f nm" % (wavelength, wavelength * 10**9) )
```

Execute the script.

```
% chmod a+x advobs202202_s01_09.py
% ./advobs202202_s01_09.py
energy of a photon = 24.590000 eV = 3.9344e-18 J
wavelength = 5.05541e-08 m = 50.554087 nm
```

10.2 Magnitude

Question: The monochromatic flux at the center of the V band-pass (550 nm) for a certain star is 30.0 Jy. If this star has a visual magnitude of $m_V = 5.21$, what is the monochromatic flux, in Jy, at 550 nm for a star with $m_V = 9.65$?

Python Code 10: advobs202202_s01_10.py

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

# flux of object 1
F1 = 30.0

# magnitude of object 1
m1 = 5.21
```

```
# magnitude of object 2
m2 = 9.65

# calculation of flux of object 2 using Pogson's formula
F2 = F1 * 10**(0.4 * (m1 - m2) )

# printing result
print ("m1 = %f" % m1)
print ("F1 = %f Jy" % F1)
print ("m2 = %f" % m2)
print ("F2 = F1 * 10**(0.4 * (m1 - m2) ) = %f Jy" % F2)
```

Execute the script.

```
% chmod a+x advobs202202_s01_10.py
% ./advobs202202_s01_10.py
m1 = 5.210000
F1 = 30.000000 Jy
m2 = 9.650000
F2 = F1 * 10**(0.4 * (m1 - m2) ) = 0.502483 Jy
```

11 Math module

Try `math` module.

11.1 Conversion from radian into arcmin

The diameter of the Moon is 3476 km. The mean distance from the Earth to the Moon is 384400 km. Calculate the angular diameter of the Moon in arcmin.

Python Code 11: advobs202202_s01_11.py

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

# importing math module
import math

# value of pi
pi = math.pi

# diameter of the Moon in km
diameter = 3476

# mean distance from the Earth to the Moon in km
distance = 384400

# angular diameter of the Moon in radian
a_rad = diameter / distance

# conversion from radian into degree
a_deg = a_rad / pi * 180

# conversion from degree into arcmin
a_arcmin = a_deg * 60

# printing result
print ("angular diameter of the Moon = %f arcmin" % a_arcmin)
```

Execute the script and check the result.

```
% chmod a+x advobs202202_s01_11.py
% ./advobs202202_s01_11.py
angular diameter of the Moon = 31.086389 arcmin
```

11.2 Size of Betelgeuse

Luminosity of Betelgeuse is $\sim$ 126,000 $L_\odot$, and the effective temperature of Betelgeuse is $\sim$ 3600 K. Calculate the radius of Betelgeuse.

Python Code 12: advobs202202_s01_12.py

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

# importing math module
import math

# luminosity of Betelgeuse
L_betelgeuse = 126000

# effective temperature of Betelgeuse
T_betelgeuse = 3600

# luminosity of the Sun
L_sun        = 1

# effective temperature of the Sun
T_sun        = 5800

# calculation of radius of Betelgeuse in solar radius
R_betelgeuse = math.sqrt (L_betelgeuse / L_sun) * (T_betelgeuse / T_sun)**-2

# printing result
print ("radius of Betelgeuse = %f solar radii" % R_betelgeuse)
```

Execute the script, and check the result.

```
% chmod a+x advobs202202_s01_12.py
% ./advobs202202_s01_12.py
radius of Betelgeuse = 921.374648 solar radii
```

11.3 Calculation of magnitude of an object

You have observed a target object, and measured net flux from the object is 6500 ADU/sec. You also observed a photometric standard star under the same condition with the target object. Measured net flux from the photometric standard star is 730000 ADU/sec, and the magnitude of the photometric standard star is 13.55 mag. Calculate the magnitude of the target object.

Python Code 13: advobs202202_s01_13.py

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

# importing math module
import math

# magnitude of standard star
```

```
m_std = 13.55

# flux of standard star
F_std = 730000

# flux of target object
F_obj = 6500

# calculation of magnitude of target object
m_obj = m_std - 2.5 * math.log10 (F_obj / F_std)

# printing result
print ("flux of standard star      = %f ADU/sec" % F_std)
print ("flux of target object      = %f ADU/sec" % F_obj)
print ("magnitude of standard star = %f mag" % m_std)
print ("magnitude of target object = %f mag" % m_obj)
```

Execute the script, and check the result.

```
% chmod a+x advobs202202_s01_13.py
% ./advobs202202_s01_13.py
flux of standard star      = 730000.000000 ADU/sec
flux of target object      = 6500.000000 ADU/sec
magnitude of standard star = 13.550000 mag
magnitude of target object = 18.676024 mag
```

11.4 Confirming the relation $\sin^2\theta + \cos^2\theta = 1$

Confirm the relation $\sin^2\theta + \cos^2\theta = 1$.

Python Code 14: advobs202202_s01_14.py

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

# importing math module
import math

# pi
pi = math.pi

# angles in degree: [0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180]
list_angles = range (0, 181, 15)

# processing each angle
for a_deg in list_angles:
    # conversion from deg to rad
    a_rad = a_deg / 180.0 * pi
    # calculation of sin and cos
    sin_a = math.sin (a_rad)
    cos_a = math.cos (a_rad)
    # calculation of sin^2 a + cos^2 a
    result = sin_a**2 + cos_a**2
    # printing result
    print ("sin^2 (%5.1f deg) + cos^2 (%5.1f deg) = %f + %f = %f" \
           % (a_deg, a_deg, sin_a**2, cos_a**2, result) )
```

Execute the script, and check the result.

```
% chmod a+x advobs202202_s01_14.py
% ./advobs202202_s01_14.py
sin^2 (  0.0 deg) + cos^2 (  0.0 deg) = 0.000000 + 1.000000 = 1.000000
sin^2 ( 15.0 deg) + cos^2 ( 15.0 deg) = 0.066987 + 0.933013 = 1.000000
sin^2 ( 30.0 deg) + cos^2 ( 30.0 deg) = 0.250000 + 0.750000 = 1.000000
sin^2 ( 45.0 deg) + cos^2 ( 45.0 deg) = 0.500000 + 0.500000 = 1.000000
sin^2 ( 60.0 deg) + cos^2 ( 60.0 deg) = 0.750000 + 0.250000 = 1.000000
sin^2 ( 75.0 deg) + cos^2 ( 75.0 deg) = 0.933013 + 0.066987 = 1.000000
sin^2 ( 90.0 deg) + cos^2 ( 90.0 deg) = 1.000000 + 0.000000 = 1.000000
sin^2 (105.0 deg) + cos^2 (105.0 deg) = 0.933013 + 0.066987 = 1.000000
sin^2 (120.0 deg) + cos^2 (120.0 deg) = 0.750000 + 0.250000 = 1.000000
sin^2 (135.0 deg) + cos^2 (135.0 deg) = 0.500000 + 0.500000 = 1.000000
sin^2 (150.0 deg) + cos^2 (150.0 deg) = 0.250000 + 0.750000 = 1.000000
sin^2 (165.0 deg) + cos^2 (165.0 deg) = 0.066987 + 0.933013 = 1.000000
sin^2 (180.0 deg) + cos^2 (180.0 deg) = 0.000000 + 1.000000 = 1.000000
```

12 Lists and dictionaries

Python provides powerful data structure, such as lists and dictionaries.

12.1 Lists

Lists are collections of objects, and can be used for data analysis. Play with lists.

Python Code 15: advobs202202_s01_15.py

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

# making a list
list_a = [1, 5, 2, 8, 3, 6, 0, 9, 4, 7]

# printing a list
print ("list_a =", list_a)

# accessing to a value using index
print ("list_a[0] =", list_a[0])
print ("list_a[5] =", list_a[5])
print ("list_a[-2] =", list_a[-2])

# accessing to values using slicing
print ("list_a[2:6] =", list_a[2:6])
print ("list_a[:4] =", list_a[:4])
print ("list_a[3:-4] =", list_a[3:-4])
print ("list_a[:] =", list_a[:])

# copying a value using index
scalar_b = list_a[7]
print ("scalar_b =", scalar_b)

# copying values using slicing
list_c = list_a[1:7]
print ("list_c =", list_c)

# number of elements
n_a = len (list_a)
print ("number of elements of list_a =", n_a)
n_c = len (list_c)
```

```
print ("number of elements of list_c =", n_c)

# appending an element to the end of list_c
list_c.append (10)
print ("list_c =", list_c)

# appending list to the end of list
list_c.extend ([50, 20, 30])
print ("list_c =", list_c)

# sorting list
list_d = sorted (list_c)
print ("list_d =", list_d)
```

Execute the script.

```
% chmod a+x advobs202202_s01_15.py
% ./advobs202202_s01_15.py
list_a = [1, 5, 2, 8, 3, 6, 0, 9, 4, 7]
list_a[0] = 1
list_a[5] = 6
list_a[-2] = 4
list_a[2:6] = [2, 8, 3, 6]
list_a[:4] = [1, 5, 2, 8]
list_a[3:-4] = [8, 3, 6]
list_a[:] = [1, 5, 2, 8, 3, 6, 0, 9, 4, 7]
scalar_b = 9
list_c = [5, 2, 8, 3, 6, 0]
number of elements of list_a = 10
number of elements of list_c = 6
list_c = [5, 2, 8, 3, 6, 0, 10]
list_c = [5, 2, 8, 3, 6, 0, 10, 50, 20, 30]
list_d = [0, 2, 3, 5, 6, 8, 10, 20, 30, 50]
```

2-dimensional list can also be used.

Python Code 16: advobs202202_s01_16.py

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

# making a 2-dim. list
list_a = [
    [1, 5, 2],
    [8, 3, 6],
    [4, 9, 7],
    ]

# printing list
print ("list_a =", list_a)

# accessing to a value using index
print ("list_a[1][2] =", list_a[1][2])
print ("list_a[2][0] =", list_a[2][0])
```

Execute the script.

```
% chmod a+x advobs202202_s01_16.py
% ./advobs202202_s01_16.py
list_a = [[1, 5, 2], [8, 3, 6], [4, 9, 7]]
list_a[1][2] = 6
list_a[2][0] = 4
```

12.2 Dictionaries

Dictionaries are collection of data of keys and values. Play with dictionaries.

Python Code 17: advobs202202_s01_17.py

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

# making a dictionary
dic_star_mag = {
    'Sirius': -1.46,
    'Canopus': -0.72,
    'Rigil Kentaurus': -0.27,
    'Arcturus': -0.04,
    'Vega': 0.03,
    'Capella': 0.08,
    'Rigel': 0.12,
    'Procyon': 0.38,
    'Achernar': 0.46,
    'Betelgeuse': 0.50,
}

# printing a dictionary
print (dic_star_mag)

# adding an element
dic_star_mag['Polaris'] = 1.98

# printing a dictionary
print (dic_star_mag)

# accessing an element
print ("visual mag of Betelgeuse =", dic_star_mag['Betelgeuse'])
```

Execute the script.

```
% chmod a+x advobs202202_s01_17.py
% ./advobs202202_s01_17.py
{'Sirius': -1.46, 'Canopus': -0.72, 'Rigil Kentaurus': -0.27, 'Arcturus': -0.04,
 'Vega': 0.03, 'Capella': 0.08, 'Rigel': 0.12, 'Procyon': 0.38, 'Achernar': 0.46
, 'Betelgeuse': 0.5}
{'Sirius': -1.46, 'Canopus': -0.72, 'Rigil Kentaurus': -0.27, 'Arcturus': -0.04,
 'Vega': 0.03, 'Capella': 0.08, 'Rigel': 0.12, 'Procyon': 0.38, 'Achernar': 0.46
, 'Betelgeuse': 0.5, 'Polaris': 1.98}
visual mag of Betelgeuse = 0.5
```

Multi-dimensional dictionaries can also be used.

Python Code 18: advobs202202_s01_18.py

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

# making a dictionaries of dictionary
dic_stars = {
    'Sirius': {
        'mag': -1.46,
        'dist': 2.670,
        'sptype': 'A0V',
    },
    'Canopus': {
        'mag': -0.72,
        'dist': 95,
        'sptype': 'A9II',
    },
    'Rigil Kentaurus': {
        'mag': -0.27,
        'dist': 1.339,
        'sptype': 'G2V',
    },
    'Arcturus': {
        'mag': -0.04,
        'dist': 11.26,
        'sptype': 'K1.5III',
    },
    'Vega': {
        'mag': -0.04,
        'dist': 7.68,
        'sptype': 'A0V',
    },
}

# printing dictionary of dictionaries
print (dic_stars)

# accessing to an element
print ("dic_stars['Vega']['dist'] =", dic_stars['Vega']['dist'], "pc")
print ("dic_stars['Canopus']['sptype'] =", dic_stars['Canopus']['sptype'])
```

Execute the script.

```
% chmod a+x advobs202202_s01_18.py
% ./advobs202202_s01_18.py
{'Sirius': {'mag': -1.46, 'dist': 2.67, 'sptype': 'A0V'}, 'Canopus': {'mag': -0.
72, 'dist': 95, 'sptype': 'A9II'}, 'Rigil Kentaurus': {'mag': -0.27, 'dist': 1.3
39, 'sptype': 'G2V'}, 'Arcturus': {'mag': -0.04, 'dist': 11.26, 'sptype': 'K1.5I
II'}, 'Vega': {'mag': -0.04, 'dist': 7.68, 'sptype': 'A0V'}}
dic_stars['Vega']['dist'] = 7.68 pc
dic_stars['Canopus']['sptype'] = A9II
```

13 Control flow statements

Python offers number of control flow statements.

13.1 for statement

Try “for” statement.

Python Code 19: advobs202202_s01_19.py

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

# importing math module
import math

# pi
pi = math.pi

# making a list
list_a = list (range (0, 181, 30))

# using "for" statement
for i in list_a:
    print ("sin (%03d deg) = %f" % (i, math.sin (i / 180.0 * pi) ) )

# initialising a parameter "total"
total = 0

# adding numbers from 0 to 10
for i in range (0, 11):
    total += i

# printing "total"
print ("0 + 1 + 2 + ... + 8 + 9 + 10 =", total)
```

Execute the script.

```
% chmod a+x advobs202202_s01_19.py
% ./advobs202202_s01_19.py
sin (000 deg) = 0.000000
sin (030 deg) = 0.500000
sin (060 deg) = 0.866025
sin (090 deg) = 1.000000
sin (120 deg) = 0.866025
sin (150 deg) = 0.500000
sin (180 deg) = 0.000000
0 + 1 + 2 + ... + 8 + 9 + 10 = 55
```

13.2 while statement

Play with `while` statement.

Python Code 20: advobs202202_s01_20.py

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

# counter
i = 0

# maximum value
i_max = 100

# total
total = 0

# using "while" statement
```

```
while (i <= i_max):
    # adding number to "total"
    total += i
    # incrementing "i"
    i += 1

# printing result
print ("0 + 1 + 2 + ... + %d = %d" % (i_max, total) )
```

Execute the script.

```
% chmod a+x advobs202202_s01_20.py
% ./advobs202202_s01_20.py
0 + 1 + 2 + ... + 100 = 5050
```

13.3 if statement

Play with `if`, `elif`, and `else` statements.

Python Code 21: advobs202202_s01_21.py

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

# importing argparse module
import argparse

# construction of parser object
desc = 'divisible by 3?'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('numbers', type=int, nargs='+', help='numbers')

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

# numbers
numbers = args.numbers

# processing each number
for i in numbers:
    # remainder
    remainder = i % 3
    # use of "if" statement
    if (remainder == 0):
        print ("%d is divisible by 3, and remainder is 0." % i)
    elif (remainder == 1):
        print ("%d is not divisible by 3, and remainder is 1." % i)
    else:
        print ("%d is not divisible by 3, and remainder is 2." % i)
```

Execute the script.

```
% chmod a+x advobs202202_s01_21.py
% ./advobs202202_s01_21.py 1 2 3 4 5 6 7 8 9
1 is not divisible by 3, and remainder is 1.
2 is not divisible by 3, and remainder is 2.
```

```
3 is divisible by 3, and remainder is 0.
4 is not divisible by 3, and remainder is 1.
5 is not divisible by 3, and remainder is 2.
6 is divisible by 3, and remainder is 0.
7 is not divisible by 3, and remainder is 1.
8 is not divisible by 3, and remainder is 2.
9 is divisible by 3, and remainder is 0.
```

14 Exercises

1. If you do not have Linux (or BSD) on your computer, install Linux (or BSD) on your computer. If you do not wish to destroy your favourite operating system currently running on your computer, you may use hypervisor to install Linux (or BSD) on a virtual machine. Describe the way to install Linux (or BSD).

2. If you do not have Python on your computer, install the latest version of 3.9 series of Python on your computer. Describe the way to install Python.

3. If you do not have NumPy, SciPy, Matplotlib, Astropy, Astroquery, Photutils, and ginga, install them on your computer. Describe the way to install those packages.

4. Find your favourite text editor. Describe how you like it. What is the unique feature of your favourite text editor? Describe basic usage of the text editor of your choice.

5. Make three useful Python scripts which utilise `argparse` module. Show the source code of those Python scripts. Execute them. Take a screenshot of your computer display, and show the result of the execution. Describe the design of your scripts.

6. Read “Python Tutorial” on the Python official website, and learn more about Python. Summarise what you have learnt.