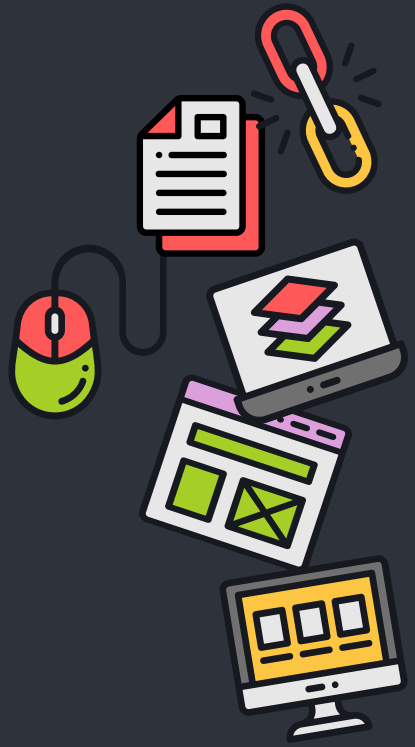
codeastro.py

Day_2.json

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
import astropy
```



Maria Vincent

codeastro.py

Day_2.json

Contents of This Lesson:

{

- What is astropy
- Subpackages
- Units Quantities and Constants
- Observation Planning
- Inputs and Outputs
- Q&A

}

Maria Vincent

codeastro.py

Day_2.json

```
class what_is_astropy:
    def __init__(self):
        return definition
```

Maria Vincent

codeastro.py

Day_2.txt

Astropy is a Python package that offers tools and functions that are useful for various tasks in astronomy and astrophysics, including and not limited to planning an observation, reducing the data from the observation, analysing it, and other numerical and modeling tasks.

codeastro.py

Day_2.json

```
class subpackages:
    def __init__(self):
        return list_of_possibilities
```

codeastro.py

Day_2.txt

```
from astropy import subpackage
import astropy.subpackage as s
from astropy.subpackage import class

"""
Most of astropy's functionalities lie in
subpackages, that can be imported as above,
or using shortcuts, or a class alone can be
imported from a subpackage
"""
```

codeastro.py Day_2.txt

```
1
2   Coordinated and affiliated packages
3
4   """
5
6   Packages within the Astropy Project Community.
7   """
8
9
10  Coordinated: Packages maintained by astropy
11
12  Affiliated:  Other packages for astronomy
13
14
```

Subpackages

units	Handles defining, converting between, and performing arithmetic with physical quantities
constants	Contains a number of physical constants useful in Astronomy
coordinates	Contains classes for celestial/spatial coordinates and their velocity components, and tools to uniformly convert between common coordinate systems.

Coordinated packages

astroquery	Tools for querying online astronomical data sources.
ccdproc	Package to do basic CCD data reduction.
photutils	Photometry and related image-processing tools.

Affiliated packages

astroML	Tools for machine learning and data mining in Astronomy.
astroplan	An open source Python package to help astronomers plan observations.
dust_extinction	Interstellar dust extinction curves

codeastro.py

Day_2.json

```
class quantities:
    def __init__(self):
        return list_of_possibilities
```

codeastro.py

quantities.py

```
1   import astropy.units as u
2
3
4
5    """
6    Helpful when handling and calculating using
7    different quantities with different units
8    """
9
10
11
12
13
14
```

Maria Vincent

codeastro.py

quantities.py

```
import astropy.units as u


q = 15.1 * u.meter / (32.0 * u.second)
print(q.value)
print(q.unit)
```

codeastro.py

quantities.py

```
1  import astropy.units as u
2
3
4  q = 15.1 * u.meter / (32.0 * u.second)
5  print(q.value)
6  print(q.unit)
7
8
```

```
0.471875
m / s
```

codeastro.py

quantities.py

```
1   import astropy.units as u
2
3
4   x = 1.0 * u.parsec
5   x.to(u.km)
6
7
8
9
10
11
12
13
14
```

codeastro.py

quantities.py

```python
import astropy.units as u


x = 1.0 * u.parsec
x.to(u.km)


```

```
<Quantity 3.08567758e+13 km>
```

codeastro.py

quantities.py

```
1   import astropy.constants as c
2
3
4
5    """
6    A library of the oft-used constants in
7
8    astronomy, to avoid having to create
9    variables
10   """
11
12
13
14
```

Maria Vincent

codeastro.py

quantities.py

```
1   import astropy.constants as c
2
3
4   print(c.G)
5
6
7
8
9
10
11
12
13
14
```

codeastro.py quantities.py

```python
import astropy.constants as c


print(c.G)


```

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

Maria Vincent

codeastro.py

quantities.py

```
import astropy.units as u
import astropy.constants as c



Calculate the gravitational force between a 3
solar mass star and its 10 Jupiter mass planet
2 au away.
```

ASSIGNMENT

ASSIGNMENT

codeastro.py

quantities.py

```python
import astropy.units as u
import astropy.constants as c

m_star = 3 * c.M_sun
m_planet = 10 * c.M_jup
distance = 2 * u.au
F = c.G * m_star * m_planet / distance**2
print(F)
```

```
1.889285600523636e+48 kg m3 / (AU2 s2)
```

Maria Vincent

codeastro.py

Day_2.json

```
class input_out:
    def __init__(self):
        return i/o
```

Maria Vincent

codeastro.py

input_output.txt

A unified interface for reading and writing data in different formats

Maria Vincent

codeastro.py

input_output.py

```
1   from astropy.table import Table
2
3
4   t = Table.read('photometry.dat',format='ascii.daophot')
5   filename = 'photometry_latex.tex'
6   t.write(filename, format='latex')
7
8
9
10
11
12
13
14
```

codeastro.py

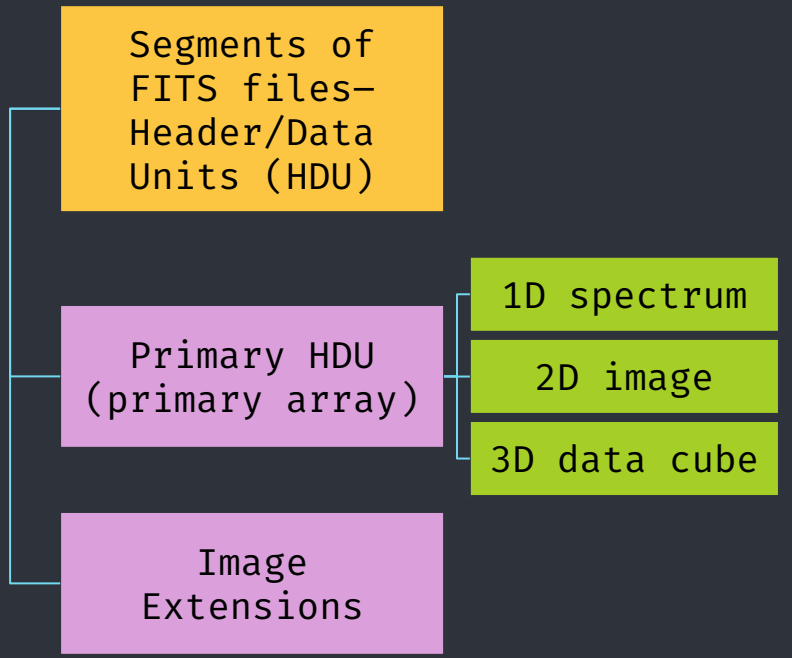
input_output.py

```
FITS (Flexible Image Transport
System) is the data format most
widely used within astronomy for
transporting, analyzing, and
archiving scientific data files.
FITS is much more than just
another image format (such as JPG
or GIF) and is primarily designed
to store scientific data sets
consisting of multidimensional
arrays (images) and 2-dimensional
tables organized into rows and
columns of information

```

Maria Vincent

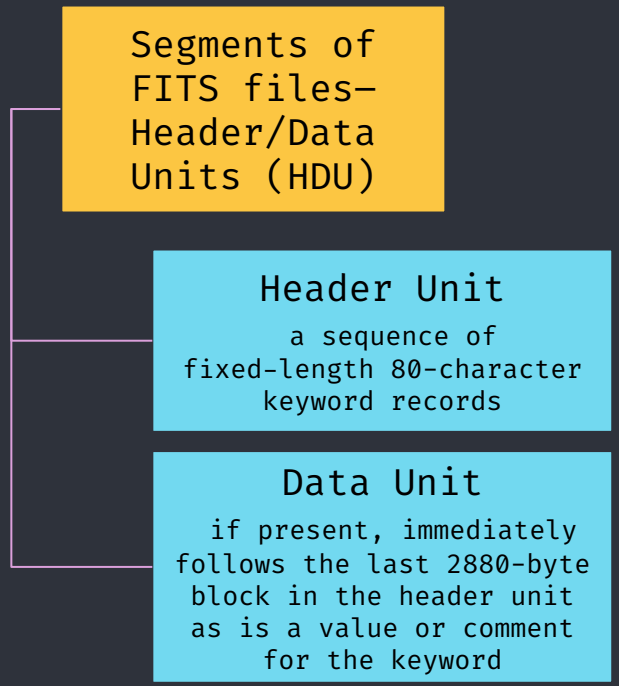
FITS (Flexible Image Transport System) is the data format most widely used within astronomy for transporting, analyzing, and archiving scientific data files. FITS is much more than just another image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multidimensional arrays (images) and 2-dimensional tables organized into rows and columns of information



codeastro.py

input_output.py

FITS (Flexible Image Transport System) is the data format most widely used within astronomy for transporting, analyzing, and archiving scientific data files. FITS is much more than just another image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multidimensional arrays (images) and 2-dimensional tables organized into rows and columns of information



codeastro.py

input_output.py

```
1   from astropy.io import fits
2   fits_img_fn =
3   fits.util.get_testdata_filepath('test0.fits')
4   # Data that came with astropy installation
5   hdul = fits.open(fits_img_fn) # header data unit (HDU)
6   list
7   hdul.info()
8
9
10
11
12
13
14
```

codeastro.py

input_output.py

```python
from astropy.io import fits
fits_img_fn =
fits.util.get_testdata_filepath('test0.fits')
# Data that came with astropy installation
hdul = fits.open(fits_img_fn) # header data unit (HDU)
list
hdul.info()
```

```
Filename:
/Users/mariavincent/anaconda3/lib/python3.11/site-packages/astropy/io/fits/t
ests/data/test0.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     138   ()
  1  SCI           1 ImageHDU        61   (40, 40)   int16
  2  SCI           2 ImageHDU        61   (40, 40)   int16
  3  SCI           3 ImageHDU        61   (40, 40)   int16
  4  SCI           4 ImageHDU        61   (40, 40)   int16
```

codeastro.py

input_output.py

```
1  from astropy.io import fits
2  fits_img_fn =
3  fits.util.get_testdata_filepath('test0.fits')
4  # Data that came with astropy install
5  hdul = fits.open(fits_img_fn)
6  list
7  hdul.info()
```

What do up get from generating the data in the file?

```
Filena
                                    ython3.11/site-packages/astropy/io/fits/t
N                    Type      Cards   Dimensions   Format
                  1 PrimaryHDU     138   ()
1  SCI            1 ImageHDU        61   (40, 40)   int16
2  SCI            2 ImageHDU        61   (40, 40)   int16
3  SCI            3 ImageHDU        61   (40, 40)   int16
4  SCI            4 ImageHDU        61   (40, 40)   int16
```

codeastro.py

Day_2.json

```
class observations:
    def __init__(self):
        return planning_observations
```

Maria Vincent

codeastro.py

observations.txt

```
1
2
3
4
5       Plan observations
6       taking into account the
7       targets, dates and
8       times, location of
9
10      observing, moon
11
12
13
14
```

Maria Vincent

codeastro.py observations.py

```
1  from astropy.coordinates import SkyCoord, EarthLocation,
2  AltAz
3  from astroplan import Observer
4
5
6
7
8
9
10
11
12
13
14
```

You have to plan an observation of Fomalhaut from Subaru. The first step is to generate the coordinates of the star and the observing site. The packages here are a hint

ASSIGNMENT

ASSIGNMENT

codeastro.py

observations.py

```python
from astropy.coordinates import SkyCoord, EarthLocation, AltAz
from astroplan import Observer
fomalhaut = SkyCoord.from_name('Fomalhaut')
subaru = Observer.at_site('Subaru', timezone ='US/Hawaii')
subaru_loc = EarthLocation(lat=subaru.latitude, lon=subaru.longitude)
print(f'Sky Coords of Fomalhaut: {fomalhaut} and Geocentric Coords of Subaru: {subaru_loc}')
```

```
Sky Coords of Fomalhaut: <SkyCoord (ICRS): (ra, dec) in deg
    (344.41269272, -29.62223703)> and Geocentric Coords of
Subaru: (-5460925.6854608, -2491437.49064199, 2149539.83199706)
m
```

codeastro.py

observations.txt

Optional exercise for the attendees: There are other tools you need like astropy.time and astropy.visualization to plot and visualize the night-time observability

*Because the presenter may have been tired…

OPTIONAL ASSIGNMENT

Maria Vincent

codeastro.py

Day_2.json

```
class questions:
    def __init__(self):
        return answers
```

Maria Vincent

References:

- https://philuttley.github.io/prog4aa_lesson2/09-astropyintro/index.html
- https://www.astropy.org/affiliated/index.html
- https://docs.astropy.org/en/stable/units/
- https://fits.gsfc.nasa.gov/fits_primer.html

What I hope you learnt:

- The existence of astropy
- Some of the tools available in astropy for astronomical math, observation planning, and data analysis
- How to wield these tools