

# 12.010 Computational Methods of Scientific Programming

Lecture 12: Libraries, Geopandas, xarray

# Summary

- Finding libraries for Python. Examples for some of the topics already covered

# Installing geopandas (or use navigator)

```
conda create -n geo_env # Creating a new environment should protect base installation
conda activate geo_env
conda config --env --add channels conda-forge
# conda config --env --set channel_priority strict # Might be dangerous in terms of why packages come from.
conda init tcsh
tcsh
conda install geopandas
conda install jupyter notebook
conda install xarray
# Note sure is needed:
#python -m ipykernel install --name geo_env
# Run from terminal instead of anaconda.
jupyter notebook
```

In anaconda: A geo\_env environment should now appear and when selected, geopandas should be available.

# Finding libraries – there are lots!

- SciPy – catalog (<https://www.scipy.org/topical-software.html>)
  - The SciPy catalog pages have links to many useful tools in different categories
    - Environments e.g. Anaconda, Spyder etc...
    - Science tools e.. ImageIO, AstroPy, SpacePy, BioPython, ClimPy, SymPy
- Example notebooks –
  - <https://www.earthcube.org/notebooks> - some showcases for Earth science
  - <https://joss.theoj.org> - a general online journal for scientific software cataloging (and DOI)
  - <https://paperswithcode.com> ) – ML focused
- PyPi.org
  - tensorflow, pytorch, JAX, ....

332,311 projects

2,934,040 releases

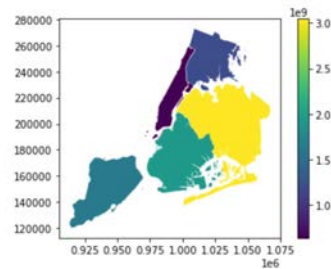
4,979,823 files

542,528 users

# Tabular data beyond Pandas

- Pandas provides tabular/column data in “2d”
- geopandas
  - It builds on top of pandas, and adds some geospatial concepts built-in

```
# Libraries like geopandas have higher-level functions
# Let's plot the shapes of each Borough, colored by
#
# this only needs one line because the gdf.plot function
# gdf object (a GeoDataFrame) has an attribute .geometry
#
gdf.plot("area", legend=True)
```



- xarray
  - similar but geared to multi-dimensional data more generally (also builds on pandas)

xarray.Dataset				
» Dimensions: (lat: 25, lon: 53, time: 2920)				
▼ Coordinates:				
lat	(lat)	float32	75.0 72.5 70.0 ... 20.0 17.5 15.0	
lon	(lon)	float32	200.0 202.5 205.0 ... 327.5 330.0	
time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	
▼ Data variables:				
air	(time, lat, lon)	float32	-31.95 -30.65 ... 23.04 22.54	
▼ Attributes:				
Conventions :	COARDS			
title :	4x daily NMC reanalysis (1948)			
description :	Data is from NMC initialized reanalysis (4x/day). These are the 0.9950 sigma level values.			
platform :	Model			
references :	<a href="http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html">http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html</a>			

# Tabular data beyond Pandas

- geopandas - builds on top of pandas, adds some geospatial concepts built-in

```
: # GeoDataFrame has a default display method for geometry
display( gdf.geometry[3] )
display( gdf.boundary[3] )
```



```
# Simple geopandas example ( from https://geopandas.org/ge
#
# illustrate "polyarea" like problem with builtin New Yor
#
import geopandas

path_to_data = geopandas.datasets.get_path("nybb")
gdf = geopandas.read_file(path_to_data)

gdf
```

	BoroCode	BoroName	Shape_Leng	Shape_Area	
0	5	Staten Island	330470.010332	1.623820e+09	MULTIPOLY
1	4	Queens	896344.047763	3.045213e+09	MULTIPOLY
2	3	Brooklyn	741080.523166	1.937479e+09	MULTIPOL
3	1	Manhattan	359299.096471	6.364715e+08	MULTIPOLY
4	2	Bronx	464392.991824	1.186925e+09	MULTIPOLY

# Tabular data beyond Pandas

- geopandas – create a polygon example

```
[22]: gb=g.buffer(0.5)  
display( gb[0], gb[1], gb[2] )
```



10/24/24

```
[11]: import geopandas  
from geopandas import GeoSeries  
from shapely.geometry import Polygon  
p1 = Polygon([(0, 0), (1, 0), (1, 1)])  
p2 = Polygon([(0, 0), (1, 0), (1, 1), (0, 1)])  
p3 = Polygon([(2, 0), (3, 0), (3, 1), (2, 1)])  
g = GeoSeries([p1, p2, p3])  
g  
  
[11]: 0    POLYGON ((0.00000 0.00000, 1.00000 0.00000, 1....  
1    POLYGON ((0.00000 0.00000, 1.00000 0.00000, 1....  
2    POLYGON ((2.00000 0.00000, 3.00000 0.00000, 3....  
dtype: geometry
```

```
[13]: g.area
```

```
[13]: 0    0.5  
1    1.0  
2    1.0  
dtype: float64
```

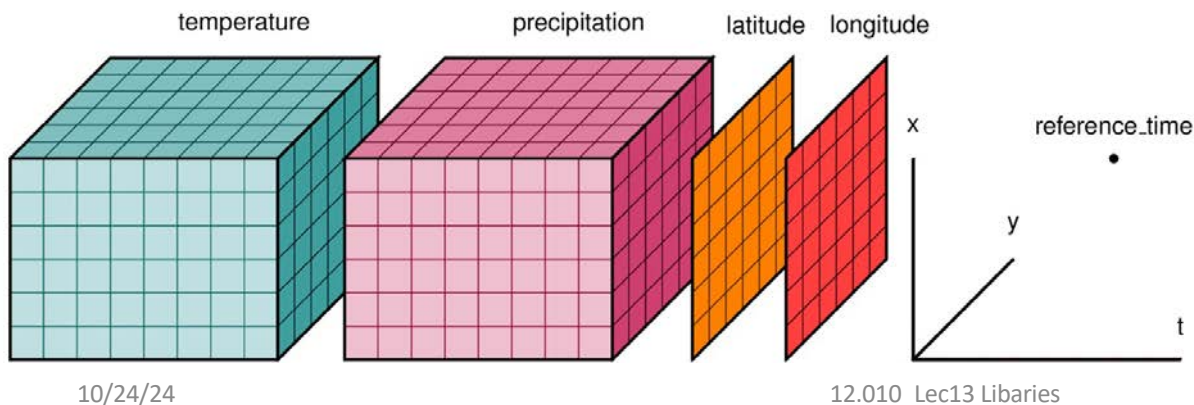
```
[19]: display( g[0],g[1],g[2] )
```



12.010\_Lec13 Libai

# Tabular data beyond Pandas

- xarray - geared to multi-dimensional data more generally (also builds on pandas)
  - central abstraction is a “Dataset”
  - provides I/O, plotting, math, and sampling APIs for “Datasets”.



```
display( ds )
display( ds.air.coords )
tn=0
plt.contourf(ds.lon,ds.lat,ds.air[tn,:,:])
ts=np.datetime_as_string(ds.time[tn].values,'h');
plt.title(r"Air Temperature ($^{\rm o}$C) t=%s"%(ts));
```

xarray.Dataset

► Dimensions: (lat: 25, lon: 53, time: 2920)

▼ Coordinates:

lat	(lat)	float32	75.0 72.5
lon	(lon)	float32	200.0 202.5
time	(time)	datetime64[ns]	2013-01-01 ... 2014-01-01

▼ Data variables:

air	(time, lat, lon)	float32	-31.95 -30.0
-----	------------------	---------	--------------

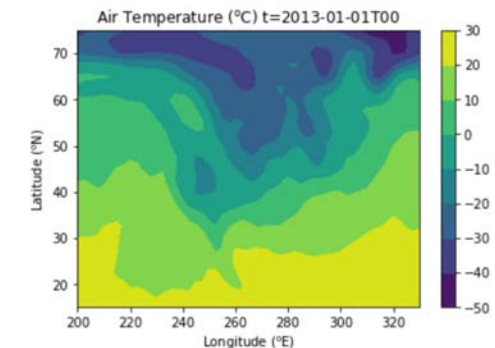
▼ Attributes:

Conventions : COARDS  
 title : 4x daily NMC reanalysis (1948)  
 description : Data is from NMC initialized reanalysis (4x/day). These are the 0.9950 sigma level  
 platform : Model  
 references : <http://www.esrl.noaa.gov/psd/data/gridded>

Coordinates:

```
* lat (lat) float32 75.0 72.5 70.0 67.5 65.0 ...
* lon (lon) float32 200.0 202.5 205.0 207.5 ...
* time (time) datetime64[ns] 2013-01-01 ... 2014-01-01
```

<matplotlib.colorbar.Colorbar at 0x7f6988522110>

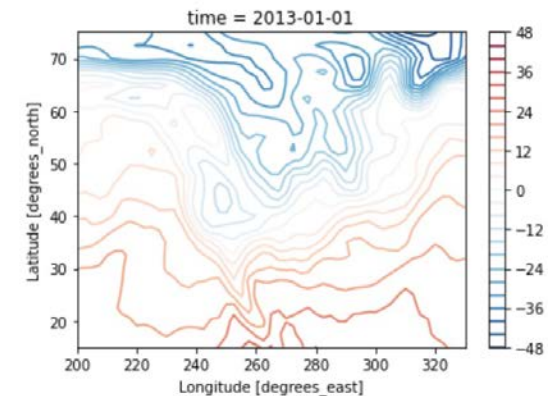




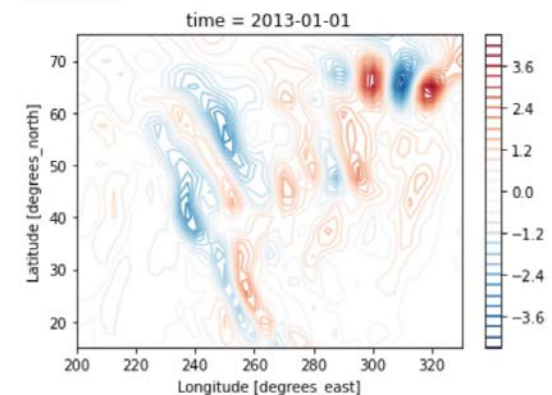
# Tabular data beyond Pandas

- xarray - calculate derivative example
  - xarray abstraction understands a bit about math operations on fields
  - for example, we can ask for a derivative wrt a particular “coordinate” dimension
    - derivative returns centered derivative where the field is replaced with its derivative (but the name stays the same!).

```
[13]: ds.air[0,:,:].plot.contour(levels=30,add_colorbar=True)
plt.show()
ds.differentiate("lon").air[0,:,:].plot.contour(levels=30,add_colorbar=True)
```



```
[13]: <matplotlib.contour.QuadContourSet at 0x7f8e8235250>
```

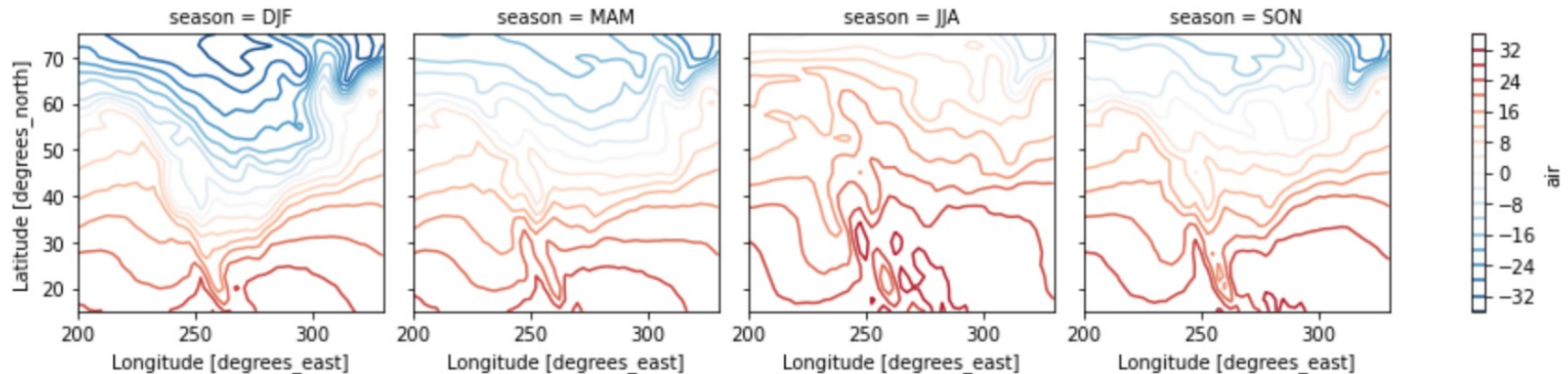


# Tabular data beyond Pandas

- xarray - quick plot example

```
[25]: seasonal_mean = ds.groupby("time.season").mean()  
seasonal_mean = seasonal_mean.reindex(season=["DJF", "MAM", "JJA", "SON"])  
seasonal_mean.air.plot.contour(col="season", levels=20, add_colorbar=True)
```

```
[25]: <xarray.plot.facetgrid.FacetGrid at 0x7f69767d8fd0>
```



# Try

`Lec13-geopandas-example.ipynb`

`Lec13-xarray-example.ipynb`

# Math beyond SciPy/Numpy















- Numpy “taxonomy” (numpy.org)

## ECOSYSTEM

SCIENTIFIC DOMAINS   ARRAY LIBRARIES   DATA SCIENCE   MACHINE LEARNING   VISUALIZATION

Nearly every scientist working in Python draws on the power of NumPy.

NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes simplicity: a solution in NumPy is often clear and elegant.

<b>Quantum Computing</b>  QuTIP PyQuil Qiskit	<b>Statistical Computing</b>  Pandas statsmodels Xarray Seaborn	<b>Signal Processing</b>  SciPy PyWavelets python-control	<b>Image Processing</b>  Scikit-image OpenCV Mahotas	<b>Graphs and Networks</b>  NetworkX graph-tool igraph PyGSP	<b>Astronomy Processes</b>  AstroPy SunPy SpacePy	<b>Cognitive Psychology</b>  PsychoPy
<b>Bioinformatics</b>  BioPython Scikit-Bio PyEnsembl ETE	<b>Bayesian Inference</b>  PyStan PyMC3 ArviZ emcee	<b>Mathematical Analysis</b>  SciPy SymPy cvxpy FEniCS	<b>Chemistry</b>  Cantera MDAnalysis RDKit	<b>Geoscience</b>  Pangeo Simpeg ObsPy Fatiando a Terra	<b>Geographic Processing</b>  Shapely GeoPandas Folium	<b>Architecture &amp; Engineering</b>  COMPAS City Energy Analyst Sverchok














© 2025 NumPy team. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Math beyond SciPy/Numpy

- Numpy “taxonomy” (numpy.org)

SCIENTIFIC DOMAINS   ARRAY LIBRARIES   DATA SCIENCE   MACHINE LEARNING   VISUALIZATION

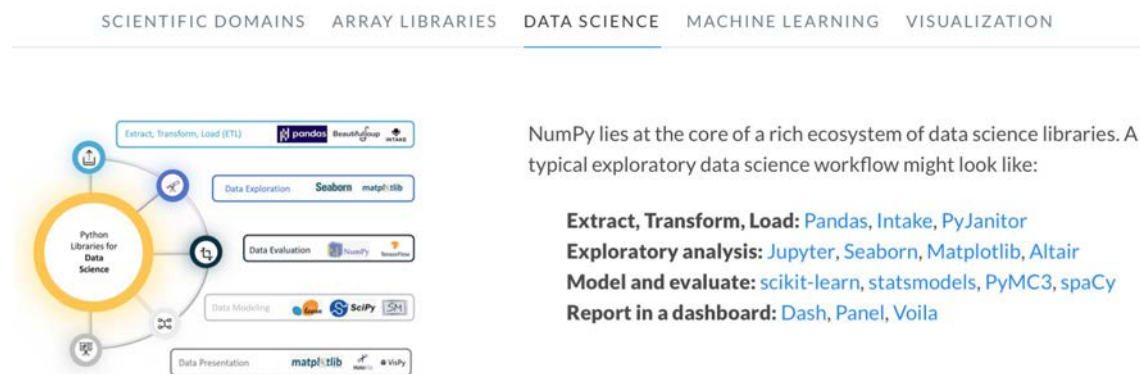
NumPy's API is the starting point when libraries are written to exploit innovative hardware, create specialized array types, or add capabilities beyond what NumPy provides.

	Array Library	Capabilities & Application areas
	Dask	Distributed arrays and advanced parallelism for analytics, enabling performance at scale.
	CuPy	NumPy-compatible array library for GPU-accelerated computing with Python.
	JAX	Composable transformations of NumPy programs: differentiate, vectorize, just-in-time compilation to GPU/TPU.
	Xarray	Labeled, indexed multi-dimensional arrays for advanced analytics and visualization
	Sparse	NumPy-compatible sparse array library that integrates with Dask and SciPy's sparse linear algebra.
	PyTorch	Deep learning framework that accelerates the path from research prototyping to production deployment.
	TensorFlow	An end-to-end platform for machine learning to easily build and deploy ML powered applications.
	MXNet	Deep learning framework suited for flexible research prototyping and production.
	Arrow	A cross-language development platform for columnar in-memory data and analytics.
	xtensor	Multi-dimensional arrays with broadcasting and lazy computing for numerical analysis.
	XND	Develop libraries for array computing, recreating NumPy's foundational concepts.
	uarray	Python backend system that decouples API from implementation; unumpy provides a NumPy API.
	tensorly	Tensor learning, algebra and backends to seamlessly use NumPy, MXNet, PyTorch, TensorFlow or CuPy.

© 2025 NumPy team. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Math beyond SciPy/Numpy

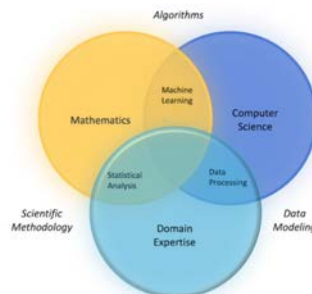
- Numpy “taxonomy” (numpy.org)



NumPy lies at the core of a rich ecosystem of data science libraries. A typical exploratory data science workflow might look like:

**Extract, Transform, Load:** Pandas, Intake, PyJanitor  
**Exploratory analysis:** Jupyter, Seaborn, Matplotlib, Altair  
**Model and evaluate:** scikit-learn, statsmodels, PyMC3, spaCy  
**Report in a dashboard:** Dash, Panel, Voila

For high data volumes, Dask and Ray are designed to scale. Stable deployments rely on data versioning (DVC), experiment tracking (MLFlow), and workflow automation (Airflow and Prefect).

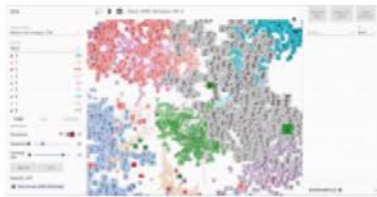


© 2025 NumPy team. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Math beyond SciPy/Numpy

- Numpy “taxonomy” (numpy.org)

SCIENTIFIC DOMAINS   ARRAY LIBRARIES   DATA SCIENCE   MACHINE LEARNING   VISUALIZATION



Source: Google AI Blog

NumPy forms the basis of powerful machine learning libraries like [scikit-learn](#) and [SciPy](#). As machine learning grows, so does the list of libraries built on NumPy. [TensorFlow](#)'s deep learning capabilities have broad applications — among them speech and image recognition, text-based applications, time-series analysis, and video detection. [PyTorch](#), another deep learning library, is popular among researchers in computer vision and natural language processing. [MXNet](#) is another AI package, providing blueprints and templates for deep learning.

Statistical techniques called [ensemble](#) methods such as binning, bagging, stacking, and boosting are among the ML algorithms implemented by tools such as [XGBoost](#), [LightGBM](#), and [CatBoost](#) — one of the fastest inference engines. [Yellowbrick](#) and [Eli5](#) offer machine learning visualizations.

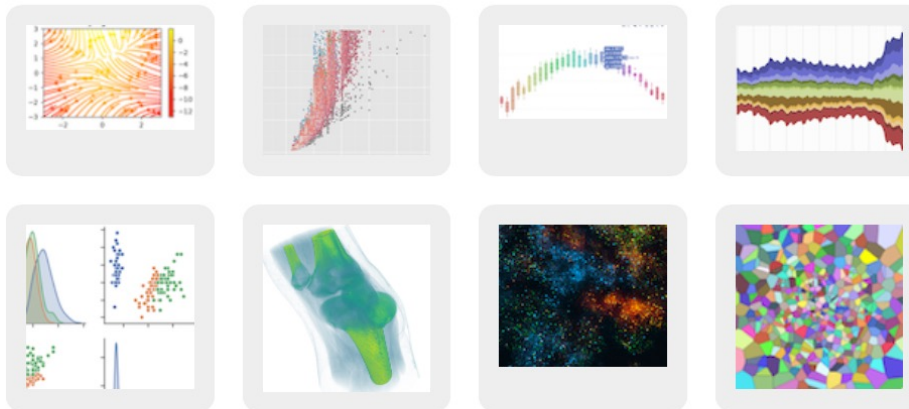
© Google. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.



# Math beyond SciPy/NumPy

- NumPy “taxonomy” (numpy.org)

SCIENTIFIC DOMAINS   ARRAY LIBRARIES   DATA SCIENCE   MACHINE LEARNING   VISUALIZATION



NumPy is an essential component in the burgeoning [Python visualization landscape](#), which includes [Matplotlib](#), [Seaborn](#), [Plotly](#), [Altair](#), [Bokeh](#), [Holoviz](#), [Vispy](#), [Napari](#), and [PyVista](#), to name a few.

NumPy’s accelerated processing of large arrays allows researchers to visualize datasets far larger than native Python could handle.

© 2025 NumPy team. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.



# Math beyond SciPy/Numpy

**nature** | **methods**

**PERSPECTIVE**

<https://doi.org/10.1038/s41592-019-0686-2>

There are amendments to this paper

**OPEN**

## **SciPy 1.0: fundamental algorithms for scientific computing in Python**

<https://www.nature.com/articles/s41592-019-0686-2.pdf>

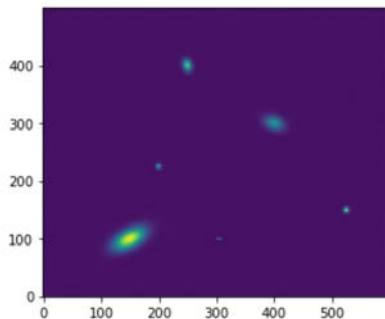
© 2025 Springer Nature Limited. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Astropy (<https://astropy.org>)

Example of an external package that has a lot of useful applications generally.

```
[2]: import numpy as np
      from astropy.modeling.models import Gaussian2D
      y, x = np.mgrid[0:500, 0:600]
      data = (Gaussian2D(1, 150, 100, 20, 10, theta=0.5)(x, y)
              + Gaussian2D(0.5, 400, 300, 8, 12, theta=1.2)(x, y)
              + Gaussian2D(0.75, 250, 400, 5, 7, theta=0.23)(x, y)
              + Gaussian2D(0.9, 525, 150, 3, 3)(x, y)
              + Gaussian2D(0.6, 200, 225, 3, 3)(x, y))
      data += 0.01 * np.random.randn(500, 600)
      cosmic_ray_value = 0.997
      data[100, 300:310] = cosmic_ray_value
```

```
[4]: import matplotlib.pyplot as plt
      plt.imshow(data, origin='lower');
```



© 2013, Kyle Barbary. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Sympy (<https://www.sympy.org>)

## Symbolic manipulation

```
sympy.Matrix([[a, b, c], [d, e, f], [g, h, i]]).inv()
```

$$\begin{bmatrix} \frac{ei-fh}{aei-afh-bdi+bfh+cdh-ceg} & \frac{-bi+ch}{aei-afh-bdi+bfh+cdh-ceg} & \frac{bf-ce}{aei-afh-bdi+bfh+cdh-ceg} \\ \frac{-di+fg}{aei-afh-bdi+bfh+cdh-ceg} & \frac{ai-cg}{aei-afh-bdi+bfh+cdh-ceg} & \frac{-af+cd}{aei-afh-bdi+bfh+cdh-ceg} \\ \frac{dh-eg}{aei-afh-bdi+bfh+cdh-ceg} & \frac{-ah+bg}{aei-afh-bdi+bfh+cdh-ceg} & \frac{ae-bd}{aei-afh-bdi+bfh+cdh-ceg} \end{bmatrix}$$

## unyt (<https://unyt.readthedocs.io>)

```
>>> import unyt as u
>>> distance_traveled = [3.4, 5.8, 7.2] * u.mile
>>> print(distance_traveled.to('km'))
[ 5.4717696  9.3341952 11.5872768] km
```



© Alexey U. Gudchenko. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

### unyt

pypi v2.8.0 conda-forge v2.6.0 V2 CI passing docs passing codecov 100% JOSS 10.2/1105/papers.00809



A package for handling numpy arrays with units.

Often writing code that deals with data that has units can be confusing. A function might return an array but at least with plain NumPy arrays, there is no way to easily tell what the units of the data are without somehow knowing a priori.

The unyt package (pronounced like "unit") provides a subclass of NumPy's `ndarray` class that knows about units. For example, one could do:

© 2022 The yt Project . All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

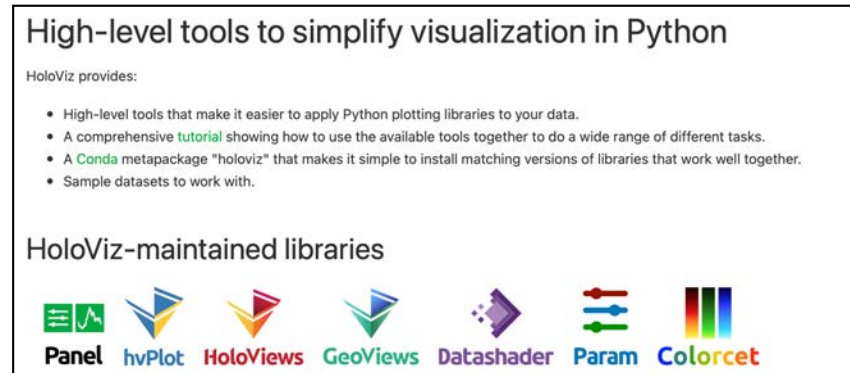
# Try

Lec13\_astropy-example.ipynb

Lec13\_sympy-example.ipynb

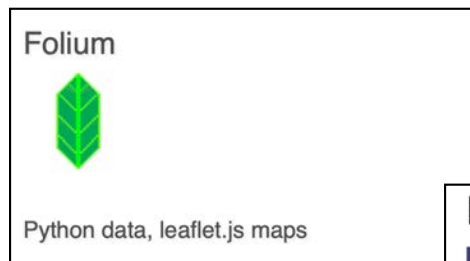
# Jupyter Plotting beyond Matplotlib

- Holoviz and Geoviews



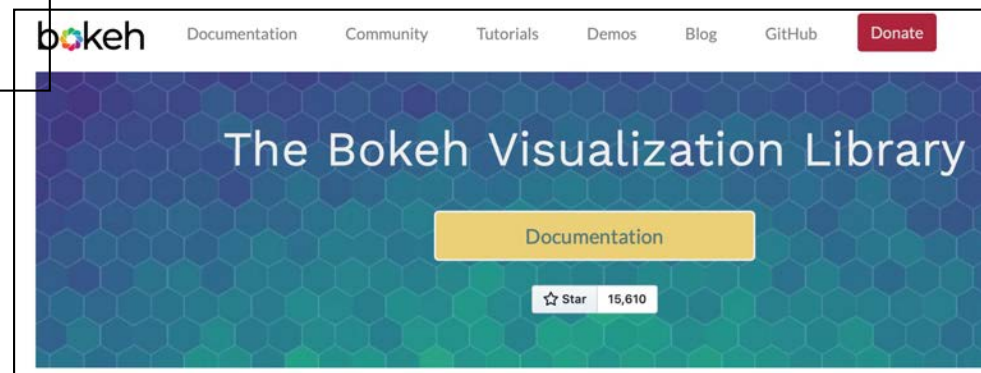
© Source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

- Folium



- Bokeh

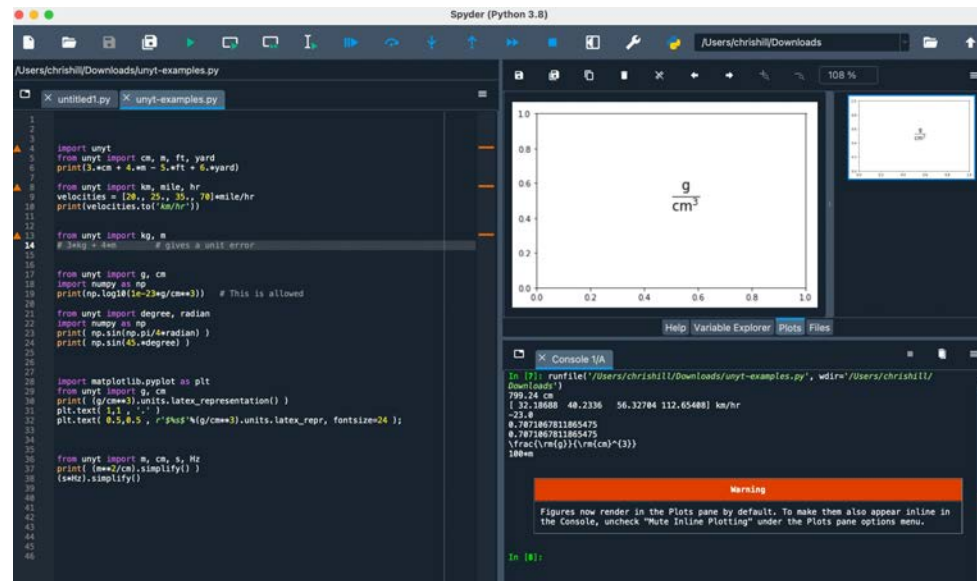
© Rob Story. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.



© 2025 Bokeh contributors. All rights reserved. This content is excluded from our Creative Commons license. For more information, see <https://ocw.mit.edu/help/faq-fair-use>.

# Non - Jupyter

- Spyder
- VS-code, github.dev/



Lec011\_astropy-example.ipynb > # v# Notebook illustrating a few astropy (https://www.astropy.org) examples. Although astropy was designed for the astron  
+ Code + Markdown

```
# Notebook illustrating a few astropy (https://www.astropy.org) examples.
# Although astropy was designed for the astronomical community many of its
# builtin functions are general and can be used for a wide array of
# image and time series processing for all sorts of data.

#
# Functions like Gauss2D (see below) provide simple ways to generate
# Gaussian distributions with various parameters.
#
import numpy as np
from astropy.modeling.models import Gaussian2D
y, x = np.mgrid[0:100, 0:100]
data = (Gaussian2D(1, 150, 100, 20, 10, theta=0.5)(x, y) +
        Gaussian2D(0.5, 400, 300, 8, 12, theta=1.23)(x, y) +
        Gaussian2D(0.75, 250, 400, 5, 7, theta=0.23)(x, y) +
        Gaussian2D(0.9, 520, 150, 3, 3)(x, y) +
        Gaussian2D(0.6, 200, 225, 3, 3)(x, y))
data += 0.01 * np.random.randn(100, 100)
#
# This is an example of some "noise"!
#
cosmic_ray_value = 0.997
data[100, 300:310] = cosmic_ray_value

import matplotlib.pyplot as plt
plt.imshow(data, origin='lower');
```

MIT OpenCourseWare

<https://ocw.mit.edu>

12.010 Computational Methods of Scientific Programming, Fall 2024

For more information about citing these materials or our Terms of Use, visit <https://ocw.mit.edu/terms>.