Python,
Machine Learning,
and Astronomy

Dr. Brian R. Kent, Director of Science Communications
National Radio Astronomy Observatory https://www.cv.nrao.edu/~bkent/
BlueSky, Twitter & Instagram: @VizAstro
YouTube: Visualize Astronomy

June 2025

Overview

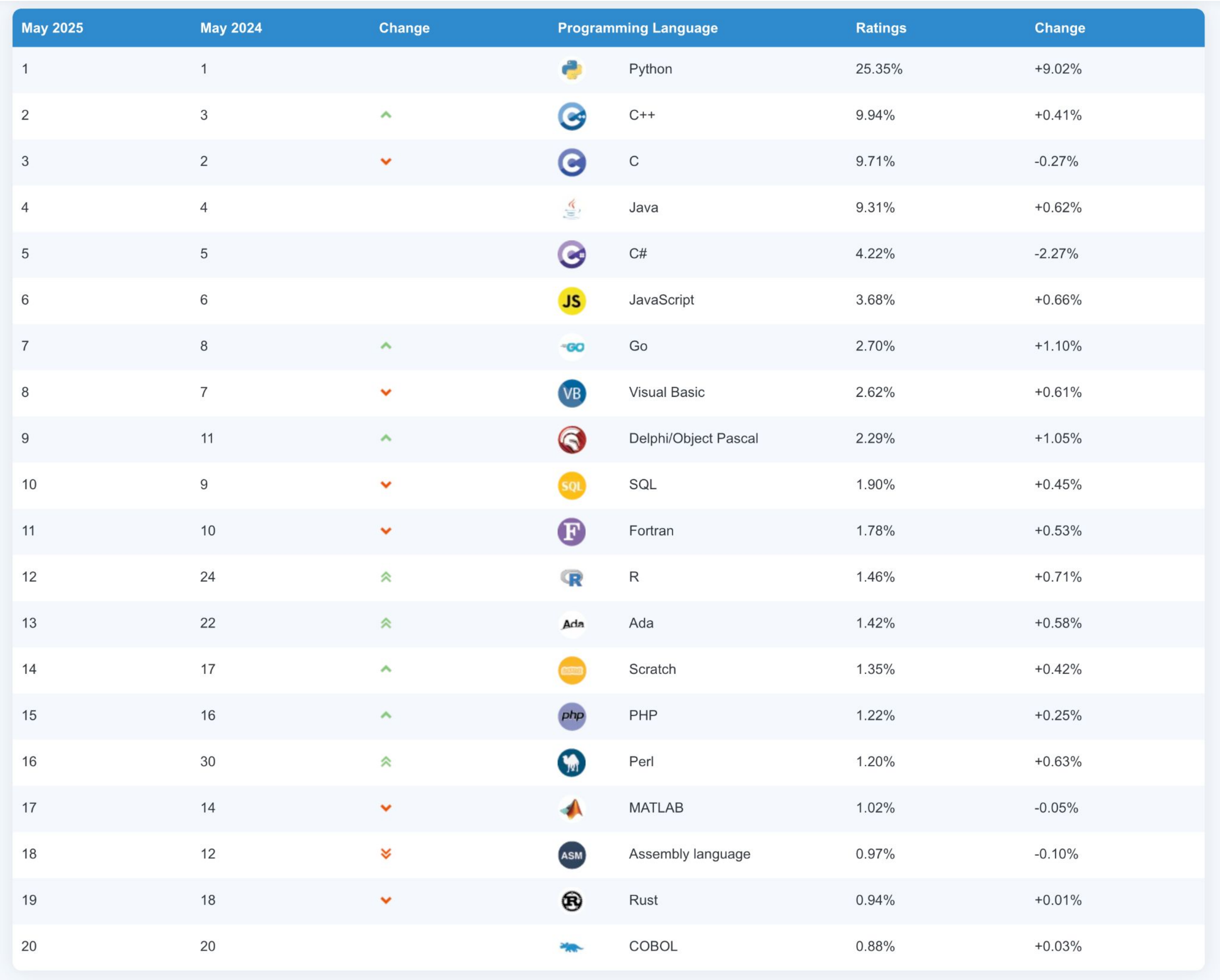
- History of Python
- Why use Python in Astronomy?
- Why Revision Control is important...
- Google Colab and simple AI examples
- Data Colorimetry and Presentation
- Machine Learning Example with Tensorflow
- Resources to explore further
- All the slides and notebooks will be shared
- ***Recommendations***

python

- High level programming language that is scriptable and supports many programming styles/paradigms.
- Created in the late 80s/early 90s
  by *Guido van Rossum*,
  named after *Monty Python*
- The **Python Standard Library**
  can be used to accomplish a wide
  variety of tasks - always check there first!

https://ivastar.github.io/python-novice-astro/

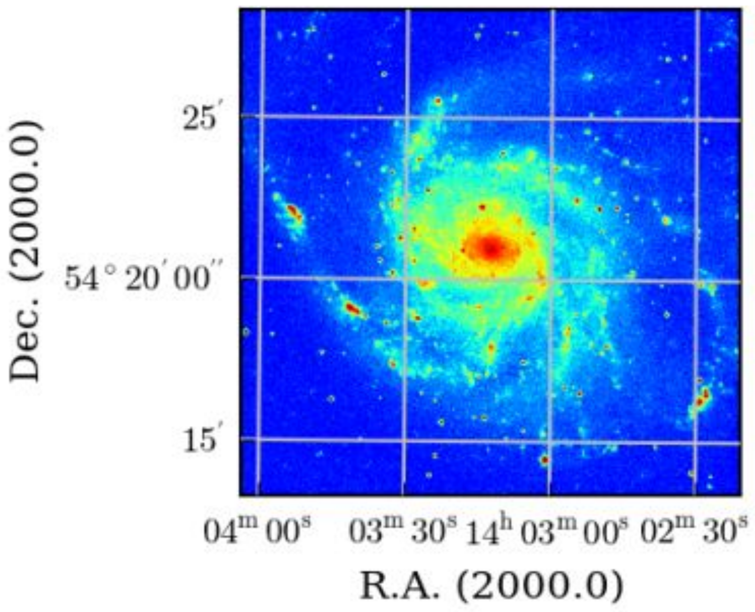
Tiobe index for June 2025



May 2025	May 2024	Change	Programming Language	Ratings	Change
1	1		Python	25.35%	+9.02%
2	3	↑	C++	9.94%	+0.41%
3	2	↓	C	9.71%	-0.27%
4	4		Java	9.31%	+0.62%
5	5		C#	4.22%	-2.27%
6	6		JavaScript	3.68%	+0.66%
7	8	↑	Go	2.70%	+1.10%
8	7	↓	Visual Basic	2.62%	+0.61%
9	11	↑	Delphi/Object Pascal	2.29%	+1.05%
10	9	↓	SQL	1.90%	+0.45%
11	10	↓	Fortran	1.78%	+0.53%
12	24	⇈	R	1.46%	+0.71%
13	22	⇈	Ada	1.42%	+0.58%
14	17	↑	Scratch	1.35%	+0.42%
15	16	↑	PHP	1.22%	+0.25%
16	30	⇈	Perl	1.20%	+0.63%
17	14	↓	MATLAB	1.02%	-0.05%
18	12	⇊	Assembly language	0.97%	-0.10%
19	18	↓	Rust	0.94%	+0.01%
20	20		COBOL	0.88%	+0.03%

Python and Astronomy

- With iPython, it is used as the command line shell & interpreter for CASA
- Used for the framework for VLA/GBT imaging/ALMA pipelines
- Can interface with AIPS via Obit
- Soon to be used - Dysh for GBT
- Used in PRESTO - pulsar data reduction
- Many modules and libraries available - numpy, matplotlib, Kapteyn, astropy, Pandas, AplPy, etc.
- Managed via pip or Anaconda
- Scriptable for astronomers OOP for engineers
- Use in environments like Colab and github.dev



Popular Python Resources

iPython - more user friendly shell

Astropy - great for data import, manipulation, catalog queries

AplPy - general image/coordinate display utility

Matplotlib - general purpose plotting tool

Scipy - numpy and fitting routines (some overlap with astropy…)

Kapteyn - Good for mapping projections

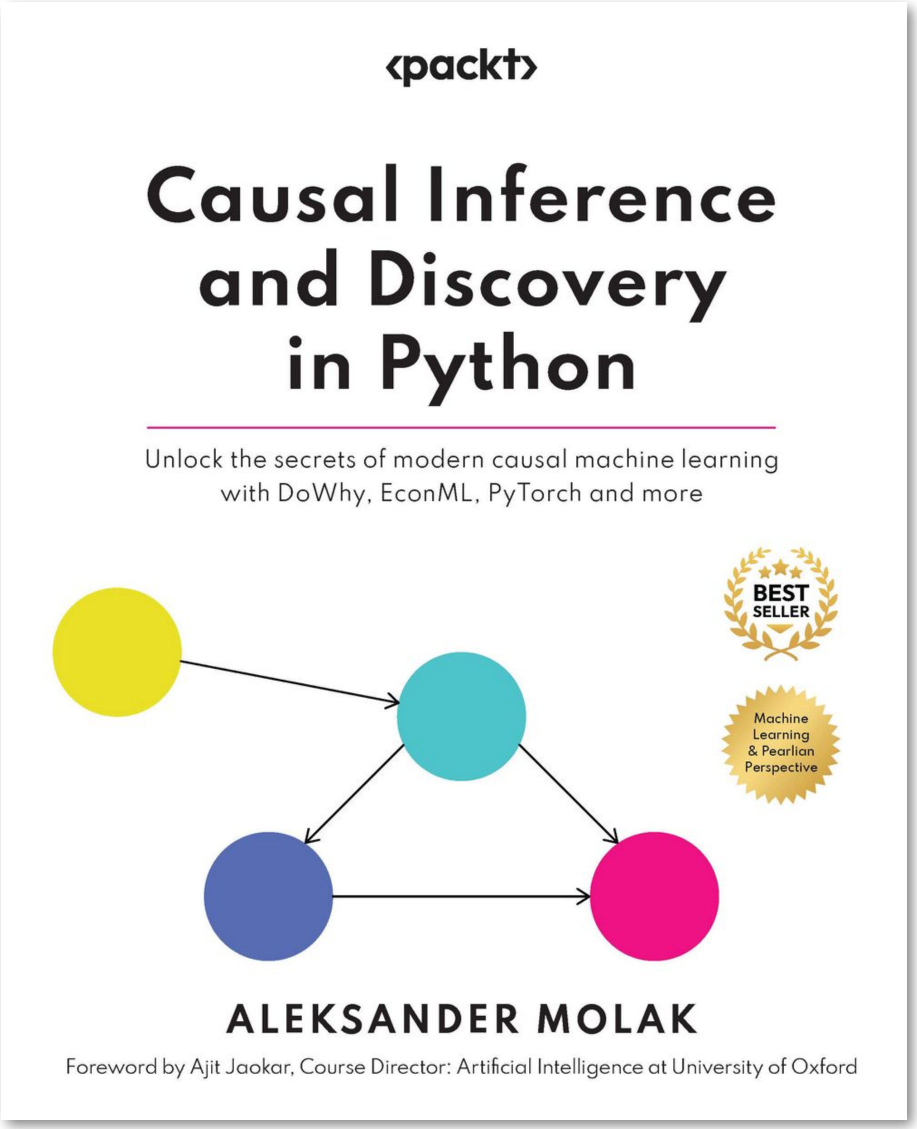
AstroML - Machine Learning http://www.astroml.org/index.html

PANDAS - Data Analysis Library http://pandas.pydata.org/

Python Data Science Handbook:

https://jakevdp.github.io/PythonDataScienceHandbook/

Recommendation: **Learn to use these inside and out!**



‹packt›

Causal Inference and Discovery in Python

Unlock the secrets of modern causal machine learning
with DoWhy, EconML, PyTorch and more

BEST SELLER

Machine Learning & Pearlian Perspective

ALEKSANDER MOLAK

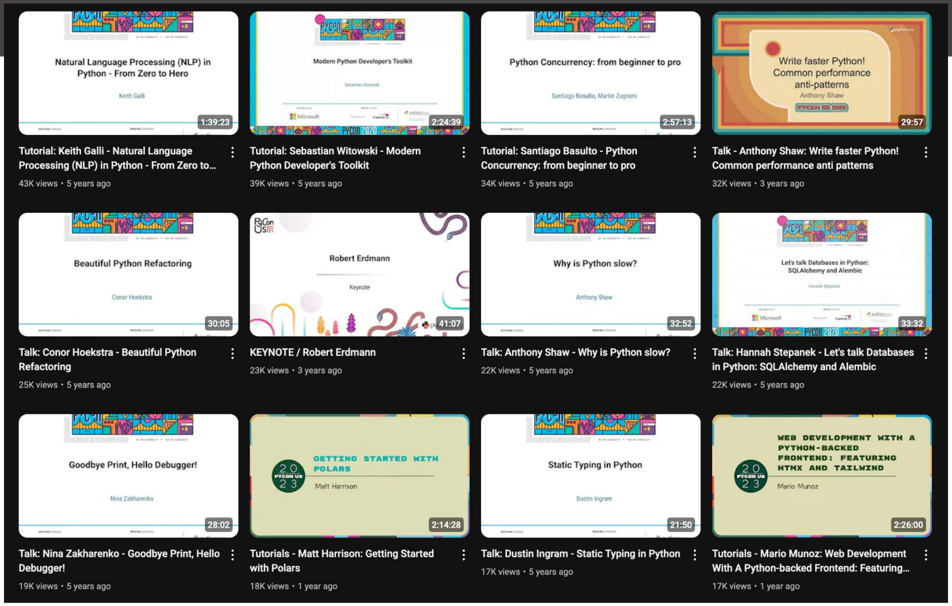
Foreword by Ajit Jaokar, Course Director: Artificial Intelligence at University of Oxford

https://github.com/PacktPublishing/Causal-Inference-and-Discovery-in-Python/tree/main

Where to start?

- https://www.youtube.com/@PyConUS/videos
- Pycon: https://us.pycon.org/2025 (and video tutorials)

PyConUS 2025


Natural Language Processing (NLP) in Python - From Zero to Hero
Keith Galli
1:39:23
Modern Python Developer's Toolkit
Sebastian Witowski
2:24:39
Python Concurrency: from beginner to pro
Santiago Basulto, Martin Zugnoni
2:57:13
Write faster Python! Common performance anti-patterns
Anthony Shaw
29:57
Beautiful Python Refactoring
Conor Hoekstra
30:05
Robert Erdmann
Keynote
41:07
Why is Python slow?
Anthony Shaw
32:52
Let's talk Databases in Python: SQLAlchemy and Alembic
Hannah Stepanek
33:32
Goodbye Print, Hello Debugger!
Nina Zakharenko
28:02
GETTING STARTED WITH POLARS
Matt Harrison
2:14:28
Static Typing in Python
Dustin Ingram
21:50
WEB DEVELOPMENT WITH A PYTHON-BACKED FRONTEND: FEATURING HTMX AND TAILWIND
Mario Munoz
2:26:00

Tutorial: Keith Galli - Natural Language Processing (NLP) in Python - From Zero to...
43K views • 5 years ago

Tutorial: Sebastian Witowski - Modern Python Developer's Toolkit
39K views • 5 years ago

Tutorial: Santiago Basulto - Python Concurrency: from beginner to pro
34K views • 5 years ago

Talk - Anthony Shaw: Write faster Python! Common performance anti patterns
32K views • 3 years ago

Talk: Conor Hoekstra - Beautiful Python Refactoring
25K views • 5 years ago

KEYNOTE / Robert Erdmann
23K views • 3 years ago

Talk: Anthony Shaw - Why is Python slow?
22K views • 5 years ago

Talk: Hannah Stepanek - Let's talk Databases in Python: SQLAlchemy and Alembic
22K views • 5 years ago

Talk: Nina Zakharenko - Goodbye Print, Hello Debugger!
19K views • 5 years ago

Tutorials - Matt Harrison: Getting Started with Polars
18K views • 1 year ago

Talk: Dustin Ingram - Static Typing in Python
17K views • 5 years ago

Tutorials - Mario Munoz: Web Development With A Python-backed Frontend: Featuring...
17K views • 1 year ago

Revision Control

Revision Control for Software

Popular revision control systems (RCS) include SVN, CVS, Mercurial, and **git**.

Tracks changes made to files - good for collaboration among teams or to see when and where changes were made to your code!

Repo for this presentation:

https://github.com/brkent/SummerStudent2025

1. Plotting
2. 2D Plotting
3. HR Diagram
4. Cosmology
5. Fitting Function
6. FITS images
7. VLASS and subimages
8. Image comparison
9. Contours
10. Neural Networks

git revision control

```
mkdir sumstudent

cd sumstudent

git clone https://github.com/brkent/SummerStudent2025.git
```

This will create and link your directory to the git repository

git revision control

```
#Create a readme file
touch README.md
git add README.md
git commit -m “Initial README commit”
git push -u origin main
```

This is your modern scientific lab notebook!

Recommendation:

If you do not already know how to use git, please take some time this summer and practice with these tutorials:

[**https://docs.github.com/en/get-started**](https://docs.github.com/en/get-started)

Google Colab

Colaboratory is a **Google** research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud. Now interfaces with Gemini…

https://github.com/brkent/SummerStudent2025/blob/main/example05_gaussian_fit_seaborn.ipynb

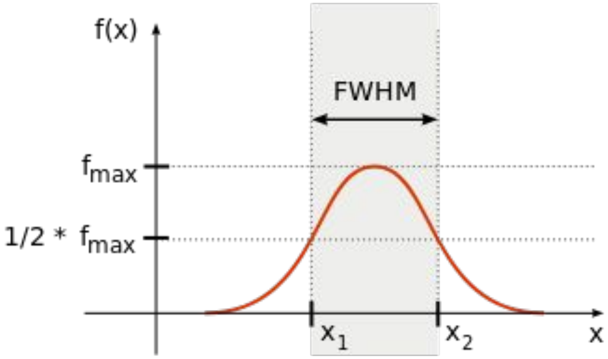
Also works in github.dev…

Recommendation: Learn to prototype bit of code in the cloud and make **clean** Jupyter notebooks. Make your work portable and shareable! If you can point potential employers at your work and it is organized/documented - all the better!

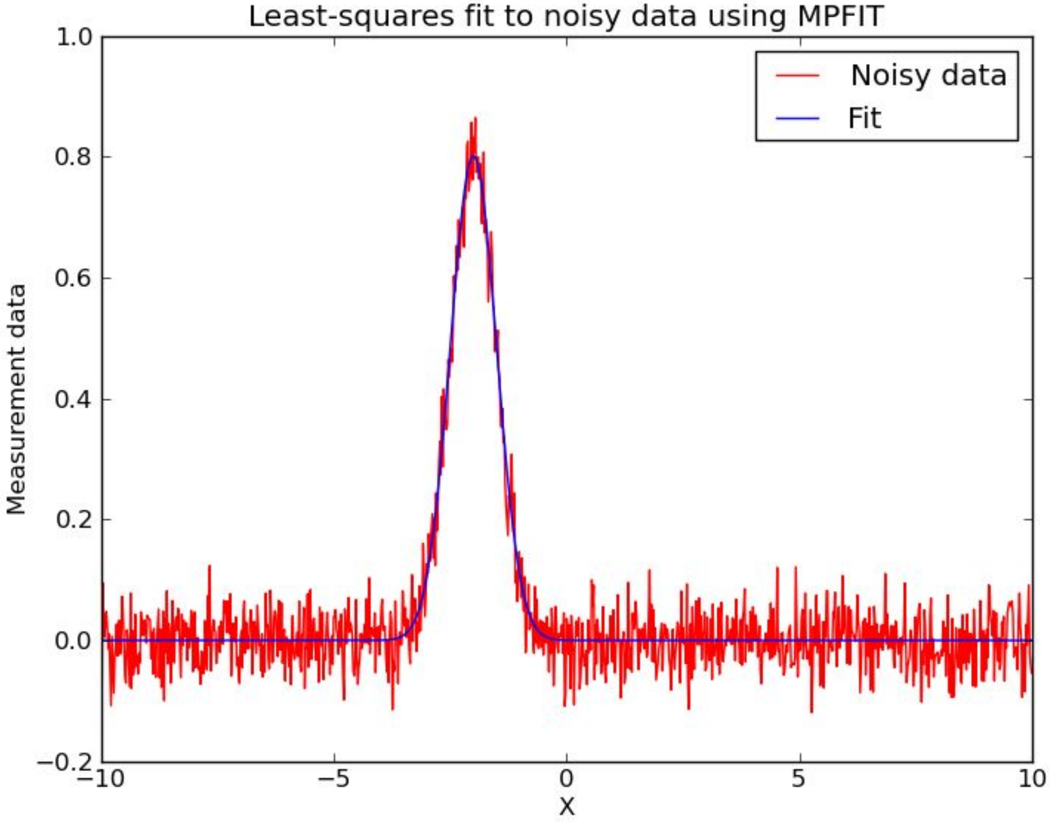
Example: Create a noisy spectrum and fit a Gaussian

https://github.com/brkent/SummerStudent2025/blob/main/example05_gaussian_fit_seaborn.ipynb

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]$$



$$\mathrm{FWHM} = 2\sqrt{2\ln 2}\,\sigma \approx 2.355\,\sigma.$$

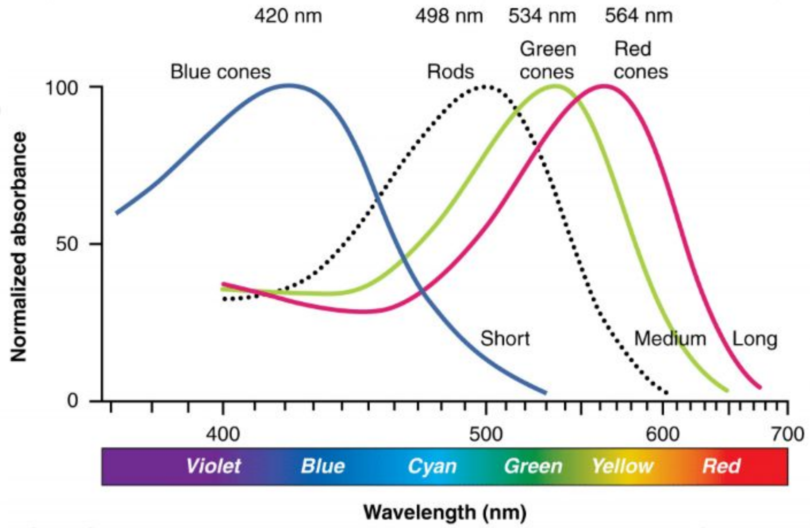


Another series of great fitting examples...

https://learn.astropy.org/tutorials/Models-Quick-Fit.html

Colorimetry for Visualization

Want your plots
to look top notch?



seaborn

https://seaborn.pydata.org/

https://seaborn.pydata.org/tutorial/color_palettes.html#general-principles-for-using-color-in-plots

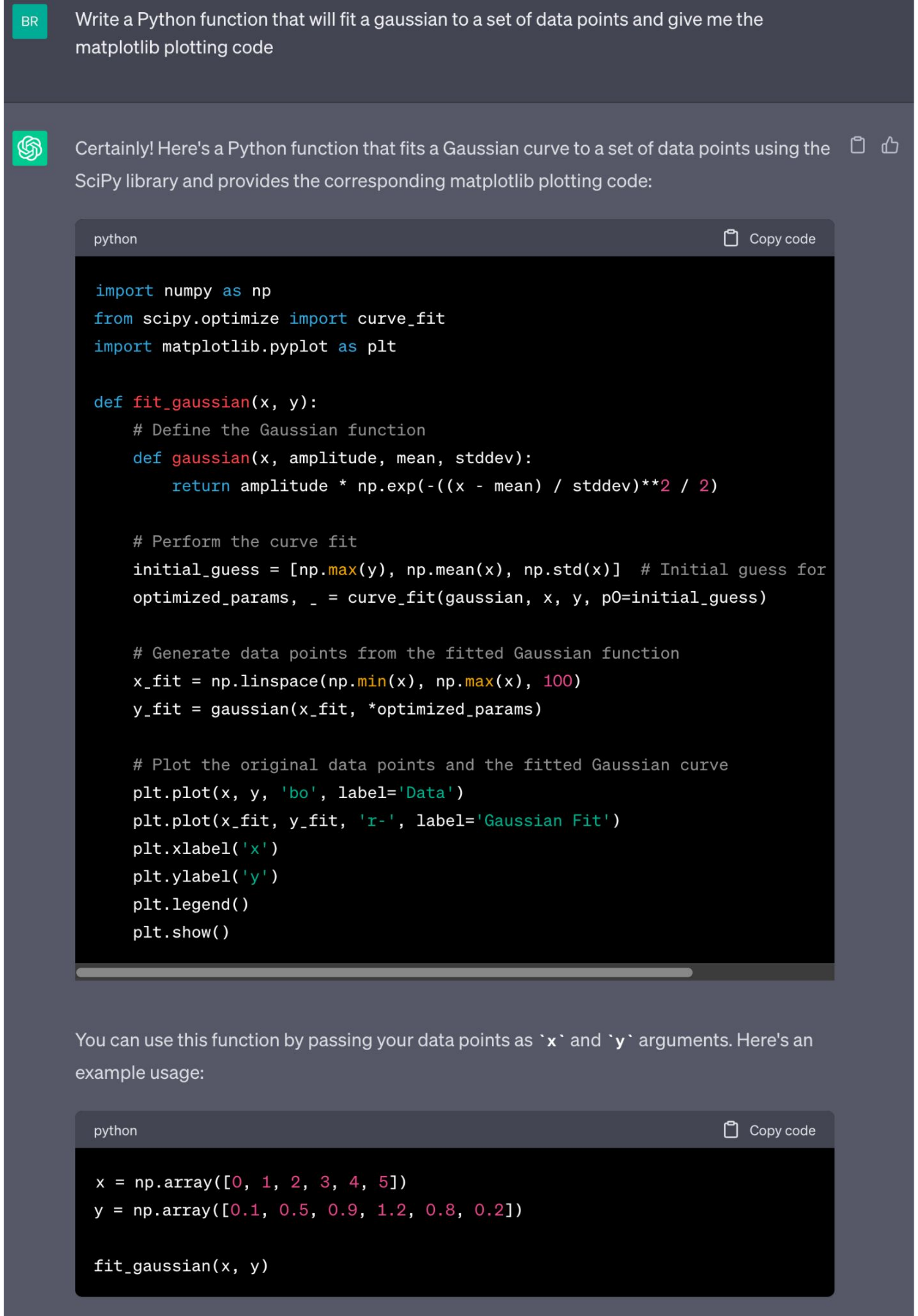
See references at the bottom of this page:
https://medium.com/hipster-color-science/a-beginners-guide-to-colorimetry-401f1830b65a

Talk by Dr. Michael Waters:
https://www.dropbox.com/s/7s9seplrnw3ea7p/Practical_Colorimetry_for_Scientific_Visualization_-_Michael_J_Waters_-_2021_3_14.pdf

Recommendation: Make your plots and graphics stand out for publication and presentation!

Examples

What if we had tools to prototype and make our workflow a bit faster?



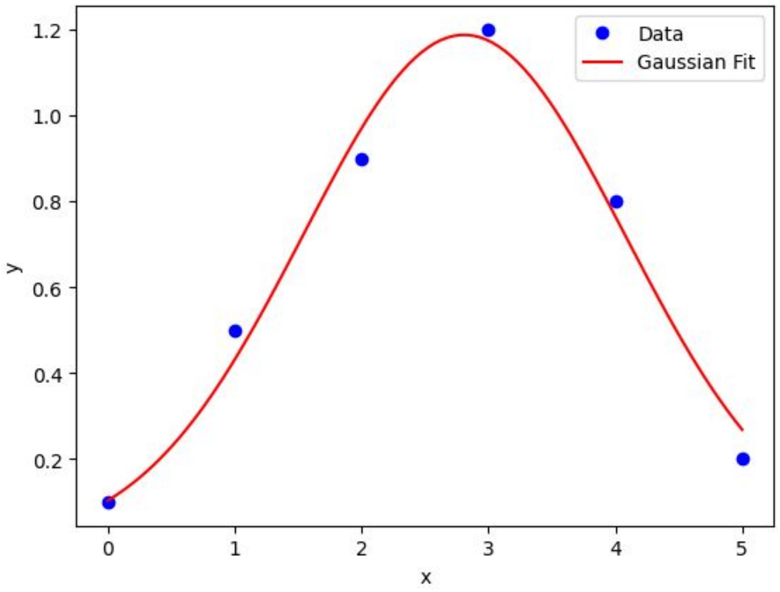
Write a Python function that will fit a gaussian to a set of data points and give me the matplotlib plotting code

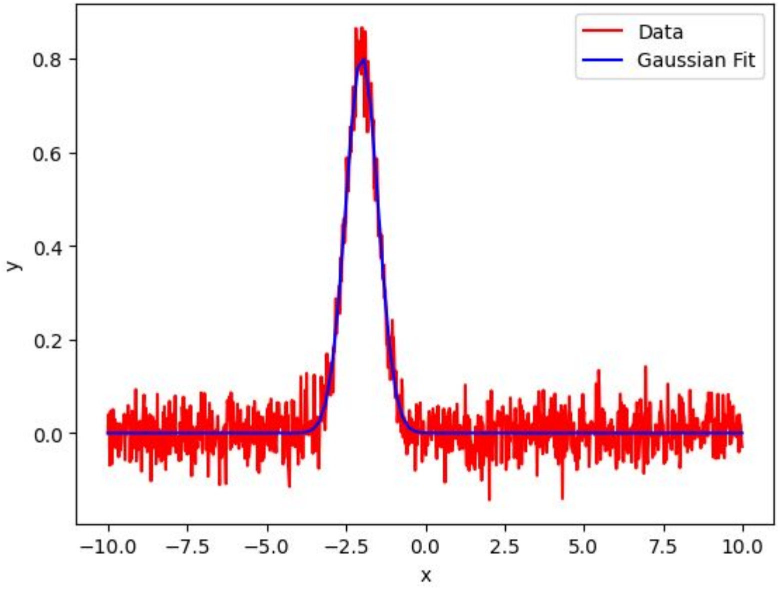
Certainly! Here's a Python function that fits a Gaussian curve to a set of data points using the SciPy library and provides the corresponding matplotlib plotting code:

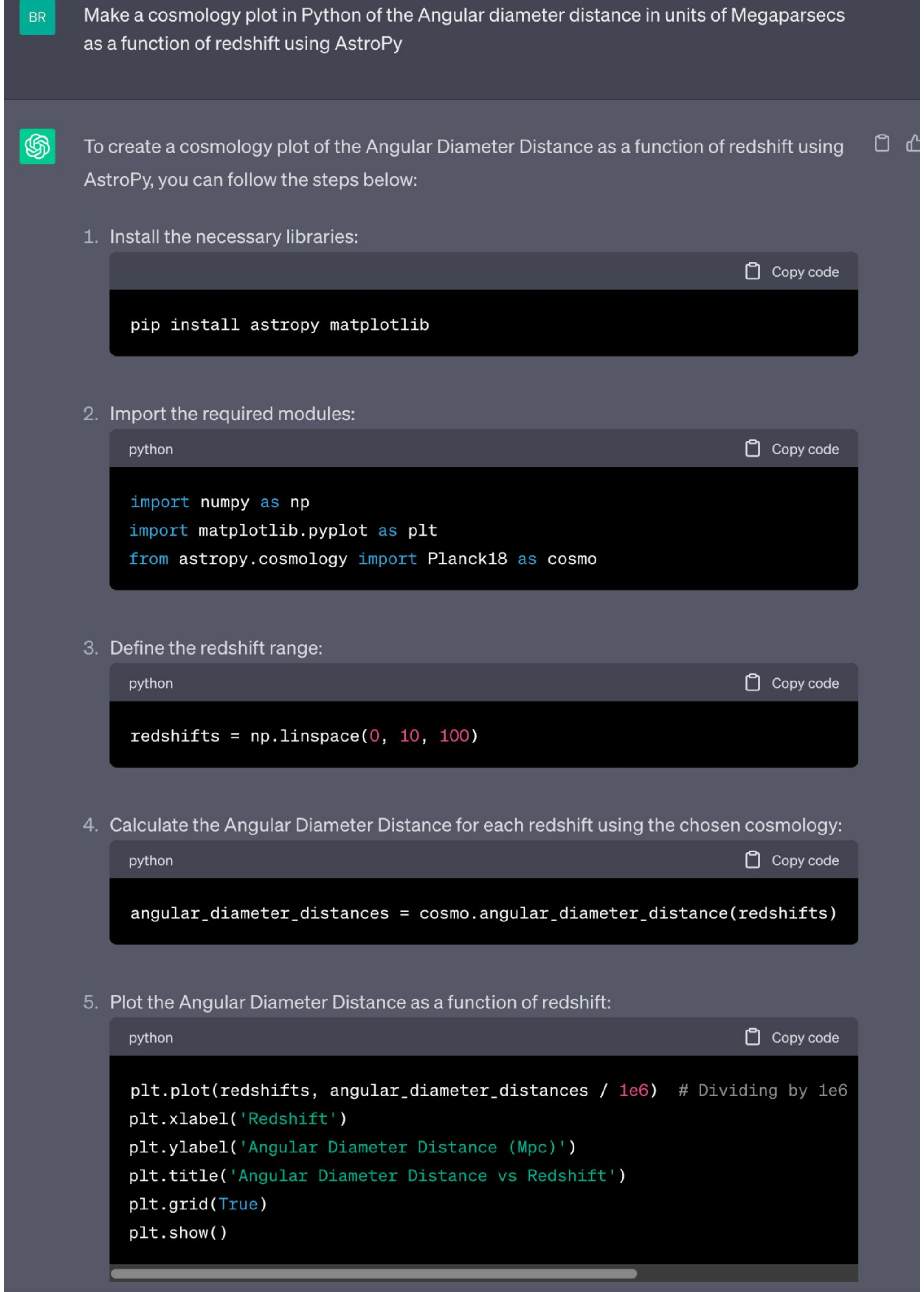
```python
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def fit_gaussian(x, y):
    # Define the Gaussian function
    def gaussian(x, amplitude, mean, stddev):
        return amplitude * np.exp(-((x - mean) / stddev)**2 / 2)

    # Perform the curve fit
    initial_guess = [np.max(y), np.mean(x), np.std(x)]  # Initial guess for
    optimized_params, _ = curve_fit(gaussian, x, y, p0=initial_guess)

    # Generate data points from the fitted Gaussian function
    x_fit = np.linspace(np.min(x), np.max(x), 100)
    y_fit = gaussian(x_fit, *optimized_params)

    # Plot the original data points and the fitted Gaussian curve
    plt.plot(x, y, 'bo', label='Data')
    plt.plot(x_fit, y_fit, 'r-', label='Gaussian Fit')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.show()
```

You can use this function by passing your data points as `x` and `y` arguments. Here's an example usage:

```python
x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([0.1, 0.5, 0.9, 1.2, 0.8, 0.2])

fit_gaussian(x, y)
```







BR

Make a cosmology plot in Python of the Angular diameter distance in units of Megaparsecs as a function of redshift using AstroPy

To create a cosmology plot of the Angular Diameter Distance as a function of redshift using AstroPy, you can follow the steps below:

1. Install the necessary libraries:

```
pip install astropy matplotlib
```

2. Import the required modules:

```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.cosmology import Planck18 as cosmo
```

3. Define the redshift range:

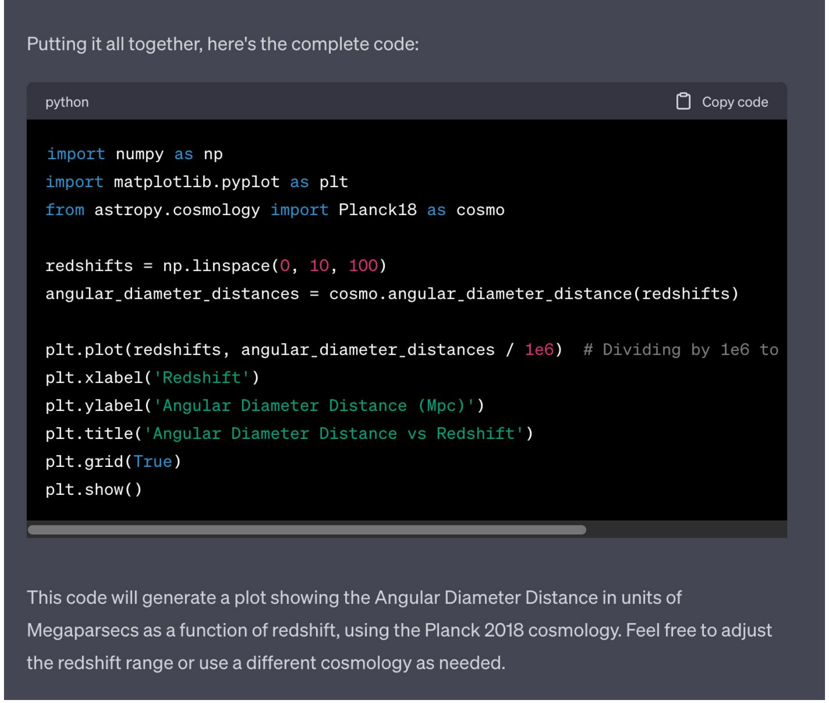
```python
redshifts = np.linspace(0, 10, 100)
```

4. Calculate the Angular Diameter Distance for each redshift using the chosen cosmology:

```python
angular_diameter_distances = cosmo.angular_diameter_distance(redshifts)
```

5. Plot the Angular Diameter Distance as a function of redshift:

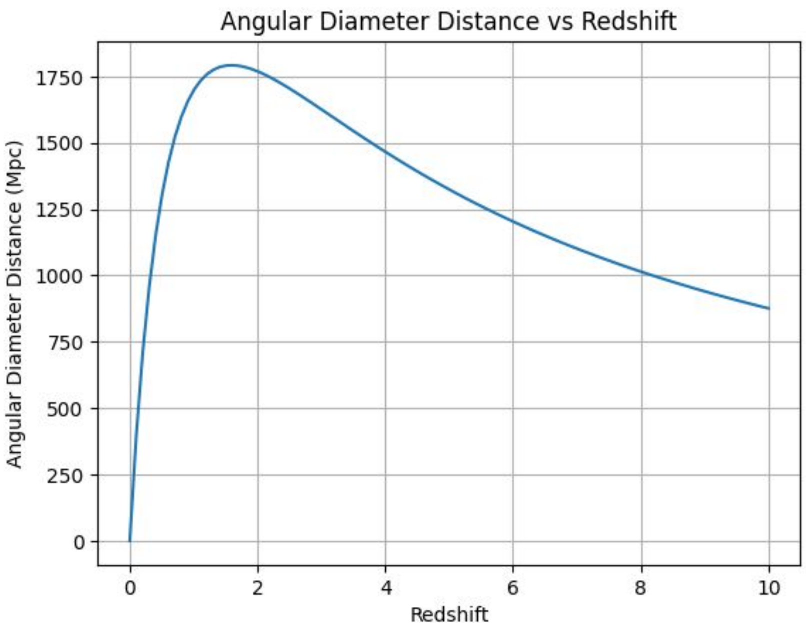
```python
plt.plot(redshifts, angular_diameter_distances / 1e6)  # Dividing by 1e6
plt.xlabel('Redshift')
plt.ylabel('Angular Diameter Distance (Mpc)')
plt.title('Angular Diameter Distance vs Redshift')
plt.grid(True)
plt.show()
```



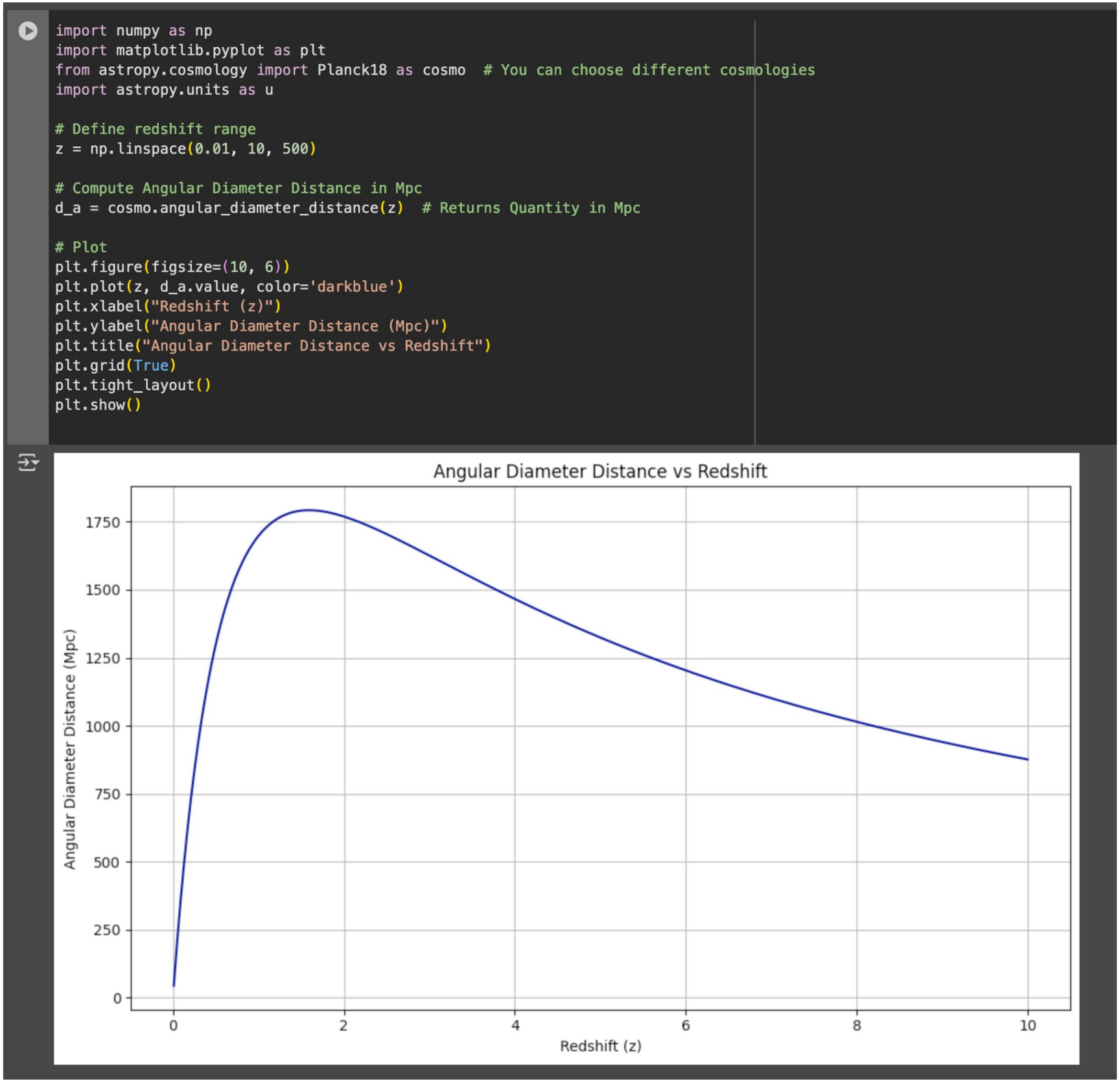
Putting it all together, here's the complete code:

```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.cosmology import Planck18 as cosmo

redshifts = np.linspace(0, 10, 100)
angular_diameter_distances = cosmo.angular_diameter_distance(redshifts)

plt.plot(redshifts, angular_diameter_distances / 1e6)  # Dividing by 1e6 to
plt.xlabel('Redshift')
plt.ylabel('Angular Diameter Distance (Mpc)')
plt.title('Angular Diameter Distance vs Redshift')
plt.grid(True)
plt.show()
```

This code will generate a plot showing the Angular Diameter Distance in units of Megaparsecs as a function of redshift, using the Planck 2018 cosmology. Feel free to adjust the redshift range or use a different cosmology as needed.

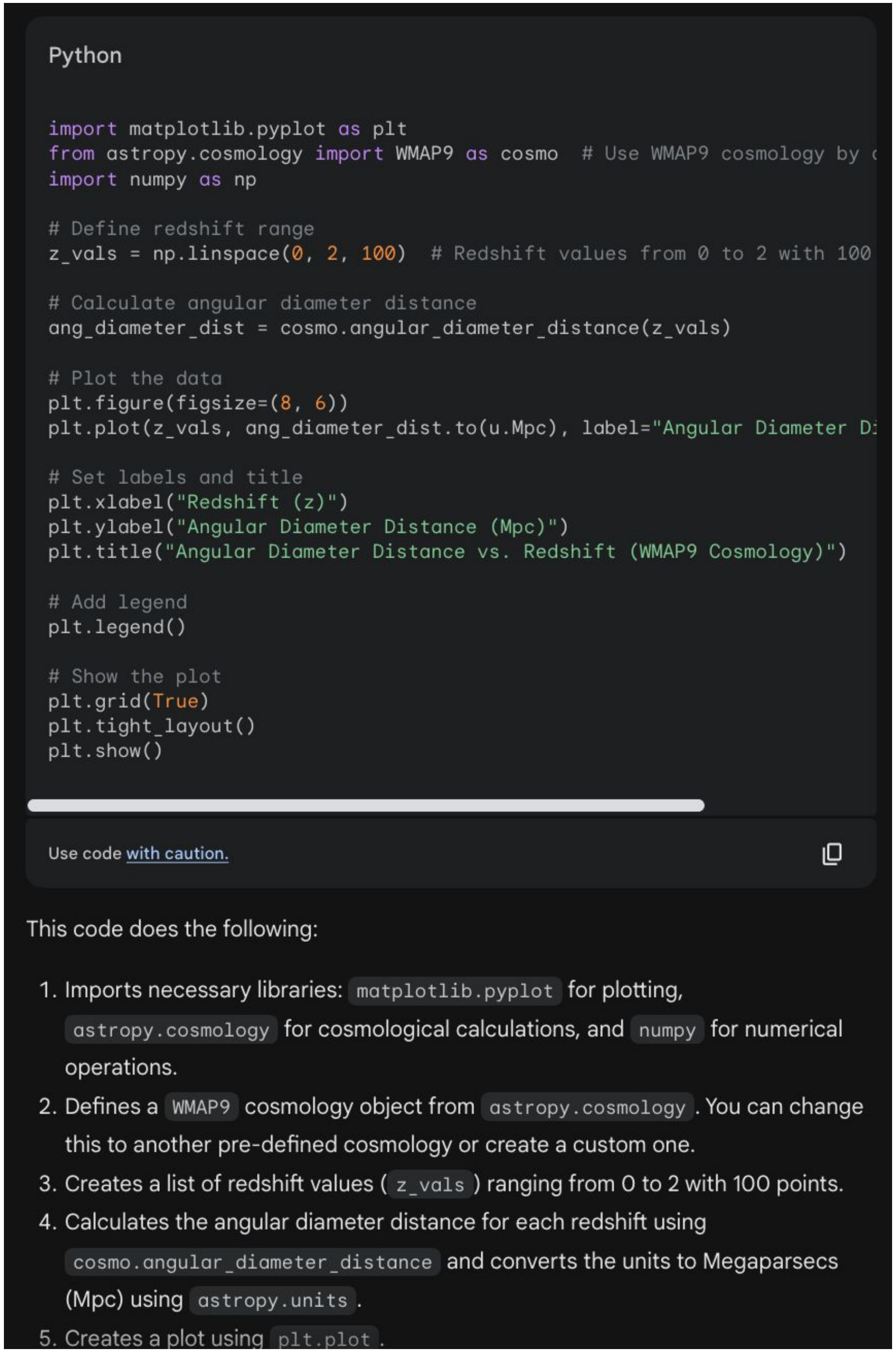


*Had to fix the units with ChatGPT…but it was close!



```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.cosmology import Planck18 as cosmo  # You can choose different cosmologies
import astropy.units as u

# Define redshift range
z = np.linspace(0.01, 10, 500)

# Compute Angular Diameter Distance in Mpc
d_a = cosmo.angular_diameter_distance(z)  # Returns Quantity in Mpc

# Plot
plt.figure(figsize=(10, 6))
plt.plot(z, d_a.value, color='darkblue')
plt.xlabel("Redshift (z)")
plt.ylabel("Angular Diameter Distance (Mpc)")
plt.title("Angular Diameter Distance vs Redshift")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Angular Diameter Distance vs Redshift
Angular Diameter Distance (Mpc)
1750
1500
1250
1000
750
500
250
0
0
2
4
6
8
10
Redshift (z)



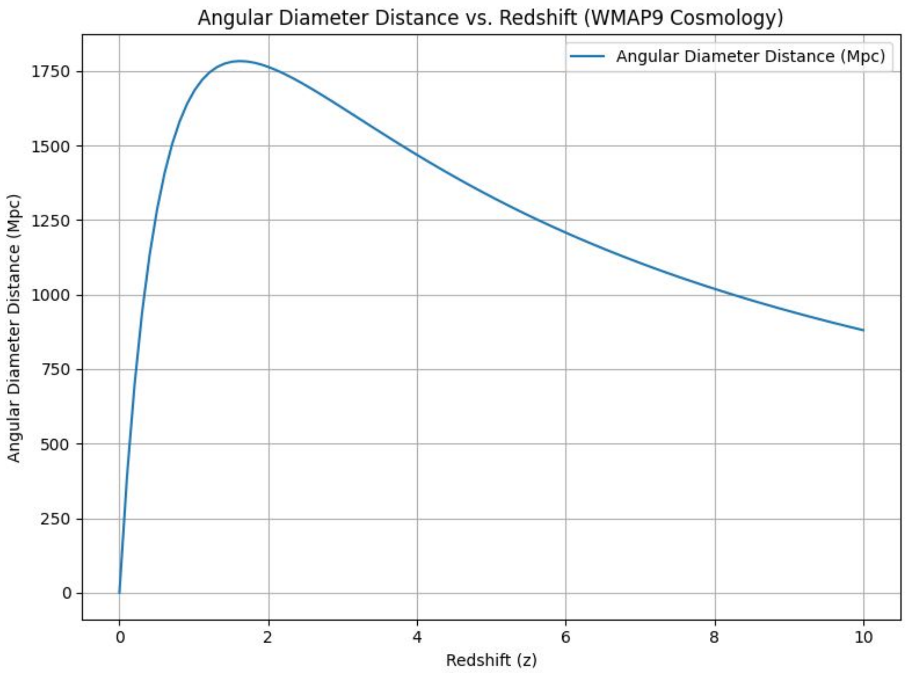
Python

```
import matplotlib.pyplot as plt
from astropy.cosmology import WMAP9 as cosmo  # Use WMAP9 cosmology by
import numpy as np

# Define redshift range
z_vals = np.linspace(0, 2, 100)  # Redshift values from 0 to 2 with 100

# Calculate angular diameter distance
ang_diameter_dist = cosmo.angular_diameter_distance(z_vals)

# Plot the data
plt.figure(figsize=(8, 6))
plt.plot(z_vals, ang_diameter_dist.to(u.Mpc), label="Angular Diameter Di

# Set labels and title
plt.xlabel("Redshift (z)")
plt.ylabel("Angular Diameter Distance (Mpc)")
plt.title("Angular Diameter Distance vs. Redshift (WMAP9 Cosmology)")

# Add legend
plt.legend()

# Show the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```

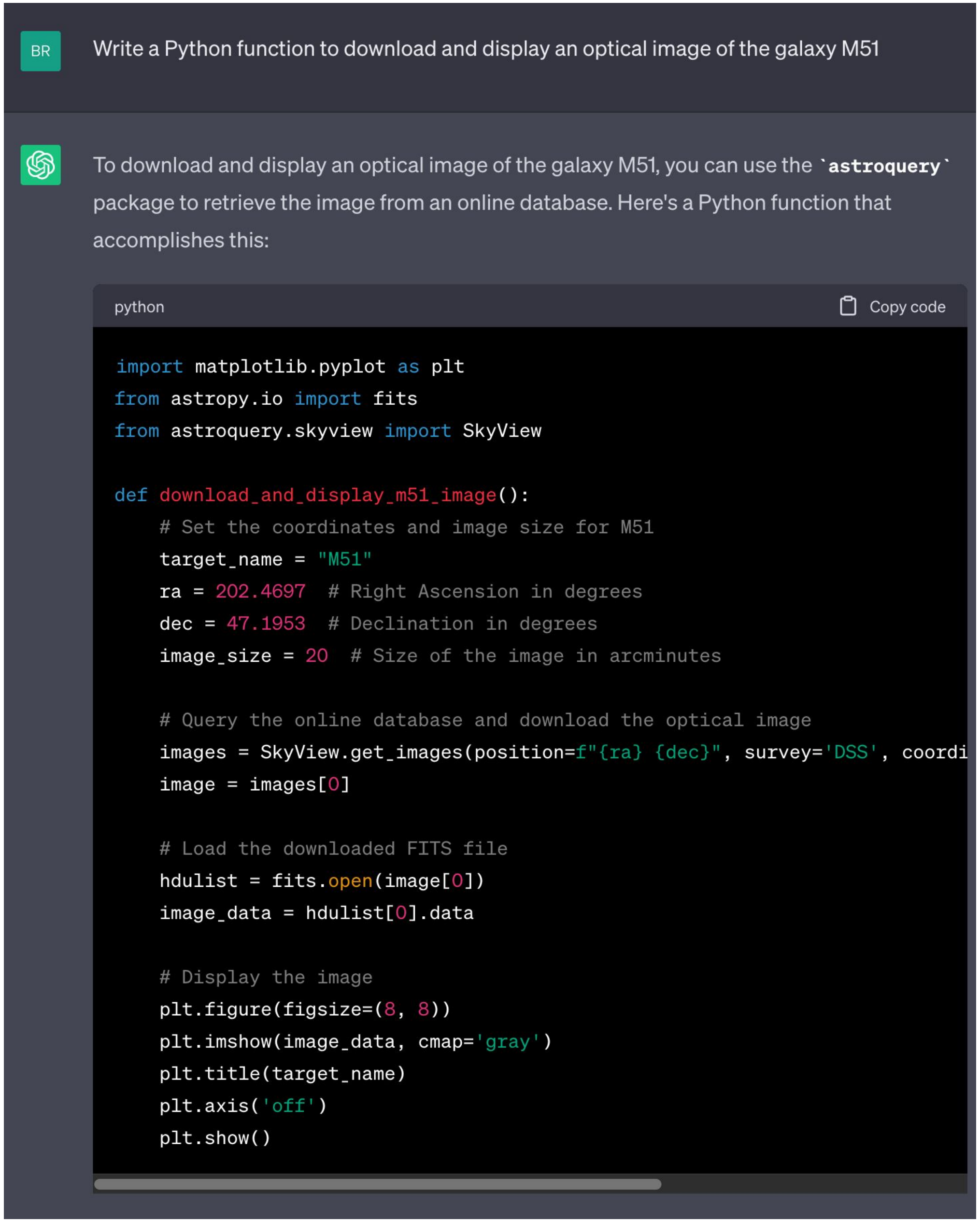
Use code with caution.

This code does the following:

1. Imports necessary libraries: `matplotlib.pyplot` for plotting, `astropy.cosmology` for cosmological calculations, and `numpy` for numerical operations.
2. Defines a `WMAP9` cosmology object from `astropy.cosmology`. You can change this to another pre-defined cosmology or create a custom one.
3. Creates a list of redshift values (`z_vals`) ranging from 0 to 2 with 100 points.
4. Calculates the angular diameter distance for each redshift using `cosmo.angular_diameter_distance` and converts the units to Megaparsecs (Mpc) using `astropy.units`.
5. Creates a plot using `plt.plot`.



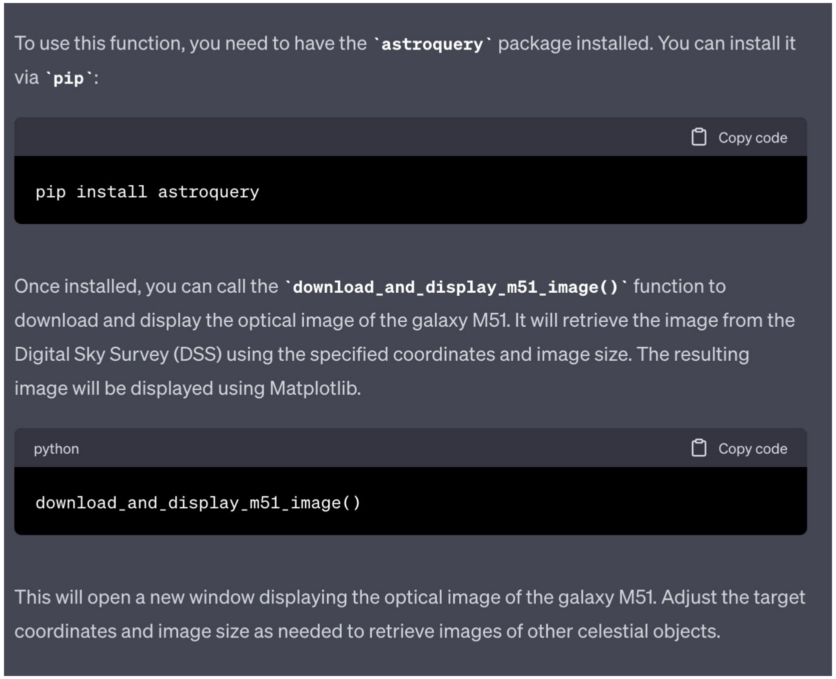
*Google Gemini was close - did not import the units from AstroPy, and only went to a redshift of 2.



Write a Python function to download and display an optical image of the galaxy M51

To download and display an optical image of the galaxy M51, you can use the `astroquery` package to retrieve the image from an online database. Here's a Python function that accomplishes this:

```python
import matplotlib.pyplot as plt
from astropy.io import fits
from astroquery.skyview import SkyView

def download_and_display_m51_image():
    # Set the coordinates and image size for M51
    target_name = "M51"
    ra = 202.4697  # Right Ascension in degrees
    dec = 47.1953  # Declination in degrees
    image_size = 20  # Size of the image in arcminutes

    # Query the online database and download the optical image
    images = SkyView.get_images(position=f"{ra} {dec}", survey='DSS', coordi
    image = images[0]

    # Load the downloaded FITS file
    hdulist = fits.open(image[0])
    image_data = hdulist[0].data

    # Display the image
    plt.figure(figsize=(8, 8))
    plt.imshow(image_data, cmap='gray')
    plt.title(target_name)
    plt.axis('off')
    plt.show()
```



To use this function, you need to have the `astroquery` package installed. You can install it via `pip`:

```
pip install astroquery
```

Once installed, you can call the `download_and_display_m51_image()` function to download and display the optical image of the galaxy M51. It will retrieve the image from the Digital Sky Survey (DSS) using the specified coordinates and image size. The resulting image will be displayed using Matplotlib.

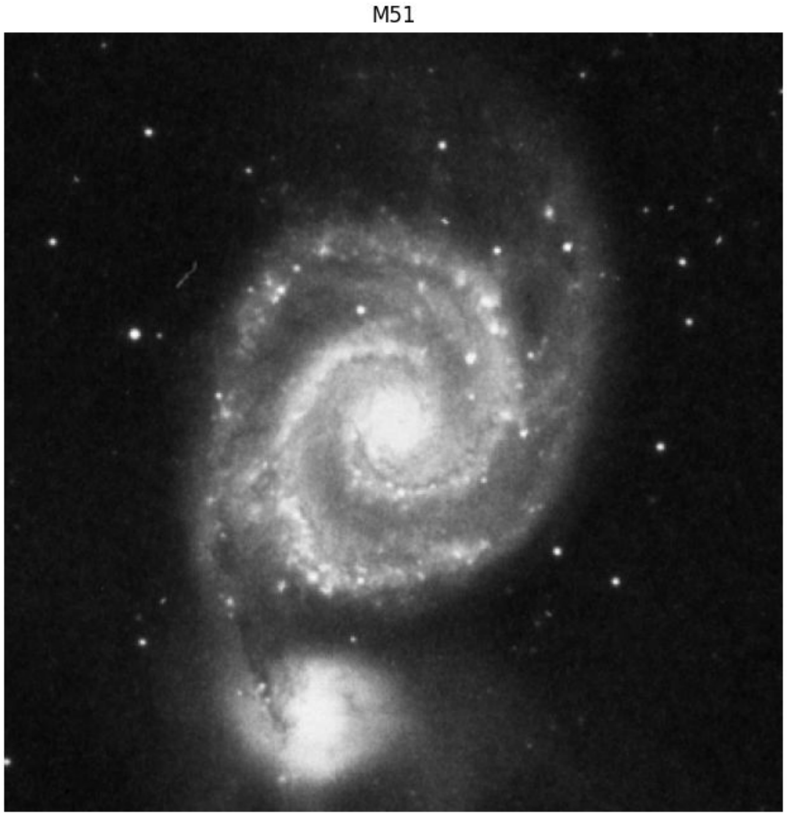
```python
download_and_display_m51_image()
```

This will open a new window displaying the optical image of the galaxy M51. Adjust the target coordinates and image size as needed to retrieve images of other celestial objects.

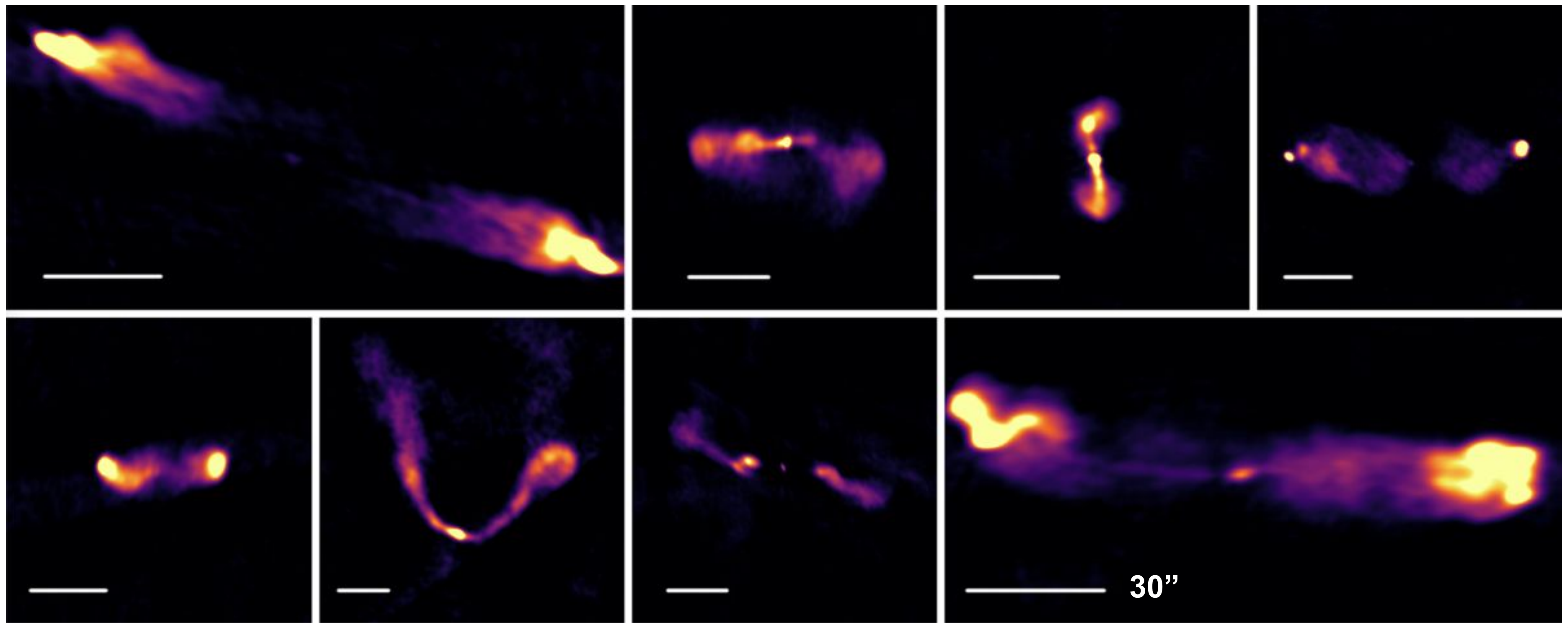
There are some issues with this one - but they are fixable…

Query for an image…

```
!pip install astroquery


import matplotlib.pyplot as plt
from astropy.io import fits
from astroquery.skyview import SkyView


def download_and_display_m51_image():
   # Set the coordinates and image size for M51
   target_name = "M51"
   ra = 202.4697  # Right Ascension in degrees
   dec = 47.1953  # Declination in degrees
   image_size = 400  # Size of the image in pixels


   # Query the online database and download the optical image
   images = SkyView.get_images(position=f"{ra} {dec}",
                survey='DSS',
                coordinates='J2000', pixels=image_size)
   image = images[0]


   # Load the downloaded FITS file
   hdulist = image[0]
   image_data = hdulist.data


   # Display the image
   plt.figure(figsize=(8, 8))
   plt.imshow(image_data, cmap='gray')
   plt.title(target_name)
   plt.axis('off')
   plt.show()
```



TensorFlow Example: VLASS Galaxy Classification



https://github.com/brkent/SummerStudent2025

See the paper by Lacy et al. 2020 https://ui.adsabs.harvard.edu/abs/2020PASP..132c5001L/abstract

Over 600 citations so far. 2641 refereed authors and counting…

Machine Learning/Deep Learning Neural Networks/Artificial Intelligence

NVidia Deep Learning:

https://www.nvidia.com/en-us/deep-learning-ai/education/

PyTorch (Meta/Facebook):
https://pytorch.org/

Tensorflow (Google):

https://www.tensorflow.org/

Check out the NVidia GTC: https://www.nvidia.com/en-us/gtc/

Recommendation: *Please subscribe and sign up for these free tutorials and resources!*