# Introduction to Python 2.7

Sarrvesh S. Sridhar

ASTRON

June 18, 2018

# Course objectives

- Learn the basic building blocks of python
- Learn to scripting in python
- What are python packages?
- Astronomical packages

- **Disclosure:** This is not a comprehensive tour of Python 2.7. I will cover only what is necessary/useful for this workshop. See slide on "Further reading" for references.

ASTRON

# What is python?

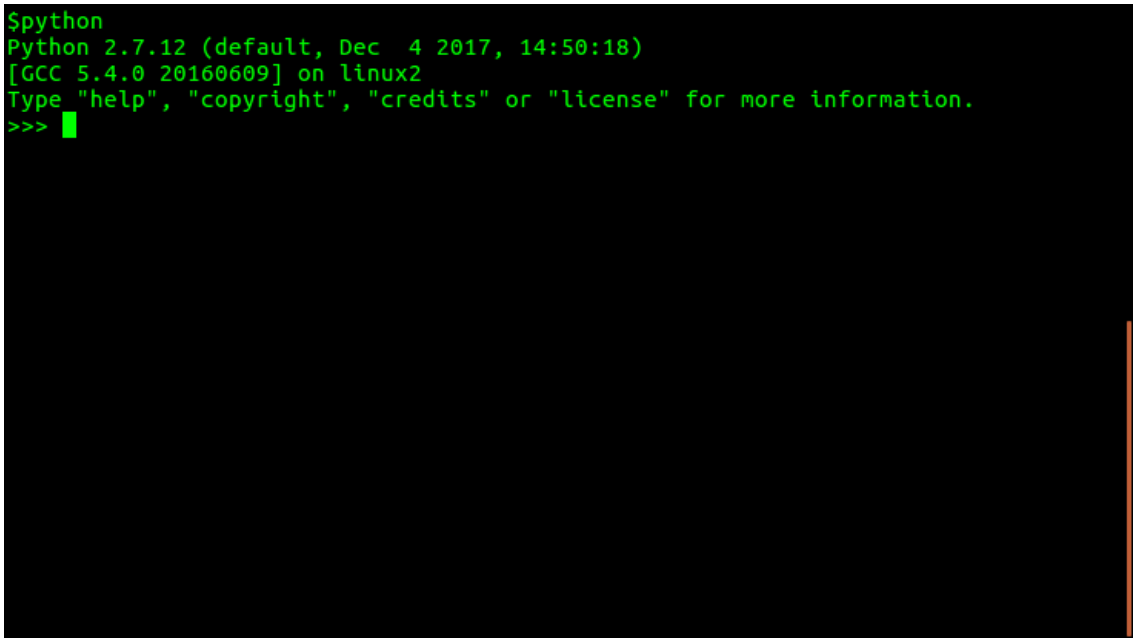
- Programming language
  - general purpose
  - interpreted
  - high-level
  - Created by Guido van Rossum almost 30 years ago

  - Widely used in astronomy
  - Numerous packages exist
    - ★ AstroPy - Common astronomy utilities
    - ★ SunPy - Solar data analysis
    - ★ GammaPy - Gamma-ray astronomy

ASTRON

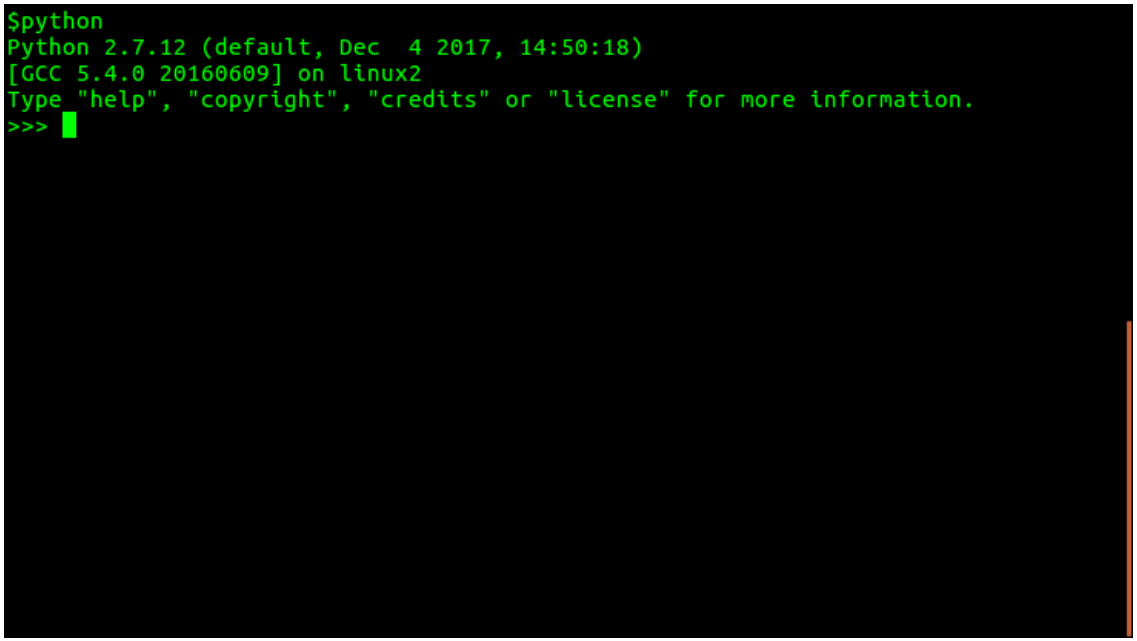
# Python interpreter

- Start the interpreter by typing **python** in your command prompt



# Python interpreter

- Start the interpreter by typing **python** in your command prompt




- Type the following commands in the prompt
  - a = 10
  - b = 20.5
  - c = a+b
  - type(a); type(b); type(c)
- Try out other basic math operations

# Scripting in python

- Create a new file called **mycode.py**
- Add the following code to the file:

```
a = 10
b = 20.5
# Add two numbers
c = a+b
# Now, print the number
print c
```

- Save the file and execute it as **python mycode.py**

ASTRON

# Scripting in python

- Create a new file called **mycode.py**
- Add the following code to the file:

```
a = 10
b = 20.5
# Add two numbers
c = a+b
# Now, print the number
print c
```

- Save the file and execute it as **python mycode.py**
- Or, you can add **#!/usr/bin/env python** as the first line in mycode.py
- Change the file permission with **chmod u+x mycode.py**
- Finally, execute as **./mycode.py**

ASTRON

# Strings in python

- Strings, or a set of characters represented inside ' ', " ", or """ """.
- Create a string with

```
mystring = "Hello, Python!"
```

- **print mystring** will print the entire string.

# Strings in python

- Strings, or a set of characters represented inside ' ', " ", or """ """.
- Create a string with

```
mystring = "Hello, Python!"
```

- **print mystring** will print the entire string.
- **print mystring[2:10]** will print 'llo, Pyt'.

# Strings in python

- Strings, or a set of characters represented inside ' ', " ", or """ """.
- Create a string with

```
mystring = "Hello, Python!"
```

- **print mystring** will print the entire string.
- **print mystring[2:10]** will print 'llo, Pyt'.
- **print mystring[-1]** will print '!'.

# Strings in python

- Strings, or a set of characters represented inside ' ', " ", or """ """.
- Create a string with

```
mystring = "Hello, Python!"
```

- **print mystring** will print the entire string.
- **print mystring[2:10]** will print 'llo, Pyt'.
- **print mystring[-1]** will print '!'.

## Exercise:

- Can you print just 'Hello'?
- Can you print the string in reverse? That is, display '!nohtyP ,olleH'.

# Lists

- Lists are pretty useful components of Python
- It is a collection of items inside [] and separated by commas.
- Example:

```
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]
print list1[1]
print list1[2:4]
print list1 + list2
```

ASTRON

# Lists

- Lists are pretty useful components of Python
- It is a collection of items inside [] and separated by commas.
- Example:

```
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]
print list1[1]
print list1[2:4]
print list1 + list2
```

- You can add more items with **list1.append(6)**

ASTRON

# Lists

- Lists are pretty useful components of Python
- It is a collection of items inside [] and separated by commas.
- Example:

```
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]
print list1[1]
print list1[2:4]
print list1 + list2
```

- You can add more items with **list1.append(6)**
- Lists can contain items of different data types

```
list3 = [1, 2.5, 'three']
```

# Lists

- Lists are pretty useful components of Python
- It is a collection of items inside [] and separated by commas.
- Example:

```
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]
print list1[1]
print list1[2:4]
print list1 + list2
```

- You can add more items with **list1.append(6)**
- Lists can contain items of different data types

```
list3 = [1, 2.5, 'three']
```

## Exercise:

- Can you display the reverse of list3? That is ['three', 2.5, 1]
- Can you display the length/size of list3?

# Further reading

- There is more to python than what we just saw
- ... but this should get you started.

- `https://www.python.org/about/gettingstarted/`
- `https://www.learnpython.org/`
- `https://www.tutorialspoint.com/python/index.htm`
- ...

- Or, just google your problem!

ASTRON

# Python packages

- Support functionalities beyond the standard python library.
- For example: you cannot plot using standard python. Use matplotlib.

ASTRON

# Python packages

- Support functionalities beyond the standard python library.
- For example: you cannot plot using standard python. Use matplotlib.
- You need to install the libraries separately.
- We'll use **pip** to install some useful libraries.
- pip install "name" - -user
  - "name" is the name of the library to install
  - - -user allows you to install without root previliges.

ASTRON

# Python packages

- Support functionalities beyond the standard python library.
- For example: you cannot plot using standard python. Use matplotlib.
- You need to install the libraries separately.
- We'll use **pip** to install some useful libraries.
- pip install "name" - -user
  - "name" is the name of the library to install
  - - -user allows you to install without root previliges.
- Useful packages for this workshop:
  - NumPy - math operations on N-dimensional arrays
  - Matplotlib - plotting library
  - AstroPy
  - ApIPy - Plotting/visualizing images.

ASTRON

# Brief overview of AplPy

- APLPy - Astronomical Plotting Library in Python.
- Support for FITS datasets.
- Can produce publication-quality images

ASTRON

# Brief overview of AplPy

- APLPy - Astronomical Plotting Library in Python.
- Support for FITS datasets.
- Can produce publication-quality images

- Import the library in your script using

```
import aplpy as a
```

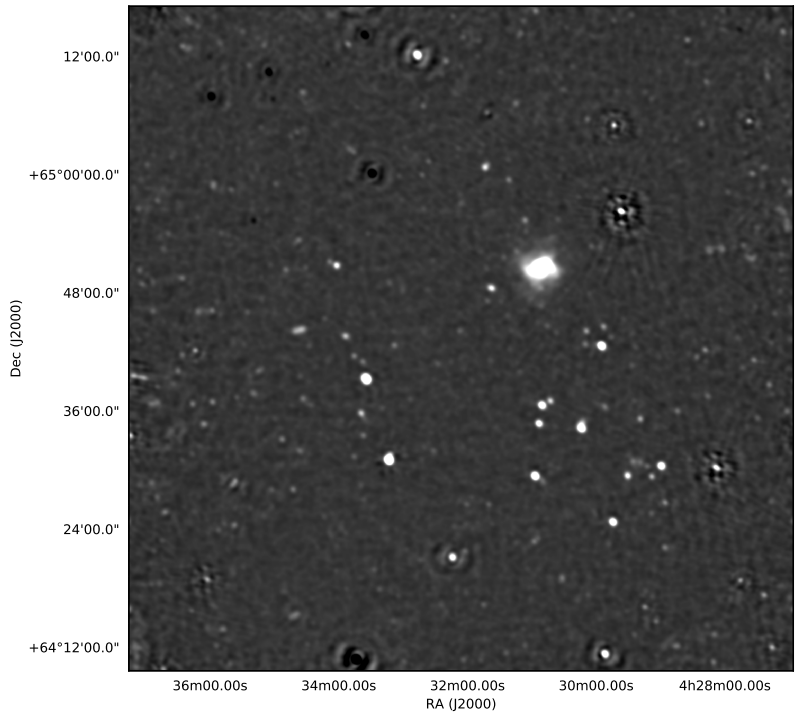
ASTRON

# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

f.show_grayscale()

f.save('n1569_orig.pdf', dpi=200)
```

# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

f.show_grayscale()

f.save('n1569_orig.pdf', dpi=200)
```



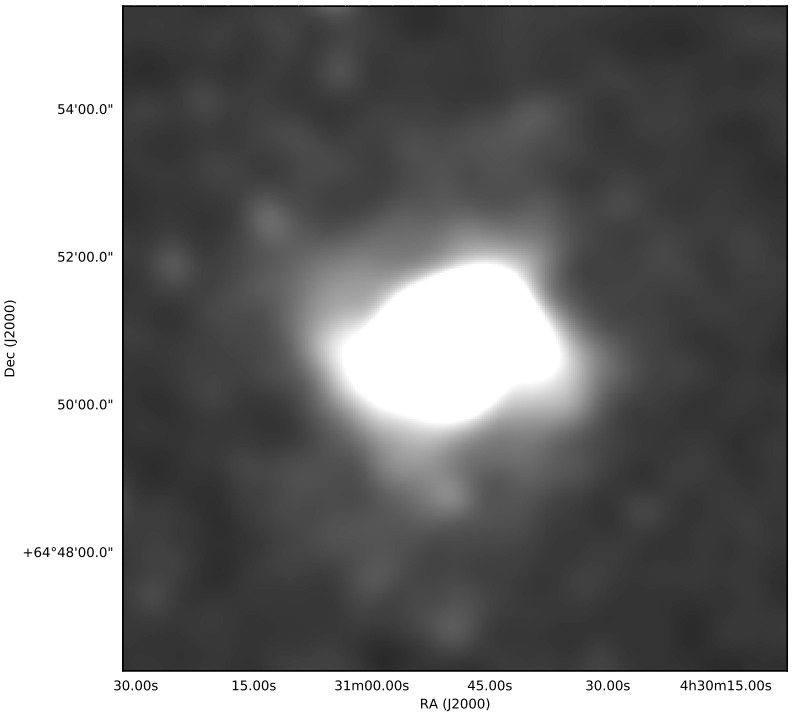
# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_grayscale()

f.save('n1569_orig.pdf', dpi=200)
```

ASTRON

# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_grayscale()

f.save('n1569_orig.pdf', dpi=200)
```

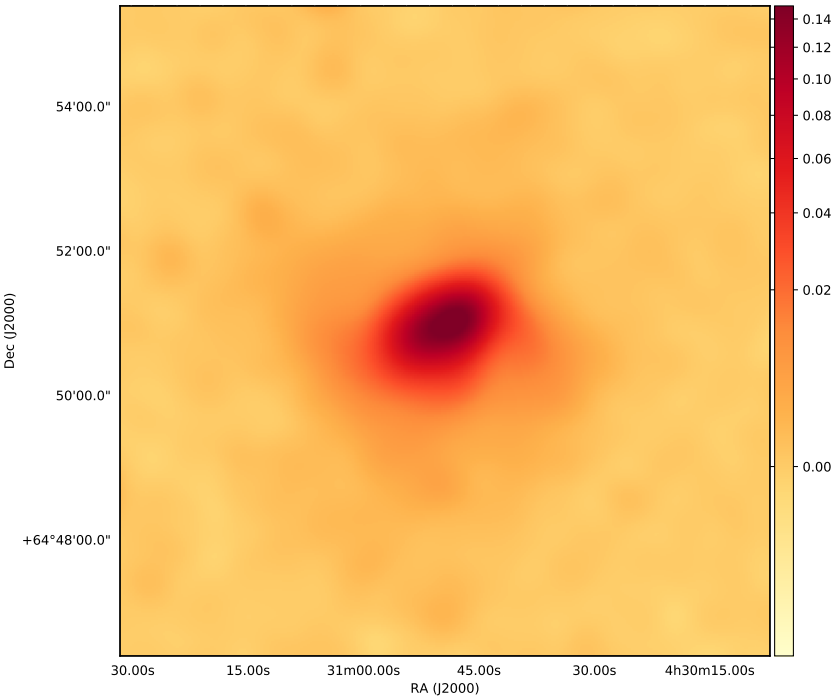


# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_colorscale(cmap='YlOrRd', \
                  vmin=-0.0025, \
                  vmax=0.15, \
                  stretch='power',\
                  exponent=0.3)
f.show_colorbar()

f.save('n1569_orig.pdf', dpi=200)
```

# Plotting with APlPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_colorscale(cmap='YlOrRd', \
                  vmin=-0.0025, \
                  vmax=0.15, \
                  stretch='power',\
                  exponent=0.3)
f.show_colorbar()

f.save('n1569_orig.pdf', dpi=200)
```



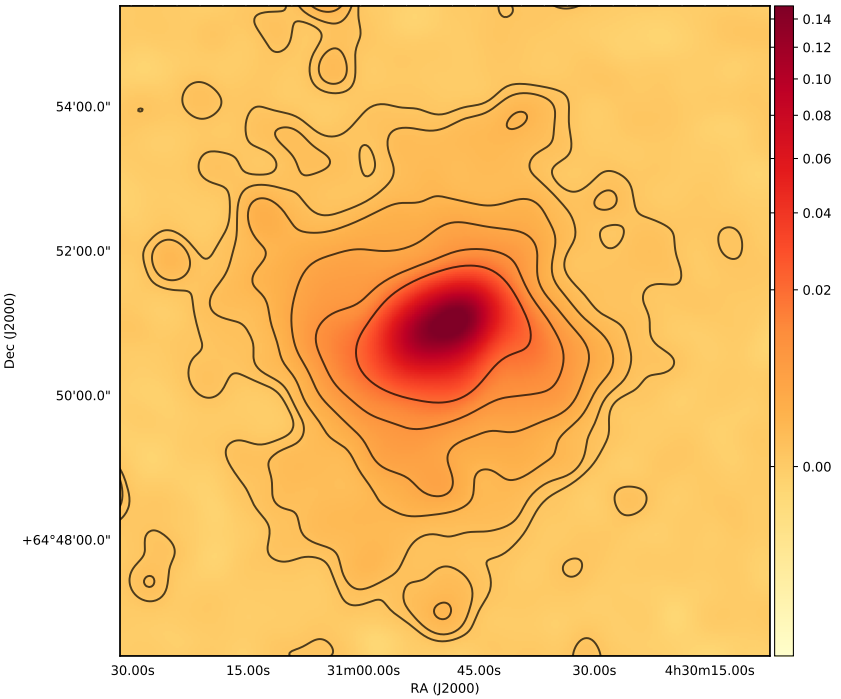
# Plotting with AplPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_colorscale(cmap='YlOrRd', \
    vmin=-0.0025, vmax=0.15, \
    stretch='power', exponent=0.3)
f.show_colorbar()

contours = [0.000315*2, \
    0.000315*4, 0.000315*8, \
    0.000315*16, 0.000315*32,\
    0.000315*64]
f.show_contour(filename, \
    levels=contours, colors='black',\
    alpha=0.7)

f.save('n1569_orig.pdf', dpi=200)
```

ASTRON

# Plotting with APlPy

```
#!/usr/bin/env python
import aplpy as a

filename = 'n1569.fits'
f = a.FITSFigure(filename)

ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

f.show_colorscale(cmap='YlOrRd', \
    vmin=-0.0025, vmax=0.15, \
    stretch='power', exponent=0.3)
f.show_colorbar()

contours = [0.000315*2, \
    0.000315*4, 0.000315*8, \
    0.000315*16, 0.000315*32,\
    0.000315*64]
f.show_contour(filename, \
    levels=contours, colors='black',\
    alpha=0.7)

f.save('n1569_orig.pdf', dpi=200)
```



# Brief overview of NumPy

- NumPy is a wide-used python library for N-dim array manipulation
- Import NumPy with the line

```
import numpy as np
```

# 1-D array

- Create an array with

```
a = np.array([1,2,3,4,5])
```

- **a.ndim** tells you the dimensions of the array
- **a.shape** tells you the shape of the array

ASTRON

# 1-D array

- Create an array with

```
a = np.array([1,2,3,4,5])
```

- **a.ndim** tells you the dimensions of the array
- **a.shape** tells you the shape of the array
- **np.max(a)** and **np.min(a)** tell you the max and min values in the array

ASTRON

# 1-D array

- Create an array with

```
a = np.array([1,2,3,4,5])
```

- **a.ndim** tells you the dimensions of the array
- **a.shape** tells you the shape of the array
- **np.max(a)** and **np.min(a)** tell you the max and min values in the array
- Try **np.argmax(a)** and **np.argmin(a)**. What do they tell us?

ASTRON

# 1-D array

- Create an array with

```
a = np.array([1,2,3,4,5])
```

- **a.ndim** tells you the dimensions of the array
- **a.shape** tells you the shape of the array
- **np.max(a)** and **np.min(a)** tell you the max and min values in the array
- Try **np.argmax(a)** and **np.argmin(a)**. What do they tell us?
- What happens when you do **a*10**?

ASTRON

# 1-D array

- Create an array with

```
a = np.array([1,2,3,4,5])
```

- **a.ndim** tells you the dimensions of the array
- **a.shape** tells you the shape of the array
- **np.max(a)** and **np.min(a)** tell you the max and min values in the array
- Try **np.argmax(a)** and **np.argmin(a)**. What do they tell us?
- What happens when you do **a*10**?

## Exercise

- Create an array with the **b=np.zeros((5,5))**. Try using the above-mentioned methods on **b**. What did **np.zeros** create?

ASTRON

# Indexing and slicing

- Recall how we used indices while discussing Lists and Tuples.
- Now, create an array with **c=np.array([1,2,3,4,5])**.

**Exercise:**

- Using indexing on **c**, can you produce this array: [2,3,4]?
- Can you print the reverse of **c**?

ASTRON

# Indexing and slicing

- Recall how we used indices while discussing Lists and Tuples.
- Now, create an array with **c=np.array([1,2,3,4,5])**.

## Exercise:

- Using indexing on **c**, can you produce this array: [2,3,4]?
- Can you print the reverse of **c**?

## Some useful functions

- np.arange() and np.linespace()
- np.ones(), np.ones_like(), np.zeros(), and np.zeros_like()
- np.eye()

ASTRON

# More indexing and slicing

- Create an array with **d = np.random.randint(1,10,10)**
- Print the content of **d**. Can you guess what the above function did?

ASTRON

# More indexing and slicing

- Create an array with **d = np.random.randint(1,10,10)**
- Print the content of **d**. Can you guess what the above function did?
- Execute **d>3**. Can you interpret the result?

ASTRON

# More indexing and slicing

- Create an array with **d = np.random.randint(1,10,10)**
- Print the content of **d**. Can you guess what the above function did?
- Execute **d>3**. Can you interpret the result?
- Execute **d[d>3]**. Can you interpret the result?

ASTRON

# Simple plotting examples with Matplotlib

- Create an array with **x=np.linspace(0,10,1000)**
- Compute the sin of **x** with **y=np.sin(x)**

ASTRON

# Simple plotting examples with Matplotlib

- Create an array with **x=np.linspace(0,10,1000)**
- Compute the sin of **x** with **y=np.sin(x)**
- Now, let's plot this sin curve

```
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

- What happens if we replace the above plot with **plt.plot(x,y,'k−−')**?

# Simple plotting examples with Matplotlib

- Create an array with **x=np.linspace(0,10,1000)**
- Compute the sin of **x** with **y=np.sin(x)**
- Now, let's plot this sin curve

```
import matplotlib.pyplot as plt
plt.plot(x,y)
plt.show()
```

- What happens if we replace the above plot with **plt.plot(x,y,'k−−')**?

## Exercise:

- Can you plot the cosine of **x** on the above plot?
- What is the purpose of **plt.xlabel()** and **plt.ylabel()**? Use python help.

ASTRON

# How to read FITS images using AstroPy?

- AstroPy is more than just a tool to read FITS images
- You can open a FITS image and create a new FITS image using

```
import astropy.io.fits as pf
data = pf.open('n1569.fits')[0].data
# Do something with the data
pf.PrimaryHDU(data=data)
pf.writeto('new_fits_image.fits')
```

ASTRON

# How to read FITS images using AstroPy?

- AstroPy is more than just a tool to read FITS images
- You can open a FITS image and create a new FITS image using

```
import astropy.io.fits as pf
data = pf.open('n1569.fits')[0].data
# Do something with the data
pf.PrimaryHDU(data=data)
pf.writeto('new_fits_image.fits')
```

## Exercise:

- In the above code, can you find the size of **data**?

ASTRON

# How to read FITS images using AstroPy?

- AstroPy is more than just a tool to read FITS images
- You can open a FITS image and create a new FITS image using

```
import astropy.io.fits as pf
data = pf.open('n1569.fits')[0].data
# Do something with the data
pf.PrimaryHDU(data=data)
pf.writeto('new_fits_image.fits')
```

## Exercise:

- In the above code, can you find the size of **data**?
- What is the maximum value in **data**?

# How to read FITS images using AstroPy?

- AstroPy is more than just a tool to read FITS images
- You can open a FITS image and create a new FITS image using

```
import astropy.io.fits as pf
data = pf.open('n1569.fits')[0].data
# Do something with the data
pf.PrimaryHDU(data=data)
pf.writeto('new_fits_image.fits')
```

## Exercise:

- In the above code, can you find the size of **data**?
- What is the maximum value in **data**?
- What is the smallest values in **data**?

# How to read FITS images using AstroPy?

- AstroPy is more than just a tool to read FITS images
- You can open a FITS image and create a new FITS image using

```
import astropy.io.fits as pf
data = pf.open('n1569.fits')[0].data
# Do something with the data
pf.PrimaryHDU(data=data)
pf.writeto('new_fits_image.fits')
```

## Exercise:

- In the above code, can you find the size of **data**?
- What is the maximum value in **data**?
- What is the smallest values in **data**?
- Can you create a new array called **mask** that is **1** when **data**$>0.08$ and **0** everywhere?

ASTRON