

Introduction to Python 3

Sarrvesh S. Sridhar

ASTRON, the Netherlands

July 1, 2019

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 3. I will cover only what is necessary/useful for this workshop. See slide on “Further reading” for references.

What is python?

- Programming language
 - ▶ general purpose
 - ▶ interpreted
 - ▶ high-level
 - ▶ Created by Guido van Rossum almost 30 years ago
 - ▶ **All support for Python 2 ends on Jan 1, 2020.**

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

Python interpreter

- Start the interpreter by typing **python3** in your command prompt

```
darkknight@phoenix:~$ python3
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Python interpreter

- Start the interpreter by typing **python3** in your command prompt

```
darkknight@phoenix:~$ python3
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

- Type the following commands in the prompt
 - ▶ `a = 10`
 - ▶ `b = 20.5`
 - ▶ `c = a+b`
 - ▶ `type(a); type(b); type(c)`

Python interpreter

- Start the interpreter by typing **python3** in your command prompt

```
darkknight@phoenix:~$ python3
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

- 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** – especially division.

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 **python3 mycode.py**

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 **python3 mycode.py**
- Or, you can add **#!/usr/bin/env python3** as the first line in mycode.py
- Change the file permission with **chmod u+x mycode.py**
- Finally, execute as **./mycode.py**

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])**

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])**

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)
```

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

- 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!

Python packages

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

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 privileges.

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 privileges.
- Useful packages for this workshop:
 - ▶ NumPy - math operations on N-dimensional arrays
 - ▶ Matplotlib - plotting library
 - ▶ AstroPy
 - ▶ AplPy - Plotting/visualizing images.

Brief overview of APLPy

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

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
```


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)
```

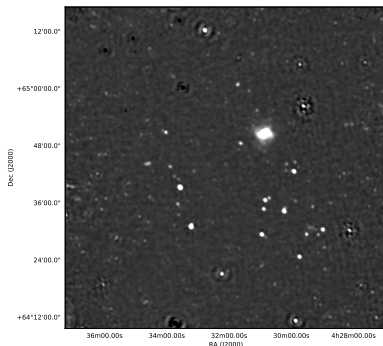
Plotting with APLPy

```
#!/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)
```

Plotting with AplPy

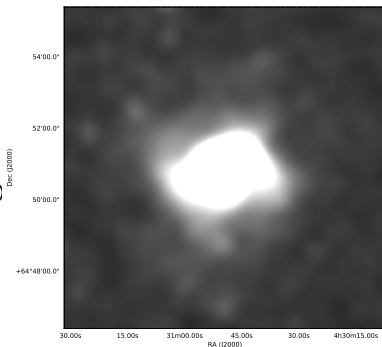
```
#!/usr/bin/env python
import aplpy as a

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

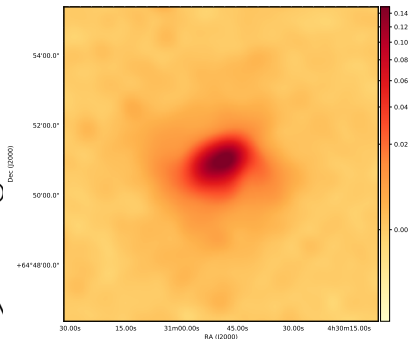
filename = 'n1569.fits'
f = a.FITSFigure(filename)

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)
```

Plotting with APLPy

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

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

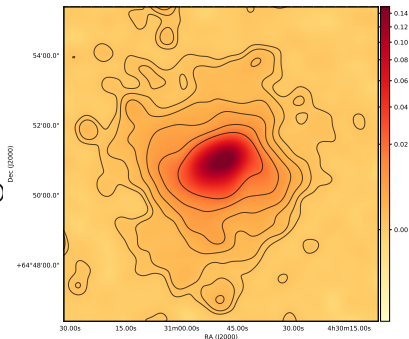
ra = 67.705; dec = 64.8479
f.recenter(ra, dec, radius=4.5/60.)

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

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

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?

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**?

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 arrays with the **b=np.zeros((5,5))** and **c=np.ones((6,6))**. Try using the above-mentioned methods on **b** and **c**. What did **np.zeros** create?

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`?

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.linspace()`
- `np.ones()`, `np.ones_like()`, `np.zeros()`, and `np.zeros_like()`
- `np.eye()`

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?

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?

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?

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)`

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.

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')
```

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**?

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?