

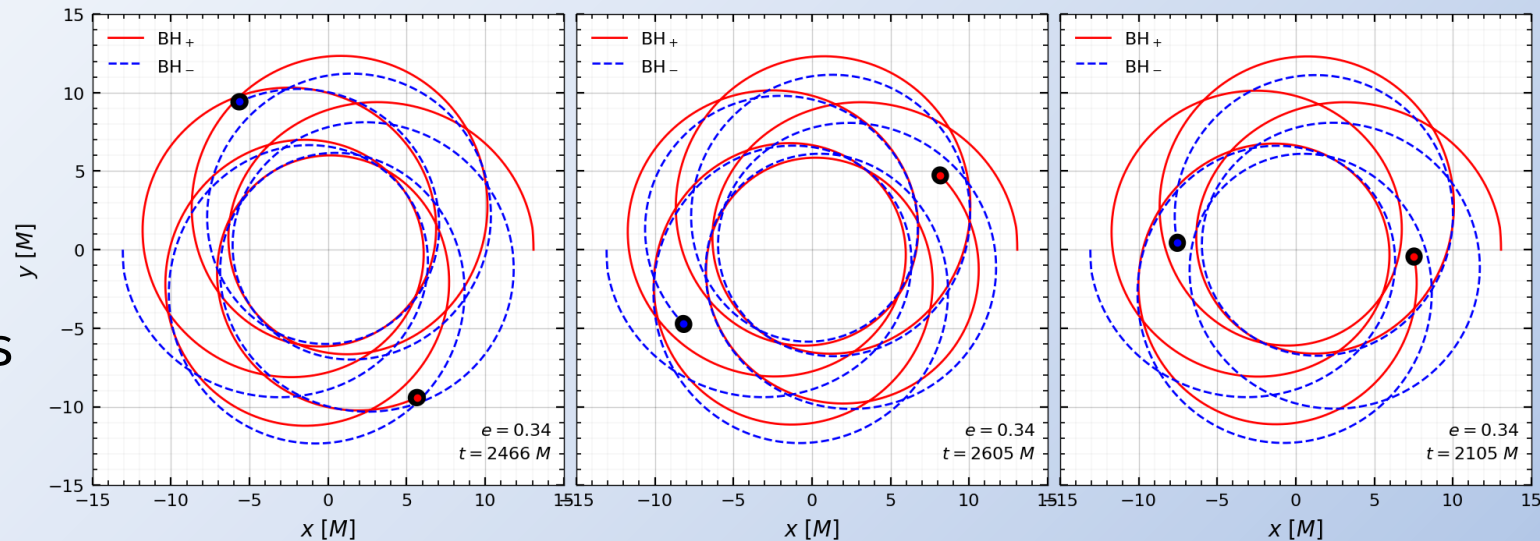
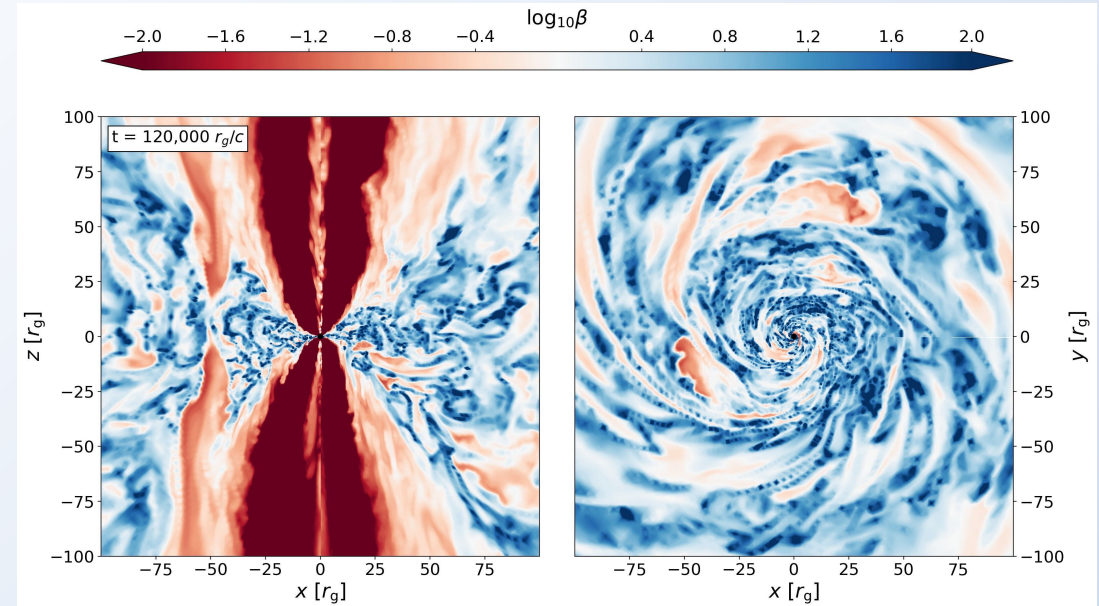
Data Visualization

COMPASS Workshop

```
Instructors = ["Vikram Manikantan",  
               "Erik Wessel",  
               "Vasilis Paschalidis"]
```

Contents

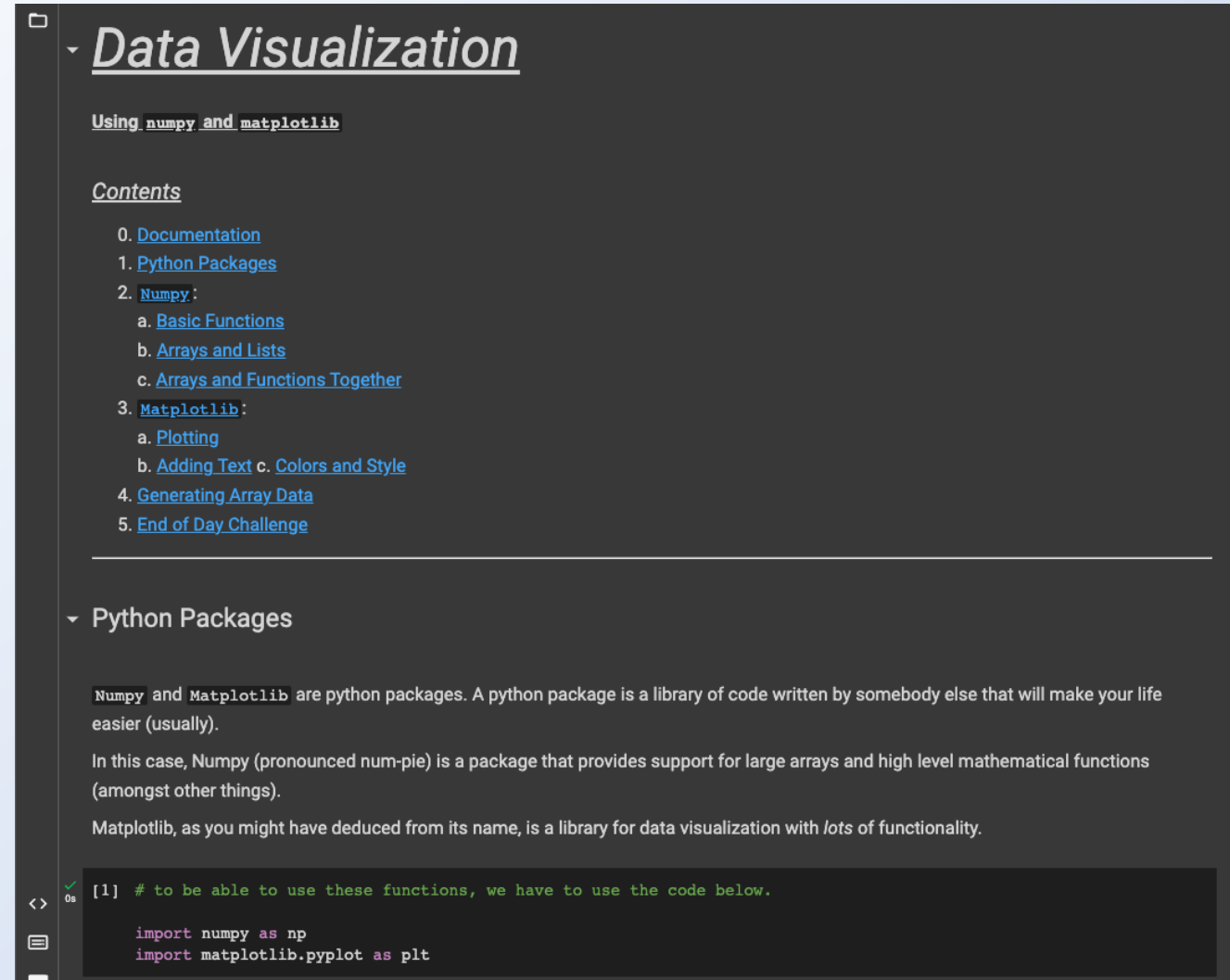
1. Introduction to Python Packages
2. Documentation
3. Numpy
 - a. Math Functions
 - b. Arrays
4. Matplotlib
 - a. Plotting Basics
 - b. Labels & Titles
 - c. Legends & Line styles



Google is your friend...

Today's Notebook

[tinyurl.com/
compassnumpy](https://tinyurl.com/compassnumpy)



The screenshot shows a Jupyter Notebook with a dark theme. The first cell is a markdown cell titled **Data Visualization** with a subtitle **Using numpy and matplotlib**. It contains a **Contents** table of contents with links to documentation, python packages, numpy basics, matplotlib plotting, and an end-of-day challenge. The second cell is a markdown cell titled **Python Packages** explaining that Numpy and Matplotlib are python packages and describing their functions. The third cell is a code cell with a comment and two import statements.

Data Visualization

Using `numpy` and `matplotlib`

Contents

- 0. [Documentation](#)
- 1. [Python Packages](#)
- 2. [Numpy](#):
 - a. [Basic Functions](#)
 - b. [Arrays and Lists](#)
 - c. [Arrays and Functions Together](#)
- 3. [Matplotlib](#):
 - a. [Plotting](#)
 - b. [Adding Text](#)
 - c. [Colors and Style](#)
- 4. [Generating Array Data](#)
- 5. [End of Day Challenge](#)

Python Packages

`Numpy` and `Matplotlib` are python packages. A python package is a library of code written by somebody else that will make your life easier (usually).

In this case, Numpy (pronounced num-pie) is a package that provides support for large arrays and high level mathematical functions (amongst other things).

Matplotlib, as you might have deduced from its name, is a library for data visualization with *lots* of functionality.

```
[1] # to be able to use these functions, we have to use the code below.  
  
import numpy as np  
import matplotlib.pyplot as plt
```

Python Packages

- A collection of functions and code 'packaged' together
- Sometimes called a 'library' of code
- Different packages have different purposes

```
>>> import numpy as np  
>>> import matplotlib.pyplot as plt
```

we talked about functions

```
def calc_quadratic()
```

a function is like a book

```
def calc_quadratic()
```

a package is a library of functions

| | | | | | |
|-----------------------------------|--------------------------------|-------------------------------|----------------------|----------------------|----------------------|
| <code>def calc_quadratic()</code> | <code>def calc_lienar()</code> | <code>def calc_cubic()</code> | <code>def ...</code> | <code>def ...</code> | <code>def ...</code> |
|-----------------------------------|--------------------------------|-------------------------------|----------------------|----------------------|----------------------|

and many more...

Numpy

- High-level math functions
- Support for large-arrays
- Fundamental for science

```
>>> import numpy as np
>>> np.sin(0)
0
>>> np.cos(0)
1
>>> np.array([1,2,3])
[1 2 3]
```

Aside: Documentation

- Description of what code accomplishes in Human-readable language
- More descriptive than commenting
- Example of commenting:

```
# this is a comment:  
# this code takes the  
# variables a and b and adds  
# them together
```

```
>>> a = 5  
>>> b = 3  
>>> print("a + b = ", a+b)  
a + b = 8
```

Example of Documentation

- Describes a function by its inputs and outputs
- Neatly organized for future users and developers
- Online Example

```
def multiply(a,b):  
    """  
    inputs:  
        a: double, first number to  
        be multiplied  
        b: double, second number  
    returns:  
        double: product of inputs  
    """  
    return a*b
```

Numpy Arrays

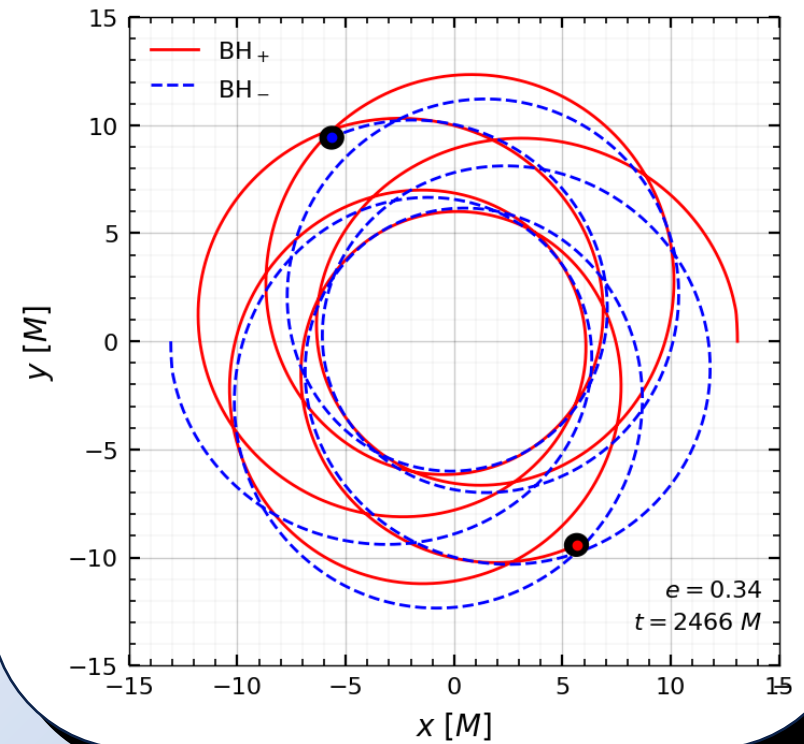
- Like the arrays we looked at yesterday but more powerful
- We can treat arrays like a number, and do **arithmetic** on them
- Let's look at the notebook...

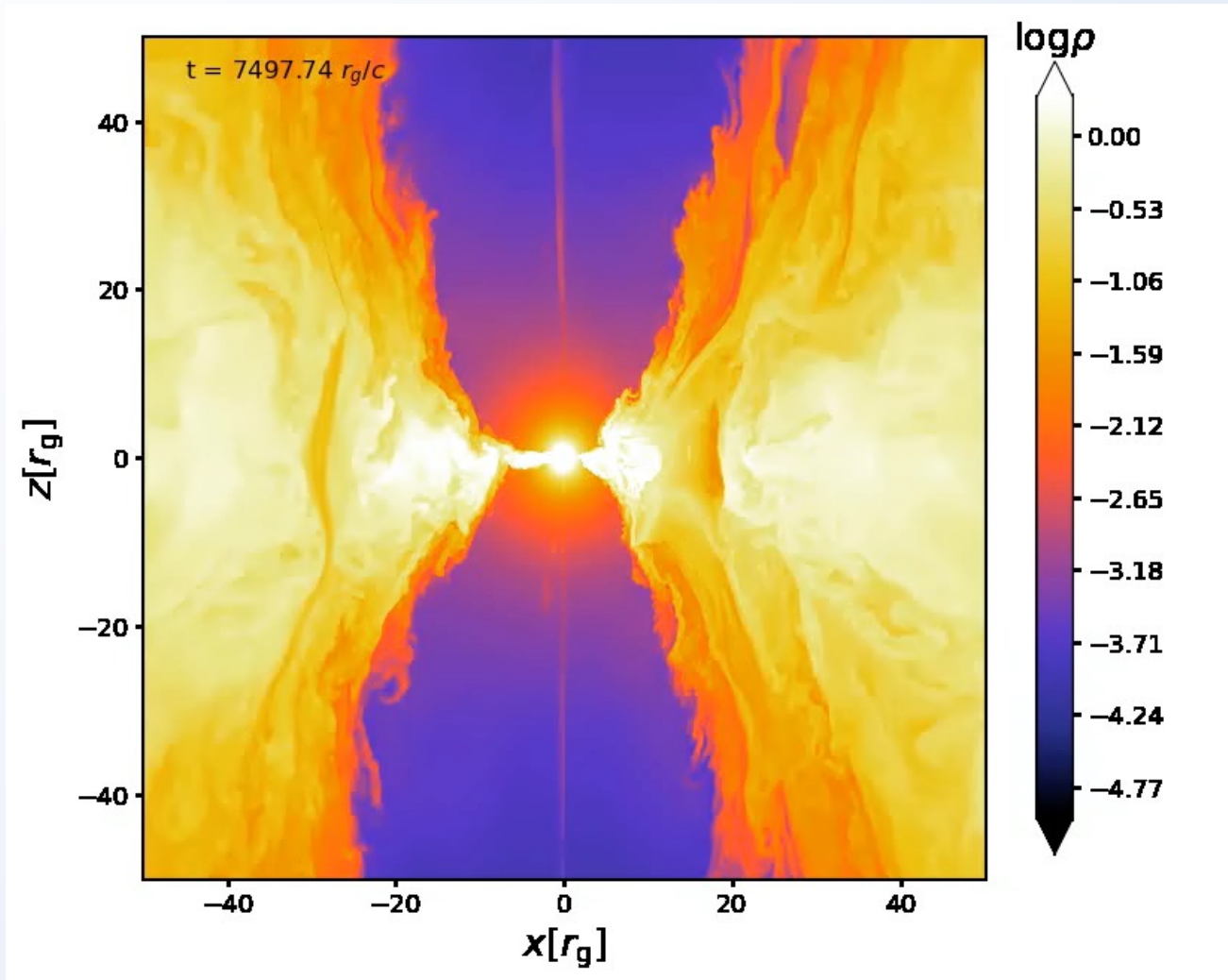
```
>>> import numpy as np
>>> arr = np.array([1,2,3])
>>> print(arr)
[1 2 3]
>>>
>>> arr*5
[5 10 15]
>>>
```

Matplotlib

- Library for creating visualizations
- Static, animated, and interactive
- Very comprehensive

```
>>> import matplotlib.pyplot  
as plt  
>>> plt.plot(x1, y1, color =  
'blue')
```

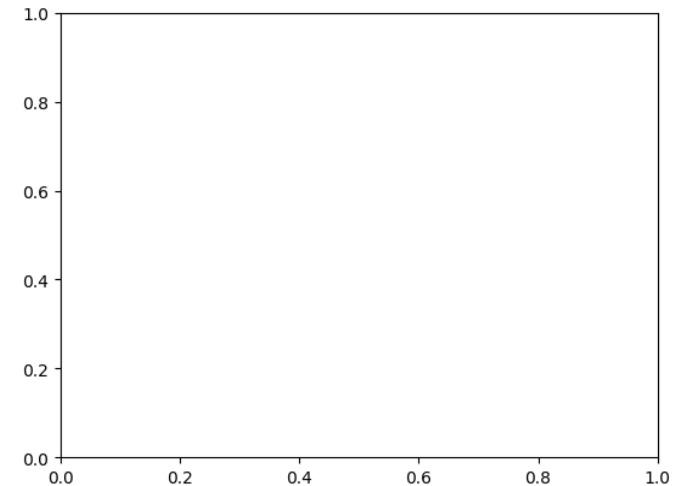




Creating the Graph

- First, import the package
- ax stands for 'Axes'.
This is what you will be
drawing your graph 'on'

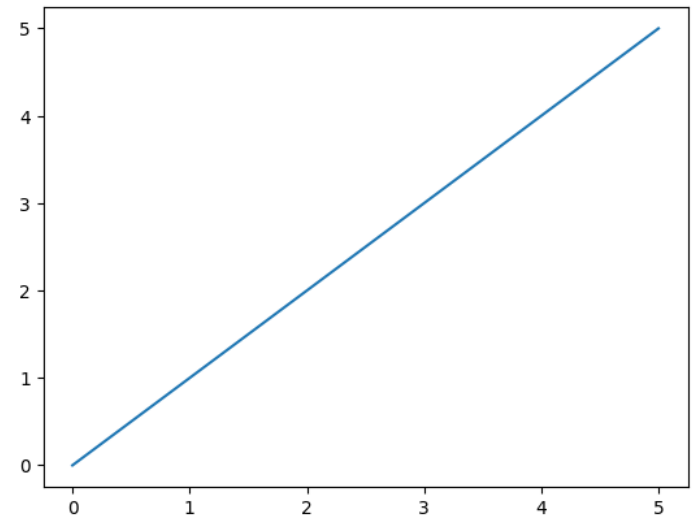
```
>>> import matplotlib.pyplot  
as plt  
>>> fig, ax = plt.subplots()
```



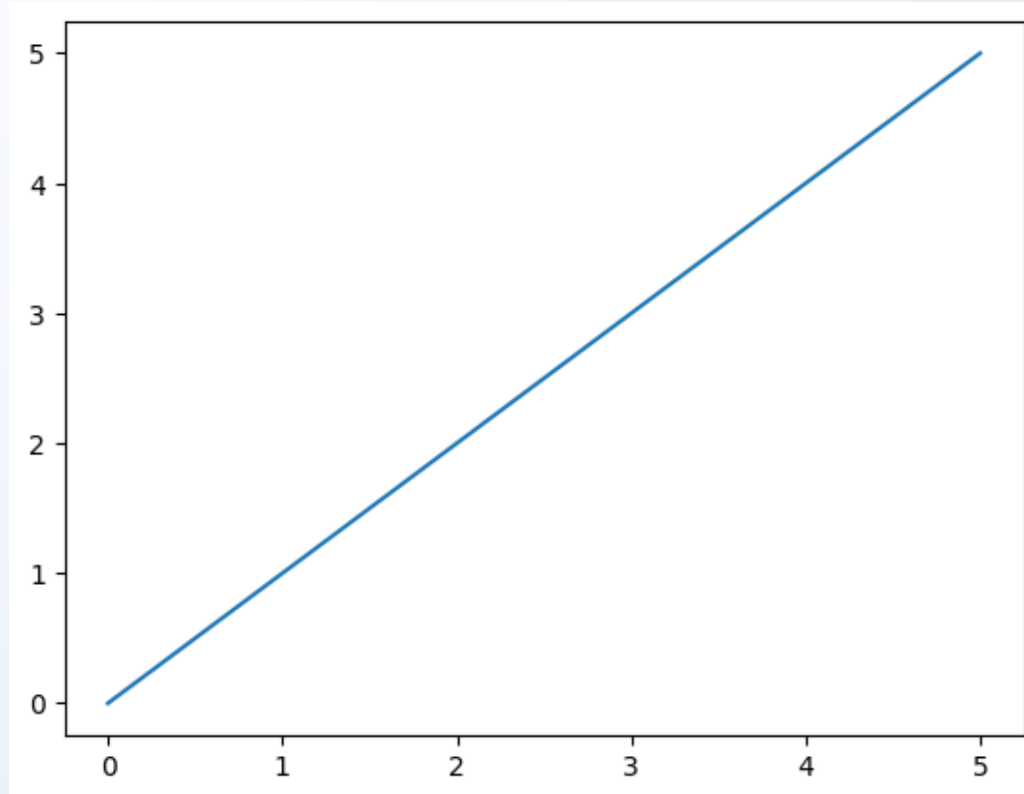
Adding Lines

- Now we need something to plot
- Make our own data

```
>>> data_x = [0, 1, 2, 3, 4]  
>>> data_y = [0, 1, 2, 3, 4]  
>>> ax.plot(data_x, data_y)
```

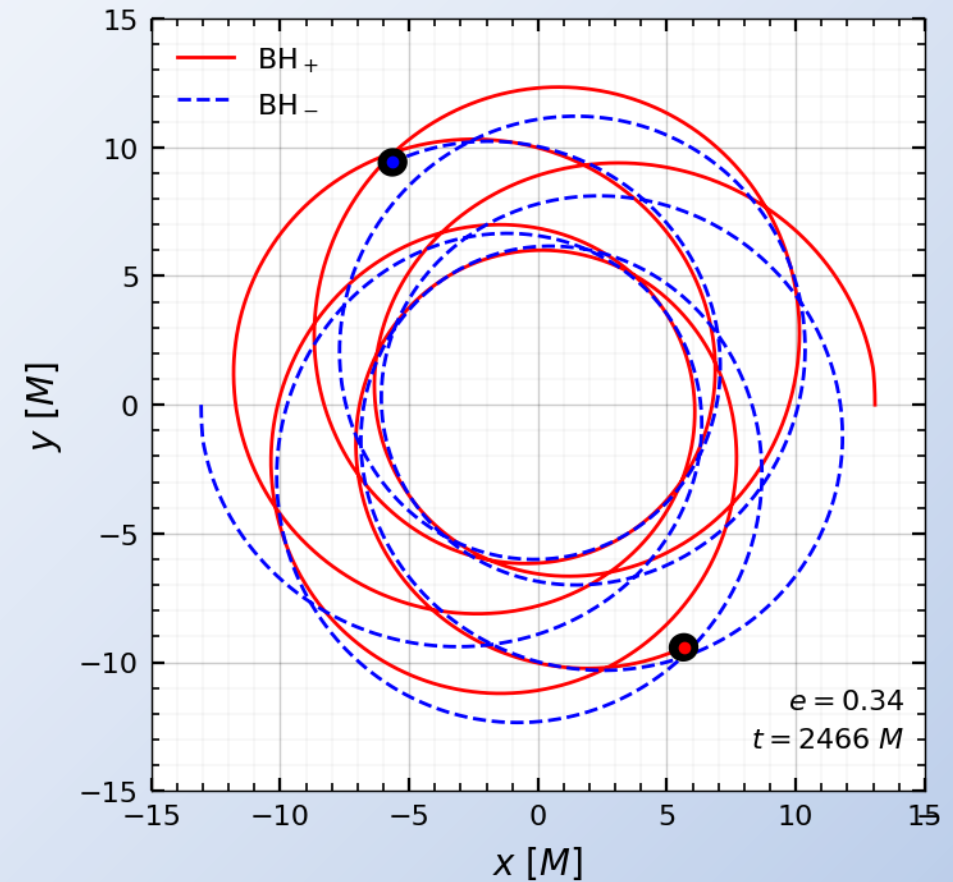


How do we make this better?



Additional Elements

- Title
- Axes labels
- Legends
- Colors
- Linestyles
- Grid?
- ...and many more



Generating Data

- You can create filled arrays with numpy
- `numpy.arange(start, end, step)`
- `numpy.linspace(start, end, number of elements)`

```
>>> import numpy as np
>>> np.arange(0,10,2)
[0 2 4 6 8]
>>>
>>> np.linspace(0, 10, 6)
[0 2 4 6 8 10]
>>>
```