

Introduction to Python and ObsPy



ObsPy

A Python Framework for Seismology

2016 IRIS-EarthScope Short Course

Erin Wirth

University of Washington

ewirth@uw.edu

Overview

- “Why use Python?”

Overview

- “Why use Python?”
- Introduction to Python and Jupyter Notebooks

Overview

- “Why use Python?”
- Introduction to Python and Jupyter Notebooks
- ObsPy Examples

Overview

- “Why use Python?”
- Introduction to Python and Jupyter Notebooks
- ObsPy Examples

First... What is Python?!

First... What is Python?!

- Interpreted

First... What is Python?!

- Interpreted
- High-level

First... What is Python?!

- Interpreted
- High-level
- Free and open-source

First... What is Python?!

- Interpreted
- High-level
- Free and open-source
- Object-oriented
 - Each object has various *attributes* and *methods*

“Why do I use Python?”

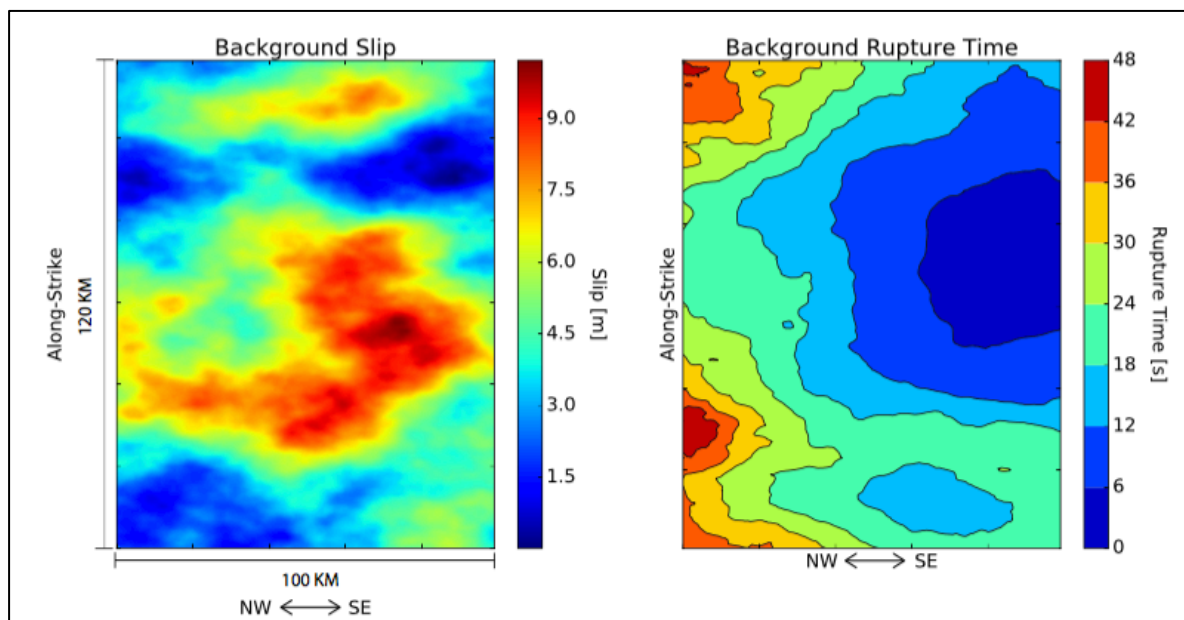
“Why do I use Python?”

Erin's Research:

How can we better understand and prepare for large, subduction zone earthquakes in Cascadia?

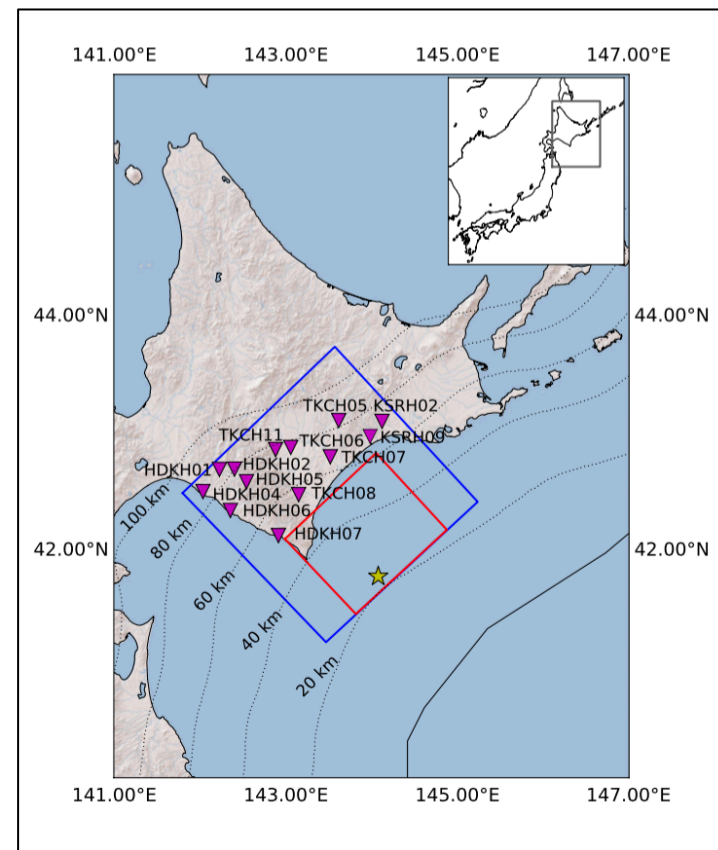
“Why do I use Python?”

- Actual Data and Synthetic Waveforms



“Why do I use Python?”

- Actual Data and Synthetic Waveforms
- Station Information
 - Lat./Lon./Depth/Component
- Event Information
 - Lat./Lon./Depth/Origin Time



“Why do I use Python?”

- Actual Data and Synthetic Waveforms
- Station Information
 - Lat./Lon./Depth/Component
- Event Information
 - Lat./Lon./Depth/Origin Time
- Start and End Time of Waveforms
- Sampling Rate

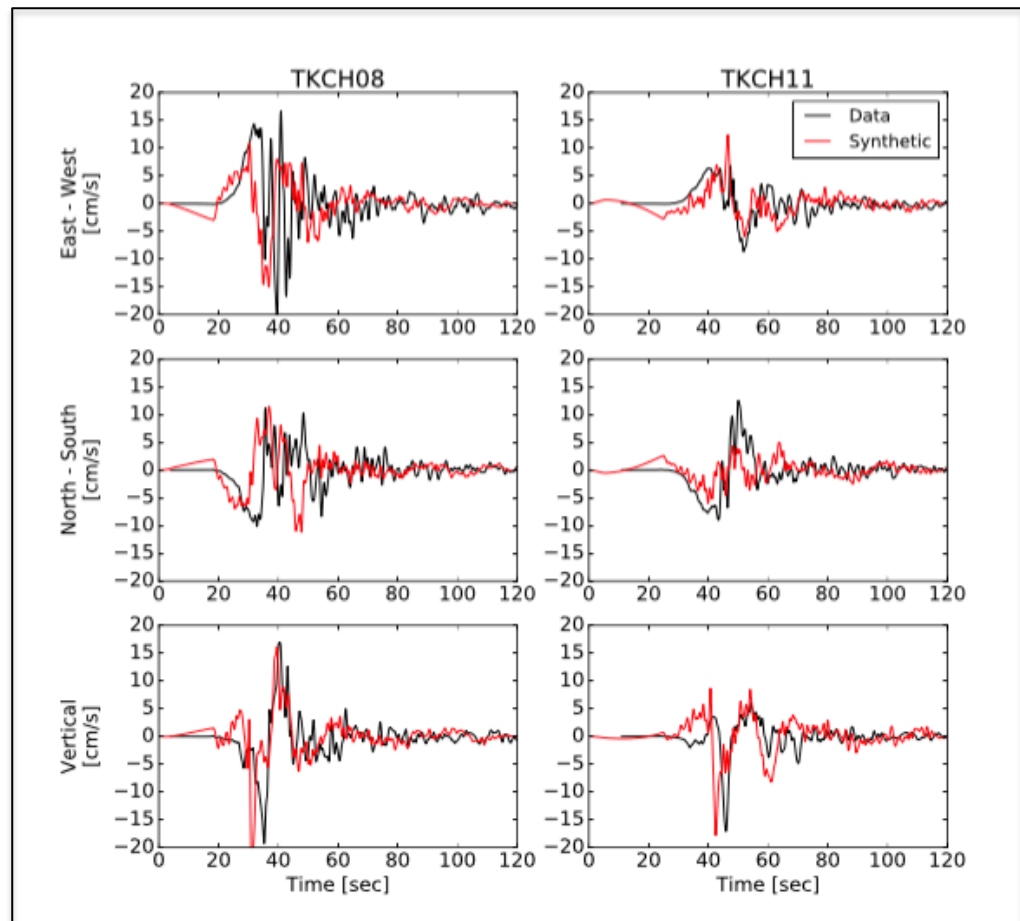


METADATA!

“Why do I use Python?”

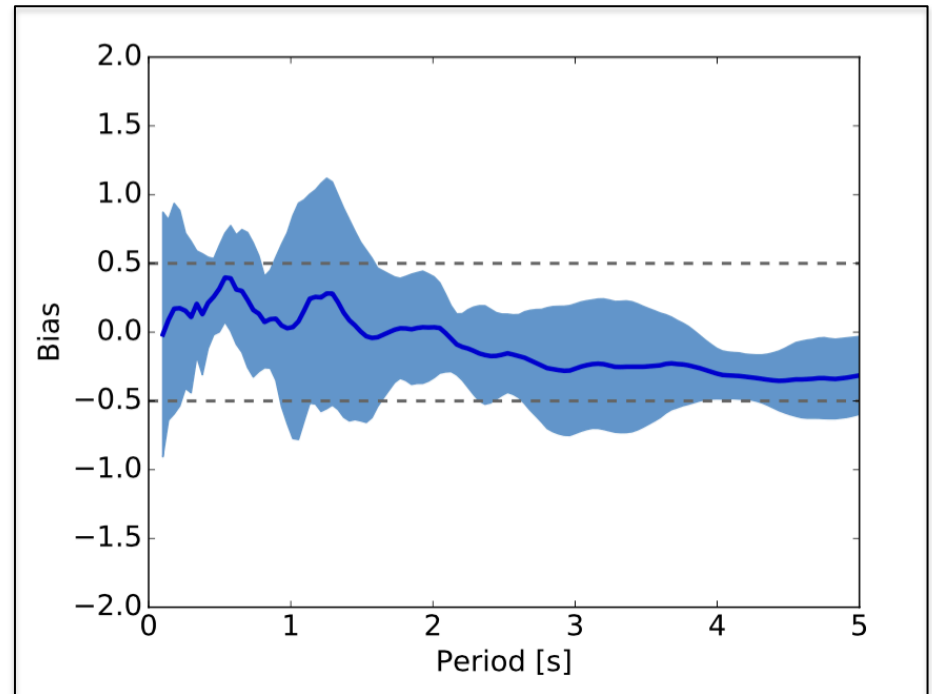
P
R
O
C
E
S
S
I
N
G

- Demean and Detrend
- Filter
- Taper
- Integrate



“Why do I use Python?”

- Demean and Detrend
- Filter
- Taper
- Integrate
- Compare
 - Visually (e.g., a plot)
 - Quantitatively (e.g., numerical operations)



“Why do I use Python?”

SAC is great for viewing header information and performing operations.

BUT, SAC macros are cumbersome and the plotting is ugly.

So, why use SAC when I can do ALL of this in Python?!

“Why do I use Python?”

There's even *more* **good news**...

Seismology-friendly data structures already exist in a Python package called ObsPy.

“Why do I use Python?”

There's even *more* **good news**...

Seismology-friendly data structures already exist in a Python package called ObsPy.

With Python + ObsPy, you can use powerful numerical and scientific libraries to **process and visualize your data**.

Overview

- “Why use Python?”
- **Introduction to Python and Jupyter Notebooks**
- ObsPy Examples

Introduction to Python

Different Ways to Use Python

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else
2. Use the **interactive IPython shell**
 - Code can be executed line-by-line, results are immediately visible
 - Quick, interactive

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else
2. Use the **interactive IPython shell**
 - Code can be executed line-by-line, results are immediately visible
 - Quick, interactive
3. In a **browser**, using **IPython/Jupyter Notebooks**
 - Code can be mixed with images, formulas, and text
 - Good for teaching!
 - Can be a little confusing

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else
2. Use the **interactive IPython shell**
 - Code can be executed line-by-line, results are immediately visible
 - Quick, interactive
3. In a **browser**, using **IPython/Jupyter Notebooks**
 - Code can be mixed with images, formulas, and text
 - Good for teaching!
 - Can be a little confusing

Introduction to Python -- *IPython*

Simple Printing and Variable Types

Introduction to Python -- *IPython*

Simple Printing and Variable Types

In Terminal, start IPython:

```
>> ipython
```

Now, in IPython shell:

```
>> print('Hello, World!')
```

Introduction to Python -- *IPython*

Simple Printing and Variable Types

In IPython shell:

```
>> a = '1'
```

str

```
>> a = 1
```

int

```
>> a = 1.1
```

float

```
>> a = 1+2j
```

complex

Variable
Types



Introduction to Python -- *IPython*

Simple Printing and Variable Types

In IPython shell:

>> *a = [42, 17, 6]* ← *a list*

>> *a[1]*

>> *a.<tab>* ←

In IPython, use **<tab>**
and **?** to explore
attributes/methods of
an object

Introduction to Python -- *IPython*

Simple Printing and Variable Types

In IPython shell:

>> *a = 'hello there'* ← *a string*

>> *a[1]*

>> *a.<tab>* ←

In IPython, use **<tab>**
and **?** to explore
attributes/methods of
an object

Introduction to Python – *Ipython*

If/Else Statement

In IPython shell:

```
>> grade = 100  
>> if (grade > 65):  
...   print('You passed!')  
... else:  
...   print('Yikes.')
```



This MUST be 4 Spaces! (or 1 Tab)

Introduction to Python – *Ipython*

While Loop

In IPython shell:

```
>> a = 0
```

```
>> while (a < 5):
```

```
...     print(a)
```

```
...     a += 1
```



4 Spaces!

Introduction to Python – *Ipython*

For Loop

In IPython shell:

```
>> for i in range(5):  
...   print(i, 'Hello!')
```



4 Spaces!

Introduction to Python – *Ipython*

For Loop – Looping over Lists

In IPython shell:

```
>> names = ['Bob', 'Liz', 'Mel']
```

```
>> for name in names:
```

```
...     print(name + ' loves Python')
```



4 Spaces!

Introduction to Python

Using Modules

What is a **module**?

*“Simply, a **module** is a file consisting of Python code. A module can **define functions, classes, and variables.**”*

Python does not load modules unless you ask for them!

```
>> import os
>> help (os)
>> os.environ
>> myname=os.environ['USER']
>> print(myname)
```


Introduction to Python

Write Your Own Module

In a file called *mymodule.py*:

```
import os
```


```
def sayhello():
```



a function

```
    myname = os.environ['USER']  
    print('Hello, ' + myname + '!')
```

```
def addthese(a,b):
```



*a function with
arguments*

```
    c = a+b  
    print(str(c))  
    return c
```

Introduction to Python

Use Your Own Module

In IPython shell:

```
>> import mymodule  
>> mymodule.sayhello()  
>> a = mymodule.addthese(3.1, 2.7)
```

Or, if you're feeling lazy:

```
>> import mymodule as mm  
>> mm.sayhello()  
>> a = mm.addthese(3.1, 2.7)
```

Introduction to Python

Useful, Preexisting Modules

Introduction to Python

Useful, Preexisting Modules

NumPy

“the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities”

Introduction to Python

Useful, Preexisting Modules

SciPy

“a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and much more”

Introduction to Python

Useful, Preexisting Modules

matplotlib

“matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms”

→ gives a MATLAB-like syntax

Introduction to Python

Now you know a little about...

- Data Types
- Object Orientation
 - Attributes
 - Methods
- Flow Control Statements (e.g., if/while/for)
 - Mandatory Indentation!
- Importing Modules and Writing Functions

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else
2. Use the **interactive IPython shell**
 - Code can be executed line-by-line, results are immediately visible
 - Quick, interactive
3. In a **browser**, using **IPython/Jupyter Notebooks**
 - Code can be mixed with images, formulas, and text
 - Good for teaching!
 - Can be a little confusing

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)

- `>> python my_program.py`
- Use a text editor of your choice
- Easy to give this file to someone else

2. Use the **interactive IPython shell**

- Code can be executed line-by-line, results are immediately visible
- Quick, interactive

3. In a **browser**, using **IPython/Jupyter Notebooks**

- Code can be mixed with images, formulas, and text
- Good for teaching!
- Can be a little confusing

Introduction to Python – *Scripting*

Writing Your Own *.py* Programs

Example Code (*mylinefit.py*)

- Generate an array of **x** values
- Calculate a function **y(x)**
- Add random noise to **y(x)**
- **Fit a line** through the noisy data
- **Plot** the original function, noisy data, and best-fit line

Introduction to Python – *Scripting*

Writing Your Own .py Programs

Run Script from IPython:

```
>> run mylinefit.py
```

Run in Terminal:

```
>> python mylinefit.py
```

Introduction to Python – *Scripting*

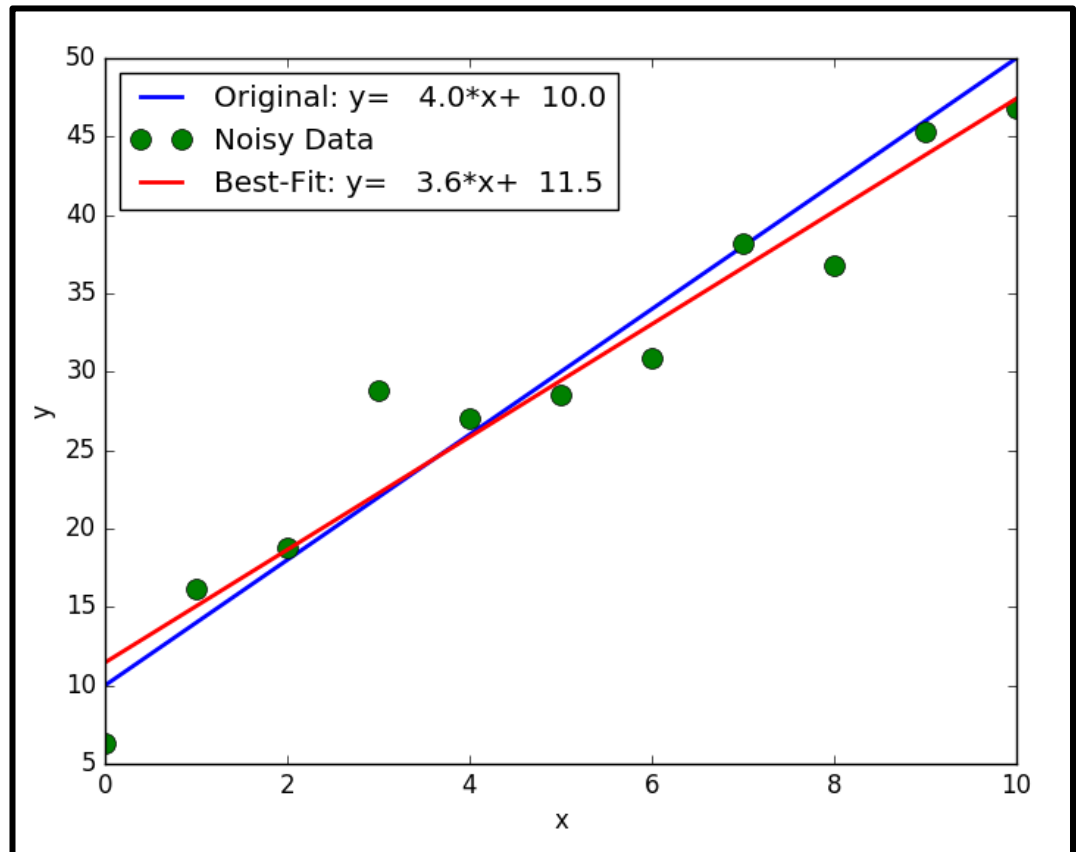
Writing Your Own .py Programs

Run Script from IPython:

```
>> run mylinefit.py
```

Run in Terminal:

```
>> python mylinefit.py
```



Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)

- `>> python my_program.py`
- Use a text editor of your choice
- Easy to give this file to someone else

2. Use the **interactive IPython shell**

- Code can be executed line-by-line, results are immediately visible
- Quick, interactive

3. In a **browser**, using **IPython/Jupyter Notebooks**

- Code can be mixed with images, formulas, and text
- Good for teaching!
- Can be a little confusing

Introduction to Python

Different Ways to Use Python

1. Write a **Python program** (e.g., my_program.py)
 - `>> python my_program.py`
 - Use a text editor of your choice
 - Easy to give this file to someone else
2. Use the **interactive IPython shell**
 - Code can be executed line-by-line, results are immediately visible
 - Quick, interactive
3. In a **browser**, using **IPython/Jupyter Notebooks**
 - Code can be mixed with images, formulas, and text
 - Good for teaching!
 - Can be a little confusing

More Advanced Python

(in Jupyter
Notebooks!)

02_Additional_Python_Practice.ipynb

Overview

- “Why use Python?”
- Introduction to Python and Jupyter Notebooks
- **ObsPy Examples**

ObsPy

A Python Toolbox for Seismology

“an open-source project dedicated to provide a Python framework for processing seismological data. It provides parsers for common file formats and seismological signal processing routines which allow the manipulation of seismological time series”

ObsPy

A Python package for Seismology

(in Jupyter
Notebooks!)

03_Introduction_to_ObsPy.ipynb

Thank You!

ewirth@uw.edu

- Python & ObsPy
 - Anaconda Python
 - <https://www.continuum.io/downloads>
 - ObsPy
 - <https://github.com/obspy/obspy/wiki>
- Learning Resources
 - *A Byte of Python*
 - <http://swaroopch.com/notes/python/>
 - *ObsPy Tutorials*
 - <http://docs.obspy.org/tutorial/>
 - *Python Scripting for Computational Sciences*
 - <http://folk.uio.no/hpl/scripting/index.html>
 - *Python Scientific Lecture Notes*
 - <http://scipy-lectures.github.io/index.html>

Thank You!

ewirth@uw.edu

- Python & ObsPy
 - Anaconda Python
 - <https://www.continuum.io/downloads>
 - ObsPy
 - <https://github.com/obspy/obspy/wiki>
- Learning Resources
 - *A Byte of Python*
 - <http://swaroopch.com/notes/python/>
 - *ObsPy Tutorials*
 - <http://docs.obspy.org/tutorial/>
 - *Python Scripting for Computational Sciences*
 - <http://folk.uio.no/hpl/scripting/index.html>
 - *Python Scientific Lecture Notes*
 - <http://scipy-lectures.github.io/index.html>

Thank You!

ewirth@uw.edu

- *ObsPy Tutorials*

- <http://docs.obspy.org/tutorial/>

ObsPy Documentation (1.0.1) [index](#) | [modules](#) | [next](#) | [previous](#)

ObsPy Tutorial

Note
A one-hour introduction to ObsPy is [available at YouTube](#).

This tutorial does not attempt to be comprehensive and cover every single feature. Instead, it introduces many of ObsPy's most noteworthy features, and will give you a good idea of the library's flavor and style.

A pdf version of the Tutorial is [available here](#).

There are also IPython notebooks available online with an [introduction to Python](#) ([with solutions/output](#)), an [introduction to ObsPy](#) ([with solutions/output](#)) and an [brief primer on data center access and visualization with ObsPy](#).

Introduction to ObsPy

- 1. Python Introduction for Seismologists
- 2. UTCDateTime
 - 2.1. Initialization
 - 2.2. Attribute Access
 - 2.3. Handling time differences
 - 2.4. Exercises

Additional Exercise

04_Optional_Exercise.ipynb