Data Magic with Python

a one day workshop with Olve Maudal, EDC Software 2022, Sep 14

What is your background with programming in general?

What is your experience with Python?

What is your expectations for this course?

#fc0243 Data Magic with Python | 8 hours | Cloud | Olve Maudal

>> Title: Data Magic with Python
>> Speaker(s): Olve Maudal
>> Length of session: 8 hours
>> Room: Energy Hall 2 (200)
>> Max # of participants: 50
>> Type of session: workshop
>> Scheduled: Wednesday 0800
>> Description: This is an extended version of a very popular in-house course that we have done over video many times. The focus will still be on Python, Azure, Unix, GitHub, Visual Code - but this time there will be more hands-on exercises done in groups. Here are some of the topics we will cover: - Using a cloud based developing environment (cloudspace or gitpod); - Accessing an Azure Data Storage (a large dataset from Northern Lights); - Working on a Unix command line (kungfu ninja skills); - Writing small Python scripts (getting started); - Developing a proper tool in Python (with discussions about best practices in software); - Spin up a simple web application (using Flask); - Interactive Python (Jupyter, numpy and matplotlib)
>> Requirements: To do the exercises you need a GitHub account connected to the Equinor organization, and you must have some basic skills in Python or another programming language already.
>> Level: beginner/intermediate but experts are also very, very welcome!
>> Extra info: Experienced Python programmers, software engineers and data scientists are particularly welcome and invited to this course - I really need your insight and expertise to add depth to the conversation, but also to help out with the group exercise.
>> Speaker bio: Olve started flirting with Python back in 2004. Olve is teaching both beginner and intermediate/advanced courses in Python on a regular basis.

Data Magic with Python - an introduction

Here are some of the topics we will cover:

- Using a cloud based developing environment
- Accessing an Azure Data Storage
- Working on the Unix command line
- Writing small Python scripts
- Developing a proper tool in Python
- Spin up a simple web application with Flask
- Interactive Python with numpy and matplotlib

Requirement for attending:

- you have a GitHub account connected to the Equinor organization
- you have some basic skill in Python or another programming language already

There will be lunch from 12:00-13:00

Outline

- Welcome (approx 0805)
- Using GitHub and a cloud based developing environment
- Accessing an Azure Data Storage (listfiles.py)
- **Exercise:** Create dev env, list files in data storage, **help others!**
- Working on the Unix command line (grep, wc, head, cut, uniq, sort, sed)
- Develop a proper tool in Python (mylastool.py)
- **Lunch (approx 1200-1300)**
- **Exercise**: Extend mylastool to print out all curve mnemonics of a LAS file
- Spin up a simple web application with Flask (mylaswebapp.py)
- Using matplotlib and segyio (mysegydemo.py)
- Interactive Python with numpy and matplotlib (mynotebook.py)
- Discussion about software development (demoapp.py)
- **End of session** (approx 1600)

https://github.com/equinor/

Equinor | WELP | NES | IT | GH | Governance | Omnia | Gathering | KM | COP | Azure | Stuff | Gov | ETS | aoc | Other favorit

Search or jump to... / Pull requests Issues Marketplace Explore

Equinor

http://www.equinor.com Part of Equinor ASA

Overview | Repositories 2.7k | Packages | People 2.1k | Teams 625 | Projects 83 | Insights

Pinned

design-system Public
The Equinor design system
TypeScript 47 41

dlisio Public
Python library for working with the well log formats Digital Log Interchange Standard (DLIS V1) and Log Information Standard (LIS79)
Python 76 31

segyio-notebooks Public
Notebooks with examples and demos of segyio
Jupyter Notebook 77 56

segyio Public
Fast Python library for SEGY files.
Python 355 182

xtgeo Public
XTGeo Python class library for subsurface Surfaces, Cubes, Wells, Grids, Points, etc
Python 74 32

configsuite Public
Config Suite is the result of recognizing the complexity of software configuration.
Python 7 13

Repositories

Find a repository... | Type | Language | Sort | New

energyvision Public
Home to the next version of the equinor.com website
TypeScript 14 MIT 5 195 (1 issue needs help) 6 Updated 34 seconds ago

everest Internal
Python 2 12 20 1 Updated 1 minute ago

People

View all

Top languages

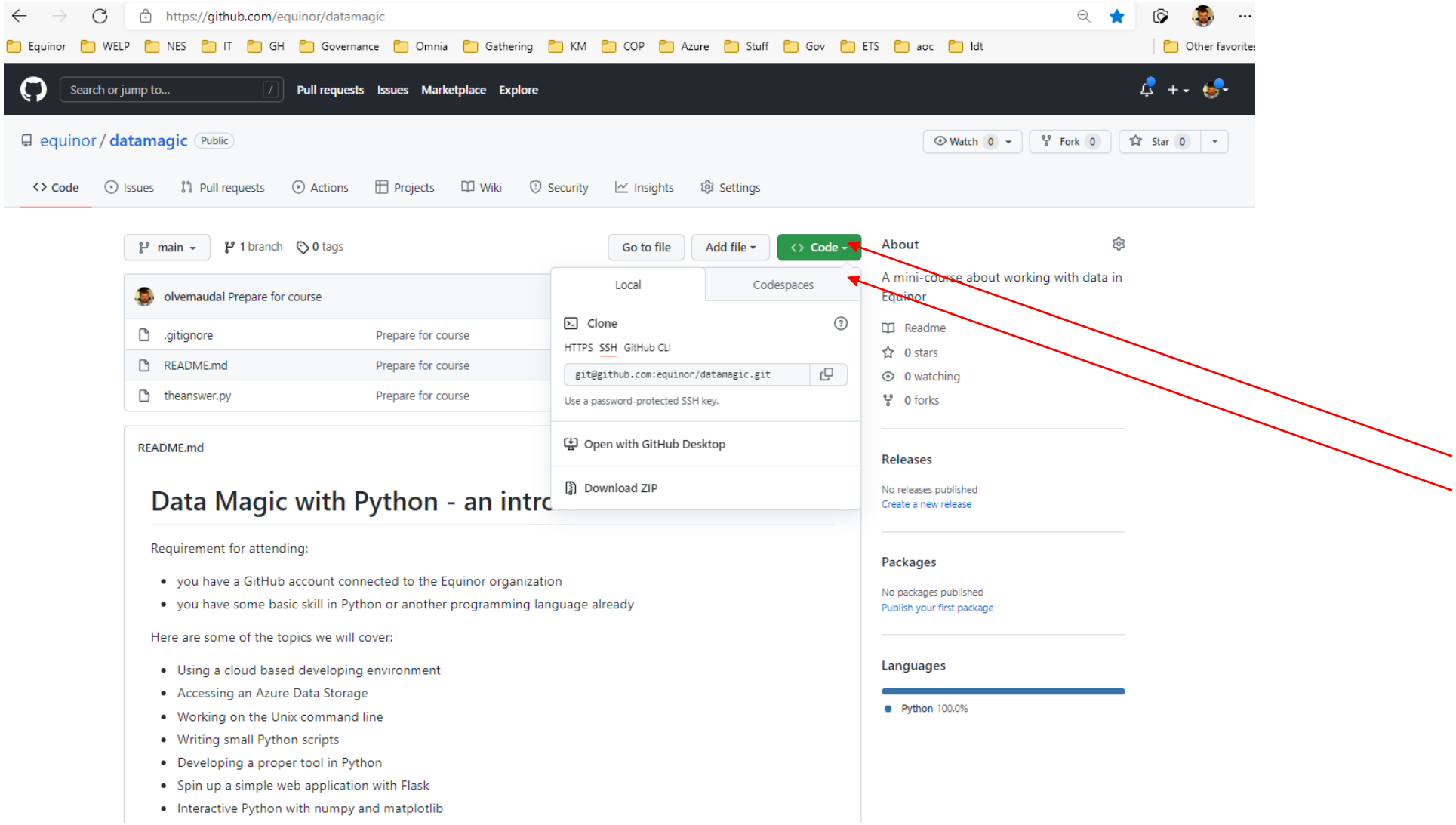
Python C# JavaScript Shell TypeScript

Most used topics

Manage

hacktoberfest python radix webviz rock-n-log

https://github.com/equinor/datamagic



equinor/datamagic: A mini-cours | theanswer.py - datamagic [Code

https://olvemaudal-equinor-datamagic-jwqv672pj7j.github.dev

Equinor | WELP | NES | IT | GH | Governance | Omnia | Gathering | KM | COP | Azure | Stuff | Gov | ETS | aoc | Idt | Other favorites

EXPLORER

DATAMAGIC [CODESPACES]

- .venv
- .gitignore
- README.md
- theanswer.py

theanswer.py

theanswer.py

```python
a = 6
b = 9
c = a * (b - 3)
print(f"The answer is {c}!")
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

bash

```
@olvemaudal ➜ /workspaces/datamagic (main) $
```

Do you want to install the recommended extensions for Python?

Install | Show Recommendations

OUTLINE

TIMELINE

Codespaces | main | 0 ⚠ 0 | 0

Ln 1, Col 1 | Spaces: 4 | UTF-8 | LF | Python | Layout: Norwegian

https://olvemaudal-equinor-datamagic-jwqv672pj7j.github.dev

Equinor | WELP | NES | IT | GH | Governance | Omnia | Gathering | KM | COP | Azure | Stuff | Gov | ETS | aoc | Idt | Other favorites

EXPLORER

DATAMAGIC [CODESPACES]

- .venv
- .gitignore
- README.md
- theanswer.py

Extension: Python | theanswer.py

theanswer.py > a

```python
a = 6
b = 9
c = a * (b - 3)
print(f"The answer is {c}!")
```

PROBLEMS 4 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
source /workspaces/datamagic/.venv/bin/activate
/workspaces/datamagic/.venv/bin/python /workspaces/datamagic/theanswer.py
@olvemaudal ➜/workspaces/datamagic (main) $ source /workspaces/datamagic/.venv/bin/activate
(.venv) @olvemaudal ➜/workspaces/datamagic (main) $ /workspaces/datamagic/.venv/bin/python /workspaces/datamagic/theanswer.py
The answer is 36!
(.venv) @olvemaudal ➜/workspaces/datamagic (main) $ python theanswer.py
The answer is 36!
(.venv) @olvemaudal ➜/workspaces/datamagic (main) $
```

bash | Python

OUTLINE | TIMELINE

Codespaces | main | 0 0 4 | 0 | Ln 1, Col 1 | Spaces: 4 | UTF-8 | LF | {} Python | Layout: Norwegian

File Edit Selection View Go Run Terminal Help theanswer.py - datamagic [Codespaces] - Visual Studio Code

EXPLORER

DATAMAGIC [CODESPACES]

- .venv
- .gitignore
- README.md
- theanswer.py

theanswer.py > a

```python
a = 6
b = 9
c = a * (b - 3)
print(f"The answer is {c}!")
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
source /workspaces/datamagic/.venv/bin/activate
/workspaces/datamagic/.venv/bin/python /workspaces/datamagic/theanswer.py
@olvemaudal ➜ /workspaces/datamagic (main) $ source /workspaces/datamagic/.venv/bin/activate
(.venv) @olvemaudal ➜ /workspaces/datamagic (main) $ /workspaces/datamagic/.venv/bin/python /workspaces/datamagic/theanswer.py
The answer is 36!
(.venv) @olvemaudal ➜ /workspaces/datamagic (main) $
```

OUTLINE

TIMELINE

Codespaces main ⊗ 0 ⚠ 0 ⓘ 4 0 Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.8.12 ('.venv': venv)

```
$ cd ~
$ cd sb
$ git clone https://github.com/equinor/datamagic
Cloning into 'datamagic'...
remote: Enumerating objects: 174, done.
remote: Counting objects: 100% (174/174), done.
remote: Compressing objects: 100% (121/121), done.
remote: Total 174 (delta 86), reused 137 (delta 52), pack-reused 0
Receiving objects: 100% (174/174), 259.37 KiB | 5.29 MiB/s, done.
Resolving deltas: 100% (86/86), done.
$ cd datamagic
$ ls -al
total 25
drwxr-xr-x 1 OLVM 1049089    0 Mar 14 09:46 ./
drwxr-xr-x 1 OLVM 1049089    0 Mar 14 09:46 ../
drwxr-xr-x 1 OLVM 1049089    0 Mar 14 09:46 .git/
-rw-r--r-- 1 OLVM 1049089 1973 Mar 14 09:46 .gitignore
-rw-r--r-- 1 OLVM 1049089 1067 Mar 14 09:46 README.md
-rw-r--r-- 1 OLVM 1049089   61 Mar 14 09:46 theanswer.py
$ cat theanswer.py
a = 6
b = 9
c = a * (b - 3)
print(f"The answer is {c}!")
$ python theanswer.py
The answer is 36!
$
```

(while it is possible to install locally, I encourage you to use a cloud based development environment instead)

https://gitpod.io/#https://github.com/equinor/datamagic

theanswer.py - datamagic - Gi

https://equinor-datamagic-22o0xraxgbw.ws-eu34.gitpod.io

EXPLORER

DATAMAGIC

.gitignore

README.md

theanswer.py

Extension: Python

theanswer.py

```
a = 6
b = 9
c = a * (b - 3)
print(f"The answer is {c}!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
pyenv shell 3.8.12
/home/gitpod/.pyenv/versions/3.8.12/bin/python /workspace/datamagic/theanswer.py
gitpod /workspace/datamagic (main) $ pyenv shell 3.8.12
gitpod /workspace/datamagic (main) $ /home/gitpod/.pyenv/versions/3.8.12/bin/python /workspace/datamagic/theanswer.py
The answer is 36!
gitpod /workspace/datamagic (main) $
```

bash

Python

ALTERNATIVE

OUTLINE

No Notifications

Gitpod Python 3.8.12 64-bit ('3.8.12': pyenv) ⊗ 0 ⚠ 0 Ln 5, Col 1 Spaces: 4 UTF-8 LF Python Layout: US No open ports

(when asked, install the Python extension)

theanswer.py

```
a = 6
b = 9
c = a * (b - 2)
print(f"The answer is {c}!")
```

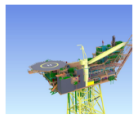
```
$ python
Python 3.8.12 (default, Oct  4 2021, 15:56:52)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 6 * 7
42
>>> quit()
$ python theanswer.py
The answer is 42!
$
```

https://data.equinor.com/

equinor

Forum FAQ Oggi Stindberg

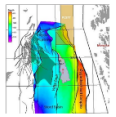
Find open data



Plant 3D models

01/11/21

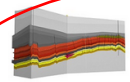
Equinor has a large portfolio of 3D models representing all our topside assets. To enable innovations within new ways of working and also use of 3D models within learning institutions, we will now share the 3D model of our Huldra asset.



Smeaheia dataset

11/03/21

The Smeaheia dataset is a reference dataset containing subsurface data, reports and geomodels related to assessment of proposed CO2 storage sites in the Smeaheia region, Hordaland Platform offshore Norway.



Northern Lights

28/09/20

The Northern Lights partnership releases all relevant well data and reports from 31/5-7 CCS verification well in exploitation license EL001.

Read more Northern Lights Homepage

Sleipner CO2 reference dataset

03/02/20

Equinor and its partners will disclose datasets from the Sleipner field; the world's first offshore CCS plant, in a push to advance innovation and development on the field of CO2 storage.

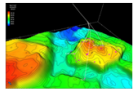
Read more

Hywind Scotland operational data

28/11/19

Equinor and ORE Catapult collaborating to share Hywind Scotland operational data.

Read more

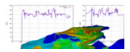


Volve Data Village

14/06/18

For the first time all subsurface and production data from a field on the Norwegian continental shelf (NCS) will be disclosed.

Read more



Norne benchmark case

23/04/18

Equinor and the the Norne license partners has released a subsurface dataset for educational

equinor

Forum FAQ Oggi Stindberg

Northern Lights

Last update: 23/10/20
Topic: Northern Lights - Dataset from NO 31/5-7 (Eos) well

The data is published under TERMS AND CONDITIONS FOR USE OF LICENSE TO DATA.

The Norwegian Full-Scale CCS project is the first industry scale project for capturing CO2 from industrial sources and storage on the Norwegian continental shelf.

In January 2019, Exploitation license EL001 for CO2 storage, located south of the Troll field was awarded to Equinor and the confirmation well 31/5-7 (Eos) was drilled between December 2019 and March 2020 with Shell and Total as partners. The aim was to confirm that EL001 is suitable for CO2 storage.

The CO2 storage complex is defined by the Lower Jurassic Dunlin Group at depths between approximately 2510 to 2830m below sea level at the Eos well location. The Eos well targeted the Dunlin Gp. as the primary storage, where the sandstone-bearing Cook and Johansen Formations both serve as storage units for the injected CO2. They are capped by the Drake Formation consisting of impermeable claystones, preventing the CO2 from migrating out of the Dunlin Group vertically.

The Northern Lights partnership releases all relevant well data and reports from 31/5-7 CCS verification well in exploitation license EL001.

The dataset contains log data from all sections, biostratigraphy, formation pressure, fluid data and well test data among many others.

More

Disclaimer

The Northern Lights dataset is released from October 2020. When asking for access to the data, name, email-address, company and date of download will be registered. The said information might be used for further contact with you or the company that you represent, in order for Equinor to assess the utility of the sharing, invite to further development, experience-sharing, conferences/"hackathons" etc. The personal data will be kept for up to three years. For further information, please see Equinor's privacy policy available on equinor.com

The users of the information and data are expressly informed that the data provided by Equinor ASA and collaboration partners Total E&P Norge AS and A/S Norske Shell is provided on an "as is" basis and may contain errors and omissions. Equinor ASA and collaboration partners Total E&P Norge AS and A/S Norske Shell provide no warranties, expressed or implied, either relating to the content or to the relevance of the data. Equinor disclaims any liability for errors and defects associated with the data to the maximum extent permitted by law. Equinor ASA and collaboration partners Total E&P Norge AS and A/S Norske Shell shall not be liable for any direct or indirect losses as a result of use of the data, or concerning possible infringement of any intellectual property and/or patent rights of a third party.

The data set shall not be resold.

The data is published under TERMS AND CONDITIONS FOR USE OF LICENSE TO DATA

GO BACK SUBMIT

Northern Lights

Last update: 23/10/20
Topic: Northern Lights - Dataset from NO 31/5-7 (Eos) well

The data is published under TERMS AND CONDITIONS FOR USE OF LICENSE TO DATA.

The Norwegian Full-Scale CCS project is the first industry scale project for capturing CO2 from industrial sources and storage on the Norwegian continental shelf.

In January 2019, Exploitation license EL001 for CO2 storage, located south of the Troll field was awarded to Equinor and the confirmation well 31/5-7 (Eos) was drilled between December 2019 and March 2020 with Shell and Total as partners. The aim was to confirm that EL001 is suitable for CO2 storage.

The CO2 storage complex is defined by the Lower Jurassic Dunlin Group at depths between approximately 2510 to 2830m below sea level at the Eos well location. The Eos well targeted the Dunlin Gp. as the primary storage, where the sandstone-bearing Cook and Johansen Formations both serve as storage units for the injected CO2. They are capped by the Drake Formation consisting of impermeable claystones, preventing the CO2 from migrating out of the Dunlin Group vertically.

The Northern Lights partnership releases all relevant well data and reports from 31/5-7 CCS verification well in exploitation license EL001.

The dataset contains log data from all sections, biostratigraphy, formation pressure, fluid data and well test data among many others.

More

Data links

Usage

We recommend using *Azure Storage Explorer* when exploring the content of, or parts of this dataset. Using this tool will enable you to explore and download only the parts you are interested in. Follow instructions here for download and installation

For connecting to this dataset in Azure Storage Explorer, use the shared access signature URI:

https://datavillagesa.blob.core.windows.net/northernlights?sv=2018-03-28&sr=c&sig=b0R9[redacted]yE%3D&se=2022-03-16T09%3A37%3A06Z&sp=rl

https://datavillagesa.blob.core.windows.net/northernlights?sv=2018-03-28&sr=c&sig=VTWTxWY%2BTXXX8Y3...

listfiles.py

```
import os
import azure.storage.blob

URL = os.environ['CONTAINER_URL']

container = azure.storage.blob.ContainerClient.from_container_url(URL)

for blob in container.list_blobs():
    print(f'{blob.size:<20}{blob.name}')
```

```
$ pip install azure-storage-blob
$ export CONTAINER_URL="https://datavillagesa.blob.core.windows.net/northernlights?sv=2018-03-28&sr=c&sig=VTWTxWY%2BTXXX8Y3...
$ python listfiles.py > files.dat
$ wc -l files.dat
843 files.dat
$ head -5 files.dat
654174              31_5-7 Eos/02.Drilling_and_Completion/CORING_2020-01-14_REPORT_1.PDF
2392571             31_5-7 Eos/03.Directional_Surveys/WELLPATH_COMPUTED_1.ASC
7985                31_5-7 Eos/03.Directional_Surveys/WELLPATH_ORIGINAL_SURVEY_POINTS_1.ASC
1303736             31_5-7 Eos/05.LWD_Log_data/WLC_RAW_BHPR-GR-MECH_TIME_MWD_PLOT_1.PDF
13844987            31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_MD_MWD_PLOT_1.PDF
$ tail -12 files.dat
12107352            31_5-7 Eos/13.Petrophysical_Data_Evaluations/Composite/WLC_PETROPHYSICAL_COMPOSITE_1.DLIS
382845              31_5-7 Eos/13.Petrophysical_Data_Evaluations/Composite/WLC_PETROPHYSICAL_COMPOSITE_1_INF_1.PDF
11996675            31_5-7 Eos/14.Final_Well_Report_and_Completion_Log/C236-DO-U-RY-00001_02_I_As built report.pdf
1879                31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC-AIMG-CCL-GR_2020-01-17_1-3_INF_1.ASC
287293204           31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC-AIMG-CCL-GR_2020-01-17_1.DLIS
60909688            31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC-AIMG-CCL-GR_2020-01-17_2.DLIS
62413600            31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC-AIMG-CCL-GR_2020-01-17_3.DLIS
16765648            31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC-AIMG-CCL-GR_2020-01-17_PLOT_1.PDF
1222896388          31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC_2019-12-26_1.DLIS
1258                31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC_2019-12-26_1_INF_1.ASC
8467628             31_5-7 Eos/15.Production_Logs/WL_RAW_PROD_AAC_2019-12-26_PLOT_1.PDF
244                 31_5-7 Eos/Read me.txt
$
```

Group Exercise

Step 1:

- Visit https://github.com/equinor/datamagic
- Create a Codespace, install python extension when asked
- Run theanswer.py, check output
- Edit theanswer.py so it computes the right value
- Help everyone else in your group to also get up to speed!

Step 2:

- Goto data.equinor.com, get your personal URL to Northern Lights
- Run listfiles.py

Step 3 (optional):

- Try to calculate some statistics on the list of files in the dataset
- Eg, how many files? How many bytes? How many LAS-files?

files.dat

```
654174              31_5-7 Eos/02.Drilling_and_Completion/CORING_2020-01-14_REPORT_1.PDF
2392571             31_5-7 Eos/03.Directional_Surveys/WELLPATH_COMPUTED_1.ASC
7985                31_5-7 Eos/03.Directional_Surveys/WELLPATH_ORIGINAL_SURVEY_POINTS_1.ASC
1303736             31_5-7 Eos/05.LWD_Log_data/WLC_RAW_BHPR-GR-MECH_TIME_MWD_PLOT_1.PDF
13844987            31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_MD_MWD_PLOT_1.PDF
7001843             31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_MD_MWD_PLOT_2.PDF
3176542             31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_MWD_1.DLIS
1283                31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_MWD_1_INF_1.ASC
7129400             31_5-7 Eos/05.LWD_Log_data/WLC_RAW_CAL-DEN-GR-NEU-REMP_TVD_MWD_PLOT_1.PDF
894926              31_5-7 Eos/05.LWD_Log_data/WLC_RAW_GR-REMP-RLL_MD_MWD_PLOT_1.PDF
    ... 818 files not shown ...
292                 31_5-7 Eos/12.Geology_Data_and_Evaluations/Biostratigraphy/BIOSTRAT_COMPUTED_1_INF_1.ASC
1155879262          31_5-7 Eos/12.Geology_Data_and_Evaluations/Image log processed/NO_31_5-7___STAT_IMAGE_RES.dlis
2593                31_5-7 Eos/13.Petrophysical_Data_Evaluations/CPI/WLC_PETRO_COMPUTED_1_INF_1.ASC
4649037             31_5-7 Eo
680275              31_5-7 Eo
1159572             31_5-7 Eo
351619              31_5-7 Eo
8075509             31_5-7 Eo
12107352            31_5-7 Eo
382845              31_5-7 Eo
11996675            31_5-7 Eo
1879                31_5-7 Eo
287293204           31_5-7 Eo
60909688            31_5-7 Eo
62413600            31_5-7 Eo
16765648            31_5-7 Eo
1222896388          31_5-7 Eo
1258                31_5-7 Eo
8467628             31_5-7 Eo
244                 315-7 Eos
```

```
$ cat files.dat | wc -l
843
$ cat files.dat | grep -i "\.las$" | wc -l
19
$ cat files.dat | grep -i "\.las$" | cut -f 1 -d' ' | xargs echo
1184900 1720411 1965200 1904013 3342200 1139000 710600 679490 1139000 585275 2877827 108614 55181 77005 77033 513552 105624 4649037 680275
$ cat files.dat | grep -i "\.las$" | cut -f 1 -d' ' | xargs echo | tr ' ' '+'
1184900+1720411+1965200+1904013+3342200+1139000+710600+679490+1139000+585275+2877827+108614+55181+77005+77033+513552+105624+4649037+680275
$ cat files.dat | grep -i "\.las$" | cut -f 1 -d' ' | xargs echo | tr ' ' '+' | bc
23514237
$ N=$(cat files.dat | grep -i "\.las$" | cut -f 1 -d' ' | xargs echo | tr ' ' + | bc)
$ echo "$N / (2^20)" | bc
22
$ $ cat files.dat | rev | cut -f1 -d. | rev | sort | uniq -c | sort -nr
    536 TIF
     96 DLIS
     83 ASC
     70 PDF
     31 SEGY
     18 LAS
      2 dlis
      1 xlsx
      1 txt
      1 SPWLA
      1 sbg
      1 pdf
      1 las
      1 DEX
$
```

mylastool.py (step 0)

```
import os
import azure.storage.blob

def get_container():
    url = os.environ['CONTAINER_URL']
    return azure.storage.blob.ContainerClient.from_container_url(url)

def get_list_of_lasfiles(container):
    """Get list of LAS files in a container."""
    files = []
    for blob in container.list_blobs():
        if blob.name.endswith('.LAS'):
            files.append(blob.name)
    return files


def print_list_of_lasfiles(container):
    """Print pretty directory listing LAS file in container."""
    for name in get_list_of_lasfiles(container):
        print(name)

def main():
    container = get_container()
    print_list_of_lasfiles(container)

if __name__ == '__main__':
    main()
```

mylastool.py (step 1)

```
...

def read_lasfile(container, filename):
    """Read given LAS file from container."""
    if not filename.endswith('.LAS'):
        raise OSError("Probably not a LAS file")
    blob_client = container.get_blob_client(filename)
    data = blob_client.download_blob().content_as_bytes()
    lines = []
    for line in data.splitlines():
        lines.append(line.decode("ascii", errors='ignore'))
    return lines

def main():
    container = get_container()
    #print_directory(container)

    lasfile = '31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_2.LAS'
    lines = read_lasfile(container, lasfile)
    for line in lines:
        print(line)

if __name__ == '__main__':
    main()
```

mylastool.py (step 2)

```
...

def print_header_section(lines):
    for line in lines:
        if line.startswith('~A'):
            break
        print(line)

def print_data_section(lines):
    idx = 0
    for line in lines:
        if line.startswith('~A'):
            break
        idx += 1
    for line in lines[idx+1:]:
        print(line)

def main():
    container = get_container()
    #print_directory(container)

    lasfile = '31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_2.LAS'
    lines = read_lasfile(container, lasfile)
    print_header_section(lines)
    print_data_section(lines)

if __name__ == '__main__':
    main()
```

mylastool.py (step 3)

```
...

def find_section_index(lines, prefix):
    """Find index of first line with given prefix."""
    idx = 0
    for line in lines:
        if line.startswith(prefix):
            break
        idx += 1
    return idx


def get_header_section(lines):
    """Return the lines for the header section."""
    return lines[:find_section_index(lines, '~A')]


def get_data_section(lines):
    """Return the lines for the data section."""
    return lines[find_section_index(lines, '~A')+1:]


def print_header_section(lines):
    """Print the header section."""
    for line in get_header_section(lines):
        print(line)


def print_data_section(lines):
    """Print the data section."""
    for line in get_data_section(lines):
        print(line)

...
```

mylastool.py (step 4)

```
def print_helpmessage():
    """Print help message."""
    print("usage: mylastool.py <command> [file]")
    print("examples:")
    print("    python mylastool.py list")
    print("    python mylastool.py header A/B/C.LAS")
    print("    python mylastool.py data   A/B/C.LAS")
    print("also, remember to set CONTAINER_URL")
```

```
def main(argv):
    """Parse as list of arguments and do magic."""
    if len(argv) < 2:
        print_helpmessage()
        return 1

    command = argv[1]

    if command not in ('list', 'header', 'data'):
        print('error: unknown command')
        print_helpmessage()
        return 1

    container = get_container()

    if command == 'list':
        print_list_of_lasfiles(container)
        return 0

    if len(argv) < 3:
        print('error: expected a filename')
        print_helpmessage()
        return 1

    lasfile = argv[2]
    lines = read_lasfile(container, lasfile)

    if command == 'header':
        print_header_section(lines)
        return 0

    if command == 'data':
        print_data_section(lines)
        return 0

    print('Huh?')
    print_helpmessage()
    return 1


if __name__ == '__main__':
    sys.exit(main(sys.argv))
```

Group Exercise

- Download mylastool.py, try to run it and then discuss it
- As a group, discuss how to write a script to only list the curve mnemonics of a given LAS file
- Discuss the design of a script that could scan through all the LAS files in a storage container, parse the headers, find the curvemnemonics, and then create an overview of all curves present in that particular storage container.
- Once everyone is "onboard" on the design, develop the code **together**

```
XWELL .M          524299.53                          :X-coordinate of Well Hea
YWELL .M          6715849.60                         :Y-coordinate of Well Hea
~Curve Information Block
#MNEM.UNIT        API Code                 Curve     Description
#--------         --------------           -----     ------------------
Time      .S                               :0        Index TWT referred to MSL
SMUO_45Hz .                                :1        Zero Phase 45Hz Ricker Sy
SMUO_35Hz .                                :2        Zero Phase 35Hz Ricker Sy
SMUO_25Hz .                                :3        Zero Phase 25Hz Ricker Sy
SMUO_20Hz .                                :4        Zero Phase 20Hz Ricker Sy
~Ascii Data Section
    1.84600      -999.25000     -999.25000     -999.25000       0.00000
    1.84700      -999.25000     -999.25000     -999.25000       0.00000
```

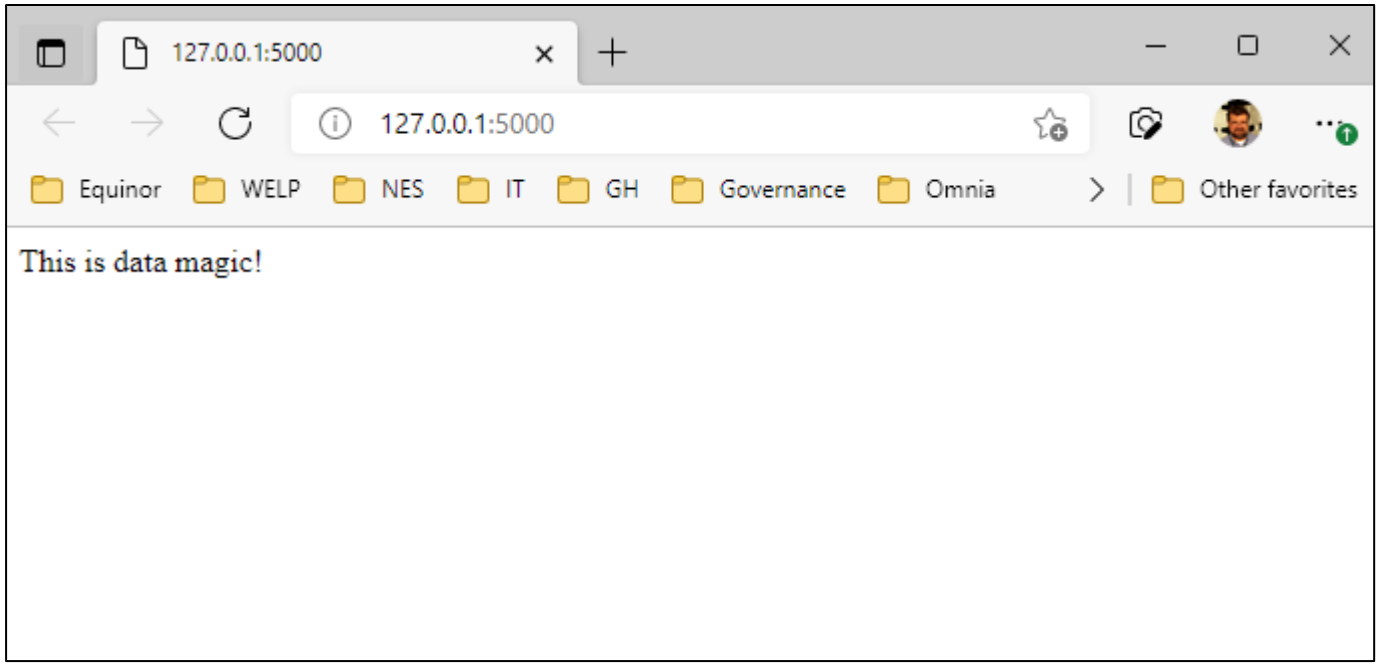
mylastool.py (step 5)

```
def get_curve_section(lines):
    """Return the lines for the curve section."""
    start_idx = find_section_index(lines, '~C')
    end_idx = find_section_index(lines[start_idx+1:], '~')
    return lines[start_idx+1:start_idx+1+end_idx]


def get_curve_mnemonics(lines):
    """Get a list of curve names."""
    names = []
    for line in get_curve_section(lines):
        if line.strip().startswith('#'):
            continue
        names.append(line.split('.')[0].strip())
    return names


def print_curve_mnemonics(lines):
    """Pretty print the curve names."""
    print(*get_curve_mnemonics(lines), sep=' | ')
```

```python
"""My tool for working with LAS files in an azure storage container."""

import os
import sys
import azure.storage.blob


def get_container():
    """Creating container from CONTAINER_URL."""
    url = os.environ['CONTAINER_URL']
    return azure.storage.blob.ContainerClient.from_container_url(url)


def get_list_of_lasfiles(container):
    """Get list of LAS files in a container."""
    files = []
    for blob in container.list_blobs():
        if blob.name.endswith('.LAS'):
            files.append(blob.name)
    return files


def print_list_of_lasfiles(container):
    """Print pretty directory listing LAS file in container."""
    files = get_list_of_lasfiles(container)
    for name in files:
        print(name)


def read_lasfile(container, filename):
    """Read given LAS file from container."""
    if not filename.endswith('.LAS'):
        raise OSError("Probably not a LAS file")
    blob_client = container.get_blob_client(filename)
    data = blob_client.download_blob().content_as_bytes()
    lines = []
    for line in data.splitlines():
        lines.append(line.decode("ascii", errors='ignore'))
    return lines


def find_section_index(lines, prefix):
    """Find index of first line with given prefix."""
    idx = 0
    for line in lines:
        if line.startswith(prefix):
            break
        idx += 1
    return idx


def get_header_section(lines):
    """Return the lines for the header section."""
    return lines[:find_section_index(lines, '~A')]


def get_data_section(lines):
    """Return the lines for the data section."""
    return lines[find_section_index(lines, '~A')+1:]


def print_header_section(lines):
    """Print the header section."""
    for line in get_header_section(lines):
        print(line)


def print_data_section(lines):
    """Print the data section."""
    for line in get_data_section(lines):
        print(line)


def get_curve_section(lines):
    """Return the lines for the curve section."""
    start_idx = find_section_index(lines, '~C')
    end_idx = find_section_index(lines[start_idx+1:], '~')
    return lines[start_idx+1:start_idx+1+end_idx]


def get_curve_mnemonics(lines):
    """Get a list of curve names."""
    names = []
    for line in get_curve_section(lines):
        if line.strip().startswith('#'):
            continue
        names.append(line.split('.')[0].strip())
    return names


def print_curve_mnemonics(lines):
    """Pretty print the curve names."""
    print(*get_curve_mnemonics(lines), sep=' | ')


def print_helpmessage():
    """Print help message."""
    print("usage: mylastool.py <command> [file]")
    print("examples:")
    print("    python mylastool.py list")
    print("    python mylastool.py header A/B/C.LAS")
    print("    python mylastool.py data   A/B/C.LAS")
    print("    python mylastool.py curves A/B/C.LAS")
    print("also, remember to set CONTAINER_URL")


def main(argv):
    """Parse a list of arguments and do magic."""

    if len(argv) < 2:
        print_helpmessage()
        return 1

    command = argv[1]

    if command not in ('list', 'header', 'data', 'curves'):
        print('error: unknown command')
        print_helpmessage()
        return 1

    container = get_container()

    if command == 'list':
        print_list_of_lasfiles(container)
        return 0

    if len(argv) < 3:
        print('error: expected a filename')
        print_helpmessage()
        return 1

    lasfile = argv[2]
    lines = read_lasfile(container, lasfile)

    if command == 'header':
        print_header_section(lines)
        return 0

    if command == 'data':
        print_data_section(lines)
        return 0

    if command == 'curves':
        print_curve_mnemonics(lines)
        return 0

    print('Huh?')
    print_helpmessage()
    return 1


if __name__ == '__main__':
    sys.exit(main(sys.argv))

# References:
# https://www.cwls.org/wp-content/uploads/2017/02/Las2_Update_Feb2017.pdf
# https://docs.microsoft.com/en-us/python/api/azure-storage-blob/?view=azure-python
```

```python
import flask

app = flask.Flask(__name__)

@app.route('/')
def index():
    return "This is data magic!"
```

```
$ flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [14/Feb/2022 20:30:03] "GET / HTTP/1.1" 200 -
```



```python
"""My webapp showing all LAS files in an azure storage container."""

import flask
import mylastool

app = flask.Flask(__name__)

container = mylastool.get_container()
lasfiles = mylastool.get_list_of_lasfiles(container)


@app.route('/')
def index():
    """Build and return a list of all files in storage account."""
    lines = []
    lines.append('<!doctype html>')
    lines.append('<html lang=en>')
    lines.append('<head><meta charset=utf-8><title>My webapp</title></head>')
    lines.append('<body>')
    lines.append('<pre>')
    for (idx, filename) in enumerate(lasfiles):
        lines.append(f'<a href="/header/{idx}">{filename}</a>')
    lines.append('</pre>')
    lines.append('</body>')
    lines.append('</html>')
    return '\n'.join(lines)


@app.route('/header/<idx>')
def header(idx):
    """Return header of LAS file from container."""
    filename = lasfiles[int(idx)]
    lasfile = mylastool.read_lasfile(container, filename)
    headersection = mylastool.get_header_section(lasfile)
    lines = []
    lines.append('<!doctype html>')
    lines.append('<html lang=en>')
    lines.append('<head><meta charset=utf-8><title>My webapp</title></head>')
    lines.append('<body>')
    lines.append('<pre>')
    lines.extend(headersection)
    lines.append('</pre>')
    lines.append('</body>')
    lines.append('</html>')
    return '\n'.join(lines)


def main():
    """Run main function of the program."""
    app.debug = True
    app.run()


if __name__ == '__main__':
    main()
```

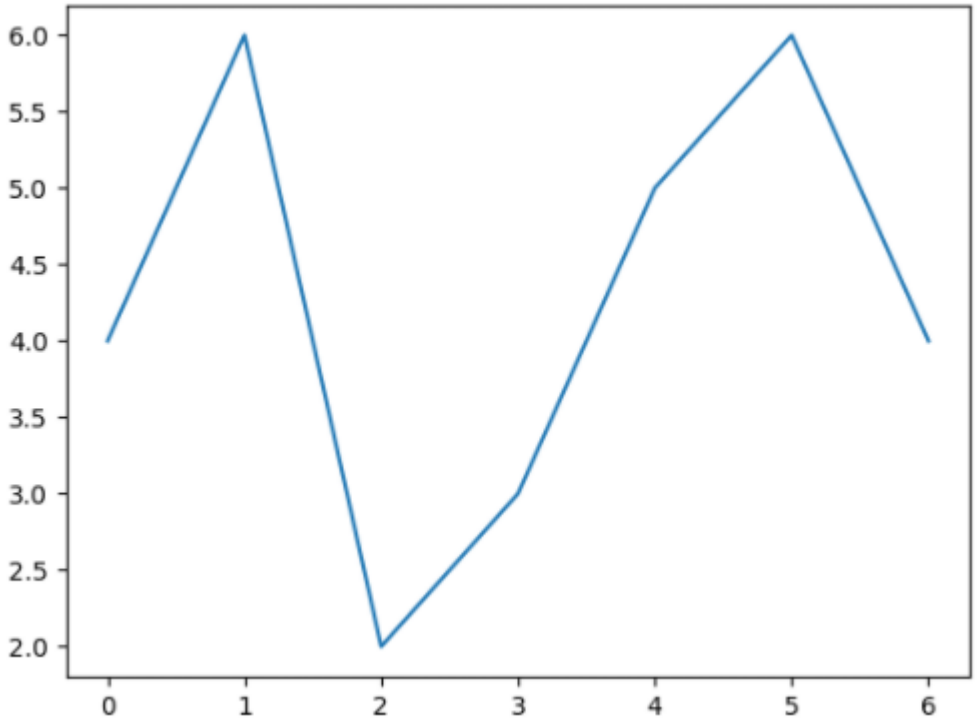
Exercise

- Download mylaswebapp.py, try to run it
- Try to modify the app so that it prints out the curve mnemonics only, instead of the whole header

plotdemo.py

```
import matplotlib.pyplot as plt

numbers = [4, 6, 2, 3, 5, 6, 4]
plt.plot(numbers)
plt.savefig('plotdemo.png')
```

```
$ pip install matplotlib
$ python plotdemo.py
$ ls plotdemo*
plotdemo.png  plotdemo.py
$
```



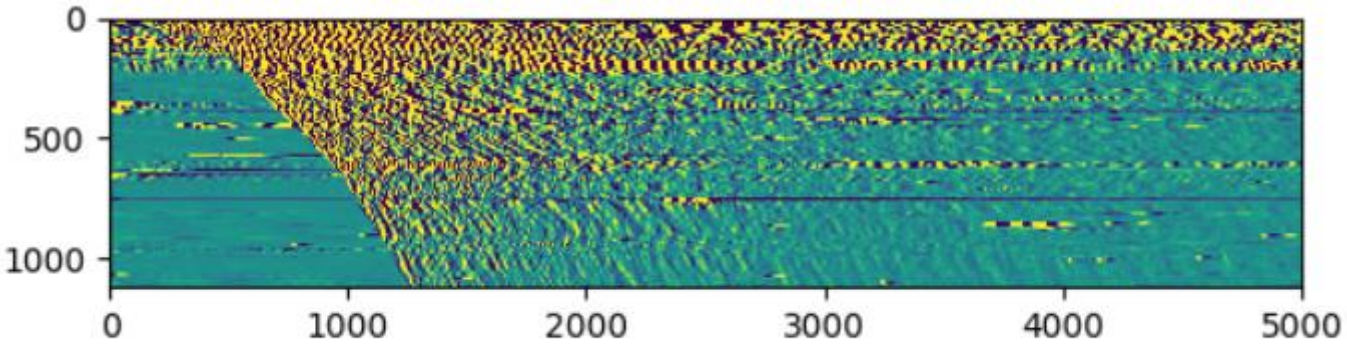
downloadfile.py

```
import os
import azure.storage.blob

url = os.environ['CONTAINER_URL']
container = azure.storage.blob.ContainerClient.from_container_url(url)

path = "31_5-7 Eos/07.Borehole_Seismic/VSPZO_RAW_2020-01-17_4.SEGY"
blob_client = container.get_blob_client(path)

data = blob_client.download_blob().readall()

filename = os.path.basename(path)
with open(filename, "wb") as file:
    file.write(data)
```

plotsegydemo.py

```
import segyio
import matplotlib.pyplot as plt

traces = []
filename = 'VSPZO_RAW_2020-01-17_4.SEGY'
with segyio.open(filename, strict=False) as f:
    for trace in f.trace:
        traces.append(list(trace))
plt.imshow(traces, vmin=-0.01, vmax=0.01)
plt.savefig('plotsegydemo.png')
```

```
$ pip install segyio
$ python plotsegydemo.py
$ ls plotsegydemo*
plotsegydemo.png  plotsegydemo.py
$
```



datamagic / mynotebook.ipynb

In [3]:

```
import mylastool
import os

container = mylastool.get_container()
mylastool.print_list_of_lasfiles(container)
```

```
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_1.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_2.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_3.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_4.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_5.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_6.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_7.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_8.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_BHPR-GR-MECH_TIME_MWD_9.LAS
31_5-7 Eos/05.LWD_Log_data/WL_RAW_GR-MECH_TIME_MWD_1.LAS
31_5-7 Eos/07.Borehole_Seismic/TZV_DEPTH_MD_COMPUTED_2020-01-17_1.LAS
31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_1.LAS
31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_2.LAS
31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_3.LAS
31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_4.LAS
31_5-7 Eos/07.Borehole_Seismic/VSPZO_COMPUTED_TIME_CS_2020-01-17_5.LAS
31_5-7 Eos/13.Petrophysical_Data_Evaluations/CPI/WLC_PETRO_COMPUTED_INPUT_1.LAS
31_5-7 Eos/13.Petrophysical_Data_Evaluations/CPI/WLC_PETRO_COMPUTED_OUTPUT_1.LAS
```

In [4]:

```
filename = "31_5-7 Eos/07.Borehole_Seismic/TZV_TIME_SYNSEIS_2020-01-17_2.LAS"
lines = mylastool.read_lasfile(container, filename)
print(len(lines))
print(*mylastool.get_header_section(lines), sep='\n')
```

```
759
~Version Information
VERS   .          2.0                           :CWLS Log Ascii Standard - Version 2.0
WRAP   .          NO                            :One line per depth step
~Well Information Block
#MNEM.UNIT        Data Type                     Information
#---------        ---------                     ------------------------------
STRT   .S         1.84600                       :START DEPTH
STOP   .S         2.56700                       :STOP DEPTH
STEP   .S         0.00100                       :INCREMENT
NULL   .          -999.25                       :NULL DATA VALUE
COMP   .          Equinor                       :COMPANY
WELL   .          31/5-7                        :WELL
FLD    .          Eos                           :FIELD
LOC    .                                        :LOCATION
PROV   .                                        :PROVINCE
STAT   .                                        :STATE
CNTY   .                                        :COUNTY
CTRY   .          Norway                        :COUNTRY
DATE   .                                        :DATE
UWI    .          31/5-7                        :UNIQUE WELL UWI
LIC    .                                        :LICENSE NUMBER
API    .                                        :API NUMBER
SRVC   .                                        :SERVICE COMPANY
~Parameter Information Block
#MNEM.UNIT        Value                         Description
#---------        ---------                     -------------
XWELL .M          524299.53                     :X-coordinate of Well Head
YWELL .M          6715849.60                    :Y-coordinate of Well Head
~Curve Information Block
#MNEM.UNIT        API Code                Curve    Description
#--------         --------------          -----    -------------------
Time       .S                             :0       Index TWT referred to MSL
SREF_45Hz .                               :1       Zero Phase 45Hz Ricker Synthetic Seismogram (Convolved Reflection Data)
SREF_35Hz .                               :2       Zero Phase 35Hz Ricker Synthetic Seismogram (Convolved Reflection Data)
SREF_25Hz .                               :3       Zero Phase 25Hz Ricker Synthetic Seismogram (Convolved Reflection Data)
SREF_20Hz .                               :4       Zero Phase 20Hz Ricker Synthetic Seismogram (Convolved Reflection Data)
```

In [11]:

```
values = []
for row in mylastool.get_data_section(lines):
    values.append([float(col) for col in row.split()])
curves = list(zip(*values))

for curve in curves:
    curve_values = [value for value in curve if value != -999.25]
    print(f'min={min(curve_values):<8} max={max(curve_values):<8}')
```

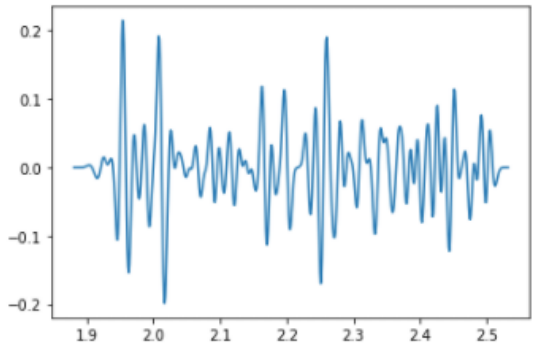
```
min=1.846    max=2.567
min=-0.19864 max=0.21486
min=-0.20577 max=0.20651
min=-0.19672 max=0.1975
min=-0.18145 max=0.17892
```

In [9]:

```
import numpy as np
curves = np.array(values).T
curves = np.where(curves==-999.25, np.nan, curves)

import matplotlib.pyplot as plt
plt.plot(curves[0], curves[1])
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7fbcd474fd00>]
```

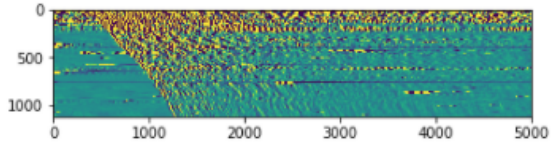


In [10]:

```
import segyio
import matplotlib.pyplot as plt

traces = []
with segyio.open('VSPZO_RAW_2020-01-17_4.SEGY', strict=False) as f:
    for trace in f.trace:
        traces.append(list(trace))
plt.imshow(traces, vmin=-0.01, vmax=0.01)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x7fbc9488a190>
```



!

Microsoft Azure Storage Explorer

File Edit View Help

EXPLORER

Search for resources

Collapse All Refresh All

- Quick Access
- Local & Attached
  - Storage Accounts
    - (Attached Containers)
      - Blob Containers
        - northernlights (SAS)
        - volve (SAS)
      - File Shares
      - Queues
      - Tables
    - (Emulator - Default Ports) (Key)
  - Cosmos DB Accounts (Deprecated)
  - Data Lake Storage Gen1 (Preview)

Actions Properties

URL	https://datavillagesa.blob.core.windows.net/northernlights
Custom Domain	
Type	Blob Container
HNS Enabled	false
Shared Access Signature	••••••••••
Supports Tags	Unknown

northernlights

Upload Download Open New Folder Select All Copy Paste Clone Delete Undelete Manage History Folder Statistics Refresh

Active blobs (default) northernlights > 31_5-7 Eos Show Filter Panel

Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type	Content Type	Size	Status	Rem
02.Drilling_and_Completion					Folder			
03.Directional_Surveys					Folder			
05.LWD_Log_data					Folder			
06.Wireline_Log_Data					Folder			
07.Borehole_Seismic					Folder			
08.Formation_Pressure_Data					Folder			
09.Well_Test_Data					Folder			
10.Fluid_Data					Folder			
11.Core_Data					Folder			
12.Geology_Data_and_Evaluations					Folder			
13.Petrophysical_Data_Evaluations					Folder			
14.Final_Well_Report_and_Completion_Log					Folder			
15.Production_Logs					Folder			
Read me.txt	Hot (inferred)		9/29/2021, 3:07:03 PM	Block Blob	text/plain	244 B	Active	

Showing 1 to 14 of 14 cached items

Activities