Data Science
Survival Skills

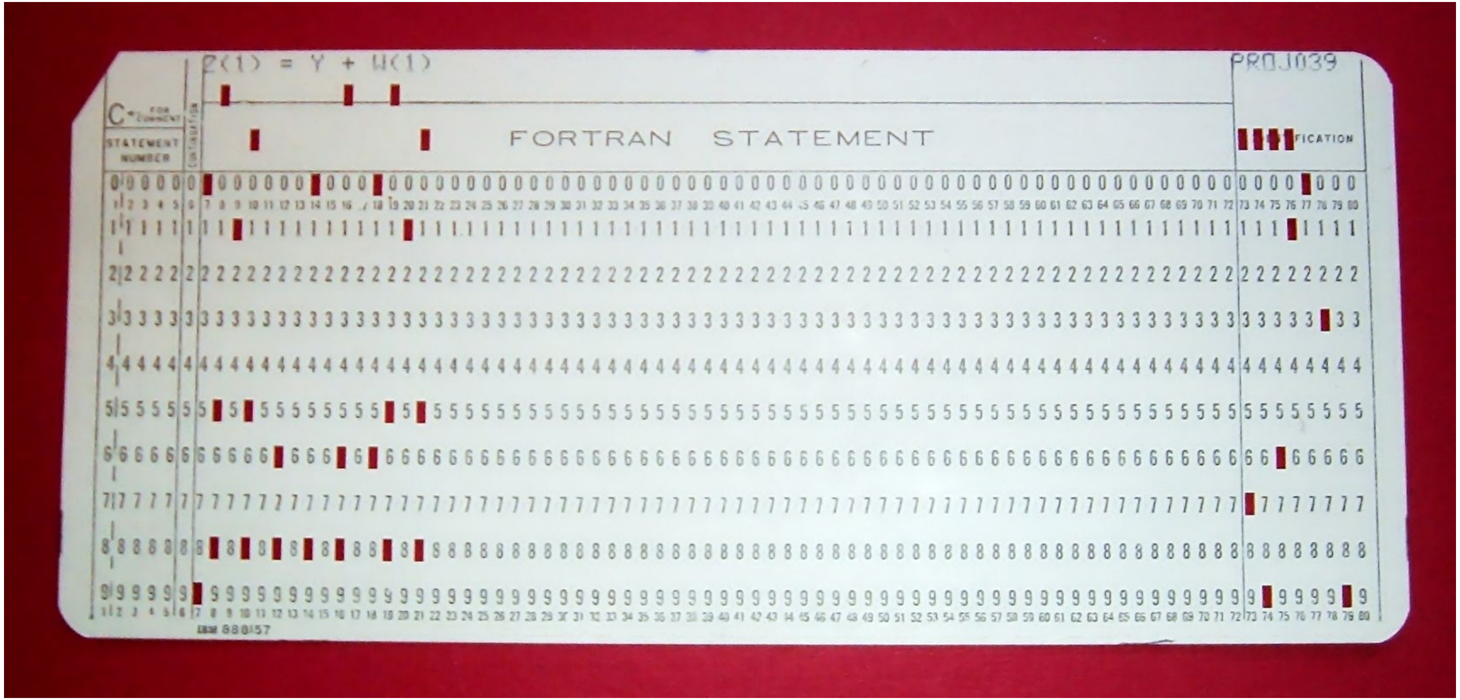
Files

Agenda

- What is a file...?
- How kinds of data do we have?
- How is data stored meaningfully?
- What is lossy/lossless compression?
- What are container formats?

A file

- Entity of content
- Back in the days: punch cards



File identification

- Root/stem ➜ identifier
- Extension ➜ File type
- Path ➜ Location

File root, stem

dir_name

C:\users\OddThinking\Documents\My Source\Widget\foo.src

Basename

Extension

full/absolute path

https://stackoverflow.com/questions/2235173/what-is-the-naming-standard-for-path-components

File size

- Maybe trivial, but it is measured in bytes
- Remember the 4 GB max file size on FAT32?

  2^32 - 1 ➜ 4,294,967,295 ($2^{32} - 1$) bytes, ca 4 GB max

Traditional units				
Name	**Symbol**	**Binary**	**Number of bytes**	**Equal to**
Kilobyte	kB	2^{10}	1,024	1024 B
Megabyte	MB	2^{20}	1,048,576	1024 KB
Gigabyte	GB	2^{30}	1,073,741,824	1024 MB
Terabyte	TB	2^{40}	1,099,511,627,776	1024 GB
Petabyte	PB	2^{50}	1,125,899,906,842,624	1024 TB
Exabyte	EB	2^{60}	1,152,921,504,606,846,976	1024 PB
Zettabyte	ZB	2^{70}	1,180,591,620,717,411,303,424	1024 EB
Yottabyte	YB	2^{80}	1,208,925,819,614,629,174,706,176	1024 ZB

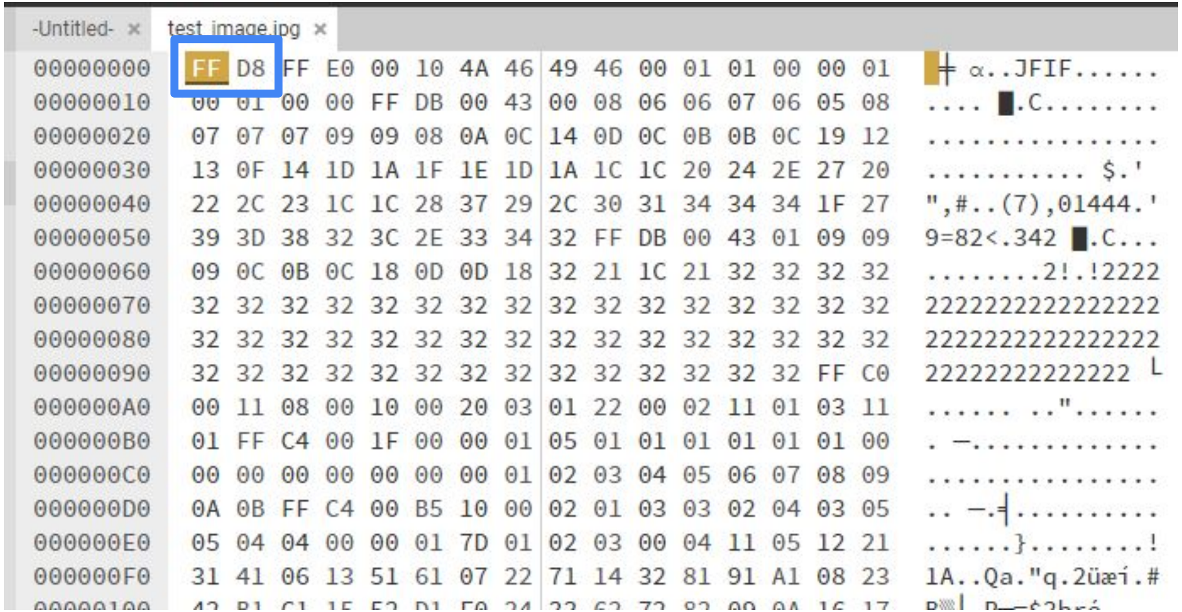
What can I do with files - in general?

- Create a new file
- Change the access permissions and attributes of a file
- Open a file, which makes the file contents available to the program
- Read data from a file
- Write data to a file
- Delete a file
- Close a file, terminating the association between it and the program
- Truncate a file, shortening it to a specified size within the file system without rewriting any content

Files' internal metadata

Magic Numbers:

Beginning of file tells you which file type it is!



File extensions are arbitrary

Extensions help to decipher the file content, but the file needs still to follow the file type’s organization.

For example:
Renaming image.**png** to image.**jpg** does not convert the file to the JPG standard.
It has still the SAME CONTENT (--> being a PNG file)

File types commonly used in Data Science

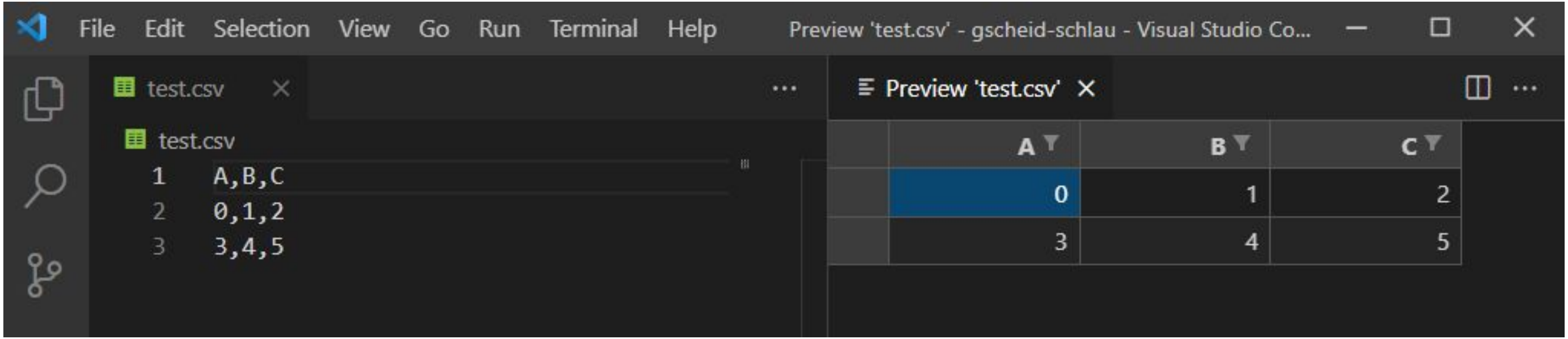
- Plain text (common extensions *.txt, *.csv, *.log, *.json, *.xml) - Python program code!
- Spreadsheets (*.xlsx)
- Word processing files (*.docx)
- Images (*.jpg -> Camera, *.png -> Scientific data, *.tif -> Microscopy)
- Videos (*.avi -> mostly raw data, *.mp4 almost everything, commonly h264 codec)
- Medical imaging data (DICOM, Nifti *.nii and *.nii.gz)
- Vector graphics (*.pdf, *.svg, *.ai)
- Container files (*.hdf5)
- Archives (*.zip, *.tar.gz, *.7z, *.rar)
- Database (*.sqlite)
- Deep Neural Networks (*.pb, *.h5, *.tflite, ...)

Software you should have around

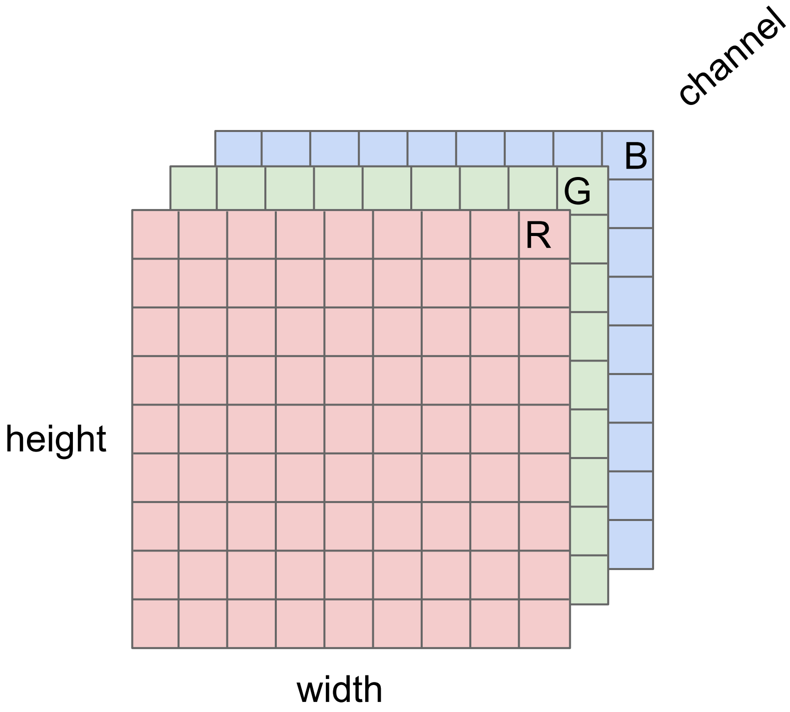
These are EXAMPLES that e.g. work for me. They can be replaced by various other tools. Everything is free except indicated.

- Visual Studio Code (plain text, CSV files, JSON, XML)
- LibreOffice/M$ Office/Google Docs (docx, xlsx, pptx,...)
- FIJI / ImageJ (Microscopy images) and paint.NET (all purpose images)
- VLC (Videos)
- Inkscape (free) or Adobe Illustrator ($$$) (vector graphics)
- 7zip (all kinds of archives)
- HDF5View (HDF5 container files)
- Netron (universal cross-platform deep neural network viewer)

Plain text file



An image consists of many pixels



<u>Very common</u>:

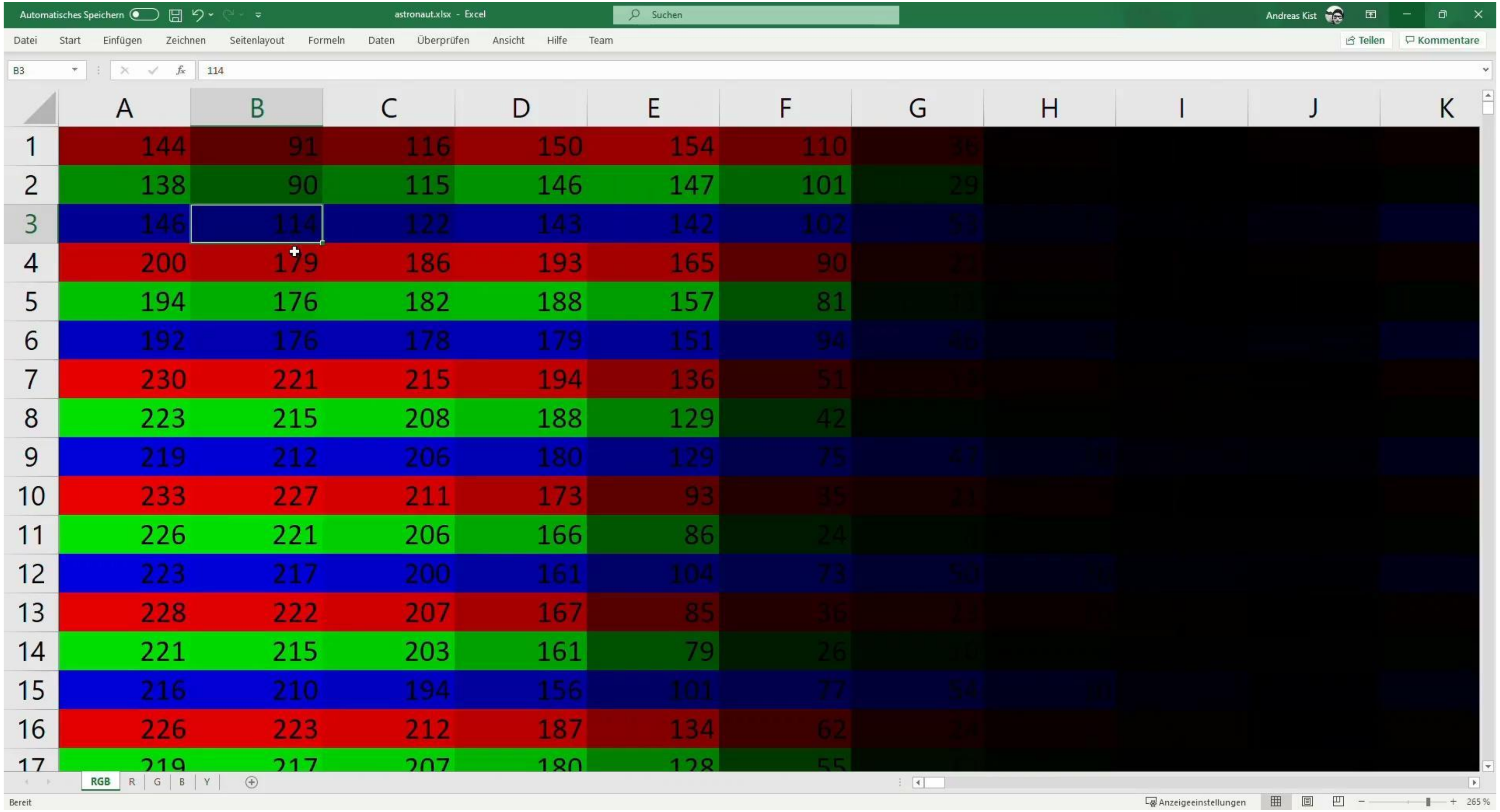
RGB (height x width x channels ⇒ HxWx3)
RGBA (HxWx4, last channel is alpha ⇔ transparency)
Monochrome (HxWx1 ⇒ HxW)

<u>Microscopy data:</u>

HxWxC,
where C is e.g. DAPI, GFP, Alexa488, mCherry, ….

E.g. an image of HxWxC = 256x256x3,
has 256x256x3 = 196,608 units, that we call **pixels**!

Images are just Excel sheets



	A	B	C	D	E	F
1	144	91	116	150	154	110
2	138	90	115	146	147	101
3	146	114	122	143	142	102
4	200	179	186	193	165	90
5	194	176	182	188	157	81
6	192	176	178	179	151	94
7	230	221	215	194	136	51
8	223	215	208	188	129	42
9	219	212	206	180	129	75
10	233	227	211	173	93	35
11	226	221	206	166	86	24
12	223	217	200	161	104	73
13	228	222	207	167	85	36
14	221	215	203	161	79	26
15	216	210	194	156	101	77
16	226	223	212	187	134	62
17	219	217	207	180	128	55

Interacting with images in Python

OPENING/SAVING

imageio - Python library for reading and writing image data

scikit-image
image processing in python

OpenCV

PROCESSING

NumPy

Multi-dimensional image processing (scipy.ndimage)¶

scikit-image
image processing in python

OpenCV

PLOTTING

matplotlib

seaborn: statistical data visualization

PyQtGraph
Scientific Graphics and GUI Library for Python

Images in a scientific environment

TIFF

- Saves raw data
- Multiple channels
- Multiple bit depth levels

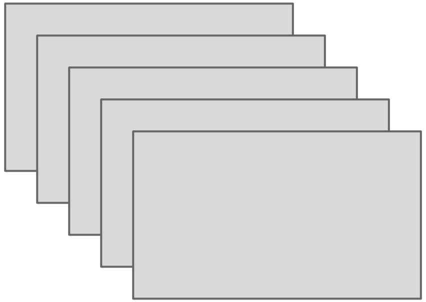
PNG

- Lossless compression
- Up to 4 channels (RGBA)

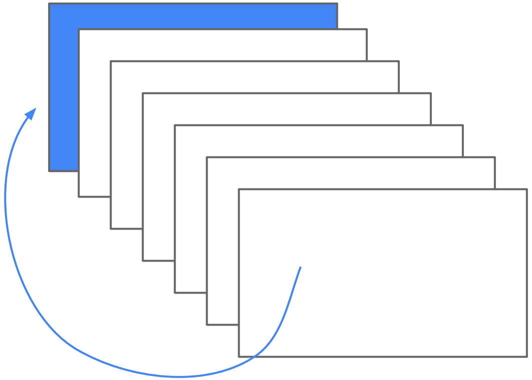
JPG

- Lossy compression
- Fine for photography
- Compression artifacts

Videos

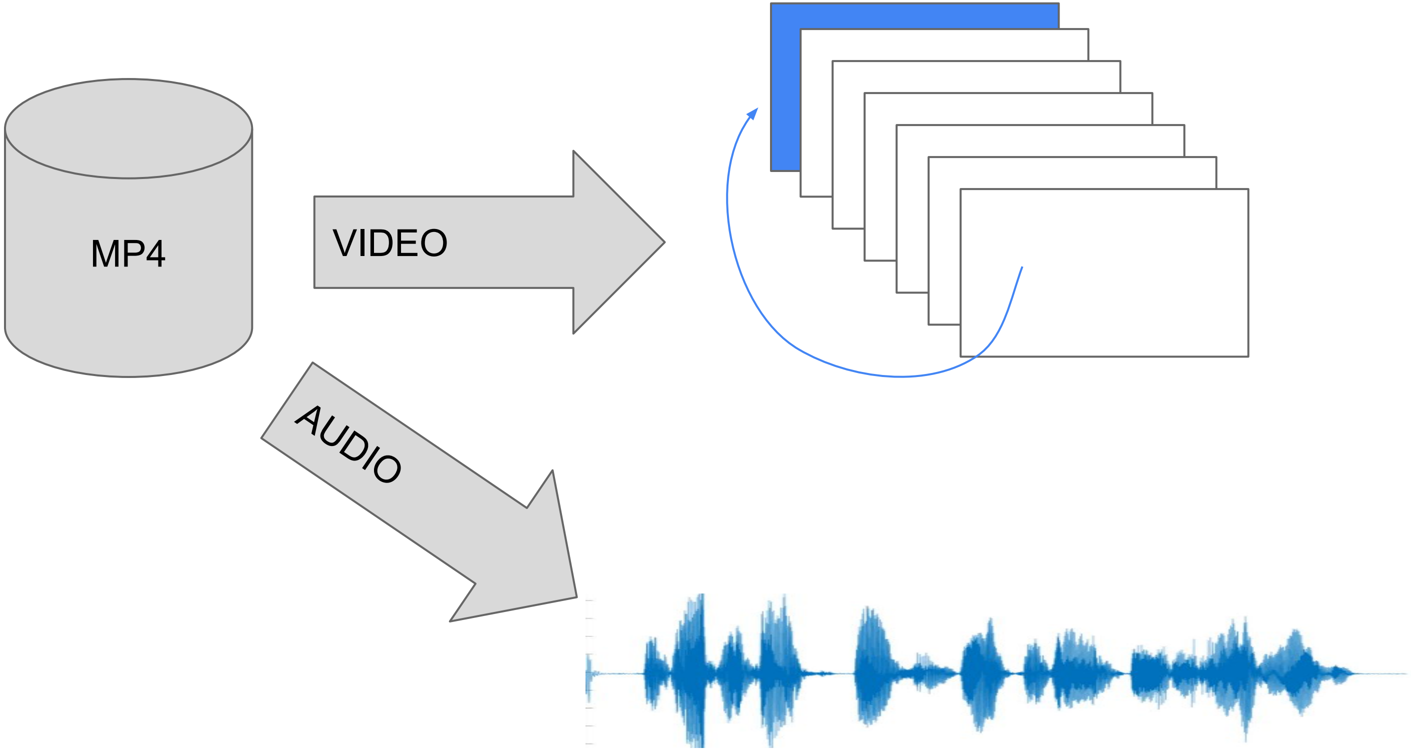


WAY 1: Store each frame one after another, each frame is independent



WAY 2: Store **key frames** and then store only the difference relative to the key frames

H264 codec in MP4 container



H264

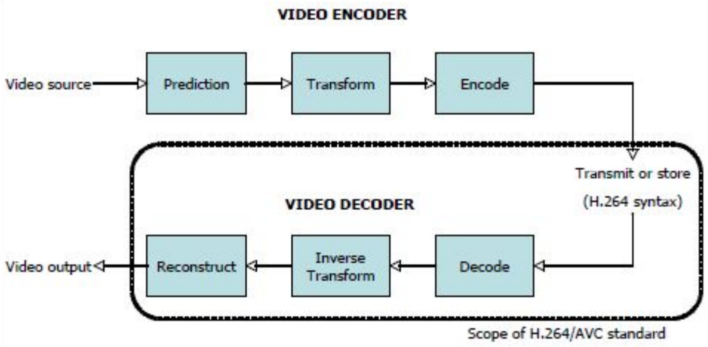


Figure 1 The H.264 video coding and decoding process

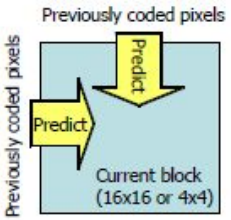


Figure 2 Intra prediction

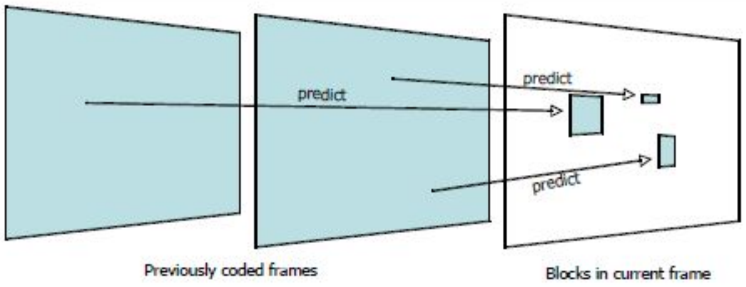


Figure 3 Inter prediction

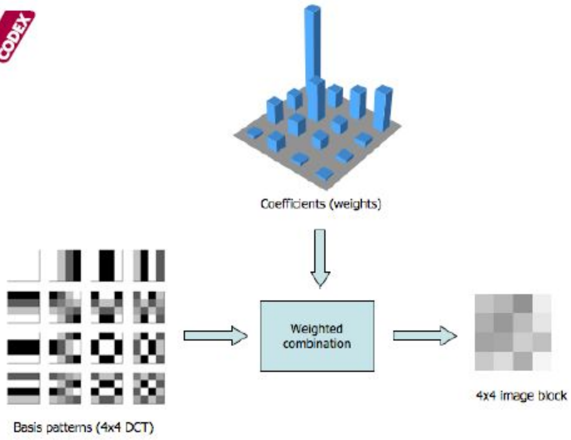


Figure 4 Inverse transform: combining weighted basis patterns to create a 4x4 image block

H264 performance

Figure 5 A video frame compressed at the same bitrate using MPEG-2 (left), MPEG-4 Visual (centre) and H.264 compression (right)

H264 is a great encoder, however, with the default settings you encode your data **lossy**!

LOSSLESS!!!

```
np.random.seed(42)
# Random video
ims_in = (np.random.randn(200, 256, 256, 3) ** 2).astype(np.uint8)

io.mimwrite("file.mp4",
            ims_in, # images
            codec='libx264rgb', # use the right codec
            pixelformat='rgb24', # and pixel format
            output_params=['-crf', '0', # Ensure setting crf to 0
                           '-preset', 'ultrafast']) # Maximum compression: veryslow,
                                                     # maximum speed: ultrafast

ims_out = io.mimread("file.mp4")
np.allclose(ims_in, ims_out)
# True
```

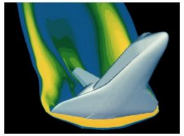
anki-xyz / lossless Public

Storing in mp4 is convenient for sharing and inspection using VLC

HDF5 - the universal file container

Scientific Fields


Astronomy


Computational Fluid Dynamics

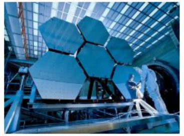

Earth Sciences


Engineering


Finance


Genomics


Medicine


Physics

How to handle/open/save HDF5?

```
pip install flammkuchen
```

```python
import flammkuchen as fl

d = {
    'foo': np.ones((10, 20)),
    'sub': {
        'bar': 'a string',
        'baz': 1.23,
    },
}
fl.save('test.h5', d)
```

Numpy ndarray
(e.g. multi-channel z-stack…)

E.g. some metadata...

Command line tool

Or, better yet, our custom tool `ddls` (or `python -m fl.ls`):

```
$ ddls test.h5
/foo                        array (10, 20) [float64]
/sub                        dict
/sub/bar                    'a string' (8) [unicode]
/sub/baz                    1.23 [float64]
```

Compression

Intelligent lossless compression,
A general feature of many libraries!

Check your data dtype!
You may save a lot of space!

Method	Compression	Space (MB)	Write time (s)	Read time (s)
scipy's mmwrite	N	145	79	40
numpy's save	N	134	1.36	0.75
pickle	N	115	0.63	0.17
deepdish (no compression)	N	115	0.52	0.17
numpy's savez_compressed	Y	32	8.88	1.33
pickle (gzip)	Y	29	5.19	0.86
deepdish (blosc)	Y	24	0.36	0.37
deepdish (zlib)	Y	21	9.01	0.83

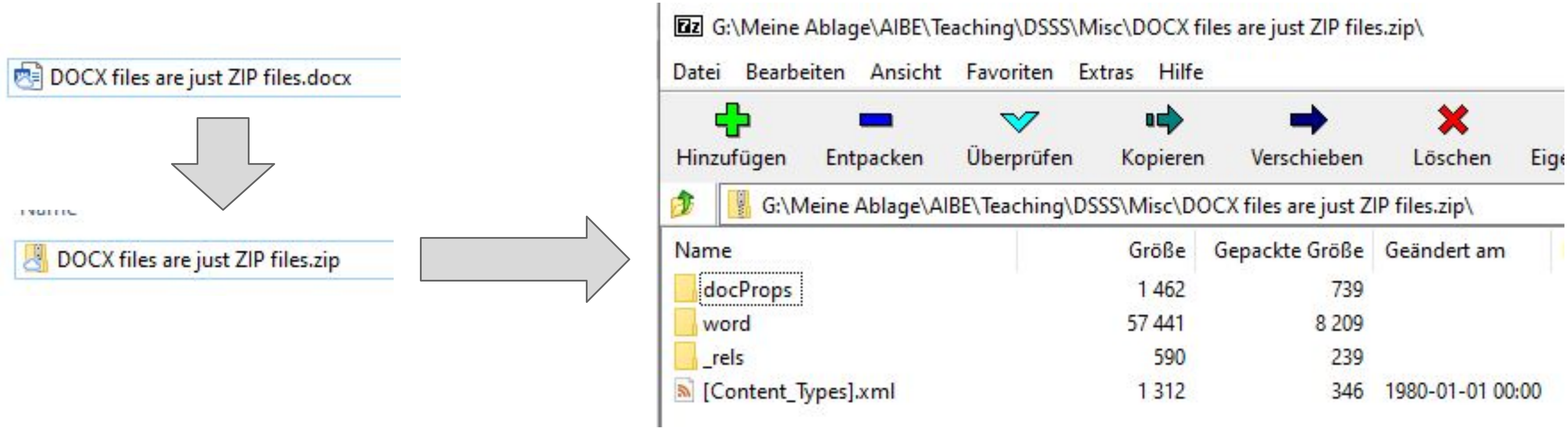
In [19]:
```
import flammkuchen as fl
import numpy as np
import os
```

In [20]:
```
x = np.random.randint(0, 2, (120, 512, 512, 3))  # int32!!
```

In [33]:
```
for i in range(10):
    %time fl.save("test_compression{}.h5".format(i), dict(x=x), compression=("blosc", i))
    print("compression level {}, file size: {:.2f} MB".format(i,
        os.path.getsize("test_compression{}.h5".format(i))/1048576))
```

```
Wall time: 283 ms
compression level 0, file size: 384.01 MB
Wall time: 496 ms
compression level 1, file size: 155.01 MB
Wall time: 704 ms
compression level 2, file size: 69.06 MB
Wall time: 855 ms
compression level 3, file size: 90.59 MB
Wall time: 825 ms
compression level 4, file size: 46.72 MB
Wall time: 805 ms
compression level 5, file size: 46.72 MB
Wall time: 789 ms
compression level 6, file size: 46.72 MB
Wall time: 782 ms
compression level 7, file size: 46.72 MB
Wall time: 763 ms
compression level 8, file size: 46.72 MB
Wall time: 772 ms
compression level 9, file size: 46.72 MB
```

Fun fact: DOCX files are just ZIP files...

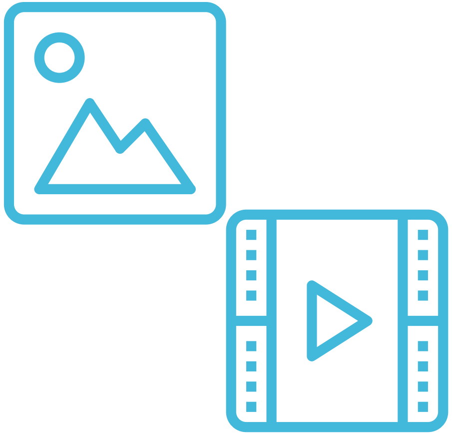


Exercise

Description of the exercise

This exercise will involve:

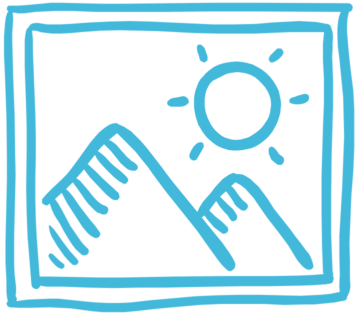
- Images: Their properties and some libraries to work with.
- Video: How they are represented.
- Tabular data.
- Arbitrary data.



Description of the exercise

Images and Video:

For images, you will work with color channels and the Peak signal-to-noise ratio (PSNR). For videos, on the other hand, your assignment is to see how NumPy arrays can represent them and their frames.



Description of the exercise

We present an introduction to JSON, CSV, and the tool for data analysis and manipulation: Pandas.

Here you will have to append data into a JSON file and edit a part of the Pandas data.

pandas