Verzeichnisstruktur aufbauen

- kleines skript fürs verzeichnis struktur bauen?
- seafile library anlegen
- verzeichnisstruktur anlegen in dateimanager o.ä

```
|-- code
    |-- docs
    |-- src
    |-- examples
    |-- output
    |-- main.py
    |-- README.rst
```

- verzeichnis code mit PyCharm öffnen
- python venv interpreter einrichten
- Docstring Format
  str alt s | Tools | Python Integrated Tools | Docstrings > Docstring Format: NumPy
- in `src/`
  - **init**
  - linalg.py
    - class csr_matrix
    - def axpy()
    - def scal()
    - def nrm()
    - def csrmv()
  - in **main.py**
    - import src
    - if **name**_
    - ....
    - edit config: run with console
  - linalg.py etwas ausbauen
    - `def tests()`...
  - in **main.py**
    - import src
    - run tests
  - **Terminal** Reiter
    - ggf. `source .venv/bin/active`
      - source runs code from file in current bash session
    - sonst: nochmal in den Einstellungen schauen
    - hier könnten wir mit pip pakete installieren
      - import numpy --> fails
      - siehe .venv/lib/site-packages
      - pip3 install numpy

        - siehe .venv/lib/site-packages
        - import numpy
    - requirements.txt
        - ggf. plugin installieren
        - dann import und requirements.txt updaten

Dokumentation mit `sphinx`

- Dokumentation zeigen im Browser
    - unten im Footer meist Info über verwendete Software
- Allgemeine Herangehenweise
    - wir wollen u.a..:
        - Automatisierung
        - Information nur an einer Stelle: Sync code und doc
        - ggf. mathematische Formeln (brauchen entsprechende Markup Language)
- Hier an einem Beispiel: sphinx (geschrieben in Python)
- Dokumentation lesen:
    - https://www.sphinx-doc.org/en/master/index.html
    - https://sphinxcontrib-napoleon.readthedocs.io/en/latest/index.html
- Installation
    - gui: strg+alt+s: project, interpreter --> install sphinx
    - terminal: pip3 install sphinx

Getting Started

- pycharm: terminal (.venv) $
- Intressehalber Version checken: `sphinx-build --version`

in `code/`

- `sphinx-quickstart <PATH-to-ROOTDIR>` (hier einfach `docs/`)
    - sep source and build: y
    - name:
    - release: 0.1
- Verzeichnis docs/ inspizieren
    - `source/`
        - hier liegen die Quelldateien um die Dokumentation zu steuern
        - conf.py
            - hier können wir die Dokumentation konfigurieren
            - u.a. wurden unsere Abfragen dort gespeichert
        - index.rst ist unsere Startseite (unsere Hauptdatei/"Kleber")
            - rst wie reStructuredText (ähnlich wie markdown)
            - Direktiven und deren Optionen
            - Überschriften `===, ----`
            - setze zB `.. note::`
    - `build/`

        - dieses Verzeichnis ist noch leer
        - das ändern wir nun
- **build**
    - `sphinx-build -b html docs/source/ docs/build/html`
    - in `build/` Verzeichnis schauen
    - wir finden nun einen Ordner `html`
    - wir öffnen die Datei `index.html` mit einer geeigneten Software (=Browser)
        - zB mal nach der Notiz suchen
    - `index.rst` [in reStructuredText] becomes index.html
        - The file index.rst created by sphinx-quickstart is the root document, whose main function
          is to serve as a welcome page and to contain the root of the “table of contents tree” (or toctree).
          alternativer build command ist nun: `make html` in docs/
- Bemerkung zu den `sphinx-` binaries:
    - Diese befinden sich in unserer venv `/code/.<VENV-NAME>/bin`
    - siehe auch: `$ locate sphinx-*` (ggf. `sudo updatedb` notwendig)
      (alternativ: `type sphinx-*`)

Maßschneiderung/Costumization `conf.py`

`extensions = []`

- Erweiterungen stellen zusätzliche Funktionalitäten bereit
- hier können wir leicht Schalter setzen, um das Erscheinungsbild zu steuern

Allgemeines Erscheinungsbild: `html_theme`

- **builtin schemes**
    - https://www.sphinx-doc.org/en/master/usage/theming.html#builtin-themes
    - Zb `classic`
    - in conf.py: `html_theme = 'classic'`
    - ... `docs/ $ make html`
- **third-party schemes**
    - https://sphinx-themes.org/
    - zB `sphinx-rtd-theme (read the docs)`
    - muss installiert werden (da nicht in standard sphinx installation enthalten)
        - `pip install sphinx-rtd-theme` oder über GUI
    - in conf.py: `html_theme = 'sphinx_rtd_theme'`

Struktur ausbauen

- Neue Unterseiten anlegen, zB
  `docs/source/usage.rst`

  mit Inhalt:

```
Überschrift
===========

Etwas Text, blabla

.. code-block:: console

   (.venv) $ pip install sphinx
```

- build
- Warnung: `checking consistency... /.../usage.rst: WARNING: document isn't included in any toctree`
- Diese Seite möchte noch verlinkt werden damit wir darauf zugreifen können
- in unserer Hauptdatei zum Inhaltsbaum hinzufügen:

```
.. toctree::

   usage
```

Synchronisation von Code und Dokumentation: `autodoc`

- nun wollen wir in der Dokumentation den Code beschreiben und dabei auf vorhandene docstrings zurückgreifen
- das ist eine zusätzliche Funktionalität, die durch eine Erweiterung bereitgestellt wird:
  - genauer `autodoc`: https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html
- aktivieren in docs/source/conf.py
  - zur Liste hinzufügen: `extensions = ['sphinx.ext.autodoc', ]`
  - sonst Fehler wie `ERROR: Unknown directive type "autofunction".`
- nun können wir zum Beispiel folgene Direktiven verwenden

```
.. autofunction:: src.linalg.axpy

.. autoclass:: src.linalg.csr_matrix
```

- Wir bekommen weiterhin Fehler, da die relativen Pfade src.linalg nicht erkannt werden, daher den absoluten Pfad zu `<PATH>/code/` dem .venv Interpreter mitgeben:
  - **Pfade zum code hinzufügen in conf.py**

```
import pathlib
import sys
sys.path.insert(0, pathlib.Path(__file__).parents[2].resolve().as_posix())
```

Der String `pathlib.Path(__file__).parents[2].resolve().as_posix()` sollte genau den Pfad zu `/.../code/` enthalten

- Wir bekommen weiterhin einen Fehler, da docstring kein reStructuredText sind sondern NumPy Format

docstring Style: reStructuredText, NumPy, Google

- für NumPy Format: `sphinxcontrib-napoleon`
- Installation: gui oder cli

```
(.venv) $ pip install sphinxcontrib-napoleon
```

- conf.py: extension hinzufügen
- read doc of Napolean
- TypeSetting
- Examples in docstring, code via `>>>`

Indices
Nun auch nochmal schauen:

- global
- module
- Search Page

Reference Guide: Automatisch alle Code-Bestandteile auflisten mit `.. autosummary::`

- API references/Reference Guide VS User Guide
- wir legen dazu eine neue Unterseite an: `api.rst`
- toctree in `index.rst` ergänzen
- Die Direktive `.. autosummary::` wird durch die Extension `sphinx.ext.autosummary` freigeschaltet
- in `docs/source/api.rst`

```
.. autosummary::
    :toctree: generated

    src.linalg
```

Links zwischen Quellcode und docs: `[source]`, `[docs]`

- extension aktivieren: `sphinx.ext.viewcode`

Cross--References: Links innerhalb der Dokumentation

- label/tag setzen vor Überschrift
  `.. _LABEL-TAG:`
- Referenz auf den ge-label-ten Abschnitt
  `:ref:`LINK-NAME <LABEL-TAG>`

Mathematische Formeln `.. math::`

Andere Build-Formate

- in code/docs/
  `make latexpdf`
- oder in code/
  `sphinx-build -b latex docs/source/ docs/build/latex/`