Notizen aus dem PyCharm-Tutorial

Übergeordnetet Links

Die meisten der verlinkten Seiten liegen auf:

- http://www.academis.eu
- https://python-basics-tutorial.readthedocs.io/de/latest/
- https://www.python4data.science/de/latest/

Ein neues Projekt in PyCharm anlegen

- Projekt meist nützlicher als standalone Skript
- erkennbar am `.idea`-Ordner
- `.idea` kann ins `.gitignore` muß aber nicht

Virtuelle Umgebungen

- Empfehlung: eine Environment pro Projekt, außer es ist eine "Spielwiese"
- virtual envs mit conda
- eventuell muß der Pfad zu `conda.exe` (`Scripts\`) zur Umgebungsvariable `PATH` hinzugefügt werden

Pakete installieren

- Installieren von mehreren Paketen aus dem Terminal: `pip -r requirements.txt`
- aus PyCharm mit `Tools-> Sync Requirements`
- Pakete werden nach `lib/site-packages` installiert
- pip installiert Pakete von pypi.org
- statt `requirements.txt` wird neuerdings auch oft `pyproject.toml` verwendet
- siehe auch Dependencies in PyCharm verwalten

Terminal aktivieren

In der Powershell lässt sich conda aktivieren mit:

```
set-executionpolicy unrestricted
conda init powershell
```

siehe auch conda in Powershell aktivieren

Importpfade

- in PyCharm beeinflusst `Mark as -> Source root` den import-Pfad
- auch die Umgebungsvariable `PYTHONPATH` enthält Verzeichnisse zum Importieren

- Um Verwirrung beim Import zu vermeiden sollten alle Pakete und Module unterschiedliche Namen haben

Dateipfade

Beim Importieren aus mehreren Verzeichnisen musst Du für die meisten Dateien **absolute Pfade** verwenden. Folgender Zweizeiler hilft dabei:

```
# ermittelt das Verzeichnis der aktuellen Datei
BASE_PATH = os.path.split(__file__)[0]
MY_CSV = os.path.join(BASE_PATH, "hello.csv")
```

Versionskontrolle

- Code verwalten mit Git
- Git Introduction
- Pro Git - Buch von Scott Chacon

Debugging

- Kristians Debugging-Tutorial - Video
- Kristians Debugging-Tutorial - Code
- Arten von Bugs und Debugging Techniken

Automatische Tests

- Kristians pytest Tutorial
- pytest im Python Basics Tutorial
- pytest Doku
- Continuous Integration

Tools zur Codepflege

- black
- flake8
- mypy
- pydantic
- Logging

Refactoring

- Kristians Refactoring-Tutorial - Video
- Kristians Refactoring-Tutorial - Code
- Liste von Refactorings auf sourcemaking.com

Python Kommandozeilenparameter

- Kommandozeilenaufrufe folgen dem Muster **PYTHON INTERPRETER_OPTS SCRIPT SCRIPT_PARAMETERS**
- jeder Teil lässt sich in den Build Configurations angeben
- Beispiel: `python -m cProfile hello.py Ada`
- Beispiel: `python -m pytest test_twenty_questions.py`

Performance

- Standardaufruf cProfile: `python -m cProfile -s cumtime hello.py`
- im IPython Terminal auch `%time` und `%timeit`
- snakeviz
- weitere Performance Werkzeuge

Nützliche Python-Bibliotheken:

- Malen mit Numpy

Lizenz

(die meisten der verlinkten Seiten übrigens auch)

Viel Spaß!