Programmierung mit Python

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2026-06 Dozent: Karsten Flügge

Offene Themen / Spezialsyntax

try
lambda (signatur)
Iteratoren + yield
immutable types
list/dict comprehension
@dekaratoren
@dataclass
@classmethod

… Vertiefungen

… spezielle module (interne, externe) / Anwendungen

z.B. random, numpy (numerische Berechnungen, Matrizen und Vectoren),

RECAP: Variablen Parameter überschreiben / verändern

def f(x):
  # wie und wann können wir x verändern?
  x = NIE überschreiben hat **keinen Effekt** auf x, nur lokal

```
x=7
f(x)
x==7
```

Funktionen können **un**veränderliche Objekte **nicht verändern**. (Zahlen, **Tupel und Strings**)
Funktionen können **veränderliche** Objekte **verändern** (z.B. Listen, Maps, ... alles andere)

```
def change(liste):  # propagate effect "verändernde Funktion"
  liste.pop()
  liste[1]=4      # HAT EFFEKT: Wert wird verändert
  liste=[0,0,0] # KEINEN EFFEKT: Variable wird umgesetzt, alter Wert/Objekt bleibt irgendwo bestehen

l = [1,2,3]
change(l)
print(l) # 1,2
```

Man kann mit dem Objekt **interagieren**, aber man kann den Parameter (Funktions**variable**) **nicht überschreiben**.

RECAP: externe Variablen überschreiben (global) / verändern

Böses Keyword
global externe Variablen in Funktionen setzen!

```
x = 7  # externe Variable
def f():
  print(x) # LESEN GEHT IMMER
  x=8 # normalerweise kein Effekt

def f():
  global x  # keyword!
  x=8 # kann jetzt überschrieben werden!
f()
print(x) # 8

ACHTUNG:
liste=[1,2,3]
def f():
  liste[2]=3 # FOOTGUN: darf man leider, schlechter Stil
```

Man kann mit dem Objekt **interagieren**, aber man kann externe Variable nur **mit 'global' keyword überschreiben**.

Python for scripting / modding

Games and applications using python as **scripting language:**

Blender - Allows Python scripting for creating mods, automating tasks, and extending the software's capabilities.
GIMP - Uses Python for developing plugins and extensions that enhance or modify the image editing capabilities of the software.
Autodesk Maya - Supports Python for writing plugins and scripts that modify and extend functionality, often used in professional animation and modeling environments.
Microsoft XLS - soon

Civilization IV - Allows extensive modding with Python, enabling modders to tweak game mechanics, units, and AI behavior.
World of Tanks - Uses Python for modding, particularly for customizing user interfaces and gameplay elements.
The Sims 4 - Supports Python for creating mods that alter gameplay, though this is done through the game's API rather than direct scripting support.
Eve Online - Python is used to develop mods and scripts that enhance the game's interface and functionalities.

Github Versionierung

freiwillig github.com gitlab.com oder eigenes!

Vorteile von Versions-Verwaltung GIT:

- **Backup**
- **Intelligentes rückgängig machen wenn etwas schief läuft**
- **Cooperation mit anderen:**
- **Zusammenfügen (merge) von verschiedenen Lösungen / Aspekten / Ständen / …**
- **Vieles mehr**

Nachteile:
bei öffentlichen Repos KEINE SCHLÜSSEL / private DAten etc hochladen!!

clone : Repository (Projekt) Laden
init: eigenes Repository
add : Datei hinzufügen
commit : Stand speichern
push : Stand hochladen
pull : Laden

ggf git installieren: https://git-scm.com/downloads

Github Versionierung

Anlegen eines neuen Repositories,

https://github.com/new …

dann URL merken + kopieren https://github.com/pannous/alfa_python.git

und mit lokalem Ordner verbinden:

```
git remote add origin https://github.com/pannous/alfa_python.git
git branch -M main
git push -u origin main
```

git clone git@github.com:pannous/alfa_python.git

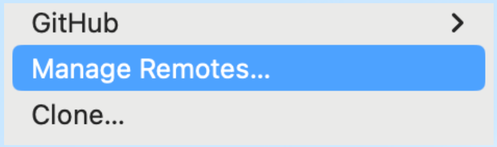
Github Versionierung

Git installieren: https://git-scm.com/downloads

Existierendes Repository mit lokalem Ordner verbinden:

```
git clone https://github.com/pannous/alfa_python.git # ohne Account
git clone git@github.com:pannous/alfa_python.git  # wer Account hat
```

oder in IDE

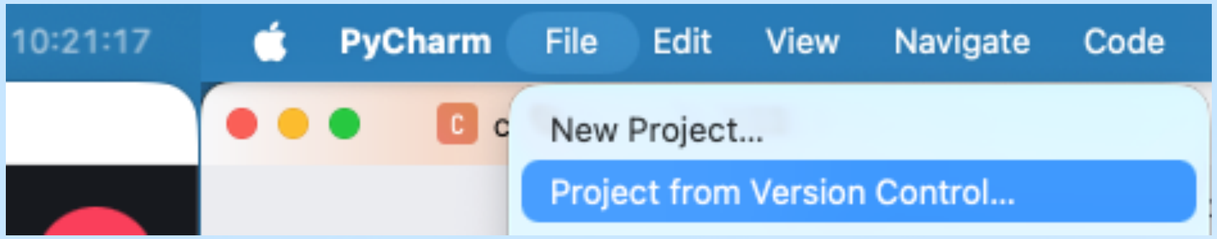


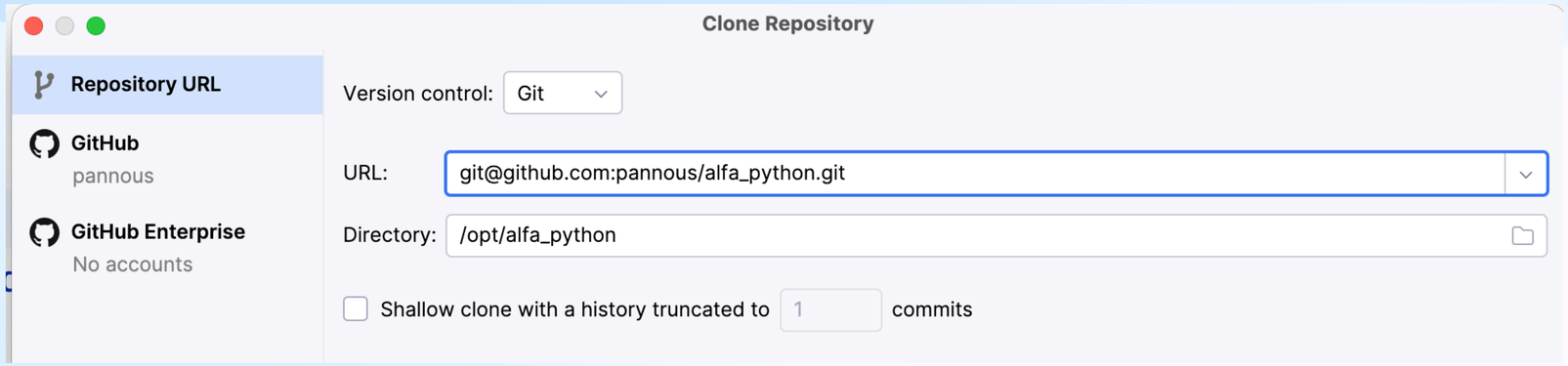
<<< Source Control

Github Versionierung

Existierendes Repository mit lokalem Ordner verbinden:

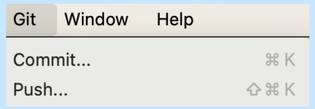
```
git clone https://github.com/pannous/alfa_python.git # ohne Account
git clone git@github.com:pannous/alfa_python.git  # wer Account hat
```

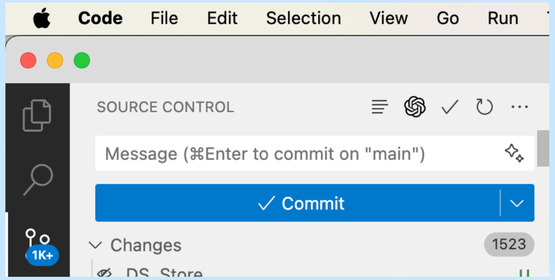


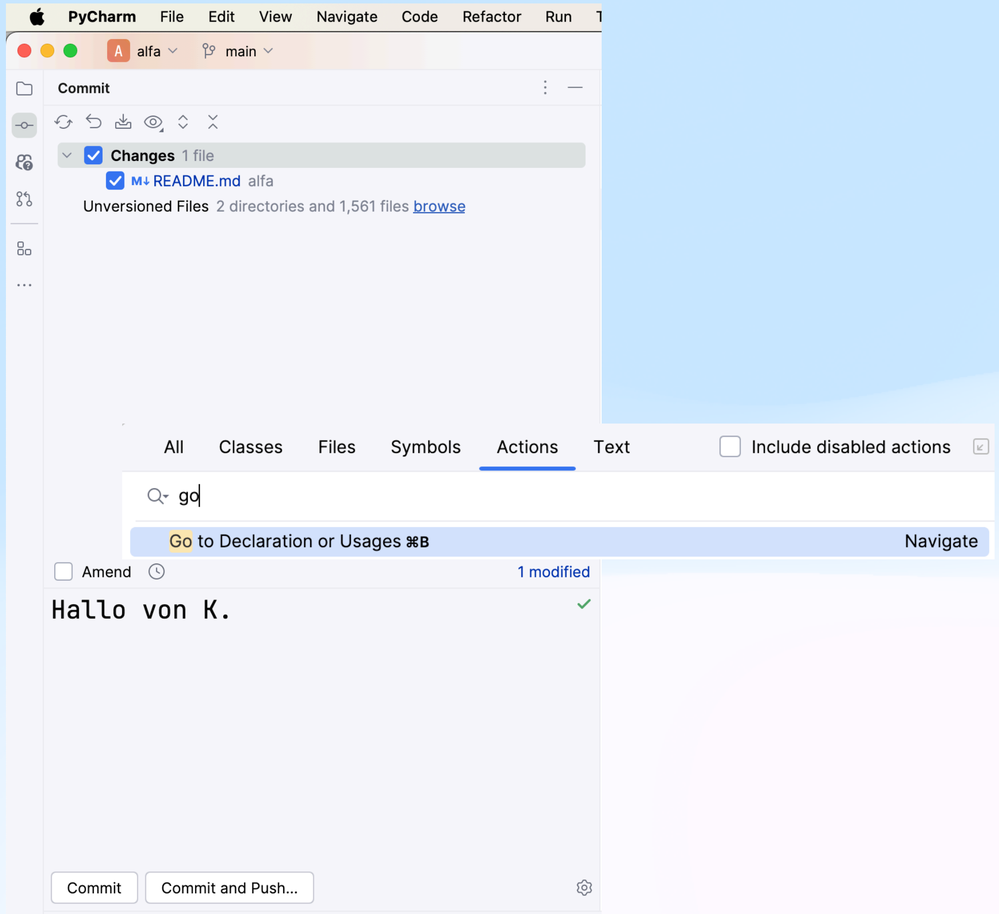


Github Änderungen 'committen'

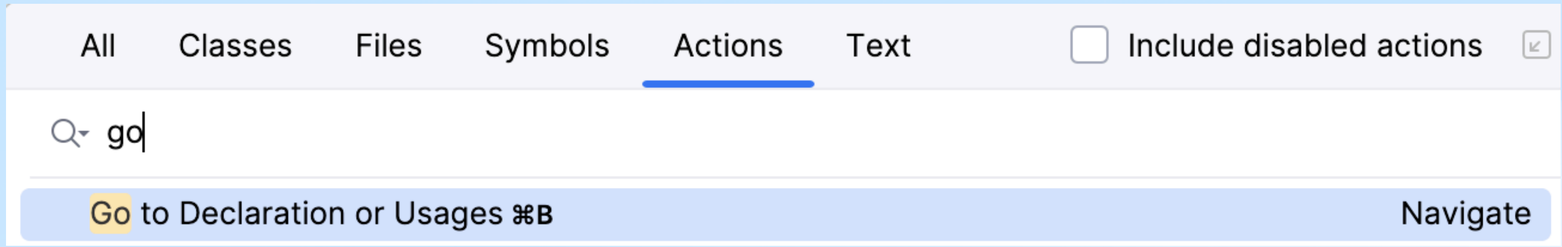
ACHTUNG: VOR PUSH muss man PULL machen





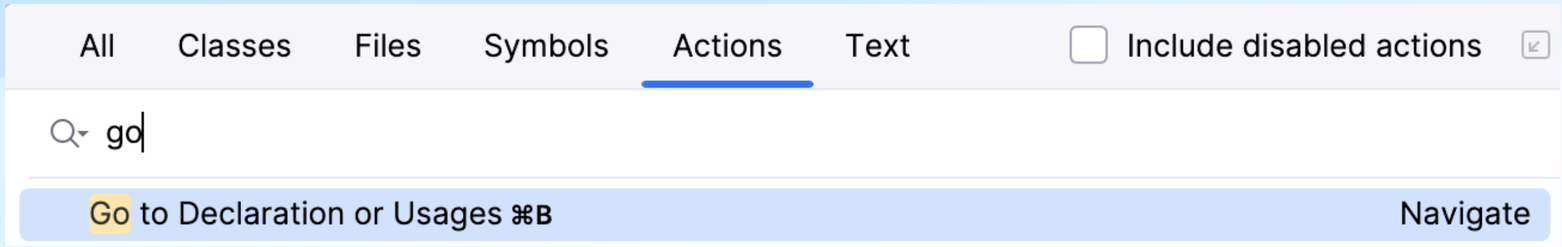


Module inspizieren



Module inspizieren

```
help(modul_name)

mouse over
```



Module inspizieren

Standard module
numpy
pandas für Datenanalyse
pygame für Spiele oder Grafiken/Interaktionen
matplotlib zum plotten
pytorch wenn ihr AI macht

- **sqlite3** — eingebaute SQL-Datenbank

mysql / postgres
jede Software hat ihre bibliothek

blender, ... **adaptoren** **Kommunikation mit Hauptprogramm**

Module inspizieren

Eingebaute Bibliotheken/Module in Python:

- **`math`** — mathematische Funktionen (`sin`, `sqrt`, `log`, …)
- **`random`** — Zufallszahlen, Mischen, Ziehen
- **`datetime`** — Datum/Zeit
- **`time`** — Zeitmessung, `sleep`
- **`os`** — Betriebssystemzugriffe, Dateien, Pfade
- `pathlib` — moderne Pfadobjekte statt Stringpfade
- **`sys`** — Python-Systemdetails, Script Argumente (py example.py -- 1 2 3)
- `json` — JSON lesen/schreiben
- **`re`** — reguläre Ausdrücke
- `collections` — Spezialcontainer (`Counter`, `defaultdict`, `deque`)
- `itertools` — kombinatorische Iteratoren
- `functools` — **`reduce`**, Caching, höhere Funktionen
- **`threading`** / `multiprocessing` — Parallelisierung
- `asyncio` — asynchrone Programme

Module inspizieren

Wissenschaft / Numerik:

- SciPy — Optimierung, Statistik, Signalverarbeitung
- SymPy — symbolische Mathematik

Machine Learning / AI:

- scikit-learn — klassische ML-Verfahren
- torch — LLMs, Sprachmodelle

Web / APIs:

- Flask — kleine Webserver
- FastAPI — moderne APIs
- Requests — Webseiten/APIs abrufen

Visualisierung:

- Plotly — interaktive Diagramme
- OpenCV — Bildverarbeitung

Spiele:

- pygame

Module inspizieren

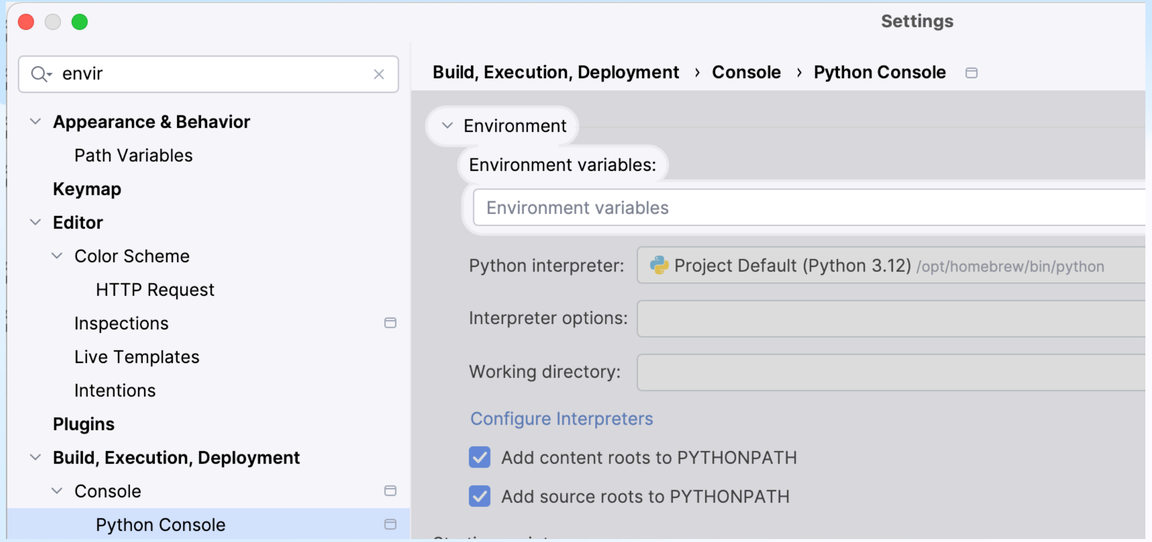
ARCHITEKTUR!

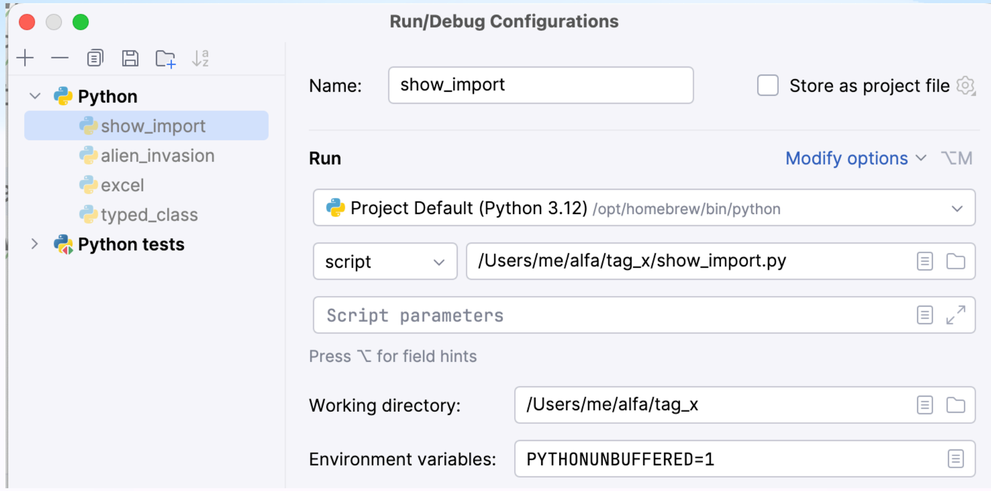
Struktur. Modularität.

Aufbau der Komponenten.

Umgebungsvariablen Setzen

```python
print("test")
print(1/0)

# Falls roter text NICHT hinter schwarzem Text erscheint:
# PYTHONUNBUFFERED=1 Umgebungsvariable setzen, damit der print Output nicht gepuffert wird sondern direkt ausgegeben
# setx PYTHONUNBUFFERED 1    # Windows in System Shell cmd.exe
# export PYTHONUNBUFFERED=1  # Linux MacOS in bash / Terminal.app

# andere sinnvolle Umgebungsvariable:
# PYTHONSTARTUP  # gebe python Datei mit die immer mit ausgelesen wird, zB für eigene extra Funktionen

setzen in IDE global oder per script (oder per Config Template irgendwo)
```

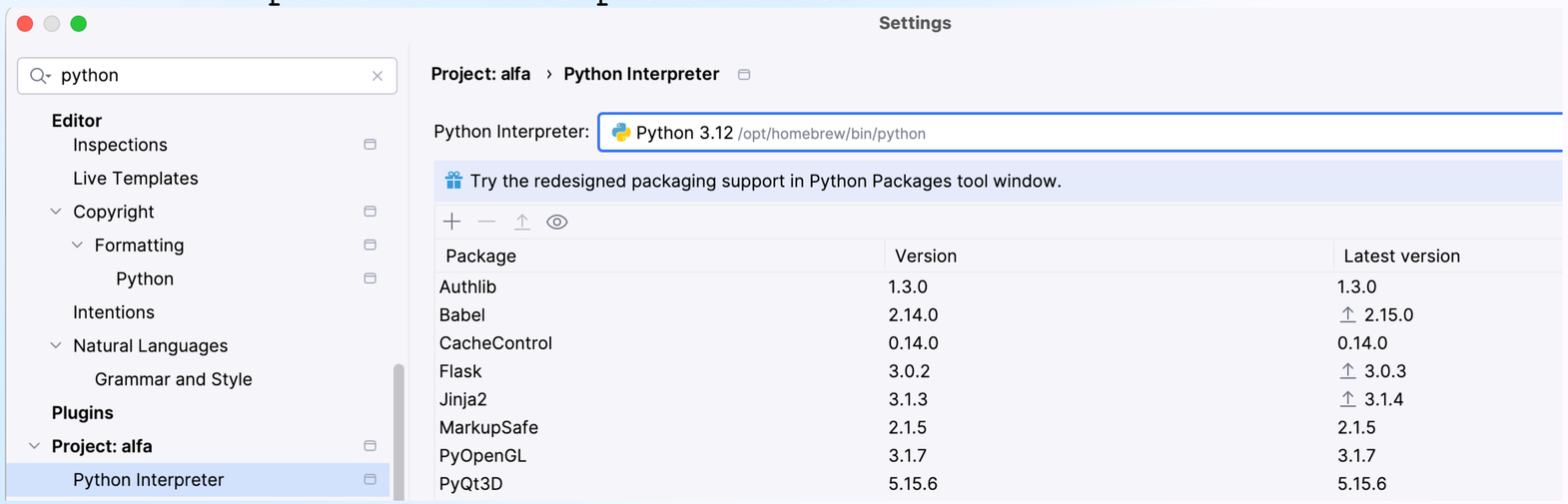




Pip Modul installieren

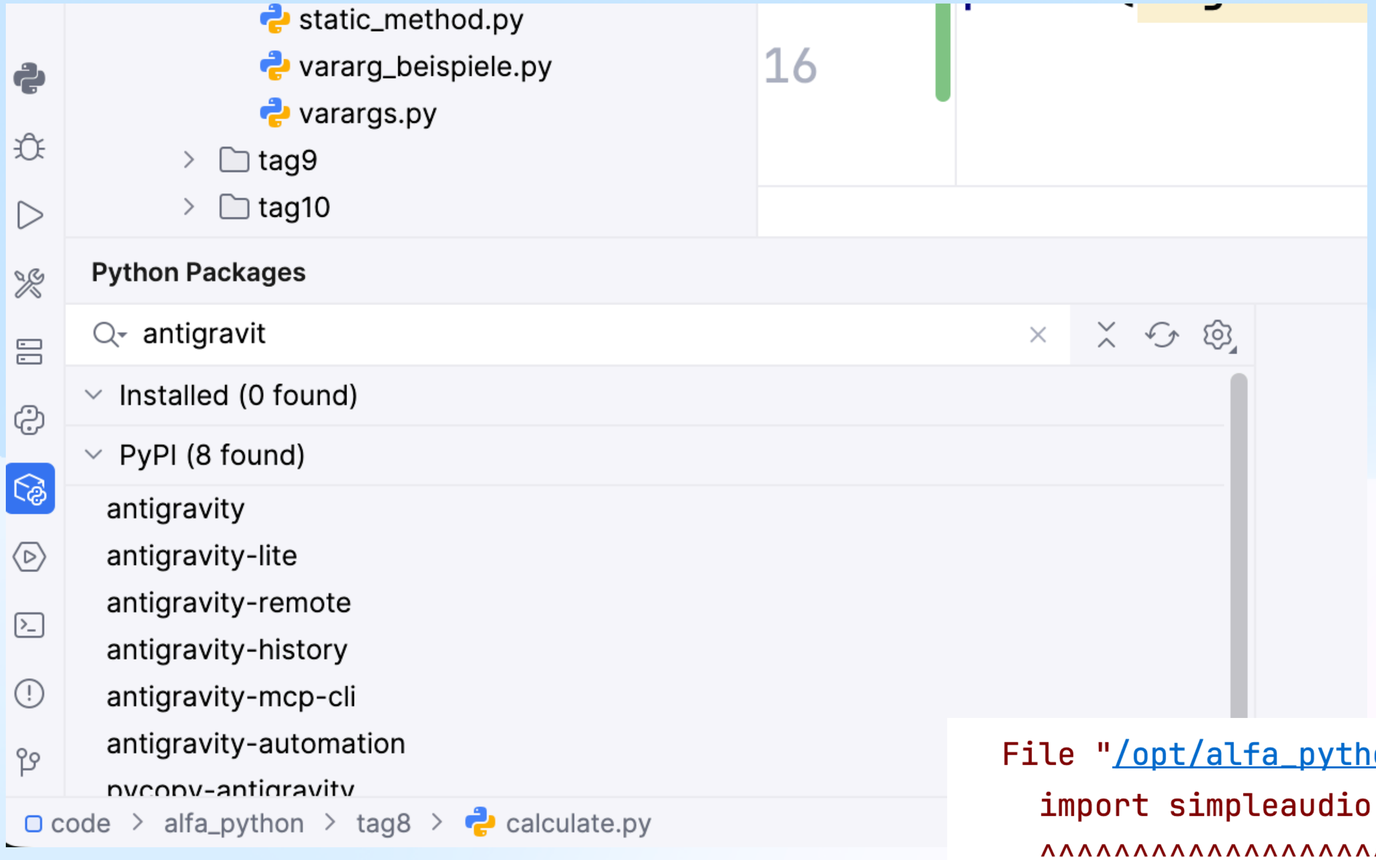
```
import antigravity
# ACHTUNG, MODULE KÖNNEN NEBENWIRKUNGEN HABEN, z.B. antigravity öffnet einen Browser mit einem Comic

# Um allgemein ein Modul zu installieren muss man entweder in IDE oder in der Konsole folgendes eingeben:
# python3 -m pip install <modulname>
# python3 -m pip install antigravity
# python -m pip install antigravity
# pip install antigravity

# pip install <modulname>   # pip ist ein Paketmanager für Python, direkt
# Analog conda bei Anaconda

# TIPP, wenn man ein Modul installiert hat und es trotzdem nicht importieren kann,
# dann sind vielleicht verschiedene Python Versionen installiert und das Modul wurde nur für eine Version installiert.
# Gegebenenfalls Python Version anzeigen: python --version
# Gegebenenfalls extra Python Version deinstallieren (IDE bringt eigenen Python Interpreter mit)
```

Oder in IDE passenden interpreter ansehen:



Pip Modul installieren

```
import antigravity
# ACHTUNG, MODULE KÖNNEN NEBENWIRKUNGEN HABEN, z.B. antigravity öffnet einen Browser mit einem Comic
```




```
  File "/opt/alfa_python/tag9/pi_sound.py", line 1, in <module>
    import simpleaudio as sa  # type: ignore[import-untyped]
    ^^^^^^^^^^^^^^^^^^^^^^^^
ModuleNotFoundError: Install Package No module named 'simpleaudio'
```

Oder klick auf "Install Package" in pycharm: