Programmierung mit Python

Grundlagen Grundbegriffe Grundübungen

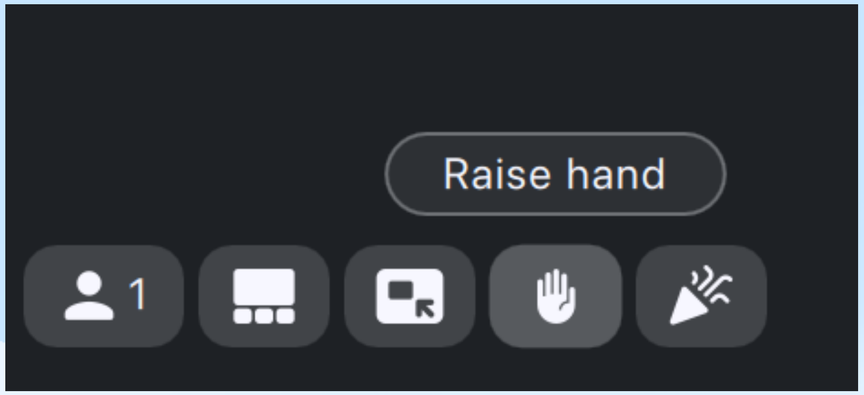
Alfatraining Kurs 2026-09 Dozent: Karsten Flügge

Willkommen!

Das wichtigste Feature im alfaview Programm: "Raise hand"

Unten Links, 4ter Button "Raise hand"

Zuerst: <u>Hand heben, wenn man mich hören kann!</u>



**Falls es irgendwelche Probleme gibt,
welche wir hier nicht lösen können,
dann zu <u>"Mein Support - Team"</u> im Browser auf
https://cloud.alfanetz.de**

Vorstellung

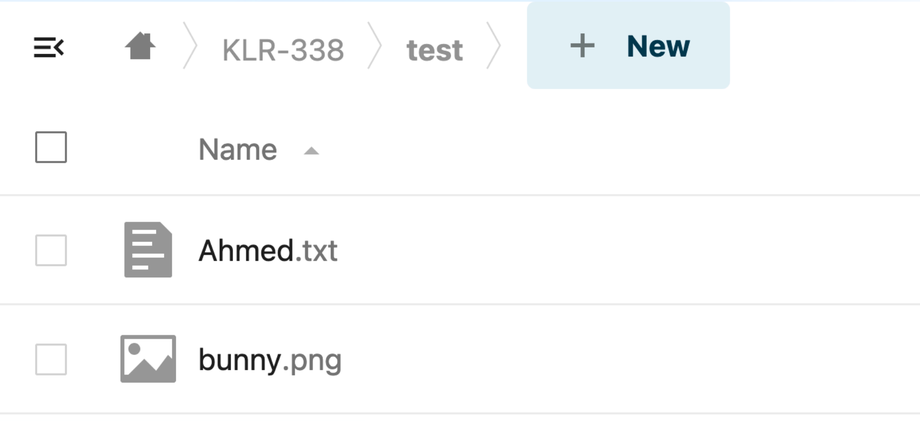
...

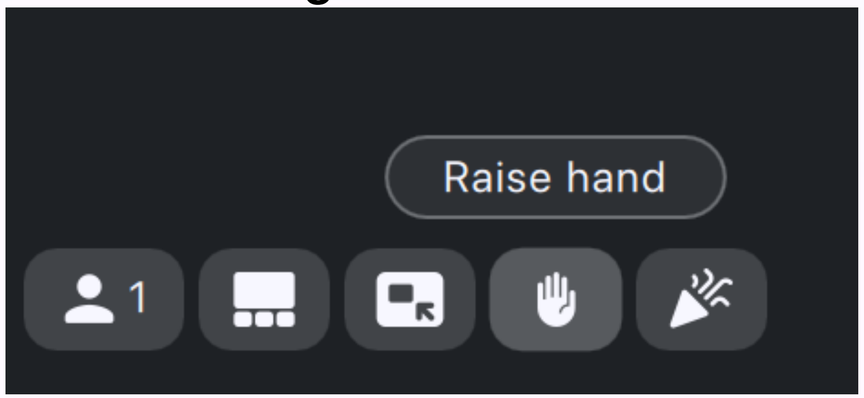
Aktuelles Ziel:

unter https://cloud.alfanetz.de/ im Browser einloggen,
dann oben Datenaustausch =>

Ordner suchen:

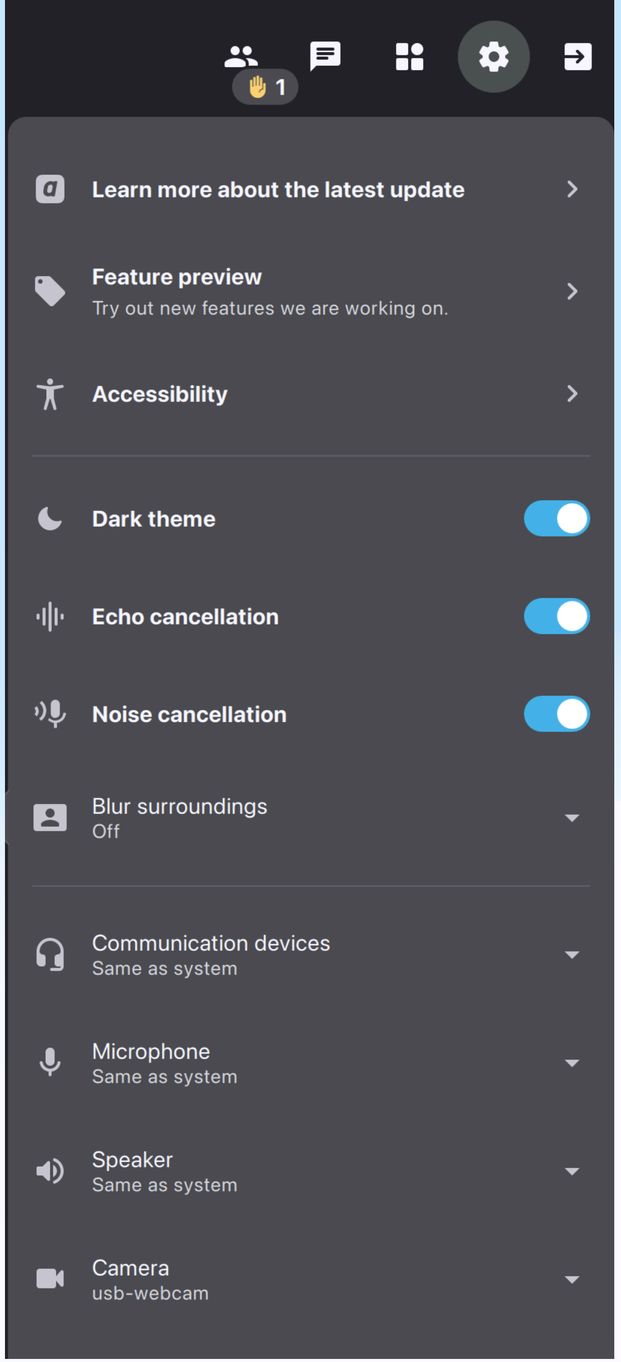
und dort eine Test Datei hochladen … und digitale Hand heben wenn fertig.





Pause

x


1
Learn more about the latest update
Feature preview
Try out new features we are working on.
Accessibility
Dark theme
Echo cancellation
Noise cancellation
Blur surroundings
Off
Communication devices
Same as system
Microphone
Same as system
Speaker
Same as system
Camera
usb-webcam

Pause

x

Geschichte

Python ist eine sehr **beliebt**e Programmiersprache, die Ende der 1980er Jahre von Guido van Rossum entwickelt wurde. Die Sprache wurde mit dem Ziel entworfen, den Code **lesbar und verständlich** zu gestalten.

Entwicklung von Python

- Python wurde erstmals 1991 veröffentlicht, begann Ende als ein Hobbyprojekt während der Weihnachtsferien.
- Die Sprache wurde nach der britischen Komödiengruppe „**Monty Python**“ benannt, was die verspielte Natur der Sprache widerspiegeln soll.
- **Guido van Rossum**, niederländischer „Benevolent Dictator For Life“
- Heute **professionelle Weiterentwicklung** im Python Steering Council
- Es gab einmal Wachstumsschmerzen beim Wechsel von python 2 auf **python 3**, aber das ist Historie
- In vielen **Industrie-Rankings** (wieder) auf den vordersten Plätzen (dank AI)

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	^	Python	15.33%	+4.47%
2	1	v	C	14.08%	-2.26%
3	2	v	Java	12.13%	+0.84%
4	4		C++	8.01%	+1.13%
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%

Philosophie

- **Die Philosophie von Python**
  - Van Rossum legte großen Wert auf die Lesbarkeit des Codes,
  - was dazu führt, dass Python eine **klare und präzise Syntax hat**.
  - Die Sprache fördert die Verwendung von weniger Codezeilen im Vergleich zu anderen Programmiersprachen, was den Code übersichtlicher und wartbarer macht.

Kleines Beispiel Programm:

```
print('Hallo, wie heißt du?') # jetzt den Namen einlesen:
meinName = input()
print('Schön dich zu treffen, ' + meinName)
```

```
Dieses kleine Programm demonstriert wie Python ohne großes Brimborium auskommt
```

Philosophie

EXTREM dynamisch

manchmal zu dynamisch? Es gibt aber Auswege.

math.pi = 4 ?? restart code?

Philosophie

Die **Philosophie** von Python ist in einem bekannten Dokument, dem **"Zen of Python"**, festgehalten. Diese Leitprinzipien wurden von Tim Peters verfasst und definieren die Designphilosophie hinter Python, die darauf abzielt, dass die Programmierung intuitiv und die Sprache leicht zu verstehen und zu verwenden ist. Die Philosophie kann durch den Befehl `import this` in einer Python-Umgebung angezeigt werden. Hier sind einige der wichtigsten Punkte aus dem "Zen of Python":

- **Schön ist besser als hässlich.**
  - Python betont die Wichtigkeit von gut lesbarem und ästhetisch ansprechendem Code.
- **Explizit ist besser als implizit.**
  - Klare und ausdrückliche Programmierstile werden gegenüber impliziten oder versteckten Mechanismen bevorzugt.
- **Einfach ist besser als kompliziert.**
  - Python strebt danach, einfach und direkt zu sein, sowohl in der Ausführung als auch im Verständnis.
- **Komplex ist besser als kompliziert.**
  - Während Einfachheit bevorzugt wird, erkennt Python an, dass komplexe Systeme nicht unnötig kompliziert gestaltet werden sollten.
- **Flach ist besser als verschachtelt.**
  - Dies fördert einen Code-Stil, der darauf abzielt, tiefe Verschachtelungen von Strukturen zu vermeiden, was die Lesbarkeit erhöht.
- **Lesbarkeit zählt.**
  - Ein grundlegender Aspekt von Python ist, dass der Code nicht nur funktionieren, sondern auch für andere leicht zu lesen und zu verstehen sein sollte.
- **Spezialfälle sind nicht speziell genug, um die Regeln zu brechen.**
  - Allgemeingültigkeit wird bevorzugt; Ausnahmen und Sonderfälle sollten die grundlegenden Designprinzipien nicht untergraben.
- **Praktisch schlägt rein.**
  - Pragmatismus wird in der Entwicklung von Python hochgeschätzt, manchmal auch auf Kosten der Perfektion.
- **Fehler sollten nie stillschweigend passieren.**
  - Programme sollten explizit fehlschlagen oder Fehlermeldungen ausgeben, anstatt stillschweigend falsche Ergebnisse zu liefern.

Weiterentwicklung

- Heute **professionelle Weiterentwicklung** im Python Steering Council

- **Python Steering Council**
  - Die Leitung von Python liegt nun beim Python Steering Council, das jährlich von aktiven Python Core Entwicklern gewählt wird.
  - Dieses **Gremium** besteht aus fünf Mitgliedern, die Entscheidungen über die Python-Entwicklung leiten und den PEP-Prozess (Python Enhancement Proposals) überwachen.
- **Python Enhancement Proposals (PEPs)**
  - PEPs sind Vorschläge für Verbesserungen an der Sprache oder dem Ökosystem, die von jedem in der Community eingereicht werden können.
  - Diese Vorschläge werden diskutiert und müssen vom Steering Council genehmigt werden, bevor sie implementiert werden.
- **Community-Beitrag**
  - Python profitiert stark von seiner aktiven und engagierten Community. Viele Entwickler aus der ganzen Welt tragen durch Code-Beiträge, Fehlerberichte, Dokumentation und mehr bei.
  - Es gibt zahlreiche Konferenzen und Meetups, wie PyCon, die dazu dienen, die Gemeinschaft zu stärken und Weiterbildung zu fördern.

(relative) neue Feature in **Python 3.14** sind
- **explizite Typen**
- oft recht speziell (gehen wir später drauf ein)

Anwendungsgebiete

Verwendung von Python

Prinzipiell ist Python "funktional vollständig", das heisst man kann theoretisch jedes Problem mit dieser Sprache lösen. In der Praxis haben sich aber folgende **Haupteinsatzgebiete** herausgebildet:

Python wird in vielen unterschiedlichen Bereichen eingesetzt, darunter

- **Webentwicklung** (server z.B. django)
- **Skripte** (kleine Programme zur Systemsteuerung)
- **Datenanalyse** & Verarbeitung & Management (**SQL**…)
- **künstliche Intelligenz** (pytorch) mit c++/cuda backbone auch riesige Daten
- wissenschaftliches Rechnen (abacus)
- **Prototypen** & Tests
- und mehr (prinzipiell überall)

Große Organisationen und Unternehmen nutzen Python aufgrund seiner Vielseitigkeit und Einfachheit.

Anti Anwendungsgebiete

Der Komfort der Programmiersprache Python kommt leider mit Nebenkosten, nämlich ist die Sprache **interpretiert,** was zur Folge hat dass sie grob um den **Faktor 5-100 langsamer** ist sein kann als andere '**kompilierte**' Sprachen. Bei Rechnern welche Milliarden Operationen pro Sekunde ausführen können ist das oft kein Problem aber man sollte es im Hinterkopf behalten.

Aus diesem Grund wird Python selten oder nie in den folgenden Gebieten eingesetzt:

- Kernel vom **Betriebssystem**
- Game **Engines** (wohl aber als Scriptsprache oder kleine Spielchen)
- TPU/GPU **Kernel** (spezielle Prozessoren & Rechenoperationen in der **KI**)

Oft ist es aber so dass Python über so genannte **Bibliotheken** (Module) indirekt auf schnelle Programme zugreifen kann welche dann in anderen Sprachen geschrieben wurden.

… **Mojo**??? compilierbares 'python' jetzt open source

Vorteile / Nachteile

Oft sofort verfügbar
Sehr **einfach**
Kein minutenlanges Warten auf compiler (c++) / aber **100 mal langsamer** in Ausführung!
Leichte **Erweiterung mit schnellen Komponenten** (**rust**/js...)
+ Lose Kopplung gut für schnelle Entwicklung / schlecht für Riesenprojekte
- Lose Kopplung: **Überraschungen/Fehler** bei zu viel Flexibilität
Abstrakt, höhere Konzepte ohne ins Detail gehen zu müssen
Sehr **viele Bibliotheken** (Komponenten, z.B. Maus Steuerung etc)
Sehr **einfache Installation** von Komponenten

Aufgaben

1. Beispiele und Gegenbeispiele von Zen of Python finden (google oder selbst?)

Aufgaben

Python Installieren

Maschinen Code / Assembly

- Rechenschieber (nicht automatisch)
- Patch Kabel
- Lochkarten
- Maschinencode (ADD $x $y) *register* manuell manipulieren
- C Fortran Cobol Go *Rust* "higher"
- Lisp (abstrakt und einfach)
- Python
- Deutsch / English / ...

WO schreibe ich Python?

•Die Programmiersprache Python kann an verschiedenen Stellen eingegeben werden:

Je nach Geschmack und Anwendungsfall kann man dabei wählen zwischen

- **Kommandozeile "Interpreter"** python.exe zum 'Evaluieren'
- **Text Editor (print!!)**
- **Entwicklungsumgebung (IDE integrated development environment) <====**
- **Notebook (jupyter)**
- Browser (!) z.B. https://www.online-python.com/
- **Lambda (serverless)** online Funktionen zur Verfügung stellen
- **App (pythonista)** iPhone / Android

- **Agenten: Deutsch / Englisch**
- **AI "2022" ChatGPT Claude Gemini... (in IDEs)**
- **Agentic AI "Neu 2025" *Codex* OpenAI / *Claude* Code Anthropic** freie

Einsatz von AI

Vokabeln **Lernen**
Ordnung!
Verständnis **muss man selber haben**
Erstmal nur als *Lernhilfe*, zum Fragen beantworten
(erstmal) NICHT zum Programme Schreiben

Programme Schreiben: vorletzte Woche

Agenten letzte Woche

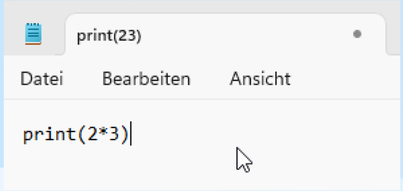
WO schreibe ich Python?

- **Text Editor**

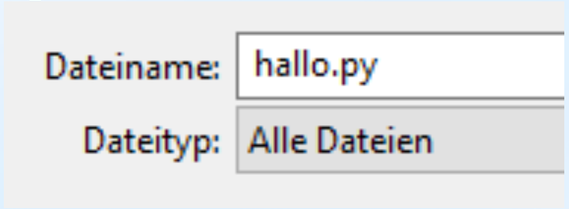
Python ist eine jener schönen Sprachen, welche man direkt im Editor schreiben kann, ohne Zuhilfenahme von besonderen Programmen.

Ein typisches erstes 'Programm' mit welchem jeder anfängt ist:

```
print("Hallo Welt")
```



Speichern unter, dann "Alle Dateien" damit .txt Endung weggeht.



Auch wenn Text Editoren zum schreiben von kleinen Programmen oder zum modifizieren von grösseren Programmen oft genügen, nutzt man in der Praxis meist doch eine der anderen Optionen:

WO schreibe ich Python?

• Entwicklungsumgebung (IDE)

Text Editoren mit Zusatzfunktionen werden oft IDE (Integrated Development Environment) genannt

Die wichtigsten der zahllosen Zusatzfunktionen:

• Syntax Highlighting:

```
print("Hallo Welt")
```

- **Fehleranzeige**

```
printt("Hallo Welt")
     ^ rote Punktlinie und schwarz statt blau: hier stimmt was nicht
```

- Programmausführung "▶️ RUN"
- Debuggen: Suchen von Fehlern
- Code Vervollständigung (demo)
- … viele weitere Zusatzfunktionen

WO schreibe ich Python?

- **Entwicklungsumgebung**
- **(IDE Integrated Desktop Environment)**

Beliebte IDEs:

- **Visual Studio Code (VSCode)**
- **Jetbrains PyCharm**
- **IDLE (kommt mit python mit)**
- **Jupyter Notebook**
- **Text Editoren mit Python Plugin:**
  - **Sublime**
  - **...**

WO schreibe ich Python?

- **Kommandozeile (englisch: shell)**

```
~/ python
Python 3.12.2 (main, Feb  6 2024, 20:19:44) [Clang 15.0.0 (clang-1500.1.0.2.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
extensions v1.0.0 loaded
Python shell history and tab completion are enabled.
>>> 2 * 3
6
>>>
```

Wie im Texteditor gibt es einen Cursor
Unterschied: Eingabe wird sofort nach enter/return ausgeführt

```
print("Hallo Welt")  # "print" kann in der shell weggelassen werden

"Besserer Taschenrechner"
```

WO schreibe ich Python?

- **Kommandozeile (english: shell / console / repl / terminal)**
  - **1. Betriebssystem:**
    - **cmd.exe / powershell.exe (Windows)**
    - **Terminal.app (Mac)**
    - **bash, zsh (Mac, Linux, Windows) …**
    - **in IDE**
  - **2. Python Kommandozeile "Interpreter"**
    - **python.exe >>>**
    - **python3**
    - **in IDE**
    - **online**

- **Der Begriff Kommandozeile ist etwas zweideutig, aber die beiden Konzepte gehören zusammen.**
- **"Ausführung der Python Kommandozeile innerhalb der System Kommandozeile"**

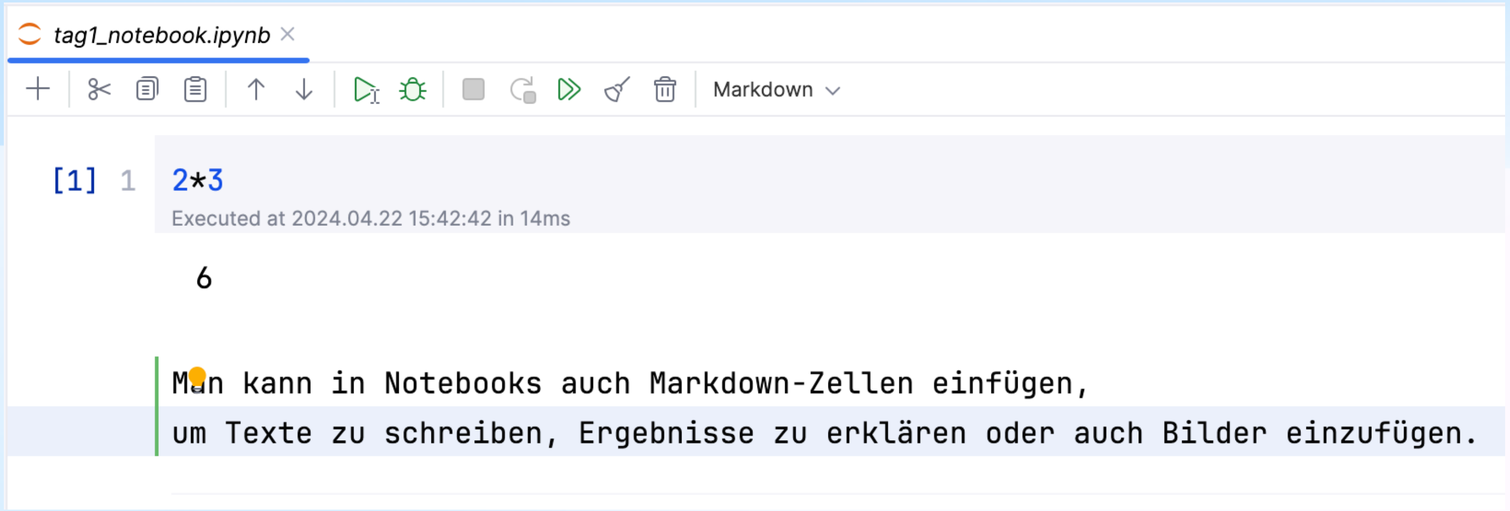
"Eine Shell ist eine Benutzerschnittstelle, die es ermöglicht, durch Eingabe von Befehlen und Betrachten von Ausgaben mit dem Betriebssystem zu interagieren."

```
~/ python.exe test.py   # Aufruf über Betriebssystem Kommandozeile
hallo
```

```
print("Hallo Welt") << "print" kann in der shell weggelassen werden
```

WO schreibe ich Python?

- Notebook
  - jupyter
  - im Browser
    - https://jupyter.org/try
  - in IDE



WO schreibe ich Python?

- **Browser**

Alle der vorher genannten Möglichkeiten zum Schreiben / Ausführen von python lassen sich heutzutage ohne installation direkt online nutzen:

- **Text Editor (⚠️ ohne Ausführung)**
- **Entwicklungsumgebung https://vscode.dev/ …**
- **Kommandozeile https://jupyterlite.github.io/demo/lab/index.html**
- **Notebook https://jupyter.org/try**
- **ChatGPT! (In der pro version)**

Je nach Fall ist es aber meist komfortabler lokal (auf dem eigenen Rechner) zu entwickeln.

…

Wie starte ich Python?

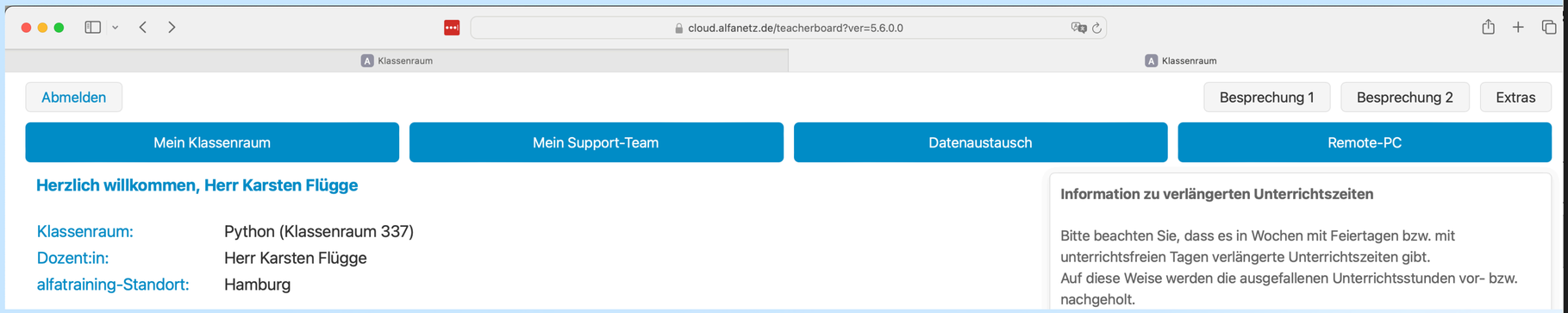
- Die Programmiersprache Python kann an verschiedenen Stellen ausgeführt werden:

Je nach Geschmack und Anwendungsfall kann man dabei wählen zwischen

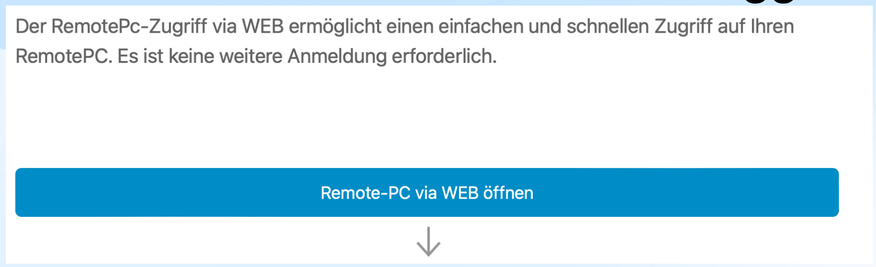
- ~~Text Editor~~
- Datei per rechter Maus?
- App per Doppelclick?
- Kommandozeile/Console python.exe <programm-name>
- Entwicklungsumgebung
- Notebook (IDE, Server oder lokaler Server)
- Browser (!)

Remote-PC

Einloggen über cloud.alfanetz.de "Remote-PC"

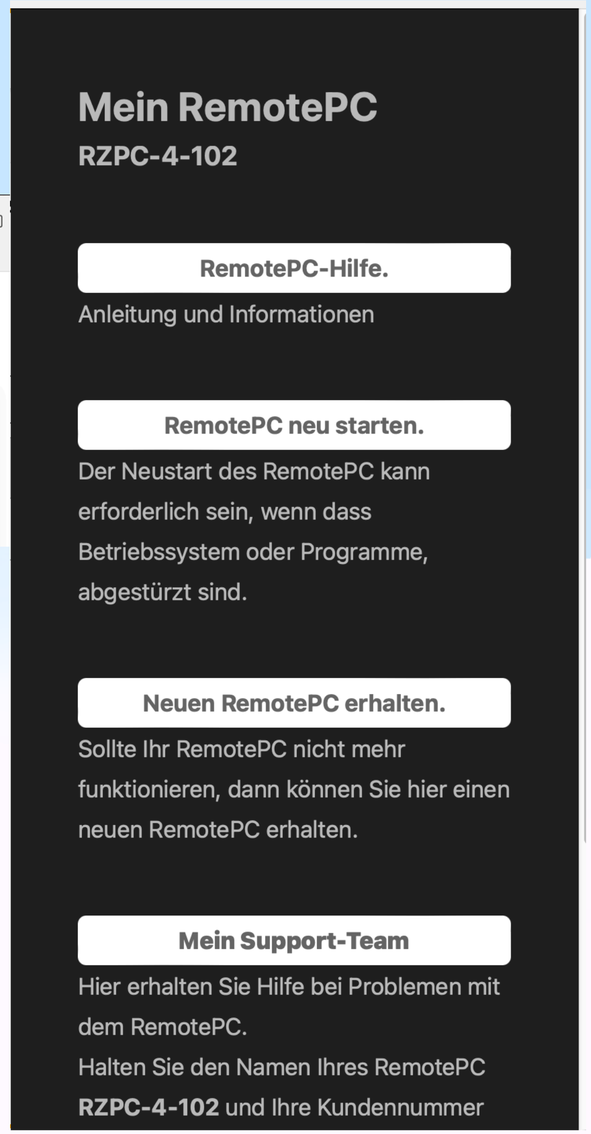


Erst mal alle über "via WEB" einloggen:



User/Password wie bei cloud.alfanetz.de

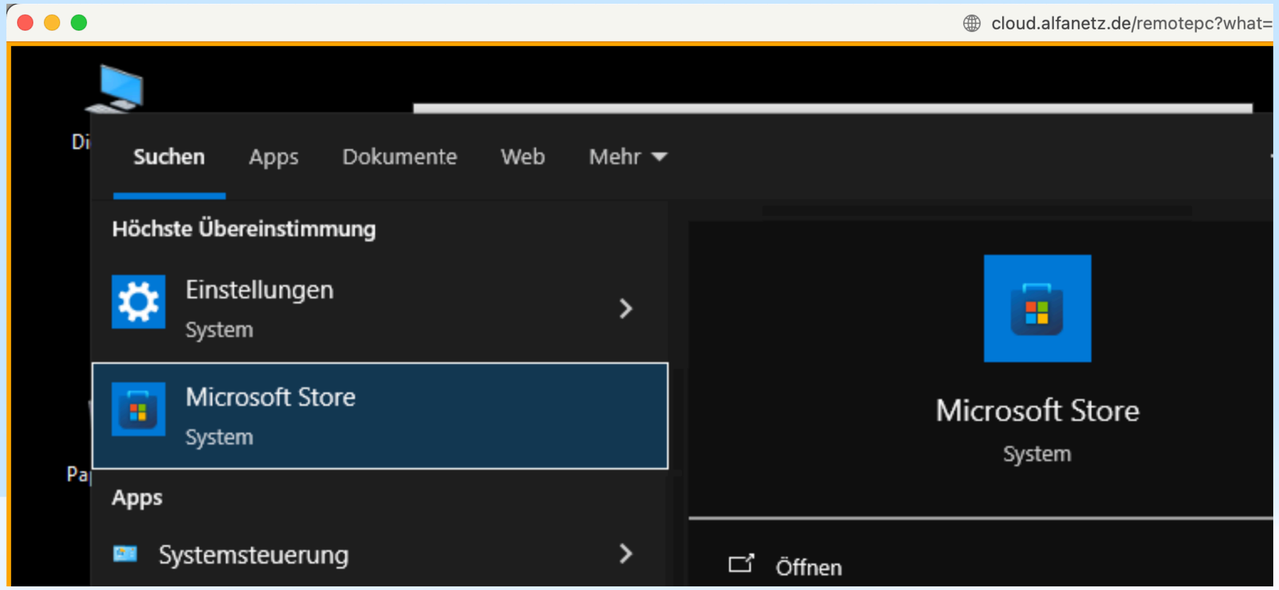
Maus am Rechten Rand öffnet "Mein RemotePC Leiste":

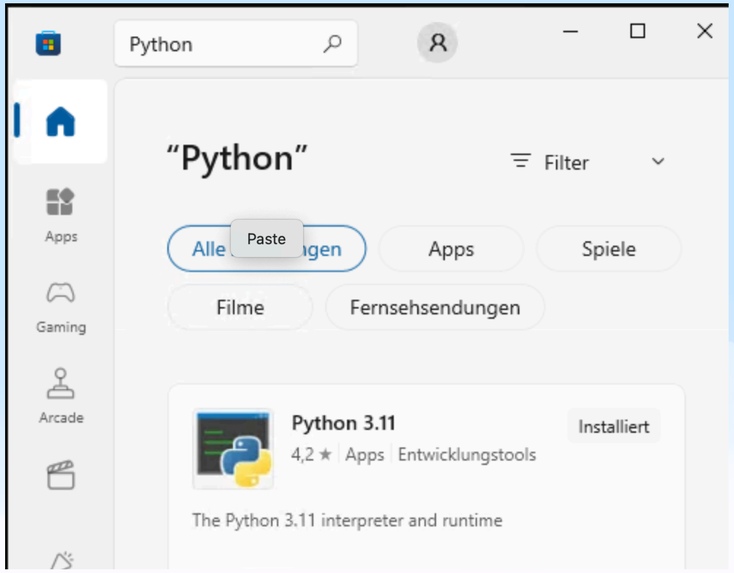


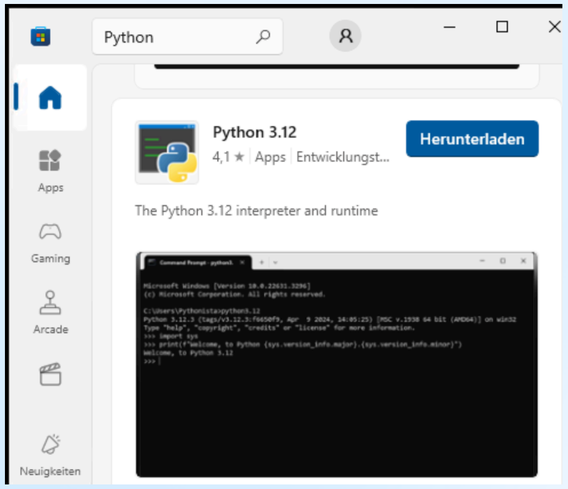
Installation

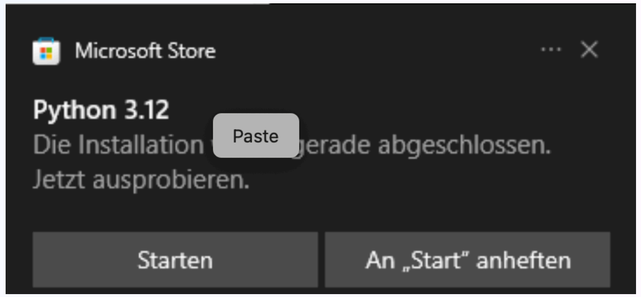
Installation von 3.12 auf Remote Desktop

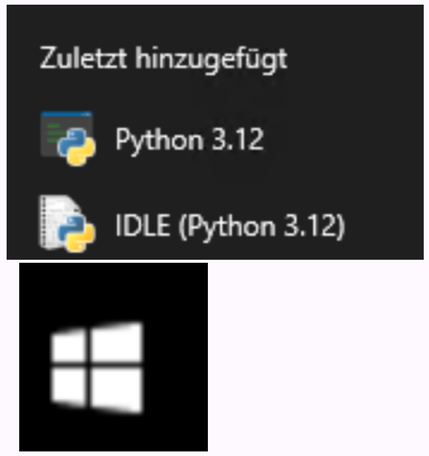
im Start menu "Store" schreiben ...









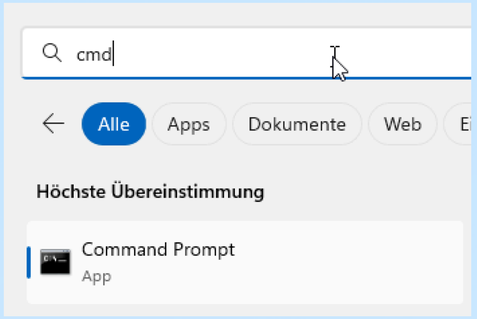


Wie starte ich Python?

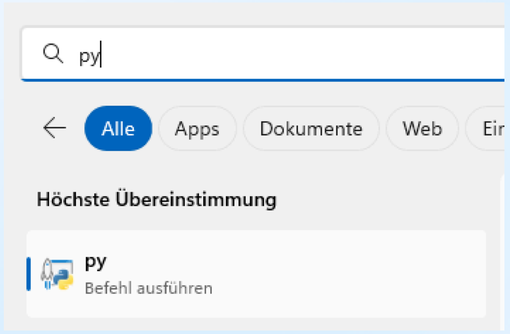
- **Kommandozeile**

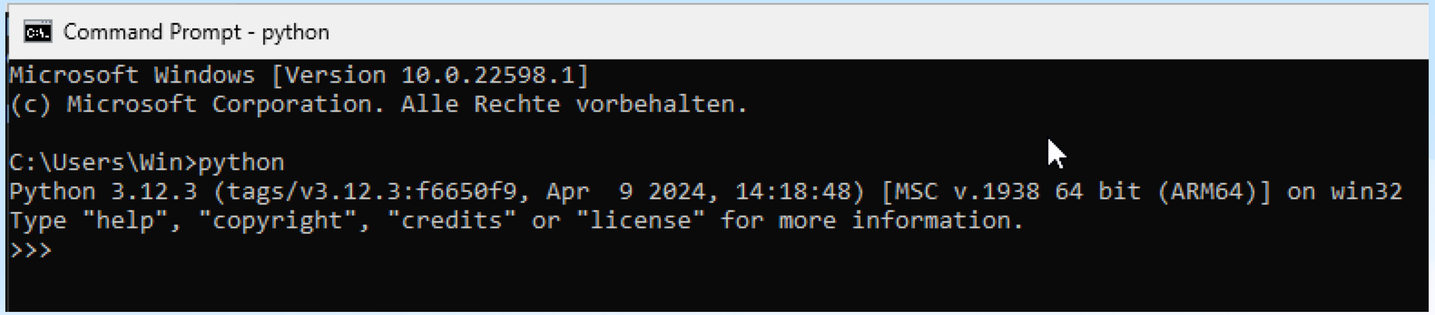
1. Starten von Kommandozeile python.exe …

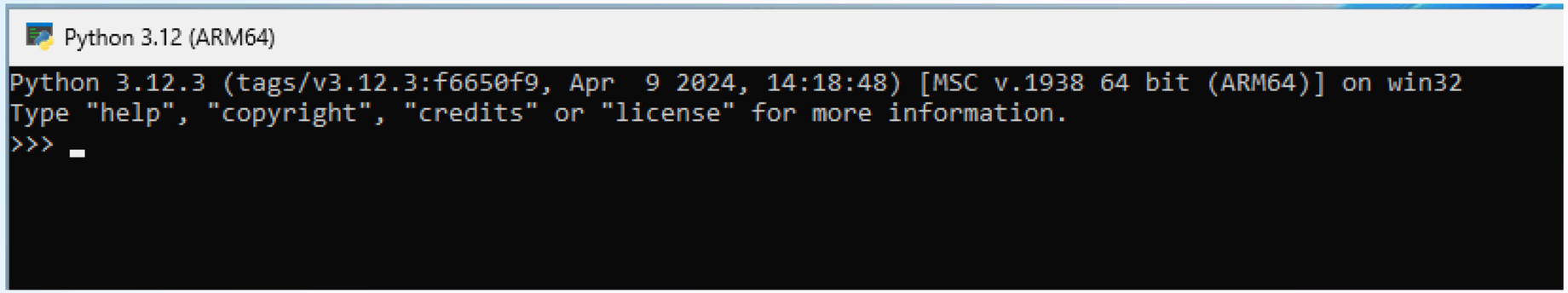
Windows



cmd oder direkt





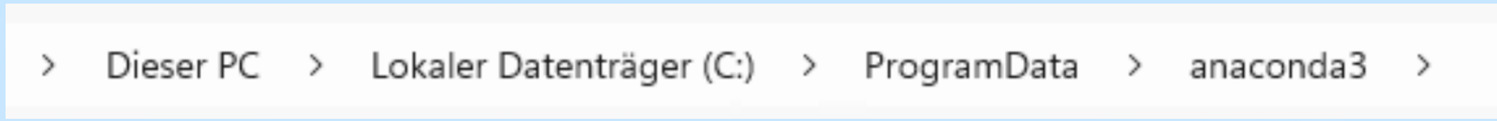


Wie starte ich Python?

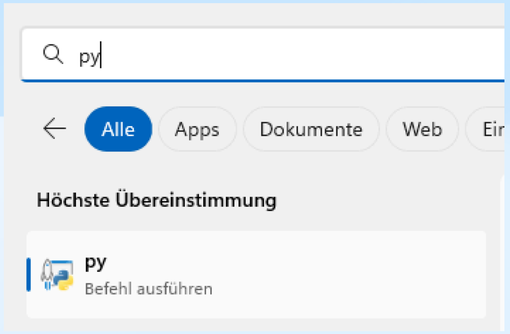
- **Kommandozeile**

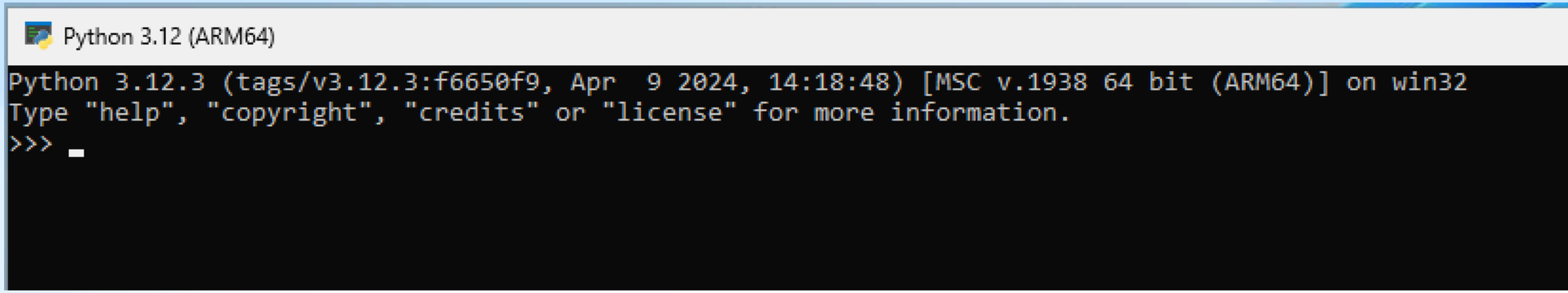
1. Starten von Kommandozeile python.exe …

Windows



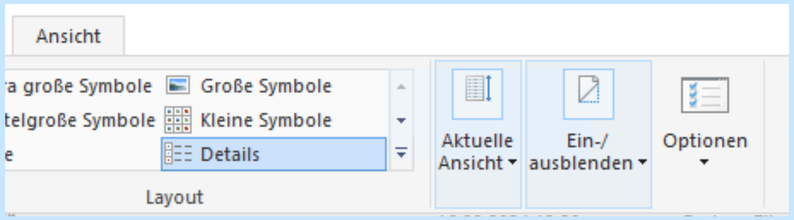
direkt





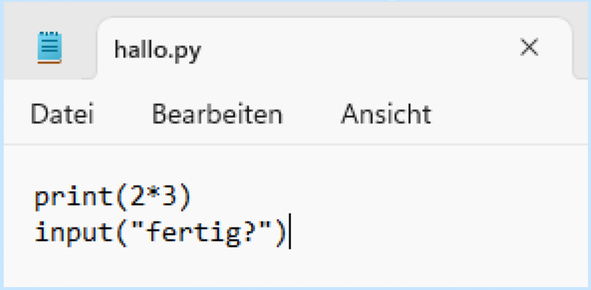
Damit lassen sich dann auch .py Dateien verknüpfen und direkt ausführen.

Wie starte ich Python?

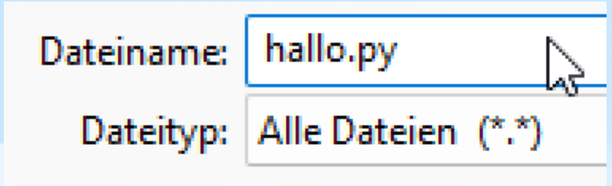


Erstellen des ersten "Programmes"

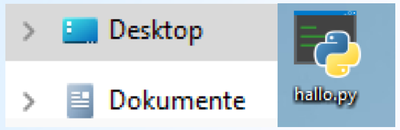
- **Text Editor z.B. notepad.exe**



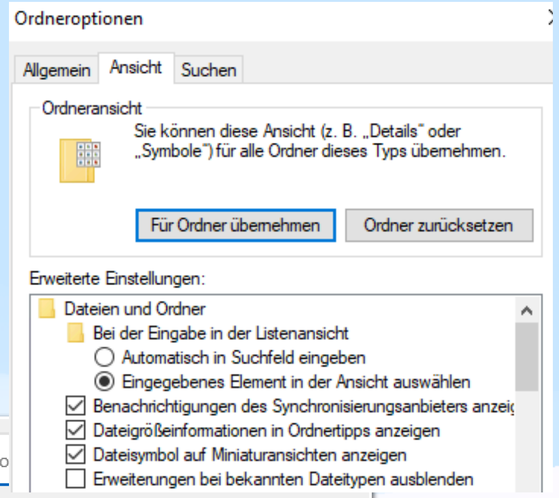
- **💡 Text Editor Datei speichern als: hallo.py**

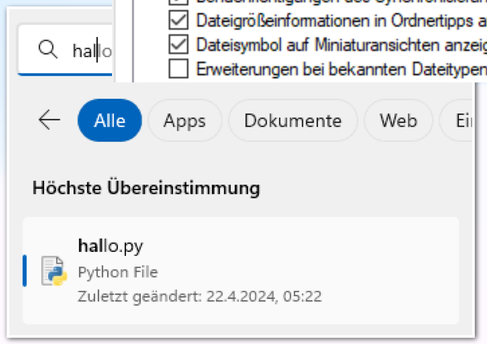


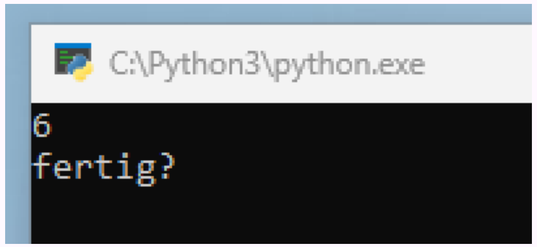
- **".py" ist die Dateiendung für python ⚠️ Dateityp .txt ändern! ⚠️**



- **hallo.py am Speicherort finden**
- **doppel-clicken**
- **⚠️ Falls das Fenster sofort wieder zugeht: input("fertig?")**

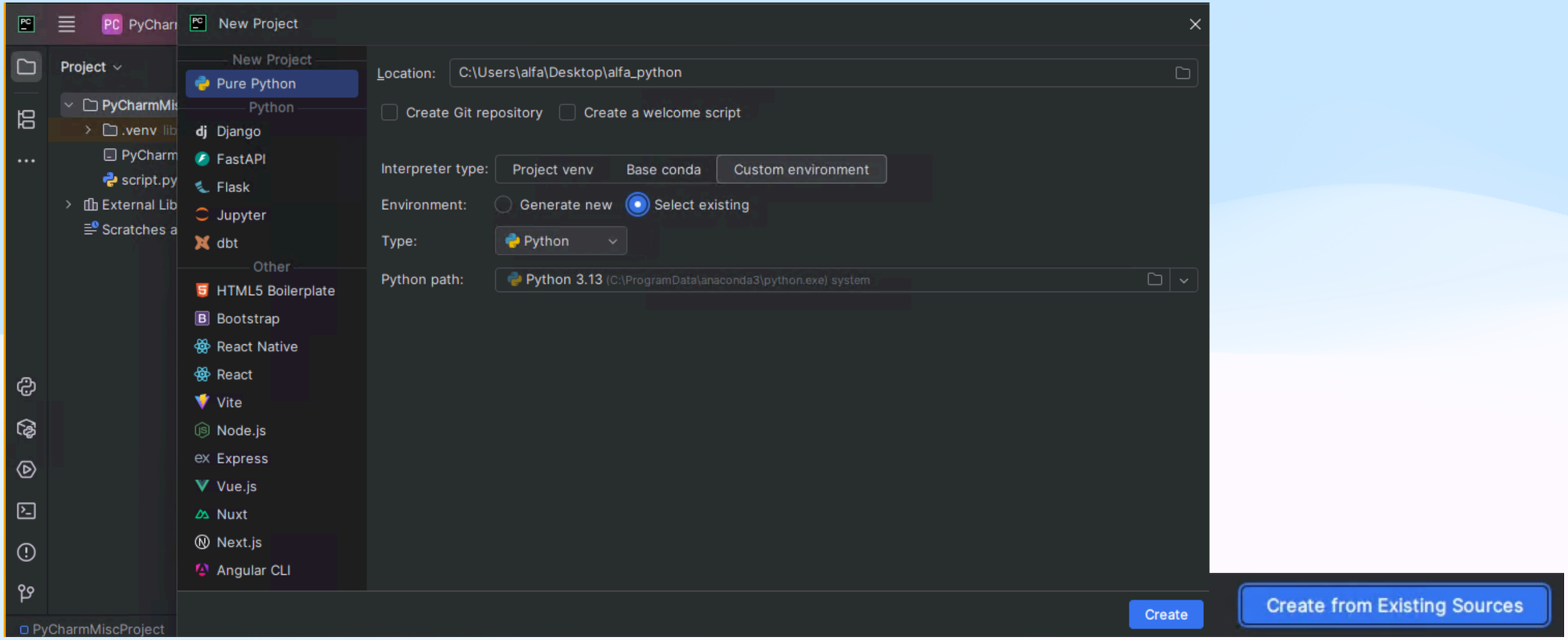






Projekt Anlegen

- Neues Projekt
- am besten direkt auf "alfa_python"
- am besten ohne venv via "Custom Environment" :
-



Wie starte ich Python?

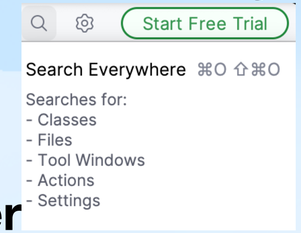
- **Entwicklungsumgebung**

Ausschau halten nach Pfeil "RUN"

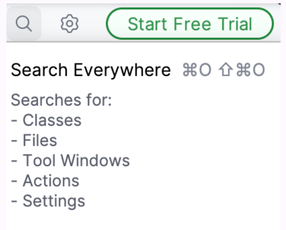
Pycharm

VS Code launch.json

Jupyter



Tipp: in Pycharm zweimal Shift : SearchAnywhere™ => Run, Debug ...
z.B. Comment, Reformat Code



Wie starte ich Python?

Im Browser

z.B.:

- **https://www.online-python.com/**

Warum Python?

"Python-Programme können in weniger Zeilen mehr erledigen, als das bei anderen Programmiersprachen der Fall ist. Die Syntax von Python hilft »sauberen« Code zu schreiben, der sich leicht lesen, leicht korrigieren und leicht erweitern lässt."

"Python wird für viele Zwecke eingesetzt: für kleine Spiele, für Webanwendungen (Server), zum Lösen von Problemen in der Geschäftswelt und zur Entwicklung interner Werkzeuge für alle möglichen Arten von Unternehmen. Auch in Wissenschaft und Forschung ist Python stark vertreten."

Begriff Syntax

Syntax ≈ "Erlaubter (Programm) Code"
Syntax ≈ Rechtschreibung / Grammatik ≈ "Zulässige Ausdrücke"

Die **Syntax** in der Programmierung bezieht sich auf die **Regeln**, die festlegen, wie Code geschrieben werden muss, damit der Computer ihn **verstehen und ausführen** kann. Man kann sich das ähnlich wie die **Grammatik** in einer Sprache vorstellen. Wenn man in der deutschen Sprache einen Satz bildet, muss man sich an bestimmte Regeln halten, wie die richtige Reihenfolge von Subjekt, Prädikat und Objekt, damit der Satz verständlich ist. In ähnlicher Weise gibt es in jeder Programmiersprache spezifische Regeln, wie Befehle, Funktionen und andere Anweisungen formuliert werden müssen.

invalid syntax
"drei minu 5 nie gut wie"

- Beenden von 'Sätzen' am Ende des Dokumentes.
- ... genaue **Syntax** lernen wir in den nächsten Wochen Schritt für Schritt

Syntax Fehler

Wenn die **Syntax ("Rechtschreibung") nicht stimmt bekommen wir Fehler:**

z.B.

```
    name=input( "what is your name?' )
                ^
SyntaxError: unterminated string literal (detected at line 1)  # hier muss " hin!

    print(3 > = 2)
            ^
SyntaxError: invalid syntax  # darf kein space zwischen > und =
```

...

Programm kann nicht gestartet werden.

Andere Fehler

Werden nicht im Vorfeld erkannt! Erst wenn sie im *Code* auftauchen.

```
print("3" > 2)

TypeError: '>' not supported between instances of 'str' and 'int'

"Text und Zahl darf man nicht vergleichen"
```

Logische Fehler:

Programm läuft ohne Fehler, aber tut nicht was es soll!
(Denkfehler, kleine Gemeinheiten)

Fehler sind gut! Sie sagen uns wo etwas nicht stimmt.

Extrembeispiel Rust: fast alles ein Fehler,
wenn kein Fehler da ist, dann läuft Programm 95% korrekt.
Kann von Agenten geschrieben werden.

Grundkonzepte (Vorschau)

Variablen und Datentypen

- **Variablen** sind Namen, die verwendet werden, um Daten im Code zu speichern. Man kann sich eine Variable wie eine Schublade vorstellen, in der Informationen aufbewahrt werden.
- **Datentypen** bestimmen, welche Art von Daten in einer Variablen gespeichert werden kann. Die häufigsten Datentypen in Python sind:

```
x = 5         # int (ganze Zahlen)
y = 3.14      # float (Fließkommazahlen)
name = "Python"  # str (Zeichenketten)
is_easy = True   # bool (Boolesche Wahrheitswerte, also True oder False)
```

Kontrollstrukturen

- **Kontrollstrukturen** steuern den Fluss eines Programms. Es gibt Bedingungen und Schleifen.
- **Bedingungen** verwenden `if`, `elif` und `else` für Entscheidungen.
- **Schleifen** wiederholen Code-Blöcke mit `for` oder `while`.

```
if x > 0:
    print("x ist positiv")
else:
    print("x ist negativ")

for i in range(5):
    print(i)
```

Funktionen

- **Funktionen** sind wiederverwendbare Codeblöcke, die spezifische Aufgaben erfüllen. Funktionen helfen dabei, Code übersichtlich und wartbar zu halten.
- Eine Funktion wird definiert mit `def` und kann Parameter enthalten, die beim Aufruf der Funktion Werte übernehmen.

```
def begruessung(name):
    return "Hallo " + name

print(begruessung("Welt"))
```

Grundkonzepte (Vorschau)

Literale "Konstanten"

Zahlen

42 # Integer-Literal

3.14 # Float-Literal "Fliess*kommaz*ahl"

Texte

"Hallo" # String-Literal

'Hallo' # ebenfalls String-Literal

Wahrheit / Existenz

True # boolescher Wert 3>2? Ja = True

None # 'nichts' spezieller Einzelwert (null, nil)

Variablen und Datentypen

- **Variablen** sind Namen, die verwendet werden, um Daten im Code zu speichern. Man kann sich eine Variable wie eine Schublade vorstellen, in der Informationen aufbewahrt werden.
- **Datentypen** bestimmen, welche Art von Daten in einer Variablen gespeichert werden kann. Die häufigsten Datentypen in Python sind:

```
x = 5          # int (ganze Zahlen)
y = 3.14       # float (Fließkommazahlen)
name = "Python"  # str (Zeichenketten)
is_easy = True   # bool (Boolesche Wahrheitswerte, also True oder False)
```

Universelle Programmierkonzepte

Viele der in diesem Kurs gelernten Konzepte lassen sich bestens auf andere Programmiersprachen übertragen, so dass ein Kurs in Python auch ein Tor in andere Sprachen sein kann.

Aufgaben Tag 1

1-0 Mit Python *PyCharm* Umgebungen vertraut machen, eine finden in welcher man sich komfortabel fühlt

1-1 Experimentieren (Chaos Monkey)
ohne viel vorweg zu nehmen einfach probieren, was dieser >>> "Taschenrechner" alles kann, bzw wie er auf unvoreingenommene Eingaben reagiert
1-1a Welche mathematischen Operationen/Funktionen funktionieren so wie gewollt? 3-7 … sqrt(wurzel) …

1-2 Interessante oder unerwartete Reaktionen vom "Taschenrechner" festhalten 3^3 ≠ 33 "potenz!"**

1-3 Unbegrenzte Fähigkeiten:
Aufschreiben welche Ideen man am liebsten Umsetzen würde (kurzfristig, mittelfristig, langfristig)

1-4 Weiter vertraut machen mit Konsole / IDE
SEARCH PyCharm ctrl+shift+A VS-Code ctrl+shift+P

1-5 Freiwillig: Installation von Python & PyCharm auf eigenem Rechner (empfohlen)

1-6 Vokabeln lernen ;)

1-7 Lesen https://edube.org/10004255/learn/pe-1/python-essentials-1-module-1-2

Aufgaben Tag 1 b

- 1-8 gebe icon/emoji aus 😺
- 1-9 berechne 3 hoch 3 (wie? frage Google / ChatGPT!)
- 1-10 schaue mit help() die print Funktion an
- 1-11 `import this`

- Was passiert bei Fehlern? Git print("Hier fehlt eine Klammer" ein und schaue was passiert
- wir haben ja schon die input() Funktion gesehen. Versuche sie zu verwenden:

```
print('Hallo, wie heißt du?') # jetzt den Namen einlesen:
meinName = input()
print('Schön dich zu treffen, ' + meinName)
```

Begriff Statement

In der Programmierung ist ein "Statement" oder eine Anweisung eine einzelne Operation oder Handlungsaufforderung, die in einer Programmiersprache ausgedrückt wird und von einem Computer ausgeführt werden kann. Statements sind die Bausteine eines Programms, die dem Computer spezifische Befehle geben.

Pause

Woche 1

Mo: Begrüßung, Vorstellungsrunde, Einführung alfaview & digitale Lernumgebung
· Grundlagen Python (Geschichte, Konzepte, Verwendung, Syntax, Lexis/Semantik, PEP-8)
· Interpreter vs. Compiler, Zahlensysteme (binär, oktal, hex), Scientific Notation

Di: Zahlen, Zeichenketten
· Datum und Zeit
· Standardeingabe und -ausgabe

Mi: Numerische Operatoren
· Vergleichs-, logische und bitweise Operatoren

Do: Datentypumwandlung
· list, tuple, dict, set
· List-Funktionen und -Methoden

Fr: List-Funktionen und -Methoden
· Verzweigungen und Schleifen (if, for, while)

Woche 2

Mo: Mitgliedsoperatoren
· String-Basics (escaping, multiline strings)
· Operatoren priorisieren und binden

Di: Eigene Funktionen definieren
· Variablen

Mi: Parameter und Argumente
· Rückgabewerte

Do: Rekursion
· Namensräume

Fr: Funktionale Programmierung
· Parameterarten (positional, keyword, mixed)

Woche 3

Mo: Defaultwerte
· Shadowing und global keyword

Übersicht Kursthemen

Woche 1

Mo: Begrüßung, Vorstellungsrunde, Einführung alfaview & digitale Lernumgebung
· **Grundlagen** Python (Geschichte, Konzepte, Verwendung, **Syntax**, **Lexis/Semantik, PEP-8**)
· **Interpreter** vs. Compiler, **Zahlensysteme** (binär, oktal, hex), Scientific Notation

Di: **Zahlen, Zeichenketten**
· **Datum** und Zeit
· Standard**eingabe** und -ausgabe

Mi: Numerische **Operatoren**
· Vergleichs-, logische und bitweise Operatoren

Do: Datentypumwandlung
· **list, tuple, dict, set**
· List-Funktionen und -Methoden

Fr: **Verzweigungen** und Schleifen (if, for, while)

Woche 2

Mo: Mitgliedsoperatoren
· **String**-Basics (escaping, multiline strings)
· Operatoren priorisieren und binden

Di: Eigene **Funktionen** definieren I
· Variablen

Mi: **Parameter** und Argumente
· Rückgabewerte

Do: Rekursion
· **Namensräume**

Fr: **Funktionale** Programmierung II
· **Parameter**arten (positional, keyword, mixed)

Übersicht Kursthemen

Woche 3

Mo: Defaultwerte
· **Shadowing** und **global** keyword
· None und Rückgabe ohne Wert

Di: **Fehlerbehandlung** (try/except, typische Fehlertypen, Exception-Hierarchie)
· Fehlerweitergabe, Programmunterbrechungen, Strukturierung der except-Blöcke
· OOP: Python-Klassen

Mi: Python-**Klassen**
· Methoden

Do: **Unveränderliche** Objekte

Fr: **Datenklasse**

Woche 4

Mo: **Vererbung**
· Projektarbeit beginnt

Di–Mi: Erstellung und Umsetzung der Projektarbeit (ganztägig)

Do: Präsentation, Abschlussbesprechung, Bewertung
· Zertifizierungsvorbereitung (Wiederholung prüfungsrelevanter Inhalte)

Fr: PCEP-Zertifizierungsprüfung ("Certified Entry-Level Python Programmer", englisch)
· Präsentation, Abschlussbesprechung, Bewertung

Projektarbeit Ausblick

Kann schon während Woche 2/3 begonnen werden.
Offiziell ab Mittwoch von Woche 4, aber gerne schon ab Dienstag.

Thema eigener Wahl und eigenem Interesse.

Ansonsten kann ich Thema vorschlagen/vorgeben.

Individuell oder Gruppe je nach Geschmack.

Datenaustausch als Verzeichnis

Windows:

In Konsole (cmd) ausführen:
net use A: https://d.alfanetz.de/remote.php/dav/files/DEINE_BENUTZERKENNUNG/ /
user:DEINE_BENUTZERKENNUNG

Mac:
mkdir austausch

```
mount_webdav -i  "https://d.alfanetz.de/remote.php/dav/files/D65263TQF/" austausch
```

Vokabular Tag 1

Kommandozeile (english: shell / console / terminal)
Kommandozeile "Interpreter" / read–eval–print loop (REPL)
Ausführen = Aufrufen = Starten = Run = Execute = laufen lassen
Bibliotheken (Modul)
Code / Programm(text)
Syntax ≈ Grammatik
Ausdruck (Berechnung)
Statement (Anweisung)
IDE
Pycharm / VSCode
Syntax Highlighting
Code Completion
GPT (Künstliche Intelligenz als Assistent)

print(…) Funktion / Statement: Ausgabe von Text, Zahlen, etc
input() Funktion : Einlesen von Text, Zahlen, etc
Tipp: _ für Resultat der letzten Berechnung im Interpreter