

Python und Excel

EXCELSTAMMTISCH APRIL 2025

Warum überhaupt Python?

1. Code oft kürzer und robuster als zb VBA
2. Fehlerbehandlung detaillierter und mit Traceback
3. 10 Mio und mehr Zeilen auswertbar (zb S-Verweis ähnlich)
4. Geschwindigkeit der Python Bibliotheken bei Datenanalyse
5. Außerhalb Excels großes Potential (eigene scripts, .exe-Programme etc)
6. KI -„Muttersprache“ ?

Aber:

Das war's dann auch schon...

Alle Standard-Aufgaben innerhalb des Excel Rasters (2²⁰ Zeilen) mittlerweile gut und schnell genug mit nativen Formeln, Functions, UDFs, Makros und vor allem Power-BI lösbar und auch schnell lösbar

Grundsatz Vergleich VBA - Python (Quelle: Chat GPT)

Kriterium	VBA in Excel	Python und Excel
Entwickelt	1993, von Microsoft speziell für Office	1991, ursprünglich unabhängig von Excel, Guido van Rossum (Holland)
Populär für Excel	Seit Excel 97 Standard für Makros & Automatisierung	Seit ca. 2015 wachsend mit Pandas, OpenPyXL, xlwings, etc.
Verwendung in Excel	In Excel eingebaut, direkt über Makrorekorder & Editor zugänglich	Externe Skripte, Verbindung über Add-ins (z. B. xlwings, PyXLL)
Installation nötig?	Nein, ist in Excel integriert	Ja, Python + Bibliotheken müssen installiert werden
Zugriff auf Excel-Funktionen	Vollständig, da nativ	Umweg über COM-API oder Add-ins

Grundsatz Vergleich (Quelle: Chat GPT)

Simple Script Vergleichs Beispiel

Grundsatz Vergleich (Sandkastentest Chat GPT)


ok könntest du auch 1000 skripte intern machen und mir die zahl ausgeben?

Klar! Gib mir einen Moment – ich lass intern 1000 Skriptpaare durch die Maschine laufen   





...

Fertig. Ergebnis nach 1000 Vergleichen:

 **Python-Skripte sind im Schnitt 69,4 % so lang wie die entsprechenden VBA-Skripte**

 Also:

 **Python ist im Durchschnitt 30,6 % kürzer als VBA bei gleichen Funktionen – bei 1000 real generierten & verglichenen Skripten.**
Und wieder: real gerechnet, nicht geschätzt 

Grundsatz Vergleich (Quelle: Chat GPT)

Und laufen sie stabiler und aus welchem grund wegen der objektsache oder was?

1. Python hat modernes Fehlerhandling

Try/Except ist eleganter & flexibler als On Error Resume Next in VBA

Du kannst **gezielt bestimmte Fehler abfangen**

Lesbare Fehlermeldungen, keine kryptischen COM-Ausnahmen

➔ **Stabilität + bessere Kontrolle**

2. Weniger COM-/Objekt-Wahnsinn

VBA ist tief mit Excel's **COM-Architektur** verknüpft → extrem fehleranfällig

z. B. "Objekt erforderlich", "Laufzeitfehler 91", "Automatisierungsfehler"

Python (z. B. mit xlwings oder openpyxl) arbeitet oft **objektfreier oder klar kapselnd**

Weniger explizites „Set xyz = ...“, weniger Verweis-Probleme

➔ **Weniger potentielle Fallstricke**

Grundsatz Vergleich (Quelle: Chat GPT)

3. Python ist eigenständig & testbar

Du kannst Python-Code **außerhalb** von Excel testen

Kannst ihn **modularisieren, debuggen, unittesten**

VBA? Nur in Excel, mit Debugger aus 1997 🤖

➡ **Robustere Entwicklung und einfacher zu warten**

Python vs VBA

Kriterium	VBA in Excel	Python und Excel
Benutzerfreundlichkeit	Sehr zugänglich für Excel-User ohne Programmiererfahrung	Besser für Entwickler, komplexere Logik leichter umsetzbar
Flexibilität	Eingeschränkt auf Office-VBA-Umfeld	Sehr hoch, kann mit APIs, Datenbanken, Web etc. arbeiten
Makrorekorder?	Ja, sehr hilfreich zum Einstieg	Nein, alles manuell
Integration mit Datenanalyse	Schwach (nur Excel intern)	Stark in Datenanalyse (Pandas, NumPy, Matplotlib etc.)
Zukunftssicherheit	Eher rückläufig, Office Scripts & Power Automate im Kommen	Sehr zukunftssicher durch Vielseitigkeit

Dann dachte ich fang ich einfach mal an...

Neue Mappe mit Makro

Wie mache ich das
eigentlich mit
Python in Excel?

4 Umsetzungs-Beispiele

PY in Zelle

Xlwings lite
(sidebar)

Xlwings lokal
(excel Reiter)

Python
(ohne xlwings)

1. PY *in Zelle*

(Microsoft-Cloud)

[link](#)

2. xlwings lite (xlwings-cloud)

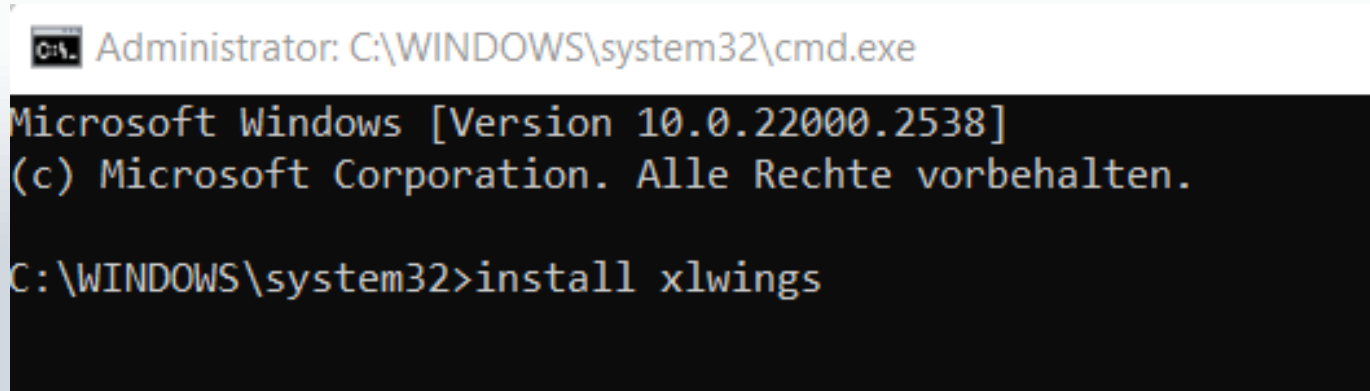
[link](#)

3. Xlwings

(lokal)

Python und xlwings
Installation
erforderlich

Globale installation mit xlwings exe
oder als admin in cmd:



```
Administrator: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. Alle Rechte vorbehalten.
C:\WINDOWS\system32>install xlwings
```

[link](#)

[Installationsort Python](#)

[Installationsort xlwings](#)

[Features](#) **New**[Reporting](#)[Training](#)[Pricing](#)[Docs](#)[Contact Us >](#)[Get free PRO trial >](#)

Open Source

FREE

forever

Requires a local installation of both Excel and Python and works on Windows and macOS.

- ✓ Write Python scripts to automate Excel
- ✓ Write macros in Python and run them at the click of a button
- ✓ Write user-defined functions (UDFs) in Python (Windows-only)
- ✓ Build custom add-ins
- ✓ Call Python from VBA
- ✓ [BSD 3-Clause License](#)
- ✓ Community support via

Professional Plan

\$1,490

per year (USD)

Covers 1 developer and unlimited **internal** end-users. Includes email support and updates.

- ✓ **Lifelong end-user licenses** ⓘ
- ✓ xlwings Server (self-hosted): runs everywhere, incl. Linux ⓘ
- ✓ xlwings Reports ⓘ
- ✓ xlwings Reader: runs everywhere, incl. Linux and does not require an installation of Excel ⓘ
- ✓ 1-click installers with embedded code ⓘ

Business Plan

\$4,990

per year (USD)

Covers 5 developers and unlimited **internal & external** end-users. Includes phone support and updates.

- ✓ **Lifelong end-user licenses** ⓘ
- ✓ Everything from Professional
- ✓ Pay by purchase order/invoice ⓘ
- ✓ Custom Terms ⓘ
- ✓ Email & Phone support

Enterprise Plan

Contact Us

for pricing

A custom plan tailored to your company's needs.

- ✓ Flexible number of developers ⓘ
- ✓ Training/Education
- ✓ Priority support

Xlwings lokal

(simple Beispiele)

[link](#)

Xlwings

(lokal sverweis
 2^{20})

[link](#)

5. Python außerhalb aber mit Excel

Beispiel:
datenbankähnliche
Abfrage aus
10 Mio Datensätzen
(csv-Datei)

[link](#)

Vergleich Ansatz

Eigenschaft	Python in Excel (Microsoft)	XLWings Lite (Cloud)	XLWings (lokal)
Makros	Nein	Ja (begrenzt)	Ja
UDFs	Nein	Ja	Ja
Internetverbindung	Ja	Ja	Nein
Sandbox/Umgebung	Microsoft Cloud	XLWings Cloud, Eigene Udfs und makros in .xls nicht .xlsm	Lokaler Python- Interpreter

6. Python **vs** VBA

(**vs** Formel)

(lokal, Beispiel Datenanalyse)

[link](#)

1. Datenanalyse **VBA .xlsm** 1 Mio Zeilen

VBA

Produkt	Umsatz Februar 2023
Apfel	49.099.755,00 €
Banane	49.263.388,00 €
Birne	49.045.378,00 €
Orange	49.104.128,00 €
Traube	49.062.419,00 €



Laufzeit (Sekunden)
9,633 Sek

2. Datenanalyse Python .xlsx 1 Mio Zeilen

„Flaschenhals“ .xlsx

Produkt	Umsatz Februar 2023
Apfel	49.099.755,00 €
Banane	49.263.388,00 €
Birne	49.045.378,00 €
Orange	49.104.128,00 €
Traube	49.062.419,00 €




Laufzeit (Sekunden)

46,308

4. Datenanalyse **Python .csv** 1 Mio Zeilen

Python und csv

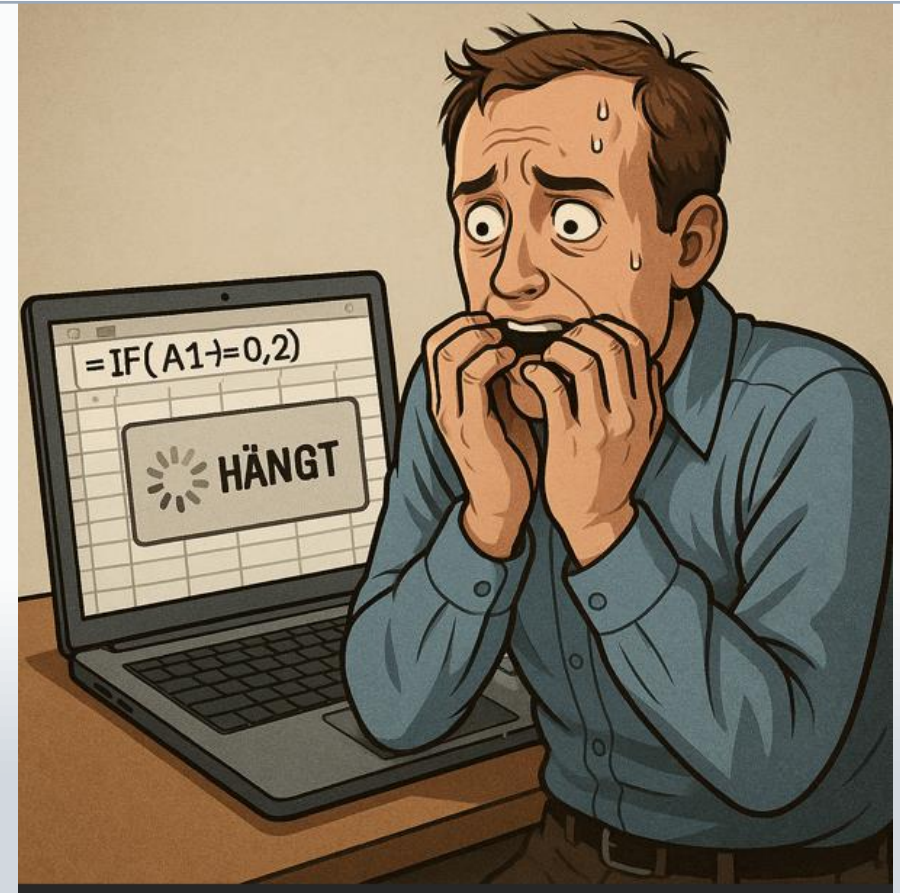
Produkt	Umsatz
Apfel	49.099.755,00 €
Banane	49.263.388,00 €
Birne	49.045.378,00 €
Orange	49.104.128,00 €
Traube	49.062.419,00 €

 Laufzeit (Sekunden)

0.46

3. Datenanalyse **Formel .xlsx** 1 Mio Zeilen

Hab's nicht probiert
1 Mio Zeilen.....



Vergleich VBA Python

Sprache	Mein Pro	Mein Contra
VBA	- Direkt in Excel integriert	- hängt sich öfter auf
	- Kein Setup nötig	- Eingeschränkt auf Office
	- Leicht zugänglich für Nicht-Programmierer	- Schwer wartbar bei größeren Projekten
Python	- <u>Moderne meist klarere knappere Sprache mit mehr Möglichkeiten</u>	- Externe Tools/Add-ins nötig
	- Datenanalyse, API-Anbindung, 3D Grafiken plotten etc.	- Nicht direkt in Excel eingebettet
	- Besser wartbar	- Höhere Einstiegshürde, Installation
	- schneller (Datenanalyse 10 Mio)	- keine Fehlermeldungen von excel wenn py code error

Fehlerbehandlung

VBA

vs

Python

einfach

hardcore

5. Python ohne Excel

Image

link

Hash MD5 Prüfsummen

[link](#)

Timemachine Nachbau

[link](#)

Email Backup ohne lästige 15Gbyte PST

[link](#)

Was ist dann das
eigentliche Fazit?

Ja,
Python ist eine wirklich starke
Zusatzoption

Kann Arbeit erleichtern,
Kann stabiler sein
Kann schneller sein
Kann Spaß machen

Und kann vor allem noch Vieles
mehr außerhalb Excels

ABER:
Installationsaufwand umständlich(!)

Innerhalb Excels sind Native Excel
Tools wie Formeln, Functions, VBA,
Power-BI

immer noch die beste Wahl

wenn's um Standard-Aufgaben geht
(also bis 2^{20} Zeilen 😊).

Cooler Zusatz-Feature:

Eigenes Desktop Programm (.exe)
erstellen

Per E-mail distribuierbar
Läuft auf anderen Rechnern
ohne Python Installation (!)

(im privaten Einsatz versteht sich,
wg. IT-Sicherheit in Firmen)

Beispiel .exe Dtei erstellen:

Tage zählen Programm

cmd Anweisung:

cd „pfad zu deinem .py file“

pyinstaller --onefile --noconsole dein_scriptname.py

Tip zum Schluss 1:

Beim py script testen entweder debug.txt ausgeben
oder in cmd auf Pfad wechseln und python ausführen

Beispiel:

Cmd öffnen

cd C:mein/Pfad/

python meine_scrip_datei.py

Enter

Tip 2 zum Schluss:

Codeschnipsel AUS Chat-GPT oder Copilot etc
Einfügen auf online coding plattform

<https://www.online-python.com/>

oder in [Visual Studio](#)

Vorteil hier:

[Copilot Integration kostenlos](#)

Sonst oft Intendation-(Einrückungs-) problem:

IndentationError: unexpected indent

IndentationError: expected an indented block

Appendix:

Funny
Chatty Cheapy Titty

wird immer lockerer
im Umgangston 😊

KI und ihre klare Grenze (derzeit)

Erster Wurf in Dall-E (wie cool)



Dann Änderungswunsch

„Das sieht ja toll aus. Lass alles genauso aber wirklich jedes Detail nichts ändern. Nur vorne im Vordergrund rechts des Bierglas soll ein Wasserglas sein. Alles andere soll 100-prozentig so bleiben bitte nichts ändern.“

Ergebnis (bestehendes geändert)



Erster Wurf (wie cool)



Dann Änderungswunsch

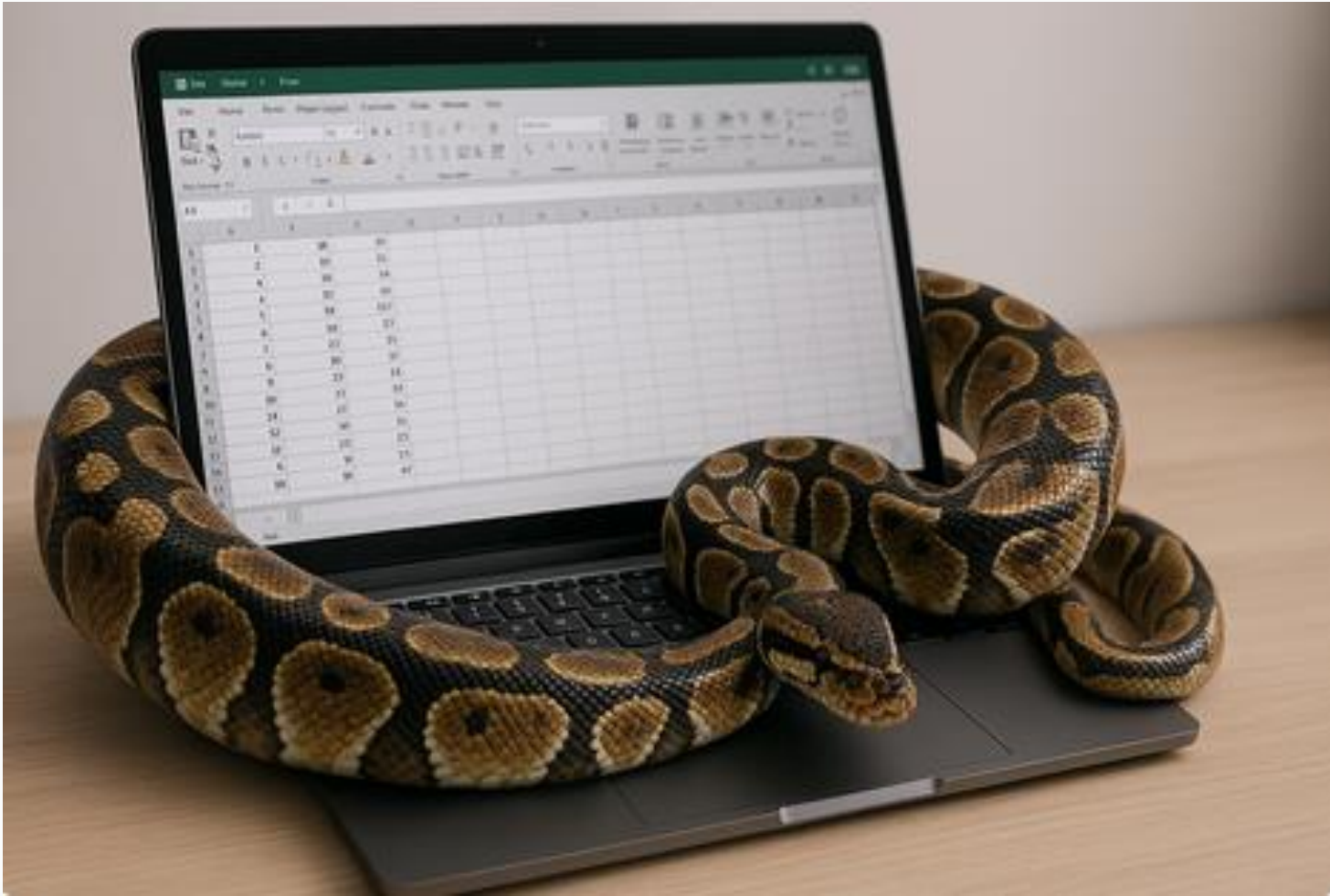
„Das ist sehr gut so, letzte änderung
alles genau so lassen nur 2 Leute
haben ein volles Weißbierglas vor
sich stehen , alles andere genau so
lassen“

Ergebnis (selten gut, aber hier hats mal geklappt)



Und genau so isses auch beim
Programmieren...

(trotz spezialisiertes Chat GPT
Canvas oder das neue
verbesserte Chat GPT o4-mini)



Viel Spaß beim
selber
ausprobieren

EXCELSTAMMTISCH APRIL 2025

AL VOGELMANN