

Python in Excel - Grundlagen

Inhalt

Neue Möglichkeiten im Controlling mit Python und Excel	2
Von VBA zu Python als weiterer Schritt im Controlling	9
Architektur, Chancen und Grenzen von Python im Unternehmenseinsatz	17
Schritt-für-Schritt-Anleitung für die Aktivierung von Python in Excel	25
Vom Excel-Anwender zum Datenanalysten mit Python	33
Empfehlungen aus dem Management-Handbuch	42

Neue Möglichkeiten im Controlling mit Python und Excel

Warum Python für das Controlling mit Excel hilfreich und oft auch notwendig ist. Vergleich der Anwendungsbereiche und mögliche Vorteile von Python mit Excel.

Zuletzt geändert am 02.03.2026



Das Controlling im Wandel: Von Tabellen zu Datenarchitekturen

Über Jahrzehnte war das Controlling stark tabellenorientiert. Excel entwickelte sich zum universellen Werkzeug für:

- Budgetplanung
- Forecast
- Investitionsrechnungen
- Abweichungsanalysen
- Management-Reporting

Excel ist flexibel, intuitiv und in nahezu jedem Unternehmen verfügbar. Kaum ein Reporting-Prozess kommt ohne Excel aus. Doch die Rahmenbedingungen haben sich grundlegend verändert:

- Datenvolumina steigen exponentiell.
- Transaktionsdaten werden granularer.
- Datenquellen werden vielfältiger.
- Entscheidungszyklen verkürzen sich.
- Prognoseanforderungen werden komplexer.

Während früher monatliche Aggregationen ausreichten, erwarten Management und Märkte heute tiefere Analysen, schnellere Aktualisierung und höhere Prognosequalität. Hier stößt die klassische Tabellenlogik zunehmend an strukturelle Grenzen.

Excel – Stärke im Frontend, Grenzen im Backend

Excel ist hervorragend geeignet für:

- Visualisierung
- Szenariorechnungen
- flexible Modellierung
- Management-Präsentationen

Doch Excel arbeitet zellenbasiert. Das bedeutet:

- Jede Berechnung steht in einer einzelnen Zelle.
- Formeln werden kopiert.
- Die Logik verteilt sich über mehrere Arbeitsblätter.
- Prozesse sind oft nur implizit dokumentiert.

Bei kleinen Datenmengen ist das effizient. Bei Hunderttausenden oder Millionen Datensätzen wird es problematisch. Denn:

- Die Performance leidet.
- Es entstehen mehr Fehlerquellen.
- Prozesse sind schwer reproduzierbar.
- Die Versionierung wird unübersichtlich.

Excel bleibt damit ein starkes Analyse-Frontend – aber keine optimale Datenverarbeitungs-Engine für große und komplexe Datenstrukturen.

Python – Datenverarbeitung als strukturierter Prozess

Python hat sich weltweit zur führenden Sprache für Datenanalyse entwickelt. Der Grund liegt nicht in technischer Komplexität, sondern in struktureller Klarheit. Anders als Excel denkt Python nicht in Zellen, sondern in Datenstrukturen.

Das zentrale Element ist der sogenannte **DataFrame** (zum Beispiel über die Bibliothek **Pandas**). Ein DataFrame ist eine Tabelle – allerdings mit programmierbaren Methoden.

Der entscheidende Unterschied ist: In Excel wird Logik häufig visuell modelliert. In Python wird Logik explizit definiert. Das sorgt für bessere:

- Transparenz
- Reproduzierbarkeit
- Skalierbarkeit
- Dokumentierbarkeit

Gerade in prüfungsrelevanten oder regulierten Umgebungen ist das ein erheblicher Vorteil.

Konkretes Beispiel: Berechnung eines Deckungsbeitrags

Im klassischen Excel-Modell würde man in einer Zelle wie C1 eine Formel hinterlegen, um den Deckungsbeitrag zu berechnen. Zum Beispiel in der Form `=A1-B1`. Diese Formel würde man dann für die gesamte Spalte kopieren und am Ende eine Summe über die Spalte C bilden.

In Python sieht die Berechnung so aus:

```
df["Deckungsbeitrag"] = df["Umsatz"] - df["variable_Kosten"]
```

Was passiert hier genau?

- **df** ist ein DataFrame – also eine komplette Tabelle.
- **df["Umsatz"]** greift auf die gesamte Umsatzspalte zu.
- **df["variable_Kosten"]** greift auf die gesamte Spalte mit den variablen Kosten zu.
- Die **Subtraktion** wird automatisch zeilenweise für alle Datensätze durchgeführt.
- Das **Ergebnis** wird als neue Spalte „Deckungsbeitrag“ gespeichert.

Das bedeutet: Egal ob 10.000 oder 1.000.000 Zeilen – die Berechnung erfolgt vollständig, konsistent und ohne manuelle Kopiervorgänge. Betriebswirtschaftlich heißt das:

- keine vergessenen Formeln
- keine inkonsistenten Bereiche
- keine manuelle Nacharbeit

Die Logik ist klar dokumentiert und jederzeit reproduzierbar.

Praxisfall: Monatsreporting mit 1,2 Millionen Datensätzen

Angenommen, ein Unternehmen exportiert monatlich Verkaufsdaten aus dem ERP-System. In einer CSV-Datei sind demnach enthalten:

- 1,2 Millionen Positionen
- 300 Produkte
- 10 Regionen
- variable Kosten je Position

Ziel des Reportings sind Berechnungen zum:

- Umsatz je Produktgruppe
- Deckungsbeitrag je Region
- Vergleich zum Vormonat
- Top-10-Kunden

In Python könnte der Prozess für die Umsatz- und Deckungsbeitragsrechnung folgendermaßen aussehen:

```
import pandas as pd
df = pd.read_csv("umsatz.csv", headers=True)
report = df.groupby("Region", as_index=False).agg({
    "Umsatz": "sum",
    "variable_Kosten": "sum"
})
report["Deckungsbeitrag"] = report["Umsatz"] - report["variable_Kosten"]
report
```

Erklärung der einzelnen Schritte

1. Einlesen der Daten

```
df = pd.read_csv("umsatz.csv")
```

Die Datei wird vollständig eingelesen und als DataFrame gespeichert. Alle Datensätze stehen nun strukturiert zur Verfügung.

2. Gruppierung der Daten

```
df.groupby(["Produktgruppe", "Region"])
```

Hier werden die Daten nach Produktgruppe und Region gruppiert. Das entspricht einer Pivot-Tabelle mit zwei Dimensionen.

3. Aggregation

```
.agg({  
    "Umsatz": "sum",  
    "variable_Kosten": "sum"  
})
```

Die Umsätze und Kosten werden je Gruppe summiert. Das Ergebnis ist eine verdichtete Tabelle – bereit für die Berichterstattung.

4. Berechnung der Kennzahl

```
report["Deckungsbeitrag"] = report["Umsatz"] - report["variable_Kosten"]  
report
```

Der Deckungsbeitrag wird auf aggregierter Ebene berechnet. Damit entsteht ein vollständiger Reporting-Datensatz.

Hinweis: Je nachdem, wie die Daten in der CSV-Datei vorliegen oder wie die Ausgabe des Ergebnisses dargestellt werden soll, ergänzen Sie den Python-Code um weitere Parameter wie `headers=True` oder `as_index=False`.

Der strategische Mehrwert

Diese Vorgehensweise verändert das Controlling in mehrfacher Hinsicht:

- **Reproduzierbarkeit:** Der Prozess ist vollständig dokumentiert. Jeder Monat wird identisch berechnet.
- **Skalierbarkeit:** Die Logik funktioniert unabhängig von der Datenmenge.
- **Qualitätssicherheit:** Manuelle Eingriffe werden minimiert. Fehlerquellen sinken deutlich.
- **Automatisierung:** Der Code kann in automatisierte Workflows integriert werden. Reports können ohne manuelle Zwischenschritte erstellt werden.

Excel bleibt – aber in neuer Rolle

Python ersetzt Excel nicht. Excel bleibt:

- Planungsinstrument
- Visualisierungsplattform
- Szenario-Simulator
- Kommunikationsmedium für das Management

Python übernimmt:

- Datenaufbereitung
- Aggregation großer Datenmengen
- komplexe Berechnungen
- statistische Analysen

Die Kombination aus Excel und Python führt zu einer klaren Arbeitsteilung:

- Excel = Frontend
- Python = Analyse-Engine

Auswirkungen auf das Rollenbild des Controllers

Der klassische Controller war vor allem Zahlenaufbereiter, Report-Ersteller und dazu Excel-Spezialist. Der moderne Controller wird zunehmend:

- Datenarchitekt
- Prozessgestalter

- Automatisierungsexperte
- Schnittstelle zwischen Fachbereich und IT

Es geht nicht darum, Data Scientist zu werden. Es geht darum, Datenprozesse zu verstehen und strukturierter zu denken.

Fazit

Excel bleibt das Herzstück der Unternehmenssteuerung. Doch Python ergänzt Excel um Skalierbarkeit, Automatisierung und strukturelle Klarheit. Die Verbindung beider Technologien ist kein kurzfristiger Trend, sondern eine logische Weiterentwicklung datengetriebener Unternehmensführung.

Unternehmen, die diese Kombination strategisch einsetzen, gewinnen:

- Geschwindigkeit
- Transparenz
- Prognosefähigkeit
- Wettbewerbsfähigkeit

Das Controlling der Zukunft ist nicht weniger Excel – sondern Excel plus Python.

Von VBA zu Python als weiterer Schritt im Controlling

Was VBA und Python bei Excel-Anwendungen unterscheidet und welche Anwendungsmöglichkeiten sie jeweils haben. Mit Vorteilen und Nachteilen von VBA sowie durch eine Ergänzung um Python.

Zuletzt geändert am 02.03.2026



Abwägung zwischen VBA und Python für Excel-Anwendungen

In vielen Unternehmen ist Excel nicht nur ein Werkzeug – es ist ein **Betriebssystem** für Planung, Reporting und Ad-hoc-Analysen. Und wenn Prozesse in Excel „zu langsam“ oder „zu manuell“ sind, heißt die Antwort oft: „Dann bauen wir eben ein Makro.“

Entsprechend hat **VBA (Visual Basic for Applications)** unzählige Controlling-Prozesse automatisiert:

- Imports aus CSV/ERP-Exports
- Aktualisierung von Pivot-Tabellen
- Erstellung von Reports, PDFs, E-Mail-Versand
- Datenbereinigung
- Standardisierter Monatsabschluss

Aber: Die Rahmenbedingungen haben sich verändert. Datenbestände sind größer, Systeme vernetzter, Sicherheitsanforderungen strenger – und Microsoft selbst verschiebt den Schwerpunkt in Richtung Cloud- und Plattform-Ökosystem. In diesem Umfeld gewinnt **Python** an Bedeutung.

Die zentrale Frage lautet daher nicht „VBA oder Python“, sondern: Welche Rolle spielt VBA künftig – und wo ist Python strategisch überlegen?

Was VBA im Controlling so stark gemacht hat

VBA wurde im Controlling groß, weil es drei Anforderungen erfüllte:

Nähe zu Excel (der größte Vorteil)

VBA ist direkt in Excel eingebettet:

- Zugriff auf Arbeitsmappen, Blätter, Zellen, Pivot-Objekte
- Benutzeroberflächen (UserForms)
- Ereignisse (Workbook_Open, Worksheet_Change)
- sehr gut geeignet für „Excel als Anwendung“

Wenn Ihr Prozess vor allem bedeutet, **Excel zu manipulieren**, ist VBA immer noch sehr effizient.

Automatisierung ohne IT-Ticket

Viele Fachbereiche konnten mit VBA Lösungen bauen, ohne:

- Serverzugriff
- Datenbankfreigaben
- Entwicklungstools
- Deploymentprozesse

Das war ein enormer Produktivitätshebel – hat aber auch Schattenseiten.

Schnelle Wirkung bei wiederkehrenden Routinen

Für das monatliche Reporting sind jedes Mal die gleichen Schritte in Excel durchzuführen:

- Daten importieren
- sortieren, filtern, berechnen
- Pivot refresh
- Charts exportieren

Das passiert automatisch, wenn ein entsprechendes VBA-Makro erstellt ist.

Warum VBA an Grenzen stößt

VBA stößt in typischen Situationen an strukturelle Grenzen.

Architektur: lokal, dateibasiert, schwer skalierbar

VBA läuft in der Regel lokal auf dem Client, innerhalb einer Excel-Datei und abhängig von Excel-Version, Add-ins und lokalen Einstellungen. Das führt zu Problemen wie:

- der Ruf des Anwenders: „Bei mir läuft es, bei dir nicht.“
- Abhängigkeiten von Pfaden, Netzlaufwerken, Zugriffsrechten
- Performanceprobleme bei großen Datenmengen
- keine echte Skalierung

Governance: schwer zu kontrollieren

Viele Organisationen haben „Makro-Wildwuchs“:

- Makros ohne Dokumentation
- Abhängigkeit von Einzelpersonen
- keine Versionsverwaltung
- keine standardisierten Tests
- kein zentraler Betrieb

Für Management und Revision ist das oft ein Risiko:

- nicht nachvollziehbare Berechnungslogik
- schwer prüfbare Ergebnisse
- hohes Risiko, von einzelnen Personen abhängig zu sein

Sicherheit: Makros sind ein bekanntes Einfallstor

VBA-Makros sind seit Jahren ein Sicherheitsthema. Viele Unternehmen:

- blockieren Makros standardmäßig
- erlauben sie nur signiert
- reduzieren Ausführungsrechte

Das ist aus Sicherheits- und Compliance-Sicht nachvollziehbar – senkt aber die Alltagstauglichkeit von VBA.

Microsoft hat zwar weiterhin den Support für VBA nicht eingestellt, aber durch die Standard-Blockierung von Makros aus dem Internet (seit 2022) haben sich die Hürden für Makros massiv erhöht.

Des Weiteren führt das **Mark of the Web (MotW)** dazu, dass digital nicht signierte Makros oft gar nicht mehr starten – ein administrativer Albtraum für unvorbereitete Abteilungen.

Datenanalyse: VBA ist nicht für moderne Analyseverfahren gebaut

VBA kann Daten verarbeiten, aber moderne Analyseverfahren wie Zeitreihen, Machine Learning oder statistische Modelle führen aus diesen Gründen zu hohem Aufwand:

- kaum Standardbibliotheken
- wenig Komfort für DataFrames oder Tabellenoperationen
- begrenzte Visualisierungsmöglichkeiten
- wenig Anschlussfähigkeit an moderne Datenplattformen

Warum Python in Unternehmen so stark wächst

Python ist im Business-Kontext deshalb beliebt, weil es die Lücken schließt, die VBA offenlässt.

Python ist ein Ökosystem, nicht nur eine Sprache

Python punktet durch Bibliotheken, die für Business Analytics Standard sind:

- **Pandas:** Tabellen und Datenaufbereitung
- **NumPy:** numerische Berechnungen
- **Matplotlib** oder **Plotly:** Visualisierung
- **Scikit-learn:** Machine Learning
- **Statsmodels:** Statistik, Zeitreihen
- **Openpyxl** oder **XlsxWriter:** Excel-Dateien schreiben, ohne dass Excel installiert sein muss

Damit wird Python zur Werkzeugkiste für:

- saubere Datenpipelines
- reproduzierbare Analysen
- komplexe Kennzahlenlogik
- Simulationen und Forecasts

Datenlogik statt Zelllogik

Excel denkt in Zellen. Python denkt in Tabellenobjekten.

Beispiel: Der Python-Befehl, um den Deckungsbeitrag zeilenweise über eine ganze Tabelle zu berechnen, lautet:

```
df["Deckungsbeitrag"] = df["Umsatz"] - df["variable_Kosten"]
```

Was bewirkt das?

- Es wird für jede Zeile automatisch Umsatz minus variable Kosten gerechnet.
- Kein Formelkopieren, kein „Zeile vergessen“, keine Inkonsistenzen.
- Die Logik ist als Code klar dokumentiert und wiederholbar.

Skalierbarkeit und Anschlussfähigkeit

Python ist anschlussfähig an:

- Datenbanken (SQL)
- APIs (Webservices)
- Cloud-Plattformen
- Data Warehouses

Das heißt: Python kann dort arbeiten, wo die Daten entstehen – nicht erst, wenn sie als Export in Excel liegen.

Ist das „Ablösung“ oder „Evolution“?

In der Praxis ist es fast immer Evolution – mit einer klaren Rollenverteilung. **Excel bleibt stark als:**

- Frontend für Fachbereiche
- Visualisierung und Kommunikation
- Planungs- und Szenario-Modelle
- „letzte Meile“ der Entscheidungsvorbereitung

Python wird stark als:

- Datenaufbereitung
- Massendatenanalyse
- Automatisierung und Wiederholbarkeit
- Statistik, Forecasting, Simulation
- komplexe Logik und Datenqualität

Microsoft hat Python direkt in Excel integriert (derzeit im Rollout). Das ist ein Argument für die „Evolution“, da die Welten verschmelzen.

Governance: Der Gamechanger – und oft das Management-Argument

Wenn Unternehmen über VBA versus Python sprechen, geht es häufig vordergründig um Technik. Aber tatsächlich geht es um die Steuerbarkeit. Typische Governance-Probleme bei VBA sind:

- Makro hängt an einer Datei
- unklare Zuständigkeiten
- Änderungen ohne Versionierung
- fehlende Tests
- schwer nachvollziehbare Logik

Die Governance-Vorteile durch Python (bei sauberer Umsetzung) sind:

- Code ist versionierbar (Git)
- Änderungen nachvollziehbar
- Tests möglich
- Deployment und Standards besser umsetzbar
- Rollenmodell (Fachbereich/IT) sauber definierbar

Gerade im Controlling ist das wichtig, weil Zahlen nicht nur „stimmen“, sondern **nachvollziehbar zustande kommen müssen**. VBA-Lösungen sind oft das Paradebeispiel für **Shadow IT (Schatten-IT)**. Python ermöglicht durch Code-Reviews den Weg zurück in eine kontrollierte IT-Struktur.

Was bedeutet das für Fachbereiche?

Sie müssen VBA nicht abschaffen. Viele VBA-Lösungen sind sinnvoll, stabil und haben sich bezahlt gemacht. Abschaffen wäre meist teuer und unnötig.

Sie sollten aber gezielt modernisieren. Ob und wo das notwendig und hilfreich sein kann, klären Sie mit diesen Fragen zur Einordnung:

- Verarbeitet der Prozess **große Datenmengen**?
- Gibt es **viele manuelle Schritte**?
- Ist die Logik **prüfungsrelevant**?
- Gibt es **Risiken** wegen Abhängigkeit von einer Person?
- Soll der Prozess **skalieren** oder in Workflows laufen?

Wenn Sie mehrere Male „Ja“ sagen, ist das ein klares Signal: Python ist eine strategische Option.

Tipp

Woran erkenne ich, dass mein Prozess bereit für Python ist?

- Die Excel-Datei braucht länger als 30 Sekunden zum Öffnen oder Berechnen.
- Mehr als drei verschiedene Datenquellen werden manuell zusammengeführt.
- Der Prozess wird von verschiedenen Personen an verschiedenen Standorten genutzt.
- Statistische Prognosen (Forecasting) werden benötigt.

Upskilling: Welche Kompetenz ist realistisch?

Niemand muss sofort Data Scientist werden. Ein realistisches Zielbild ist:

- Grundverständnis von DataFrames (Tabellenlogik) schaffen
- Basisoperationen durchführen: filtern, gruppieren, aggregieren
- einfache KPI-Berechnungen erstellen
- Verständnis für Datenqualität und Reproduzierbarkeit entwickeln

Fazit

VBA bleibt – aber Python wird zum strategischen Standard. VBA ist sinnvoll, wenn es darum geht:

- Excel zu steuern,
- Benutzerinteraktionen in Excel zu automatisieren oder
- bestehende, stabile Makroprozesse weiter zu nutzen.

Python wird zur strategischen Ergänzung, wenn es darum geht:

- große Datenmengen effizient zu verarbeiten,
- Reportingprozesse reproduzierbar zu machen,
- moderne Analytics (Forecast, Statistik, Simulation) zu nutzen oder
- Governance und Transparenz zu erhöhen.

Die Zukunft ist nicht „VBA oder Python“, sondern der Einsatz von Excel als Business-Frontend, von Python als Daten- und Analyse-Engine, und dort, wo Excel-Automation nötig ist, bleibt VBA (oder moderne Alternativen) ein Werkzeug im Portfolio.

Architektur, Chancen und Grenzen von Python im Unternehmenseinsatz

Ein Beispiel zeigt, welche Möglichkeiten Python in Excel für die Datenanalyse bietet. Sie erkennen die notwendigen Voraussetzungen und entscheidende Vorteile der cloudbasierten Datenanalyse.

Zuletzt geändert am 02.03.2026



Warum Python in Excel ein strategischer Meilenstein ist

Mit der Integration von Python in Microsoft 365 Excel hat Microsoft einen technologischen Paradigmenwechsel eingeleitet. Erstmals wird eine vollwertige Data-Analytics-Sprache direkt in Excel eingebettet.

Das ist mehr als ein Feature-Update. Es ist ein architektonischer Schritt:

- Excel wird zur Analyseplattform.
- Python wird zur integrierten Rechen-Engine.
- Die Cloud-Infrastruktur übernimmt die Ausführung.

Für das Controlling bedeutet das: Sie können umfangreiche Datenanalysen durchführen – ohne Excel zu verlassen.

Was genau ist „Python in Excel“?

„Python in Excel“ erlaubt es, Python-Code direkt in einer Excel-Zelle auszuführen. Das Ergebnis wird als Tabelle oder Objekt zurück in das Arbeitsblatt geschrieben.

Wichtig dabei:

- Der Code wird nicht lokal ausgeführt.
- Die Berechnung erfolgt in der Microsoft-Cloud.
- Excel dient als Interface.

Damit unterscheidet sich Python in Excel grundlegend von VBA.

VBA	Python in Excel
lokal	cloudbasiert
dateigebunden	konto-/mandantenbasiert
Excel-Objektmodell	DataFrame-Logik
stark UI-orientiert	stark datenorientiert

Hintergrund: Sicherheit durch Partnerschaften

Microsoft liefert Python nicht isoliert aus. Die Integration erfolgt in enger Zusammenarbeit mit **Anaconda**, dem weltweit führenden Anbieter für Data-Science-Distributionen.

Für Unternehmen bedeutet das: Sie greifen auf eine kuratierte und sicherheitsgeprüfte Bibliothek zu. Sie müssen sich nicht um die Installation von Paketen oder Sicherheitsupdates kümmern – die wichtigsten Tools für das Controlling (wie Pandas, Numpy oder Matplotlib) sind „out of the box“ vorhanden, virengeprüft und einsatzbereit.

Die technische Architektur – Was passiert im Hintergrund?

Wenn Sie Python-Code in eine Excel-Zelle eingeben, läuft folgender Prozess ab:

1. Excel sendet den Python-Code an die Cloud-Umgebung.
2. Dort wird der Code in einer isolierten Python-Umgebung ausgeführt.
3. Das Ergebnis wird zurück an Excel übertragen.
4. Excel rendert das Ergebnis als strukturierte Tabelle.

Das bedeutet:

- keine lokale Python-Installation erforderlich
- keine manuelle Paketverwaltung
- standardisierte Ausführungsumgebung

Für Unternehmen ist entscheidend:

- Daten werden in der Cloud verarbeitet
- Sicherheitsrichtlinien greifen
- Tenant- und Lizenzabhängigkeit besteht

Gerade in regulierten Branchen sollte dabei geprüft werden:

- Welche Daten dürfen verarbeitet werden?
- Gibt es Compliance-Vorgaben?
- Ist der Cloud-Transfer zulässig?

Das ist kein Hinderungsgrund – muss aber im Vorfeld geklärt werden.

Wichtig: Höchste Sicherheit durch Isolation

Ein kritischer Punkt in der IT-Governance ist oft die Ausführung von Code. Hier setzt Microsoft auf eine „**Zero-Trust**“-Architektur:

Der Python-Code läuft nicht direkt auf Ihrem Rechner und hat keinen Zugriff auf Ihr lokales Netzwerk oder Ihre privaten Dateien. Stattdessen wird er in einem isolierten **Hyper-V-Container** in der Azure-Cloud ausgeführt.

Der Container ist flüchtig – nach der Berechnung wird er gelöscht. Das bedeutet: Python in Excel ist sicherer als viele herkömmliche Add-ins, da es eine strikte Trennung zwischen der lokalen Arbeitsumgebung und der Rechen-Engine garantiert.

Ein einfaches Praxisbeispiel – Daten aus Excel analysieren

Angenommen, Sie haben folgende Tabelle in Excel:

	A	B
1	Region	Umsatz
2	Nord	120.000 €
3	Süd	150.000 €
4	Nord	90.000 €
5	West	110.000 €

Daten für die Verarbeitung mit Python

In einer Python-Zelle können Sie folgenden Code verwenden:

```
import pandas as pd
df = xl("A1:B5", headers=True)
df.groupby("Region")["Umsatz"].sum()
```

Erklärung Schritt für Schritt

```
xl("A1:B5", headers=True)
```

Diese Funktion importiert den angegebenen Zellbereich aus Excel in einen DataFrame und teilt Python mit, dass in der Tabelle in der ersten Zeile die Spaltenüberschriften stehen. Das ist die Brücke zwischen Excel und Python.

```
groupby("Region")
```

Die Daten werden nach der Spalte „Region“ gruppiert. Das entspricht funktional einer Pivot-Tabelle.

```
["Umsatz"].sum()
```

Für jede Region wird der Umsatz summiert.

Ergebnis: Python gibt eine verdichtete Tabelle zurück. Das Ergebnis erscheint direkt im Arbeitsblatt.

D1 × ✓ 123 PY import pandas as pd df = xl("A1:B5", headers=True) df.groupby("Region")["Umsatz"].sum()					
	A	B	C	D	E
1	Region	Umsatz		Region	Umsatz
2	Nord	120.000 €		Nord	210000
3	Süd	150.000 €		Süd	150000
4	Nord	90.000 €		West	110000
5	West	110.000 €			

Beispiel für einen einfachen Python-Code in Excel

Was ist mit Python in Excel konzeptionell neu?

Für die Datenanalyse würden Sie in Excel eine Pivot-Tabelle einfügen, Feldzuweisungen vornehmen und das Layout anpassen. In Python definieren Sie die Logik explizit als Code.

Das bringt mehrere Vorteile:

- Reproduzierbarkeit
- Dokumentation der Logik
- leichte Erweiterbarkeit
- Kombinierbarkeit mit komplexeren Berechnungen

Erweiterung: Berechnung einer zusätzlichen Kennzahl

Angenommen, Sie haben zusätzlich variable Kosten in Spalte C. Nun fügen Sie in Zelle E1 folgenden Python-Code ein:

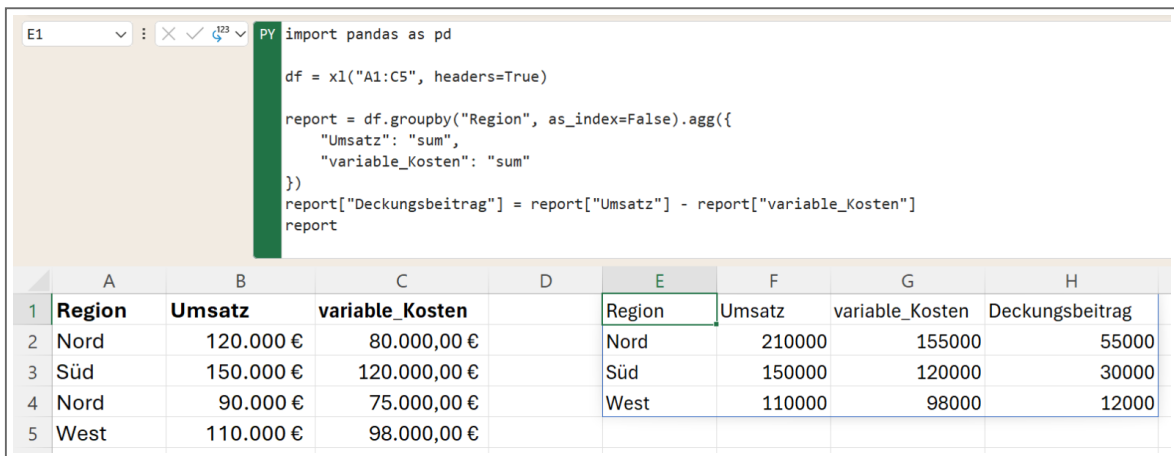
```
import pandas as pd
df = xl("A1:C5", headers=True)
```

```
report = df.groupby("Region", as_index=False).agg({
    "Umsatz": "sum",
    "variable_Kosten": "sum"
})
report["Deckungsbeitrag"] = report["Umsatz"] - report["variable_Kosten"]
report
```

Erklärung

- Die Daten werden nach Region gruppiert.
- Umsatz und Kosten werden aggregiert.
- Der Deckungsbeitrag wird berechnet.
- Das Ergebnis ist die Summe der Deckungsbeiträge je Region als Management-Kennzahl.

Das entspricht einem strukturierten Reporting-Prozess – aber in wenigen Zeilen Code.



The screenshot shows the Python in Excel environment. The code editor on the left contains the following Python code:

```
import pandas as pd

df = xl("A1:C5", headers=True)

report = df.groupby("Region", as_index=False).agg({
    "Umsatz": "sum",
    "variable_Kosten": "sum"
})

report["Deckungsbeitrag"] = report["Umsatz"] - report["variable_Kosten"]
report
```

Below the code editor, a table is displayed with the following data:

	A	B	C	D	E	F	G	H
1	Region	Umsatz	variable_Kosten		Region	Umsatz	variable_Kosten	Deckungsbeitrag
2	Nord	120.000 €	80.000,00 €		Nord	210000	155000	55000
3	Süd	150.000 €	120.000,00 €		Süd	150000	120000	30000
4	Nord	90.000 €	75.000,00 €		West	110000	98000	12000
5	West	110.000 €	98.000,00 €					

Beispiel für eine Berechnung mit Python in Excel

Chancen für Fachbereiche

Python in Excel bietet:

- **Niedrige Einstiegshürde:** Kein Systemwechsel, kein Toolwechsel.
- **Leistungsfähigere Analysen:** Zugriff auf Statistik, Zeitreihen, Simulationen.
- **Reproduzierbare Prozesse:** Logik ist explizit dokumentiert.
- **Brücke zur IT:** Python ist IT-kompatibel und cloudfähig.

Grenzen und realistische Einordnung

Trotz aller Möglichkeiten gibt es Grenzen:

- Cloud-Abhängigkeit
- Lizenzabhängigkeit (Microsoft 365)
- noch nicht in allen Umgebungen verfügbar
- kein vollständiger Ersatz für professionelle Data-Plattformen
- Performance abhängig von Umgebung

Die Cloud-Frage: Datenschutz und Performance

Da die Berechnungen in der Cloud stattfinden, sind zwei Aspekte zu beachten:

- **Data Residency:** Unternehmen sollten prüfen, ob ihre Microsoft-365-Konfiguration den Transfer von (anonymisierten) Daten zur Verarbeitung in die Cloud-Umgebung erlaubt. In der Regel ist dies über die zentralen Tenant-Einstellungen steuerbar.
- **Hardware-Unabhängigkeit:** Ein großer Vorteil der Cloud-Architektur ist die Skalierbarkeit. Da die Rechenlast nicht auf Ihrem lokalen Prozessor liegt, können auch komplexe Simulationen auf leistungsschwächeren Laptops ohne Performance-Einbußen durchgeführt werden.

Strategische Einordnung für Unternehmen

Python in Excel ist eine Technologie zwischen Fachbereich und Data-Engineering. Unternehmen können damit:

- Reporting-Prozesse modernisieren
- Controlling automatisieren
- Analysefähigkeiten erweitern
- Datenkompetenz im Fachbereich stärken

Mit Python in Excel wandelt sich die Rolle des Controllings. Es geht nicht um eine bloße Automatisierung von Makros, sondern um die Fähigkeit, **reproduzierbare Analysen** zu erstellen.

Während Excel-Formeln oft schwer zu prüfen sind, ist Python-Code wie ein Protokoll: Jeder Schritt ist logisch nachvollziehbar, testbar und erfüllt damit die Anforderungen an die **Compliance und Revisionssicherheit**.

Fazit

Mit Python in Excel verschmelzen zwei Welten: die intuitive Oberfläche von Excel mit der strukturierten Datenlogik von Python. Für das Controlling eröffnet das neue Möglichkeiten:

- skalierbare Analysen
- höhere Datenqualität
- reproduzierbare Prozesse
- erweiterte Prognosefähigkeit

Die Frage lautet nicht mehr, ob Python ins Controlling kommt. Die Frage lautet, wie strategisch Sie es einsetzen.

Schritt-für-Schritt-Anleitung für die Aktivierung von Python in Excel

In dieser Schritt-für-Schritt-Anleitung erfahren Sie, wie Sie die Python-Funktion in Excel freischalten und was technisch im Hintergrund passiert.

Zuletzt geändert am 02.03.2026



Warum Python in Excel nicht automatisch verfügbar ist

Microsoft rollt Python in Excel als Teil eines **stufenweisen Rollouts** aus. Das bedeutet, dass die Funktion zuerst im Insider-Programm getestet wird, bevor sie in den Standard-Channels (Monthly Enterprise oder Semi-Annual) landet.

Neue Funktionen dieser Art werden bei Microsoft in mehreren Phasen ausgerollt:

1. interne Tests
2. Beta-Phase im Microsoft 365 Insider-Programm
3. schrittweise Freigabe für breitere Nutzergruppen
4. Standardintegration in reguläre Versionen

Diese vorsichtige Strategie stellt sicher, dass die Cloud-Infrastruktur der Last standhält und Sicherheitslücken vor dem breiten Release geschlossen werden.

Wenn Sie Python also bislang nicht in Ihrer Excel-Umgebung sehen, liegt das nicht an einem Fehler – sondern am Rollout-Modell.

Voraussetzungen für die Anwendung der Python-Funktion

Stellen Sie sicher, dass Ihr System folgende Voraussetzungen erfüllt:

- **Abonnement:** Microsoft 365 (Single, Family oder Business)
- **Version:** Excel für Windows (Mac und Web folgen schrittweise)
- **Insider-Status:** Sie müssen dem **Beta-Kanal** (Beta Channel) beitreten
- **Internetverbindung:** Zwingend erforderlich, da der Code nicht lokal, sondern in der Azure-Cloud berechnet wird

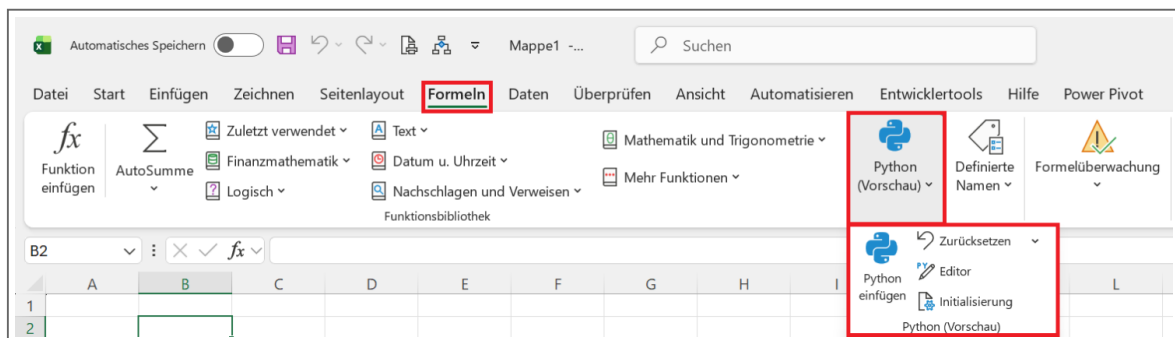
Wichtig: Python in Excel ist keine klassische Add-in-Installation, sondern ein Feature-Update innerhalb von Microsoft 365.

Wo erscheint Python in Excel?

Sind die Voraussetzungen erfüllt, finden Sie Python in der aktuellen Excel-Version:

- in der Registerkarte **Formeln**
- als neue Befehlsgruppe **Python**
- mit dem Hinweis (**Vorschau**), solange es sich um eine Preview-Version handelt

Ist Python bereits integriert, sehen Sie die Befehlsgruppe **Python einfügen**.

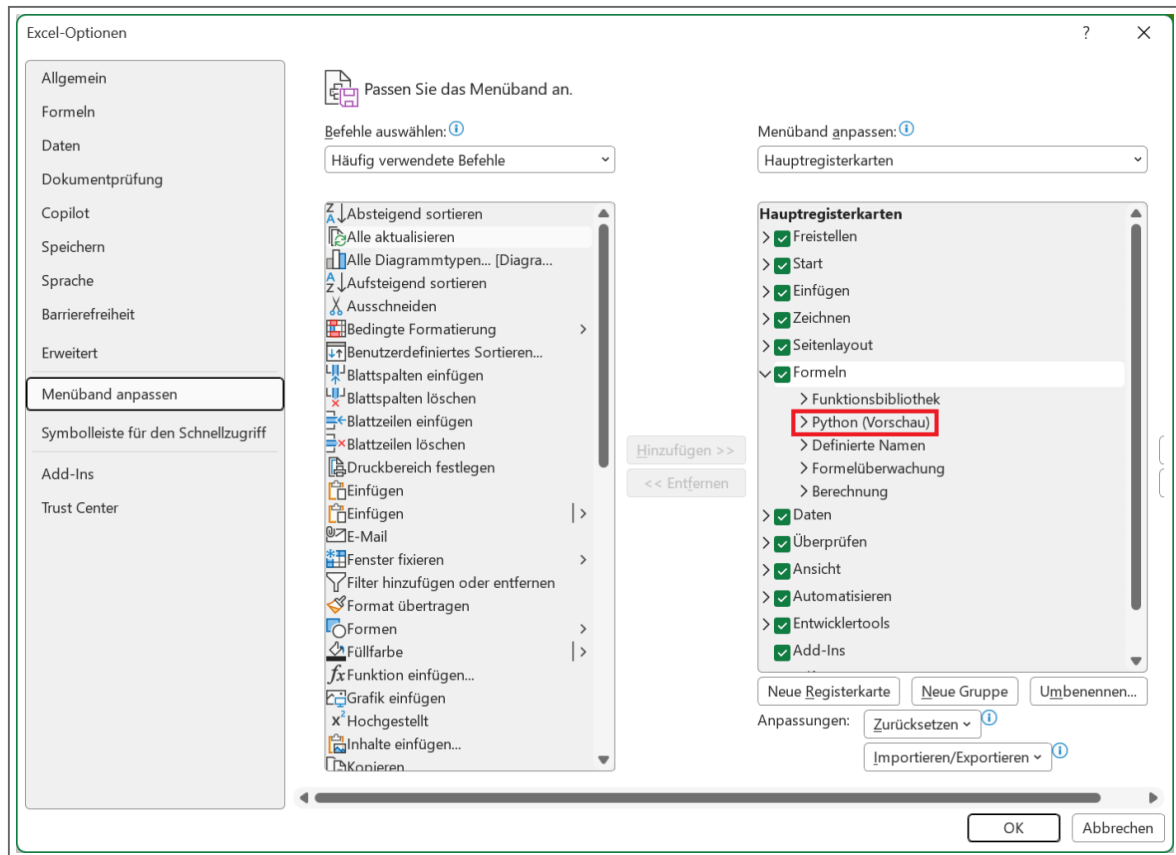


Die Python-Funktion im Excel-Menü

Alternativ kann über das Menüband geprüft werden, ob Python in Excel vorhanden ist. Aktivieren Sie die Excel-Optionen, indem Sie im Menüband auf die Befehlsfolge Registerkarte **Datei** > Befehl **Optionen** klicken.

Es öffnet sich das Dialogfeld **Excel-Optionen**. Klicken Sie hier am linken Rand auf den Eintrag **Menüband anpassen**. Im rechten Bereich können Sie jetzt unter der

Registerkarte **Formel** erkennen, ob Python vorhanden ist.



Python im Excel-Menüband einstellen

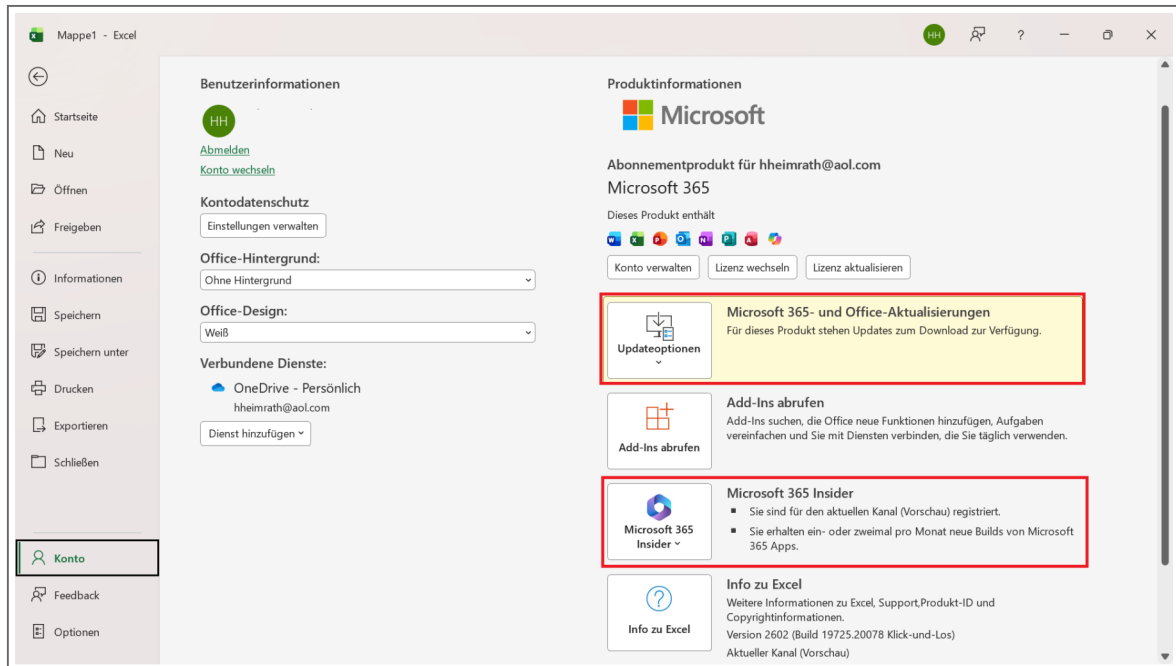
Aktivierung über das Microsoft 365 Insider-Programm

Da Python zunächst im Betakanal veröffentlicht wurde, müssen Sie sich für das Insider-Programm registrieren.

Öffnen Sie zunächst Excel und aktivieren Sie dann im Menüband die Befehlsfolge Registerkarte **Datei** > Befehl **Konto**.

Hier können Sie sich im rechten Bereich bei Microsoft 365 Insider für das Insider-Programm registrieren. Aktivieren Sie dabei den Beta-Channel. Der Beta-Channel ist wichtig, da neue Funktionen hier zuerst bereitgestellt werden.

Des Weiteren aktivieren Sie oben die **Update-Optionen**, damit Sie regelmäßig mit neu verfügbaren Updatefunktionen versorgt werden. Je nach Verbindung und Systemumgebung kann dieser Vorgang einige Minuten dauern.



Teilnahme am Microsoft-Insider-Programm aktivieren

Nach der Aktivierung des Insider-Programms und der Installation des Updates gehen Sie wie folgt vor:

- Alle Office-Anwendungen schließen
- PC neu starten
- Excel erneut öffnen

Nun sollte im Menüband unter **Formeln** das Symbol **Python (Vorschau)** erscheinen

Was tun, falls Python nicht angezeigt wird?

Falls Python nicht sichtbar sein sollte, dann gehen Sie wie folgt vor:

- Erneut unter der Registerkarte **Datei** > Befehl **Konto** prüfen, ob Sie im Beta-Channel registriert sind.
- Updates nochmals manuell starten.

- Einige Tage warten (Rollout kann gestaffelt erfolgen).
- Falls die zentrale IT in Ihrem Unternehmen die Anwendungen steuert: IT-Abteilung kontaktieren.

Gerade in Organisationen kann es sein, dass die IT den Beta-Channel blockiert oder zentral steuert.

Erste Nutzung – Python-Formel einfügen

Sie können Python über eine spezielle **Formel** in einer Zelle verwenden:

```
=PY()
```

Innerhalb dieser Funktion können Sie Python-Code schreiben. Beispiel:

```
import pandas as pd
```

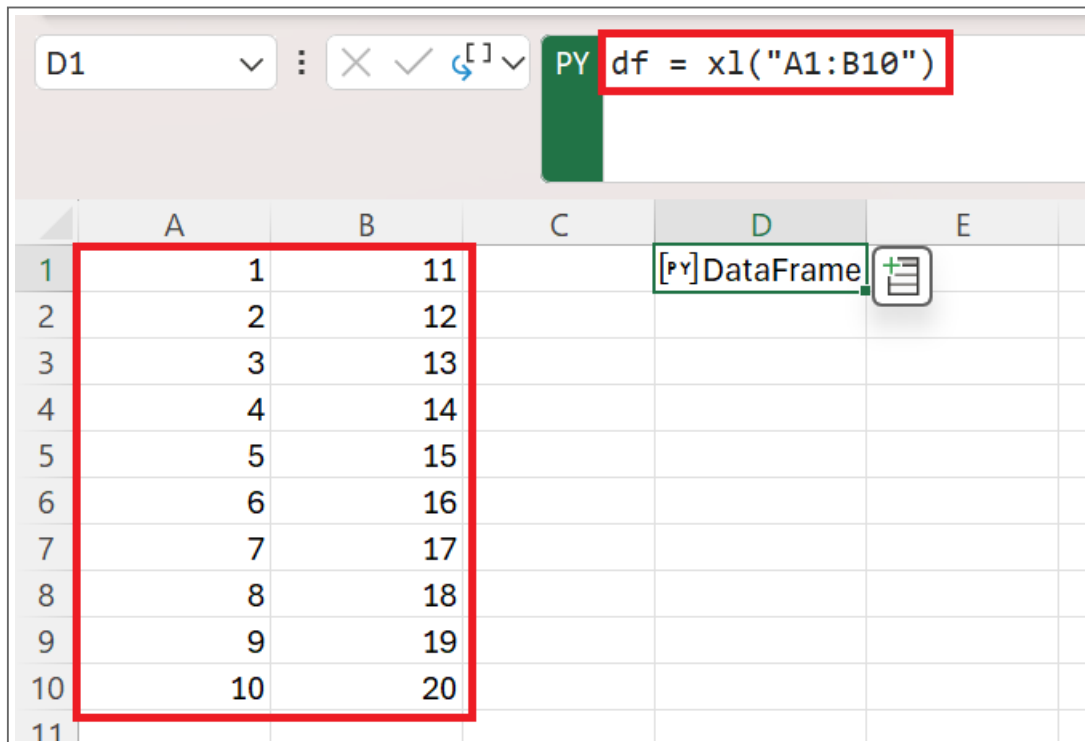
Oder mit Excel-Daten arbeiten:

```
df = xl("A1:B10")
```

Was passiert hier?

- =PY() aktiviert die Python-Ausführungsumgebung.
- df = xl("A1:B10") importiert den angegebenen Zellbereich in einen DataFrame.
- Python verarbeitet die Daten in der Cloud.
- Das Ergebnis wird an Excel zurückgegeben.

Excel fungiert hier als Interface – die Berechnung erfolgt cloudseitig.



Umgebung für die Eingabe von Python-Code in Excel

Technischer Hintergrund: Cloud-Ausführung

Das ist der wichtigste Teil für das Verständnis der Performance und Sicherheit:

Die Anaconda-Distribution

Microsoft nutzt nicht irgendein Python. In Excel ist eine Standard-Distribution von Anaconda integriert. Das bedeutet, sie haben Zugriff auf die wichtigsten Bibliotheken für Data Science, ohne etwas installieren zu müssen:

- **Pandas**: für die Datenmanipulation (DataFrames)
- **Matplotlib** oder **Seaborn**: für professionelle Visualisierungen
- **NumPy**: für komplexe mathematische Berechnungen
- **Statsmodels**: für statistische Modellierung

Cloud-Berechnung versus VBA

Im Gegensatz zu VBA (lokal) wird Python-Code in einer isolierten Hyper-V-Container-Umgebung in der Microsoft-Azure-Cloud ausgeführt.

- Vorteil: Der PC wird bei komplexen Berechnungen nicht ausgebremst.
- Sicherheit: Der Code kann nicht auf lokale Dateien, Makros oder Passwörter zugreifen (Sandboxing).

Feature	VBA	Python in Excel
Ausführung	Lokal (Client)	Cloud (Azure)
Fokus	Automatisierung / User Interface	Datenanalyse / ML
Bibliotheken	Begrenzt (Add-Ins)	Riesiges Ökosystem (Anaconda)
Sicherheit	Höheres Risiko (Makro-Viren)	Hoch (isoliert in der Cloud)

Lizenz- und Governance-Aspekte

Unternehmen sollten folgende Fragen klären:

- Ist der Beta Channel erlaubt?
- Sind Cloudberechnungen zulässig?
- Welche Daten dürfen verarbeitet werden?
- Gibt es Compliance-Vorgaben?

Für sensible Daten (zum Beispiel personenbezogene oder regulatorische Daten) ist eine Abstimmung mit IT und Datenschutz empfehlenswert.

Falls Sie in einem Unternehmen arbeiten, kann es sein, dass die IT den Beta-Kanal deaktiviert hat. Da Python-Daten die lokale Umgebung verlassen (Richtung Azure), müssen Compliance-Beauftragte dies oft erst prüfen.

Tipp: Weisen Sie darauf hin, dass die Datenübertragung verschlüsselt erfolgt und die Session nach der Berechnung sofort gelöscht wird.

Strategische Einordnung

Die Aktivierung von Python in Excel ist technisch einfach – strategisch jedoch bedeutsam. Sie eröffnet Fachbereichen:

- Zugriff auf moderne Datenanalyse
- Automatisierungsmöglichkeiten
- Reproduzierbare Prozesse
- Anschlussfähigkeit an Data-Engineering

Excel bleibt das vertraute Frontend. Python erweitert die Analysekompetenz erheblich.

Fazit

Python in Excel zu aktivieren, ist kein komplizierter Prozess – er erfordert lediglich:

- Microsoft 365
- Registrierung im Insider-Programm
- Installation aktueller Updates

Die eigentliche Bedeutung liegt jedoch nicht in der Aktivierung, sondern in der neuen Architektur:

- Excel wird zur Oberfläche.
- Python wird zur leistungsfähigen Analyse-Engine im Hintergrund.

Für Controller und datengetriebene Fachbereiche ist das ein entscheidender Kompetenzschritt. Python in Excel macht die Tabellenkalkulation zur Data-Science-Plattform. Die Hürde der Aktivierung ist klein, aber der Gewinn an analytischen Möglichkeiten ist sehr groß.

Vom Excel-Anwender zum Datenanalysten mit Python

Was Python in der Excel-Umgebung alles leisten kann und welche Vorteile und Nutzen Sie gewinnen, wenn Sie Excel, Power Query und Python kombinieren. Entdecken Sie die vielen Möglichkeiten der Datenanalyse.

Zuletzt geändert am 28.05.2026



Eine neue Ära für Excel-Anwender

Excel gehört seit Jahrzehnten zu den wichtigsten Werkzeugen im geschäftlichen Bereich. Ob Controlling, Vertrieb, Finanzwesen, Reporting oder Projektmanagement – nahezu überall wird mit Excel gearbeitet.

Gleichzeitig sind die Anforderungen an Datenanalysen in den letzten Jahren gestiegen durch:

- größere Datenmengen
- komplexere Analysen
- schnellere Entscheidungen
- höhere Automatisierung

Genau hier setzt Python in Excel an. Mit Python erhält Excel eine völlig neue analytische Dimension.

Die Funktion ist für Nutzer mit einem geeigneten Microsoft 365-Abonnement verfügbar und läuft vollständig in der Microsoft Cloud, eine lokale Python-Installation ist nicht erforderlich.

Die Aktivierung erfolgt über **Formeln > Python > Python einfügen** oder mit **=PY(...**

Ein Administrator kann die Funktion allerdings deaktivieren. Das sollten Sie prüfen, falls Sie in einer IT-regulierten Umgebung arbeiten und mit Python arbeiten wollen.

Was Python in Excel verändert

Python ersetzt Excel nicht. Python erweitert Excel.

Das ist ein entscheidender Punkt. Excel bleibt weiterhin hervorragend für Dateneingabe, Tabellenarbeit, Ad-hoc-Analysen und die Präsentation von Ergebnissen. Python ergänzt Excel um

- Automatisierung
- Statistik
- umfangreiche Datenanalyse
- Visualisierung
- skalierbare Datenverarbeitung

Die Python-Codezellen laufen dabei in einer sicheren, isolierten Azure-Umgebung und geben ihre Ergebnisse direkt zurück ins Excel-Arbeitsblatt.

Wichtig für die Praxis: Python hat keinen Zugriff auf Ihren Computer, Ihre Geräte oder das Internet und gibt Ergebnisse nur über die **=PY()-Funktion** zurück.

Mit dem normalen Microsoft 365-Abo berechnen Python-Befehle die Ergebnisse automatisch neu, wie normale Excel-Formeln. Erst mit dem kostenpflichtigen Python als Excel-Add-on erhalten Sie Premium Compute für schnellere Berechnungen und besondere Modi.

Das schützt vor unerwarteten Wartezeiten und gibt Ihnen volle Kontrolle bei großen Datenmodellen und Datenbeständen.

Warum Python besonders bei großen Datenmengen glänzt

Viele Excel-Anwender stoßen irgendwann an Grenzen: lange Ladezeiten, komplexe, verschachtelte Formeln und riesige, träge Pivot-Tabellen.

Python arbeitet anders. Vorteile von Pandas sind vektorisierte Berechnungen (über ganze Spalten statt Zelle für Zelle), spaltenorientierte Verarbeitung und effiziente Speicherstrukturen.

Dadurch können selbst große Datenmengen effizient analysiert werden.

Der klassische Excel-Workflow

Viele Excel-Anwender kennen typische Herausforderungen:

1. Dateneingabe
2. Datenüberprüfung
3. Formeln für Berechnungen erstellen
4. Formeln kopieren
5. Filter setzen
6. Pivot-Tabellen erstellen
7. Diagramme aktualisieren

Typische Probleme sind: fehleranfällig, schwer reproduzierbar, zeitaufwendig und schwierig bei großen Datenmengen. Genau hier bringt Python enorme Vorteile.

Der neue Workflow mit Python in Excel

Diese typischen Excel-Aufgaben lassen sich mit Python mit wenigen Code-Zeilen erledigen.

Aufgabe	Python-Werkzeug / Funktion	Ergebnis zurück nach Excel
Daten laden	xl()	DataFrame im Grid
Daten prüfen	info(), dtypes	Ausgabe in Zellen
Filtern	Bedingungen	Gefilterter DataFrame
Gruppieren	groupby()	Aggregierte Ergebnisse
Pivot-Analysen	pivot_table()	Analyseergebnisse im Blatt
Visualisierung	matplotlib, seaborn	Diagramme direkt im Arbeitsblatt
Zeitreihenanalysen	rolling(), pct_change()	Neue Spalten / Auswertungen

Ein konkretes Beispiel: Statt eine Pivot-Tabelle manuell zu bauen und zu aktualisieren, genügt eine Zeile Python-Code:

```
df.groupby('Kategorie')['Umsatz'].sum().reset_index()
```

Das Ergebnis erscheint sofort wieder als Excel-Tabelle. Dadurch entsteht ein moderner Analyse-Workflow, bei dem die Ergebnisse nahtlos wieder in Excel zur Verfügung stehen.

Tipp: Entscheiden Sie pro PY-Zelle, ob Sie als **Python-Objekt** (für Weiterverarbeitung) oder als **Excel-Wert** (für Kollegen ohne Python) ausgeben möchten, per Rechtsklick auf die Zelle.

Warum Pandas so mächtig ist

Das Herzstück vieler Analysen ist die Bibliothek **Pandas**. Pandas arbeitet mit sogenannten **DataFrames**. Ein DataFrame ist vereinfacht gesagt eine hochoptimierte Tabelle im Arbeitsspeicher.

Der Unterschied zu Excel: Excel speichert Formeln, Formatierungen und Zellinformationen. Pandas speichert dagegen primär strukturierte Daten, spaltenorientiert und hochoptimiert für Analysen.

Dadurch entstehen enorme **Vorteile** bei

- Geschwindigkeit,
- Skalierung und
- Analysefunktionen.

Neben Pandas stehen weitere leistungsstarke Bibliotheken wie **NumPy**, **Matplotlib**, **Seaborn** und **Statsmodels** standardmäßig zur Verfügung. Diese fünf werden sogar automatisch importiert.

Viele weitere Pakete aus dem kuratierten Anaconda-Set (rund 400) können bei Bedarf importiert werden.

Typische Praxisanwendungen

Python in Excel eignet sich besonders für:

- **Vertriebsanalysen:** Umsatzanalysen, Top-Produkte, Vergleiche von Absatzregionen
- **Controlling:** Kennzahlenberechnung, Abweichungsanalysen, Forecasting
- **Datenqualität:** fehlende Werte erkennen, Textdaten bereinigen, Ausreißer analysieren
- **Reporting:** automatisierte Analysen, reproduzierbare Auswertungen, fortgeschrittene Visualisierungen

Auf business-wissen.de erfahren Sie, wie moderne Datenanalysen mit Python entstehen. Sie finden Anleitungen und Beispiele zu:

- Grundlagen: [Python aktivieren](#), [DataFrames verstehen](#), [Daten laden](#)
- Explorative Datenanalyse: [describe\(\)](#), [info\(\)](#), [dtypes](#)
- Datenbearbeitung: [Filtern](#), [Sortieren](#), [neue Kennzahlen berechnen](#)
- Business-Analysen: [groupby\(\)](#), [agg\(\)](#), [Rankings](#), [Pivot-Analysen](#)
- Fortgeschrittene Analysen: [Zeitreihen](#), [Korrelationen](#), [Ausreißeranalysen](#)
- Visualisierung: [Diagramme](#), [Heatmaps](#), [Seaborn](#)

Excel, Power Query oder Python?

Eine wichtige Frage lautet: Welches Werkzeug sollte man wann einsetzen?

- **Excel:** für schnelle Ad-hoc-Analysen, kleinere Datenmengen und manuelle Dateneingabe
- **Power Query:** für Datenimporte, wiederkehrende Transformationen und klassische ETL-Prozesse für Daten (Extract, Transform, Load)
- **Python:** für komplexe Analysen, tiefe statistische Auswertungen, fortgeschrittene Automatisierung, professionelle Visualisierungen und große Datenmengen

Die stärkste Lösung ist meist die Kombination aller Werkzeuge, mit Power Query als stabilem Fundament für den Datenimport.

Die Grenzen von Python in Excel

So leistungsfähig Python in Excel auch ist, es gibt klar definierte Grenzen, die man kennen sollte:

- Kein pip: Eigene Pakete können nicht installiert werden, es steht nur das kuratierte Anaconda-Set mit rund 400 Bibliotheken zur Verfügung.
- Kein Internetzugang aus Python: Der Python-Code hat keinen Zugriff auf externe Daten oder APIs, der Datenzugriff erfolgt ausschließlich über xl() und Power Query.
- Keine mobilen Berechnungen: Auf Smartphones und Tablets wird der Code nicht ausgeführt, Ergebnisse sind aber sichtbar.
- Latenz bei großen Modellen: Viele Berechnungen oder große Datenmengen können spürbare Wartezeiten verursachen, Premium Compute hilft hier.
- Compliance: Daten verlassen für die Ausführung das Unternehmen und werden in die Microsoft Cloud übertragen, das muss mit internen Richtlinien vereinbar sein.

Wer diese Grenzen kennt, kann gezielt entscheiden, wann Python in Excel das richtige Werkzeug ist und wann eine lokale Python-Umgebung (zum Beispiel mit **xlwings**) sinnvoller ist.

Ein typischer Analyse-Workflow

In der Praxis könnte ein moderner, kombinierter Workflow so aussehen:

Schritt	Werkzeug	Bemerkung
Dateneingabe & Pflege	Excel	Manuelle oder strukturierte Eingabe
Datenimport & Bereinigung	Power Query	Erforderlicher Weg für externe Daten
Komplexe Analyse & Statistik	Python	Tiefe Analysen und Automatisierung
Visualisierung	Python & Excel	Professionelle Diagramme

Genau diese Kombination wird in Zukunft immer wichtiger.

Python in Excel im Team

Was passiert, wenn Kollegen ohne Python-Berechtigung die Datei öffnen?

- Ergebnisse bleiben sichtbar: Wer Python in Excel nicht aktiviert oder dessen Unternehmen die Funktion gesperrt hat, sieht die zuletzt berechneten Ergebnisse weiterhin als normale Excel-Werte.
- Kein automatisches Neuberechnen: Der Python-Code wird bei diesen Nutzern nicht ausgeführt, die Ergebnisse aktualisieren sich also nicht automatisch.
- Ausgabe als Excel-Wert: Wer seine PY-Zellen auf „als Excel-Wert“ stellt (per Rechtsklick), stellt sicher, dass die Ergebnisse für alle Kollegen als statische Werte vorliegen und weiterverwendet werden können.
- Versionierung und Nachvollziehbarkeit: Der Python-Code wird direkt in der .xlsx-Datei gespeichert. Bei Ablage auf SharePoint oder OneDrive ist die Datei damit automatisch versioniert und der Analyse-Code ist für alle im Team sichtbar und nachvollziehbar.

Tipp für Teams: Legen Sie intern fest, welche Zellen als Python-Objekt und welche als Excel-Wert ausgegeben werden, damit alle Beteiligten mit den Ergebnissen arbeiten können, unabhängig von ihrer Python-Berechtigung.

Die Rolle von KI und Automatisierung

Ein weiterer wichtiger Punkt: Python wird immer stärker mit KI kombiniert. Beispiele hierfür sind:

- automatische Analysen
- präzise Prognosen
- Machine Learning
- intelligente Datenmodelle

Gerade in Kombination mit Tools wie ChatGPT entstehen völlig neue Möglichkeiten.

Zudem unterstützt Microsoft Office Anwender zunehmend durch den **Microsoft Copilot**, der auf Knopfdruck passenden Python-Code generiert und so den Einstieg erleichtert. Copilot kann inzwischen Python in Excel in natürlicher Sprache schreiben, zum Beispiel für Forecasts oder Machine-Learning-Analysen.

Was Excel-Anwender lernen sollten

Wer sich heute weiterentwickeln möchte, sollte folgende Themen verstehen:

- Pandas: Fundament für jede moderne Datenanalyse
- Visualisierung: aussagekräftiges und professionelles Reporting
- Statistik: richtige Interpretation von Datenmustern
- Automatisierung: massive Steigerung der täglichen Effizienz
- KI-Unterstützung: die Zukunft der datengetriebenen Analyse

Bereits grundlegende Python-Kenntnisse bringen enorme Vorteile im Berufsalltag.

So fangen Sie an

Sie sind motiviert, aber wissen nicht genau, wo Sie anfangen sollen? Hier sind vier konkrete erste Schritte:

- Schritt 1 – Python aktivieren: Öffnen Sie eine Excel-Datei mit Microsoft 365 und navigieren Sie zu **Formeln > Python einfügen** oder tippen Sie direkt **=PY(** in eine Zelle. Damit starten Sie Ihre erste Python-Sitzung.
- Schritt 2 – DataFrame laden: Markieren Sie eine Ihrer vorhandenen Excel-Tabellen, rufen Sie sie mit **xl("TabellenName")** in Python auf und geben Sie sie als Python-Objekt aus. So sehen Sie sofort, wie Excel-Daten als DataFrame aussehen.
- Schritt 3 – Erste Analyse ausführen: Nutzen Sie **df.describe()** oder **df.groupby('Spalte')['Wert'].sum()**, um Ihre Daten auf einen Blick zu verstehen. Das Ergebnis erscheint direkt im Arbeitsblatt.
- Schritt 4 – Weitermachen: Schauen Sie sich die **Beiträge hier auf business-wissen.de** an. Jede Stunde Übung bringt Sie einen großen Schritt weiter.

Tipp: Starten Sie mit einer Datei, die Sie ohnehin gut kennen. So können Sie Python-Ergebnisse direkt mit Ihren gewohnten Excel-Analysen vergleichen und sehen sofort den Mehrwert.

Weiterführende Ressourcen

Möchten Sie tiefer einsteigen? Hier finden Sie die wichtigsten Anlaufstellen:

- Pandas-Dokumentation: pandas.pydata.org/docs – das vollständige Nachschlagewerk für DataFrames, Funktionen und Beispiele.
- Anaconda-Paketliste: repo.anaconda.com – eine Übersicht aller verfügbaren Pakete, die in Python in Excel genutzt werden können.

Alle Beiträge auf business-wissen.de finden Sie hier gesammelt – **von den ersten Schritten** bis zu **fortgeschrittenen Analysen**. Ideal zum Nachschlagen und Vertiefen.

Damit haben Sie bereits einen sehr großen Schritt gemacht: vom klassischen Excel-Anwender hin zum modernen Datenanalysten.

Empfehlungen aus dem Management-Handbuch

Python in Excel - erste Schritte und Berechnungen

Wie Sie aus einer Datentabelle in Excel Daten in einen DataFrame übernehmen und dann mit Python die Daten bearbeiten und Kennzahlen berechnen. Das Vorgehen wird Schritt für Schritt erklärt und mit technischen Hintergründen erläutert.

<https://www.business-wissen.de/id/kapitel/300/>

Python in Excel für die Datenanalyse

Wie Sie Datenanalyse mit Python und Pandas systematisch durchführen, Datenstrukturen verstehen, aussagekräftige Kennzahlen berechnen und schnell anschauliche Diagramme erstellen. Mit praxisnahen Beispielen und Schritt-für-Schritt-Anleitungen zur Bereinigung, Analyse und Aufbereitung von Datensätzen.

<https://www.business-wissen.de/id/kapitel/318/>

Power Pivot für die Datenanalyse

Wie Sie Excel-Power-Pivot für die Datenanalyse in Ihrem Unternehmen nutzen. Mit einer Schritt-für-Schritt-Anleitung, um Daten und Tabellen in das Power-Pivot-Datenmodell zu laden und dann mit PivotTable zu analysieren.

<https://www.business-wissen.de/id/kapitel/282/>

ChatGPT und Excel-Formeln optimieren

Wie Sie Ihre Arbeit mit Excel vereinfachen und verbessern, indem Sie ChatGPT oder ein anderes KI-Tool nutzen. Mit vielen Beispielen für typische Anwendungsfälle und Excel-Formeln und ausführlichen Erläuterungen, was ChatGPT für Ihre Excel-Probleme beitragen kann.

<https://www.business-wissen.de/id/kapitel/290/>

Nutzen Sie als
Premium-Mitglied
alle
Handbuch-Kapitel
mit mehr als
3.000 Checklisten und Excel-Vorlagen

Jetzt anmelden

www.business-wissen.de/anmelden/

Impressum

b-WISE GmbH Business Wissen Information Service
Bismarckstraße 21
76133 Karlsruhe
DEUTSCHLAND

service@business-wissen.de
Telefon +49 721 18397-0

Copyright 2026, b-wise GmbH, All Rights Reserved