LEEON
Advanced Learning

Der Datenbutler

AI-Cooking: KI-Rezepte für Alltag und Job

Für absolute Einsteiger:innen
Schritt für Schritt mit VS Code & GitHub Copilot

Autorin: Karima Charles, KI-Trainerin unterstützt durch die KI

AI-Cooking

Rezept: Der Datenbutler

Befördere deinen KI-Co-Piloten zum persönlichen Butler! In dieser Mission baust du eine wiederverwendbare "Reinigungs-Maschine" – eine Python-Funktion, die auf Knopfdruck jeden unordentlichen Datensatz für dich aufbereitet. Lerne, wie du mit GitHub Copilot ganze Arbeitsabläufe automatisierst und dir so stundenlange, repetitive Arbeit sparst!!

INHALT

Überblick

Diese Anleitung führt dich durch den Bau deiner ersten Python-Funktion mit VS Code und Copilot:

- wie du mit "Prompts im Code" (Kommentaren) GitHub Copilot anleitest, eine Funktion zu erstellen,
- wie du mehrere Reinigungsschritte in dieser einen Funktion bündelst,
- wie du die Funktion definierst und sie dann auf einen realen Datensatz anwendest,
- und wie du so einen wiederkehrenden Prozess für deinen Job-Alltag automatisierst.

Du lernst ganz nebenbei:

- Den Unterschied zwischen dem Definieren und dem Aufrufen einer Funktion.
- Wie man Code sauber und wiederverwendbar strukturiert.
- Wie du deine Effizienz mit einem integrierten KI-Assistenten auf ein neues Level hebst.

1. MISSION & VORBEREITUNG IN DER WERKSTATT

1.1. Das Ziel: Vom manuellen Schritt zur Automatisierung

Herzlich willkommen zur Mission "Der Daten-Butler"! In Woche 1 haben wir Daten Schritt für Schritt in einzelnen Jupyter-Zellen bereinigt. Das war ideal zum Lernen, aber im Job-Alltag ist es zu umständlich.

Unser Ziel ist es jetzt, diese Reinigungsschritte in einer einzigen, wiederverwendbaren Python-Funktion zu bündeln. Diese Funktion können wir dann auf jeden ähnlichen Datensatz anwenden, ohne alle Schritte erneut schreiben zu müssen. Das ist der Kern von Automatisierung.

1.2. VS Code richtig einrichten: Der wichtigste Klick des Tages

Dieser Schritt ist der wichtigste des ganzen Tages, da er den FileNotFoundError verhindert. Wir müssen VS Code mitteilen, in welchem Ordner unser Projekt liegt.

Deine Aufgabe:

1. Starte Visual Studio Code.
2. Klicke im Menü oben auf File -> Open Folder....
3. Navigiere zum Ordner Daten_Butler, in dem deine Kursmaterialien liegen.
4. Klicke den Ordner einmal an und bestätige mit "Ordner auswählen".

Ergebnis: Links im Datei-Explorer von VS Code siehst du jetzt die Dateien Daten-Butler_Vorlage.py und sales_messy.csv. Die Umgebung ist korrekt eingerichtet.

2. SCHRITT 1: DIE DATEN PRÜFEN

Zuerst verschaffen wir uns einen Überblick über die "unordentlichen" Daten.

Deine Aufgabe:

1. Öffne die Datei Daten-Butler_Vorlage.py.
2. Führe den Code mit einem Klick auf den Play-Button (▶) oben rechts aus.

Was passiert im Terminal?

Der Code aus der Vorlage wird ausgeführt. Der Output im Terminal zeigt dir:

- .info(): Die Spalte Sales hat weniger Einträge als die anderen (fehlende Werte). Die Spalte Order Date hat den Typ object (Text), anstatt eines Datumsformats.
- .head(): Du siehst die ersten fünf Zeilen der Tabelle.

Hinweis: Eine rote Meldung von conda kann erscheinen. Diese ist für unsere Aufgabe unwichtig und kann ignoriert werden.

Wir haben die Probleme identifiziert. Jetzt können wir die Funktion bauen, um sie zu beheben.

3. SCHRITT 2: DIE FUNKTION SCHRITT FÜR SCHRITT BAUEN

3.1. Arbeitsbereich vorbereiten

GitHub Copilot ist sehr hilfsbereit. Manchmal zu hilfsbereit. Wenn er den Code aus Schritt 1 sieht, wird er versuchen, die gesamte Funktion auf einmal zu schreiben. Das wollen wir nicht. Wir wollen sie Schritt für Schritt selbst bauen. Daher deaktivieren wir den Code aus Schritt 1 vorübergehend.

Deine Aufgabe:

1. Markiere den gesamten Code in deiner Daten-Butler_Vorlage.py-Datei.
2. Drücke die Tastenkombination Strg + # (Windows) oder Cmd + # (Mac).

Ergebnis: Jede Zeile beginnt nun mit einem #. Der Code ist auskommentiert und damit für Python und Copilot inaktiv. Der Arbeitsbereich ist bereit.

3.2. Das Grundgerüst der Funktion erstellen

Deine Aufgabe:

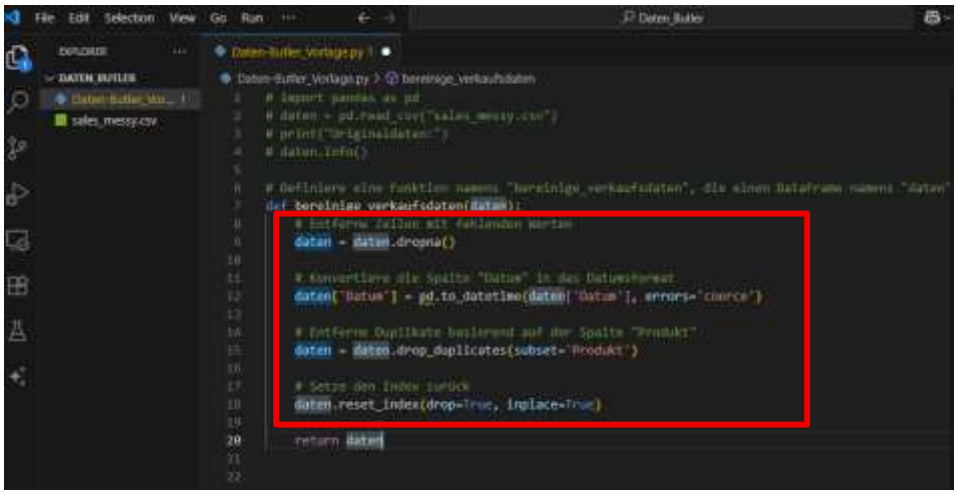
Gehe ans Ende der Datei, lasse eine Leerzeile und schreibe den folgenden Kommentar. Drücke danach die Enter-Taste.

```
# Definiere eine Funktion namens "bereinige_verkaufsdaten", die einen
DataFrame namens "daten" entgegennimmt und diesen am Ende wieder
zurückgibt.
```

Ergebnis: Copilot schlägt das Grundgerüst der Funktion vor. Nimm den Vorschlag mit der Tab-Taste an.

3.3. Reinigungs-Schritt 1 hinzufügen: Fehlende Umsätze behandeln

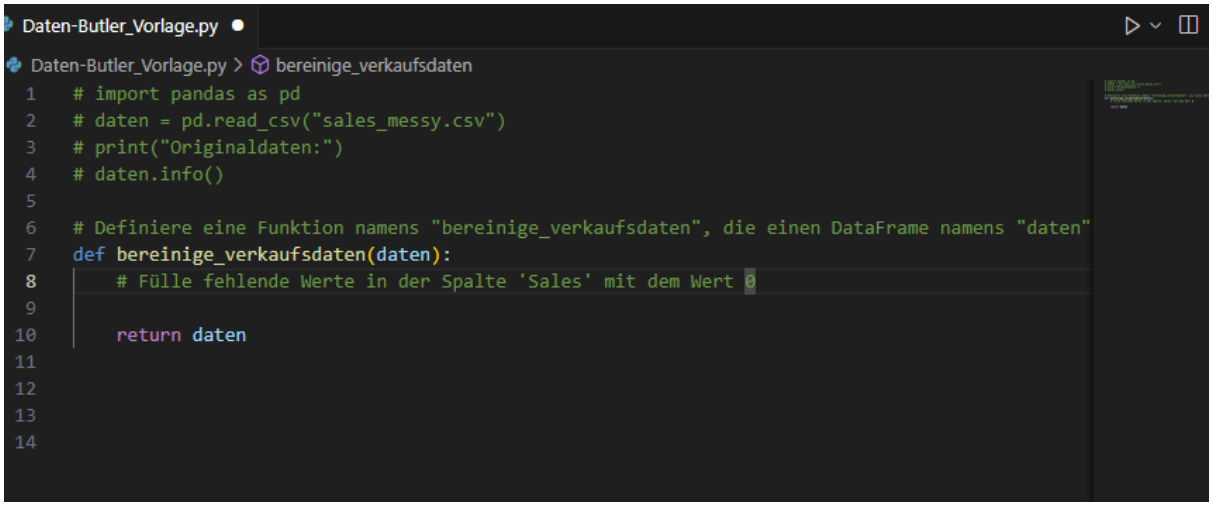
Wir fügen nun den ersten Befehl in die Funktion ein (in den eingerückten Bereich zwischen def und return). Achte auf die Einrückung.



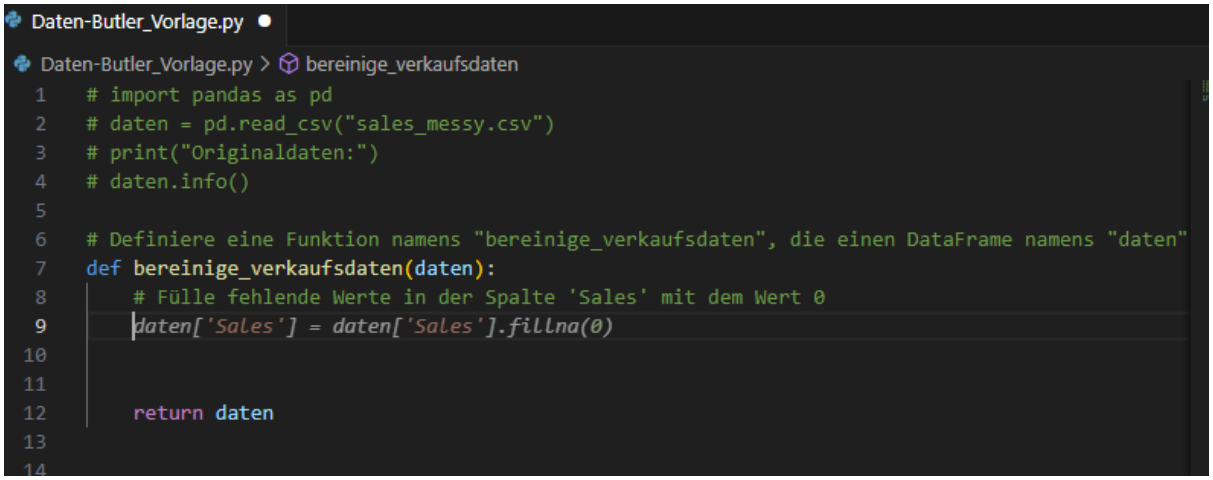
Deine Aufgabe:

Schreibe in der Funktion den folgenden Kommentar und drücke Enter.

Fülle fehlende Werte in der Spalte 'Sales' mit dem Wert 0



Ergebnis: Copilot schlägt den passenden Code-Schnipsel vor. Nimm ihn mit der Tab-Taste an.

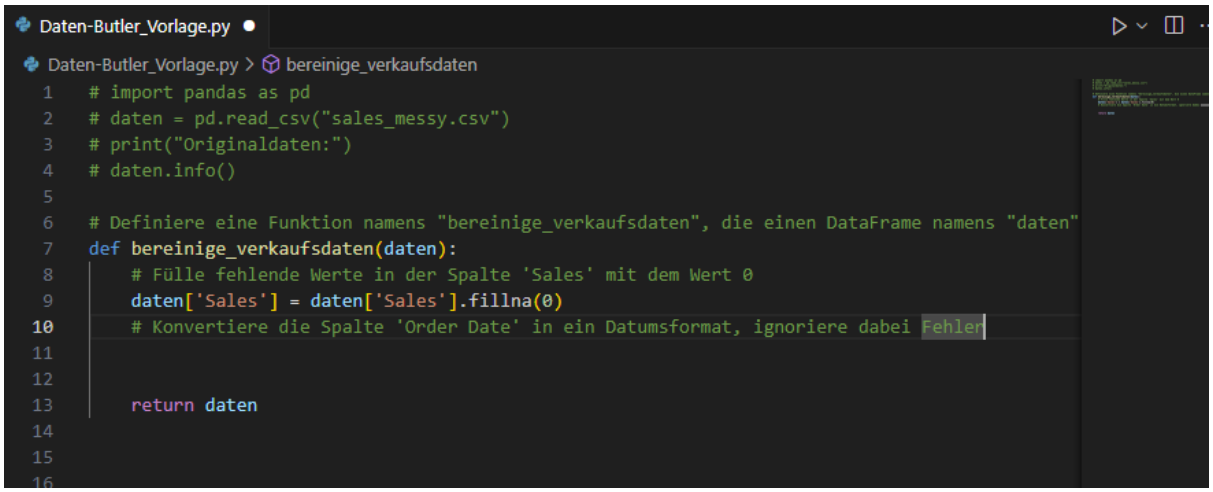


3.4. Reinigungs-Schritt 2 hinzufügen: Datumsangaben korrigieren

Deine Aufgabe:

Direkt unter der letzten Zeile (aber immer noch innerhalb der Funktion), schreibe den nächsten Kommentar und drücke Enter.

```
# Konvertiere die Spalte 'Order Date' in ein Datumsformat, ignoriere dabei Fehler
```



Ergebnis: Copilot schlägt den Code für die Datumskonvertierung vor. Nimm ihn an. Die Funktion ist jetzt fertig gebaut.

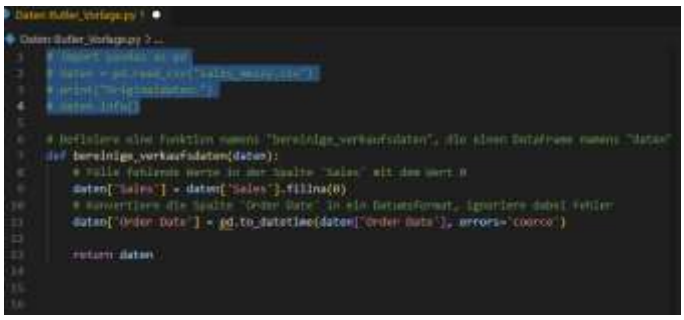
```python
# import pandas as pd
# daten = pd.read_csv("sales_messy.csv")
# print("Originaldaten:")
# daten.info()

# Definiere eine Funktion namens "bereinige_verkaufsdaten", die einen DataFrame namens "daten"
def bereinige_verkaufsdaten(daten):
    # Fülle fehlende Werte in der Spalte 'Sales' mit dem Wert 0
    daten['Sales'] = daten['Sales'].fillna(0)
    # Konvertiere die Spalte 'Order Date' in ein Datumsformat, ignoriere dabei Fehler
    daten['Order Date'] = pd.to_datetime(daten['Order Date'], errors='coerce')


    return daten
```

4. SCHRITT 3: DIE FUNKTION TESTEN

Die Funktion ist gebaut. Jetzt testen wir, ob sie funktioniert. Dazu müssen wir den Code zum Laden der Daten wieder aktivieren und unsere neue Funktion aufrufen.

Deine Aufgabe:

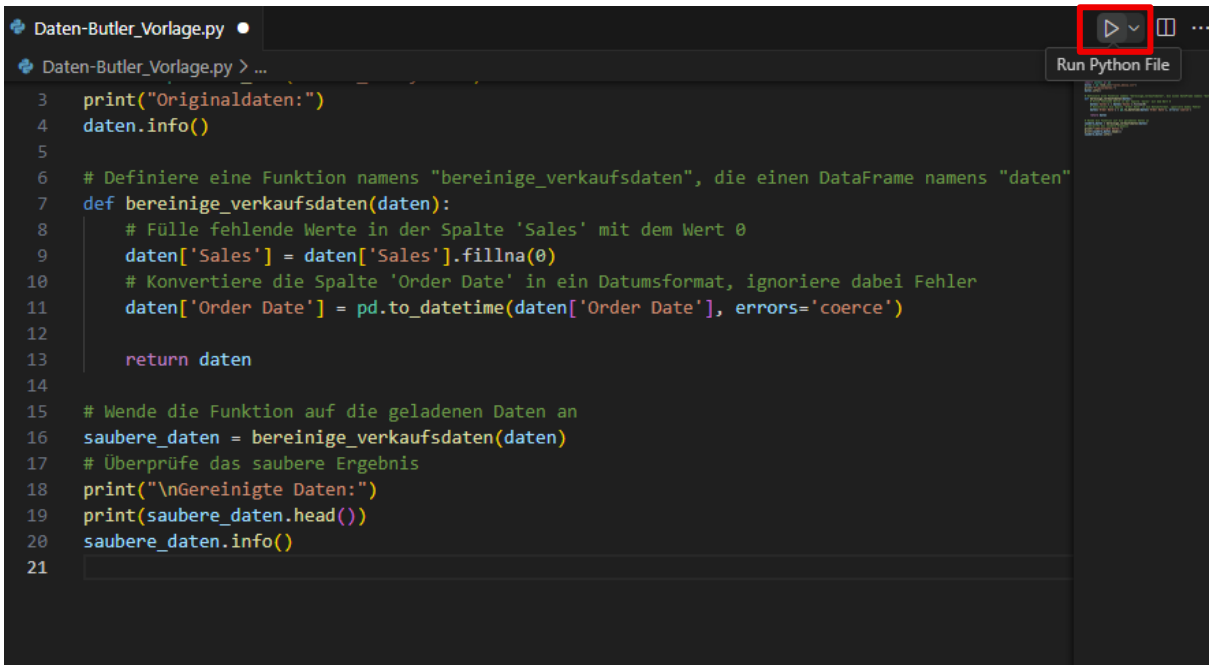
Gehe zum Anfang der Datei. Markiere den auskommentierten Block und aktiviere ihn wieder mit der Tastenkombination Strg + # oder Cmd + #.



Wird zu:

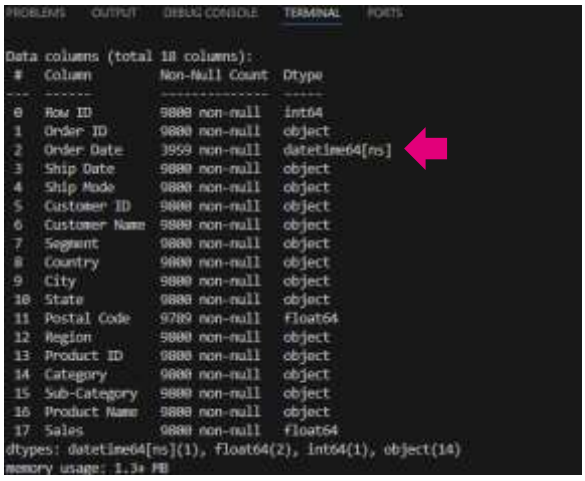
Gehe ganz ans Ende der Datei und füge den folgenden Code hinzu, um die Funktion aufzurufen und das Ergebnis zu prüfen:

```
# Wende die Funktion auf die geladenen Daten an

saubere_daten = bereinige_verkaufsdaten(daten)


# Überprüfe das saubere Ergebnis

print("\nGereinigte Daten:")

print(saubere_daten.head())

saubere_daten.info()
```



Führe den Code mit einem Klick auf den Play-Button (▶) oben rechts aus.

Ergebnis: Sieh dir den neuen Output im Terminal an. Die Sales-Spalte hat keine fehlenden Werte mehr und Order Date ist jetzt ein datetime-Objekt. Die Funktion arbeitet korrekt!



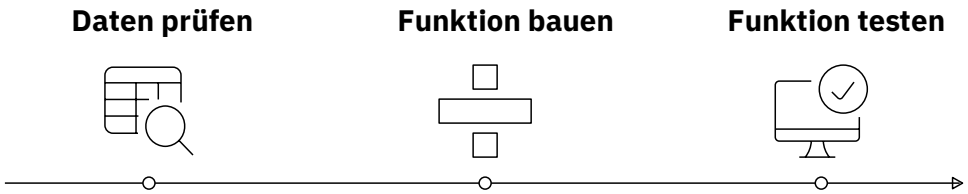
5. WEITERE FUNKTIONS-TECHNIKEN

So kannst du Funktionen noch verbessern:

- Mehrere Parameter: Eine Funktion kann mehrere Inputs annehmen, z.B. def meine_funktion(daten, land):.
- Default-Werte: Gib einem Parameter einen Standardwert, z.B. def meine_funktion(daten, land='DE'):.
- Docstrings: Beschreibe direkt unter der def-Zeile in drei Anführungszeichen """...""", was deine Funktion tut.

6. ZUSAMMENFASSUNG: DEIN WORKFLOW

Dein Workflow zur Automatisierung:



1. Daten prüfen: Probleme identifizieren.
2. Funktion bauen: Ein Grundgerüst erstellen und schrittweise mit Logik füllen.
3. Funktion testen: Auf die Daten anwenden und das Ergebnis verifizieren.

7. TYPISCHE FEHLER – UND WAS DU TUN KANNST

IndentationError:

Der häufigste Fehler bei Funktionen! Der Code innerhalb deiner Funktion muss eingerückt sein. Wenn eine Zeile nicht richtig eingerückt ist, meldet Python diesen Fehler.

NameError:

Du hast die Funktion aufgerufen, aber die Zelle/das Skript, in der du sie definiert hast, noch nicht ausgeführt.

Ganz andere Fehler? Keine Panik! Die Profi-Strategie

Wenn dein Butler nicht das tut, was er soll, und du eine lange, rote Fehlermeldung bekommst, ist das kein Grund zur Sorge. Es ist Zeit für eine Teambesprechung mit deinem KI-Co-Piloten!

Dein Prompt: Nutze unseren gelernten Profi-Prompt zur Fehlerbehebung. Kopiere einfach die folgende Vorlage, füge deine spezifischen Informationen ein und schicke sie an ChatGPT.

Hallo! Ich bin Anfänger und lerne gerade in VS Code, eine Python-Funktion mit GitHub Copilot zu erstellen.

Das wollte ich erreichen:
[Beschreibe hier dein Ziel in einfachen Worten, z.B.: "Ich wollte eine Funktion schreiben, die eine CSV-Datei bereinigt und aufrufen."].

Diesen Code habe ich dafür verwendet:

Füge hier deinen kompletten Code aus der .py-Datei ein

(sowohl die Funktions-Definition als auch den Teil, der sie aufruft)

Diese Fehlermeldung habe ich bekommen:

Code

Füge hier die komplette, rote Fehlermeldung aus dem VS Code Terminal ein

Meine Frage an dich:
Was bedeutet dieser Fehler und wie kann ich den Code korrigieren, damit das Skript funktioniert? Bitte erkläre es mir einfach

Mit dieser Methode verwandelst du Frust in einen Lern-Erfolg. Du lernst nicht nur die Lösung, sondern auch, die Sprache der Fehler zu verstehen.

07.08.2025 Karima Charles unterstützt durch die KI