LEEON
Advanced Learning

Die Visualisierungs-Maschine

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: Die Visualisierungs-Maschine

Mission: Baue eine universelle "Diagramm-Maschine"! Statt für jede Analyse einen neuen Code zu schreiben, bauen wir eine einzige, flexible Funktion, die auf Befehl verschiedene Visualisierungen für dich erstellt und speichert.

INHALT

Überblick

Diese Anleitung führt dich durch den Bau deiner ersten flexiblen "Visualisierungs-Maschine" mit VS Code und Copilot:

- wie du eine Funktion mit mehreren "Stellschrauben" (Parametern) baust.
- wie du diese Funktion anweist, verschiedene Arten von Diagrammen zu erstellen (z.B. Balken- oder Streudiagramme).
- wie du mit einer einzigen Funktion mehrere unterschiedliche, professionelle Grafiken erzeugst.

Du lernst ganz nebenbei:

- Warum Wiederverwendbarkeit von Code der Schlüssel zur Automatisierung ist.
- Wie du durch das Stellen der richtigen Fragen an Copilot zu besseren Ergebnissen kommst.

1. MISSION & VORBEREITUNG

1.1. 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, 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 visualisierungs_maschine_vorlage.py und superstore_daten.csv. Die Umgebung ist korrekt eingerichtet.

1.2. Wichtiger Hinweis: Du bist der Architekt, Copilot ist dein Assistent!

Das Ziel dieses Kurses ist es nicht, dass du Code abtippst. Das Ziel ist, dass du lernst, der KI die richtigen Anweisungen zu geben.

Wir werden dir in jedem Schritt eine Anweisung in Form eines Kommentars (# ...) geben. Deine Aufgabe ist es, diesen Kommentar in VS Code zu schreiben, **Enter** zu drücken und den Vorschlag von GitHub Copilot mit der **Tab-Taste** anzunehmen. Das ist der Dialog, den du trainieren sollst.

Damit du immer ein Sicherheitsnetz hast und dein Ergebnis vergleichen kannst, zeigen wir dir im Rezept den Code, den Copilot *idealerweise* vorschlagen sollte. **Dieser Code wurde von uns validiert und funktioniert garantiert.**

Keine Sorge, wenn Copilots Vorschlag manchmal leicht abweicht – das ist normal! Wichtig ist, dass du den Prozess übst und dein Ergebnis mit unserer Vorlage abgleichst.

2. VOM PROTOTYP ZUR MASCHINE

Fantastisch! Die Werkstatt ist eingerichtet und die Materialien liegen bereit. Dein visualisierungs_maschine_vorlage.py-Skript hat bereits den ersten, wichtigsten Schritt für dich erledigt: Es hat die Daten geladen. Das ist der professionelle "Werkstatt-Check", um sicherzustellen, dass alle Teile da sind, bevor wir mit dem Bau beginnen.

Unsere Mission beginnt jetzt in der Zeile: # HIER BEGINNT DEIN CODE...

2.1. Der Bauplan der fertigen Maschine

Dein Auftrag (Warum?):

Bevor wir eine ganze Maschine bauen, erstellen wir eine schnelle Skizze auf einem Notizblock. Wir wollen testen, ob unsere Grundidee funktioniert: Den Umsatz pro Kategorie als Balkendiagramm darzustellen. Das gibt uns die Sicherheit, dass unsere Logik stimmt, bevor wir sie in einen komplexen Bauplan gießen.

Dein 1. Prompt an Copilot (Daten vorbereiten):

Schreibe unter # HIER BEGINNT DEIN CODE... den folgenden Kommentar und drücke Enter.

```
# #Auftrag 1: Erstelle ein Balkendiagramm für den Umsatz pro Kategorie.

# Dafür zuerst die Daten nach 'Category' gruppieren und die 'Sales' summieren.

# Dann die Funktion 'erstelle_diagramm' mit den richtigen Zutaten aufrufen.


# Auftrag 2: Erstelle ein Streudiagramm für den Zusammenhang von Umsatz und Profit.

# Dafür die Funktion 'erstelle_diagramm' mit dem originalen DataFrame 'df' und anderen Zutaten aufrufen.
```

WICHTIG: DIE GOLDSTANDARD-METHODE

Drücke jetzt Enter (nachdem du alles kopiert hast). Copilot wird dir den ersten Teil des Codes vorschlagen.

Deine einzige Aufgabe ist es jetzt, immer wieder die Tab-Taste zu drücken.

Nimm den ersten Vorschlag an, drücke erneut Tab, um den nächsten zu bekommen, und so weiter, bis Copilot den gesamten Auftrag für dich geschrieben hat.

Das ist der Dialog – fließend und effizient.

Das Skript sieht so aus:

```
# Auftrag 1: Erstelle ein Balkendiagramm für den Umsatz pro Kategorie.

# Dafür zuerst die Daten nach 'Category' gruppieren und die 'Sales'
summieren.

# Dann die Funktion 'erstelle_diagramm' mit den richtigen Zutaten aufrufen.


# Auftrag 2: Erstelle ein Streudiagramm für den Zusammenhang von Umsatz und
Profit.

# Dafür die Funktion 'erstelle_diagramm' mit dem originalen DataFrame 'df'
und anderen Zutaten aufrufen.

def erstelle_diagramm(df, x, y, titel, xlabel, ylabel):

    plt.figure(figsize=(10, 6))

    plt.bar(df[x], df[y], color='skyblue')

    plt.title(titel)

    plt.xlabel(xlabel)

    plt.ylabel(ylabel)

    plt.xticks(rotation=45)

    plt.tight_layout()

    plt.show()

# Auftrag 1: Umsatz pro Kategorie

umsatz_pro_kategorie = df.groupby('Category')['Sales'].sum().reset_index()

erstelle_diagramm(

    umsatz_pro_kategorie,

    'Category',

    'Sales',

    'Umsatz pro Kategorie',

    'Kategorie',

    'Umsatz'

)

# Auftrag 2: Streudiagramm für Umsatz und Profit

def erstelle_streudiagramm(df, x, y, titel, xlabel, ylabel):

    plt.figure(figsize=(10, 6))

    plt.scatter(df[x], df[y], alpha=0.5, color='orange')

    plt.title(titel)
```

```
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.tight_layout()
    plt.show()
erstelle_streudiagramm(df, 'Sales', 'Profit', 'Zusammenhang von Umsatz und Profit', 'Umsatz', 'Profit')
```

2.2. Die letzte Anweisung: Ergebnisse speichern

Dein Auftrag (Warum?):
Die Diagramme werden angezeigt, aber ein echter Report braucht speicherbare Dateien. Wir geben Copilot eine letzte, einfache Anweisung, um das zu erledigen.

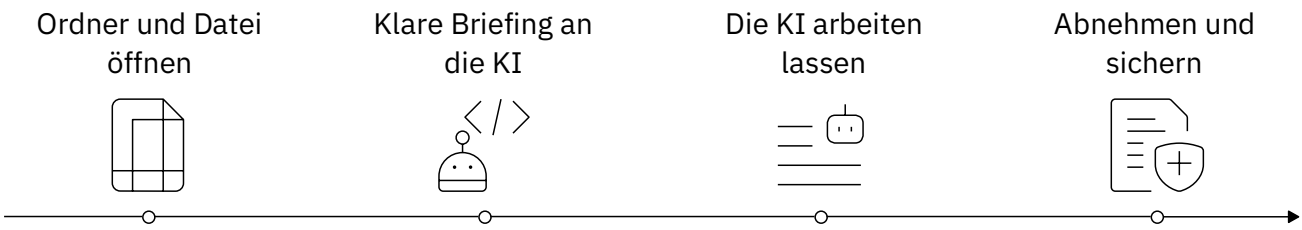
Dein Prompt an Copilot:
Füge **am Ende des gesamten Skripts** den folgenden Kommentar hinzu:

```
# Speichere beide Diagramme als PNG-Dateien
```

Was du tun musst:
Copilot wird dir jetzt Code vorschlagen, um die Grafiken zu speichern. Nimm seine Vorschläge mit der **Tab-Taste** an, bis er fertig ist.

Was ist passiert?
Du hast jetzt das Beste aus beiden Welten. Das Skript zeigt dir die Diagramme zur schnellen Kontrolle an UND speichert sie als professionelle Bilddateien in deinem Ordner.

3. ZUSAMMENFASSUNG: DEIN WORKFLOW



4. Profi-Tipp

Es wird passieren: Du drückst versehentlich Enter oder eine andere Taste, anstatt die **Tab-Taste** zu nutzen, und plötzlich ist der Vorschlag von Copilot weg oder der Code sieht seltsam aus.

Die wichtigste Regel: KEINE PANIK! Das ist kein Fehler, sondern nur eine kleine Unterbrechung im Dialog.

Stell es dir so vor: Du hast deinen Assistenten mitten im Satz unterbrochen. Er ist jetzt verwirrt. Der einfachste Weg, wieder auf Kurs zu kommen, ist, den Satz von vorne zu beginnen.

Dein einfacher 3-Schritte-Rettungsplan:

1. **Lösche** den unvollständigen Code-Schnipsel und den Kommentar, den du gerade geschrieben hast.
2. **Gib deine ursprüngliche Anweisung** (den Kommentar-Block) einfach erneut ein.
3. **Führe den Dialog jetzt konzentriert zu Ende:** Drücke Enter und dann systematisch die **Tab-Taste**, bis Copilot seinen letzten Vorschlag gemacht hat und aufhört, neuen Code zu schreiben.

Du merkst, dass Copilot fertig ist, wenn er anfängt, sich zu wiederholen oder der Cursor einfach in einer neuen, leeren Zeile wartet.

5. 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