LEEON
Advanced Learning

Der automatisierte Wochenbericht

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 automatisierte Wochenbericht

Baue eine "1-Klick-Berichtsmaschine"! In dieser Mission erstellst du ein einziges Skript, das auf Knopfdruck eine komplette Datenanalyse durchführt und einen fertigen Report in Form eines Word-Bericht ausgibt.

INHALT

Überblick

Diese Anleitung führt dich durch den Bau deines ersten, kompletten Automatisierungs-Skripts:

- wie du einen vollständigen Analyse-Workflow in einer einzigen .py-Datei programmierst.
- wie du mit parse_dates sicherstellst, dass Python Datumsangaben korrekt versteht (ein entscheidender Profi-Trick!).
- wie du mit pip install einmalig ein Spezialwerkzeug für die Arbeit mit Word-Dokumenten installierst.
- wie du Text- und Bild-Platzhalter in einer Word-Vorlage intelligent ersetzt.
- wie du deinem Skript beibringst, das aktuelle Datum zu verwenden und Dateinamen dynamisch zu erstellen...

1. MISSION & VORBEREITUNG IN DER WERKSTATT

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 automatischer_wochenbericht_vorlage.py und wochen_verkaufsdaten.csv, und berichtsvorlage.docx. Die Umgebung ist korrekt eingerichtet.

1.2. Einmaliges Werkzeug-Upgrade: Python das Sprechen mit Word beibringen

Unsere Python-Standardausstattung kann nicht mit Word-Dateien arbeiten. Wir müssen das nötige Spezialwerkzeug einmalig installieren.

Deine Aufgabe:

Gehe in VS Code unten in das Terminal-Fenster.

Tippe den folgenden Befehl exakt so ein und drücke Enter:

```
pip install python-docx
```

Warte, bis die Installation abgeschlossen ist. Deine Werkstatt ist nun bereit für Word!

1.3. 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. BAU DER BERICHTS-MASCHINE, SCHRITT FÜR SCHRITT

Jetzt bauen wir unsere Maschine, indem wir die Befehle aus unserem finalen, validierten Skript Schritt für Schritt eingeben und verstehen.

Deine Aufgabe:

Öffne die leere Datei automatischer_wochenbericht_vorlage.py und folge den Anweisungen.

2.1. Die Werkzeuge importieren

Wir sagen Python, welche Werkzeuge wir heute benötigen.

```
# Importiere die Bibliotheken pandas, matplotlib.pyplot und docx
import pandas as pd
import matplotlib.pyplot as plt
from docx import Document
from docx.shared import Inches, Cm
from datetime import date
```

2.2. Daten laden und Datum verstehen

Für dieses Rezept haben wir dir eine perfekt aufbereitete Datendatei zur Verfügung gestellt. Die Spalte 'Order Date' enthält saubere, fehlerfreie Datumsangaben. Damit unser Skript diese Spalte nicht nur als Text, sondern als echten Kalender behandelt, müssen wir Python zwei kleine Anweisungen geben.

```
# Lade die CSV-Datei und aktiviere die 'Order Date'-Spalte als
intelligenten Kalender-Index
df = pd.read_csv('wochen_verkaufsdaten.csv', parse_dates=['Order Date'],
index_col='Order Date')
```

Was passiert hier?

- parse_dates=['Order Date'] sagt Python: "Hey, die Spalte namens 'Order Date' ist kein normaler Text, das ist ein Kalender! Behandle sie bitte auch so."
- index_col='Order Date' sagt Python: "Benutze diesen Kalender jetzt als das Rückgrat, als den Haupt-Organisator unserer Tabelle."

2.3. Die Daten zusammenfassen (Aggregieren)

Wir aggregieren die Daten. Was bedeutet "aggregieren"? Stell dir vor, du hast eine Kiste mit hunderten Kassenbons – einen für jeden einzelnen Verkaufstag. Das sind zu viele Details, um einen Trend zu erkennen. Aggregieren bedeutet, dass wir diese Details auf eine höhere Ebene zusammenfassen. Wir nehmen alle Kassenbons von Montag bis Sonntag, legen sie in eine Box mit der Aufschrift "Woche 1" und schreiben die Gesamtsumme auf die Box. Das machen wir für jede Woche.

```
# Aggregiere die 'Sales'-Spalte auf wöchentlicher Basis.

woechentliche_verkaeufe = df.resample('W')['Sales'].sum()
```

Was passiert hier?

df.resample('W') ist der Befehl, die "wöchentlichen Boxen" zu erstellen.

['Sales'].sum() sagt Python, dass es sich nur für die 'Sales'-Spalte interessiert und davon die Summe bilden soll.

2.4. Grafik erstellen und speichern

Wir erstellen die Visualisierung und speichern sie als .png-Datei, die wir später im Bericht verwenden.

```
# --- GRAFIK ERSTELLEN UND SPEICHERN ---

# Erstelle eine neue Grafik mit einer angenehmen Größe

plt.figure(figsize=(10, 5))


# Erstelle ein Liniendiagramm für die woechentlichen_verkaeufe

plt.plot(woechentliche_verkaeufe.index, woechentliche_verkaeufe.values,
marker='o', linestyle='-', color='b')


# Gib der Grafik den Titel 'Wöchentliche Verkaufszahlen'

plt.title('Wöchentliche Verkaufszahlen')


# Beschrifte die y-Achse mit 'Umsatz in USD'

plt.ylabel('Umsatz in USD')

# Beschrifte die x-Achse mit 'Woche'

plt.xlabel('Woche')
```

2.5. Dynamische Berichts-Infos vorbereiten

Wir holen uns das aktuelle Datum und bereiten die KI-Zusammenfassung vor.

```
# --- BERICHTS-INFORMATIONEN VORBEREITEN ---


# 1. Die Uhr ablesen

heute = date.today()


# 2. Das Datum in zwei Formaten aufbereiten

datum_fuer_bericht = heute.strftime('%d.%m.%Y') # Format "Tag.Monat.Jahr"

datum_fuer_dateiname = heute.strftime('%Y-%m-%d') # Format "Jahr-Monat-Tag"
für saubere Dateinamen


# --- KI-ZUSAMMENFASSUNG ERSTELLEN ---


# Erstelle eine kurze, professionelle Zusammenfassung in einem Satz für
Manager.

zusammenfassung = "Die Daten zeigen, dass die wöchentlichen Verkäufe nach
einem starken Start schwanken, aber insgesamt einen leichten Aufwärtstrend
aufweisen."

ki_zusammenfassung = zusammenfassung
```

2.6. Den Word-Bericht zusammensetzen

Das ist das große Finale! Wir öffnen die Vorlage und füllen die Platzhalter.

code

Python

```
# Öffne das Word-Dokument 'berichtsvorlage.docx'

doc = Document('berichtsvorlage.docx')


# Führe "Suchen & Ersetzen" für das gesamte Dokument aus

for p in doc.paragraphs:

    # Ersetze Text-Platzhalter

    p.text = p.text.replace('{{TITEL}}', 'Wöchentlicher Verkaufsbericht')

    p.text = p.text.replace('{{ZUSAMMENFASSUNG}}', ki_zusammenfassung)

    p.text = p.text.replace('{{Date}}', datum_fuer_bericht)
```

```
    # Ersetze den Grafik-Platzhalter
    if '{{GRAFIK}}' in p.text:
        p.text = '' # Leere den Absatz
        p.add_run().add_picture('wochenbericht_grafik.png', width=Cm(15.0))
```

2.7. Den finalen Bericht speichern

Wir verwenden unseren dynamischen Dateinamen.

```
# Erstelle den dynamischen Dateinamen
finaler_dateiname = f"wochenbericht_{datum_fuer_dateiname}.docx"

# Speichere das fertige Dokument
doc.save(finaler_dateiname)

print(f"Der automatisierte Word-Bericht wurde erfolgreich erstellt!")
```

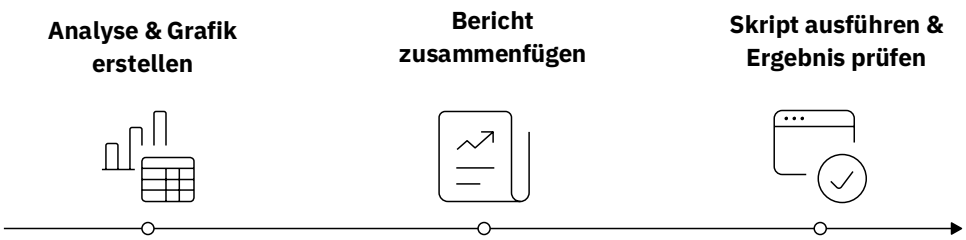
3. TESTLAUF DER FERTIGEN MASCHINE

Deine Aufgabe:

1. Führe den Code mit einem Klick auf den Play-Button (▶) oben rechts aus.
2. Führe das Skript aus. Öffne die neue .docx-Datei in Word (Python kann das nicht 😉) und bewundere deinen vollautomatischen, tagesaktuellen Bericht

3. ZUSAMMENFASSUNG: DEIN WORKFLOW

Dein Workflow zur Automatisierung:



1. Analyse & Grafik erstellen
2. Bericht zusammenfügen
3. Skript ausführen & Ergebnis prüfen

4. 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