Analyse einer Excel-Datei mit Python

TWIE23B
Joel Knorr 3552863,
Fabian Häfner 4303866,
Mike Kärcher 9613520

THALES

Gliederung

1. Erläuterung der Grundfunktion für den Anwender
2. Detaillierte Programmvorstellung
3. Demonstration des fertigen Programms & Projektreflexion

Erläuterung der Grundfunktion für den Anwender

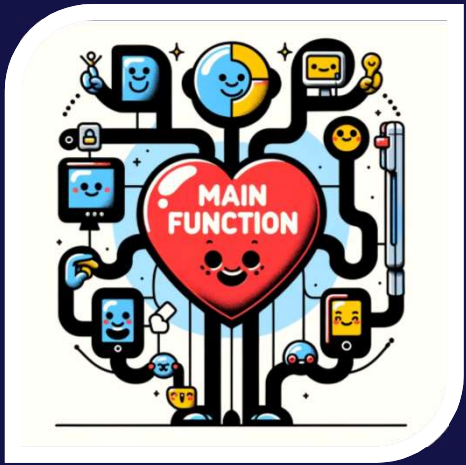
	A	B	C	D	E	F	G	H	I	J
1	Produktgruppe	Artikel	Bestellnummer	Kunde	Bestellmenge	Lieferung	Bestelldatum	Verfügbarkeit	Lieferminge	Wert
2	101	ABC	AI20001001	Muster AG	600.00	10/12/2010	9/14/2010	9/13/2010	600.00	4,794.00
3	199	DEF	AI20001001	Muster AG	25.00	10/12/2010	9/14/2010	9/13/2010	25.00	612.50
4	101	GHI	AI20001001	Muster AG	1,200.00	10/12/2010	9/14/2010	9/13/2010	1,200.00	18,000.00
5	201	XYZ	AI20001002	Test GmbH	100.00	9/27/2010	8/26/2010	8/24/2010	100.00	8,000.00
6	200	UVW	AI20001111	No Name GbR	10.00	9/27/2010	8/30/2010	8/27/2010	10.00	125.00
7	101	ABC	AI20001005	Probe GmbH	35.00	9/28/2010	9/7/2010	9/4/2010	35.00	279.65
8	201	XYZ	AI20001005	Probe GmbH	120.00	9/28/2010	9/7/2010	9/3/2010	120.00	9,600.00
9	101	ABC	AE10101678	Beispiel GmbH	250.00	9/28/2010	9/7/2010	9/3/2010	250.00	1,997.50
10	199	DEF	AE10101678	Beispiel GmbH	200.00	9/28/2010	9/7/2010	9/3/2010	200.00	4,900.00
11	101	GHI	AE10101678	Beispiel GmbH	100.00	9/28/2010	9/7/2010	9/3/2010	100.00	1,500.00
12	201	XYZ	AE10101678	Beispiel GmbH	50.00	9/28/2010	9/7/2010	9/3/2010	50.00	4,000.00
13	200	UVW	AE10101678	Beispiel GmbH	25.00	9/28/2010	9/7/2010	9/3/2010	25.00	312.50
14	101	ABC	AI20001003	Übung AG	2,000.00	10/15/2010	9/10/2010	9/8/2010	2,000.00	15,980.00
15	199	DEF	AI20001003	Übung AG	500.00	10/15/2010	9/10/2010	9/8/2010	500.00	12,250.00
16	101	GHI	AI20001003	Übung AG	100.00	10/15/2010	9/10/2010	9/8/2010	100.00	1,500.00
17	101	GHI	AE10101682	Felix Test AG	50.00	10/12/2010	9/14/2010	9/10/2010	50.00	750.00
18	200	UVW	AE10101682	Felix Test AG	1,000.00	10/12/2010	9/14/2010	9/10/2010	1,000.00	12,500.00
19	101	ABC	AI20001006	Test & Partner	90.00	9/18/2010	8/28/2010	8/26/2010	90.00	719.10
20	101	ABC	AI20001009	Muster & Söhne	10.00	9/23/2010	8/26/2010	8/24/2010	10.00	79.90
21	200	UVW	AI20001004	Übungsgesellschft	210.00	10/12/2010	9/14/2010	9/10/2010	210.00	2,625.00
22	101	ABC	AE10101683	Dummy AG	800.00	10/13/2010	9/15/2010	9/14/2010	800.00	6,392.00
23	201	XYZ	AE10101683	Dummy AG	1,250.00	10/13/2010	9/21/2010	9/20/2010	1,250.00	100,000.00
24	101	GHI	AI20001112	P. Robe GbR	50.00	9/23/2010	8/30/2010	8/27/2010	50.00	750.00
	199	DEF	AI20001112	P. Robe GbR	15.00	9/23/2010	8/30/2010	8/27/2010	15.00	367.50

Funktionen:

- ✓ Durchschnitt basierend auf einer Spalte
- ✓ gewichteter Durchschnitt basierend auf einer Spalte
- ✓ Summe des Gesamtwerts nach Gruppe und Parameter
- ✓ Zeichnen eines Balkendiagramms

main

- Steuert Ablauf des Programms
- Darunter:
  - Abfrage des Benutzers
  - Aufruf der verschiedenen Funktionen
  - Ergebnisse festhalten und an Funktion übergeben
  - Fehlerbehandlung für ungültige Benutzereingaben



xfile_read

```
#Lesen der Exceldatei
def xfile_read(filename):
    # Lesen der Daten aus der Excel-Datei in ein DataFrame.
    df = pd.read_excel(filename, dtype=str)

    # Die Spaltennamen (Kopfzeile) in eine Liste umwandeln und in header_a speichern.
    header_a = df.columns.tolist()

    # Die Datenzeilen in eine Liste von Listen umwandeln und in alldata speichern.
    alldata = df.values.tolist()

    # Die extrahierte Kopfzeile und die Datenzeilen zurückgeben.
    return header_a, alldata
```

create_header_dict

```python
# Diese Funktion erstellt ein Wörterbuch aus der übergebenen Kopfzeilen-Liste.
def create_header_dict(header_a):
    header_dict = {}  # Initialisiere ein leeres Wörterbuch

    # Durchlaufe alle Elemente in der Kopfzeilen-Liste
    for i in range(len(header_a)):
        # Füge jedes Element der Liste als Schlüssel in das Wörterbuch ein.
        # Der Wert ist der Index des Elements in der Liste.
        header_dict[header_a[i]] = i

    # Gib das fertige Wörterbuch zurück
    return header_dict
```

xfile_write

```
#Ergebnisse werden an Funktion übergeben, um in eine Excel zu schreiben
data_out = [["Suchkriterium", "Durchschnitt", "Gewichteter Durchschnitt"], [search_term, mean, weighted_mean]]
xfile_write(data_out)
```

```
#Ergebenisse in eine Exceldatei schreiben
def xfile_write(data):
    filename = "100_Pivot_Output.xlsx"

    # Daten in ein DataFrame umwandeln
    df = pd.DataFrame(data)

    # Excel-Datei erstellen oder überschreiben und DataFrame speichern
    df.to_excel(filename, index=False)
```

calc_mean_by_index

```python
# Funktion zum Berechnen des Durchschnitts basierend auf einem bestimmten Spaltenindex
def calc_mean_by_index(alldata, search_term='Liefermenge', header = 'Liefermenge'):
    # Bestimme die Anzahl der Zeilen in den Daten
    num_rows = len(alldata)

    # Bestimme den Spaltenindex für den gesuchten Begriff mithilfe des zuvor erstellten header_dict
    index = header[search_term]

    total = 0
    # Summiere alle Werte in der ausgewählten Spalte
    for row in alldata:
    # Konvertiere den Wert in der Spalte mit dem Index "index" in einen Dezimalwert (float) und addiere ihn zu "total".
    # Dabei wird die Gesamtsumme der Werte berechnet.
        total += float(row[index])

    # Berechne den Durchschnitt, wenn Daten vorhanden sindn
    mean = total / num_rows if num_rows > 0 else 0

    # Gib den berechneten Durchschnitt zurück
    return mean
```

calc_weighted_mean_by_index

```python
# Funktion zum Berechnen des gewichteten Durchschnitts basierend auf einem bestimmten Spaltenindex
def calc_weighted_mean_by_index(min_value, alldata, search_term='Liefermenge', header = 'Liefermenge'):
    # Initialisiere die gesamte gewichtete Summe und die Anzahl der Werte über dem Mindestwert
    total_weighted_sum = 0
    counter = 0

    # Bestimme den Spaltenindex für den gesuchten Begriff mithilfe des zuvor erstellten header_dict
    index = header[search_term]

    # Iteriere über alle Zeilen in den Daten
    for row in alldata:
        # Extrahiere den aktuellen Wert in der ausgewählten Spalte und konvertiere ihn in Float
        current_value = float(row[index])

        # Überprüfe, ob der aktuelle Wert größer als das Mindestwert ist
        if current_value > min_value:
            # Addiere den aktuellen Wert zur gewichteten Summe
            total_weighted_sum += current_value
            # Inkrementiere die Anzahl der Werte über dem Mindestwert
            counter += 1

    # Berechne den gewichteten Durchschnitt, wenn die Anzahl der Werte über dem Mindestwert nicht null ist
    if counter != 0:
        weighted_mean = total_weighted_sum / counter
    elif counter == 0:
        weighted_mean = 0

    # Gib den berechneten gewichteten Durchschnitt zurück
    return weighted_mean
```

```python
# Funktion zum Konvertieren von deutschen Float-Zahlen ins Englische Format
def german_to_english_float(germfloat_string):
    germfloat_string = germfloat_string.replace(",", ".")
    return germfloat_string
```

draw_graph

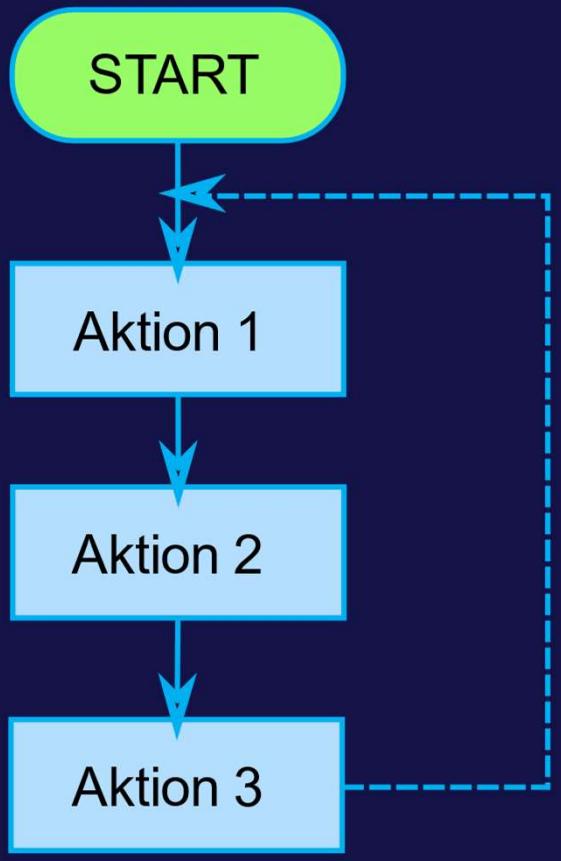
```python
# Funktion zum Zeichnen eines Matplotlib-Balkendiagramms basierend auf den Daten
def draw_graph(alldata, search_term='Liefermenge'):
    index = header_dict[search_term]  # Der Spaltenindex für den gesuchten Begriff
    x_values = range(1, len(alldata) + 1)  # X-Werte für die Datenpunkte
    y_values = [float(row[index]) for row in alldata]  # Y-Werte für die Datenpunkte

    plt.grid(True)
    plt.bar(x_values, y_values)  # Erstellt ein Matplotlib-Balkendiagramm
    plt.xlabel('Datennummer')  # Setzt das Label für die X-Achse
    plt.ylabel(search_term)  # Setzt das Label für die Y-Achse
    plt.title(f'{search_term} Verteilung')  # Setzt den Titel des Diagramms
    plt.show()  # Zeigt das Diagramm an
```

weighted_sum

```python
def weighted_sum(alldata, parameter='101', search_term='Produktgruppe', header = 'Produktgruppe'):
    # Initialisiere die Summe mit 0.0
    summe = 0.0
    index = header[search_term]
    # Durchlaufe jede Zeile in der Datenliste
    for row in alldata:
        # Prüfe, ob der Wert im angegebenen Suchfeld (`search_term`) dem `parameter` entspricht
        if row[index] == parameter:
            # Konvertiere den Wert im Feld 'Wert' zu einer Fließkommazahl und addiere ihn zur Summe
            wert = float(row[header['Wert']])
            summe += wert

    # Gib die berechnete Summe zurück
    return summe
```

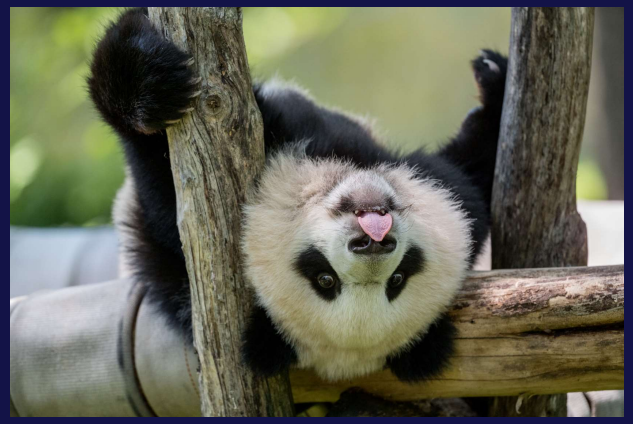


Demonstration des fertigen Programms

Projektreflexion

- Gemeinsam überlegt was zu tun
- Zusammen den Code geschrieben
- Jeder hat Ideen miteingebracht
- Nachträglich hat sich jeder mit dem Code beschäftigt → Verbesserungen gefunden → Gemeinsam über DC optimiert

- Zeitaufwendig da keine Arbeitsteilung
- Wenn Arbeitsteilung:
  - Gute Absprache im Vorhinein
  - Ständige Kommunikation und Transparenz



Vielen Dank!