elab2go-Workshop
Prof. Dr. Eva Maria Kiss | Hochschule Kaiserslautern

Wie groß ist der Anteil erneuerbarer Energien an der Stromproduktion? – Eine grafische Analyse

Wie ändert sich der Verbrauch von Strom insgesamt, Solar- und Windenergie über die Jahre? Wann im Laufe eines Jahres ist der Verbrauch am höchsten? Reicht die Energiegewinnung über erneuerbare Energien aus, um den Verbrauch zu decken?
Um diese Fragen zu beantworten, werden die Stromverbrauchsdaten in eine Jupyter Notebook-App eingelesen und grafisch dargestellt.

Übersicht

1 Vorbereitung

In den verlinkten Tutorials finden Sie Anleitungen zur Verwendung der Programmiersprache Python.

- Python Tutorial (https://www.elab2go.de/demo-py1/)
- Jupyter Notebook verwenden (https://www.elab2go.de/demo-py1/jupyter-notebooks.php)

1-1 Programmbibliotheken installieren

Um Pakete verwenden zu können, müssen sie zuvor mit Hilfe des pip-Befehls installiert werden. Z.B.

```
pip install matplotlib
pip install pandas
```

1-2 Programmbibliotheken importieren

Zunächst importieren wir die Programmbibliotheken Pandas und Matplotlib.

In Python kann man mit Hilfe der **import-Anweisung** entweder eine komplette Programmbibliothek importieren, oder nur einzelne Funktionen.

Beim Import werden für die jeweiligen Bibliotheken oder Funktionen Alias-Namen vergeben.

In [1]:
```
import pandas as pd # Wird für Datenverwaltung und Datenbereinigung benötigt
import matplotlib.pyplot as plt # Wird für die Visualisierung benötigt
# Funktion zur formatierten Ausgabe eines DataFrames
def display_dataframe( df, rows=6, cols=None):
    with pd.option_context('display.max_rows', rows, 'display.max_columns', col
        display(df);
```

2 Stromerzeugung

2-1 Daten zur Stromerzeugung einlesen

Die csv-Datei mit den Stromerzeugungs-Daten finden Sie hier:
https://www.elab2go.de/demo-py2/stromerzeugung_de_2018-2023.csv
(https://www.elab2go.de/demo-py2/stromerzeugung_de_2018-2023.csv)

In [2]:
```
# Lese CSV-Datei mit Stromerzeugungsdaten ein, die 0-te Spalte enthält Zeilenüb
file = "https://www.elab2go.de/demo-py2/stromerzeugung_de_2018-2023.csv"
df_erz = pd.read_csv(file, index_col=0)
# Ersetze fehlende Werte mit 0
df_erz = df_erz.fillna(0)
display_dataframe(df_erz)
```

Energieart	2018	2019	2020	2021	2022	2023
Steinkohle	25035	25293	22458	23499	18830	18127
Braunkohle	21275	21205	21067	20487	18544	17692
Erdgas	31361	31664	31712	31942	30553	31808
...	...	...	...	...	...	...
Wind Offshore	5051	6393	7504	7774	7787	8129
Wind Onshore	51633	52792	53184	54499	55289	57590
Sonstige Erneuerbare	496	445	407	420	352	384

16 rows × 6 columns

2-2 Farbschema festlegen

Lege für jede Energieart eine Farbe fest.
TODO: Ändere das Farbschema ab, so dass erneuerbare Energien in Grün-Schattierungen angezeigt werden.

In [3]:

```
# Farbschema als Dictionary
dict_konv = {'Steinkohle': '#6c757d', 'Braunkohle':'peru',
             'Erdgas': 'peachpuff','Erdöl': 'slateblue',
             'Kernenergie': '#dc3545', 'Ersatzbrennstoff': 'gray',
             'Sonstige Konventionelle': 'lightgray'}
dict_ern = {'Geothermal': 'coral', 'Pumpspeicher': 'deepskyblue',
           'Wasserkraft': 'lightskyblue', 'Wasser Reservoir': 'dodgerblue',
           'Biomasse': 'olive','Solar': '#ffc107', 'Wind Offshore': 'lightblue'
            'Wind Onshore': 'lightcyan','Sonstige Erneuerbare': 'lightgray'}
dict_colormap = dict_konv | dict_ern # Komplettes Farbschema
# Farbschema als Liste
colormap_konv = list(dict_konv.values());colormap_ern = list(dict_ern.values())
colormap = colormap_konv + colormap_ern;
print("Farbschema als Liste");print(colormap);
```

```
Farbschema als Liste
['#6c757d', 'peru', 'peachpuff', 'slateblue', '#dc3545', 'gray', 'lightgray', 'cor
al', 'deepskyblue', 'lightskyblue', 'dodgerblue', 'olive', '#ffc107', 'lightblue',
'lightcyan', 'lightgray']
```

2-3 Tabelle "Stromerzeugung" einfärben

In [4]:

```
def apply_color(df):
    list1 = ['background-color:']*len(df.index)  # 16x 'background-color'
    list2 = colormap  # 16 Farben
    # Elementweise Verknüpfung der beiden Listen
    style_list = [s1+s2 for s1,s2 in zip(list1,list2)]
    return style_list
df_erz.style.apply(apply_color)
```

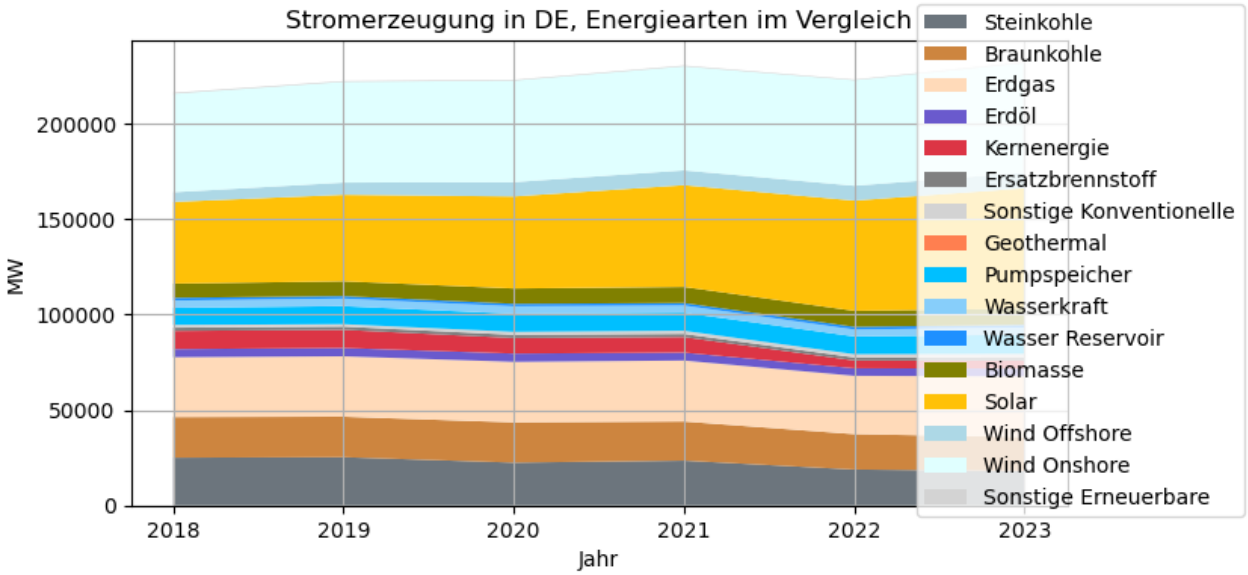
Out[4]:

Energieart	2018	2019	2020	2021	2022	2023
Steinkohle	25035	25293	22458	23499	18830	18127
Braunkohle	21275	21205	21067	20487	18544	17692
Erdgas	31361	31664	31712	31942	30553	31808
Erdöl	4271	4356	4373	4184	3966	4083
Kernenergie	9516	9516	8114	8114	4056	4056
Ersatzbrennstoff	1686	1686	1661	1640	1599	1912
Sonstige Konventionelle	1418	1235	1558	1639	1679	1706
Geothermal	38	42	47	44	51	56
Pumpspeicher	8918	9422	9422	9422	9280	10028
Wasserkraft	3860	3983	3970	3814	3660	3715
Wasser Reservoir	1440	1298	1298	1298	1397	1433
Biomasse	7396	7752	7987	8400	8332	8467
Solar	42804	45299	48206	53302	57744	63066
Wind Offshore	5051	6393	7504	7774	7787	8129
Wind Onshore	51633	52792	53184	54499	55289	57590
Sonstige Erneuerbare	496	445	407	420	352	384

2-4 Daten visualisieren

Mit Hilfe eines passenden Diagramms (hier: Stackplot) werden die Daten visualisiert. So kann man Trends auf einen Blick erkennen!

In [5]:
```
fig = plt.figure(figsize=[8, 4])
plt.stackplot(df_erz.columns, df_erz, colors=colormap, labels=df_erz.transpose(
plt.title('Stromerzeugung in DE, Energiearten im Vergleich')
plt.grid(True)
plt.xlabel('Jahr');plt.ylabel('MW');
plt.legend(bbox_to_anchor=(1.2, 1.1));
```



FAZIT: Der Anteil an erneuerbaren Energien steigt, insbesondere Wind Onshore und Solar / Photovoltaik.

2-5 Vergleich: Konventionelle vs. erneuerbare Energien

Wie ist die Entwicklung bei der Erzeugung eneuerbarer Energien vs. konventioneller Energien? Wir erstellen zwei separate Diagramme für konventionelle und erneuerbare Energien, und zeigen sie nebeneinander an.

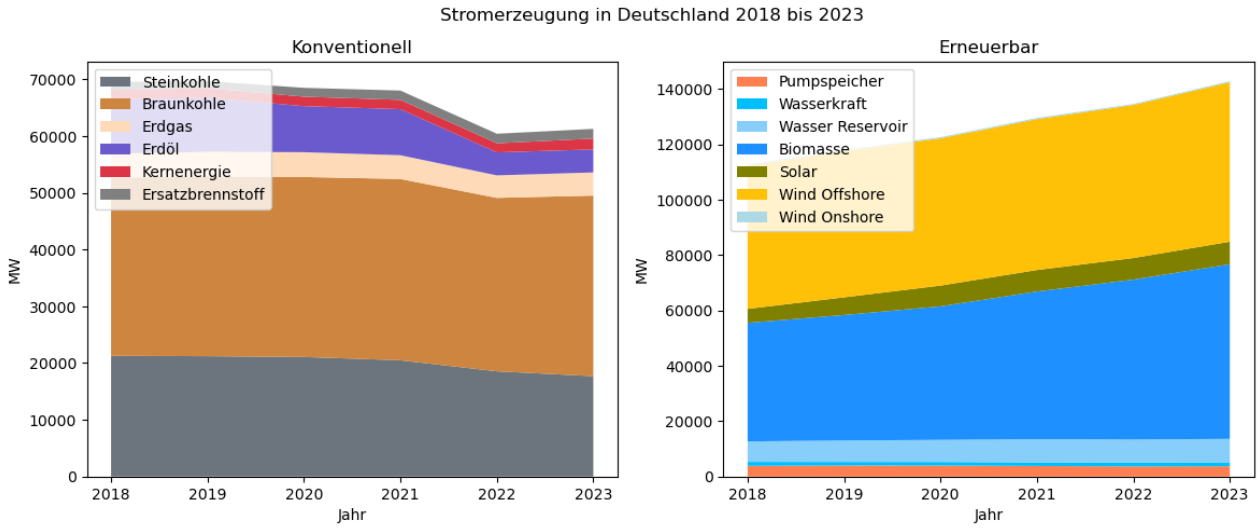
TODO:

- Ändere die Anordnung der Bilder, so dass sie untereinander erscheinen.
- Ändere die Größe der Bilder.
- Erstelle ein neues Diagramm, in dem links nur Erdöl und Erdgas und rechts nur Wind und Solar angezeigt werden.

In [6]:
```
df_konv = df_erz.iloc[0:7, :] # Konventionell= Spalten 0 bis 7
df_ern = df_erz.iloc[8:16, :] # Erneuerbar = Spalten 8 is 16
```

In [7]:

```
# Erstelle ein Diagramm mit zwei Subplots nebeneinander
fig, axs = plt.subplots(1, 2, figsize=(14, 5), sharey=False)
plt.subplot(1,2,1)
axs[0].stackplot(df_konv.columns, df_konv.iloc[1:8],
                 colors=colormap_konv, labels=df_konv.transpose().columns);
axs[0].set_title('Konventionell')
axs[0].set_xlabel('Jahr'); axs[0].set_ylabel('MW'); axs[0].legend(loc='upper le
plt.subplot(1,2,2)
axs[1].stackplot(df_ern.columns, df_ern.iloc[1:8],
                 colors=colormap_ern, labels=df_ern.transpose().columns);
axs[1].set_title('Erneuerbar')
axs[1].set_xlabel('Jahr'); axs[1].set_ylabel('MW'); axs[1].legend(loc='upper le
fig.suptitle('Stromerzeugung in Deutschland 2018 bis 2023');
```



FAZIT: Bei erneuerbaren Energien steigender Trend.

2-6 Interaktive Visualisierung

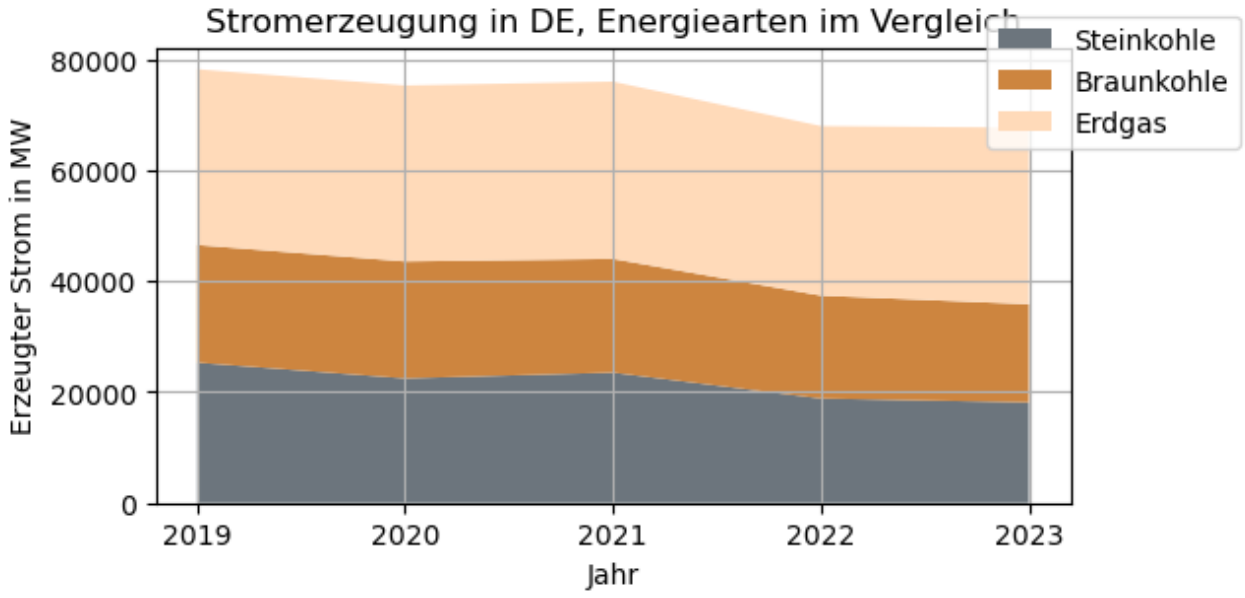
Die bisherigen Visualisierungen sind immer noch unübersichtlich: zu viele Daten auf einem Diagramm. Uns interessieren z.B. speziell die Energie-Arten Wind und Solar. In diesem Schritt soll die Auswahl bestimmter Energie-Arten und Jahre mit Hilfe einer interaktiven Visualisierung ermöglicht werden. Dafür erstellen und testen wir zunächst eine Funktion **select_data()**, die die Auswahl eines Datums-Bereichs und bestimmter Zeilen ermöglicht.

TODO:

- Die Funktion select_data() soll mit unterschiedlichen Parametern aufgerufen werden, z.B. sollen die Energiearten Solar, Wind Onshore und Wind Offshore gegenübergestellt werden.

In [8]:

```
# (1) Definiere Funktion select_data() für die Auswahl von Datum + Zeilen
def select_data(datumVon, datumBis, zeilen):
    df = df_erz.loc[zeilen, datumVon:datumBis];
    fig = plt.figure(figsize=[6, 3])
    cmap = list(dict_colormap[k] for k in df.index)
    plt.stackplot(df.columns, df, colors=cmap, labels=df.transpose().columns);
    plt.title('Stromerzeugung in DE, Energiearten im Vergleich')
    plt.grid(True);plt.xlabel('Jahr');plt.ylabel('Erzeugter Strom in MW');
    plt.legend(bbox_to_anchor=(1.2, 1.1));plt.show();
# (2) Funktion testen
select_data('2019', '2023', ['Steinkohle', 'Braunkohle', 'Erdgas'])
```



Als nächstes erstellen wir die interaktive Visualisierung mit Jupyter Widgets. **Jupyter Widgets** sind Steuerelemente, die über die Python-Bibliothek ipywidgets zur Verfügung gestellt werden, und mit deren Hilfe man ein Jupyter Notebook um grafische Benutzeroberflächen erweitern kann.

In [9]:

```
from ipywidgets import Output, Dropdown, SelectMultiple, Tab, VBox, HBox, Layou
from ipywidgets import interactive_output
out1, out2 = Output(), Output()
with out1: # Ausgabe für das erste Tab: Die Daten-Tabelle
    file = "https://www.elab2go.de/demo-py2/stromerzeugung_de_2018-2023.csv"
    df_erz = pd.read_csv(file, index_col=0)
    df_erz = df_erz.fillna(0)
    display_dataframe(df_erz)
with out2:  # Ausgabe für das zweite Tab: Das Diagramm
    # Widgets für die Auswahl des Datums und der Zeilen
    datum_von = Dropdown(description='Datum von: ', options=df_erz.columns,  va
    datum_bis = Dropdown(description='Datum bis: ', options=df_erz.columns,  va
    ui_zeilen = SelectMultiple(description='Auswahl:',
                               options=df_erz.index,
                               value=['Erdgas', 'Solar', 'Wind Offshore', 'Winc
                               rows=18)
    out = interactive_output(select_data, {'datumVon': datum_von, 'datumBis': c
    display(HBox([VBox([datum_von, datum_bis, ui_zeilen]), out]))

tab = Tab(children = [out1, out2],
          layout=Layout(width='100%'))
tab.set_title(0, 'Daten')
tab.set_title(1, 'Grafik')
display(tab)
```

```
Tab(children=(Output(), Output()), layout=Layout(width='100%'), selected_index=0,
titles=('Daten', 'Grafik'))
```

3 Stromverbrauch analysieren

3-1 Daten zum Stromverbrauch einlesen

Die csv-Datei mit den Stromverbrauchs-Daten finden Sie hier:
elab2go.de/demo-py2/stromverbrauch_de_2018-2023_stunde.csv (https://www.elab2go.de/demo-py2/stromverbrauch_de_2018-2023_stunde.csv)

In [10]:

```
# Lese CSV-Datei mit Stromverbrauchsdaten ein, die 0-te Spalte enthält den Tag
file = 'https://www.elab2go.de/demo-py2/stromverbrauch_de_2018-2023_stunde.csv'
df = pd.read_csv(file, parse_dates=True, dayfirst=True, index_col=0, thousands=
df = df.fillna(0) # Ersetze fehlende Werte mit 0
display_dataframe(df, 4) # Zeige erste und letzte zwei Zeilen zur Kontrolle an
```

Datum	Anfang	Ende	Verbrauch	Residuallast	Pumpspeicher
2018-01-01	00:00	01:00	45433.25	14015.50	1393.75
2018-01-01	01:00	02:00	44270.00	11634.25	1793.75
...	...	...	...	...	...
2023-09-11	22:00	23:00	50596.75	48413.00	37.50
2023-09-11	23:00	00:00	47181.75	45421.50	193.00

49919 rows × 5 columns

3-2 Stromverbrauch (stündlich und monatlich)

Wie hat sich der Stromverbrauch in den Jahren 2018 bis 2023 entwickelt? Dafür visualisieren wird die Daten aus der Spalte "Verbrauch", zunächst in der ursprünglichen Auflösung, dann in einer aggregierten monatlichen Auflösung. Die Aggregation wird in Python mit Hilfe der Pandas-Methode **resample()** umgesetzt.

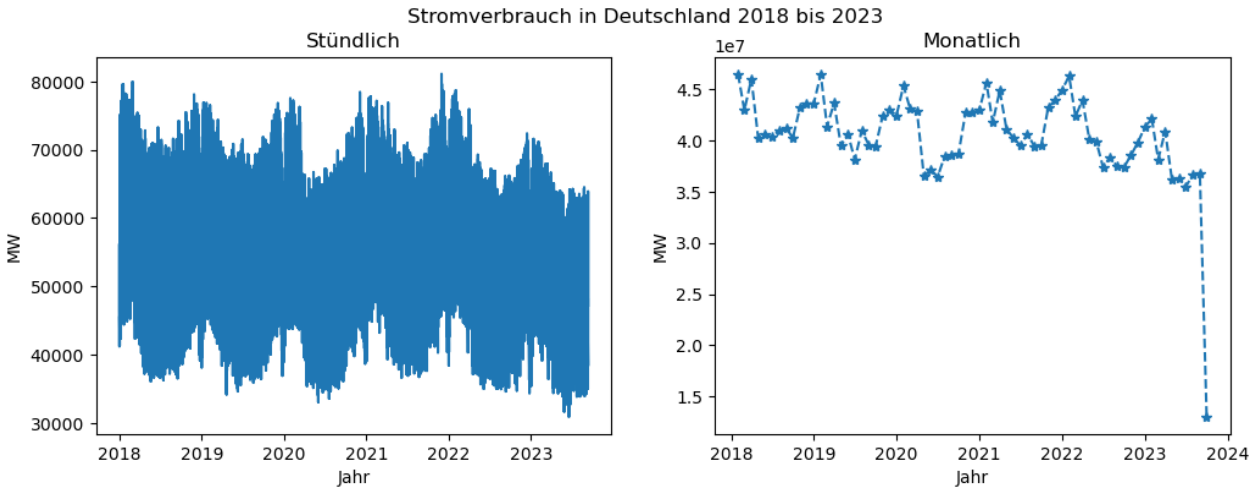
In [11]:
```
# Behalte aus dem Datensatz nur die angegebenen Spalten
df_verbrauch = df.copy().iloc[:,[0,1,2]];
# Zeige erste und letzte zwei Zeilen zur Kontrolle an
display_dataframe(df_verbrauch,4)
# Monatsweise Aggregation der Spalte Verbrauch
spalten = ['Verbrauch']
# Gruppiere Spalte Verbrauch monatsweise
df_monat = df_verbrauch[['Verbrauch']].resample('M').sum()
# Spalte Monat einfügen
df_monat.insert(0, 'Monat', df_monat.index.month)
```

Datum	Anfang	Ende	Verbrauch
2018-01-01	00:00	01:00	45433.25
2018-01-01	01:00	02:00	44270.00
...	...	...	...
2023-09-11	22:00	23:00	50596.75
2023-09-11	23:00	00:00	47181.75

49919 rows × 3 columns

In [12]:

```
# Visualisierung der Daten
fig, axs = plt.subplots(1, 2, figsize=(12, 4), sharey=False)
plt.subplot(1,2,1)
axs[0].plot(df_verbrauch['Verbrauch']);
axs[0].set_title('Stündlich')
axs[0].set_xlabel('Jahr');axs[0].set_ylabel('MW');
plt.subplot(1,2,2)
axs[1].plot(df_monat['Verbrauch'], '*--');
axs[1].set_title('Monatlich')
axs[1].set_xlabel('Jahr');axs[1].set_ylabel('MW');
fig.suptitle('Stromverbrauch in Deutschland 2018 bis 2023');
```



3-3 Jährlicher Stromverbrauch

Wir haben bisher die stündlichen Verbrauchswerte betrachtet. Für die Betrachtung des Stromverbrauchs pro Jahr aggregieren wir die stündlichen Werte.

In [13]:

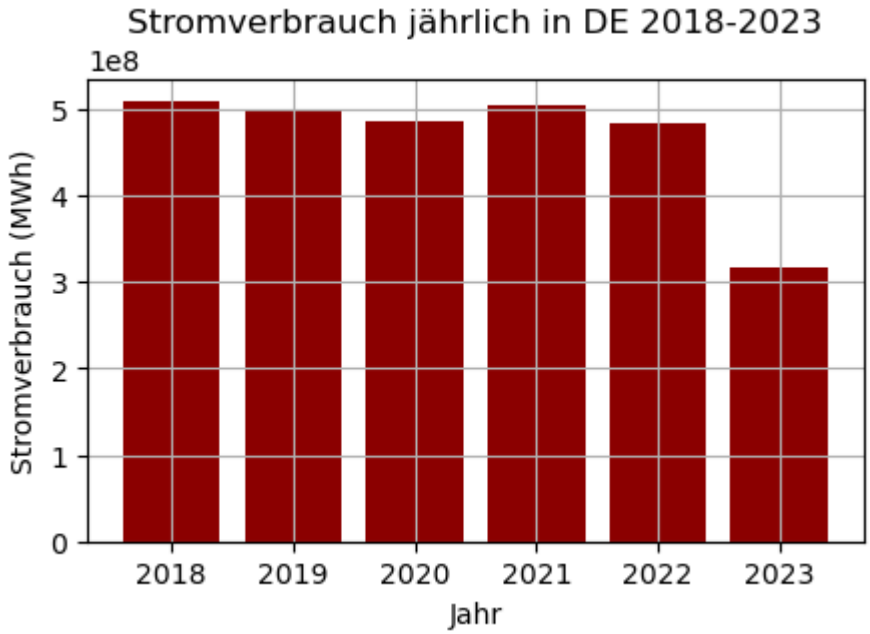
```
pd.set_option('display.float_format', '{:.2E}'.format)
# Wähle Spalte Verbrauch aus
spalten = ['Verbrauch']
# Gruppiere jede der Spalten nach Jahr
df_jahr = df_verbrauch[spalten].resample('Y').sum()
# Spalte Jahr einfügen
df_jahr.insert(0, 'Jahr', df_jahr.index.year)
df_jahr = df_jahr.rename(columns={"Verbrauch": "Verbrauch (MWh)"})
# Zeige df_jahr Kontrolle an
display_dataframe(df_jahr)
```

Datum	Jahr	Verbrauch (MWh)
2018-12-31	2018	5.09E+08
2019-12-31	2019	4.97E+08
2020-12-31	2020	4.85E+08
2021-12-31	2021	5.05E+08
2022-12-31	2022	4.83E+08
2023-12-31	2023	3.16E+08

3-4 Jährlicher Stromverbrauch - Balkendiagramm

Visualisiere jährlichen Stromverbrauch als Balkendiagramm (barplot).

In [14]:
```
fig = plt.figure(figsize=[5, 3])
plt.bar(x = list(df_jahr['Jahr']), height=list(df_jahr['Verbrauch (MWh)']), col
plt.title('Stromverbrauch jährlich in DE 2018-2023')
plt.grid(True);plt.xlabel('Jahr');plt.ylabel('Stromverbrauch (MWh)');
```



Fazit: Der Stromverbrauch ist seit 2018 unverändert hoch. Für 2023 sind die endgültigen Daten noch nicht bekannt.

4 Vergleich: Stromerzeugung vs Stromverbrauch

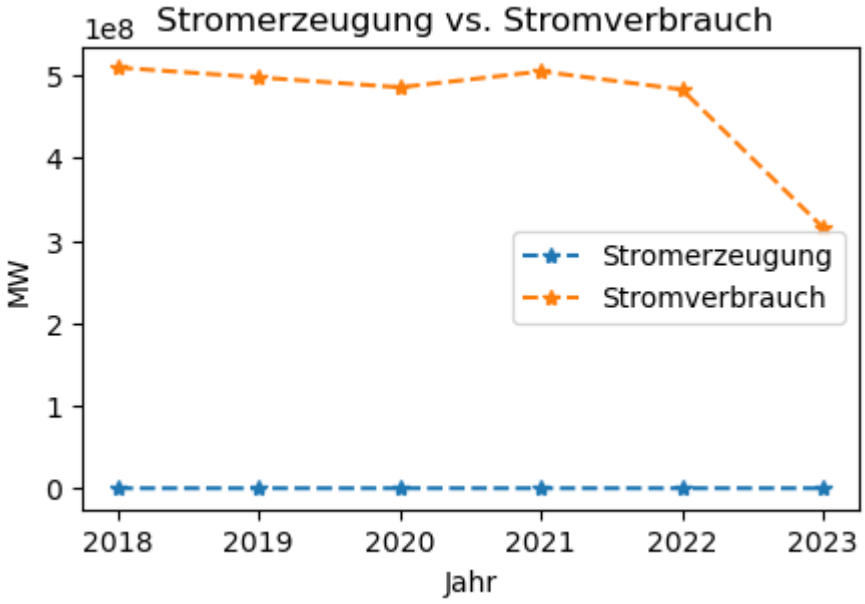
Erstelle eine Tabelle, die Stromerzeugung und Stromverbrauch gegenüberstellt.

In [15]:
```
df_erzt = df_erz.transpose()
# Füge zwei neue Spalten hinzu: Stromerzeugung und Stromverbrauch
df_erzt['Stromerzeugung'] = df_erzt[df_erzt.columns].sum(axis=1)
df_erzt['Stromverbrauch'] = list(df_jahr['Verbrauch (MWh)'])
# Zeige nur die Spalten Stromerzeugung und Stromverbrauch an
df_erzt_sel = df_erzt[["Stromerzeugung", "Stromverbrauch"]]
display_dataframe(df_erzt_sel)
```

Energieart	Stromerzeugung	Stromverbrauch
2018	216198	5.09E+08
2019	222381	4.97E+08
2020	222968	4.85E+08
2021	230478	5.05E+08
2022	223119	4.83E+08
2023	232252	3.16E+08

In [16]:

```
fig = plt.figure(figsize=[5, 3])
plt.plot(df_erzt_sel, '*--', label=df_erzt_sel.columns);
plt.title('Stromerzeugung vs. Stromverbrauch ')
plt.xlabel('Jahr');plt.ylabel('MW');
plt.legend();
```



Besuchen Sie uns auf [www.elab2go.de (http://www.elab2go.de)](http://www.elab2go.de)!