In [1]:
```
# Import necessary libraries for data analysis and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
```

In [2]:
```
# Read the Oktoberfest dataset from a CSV file into a Pandas DataFrame

df = pd.read_csv('oktoberfestgesamt19852022.csv')
```

In [3]:
```
# Display the first few rows of the DataFrame to get an initial view of the data
# This helps in understanding the structure and contents of the dataset

df.head()
```

Out[3]:

	jahr	dauer	besucher_gesamt	besucher_tag	bier_preis	bier_konsum	hendl_preis	hendl_konsum
0	1985	16	7.1	444	3.20	54541	4.77	629520
1	1986	16	6.7	419	3.30	53807	3.92	698137
2	1987	16	6.5	406	3.37	51842	3.98	732859
3	1988	16	5.7	356	3.45	50951	4.19	720139
4	1989	16	6.2	388	3.60	51241	4.22	775674

In [4]:
```
# Remove the 'besucher_tag' column from the DataFrame
# axis=1 indicates that we're dropping a column (not a row).
# inplace=True means the DataFrame is modified in place, and the result is saved in 'df'

df.drop('besucher_tag', axis=1, inplace=True)
```

In [5]:
```
# Translate the column names to english
# Create a dictionary 'german_to_english' to map German column names to English equivale
german_to_english = {'hendl_konsum':'chicken_comsump', # Rename 'hendl_konsum' to 'chick
                     'hendl_preis':'chicken_price',  # Rename 'hendl_preis' to 'chicken_
                     'bier_konsum':'bier_consump',    # Rename 'bier_konsum' to 'bier_con
                     'jahr':'year',                    # Rename 'jahr' to 'year'
                     'dauer':'duration',               # Rename 'dauer' to 'duration'
                     'besucher_gesamt':'visitors_total',  # Rename 'besucher_gesamt' to
                    'bier_preis':'bier_price'}            # Rename 'bier_preis' to 'bier

# Use the df.rename() method to rename columns based on the 'german_to_english' mapping
# inplace=True means the changes are applied directly to the DataFrame 'df'.

df.rename(columns=german_to_english, inplace = True)

# Display the first few rows of the DataFrame after renaming the columns

df.head()
```

Out[5]:

	year	duration	visitors_total	bier_price	bier_consump	chicken_price	chicken_comsump
0	1985	16	7.1	3.20	54541	4.77	629520
1	1986	16	6.7	3.30	53807	3.92	698137
2	1987	16	6.5	3.37	51842	3.98	732859
3	1988	16	5.7	3.45	50951	4.19	720139
4	1989	16	6.2	3.60	51241	4.22	775674

In [6]:
```
# Get the shape of the DataFrame 'df'
# The 'shape' attribute returns a tuple representing the dimensions of the DataFrame.
# The first element of the tuple is the number of rows, and the second element is the nu

df.shape
```

Out[6]:
```
(36, 7)
```

In [7]:
```
# Display information about the DataFrame 'df'
# The 'info()' method provides a concise summary of the DataFrame's structure and conten
# It includes details such as the number of non-null entries, data types of columns, and

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   year             36 non-null     int64  
 1   duration         36 non-null     int64  
 2   visitors_total   36 non-null     float64
 3   bier_price       36 non-null     float64
 4   bier_consump     36 non-null     int64  
 5   chicken_price    36 non-null     float64
 6   chicken_comsump  36 non-null     int64  
dtypes: float64(3), int64(4)
memory usage: 2.1 KB
```

In [8]:
```
# Generate summary statistics of the DataFrame 'df'
# The 'describe()' method computes various summary statistics for numerical columns in t
# These statistics include count, mean, standard deviation, minimum, and quartiles.

df.describe()
```

Out[8]:

	year	duration	visitors_total	bier_price	bier_consump	chicken_price	chicken_comsump
count	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000
mean	2002.555556	16.305556	6.297222	7.053333	62476.611111	7.768889	564746.138889
std	10.635371	0.624246	0.398200	2.770215	10196.196093	2.793060	129222.529768
min	1985.000000	16.000000	5.500000	3.200000	48698.000000	3.920000	313636.000000
25%	1993.750000	16.000000	5.975000	4.845000	53510.750000	5.317500	481139.250000
50%	2002.500000	16.000000	6.350000	6.750000	61467.500000	8.130000	515646.000000
75%	2011.250000	16.000000	6.500000	9.170000	71085.000000	9.890000	685465.750000
max	2022.000000	18.000000	7.100000	13.450000	79225.000000	13.960000	807710.000000

In [9]:
```
# Check if there are any missing values (NaN or null values) in the DataFrame 'df'

df.isnull().sum().any()
```

Out[9]:
```
False
```

In [10]:
```
# Get a list of column names in the DataFrame 'df'

df.columns
```

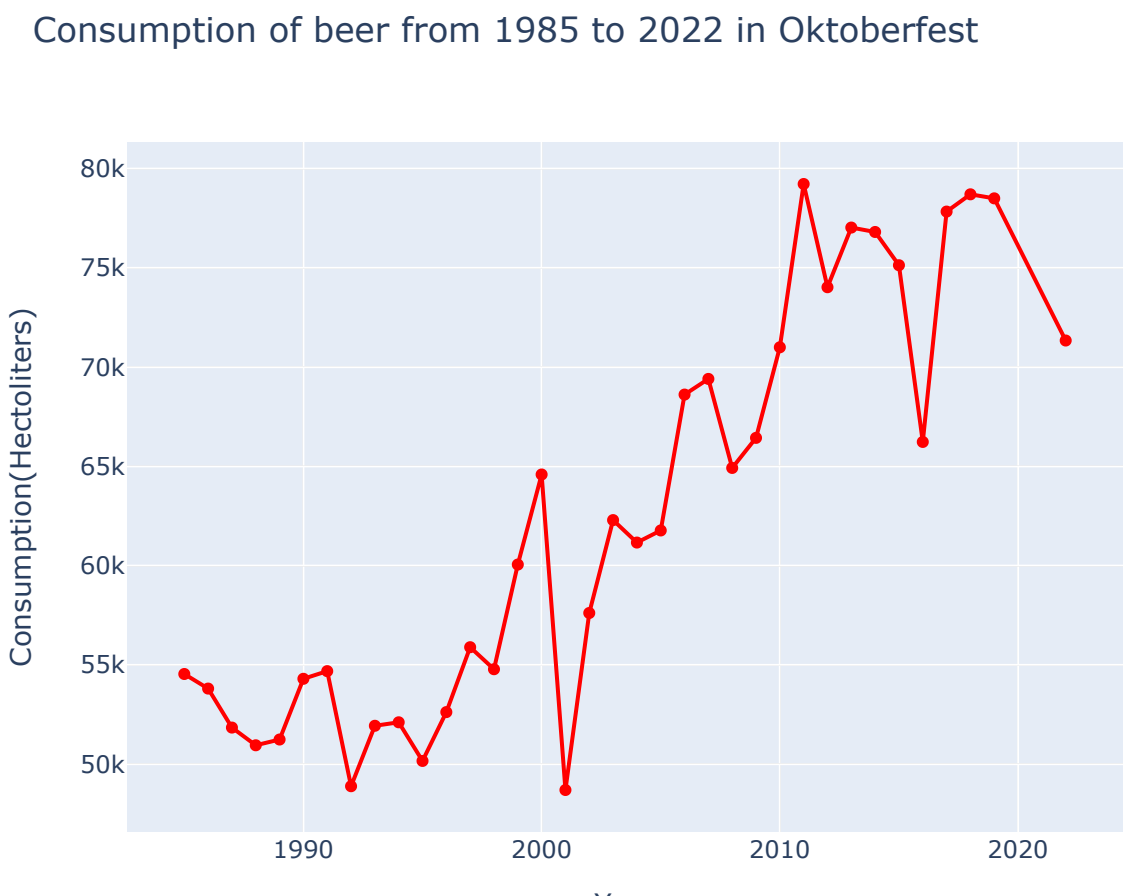
```
Index(['year', 'duration', 'visitors_total', 'bier_price', 'bier_consump',
```

Out[10]:
```
       'chicken_price', 'chicken_comsump'],
      dtype='object')
```

In [11]:
```
# Sort the DataFrame 'df' by the 'beer_consump' column in descending order
# and display the top (largest) values

df['bier_consump'].sort_values(ascending=False).head()
```

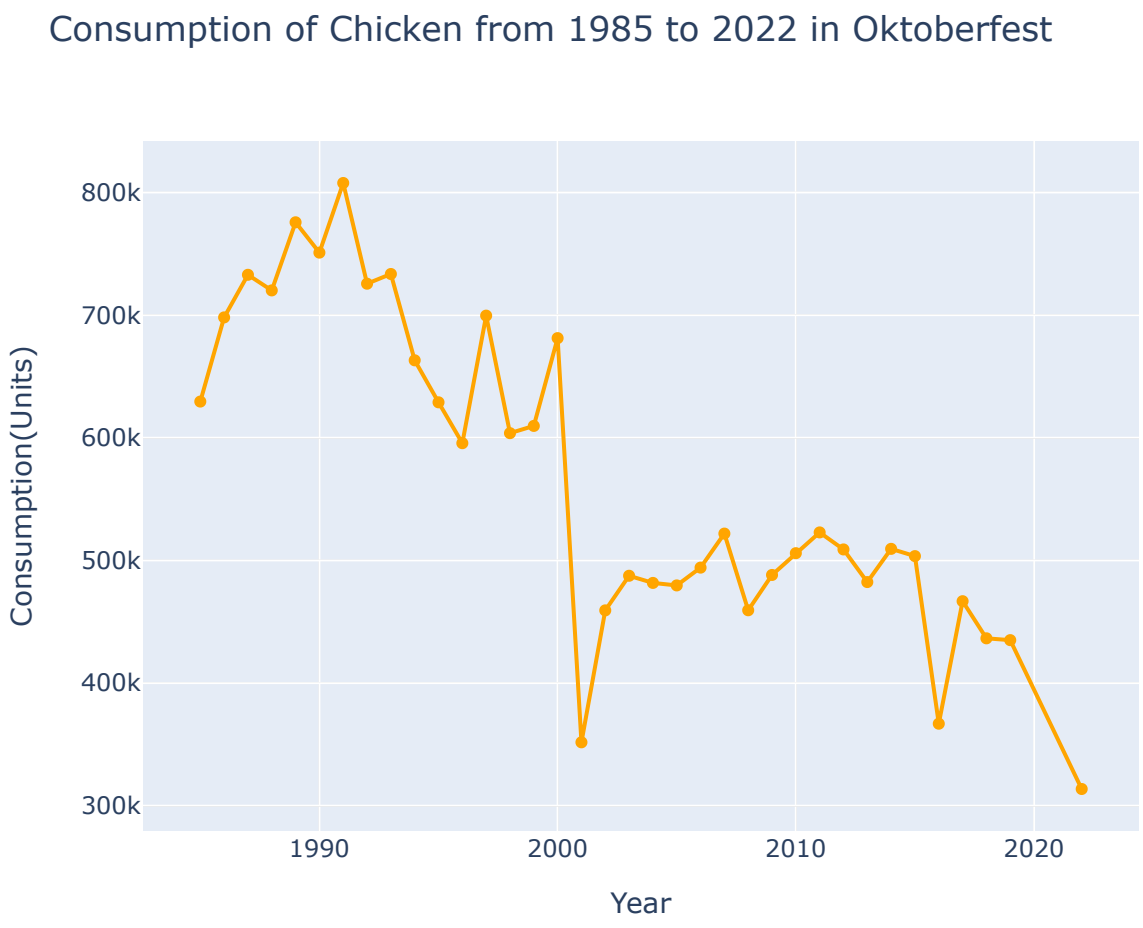
Out[11]:
```
26    79225
33    78705
34    78502
32    77836
28    77031
Name: bier_consump, dtype: int64
```

In [12]:
```
# Extract 'year' and 'beer_consump' columns from the DataFrame 'df'

year = df['year']
bier_consumption = df['bier_consump']

# Create a Plotly figure for the line plot

fig = go.Figure()

# Add a line and markers trace for bier consumption data

fig.add_trace(go.Scatter(x=year, y=bier_consumption,
                         mode='lines+markers',
                         name='Bier', line=dict(color='red')))

# Customize the layout of the plot

fig.update_layout(title='Consumption of beer from 1985 to 2022 in Oktoberfest',
                  xaxis_title='Year',
                  yaxis_title='Consumption(Hectoliters)')
```



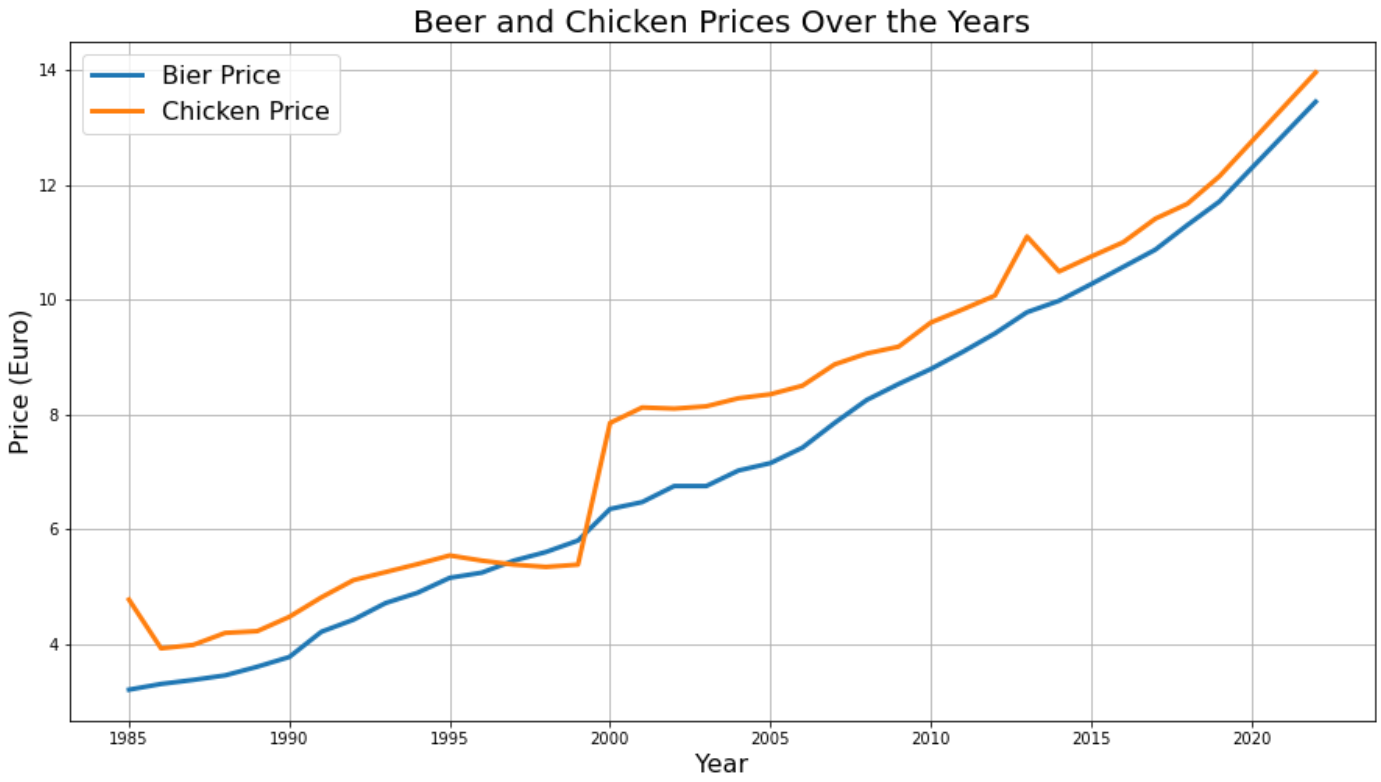
Year

In [13]:
```
# Extract 'year' and 'chicken_comsump' columns from the DataFrame 'df'

year = df['year']
chicken_consumption = df['chicken_comsump']

# Create a Plotly figure for the line plot

fig = go.Figure()

# Add a line and markers trace for chicken consumption data

fig.add_trace(go.Scatter(x=year, y=chicken_consumption,
                    mode='lines+markers',
                    name='Chicken',
                        line=dict(color='orange')))

# Customize the layout of the plot

fig.update_layout(title='Consumption of Chicken from 1985 to 2022 in Oktoberfest',
                  xaxis_title='Year',
                  yaxis_title='Consumption(Units)')
```



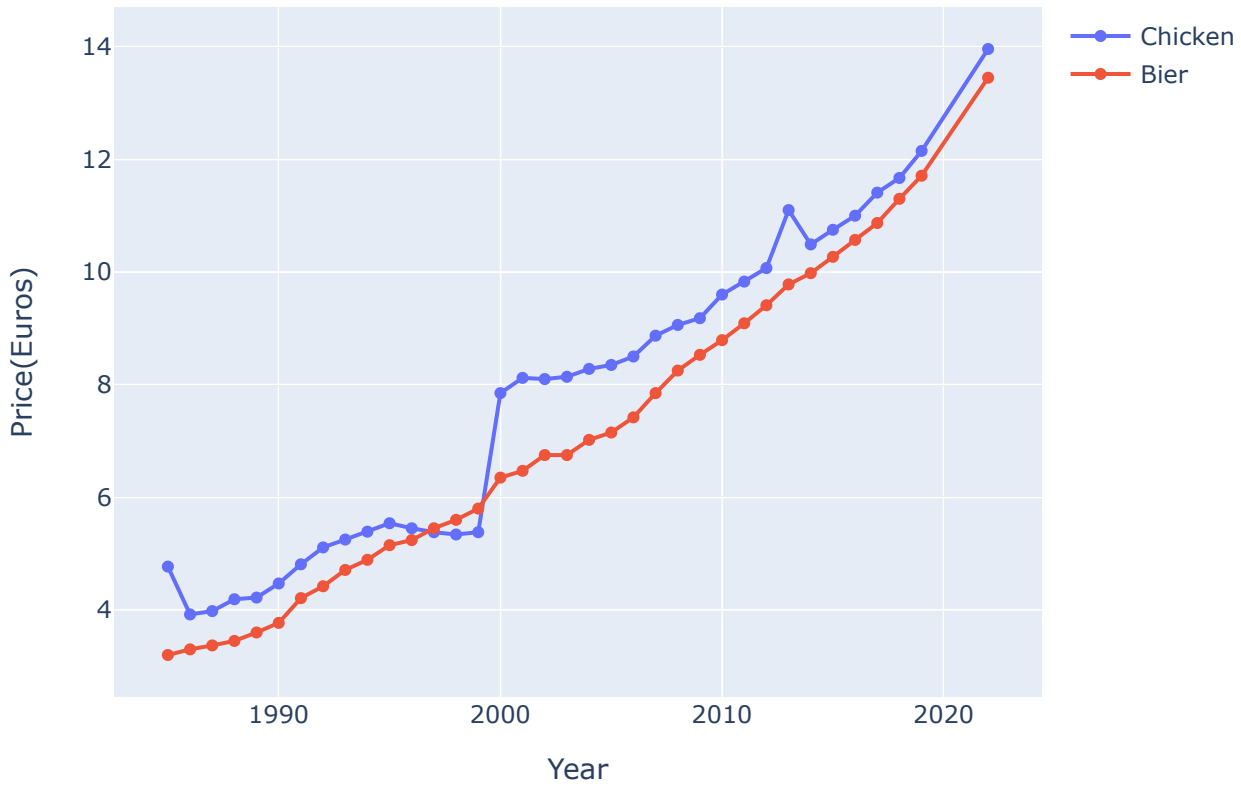
In [14]:
```
# Create a new figure with a specified size

plt.figure(figsize=(15, 8))

# Create a line plot for bier prices using Seaborn
```

```
sns.lineplot(data=df, x='year', y='bier_price', linewidth=3, label='Bier Price')

# Create a line plot for chicken price on the same plot
sns.lineplot(data=df, x='year', y='chicken_price', linewidth=3, label='Chicken Price')

# Set the title and axis labels

plt.title('Beer and Chicken Prices Over the Years', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Price (Euro)', fontsize=16)

# Display gridlines

plt.grid(True)

# Display a legend with labels for the two lines

plt.legend(fontsize=16)

# Show the plot

plt.show()
```



In [15]:

```
year = df['year']
chicken_price = df['chicken_price']
bier_price = df['bier_price']

fig = go.Figure()
fig.add_trace(go.Scatter(x=year, y=chicken_price,
                         mode='lines+markers',
                         name='Chicken'))
fig.add_trace(go.Scatter(x=year, y=bier_price,
                         mode='lines+markers',
                         name='Bier'))

fig.update_layout(title='Price Comparison between Chicken and Bier prices Over the years
                  xaxis_title='Year',
                  yaxis_title='Price(Euros)')
```



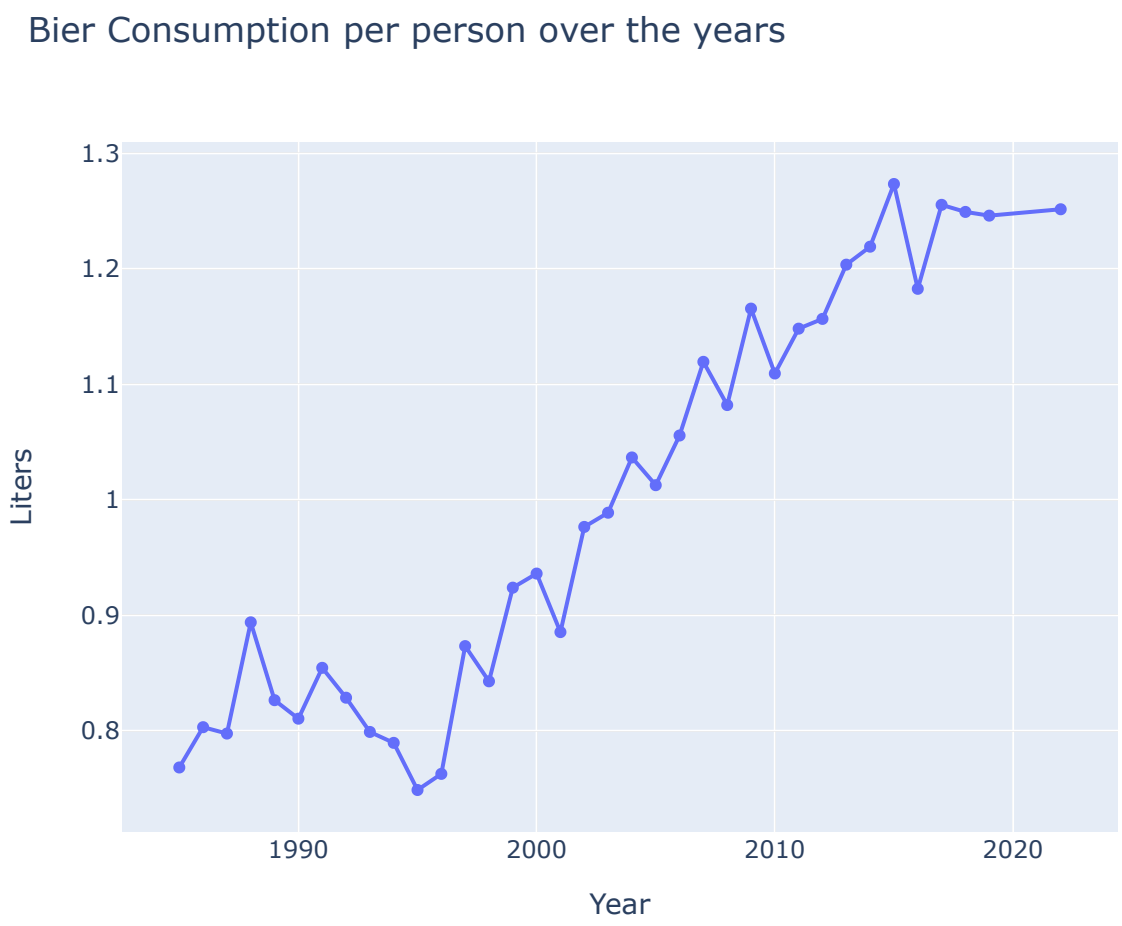
In [16]:
```
# Create a new column 'total_visitors_million' in the DataFrame 'df'
# This column represents the total number of visitors in millions

# Multiply the 'visitors_total' column by 1,000,000 to convert it to millions

df['total_visitors_million'] = df['visitors_total']*1_000_000

# Display the first few rows of the DataFrame after adding the new column

df.head()
```

Out[16]:

	year	duration	visitors_total	bier_price	bier_consump	chicken_price	chicken_comsump	total_visitors_million
0	1985	16	7.1	3.20	54541	4.77	629520	7100000.0
1	1986	16	6.7	3.30	53807	3.92	698137	6700000.0
2	1987	16	6.5	3.37	51842	3.98	732859	6500000.0
3	1988	16	5.7	3.45	50951	4.19	720139	5700000.0
4	1989	16	6.2	3.60	51241	4.22	775674	6200000.0

In [17]:
```
# Calculate the liters of beer per person by dividing 'bier_consump' by 'total_visitors_

year = df['year']
beer_per_person = df['bier_consump']*100 / df['total_visitors_million']

# Create a Plotly figure for the line plot

fig = go.Figure()
```

```
# Add a line and markers trace for beer consumption per person data

fig.add_trace(go.Scatter(x=year, y=beer_per_person,
                         mode='lines+markers',
                         name='bier/person'))

# Customize the layout of the plot

fig.update_layout(title='Bier Consumption per person over the years',
                  xaxis_title='Year',
                  yaxis_title='Liters')
```



In [18]:
```
# Calculate the correlation between beer consumption and beer price

beer_correlation = df['bier_consump'].corr(df['bier_price'])

# Calculate the correlation between chicken consumption and chicken price

chicken_correlation = df['chicken_comsump'].corr(df['chicken_price'])

# Print the calculated correlations

print(f"Correlation between Beer Consumption and Beer Price: {beer_correlation:.2f}")
print(f"Correlation between Chicken Consumption and Chicken Price: {chicken_correlation:
```

```
Correlation between Beer Consumption and Beer Price: 0.89
Correlation between Chicken Consumption and Chicken Price: -0.86
```

In [19]:
```
# Create subplots for beer and chicken correlations
# 1 row, 2 columns of subplots, figsize is set to (12, 4)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
```

```
# Create a scatter plot for beer consumption vs. beer price on the first subplot (ax1)

ax1.scatter(df['bier_consump'], df['bier_price'], alpha=0.5)
ax1.set_xlabel('Bier Consumption')
ax1.set_ylabel('Bier Price')
ax1.set_title('Bier Consumption vs. Bier Price')

# Create a scatter plot for chicken consumption vs. chicken price on the second subplot

ax2.scatter(df['chicken_comsump'], df['chicken_price'], alpha=0.5)
ax2.set_xlabel('Chicken Consumption')
ax2.set_ylabel('Chicken Price')
ax2.set_title('Chicken Consumption vs. Chicken Price')

# Adjust subplot layout to prevent overlap

plt.tight_layout()

# Show the subplots

plt.show()
```