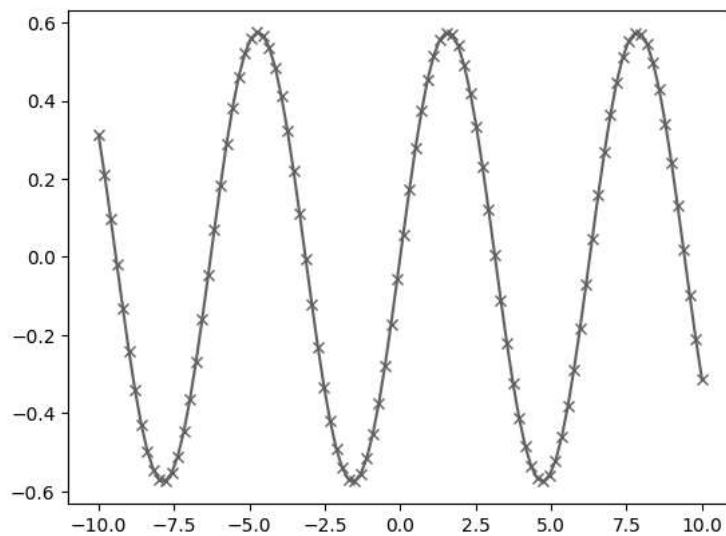


✓ Практическая работа №4-5

Визуализация данных средствами Matplotlib. Основы.

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
# Генерируем последовательность чисел от -10 до 10 с 100 шагами
x = np.linspace(-10, 10, 100)
# Генерируем случайную амплитуду для синусоиды
a = np.random.random()
# Создаем второй массив с помощью синуса
y = a*np.sin(x)
# Функция создает линейный график на основе двух массивов
plt.plot(x, y, marker="x")
```

↗ [matplotlib.lines.Line2D at 0x7a59d5cc6910]



Пробный запуск программы для построения 2D графиков и пример ее исполнения

```
import numpy as np
import pandas as pd

df_can = pd.read_excel('https://s3-api.us-gso.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx',
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2
)
print('Данные загружены и записаны в dataframe!')
```

↗ Данные загружены и записаны в dataframe!

```
df_can.head()
```


↗

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008	2009	20
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	2111	1746	17
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	560	716	5
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	4005	5393	47
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	0	0	

```
print(df_can.shape)
```


↗ (195, 43)

```
df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inplace=True)
df_can.head()
```



	OdName	AreaName	RegName	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	2006	2007	2008	2009	2010	2011
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	3009	2652	2111	1746	1758	2203
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	856	702	560	716	561	539
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3616	3626	4807	3623	4005	5393	4752	4325
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	0	1	0	0	0	0	0

```
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':'Region'}, inplace=True)
df_can.head()
```



	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	2006	2007	2008	2009	2010	2011
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	3009	2652	2111	1746	1758	2203
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	856	702	560	716	561	539
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3616	3626	4807	3623	4005	5393	4752	4325
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	0	1	0	0	0	0	0

```
all(isinstance(column, str) for column in df_can.columns)
```




False

```
df_can.columns = list(map(str, df_can.columns))
all(isinstance(column, str) for column in df_can.columns)
```




True

```
df_can.set_index('Country', inplace=True)
df_can.head()
```



	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2004	2005	2006	2007	2008	2009	2010
Country																		
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	2978	3436	3009	2652	2111	1746	1758
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1450	1223	856	702	560	716	561
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3616	3626	4807	3623	4005	5393	4752
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	0	1	0	0	0	0

```
df_can = df_can.apply(pd.to_numeric, errors='coerce')
df_can['Total'] = df_can.sum(axis=1)
df_can.head()
```



	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012
Country																			
Afghanistan	NaN	NaN	NaN	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758	2203	2635
Albania	NaN	NaN	NaN	1	0	0	0	0	0	1	...	1223	856	702	560	716	561	539	620
Algeria	NaN	NaN	NaN	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752	4325	3774
American Samoa	NaN	NaN	NaN	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	0
Andorra	NaN	NaN	NaN	0	0	0	0	0	0	2	...	0	1	1	0	0	0	0	1

5 rows × 38 columns

```
years = list(map(str, range(1980, 2014)))
df_can.sort_values(['Total'], ascending=False, axis=0, inplace=True)
df_top5 = df_can.head()
```

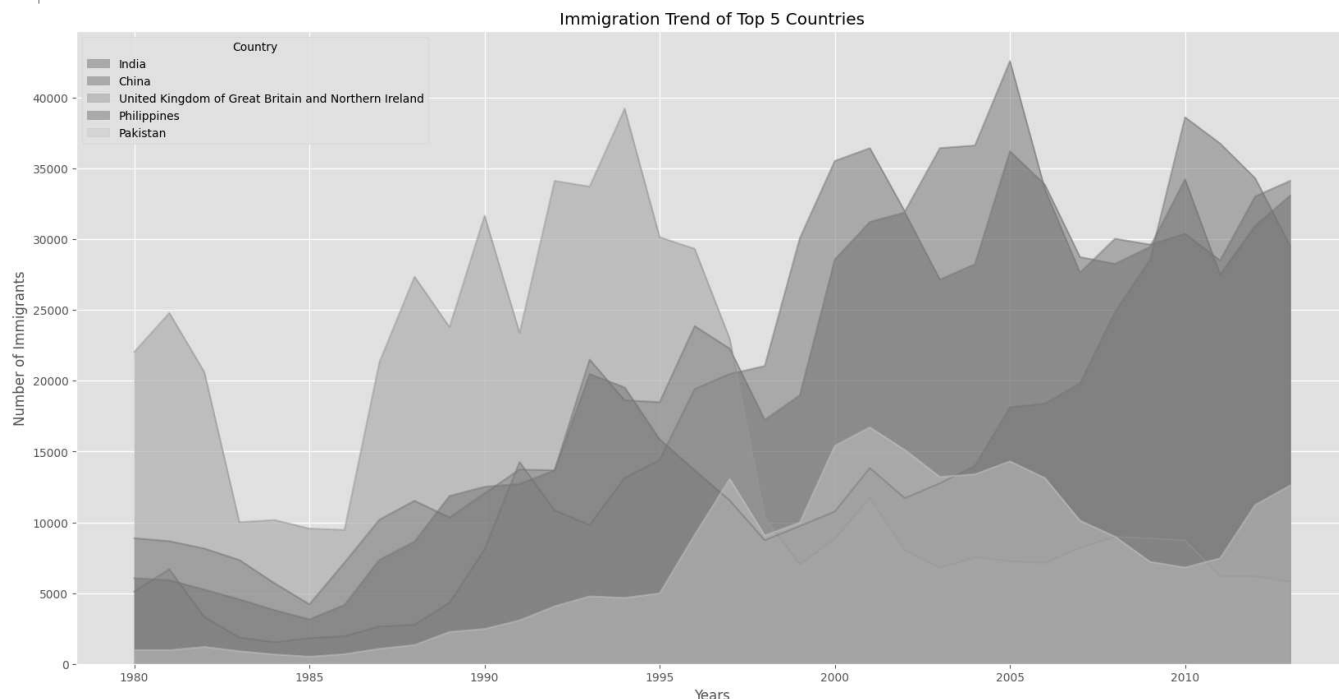
```
# Транспонирование таблицы
df_top5 = df_top5[years].transpose()
df_top5.head()
```

Country	India	China	United Kingdom of Great Britain and Northern Ireland	Philippines	Pakistan	
1980	8880	5123		22045	6051	978
1981	8670	6682		24796	5921	972
1982	8147	3308		20620	5249	1201
1983	7338	1863		10015	4562	900
1984	5704	1527		10170	3801	668

Далее: ☒ Посмотреть рекомендованные графики ☐ New interactive sheet

```
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.style.use('ggplot') # опционально: задаем стиль ggplot
# Проверяем версию Matplotlib
print ('Matplotlib version: ', mpl.__version__) # >= 2.0.0
# Для построения графика изменяем тип индексов строк (года)
# на integer
df_top5.index = df_top5.index.map(int)
# Построение графика типа 'area' встроенной
# в pandas суб-библиотекой matplotlib
df_top5.plot(kind='area',
             stacked=False,
             figsize=(20, 10), # размер области построения графика
             )
#Задаем наименование графика
plt.title('Immigration Trend of Top 5 Countries')
#Задаем наименование оси Y
plt.ylabel('Number of Immigrants')
#Задаем наименование оси X
plt.xlabel('Years')
# Выводим график со всеми параметрами на экран
plt.show()
```

Matplotlib version: 3.10.0



✓ Вывод по практической работе 4:

В процессе выполнения практической работы, столкнулся с проблемой добавления столбца Total, данная ошибка возникала из-за того, что при попытке вызова метода `sum`, в строке складывались типы данных строковый и целый. Результатом решения данной проблемы стало изменение изначального кода на `df_can = df_can.apply(pd.to_numeric, errors='coerce')` `df_can['Total'] = df_can.sum(axis=1)` `df_can.head()`, которая начинала суммировать все значения. Также было понято, что различия между `matplotlib` и `pandas` заключаются в первом выводит все данные данные, а `pandas` их обрабатывает, а затем выводит. Для вывода графика типа «Диаграмма с областями» в функции `plot` библиотеки `Pandas` необходимо задать параметр `kind` со значением `'area'`.

```
df_can['2013'].head()
```

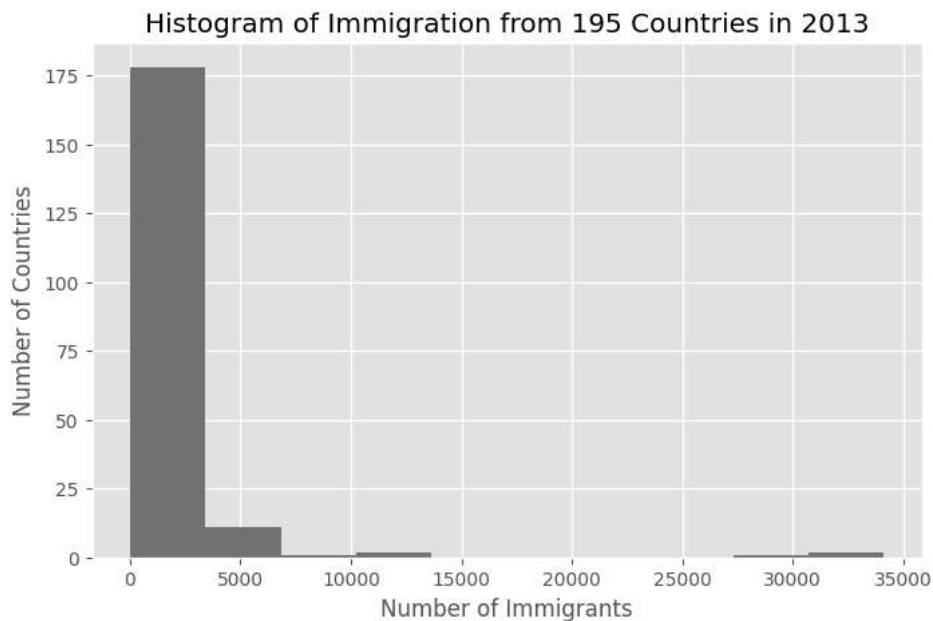


	2013
Country	
India	33087
China	34129
United Kingdom of Great Britain and Northern Ireland	5827
Philippines	29544
Pakistan	12603

dtype: int64

```
# np.histogram возвращает два значения
count, bin_edges = np.histogram(df_can['2013'])
print(count) # подсчет частоты появления данных
print(bin_edges) # количество столбцов, по умолчанию - 10
```

```
df_can['2013'].plot(kind='hist', figsize=(8, 5))
plt.title('Histogram of Immigration from 195 Countries in 2013') # добавление названия
plt.ylabel('Number of Countries') # добавление наименования оси y
plt.xlabel('Number of Immigrants') # наименование оси x
plt.show()
```



```
# step 1: get the data
df_iceland = df_can.loc['Iceland', years]
df_iceland.head()
```



Iceland	
1980	17.0
1981	33.0
1982	10.0
1983	9.0
1984	13.0

dtype: float64

```
# step 2: plot data
df_iceland.plot(kind='barh', figsize=(10, 6))
plt.xlabel('Year') # add to x-label to the plot
plt.ylabel('Number of immigrants') # add y-label to the plot
plt.title('Icelandic immigrants to Canada from 1980 to 2013') # add title to the plot
plt.show()
```

