SPRAWOZDANIE

Zajęcia: Eksploracja i wizualizacja danych
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 2
12.11.2021

Temat: Graficzna wizualizacja danych z użyciem bibliotek „matplotlib” i „seaborn"

Marek Bubiak
Informatyka II stopień,
niestacjonarne (zaoczne),
III semestr,
https://github.com/JeremyCoco/eiwd

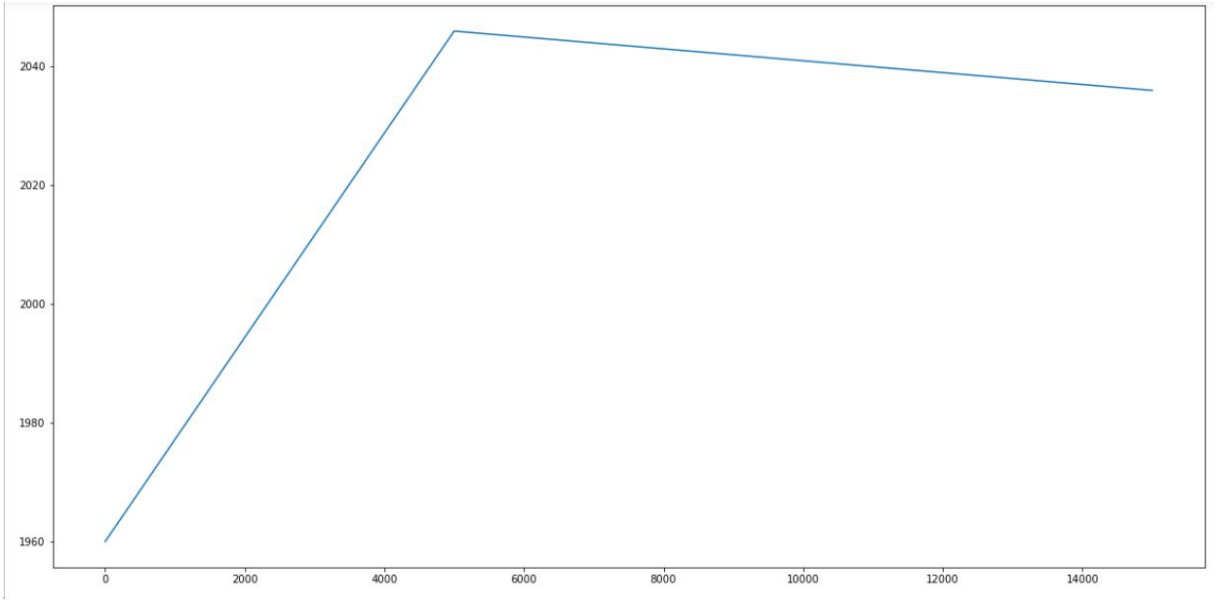
Na podstawie danych ze zbioru [2] lub danych losowych, wygenerować wykresy analogiczne do przedstawionych na zajęciach.

1. Wygenerować wykres liniowy

In [1]:
```
# załadowanie bibliotek

import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

import pandas as pd
```

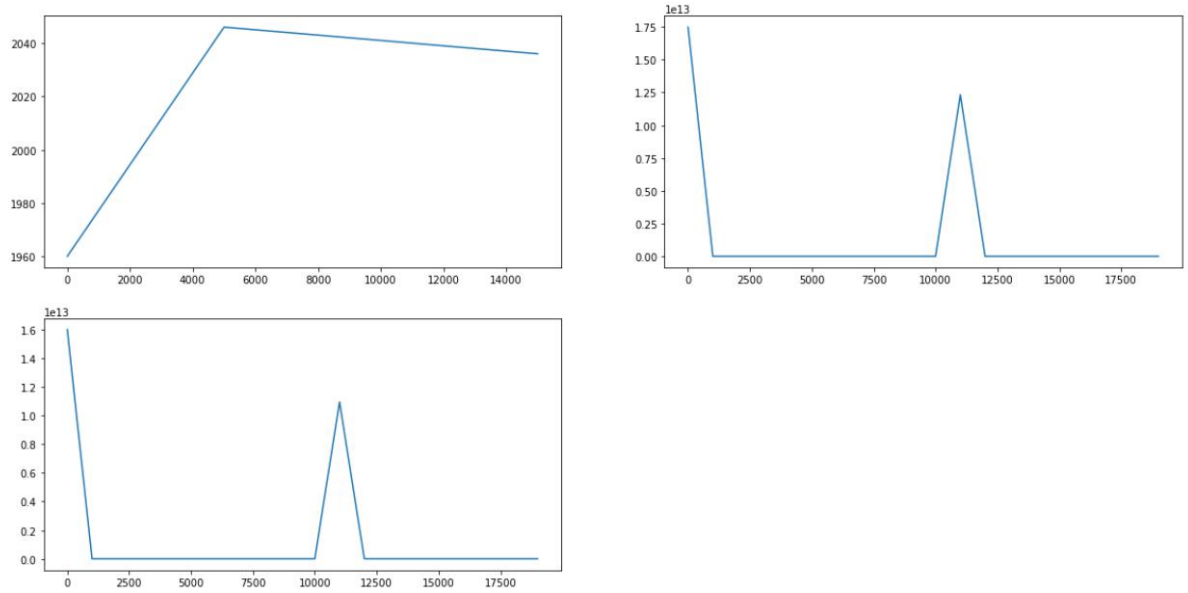
In [2]:
```
# załadowanie danych
import os

path = r"E:\studiaMGR\SEMESTR III\EiWD\2\IHME_GDP_1960_2050_CSV_1\IHME_GDP_1960_2050_Y2021M09D22.CSV"
folder = r"E:\studiaMGR\SEMESTR III\EiWD\2\wykresy"

data = pd.read_csv(path, low_memory=False)
data.head()
```

In [4]:
```
# wykres liniowy

years = data['year'][::5_000]

plt.rcParams["figure.figsize"] = (20,10)
plt.plot(years)
plt.savefig(os.path.join(folder, 'wykres1.svg'), dpi=400, bbox_inches='tight')
```



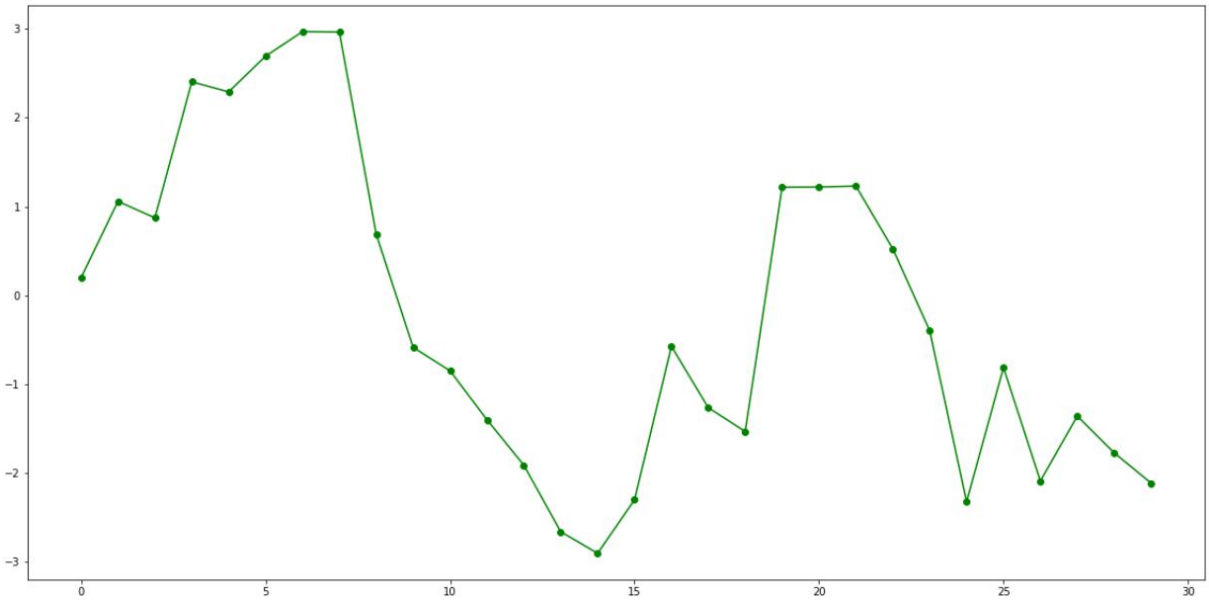
2. Wygenerować kilka wykresów obok siebie

In [5]:
```
# kilka podwykresów

fig = plt.figure()
ax0 = fig.add_subplot(2,2,1)
ax1 = fig.add_subplot(2,2,2)
ax2 = fig.add_subplot(2,2,3)

gdp_ppp_mean = data['gdp_ppp_mean'][::1_000]
gdp_ppp_lower = data['gdp_ppp_lower'][::1_000]

ax0.plot(years)
ax1.plot(gdp_ppp_mean)
ax2.plot(gdp_ppp_lower)

plt.savefig(os.path.join(folder, 'wykres2.svg'), dpi=400, bbox_inches='tight')
```



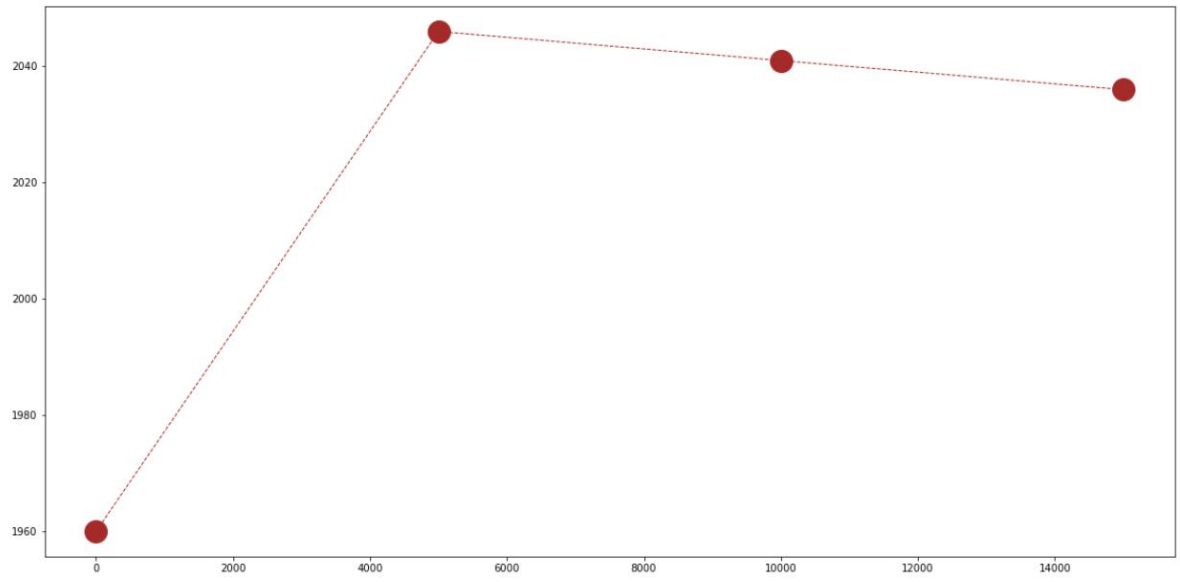
3. Wygenerować wykres liniowy i zdefiniować jego formatowanie

In [6]:
```
# wykres punktowy ze zdefiniowanymi kolorami, stylem znaczników i linii

from numpy.random import randn
plt.plot(randn(30).cumsum(), 'go-')
plt.savefig(os.path.join(folder, 'wykres3.svg'), dpi=400, bbox_inches='tight')
```



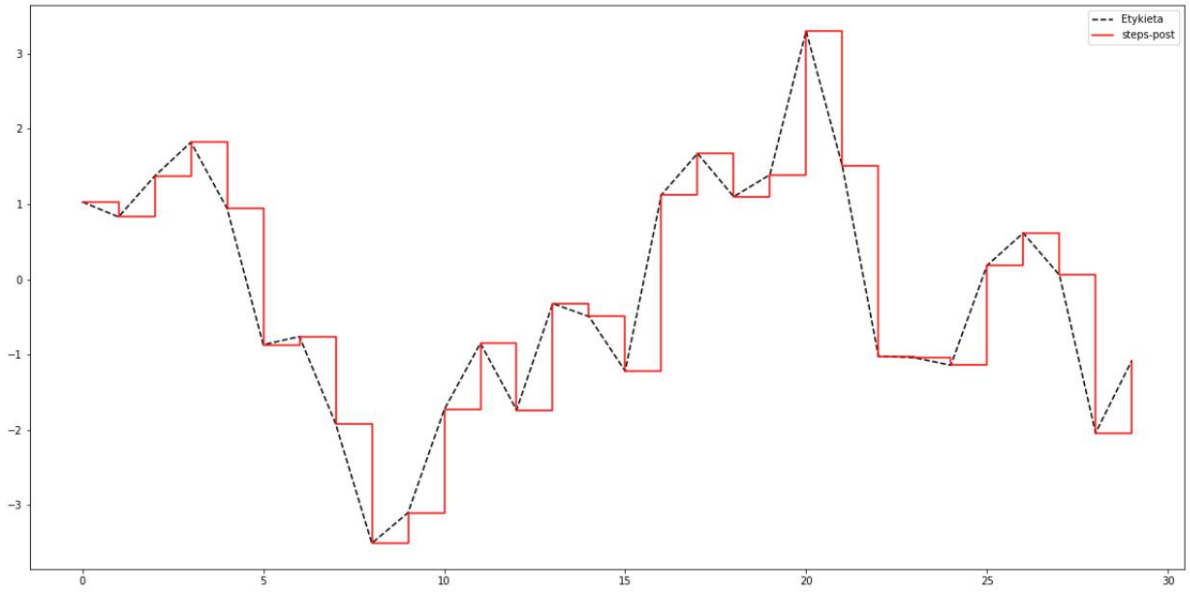
4. Wygenerować z dodanym wykresem interpolacji liniowej i legendą

```
In [7]: # wykres interpolacji liniowej

        data = np.random.randn(30).cumsum()
        plt.plot(data, 'k--', label='Etykieta')
        plt.plot(data, 'r-',  label="steps-post", drawstyle="steps-post")
        plt.legend(loc='best')

        # normalizacja
        # years_normal = years.div(years.sum())
        # elim_ch_normal = elim_ch.div(elim_ch.sum())
        # prelim_est_normal = prelim_est.div(prelim_est.sum())

        # plt.plot(years_normal, 'k--', label='years', drawstyle='steps-pre')
        # plt.plot(elim_ch_normal, 'g.', label='elim ch', drawstyle='steps-mid')
        # plt.plot(prelim_est_normal, 'r-', label='elim ch', drawstyle='steps-post')
        # plt.legend(loc='best')

        plt.savefig(os.path.join(folder, 'wykres5.png'), dpi=400, bbox_inches='tight')
```



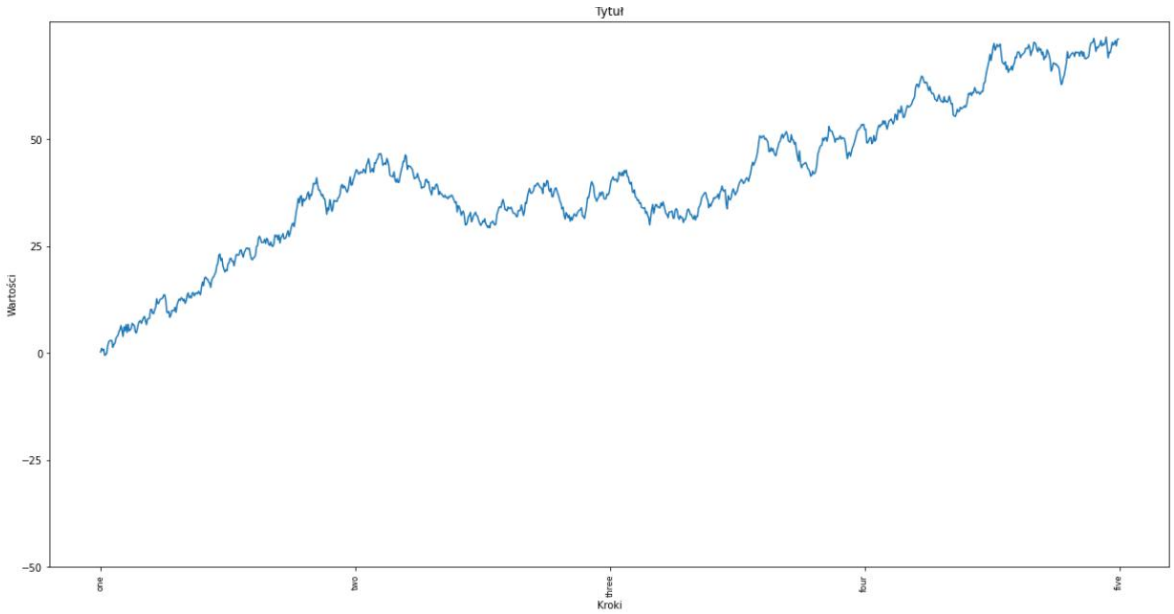
5. Wygenerować wykres liniowy i dodać do niego samodzielnie zdefiniowane etykiety osi

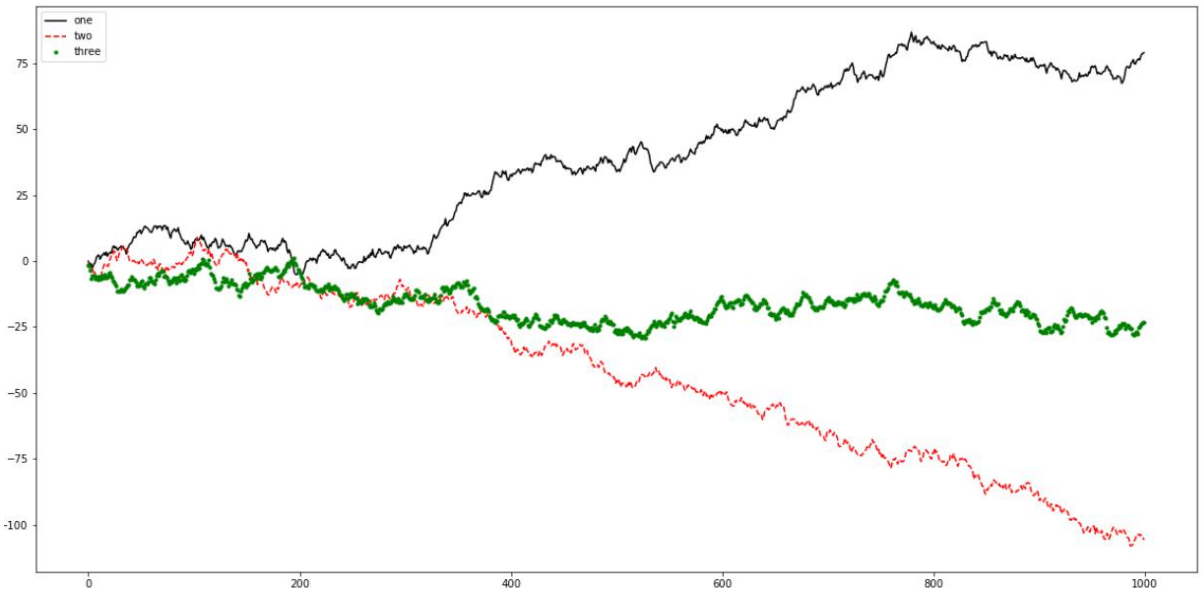
In [9]:
```
# dodanie etykiet na osi

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.plot(np.random.randn(1000).cumsum())

ax.set_xticks([0, 250, 500, 750, 1000])
ax.set_yticks([-50, -25, 0, 25, 50])

ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'], rotation=90, fontsize='small')

# dodanie właściwości wykresu słownikiem

props = {
    'title': "Tytuł",
    'xlabel': "Kroki",
    'ylabel': "Wartości"
}
ax.set(**props)

plt.savefig(os.path.join(folder, 'wykres6.svg'), dpi=400, bbox_inches='tight')
```



6. Wygenerować kilka wykresów liniowych w różnych kolorach i nałożyć je na siebie

In [10]:
```
fig = plt.figure();
ax = fig.add_subplot(1,1,1)
ax.plot(randn(1000).cumsum(), 'k', label='one')
ax.plot(randn(1000).cumsum(), 'r--', label='two')
ax.plot(randn(1000).cumsum(), 'g.', label='three')
ax.legend(loc='best')

plt.savefig(os.path.join(folder, 'wykres7.svg'), dpi=400, bbox_inches='tight')
```





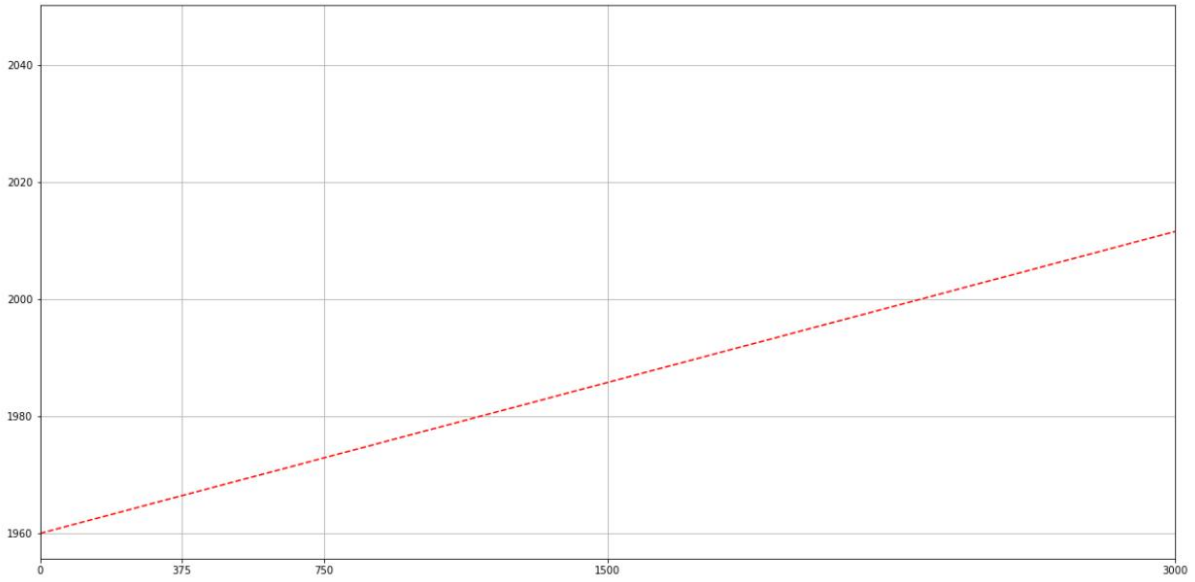
7. Zapisać wygenerowany wykres do pliku

```
In [15]: # zapis wykresu do pliku

         path = r"E:\studiaMGR\SEMESTR III\EiWD\2\wykresywykres01.svg"
         plt.savefig(path, dpi=400, bbox_inches='tight')
```

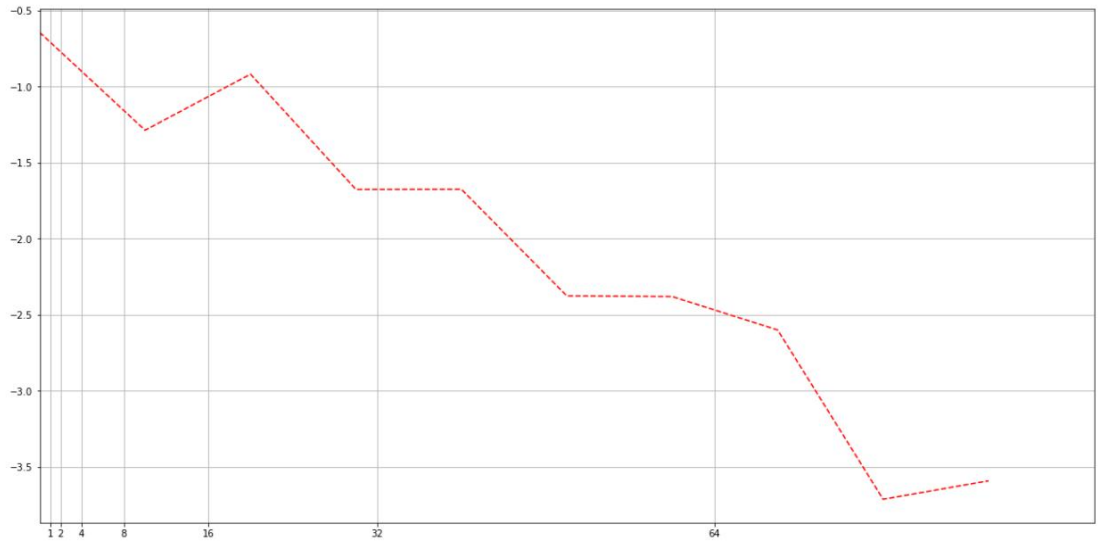
```
<Figure size 1440x720 with 0 Axes>
```

8. Wygenerować wykres liniowy z użyciem indeksu

In [21]:
```
import numpy as np
import pandas as pd

xmax = 3_000

s = pd.Series(years, index=np.arange(0, xmax*5_000, 5_000))
s.plot(kind='line', logy=False, xticks=[0, xmax/8, xmax/4, xmax/2, xmax], xlim=[0, xmax], grid=True, style='r--')

plt.savefig(os.path.join(folder, 'wykres8.svg'), dpi=400, bbox_inches='tight')
```

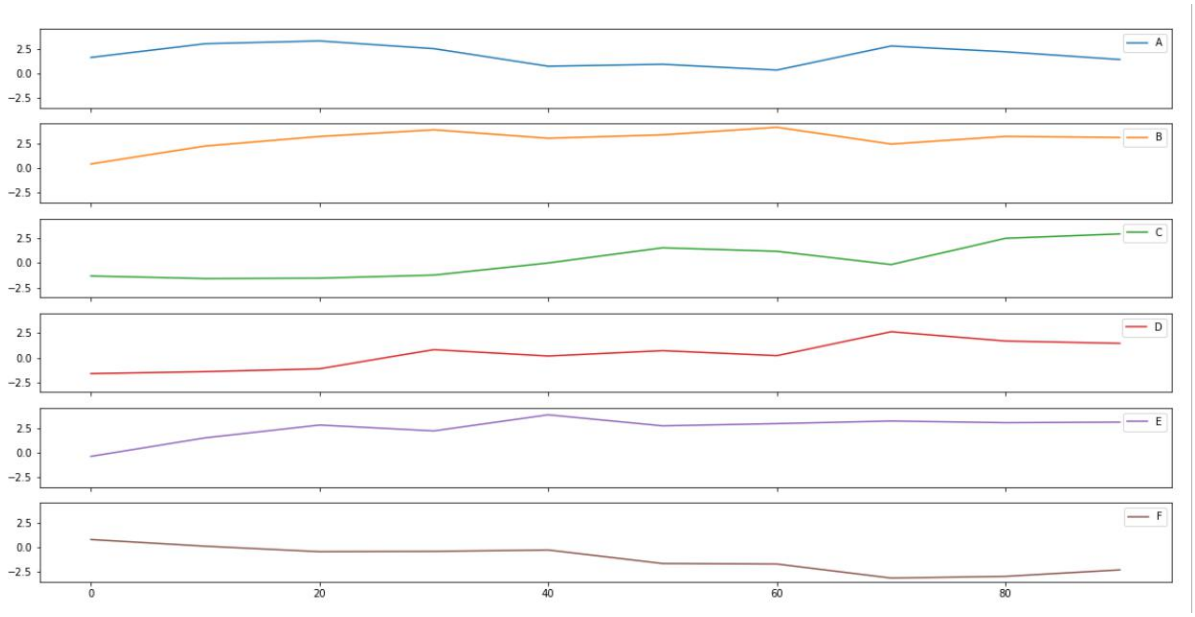


In [22]:
```
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
s.plot(kind='line', logy=False, xticks=[1, 2, 4, 8, 16, 32, 64, 128], xlim=[0, 100], grid=True, style='r--')

plt.savefig(os.path.join(folder, 'wykres8.svg'), dpi=400, bbox_inches='tight')
```



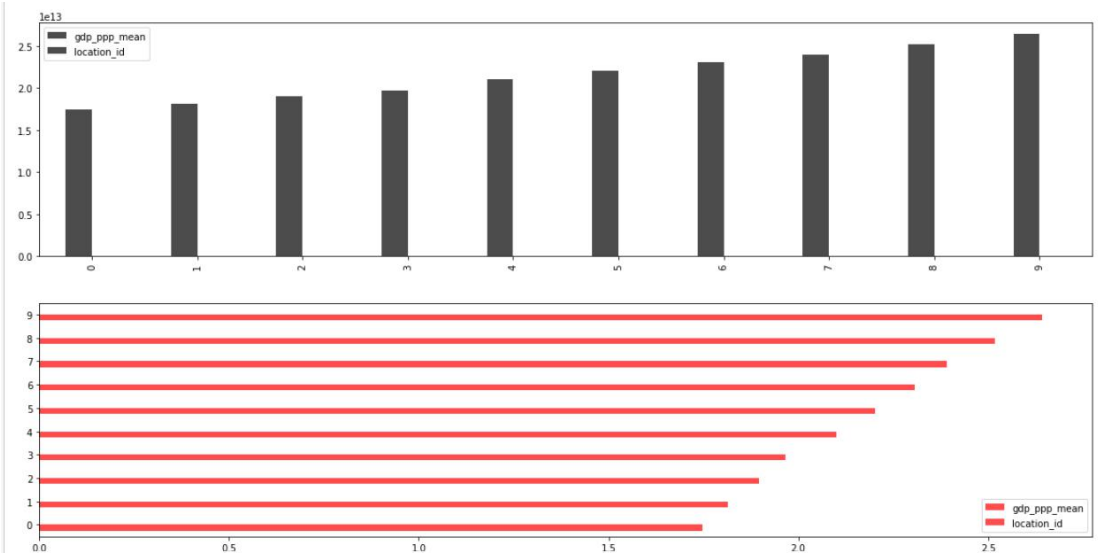
9. Wygenerować kilka wykresów liniowych ułożonych jeden pod drugim

```
In [23]: df = pd.DataFrame(np.random.randn(10, 6).cumsum(0), columns=['A', 'B', 'C', 'D', 'E', 'F'], index=np.arange(0, 100, 10))
         df.plot(subplots=True, sharex=True, sharey=True, title='Tytul wykresu', sort_columns=True)

         #df = num_data
         #df.plot(subplots=True, sharex=True, sharey=True, title='Wykresy wartości liczbowych', sort_columns=True)

         plt.savefig(os.path.join(folder, 'wykres9.png'), dpi=400, bbox_inches='tight')
```



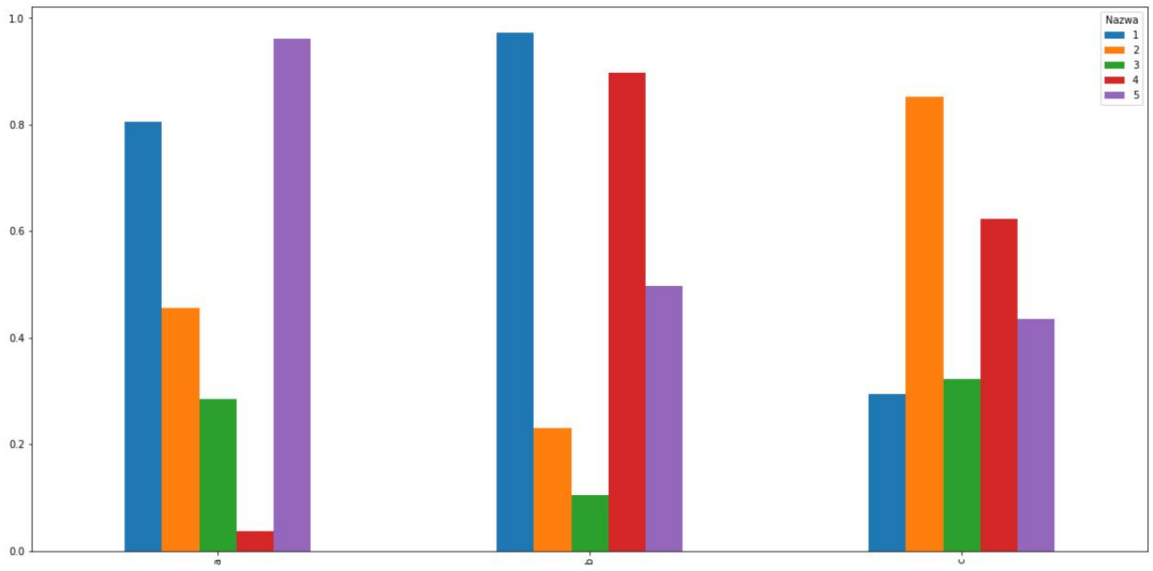
10. Wygenerować wykresy słupkowe ułożone poziomo i pionowo

```
In [26]: #data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
         data = pd.DataFrame(num_data[['location_id', 'gdp_ppp_mean']][:10], columns=['gdp_ppp_mean', 'location_id'])

         fig, axes = plt.subplots(2, 1)

         data.plot.bar(ax = axes[0], color='k', alpha=0.7)
         data.plot.barh(ax = axes[1], color='r', alpha=0.7)

         plt.savefig(os.path.join(folder, 'wykres10.png'), dpi=400, bbox_inches='tight')
```



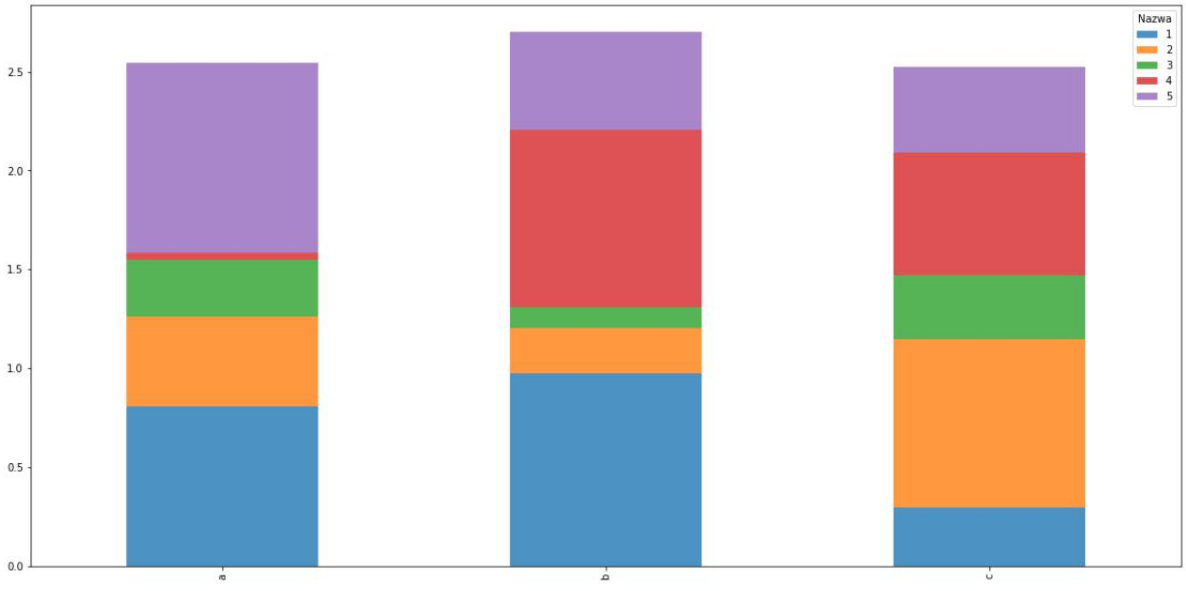
11. Wygenerować wykres słupkowy z pogrupowanymi danymi

```
In [27]: data = np.random.rand(3, 5)

         df = pd.DataFrame(data, index=['a', 'b', 'c'], columns=pd.Index(['1', '2', '3', '4', '5'], name='Nazwa'))
         df.plot.bar()

         plt.savefig(os.path.join(folder, 'wykres11.png'), dpi=400, bbox_inches='tight')
```



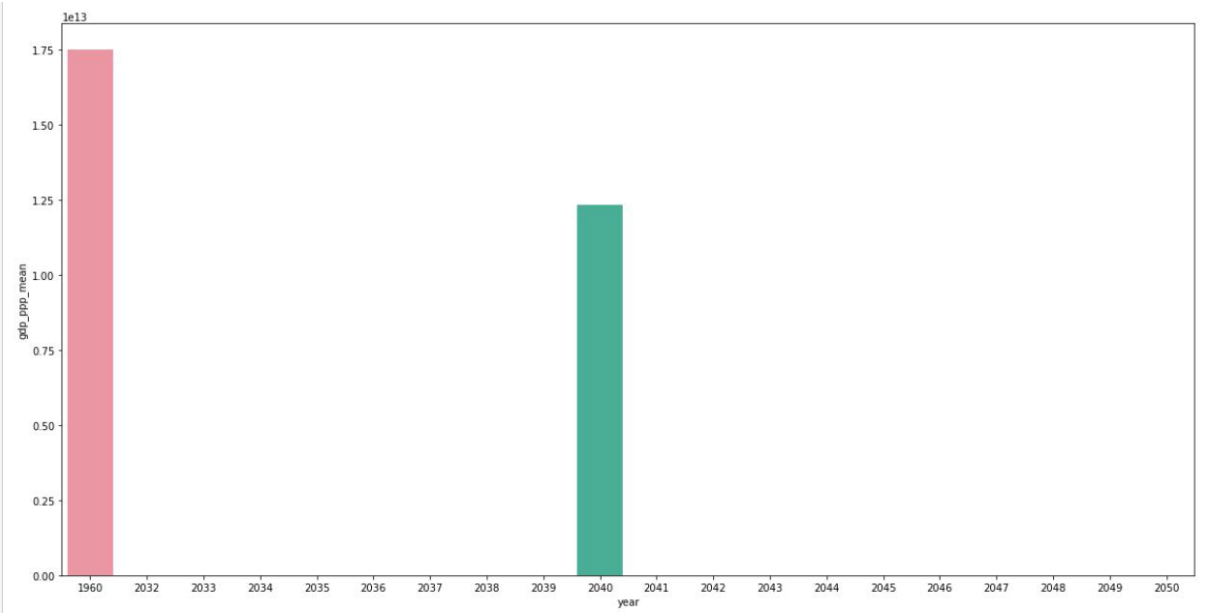
12. Wygenerować wykres słupkowy skumulowany

```
In [28]: df.plot.bar(stacked=True, alpha=0.8)

         plt.savefig(os.path.join(folder, 'wykres12.png'), dpi=400, bbox_inches='tight')
```



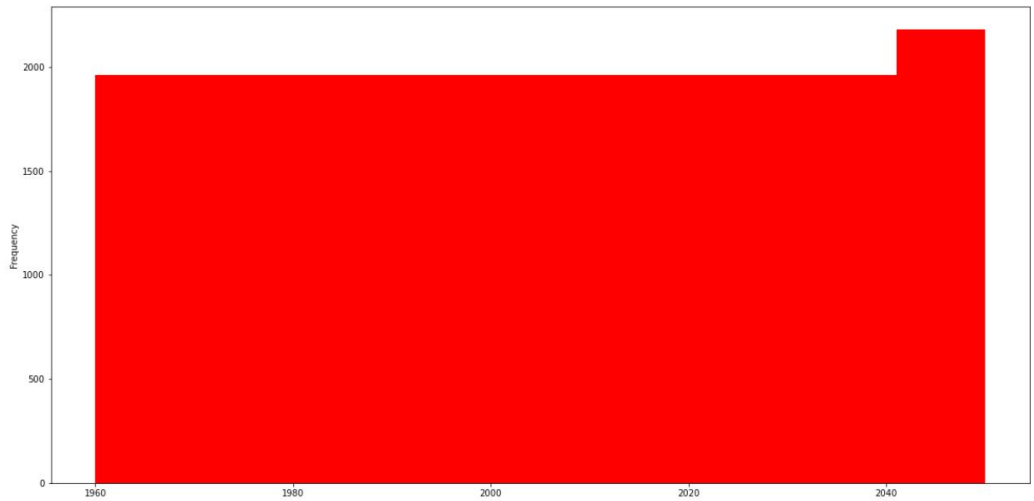
13. Wygenerować wykres słupkowy przy użyciu biblioteki seaborn

```
In [37]: #sns.barplot(x='tip_pct', y='day', data=tips, orient='h', hue='sex')
         import seaborn as sns

         max = 100_000
         per = 1_000
         sns.barplot(y=num_data.gdp_ppp_mean[:max:per], x = num_data.year[:max:per])

         plt.savefig(os.path.join(folder, 'wykres14.png'), dpi=400, bbox_inches='tight')
```



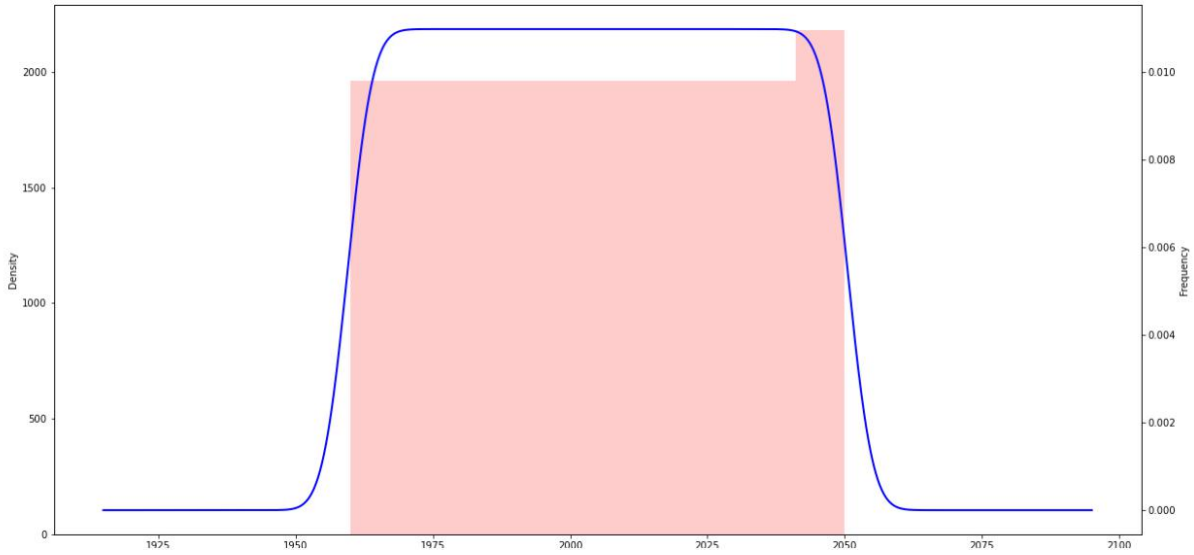
14. Wygenerować histogram

```
In [38]: # histogram
         #tips['tip_pct'].plot.hist(bins=50)
         num_data['year'].plot.hist(bins=10, color='red')

         plt.savefig(os.path.join(folder, 'wykres15.png'), dpi=400, bbox_inches='tight')
```



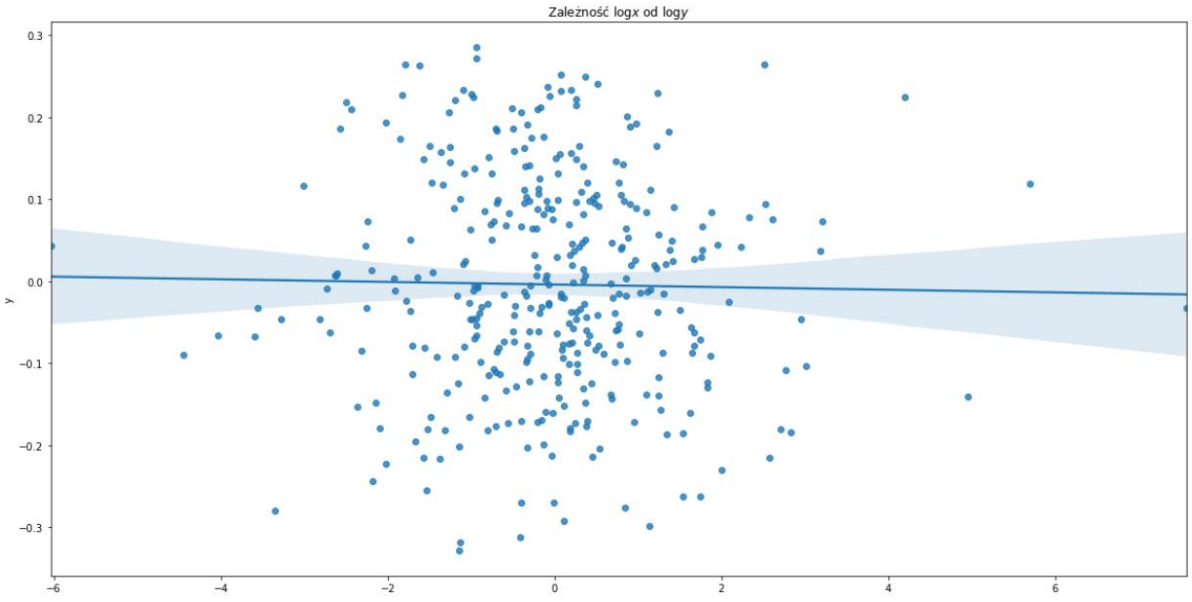
15. Wygenerować wykres gęstości prawdopodobieństwa i nałożyć go na histogram

```
In [39]: # wykres gęstości

         #tips['tip_pct'].plot.kde()
         num_data['year'].plot.hist(bins=10, color='red', alpha=0.2)
         ax = num_data['year'].plot.kde(color='blue', linewidth=2.0, secondary_y=True)
         ax.set_ylabel("Frequency", fontsize=10)

         plt.savefig(os.path.join(folder, 'wykres16.png'), dpi=400, bbox_inches='tight')
```



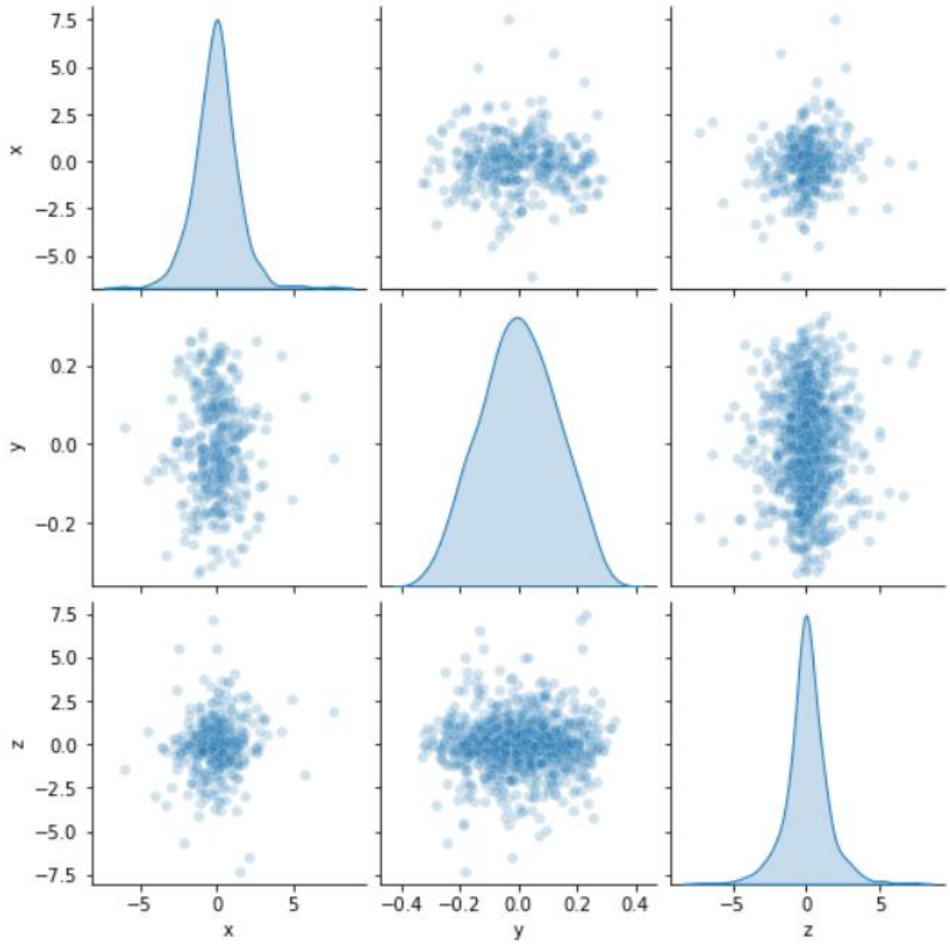
16. Wygenerować wykres regresji liniowej

```
In [41]: # wykres regresji liniowej

         # losowe dane
         n = 1_000
         data = pd.DataFrame({
             'x': np.random.rand(n)*5-2,
             'y': np.random.rand(n)*2+5,
             'z': np.random.rand(n)
         })
         # różnica logarytmów
         data = np.log(data).diff()

         sns.regplot(x='x', y='y', data=data)
         plt.title('Zależność $\log{x}$ od $\log{y}$')
```



17. Wygenerować wszystkie możliwe wykresy rozrzutu oraz wykresy gęstości

```
# wszystkie wykresy rozrzutu

#sns.pairplot(trans_data, diag_kind='kde', plot_kws={'alpha': 0.2})
sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.2})
```



18. Wygenerować wykres pudełkowy

```
import math

data['year'] = data.year.apply(lambda x: 0 if x=='-' else math.log(abs(int(x)+1),10))

sns.catplot(x='location_name', y='year', kind='box', data=data)
```