1. feladat: Adatok beolvasása és alapvető elemzés

1. Töltsd le egy CSV fájlt tartalmazó valós adatkészletet (például városok nevei, koordináták, népesség, vagy bármilyen földrajzi információ).
2. Használj **Pandas-t**, hogy:
    - Olvasd be a CSV fájlt.
    - Listázd ki az első 5 sort.
    - Nézd meg, milyen típusúak az oszlopok.
    - Számítsd ki, hány egyedi város van az adathalmazban.

```
import pandas as pd

# CSV fájl beolvasása
df = pd.read_csv('cities.csv')  # Példa fájl
print(df.head())  # Az első 5 sor
print(df.info())  # Oszloptípusok
print("Egyedi városok száma:", df['city'].nunique())
```

2. feladat: Adatok szűrése és aggregálása

1. Szűrd le azokat a városokat, ahol a népesség meghaladja az 1 milliót.
2. Számítsd ki az átlagos népességet ezekben a városokban.

```
# Szűrés
filtered_df = df[df['population'] > 1_000_000]
print(filtered_df)

# Átlagos népesség
average_population = filtered_df['population'].mean()
print("Átlagos népesség:", average_population)
```

3. feladat: Térképek készítése Folium-mal

1. Hozz létre egy térképet a városok koordinátáival.
2. Adj hozzá jelölőket a városokhoz, ahol a népesség meghaladja az 1 milliót. A jelölőknél tüntesd fel a város nevét és népességét.

```python
import folium

# Alaptérkép létrehozása
m = folium.Map(location=[47.1625, 19.5033], zoom_start=6)  # Magyarország közép

# Jelölők hozzáadása
for _, row in filtered_df.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=f"{row['city']} - Népesség: {row['population']}",
    ).add_to(m)

# Térkép mentése
m.save('cities_map.html')
```

4. feladat: Klaszterezett térkép

1. Használj Folium MarkerCluster-t, hogy csoportosított jelölőket készíts a térképen.
2. Az összes város szerepeljen rajta, nemcsak a nagy népességűek.

```python
from folium.plugins import MarkerCluster

# Klaszter létrehozása
marker_cluster = MarkerCluster().add_to(m)

# Összes város hozzáadása
for _, row in df.iterrows():
    folium.Marker(
        location=[row['latitude'], row['longitude']],
        popup=f"{row['city']} - Népesség: {row['population']}",
    ).add_to(marker_cluster)

# Klaszteres térkép mentése
m.save('clustered_map.html')
```

5. feladat: Hőtérkép készítése

1. Készíts hőtérképet a városok földrajzi elhelyezkedéséről, ahol a népesség számít a súlynak.
2. Használj Folium HeatMap-et.

```
from folium.plugins import HeatMap

# Hőtérkép adatok előkészítése
heat_data = [[row['latitude'], row['longitude'], row['population']] for _, row in df.iterrows()]

# Hőtérkép létrehozása
HeatMap(heat_data).add_to(m)

# Hőtérkép mentése
m.save('heatmap.html')
```

Adatfájl szükség esetén

Ha szükséged van példafájlra, létrehozhatod ezt egy egyszerű CSV-ben:

cities.csv:

```
city,latitude,longitude,population
Budapest,47.4979,19.0402,1752286
Debrecen,47.5316,21.6273,202402
Szeged,46.253,20.1414,160766
Pécs,46.0727,18.2323,145347
Győr,47.6875,17.6504,131267
```

6. feladat: Hiányzó értékek kezelése

1. Töltsd ki az oszlopok hiányzó értékeit:
    - Ha numerikus adat, töltsd fel az oszlop átlagával.
    - Ha szöveges adat, töltsd fel az oszlop leggyakoribb értékével.
2. Nézd meg, van-e még hiányzó érték az adathalmazban.

```python
# Hiányzó értékek keresése
print(df.isnull().sum())

# Numerikus oszlopok kitöltése az átlaggal
for col in df.select_dtypes(include='number').columns:
    df[col].fillna(df[col].mean(), inplace=True)

# Szöveges oszlopok kitöltése a leggyakoribb értékkel
for col in df.select_dtypes(include='object').columns:
    df[col].fillna(df[col].mode()[0], inplace=True)

# Ellenőrzés
print(df.isnull().sum())
```

7. feladat: Új oszlop létrehozása

1. Hozz létre egy új oszlopot, amely kiszámítja a népsűrűséget (népesség/terület).
2. Tedd ezt az oszlopot logaritmikus skálára.

```python
import numpy as np

# Példa: terület oszlop hozzáadása (km²-ben)
df['area'] = [525, 461, 280, 162, 174]  # Példa területek

# Népsűrűség kiszámítása
df['density'] = df['population'] / df['area']

# Logaritmikus skála
df['log_density'] = np.log(df['density'])

print(df[['city', 'density', 'log_density']])
```

8. feladat: Adatok rendezése

1. Rendezd az adathalmazt:
    - Először népesség szerint csökkenő sorrendbe.
    - Majd azonos népességnél az „area” oszlop szerint növekvő sorrendbe.
2. Listázd ki az első 10 sort.

```
# Rendezés népesség szerint, majd terület szerint
sorted_df = df.sort_values(by=['population', 'area'], ascending=[False, True])
print(sorted_df.head(10))
```

9. feladat: Szűrés feltételek alapján

1. Szűrd ki azokat a városokat, ahol:
   - A népesség nagyobb, mint 200 ezer.
   - És a népsűrűség 500 fő/km² alatt van.
2. Listázd ki a város nevét és népsűrűségét.

```
filtered_df = df[(df['population'] > 200_000) & (df['density'] < 500)]
print(filtered_df[['city', 'density']])
```

10. feladat: Csoportosítás és aggregálás

1. Kategorizáld a városokat népesség szerint:
   - Kisváros: < 100 ezer
   - Közepes város: 100–500 ezer
   - Nagyváros: > 500 ezer
2. Számítsd ki az átlagos népességet és népsűrűséget minden kategóriában.

```
# Kategória létrehozása
df['city_category'] = pd.cut(df['population'], bins=[0, 100_000, 500_000, np.inf],
labels=['Kisváros', 'Közepes város', 'Nagyváros'])

# Aggregálás
grouped = df.groupby('city_category').agg({'population': 'mean', 'density': 'mean'})
print(grouped)
```

11. feladat: Pivot tábla készítése

1. Készíts egy pivot táblát, amely megmutatja az átlagos népességet és népsűrűséget minden várostípusra (`city_category`).
2. Állítsd be a „city_category” oszlopot indexnek.

```
pivot_table = pd.pivot_table(df, values=['population', 'density'], index='city_category',
aggfunc='mean')
print(pivot_table)
```

12. feladat: Idő alapú elemzés

1. Tegyük fel, hogy van egy oszlop, amely a város alapítási dátumát tartalmazza (pl. „year_founded”).
    - Számítsd ki, hogy hány évesek a városok.
2. Vizsgáld meg, hogy az idősebb városoknak nagyobb-e az átlagos népsűrűsége.

```
from datetime import datetime

# Alapítási dátum (példa adatok)
df['year_founded'] = [1873, 1361, 1183, 1009, 1271]

# Életkor kiszámítása
current_year = datetime.now().year
df['age'] = current_year - df['year_founded']

# Összefüggés vizsgálata
age_density_corr = df[['age', 'density']].corr().iloc[0, 1]
print(f"Kor és népsűrűség korrelációja: {age_density_corr}")
```

13. feladat: Adatok exportálása

1. Exportáld az adathalmazt egy új CSV fájlba.
2. Győződj meg arról, hogy a kimeneti fájl tartalmazza az új oszlopokat is (népsűrűség, log_density, kor stb.).

```
df.to_csv('processed_cities.csv', index=False)
print("Adatok exportálva a 'processed_cities.csv' fájlba.")
```

14. feladat: Vizualizáció Pandas-szal

1. Készíts egy oszlopdiagramot a városok népességének megjelenítésére.
2. Ábrázold a népsűrűséget histogram formájában.

```python
import matplotlib.pyplot as plt

# Oszlopdiagram
df.set_index('city')['population'].plot(kind='bar', title='Városok népessége')
plt.ylabel('Népesség')
plt.show()

# Hisztogram
df['density'].plot(kind='hist', bins=10, title='Népsűrűség eloszlása')
plt.xlabel('Népsűrűség')
plt.show()
```

Az alábbiakban egy egyszerű, valósághű adatfájl található, amelyet letölthetsz vagy a Python-ban generálhatsz. Ez a fájl alkalmas az összes előző feladat elvégzésére. Az adatok földrajzi helyeket, népességet és területet tartalmaznak.

city,latitude,longitude,population,area
Budapest,47.4979,19.0402,1752286,525
Debrecen,47.5316,21.6273,202402,461 Szeged,46.253,20.1414,160766,280
Pécs,46.0727,18.2323,145347,162 Győr,47.6875,17.6504,131267,174
Miskolc,48.1031,20.7784,151025,236
Nyíregyháza,47.9495,21.7247,119746,274
Kecskemét,46.8964,19.6897,110687,321
Székesfehérvár,47.186,18.4221,96170,170 Eger,47.9025,20.3772,53696,92
Veszprém,47.0928,17.9093,58763,126 Tatabánya,47.5883,18.3973,66124,91
Salgótarján,48.1042,19.8033,32133,102 Kaposvár,46.3597,17.7952,60300,113

Hogyan generáld az adatfájlt Python-ban?

Használhatod az alábbi kódot, hogy létrehozd a fájlt a gépeden:

```
data = {
    "city": [
        "Budapest", "Debrecen", "Szeged", "Pécs", "Győr",
        "Miskolc", "Nyíregyháza", "Kecskemét", "Székesfehérvár",
        "Eger", "Veszprém", "Tatabánya", "Salgótarján", "Kaposvár"
    ],
    "latitude": [
        47.4979, 47.5316, 46.253, 46.0727, 47.6875,
        48.1031, 47.9495, 46.8964, 47.186,
        47.9025, 47.0928, 47.5883, 48.1042, 46.3597
    ],
    "longitude": [
        19.0402, 21.6273, 20.1414, 18.2323, 17.6504,
        20.7784, 21.7247, 19.6897, 18.4221,
        20.3772, 17.9093, 18.3973, 19.8033, 17.7952
    ],
    "population": [
        1752286, 202402, 160766, 145347, 131267,
        151025, 119746, 110687, 96170,
        53696, 58763, 66124, 32133, 60300
    ],
    "area": [
        525, 461, 280, 162, 174,
        236, 274, 321, 170,
        92, 126, 91, 102, 113
    ]
}
```

```
# DataFrame létrehozása
df = pd.DataFrame(data)

# CSV mentése
df.to_csv("cities.csv", index=False)

print("Az adatfájl létrejött: 'cities.csv'")
```

Jó munkát!