12. Előadás

Python kurzus

Tárgyfelelős:
Dr. Tejfel Máté

Előadó:
Dr. Király Roland

10. Előadás tematikája

Adatvizualizáció

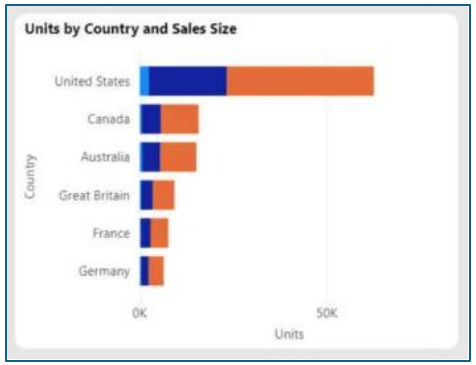
1. API és WEB – Portok (nyitott portok)
2. Matplotlib alapjai: diagramok készítése
3. Adatok megjelenítése - Pandas
4. Komplex diagramok létrehozása
5. Seaborn bevezetése és használata
6. Diagramok készítése különböző példák alapján

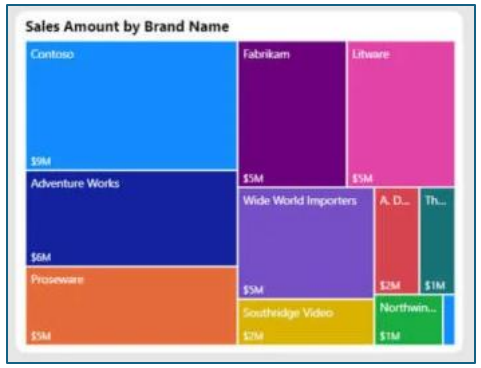
Adatvizualizáció bevezetés

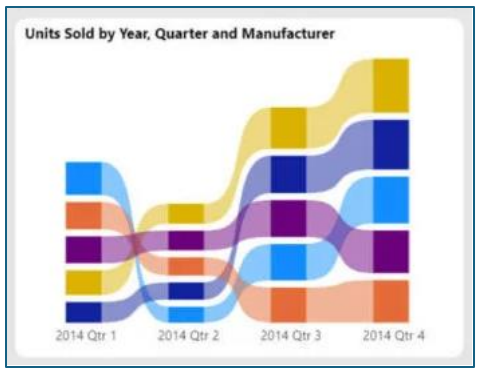
- Az adatvizualizáció segíti az adatok megértését és kommunikálását
- Kiemelt szerepe van a döntéshozatal támogatásában és a mintázatok felismerésében

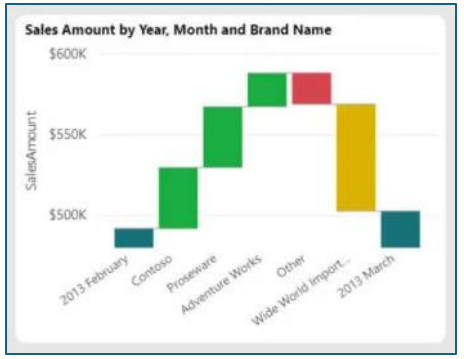
Miért fontos az adatvizualizáció?

- Komplex adatok egyszerűsítése – átláthatóság és értelmezhetőség
- Mintázatok felismerése – trendek, anomáliák azonosítása
- Döntéshozatal támogatása – vizuális információk közvetlen hatása
- Hatékony kommunikáció – információk egyszerű és érthető közvetítése
- Adatok feltárása – összefüggések és ok-okozati viszonyok felfedezése









https://digitallwork.hu/blog/a-power-bi-jelentese/

Alapvető diagram típusok

- Trendek figyelése, időbeli változások bemutatása – vonaldiagram
- Adatok eloszlása – hisztogram, eloszlásgrafikon
- Kapcsolatok azonosítása – scatter plot
- Kiugró értékek felismerése – box plot, scatter plot
- Adatok csoportos különbségei – oszlopdiagram, sávdiagram
- Összegzés és részek megoszlása – kördiagram, halmozott diagram

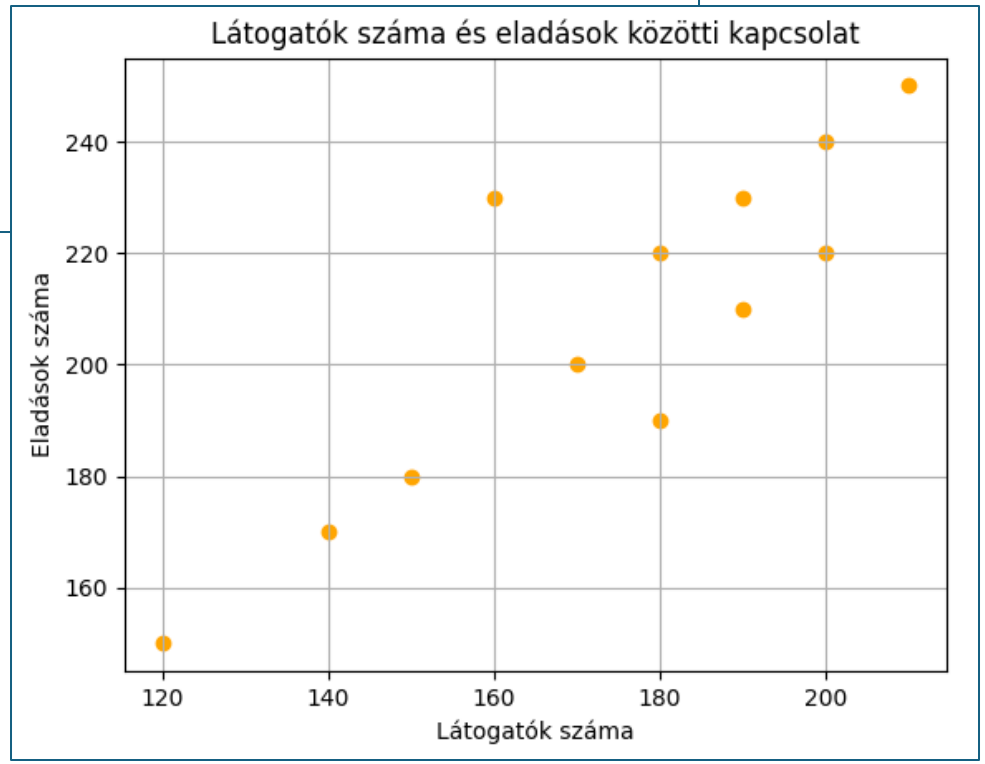
https://thenewstack.io/7-best-practices-for-data-visualization/

1. A Matplotlib alapjai: diagramok készítése

- A Matplotlib egy Python könyvtár, az adatvizualizáció egyik eszköztára
- Vonaldiagramok, hisztogramok, scatter plotok és más grafikonok készítésére használjuk
- Jól integrálható más adatfeldolgozó könyvtárakkal, mint a Pandas és a NumPy
- Telepítés: **pip(3) install matplotlib**

Scatter plotok – a korrelációk vizsgálatához

```
import matplotlib.pyplot as plt

latogatok = [120, 150, 180, 200, 170, 160, 140, 180, 190, 210, 190, 200]
eladasok = [150, 180, 220, 240, 200, 230, 170, 190, 210, 250, 230, 220]

plt.scatter(latogatok, eladasok, color='orange')
plt.title('Látogatók száma és eladások közötti kapcsolat')
plt.xlabel('Látogatók száma')
plt.ylabel('Eladások száma')
plt.grid(True)
plt.show()
```

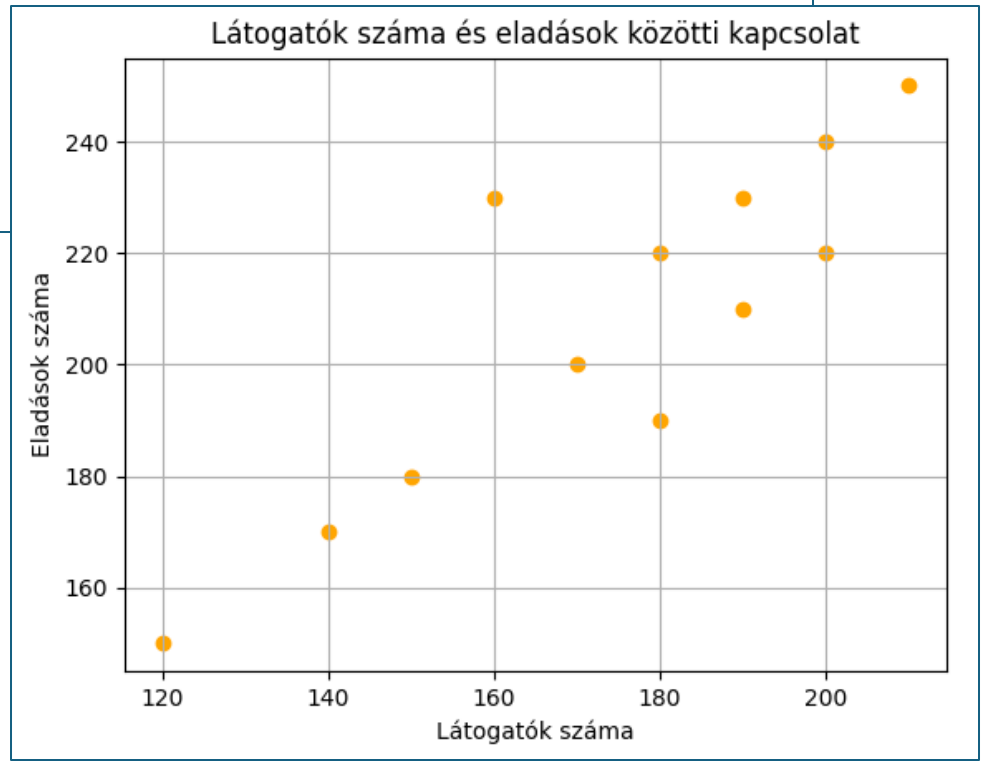


A **plt.scatter()** hozza létre a diagramot.

A **plt.grid(True)** jeleníti meg a rajzterület vezetőrácsait.

Scatter plot diagram rajzolása a plt.plot() függvénnyel

```
import matplotlib.pyplot as plt

latogatok = [120, 150, 180, 200, 170, 160, 140, 180, 190, 210, 190, 200]
eladasok = [150, 180, 220, 240, 200, 230, 170, 190, 210, 250, 230, 220]

plt.plot(latogatok, eladasok, color='orange', marker='o', linestyle='None')
plt.title('Látogatók száma és eladások közötti kapcsolat')
plt.xlabel('Látogatók száma')
plt.ylabel('Eladások száma')
plt.grid(True)
plt.show()
```



A **plt.plot()** függvény **linestyle='None'** beállítása hozza létre a scatter diagramot.

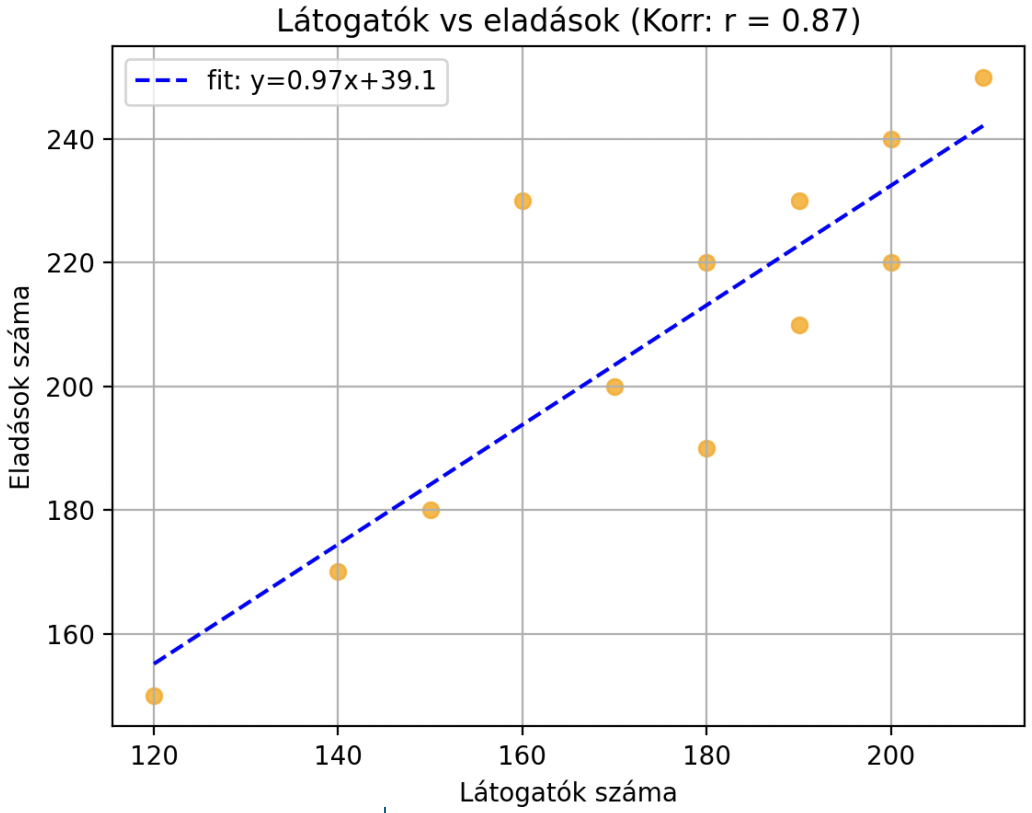
A **marker** argumentum a pontok jelölésére szolgál.

(scatter.py)

Hozzá lehet adni egy illeszkedő egyenest és kiszámolni a korrelációt.

```
import numpy as np
import matplotlib.pyplot as plt

latogatok = [120, 150, 180, 200, 170, 160, 140, 180, 190, 210, 190, 200]
eladasok   = [150, 180, 220, 240, 200, 230, 170, 190, 210, 250, 230, 220]

plt.scatter(latogatok, eladasok, color='orange', alpha=0.8)
m, b = np.polyfit(latogatok, eladasok, 1)  # lineáris illesztés
xs = np.array(sorted(latogatok))
plt.plot(xs, m*xs + b,
color='blue', linestyle='--', label=f'fit: y={m:.2f}x+{b:.1f}')

r = np.corrcoef(latogatok, eladasok)[0,1]  # korreláció
plt.title(f'Látogatók vs eladások (Pearson r = {r:.2f})')
plt.xlabel('Látogatók száma')
plt.ylabel('Eladások száma')
plt.grid(True)
plt.legend()
plt.show()
```

(koorelacio1.py)



Vonaldiagram készítése

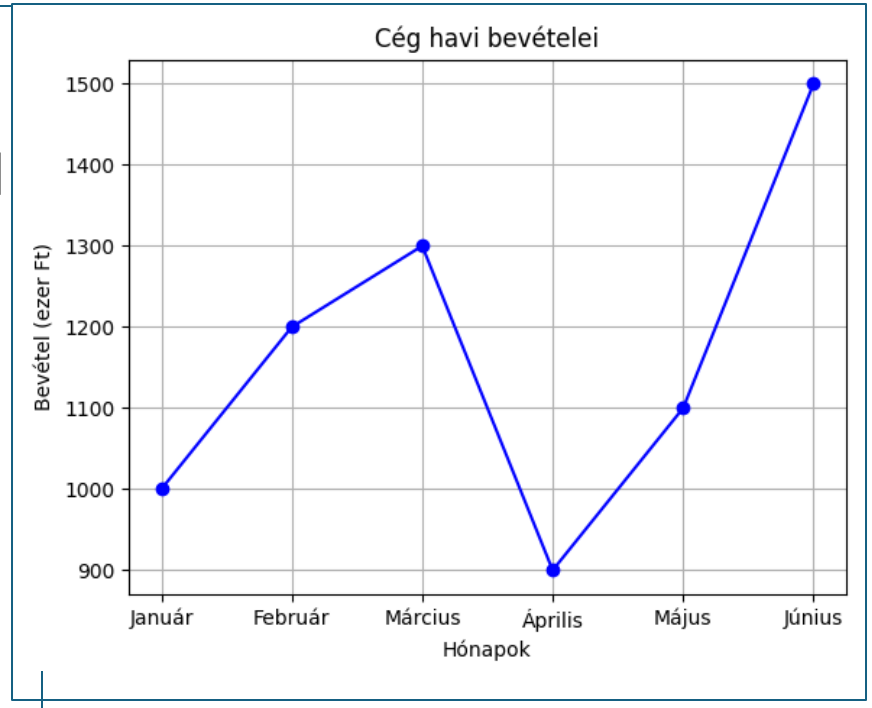
A vonaldiagram segít bemutatni egy változó időbeli alakulását.

Példa: Készítsünk egy vonaldiagramot, amely egy cég havi bevételeit ábrázolja.

Használat: **plt.plot()** függvény a grafikon létrehozásához.
Tengelyfeliratozás és -cím hozzáadása: **plt.xlabel(), plt.ylabel(), plt.title()**
A **linestyle** paraméter a vonal stílusát állítja be.

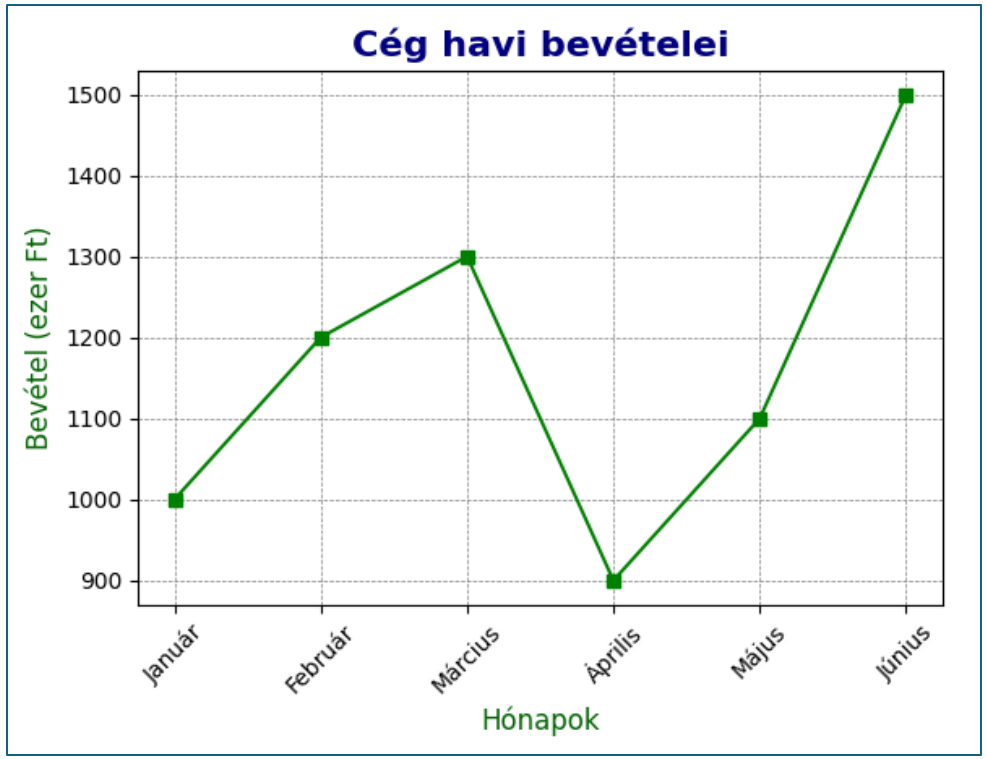
```
import matplotlib.pyplot as plt
# hónapok és bevételek
hónapok = ['Január', 'Február', 'Március', 'Április', 'Május', 'Június']
bevételek = [1000, 1200, 1300, 900, 1100, 1500]

plt.plot(hónapok, bevételek, marker='o', linestyle='-', color='b')
plt.title('Cég havi bevételei')
plt.xlabel('Hónapok')
plt.ylabel('Bevétel (ezer Ft)')
plt.grid(True)
plt.show()
```

(linedig.py)



Tengelyek, címek és feliratok testreszabása

```
plt.plot(hónapok, bevételek, marker='s', color='g')
plt.title('Cég havi bevételei', fontsize=16, fontweight='bold', color='navy')
plt.xlabel('Hónapok', fontsize=12, color='darkgreen')
plt.ylabel('Bevétel (ezer Ft)', fontsize=12, color='darkgreen')
plt.xticks(rotation=45)          # x tengely értékek elforgatása
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.show()
```



Cím és tengelyek formázása: betűméret, szín, stílus beállítása.

Elforgatás alkalmazása az x tengely feliratain: **plt.xticks(rotation=45)**

Rácsvonalak vezérlése: **plt.grid()**

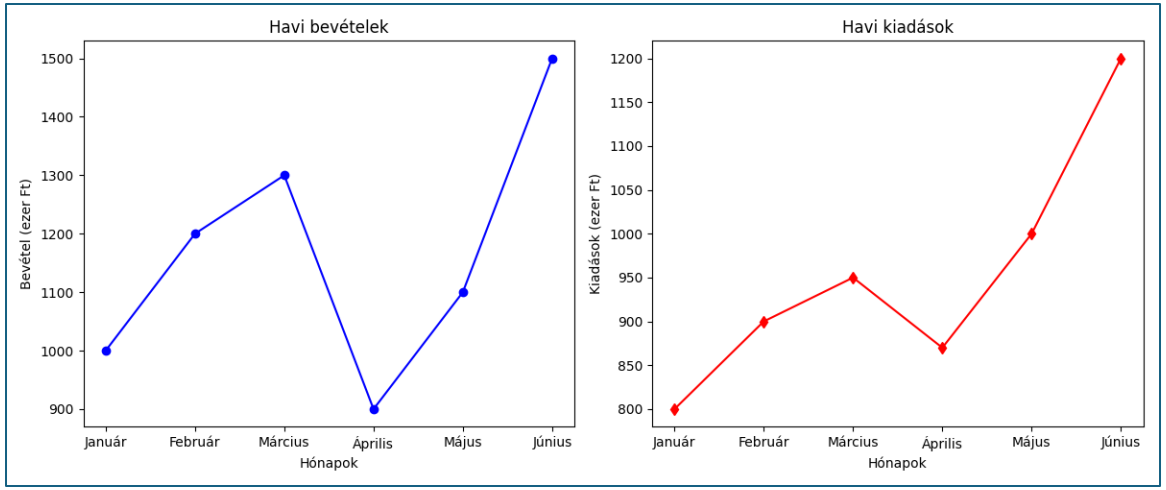
(differentplot.py)

Többszörös diagramok egy ábrán belül (subplotok)

Példa: Ábrázoljunk egyszerre két különböző diagramot egy ablakban – az egyik a bevételeket, a másik a kiadásokat mutatja.

```
kiadások = [800, 900, 950, 870, 1000, 1200]
plt.figure(figsize=(12, 5))
# Első subplot
plt.subplot(1, 2, 1)  # 1 sor, 2 oszlop, 1. subplot
plt.plot(hónapok, bevételek, color='blue', marker='o')
plt.title('Havi bevételek')
plt.xlabel('Hónapok')
plt.ylabel('Bevétel (ezer Ft)')
# Második subplot

plt.subplot(1, 2, 2)  # 1 sor, 2 oszlop, 2. subplot
plt.plot(hónapok, kiadások, color='red', marker='d')
plt.title('Havi kiadások')
plt.xlabel('Hónapok')
plt.ylabel('Kiadások (ezer Ft)')
plt.tight_layout()  # Igazítás a helyes megjelenítéshez
plt.show()
```

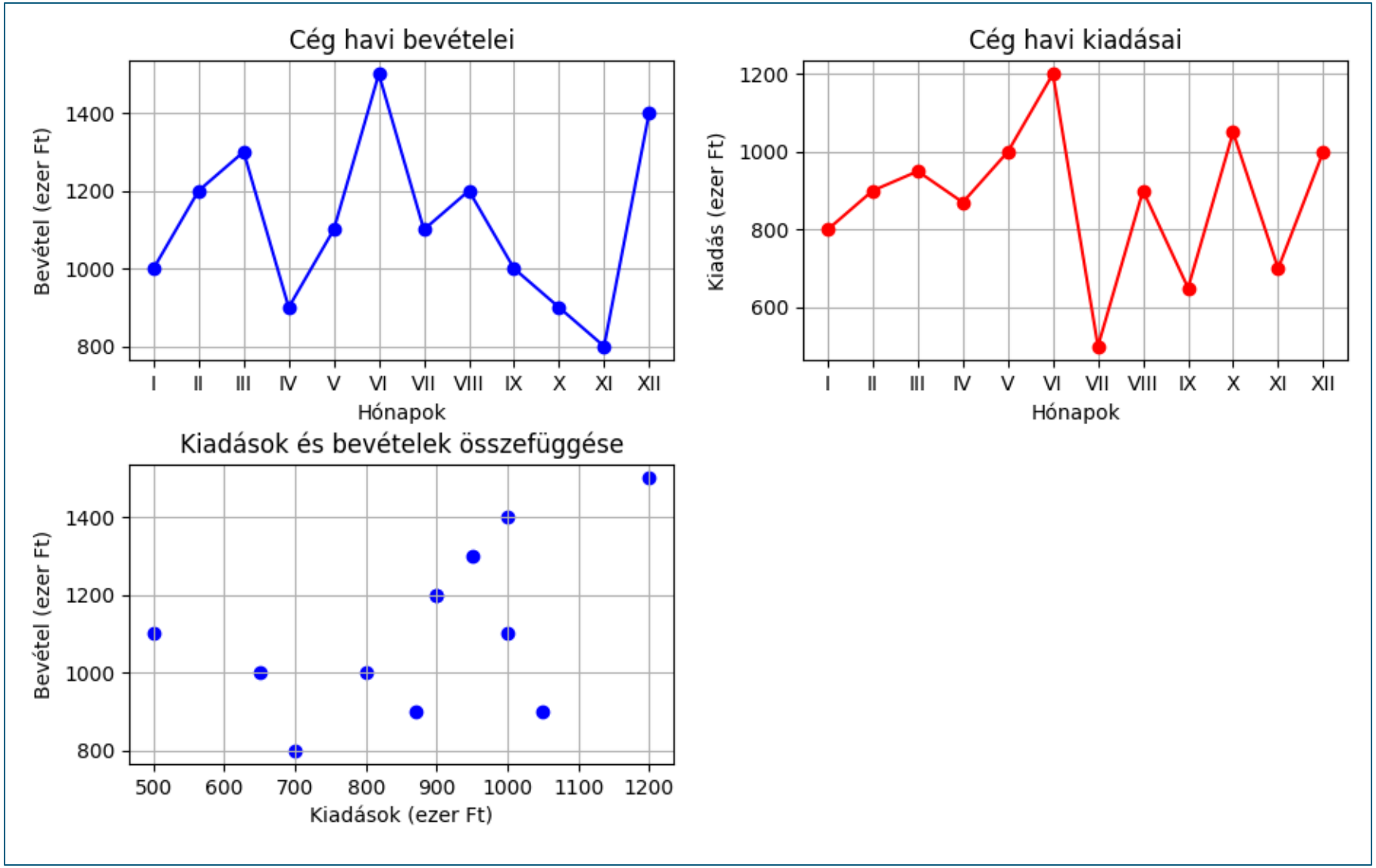
A **plt.subplot()** függvény használata:
Paraméterek: sorok, oszlopok száma és a subplot pozíciója.
A **plt.tight_layout()** az ábrák közötti térközöket automatikus optimalizálja.
(subplot.py)



Példa:
3 db subplot diagram egy Ábrában (moresubplot.py)

```
hónapok = ['I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX', 'X', 'XI', 'XII']
bevételek = [1000, 1200, 1300, 900, 1100, 1500, 1100, 1200, 1000, 900, 800, 1400]
kiadások = [800, 900, 950, 870, 1000, 1200, 500, 900, 650, 1050, 700, 1000]
plt.figure(figsize=(12, 12))
plt.subplot(2, 2, 1)                                    # 1. diagram
plt.plot(hónapok, bevételek, marker='o', linestyle='-', color='b')
plt.title('Cég havi bevételei')
plt.xlabel('Hónapok')
plt.ylabel('Bevétel (ezer Ft)')
plt.grid(True)
plt.subplot(2, 2, 2)                                    # 2. diagram
plt.plot(hónapok, kiadások, marker='o', linestyle='-', color='r')
plt.title('Cég havi kiadásai')
plt.xlabel('Hónapok')
plt.ylabel('Kiadás (ezer Ft)')
plt.grid(True)
plt.subplot(2, 2, 3)                                    # 3. diagram

plt.scatter(kiadások, bevételek, marker='o', color='b')
plt.title('Kiadások és bevételek összefüggése')
plt.xlabel('Kiadások (ezer Ft)')
plt.ylabel('Bevétel (ezer Ft)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

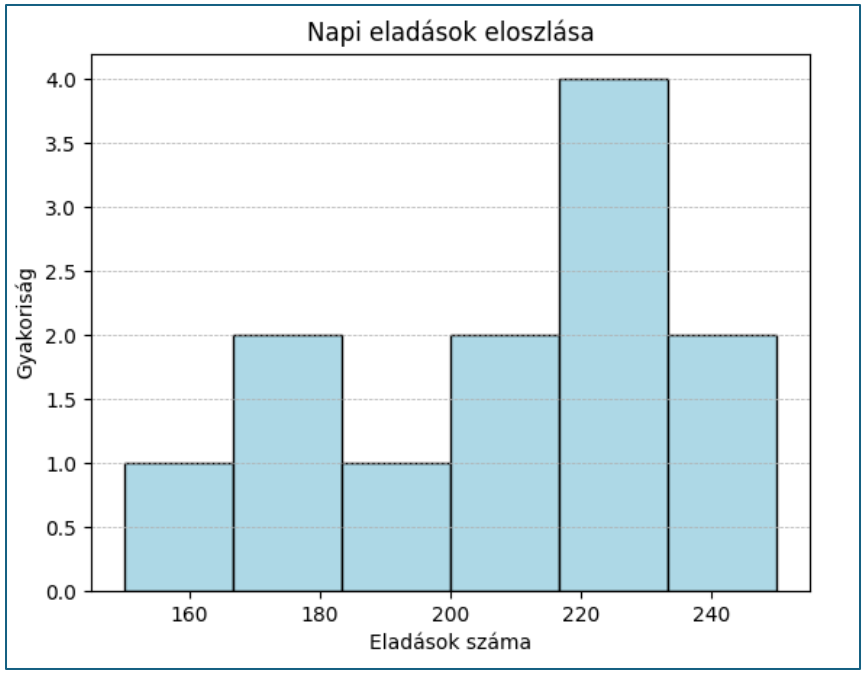
Példa: egy ábrában 3 darab subplot diagram



Hisztogramok és eloszlás-vizsgálatok (histogram.py)

A hisztogramok az adatok gyakoriságának bemutatására szolgálnak. Ez segít az adatok szimmetriájának, eloszlásának és anomáliáinak azonosításában.

```
import matplotlib.pyplot as plt
import pandas as pd

eladasok = [150, 180, 220, 240, 200, 230, 170, 190, 210, 250, 230, 220]

plt.hist(eladasok, bins=6, color='lightblue', edgecolor='black')
plt.title('Napi eladások eloszlása')
plt.xlabel('Eladások száma')
plt.ylabel('Gyakoriság')
plt.grid(axis='y', linestyle='--', linewidth=0.5)
plt.show()
```



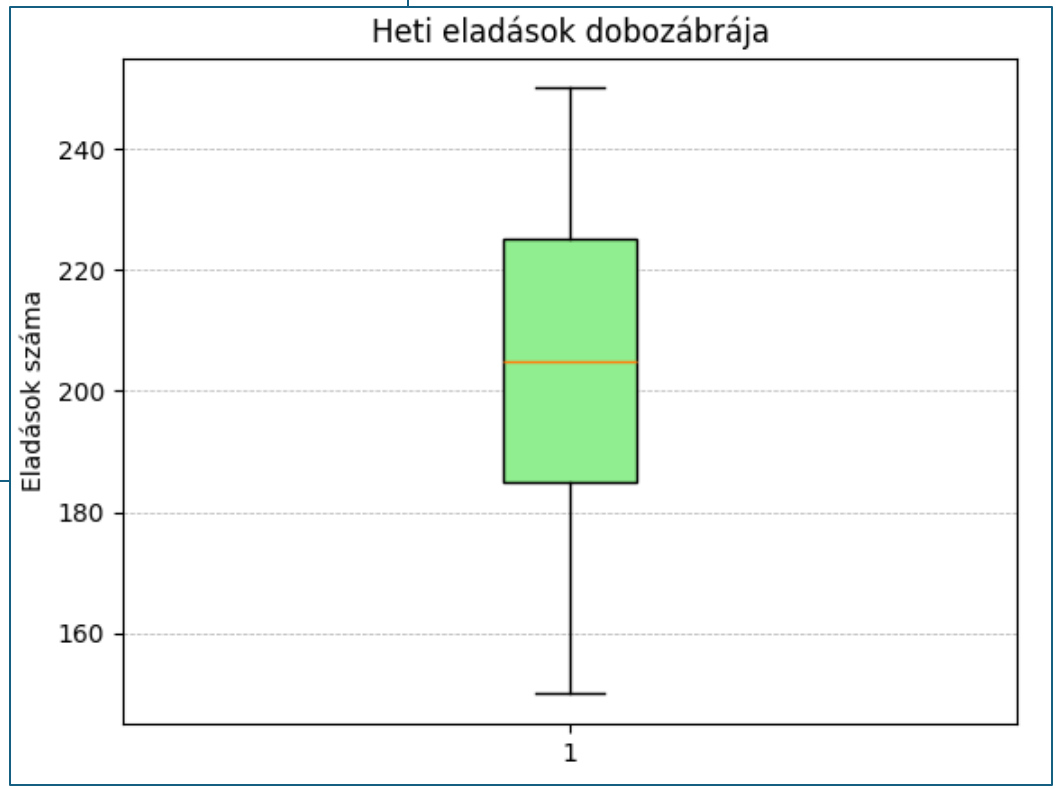
A **plt.hist()** függvény a hisztogram elkészítésére szolgál.
A **bins** paraméter határozza meg a sávok számát.
Az **edgecolor** a sávok körvonalának színét állítja be a jobb láthatóság érdekében.

Box plotok használata adatok szórásának bemutatására

A box plotok (dobozábrák) vizuálisan mutatják az adatok szórását, középértékét, valamint a kiugró értékeket. Ezek segítenek gyorsan azonosítani az adatok eloszlását és az esetleges anomáliákat. (boxplot.py)

Box plotok az adatok szórásának és kiugró értékeinek bemutatására.

```
# Heti eladások adatai
heti_eladasok = [150, 180, 220, 240, 200, 230,
      170, 190, 210, 250, 230, 220, 180, 195, 205]
# Box plot készítése
plt.boxplot(heti_eladasok, patch_artist=True,
   boxprops=dict(facecolor='lightgreen'))
plt.title('Heti eladások dobozábrája')
plt.ylabel('Eladások száma')
plt.grid(axis='y', linestyle='--', linewidth=0.5)
plt.show()
```



A **plt.boxplot()** függvény hozza létre a dobozábrát. A **patch_artist** és **boxprops** paraméterek segítségével testreszabhatjuk a doboz megjelenését. Az ábra megmutatja a **mediánt**, a **kvartiliseket,** az esetleges kiugró értékeket és a minta terjedelmét.

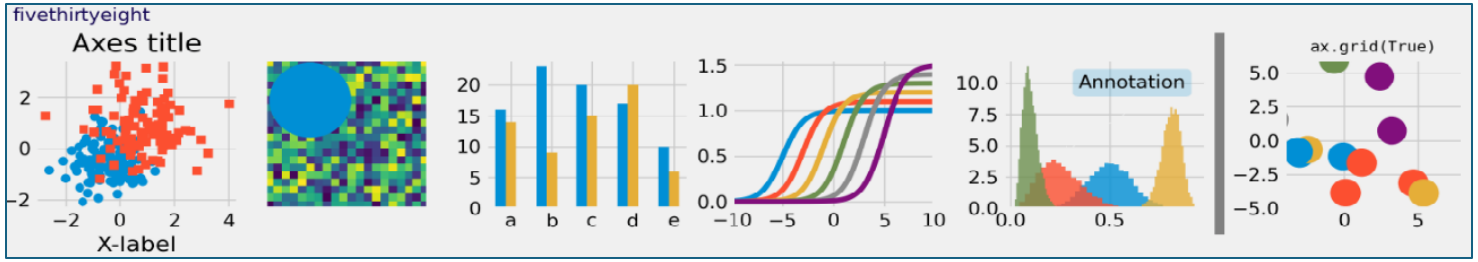
Medián és quartilisek

Medián: egy sorba rendezett adathalmaz középső értéke. Ha páros elemszám van, a két középső érték átlaga.
Quartilisek (Q1, Q2, Q3): az adatok három osztópontja, amelyek az adatokat négy egyenlő részre bontják. Q2 a medián (50%-os percentilis), Q1 a 25%-os, Q3 a 75%-os percentilis.
IQR (interquartile range): Q3 − Q1, az eloszlás középső 50%-ának terjedelme. Az 1.5·IQR szabály segít kiugró értékeket találni (alsó határ = Q1 − 1.5·IQR, felső határ = Q3 + 1.5·IQR).

Rendezett sorozat: 150, 170, 180, 190, 200, 210, 220, 220, 230, 230, 240, 250
Q1 = 185,
Q2(medián) = 215,
Q3 = 230
IQR = 45
kiugró küszöbök: [117.5, 297.5]
nincs kiugró érték

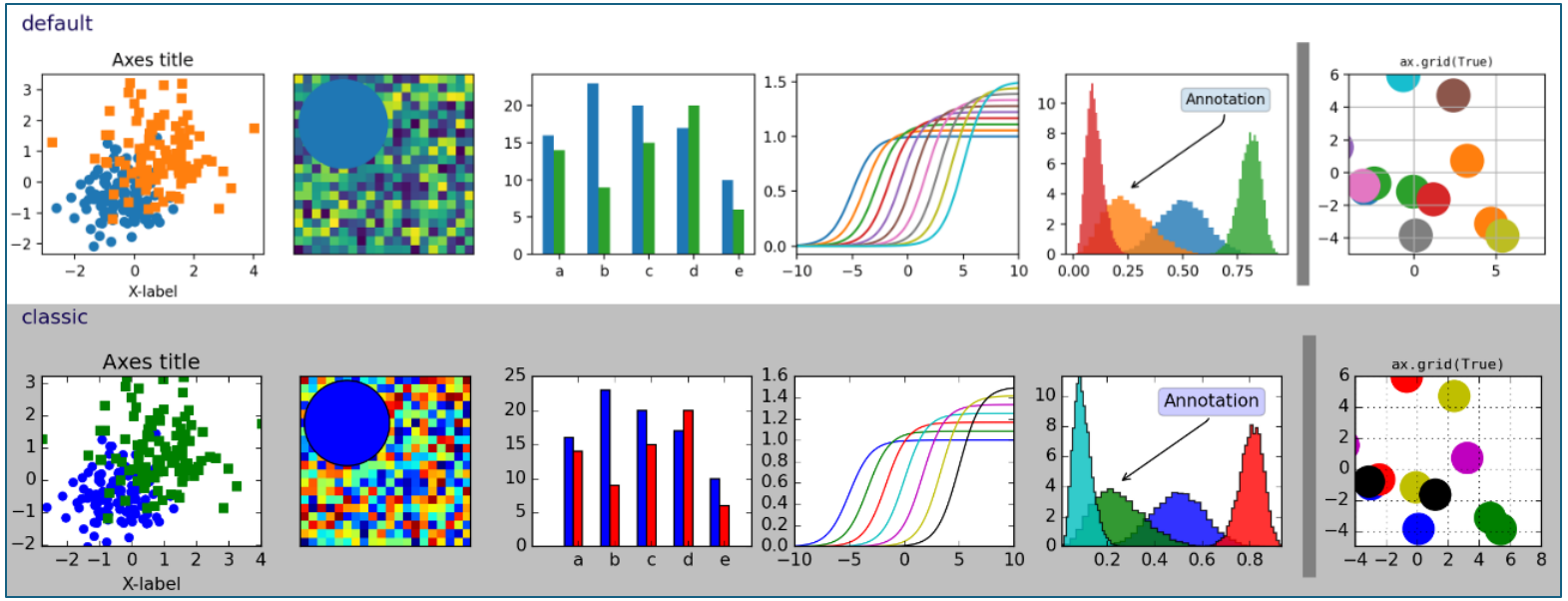
Stílusok beállítása (styles.py)

```
# Stílus beállítása
plt.style.use('fivethirtyeight')
```



A **plt.style.use()** paraméterével stíluslapot választhatunk a megjelenítéshez az alapbeállítástól eltérően.

[Style sheets reference — Matplotlib 3.9.2 documentation]



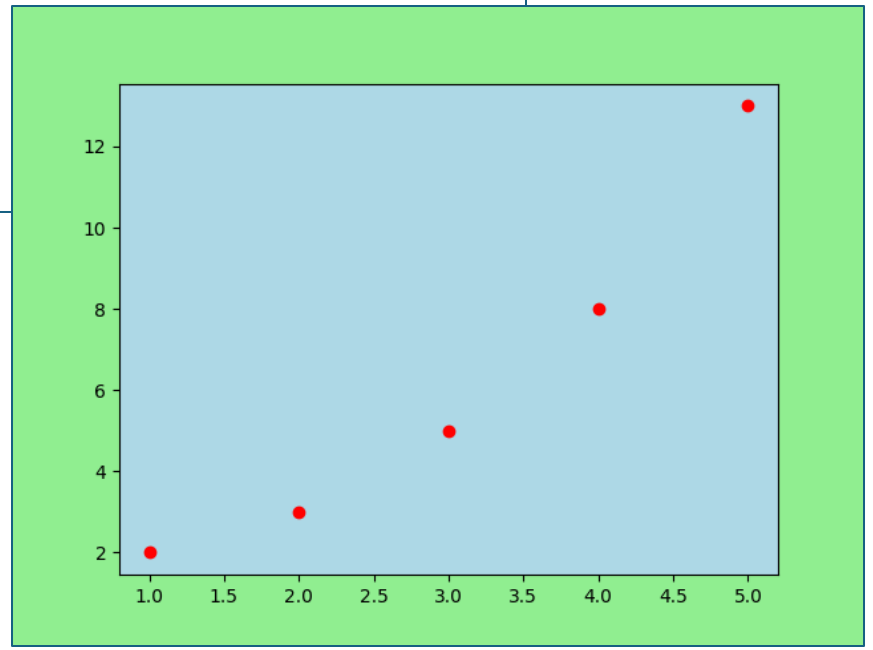
2. Adatok megjelenítése Pandas használatával

- A Pandas és a Matplotlib jól integrálhatók, lehetővé téve a DataFrame objektumok közvetlen vizualizálását.
- A Matplotlib alapértelmezés szerint megjelenik, ha a Pandas **plot()** metódusát hívjuk meg a DataFrame-re.
- Fontos csomagok importálása: **import pandas as pd**

  import matplotlib.pyplot as plt

Háttérszínek beállítása (backgroundcolor.py)

```
import matplotlib.pyplot as plt
import pandas as pd

data = {'X': [1, 2, 3, 4, 5], 'Y': [2, 3, 5, 8, 13]}
df = pd.DataFrame(data)

fig = plt.figure(facecolor='lightgreen’)  # ábra háttérszíne
ax = plt.gca()  # Az aktuális tengelyek lekérése
ax.set_facecolor('lightblue’)  # A rajzterület háttérszínének beállítása
# Adatok ábrázolása
plt.plot(df['X'], df['Y'], 'ro')
# Diagram megjelenítése
plt.show()
```

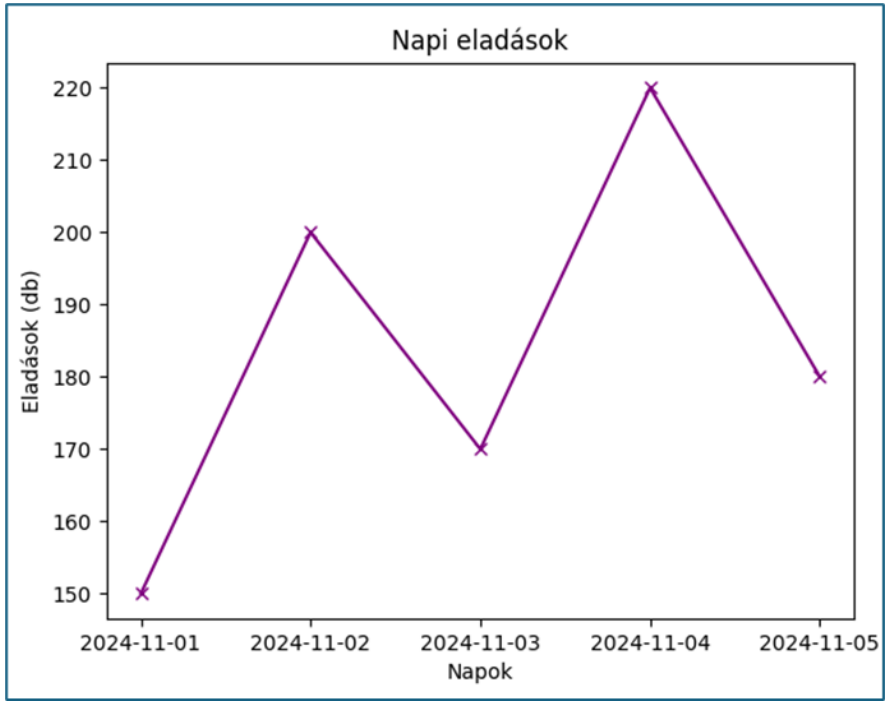


Adatok betöltése Pandas-szal és megjelenítése Matplotlib-bel

Példa: Egy CSV fájlban egy bolt napi eladásait tartjuk nyilván. Az adatok beolvasása egy DataFrame-be történik a Pandas segítségével. Ábrázolás Matplotlib használatával.

```
import pandas as pd
# CSV adatállomány betöltése
adatok = pd.read_csv('eladasok.csv')
napok = adatok['Nap']
eladások = adatok['Eladás']
# Vonaldiagram készítése
plt.plot(napok, eladások, marker='x', linestyle='-', color='purple')
plt.title('Napi eladások')
plt.xlabel('Napok')
plt.ylabel('Eladások (db)')
plt.show()
print(adatok.head())
```

	Nap	Eladás
0	2024-11-01	150
1	2024-11-02	200
2	2024-11-03	170
3	2024-11-04	220
4	2024-11-05	180



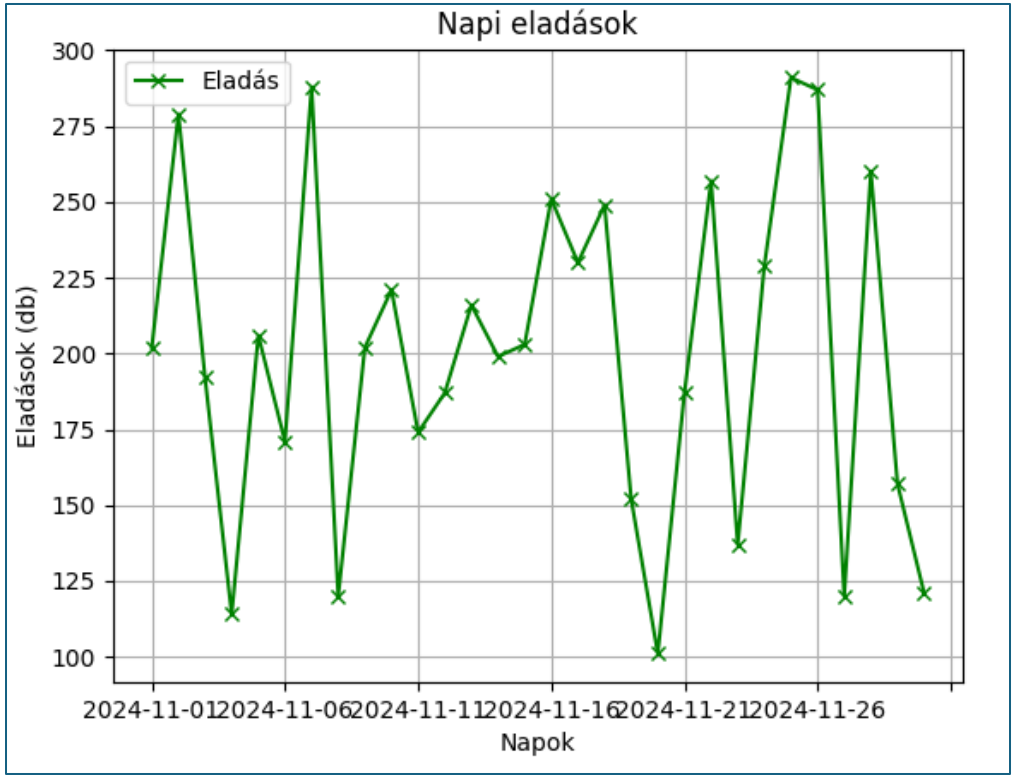
A **plt.plot()** függvényel a beolvasott adatokat ábrázoljuk, ahol az x tengelyen a napok, az y tengelyen pedig az eladások szerepelnek. (pandasplot1.py)

Adatmegjelenítés beépített Pandas plot metódussal

A Pandas DataFrame és Series objektumai beépített **plot()** metódussal rendelkeznek.

Példa: Vonal-diagram készítése beépített Pandas metódussal

```
# Vonal-diagram készítése közvetlenül Pandas segítségével
adatok = pd.read_csv('eladasok_30.csv')
adatok.plot(x='Nap', y='Eladás', kind='line', marker='x’,
        color='green', title='Napi eladások')
plt.xlabel('Napok')
plt.ylabel('Eladások (db)')
plt.grid(True)
plt.show()
# A beolvasott adatok megtekintése
print(adatok.head())
```

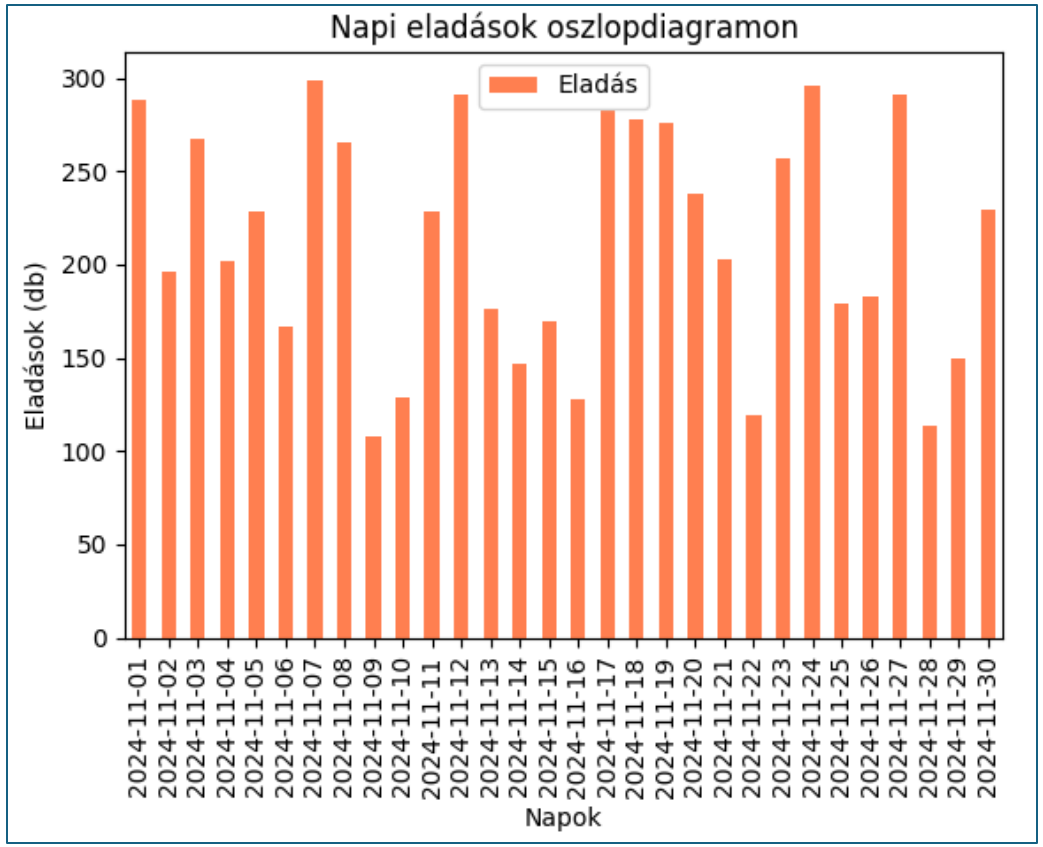


	Nap	Eladás	Látogatók
0	2024-11-01	288	78
1	2024-11-02	196	56
2	2024-11-03	267	75
3	2024-11-04	202	61
4	2024-11-05	228	70

A **kind** paraméter határozza a meg a diagram típusát (pl. line, bar), az **x és y** a tengelyek megadása, a **marker** és **color** paraméterek a megjelenést szabályozzák. (pandplotMain.py)

Oszlopdiagram készítése

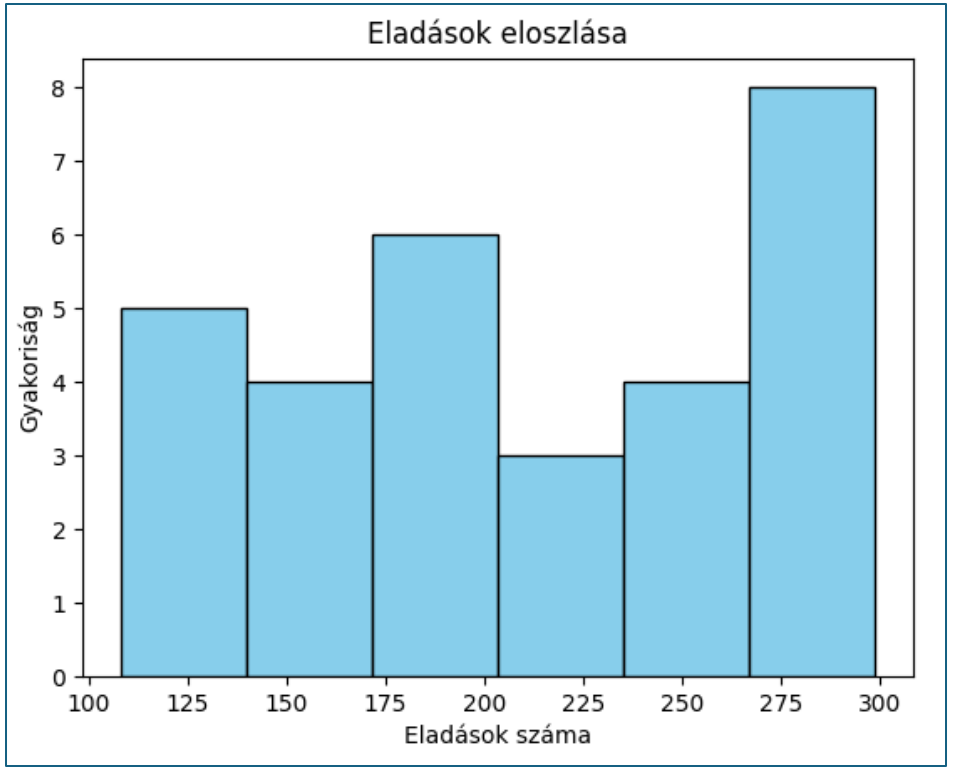
```
adatok.plot(x='Nap', y='Eladás', kind='bar', color='coral', title='Napi eladások oszlopdiagramon')
plt.xlabel('Napok')
plt.ylabel('Eladások (db)')
plt.show()
```



A **kind='bar'** beállítással oszlop-diagramot készítünk.
Ez a diagram jól használható az adatok összehasonlítására különböző időszakokban.
(oszlopdiag.py)

Hisztogram készítése az eladások eloszlásának bemutatására

```
# Hisztogram készítése
adatok['Eladás'].plot(kind='hist', bins=6, color='skyblue', title='Eladások eloszlása'),
        edgecolor='black')
plt.xlabel('Eladások száma')
plt.ylabel('Gyakoriság')
plt.show()
```



A **kind='hist'** beállítással hisztogramot készítünk, amely az adatok eloszlását mutatja.

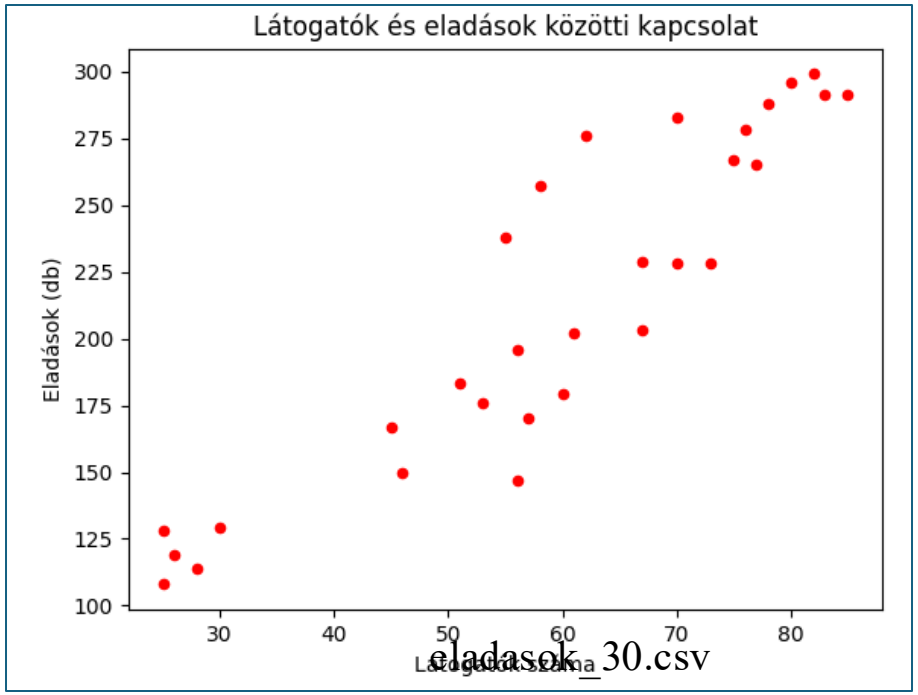
A **bins** paraméter határozza meg a csoportok számát. (histogram.py 2*)

Scatter plot a korreláció vizsgálatához

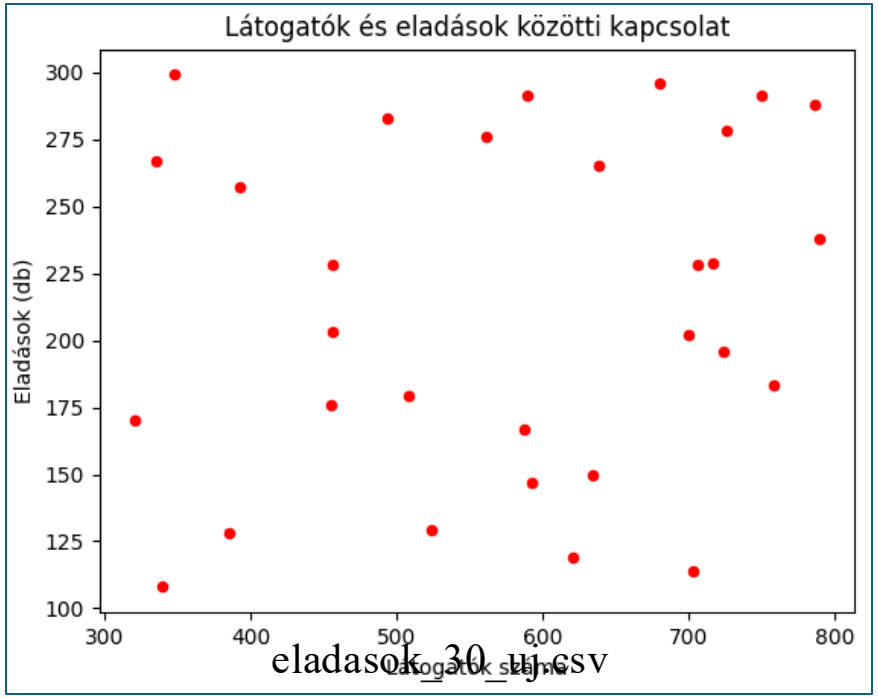
Ehhez van van egy másik oszlopunk, amely a bolt látogatóinak számát tartalmazza.

```
# Scatter plot készítése a látogatók száma és eladások közötti kapcsolat ábrázolására
adatok.plot(kind='scatter', x='Látogatók', y='Eladás', color='red’,
      title='Látogatók és eladások közötti kapcsolat')
plt.xlabel('Látogatók száma')
plt.ylabel('Eladások (db)')
plt.show()
```

(plotandscatter.py)



eladasok_30.csv



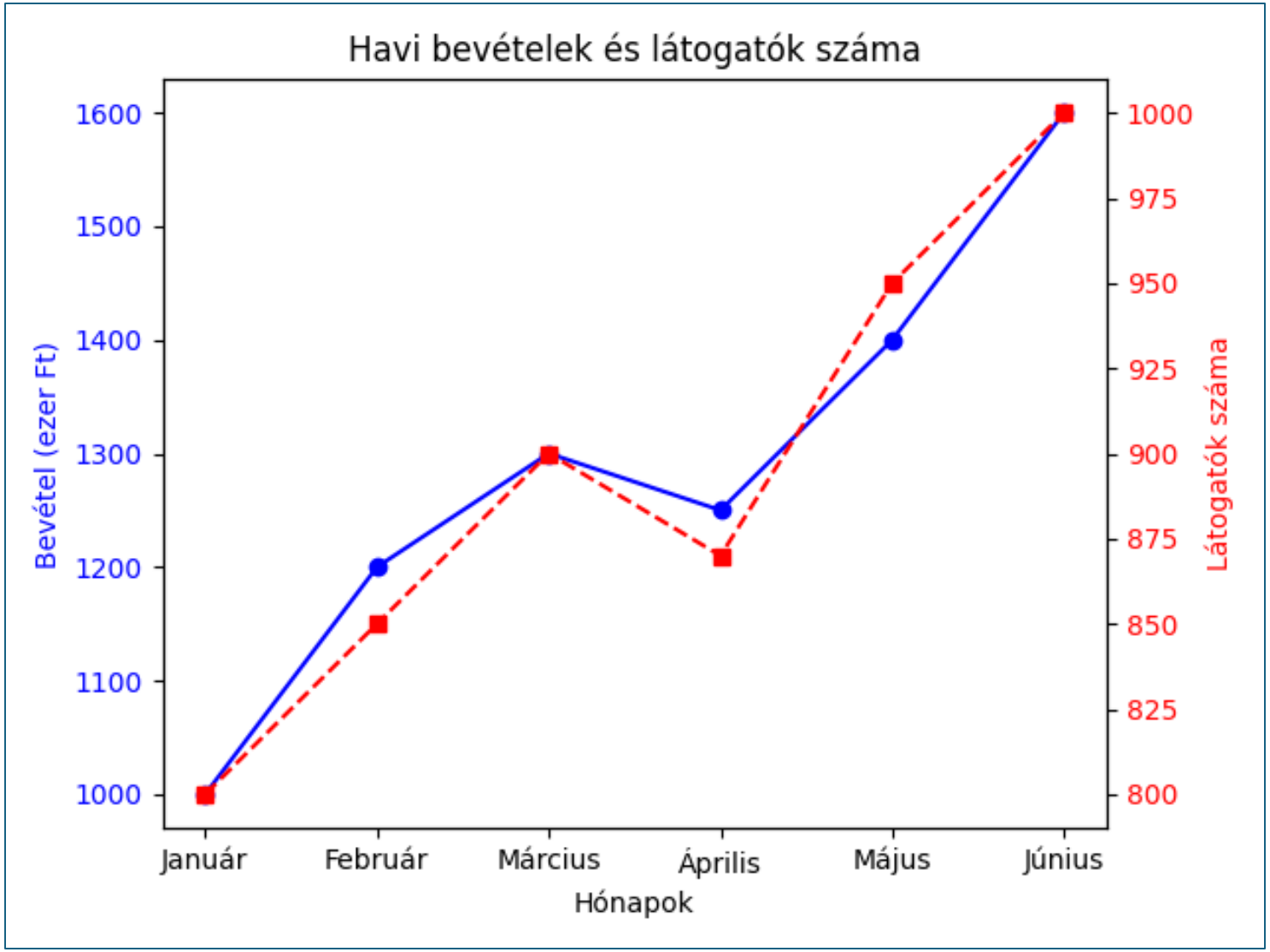
eladasok_30_uj.csv

3. Komplex diagramok létrehozása

Példa: Egy adatkészlet alapján két változó ábrázolását **egy diagramon két tengellyel** oldjuk meg. Egy bolt havi bevételeit és látogatóinak számát ábrázoljuk a hónapok függvényében.

```
# Adatok generálása
honapok = ['Január', 'Február', 'Március', 'Április', 'Május', 'Június']
bevetelek = [1000, 1200, 1300, 1250, 1400, 1600]
latogatok = [800, 850, 900, 870, 950, 1000]
# Kombinált diagram készítése
fig, ax1 = plt.subplots()                    # egy ábra és egy tengely létrehozása
# Első Y tengely: bevételek
ax1.plot(honapok, bevetelek, color='blue', marker='o', label='Bevétel')
ax1.set_xlabel('Hónapok')
ax1.set_ylabel('Bevétel (ezer Ft)', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')
# Második Y tengely: látogatók száma
ax2 = ax1.twinx()
ax2.plot(honapok, latogatok, color='red', marker='s', linestyle='--', label='Látogatók száma')
ax2.set_ylabel('Látogatók száma', color='red')
ax2.tick_params(axis='y', labelcolor='red')
plt.title('Havi bevételek és látogatók száma')
fig.tight_layout()
plt.show()
```

(complex.py)



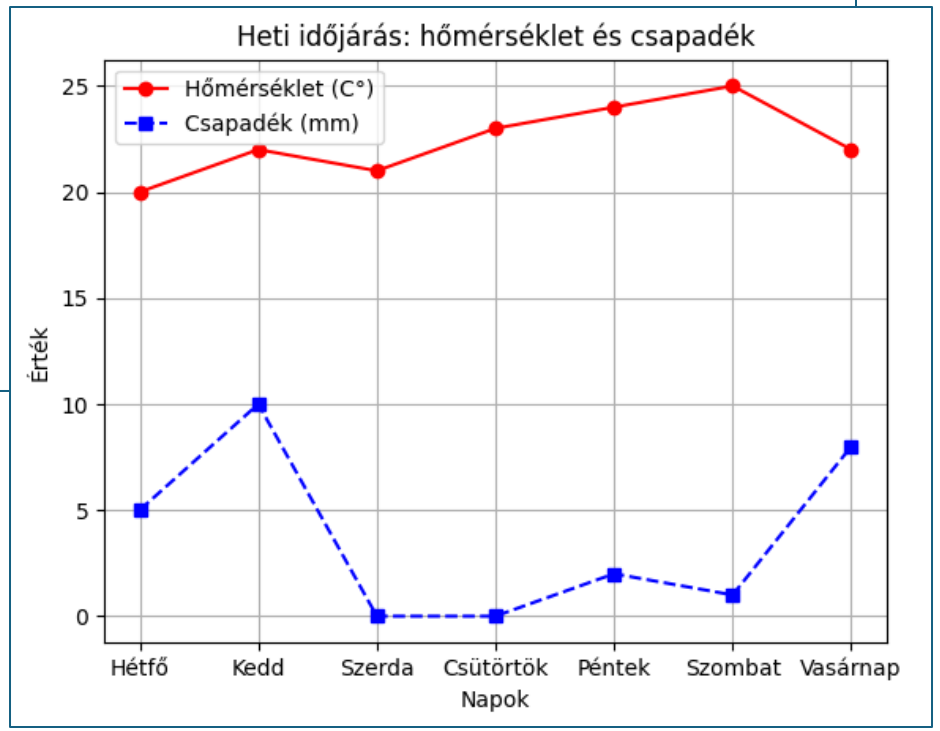
A **plt.subplots()** és a **twinx()** metódusok segítségével két y tengelyt hozhatunk létre, így egy diagramon egyszerre két különböző adatot ábrázolhatunk.

Példa: Overlay diagram készítése

Több különböző adatsor ábrázolása egy diagramon: példánkban két grafikon, amelyek egy város napi legnagyobb hőmérsékletét és napi csapadékmennyiségét mutatják.

```
napok = ['Hétfő', 'Kedd', 'Szerda', 'Csütörtök', 'Péntek', 'Szombat', 'Vasárnap']
homerseklet = [20, 22, 21, 23, 24, 25, 22]  # Celsius fok
csapadek = [5, 10, 0, 0, 2, 1, 8]  # mm
plt.plot(napok, homerseklet, marker='o', color='red', label='Hőmérséklet (C°)')
plt.plot(napok, csapadek, marker='s', color='blue', linestyle='--', label='Csapadék (mm)')
plt.title('Heti időjárás: hőmérséklet és csapadék')
plt.xlabel('Napok')
plt.ylabel('Érték')
plt.legend()
plt.grid(True)
plt.show()
```

(complex.py)



Az y tengelyre az Érték feliratot írjuk és a **label** címke, mint jelmagyarázat azonosítja az egyes grafikonokat.

4. Seaborn bevezetése és használata

A Seaborn könyvtár alkalmazásának előnyei:

- A Seaborn egy magasabb szintű Python-könyvtár Matplotlib alapokon
- Előnyei: egyszerű használat, különleges diagramok, Pandas integráció
- Jól alkalmazható statisztikai adatok vizualizálására és összetett grafikonok létrehozására
- Hőtérképek, eloszlásdiagramok és trenddiagramok
- Beépített esztétikai beállítások
- Könnyen integrálható Pandas DataFrame-ekkel
- Több beépített plot típust tartalmaz, amelyekkel különböző típusú adatvizualizációkat készíthetünk.
- Telepítés: **pip install seaborn**

Alapvető Seaborn plotok: relplot(). (seabornrelplot.py)

A **relplot()** egy sokoldalú függvény, amely kapcsolatokat jelenít meg az adatok között.
Példa: Egy cég napi eladásait és a látogatók számát ábrázoljuk scatter plot-tal:

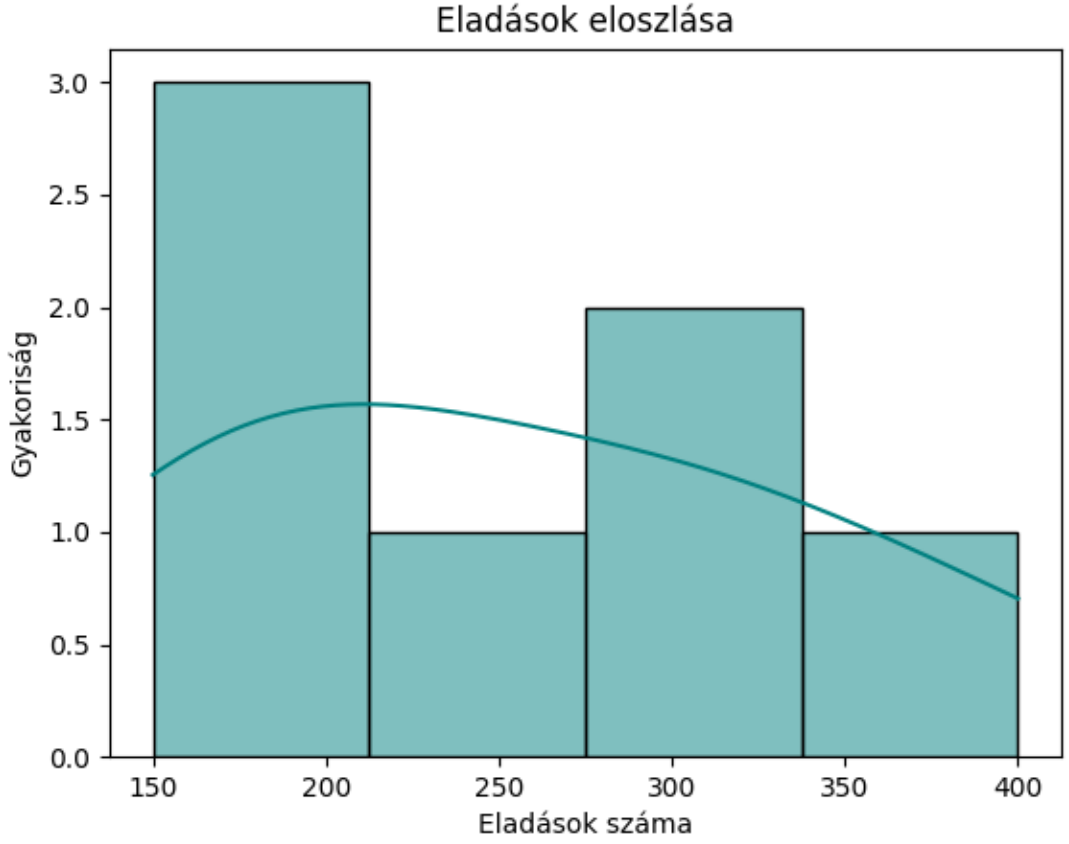
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Adatok generálása DataFrame-ben
adatok = pd.DataFrame({
    'Nap': ['Hétfő', 'Kedd', 'Szerda', 'Csütörtök', 'Péntek',
            'Szombat', 'Vasárnap'],
    'Eladások': [150, 200, 170, 240, 300, 400, 320],
    'Látogatók': [200, 250, 230, 300, 350, 500, 450]
})
# relplot használata
sns.relplot(x='Látogatók', y='Eladások', data=adatok,
            kind='scatter')
plt.title('Látogatók és eladások közötti kapcsolat')
plt.show()
```

Alapvető Seaborn plotok: histplot() (seabornhist.py)

Eladások eloszlásának megjelenítése **histplot()**-tal:

```
sns.histplot(adatok['Eladások'], kde=True, color='teal')
plt.title('Eladások eloszlása')
plt.xlabel('Eladások száma')
plt.ylabel('Gyakoriság')
plt.show()
```

A **hisztogram** mellett egy **Kernel Density Estimate** (KDE) görbét is megjelenít, amely az adatok **sűrűségét** mutatja.

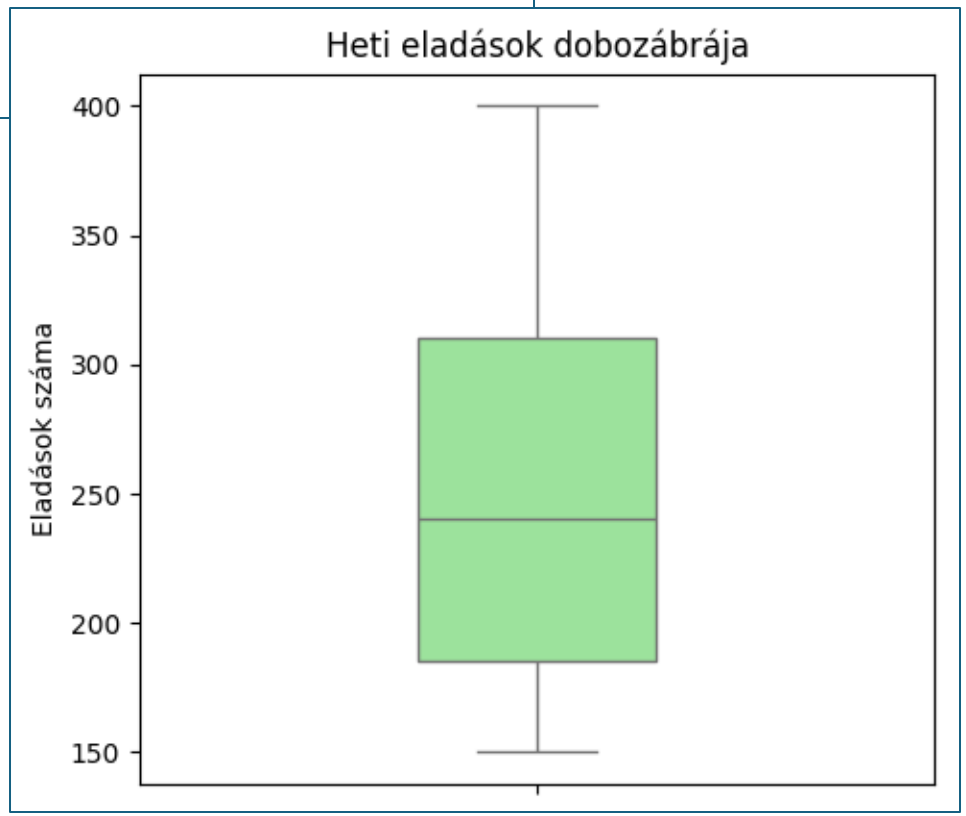


Alapvető Seaborn plotok: boxplot() (seabornboxplot.py)

A **boxplot()** segítségével az adatok szórását, mediánját és kiugró értékeit vizualizálhatjuk.

Példa: Heti eladások ábrázolása dobozábrán.

```
sns.boxplot(y=adatok['Eladások'], color='lightgreen', width=0.3)
plt.title('Heti eladások dobozábrája')
plt.ylabel('Eladások száma')
plt.show()
```

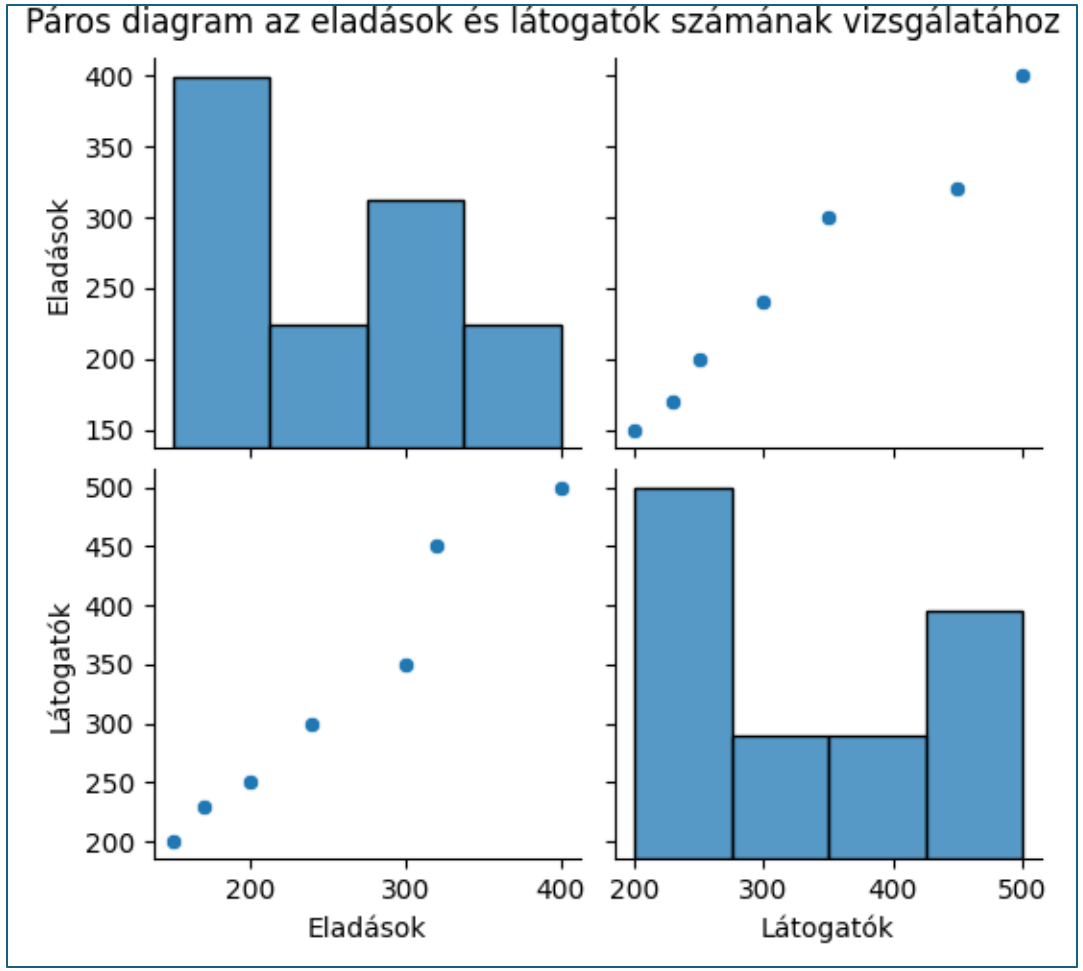


A Seaborn boxplotok téglalapja az alapbeállítás miatt szélesebb a ptl boxploténál. A **width=0.3** az eredetileg kapott téglalap szélességét 30%-ra állítja be.

Alapvető Seaborn plotok: pairplot() (seabornpairplot.py)

A **pairplot()** több változó közötti kapcsolatok és eloszlások bemutatására szolgál, és automatikusan elkészíti a scatter plotokat és hisztogramokat.
Példa: Többváltozós elemzés:

```
sns.pairplot(adatok)
plt.suptitle('Páros diagram az eladások
és látogatók számának vizsgálatához', y=1.0)
plt.show()
```



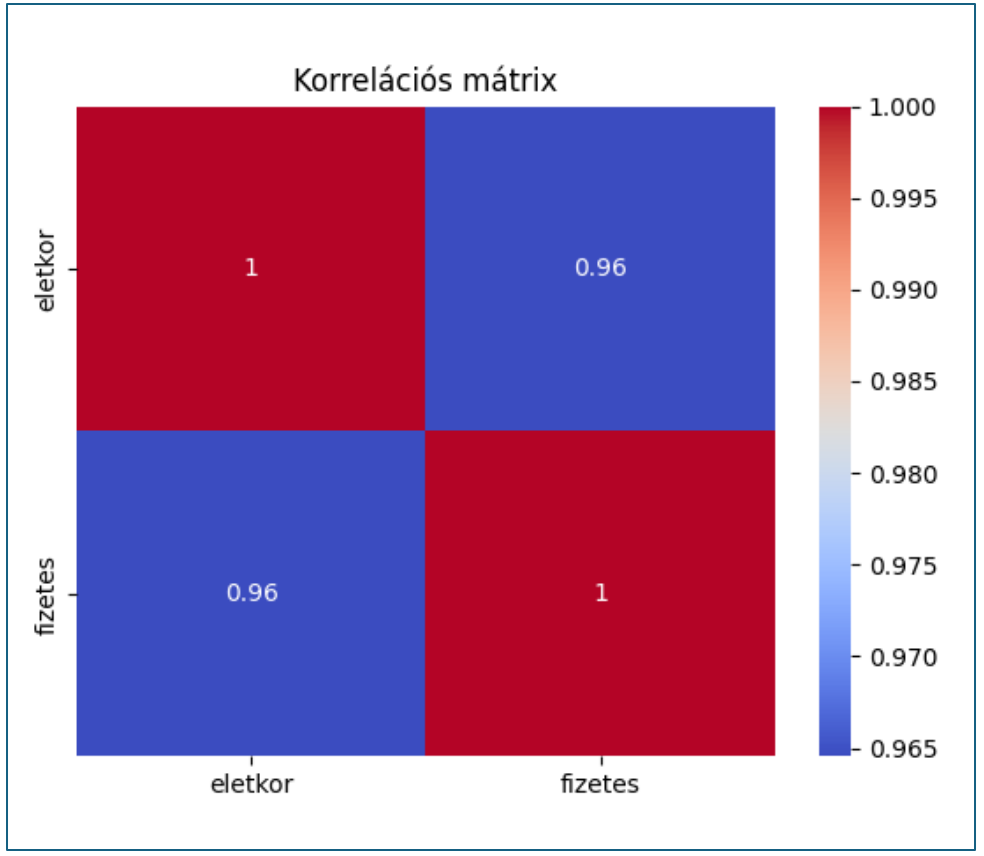
Alapvető Seaborn plotok: heatmap() (seabornheatmap.py)

Adatok korrelációjának vizualizációja

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
adatok = {
   'eletkor': [23, 45, 34, 25, 32, 40, 29, 48, 37, 22],
   'fizetes': [250, 500, 300, 260, 320, 480, 290, 520, 410, 240]
}
adat_df = pd.DataFrame(adatok)
# Korrelációs mátrix kiszámítása
korrelacio = adat_df.corr()
print("\nKorrelációs mátrix:")
print(korrelacio)
# Korrelációs mátrix megjelenítése
sns.heatmap(korrelacio, annot=True, cmap='coolwarm')
plt.title('Korrelációs mátrix')
plt.show()
```

Korrelációs mátrix:

	eletkor	fizetes
eletkor	1.000000	0.964549
fizetes	0.964549	1.000000

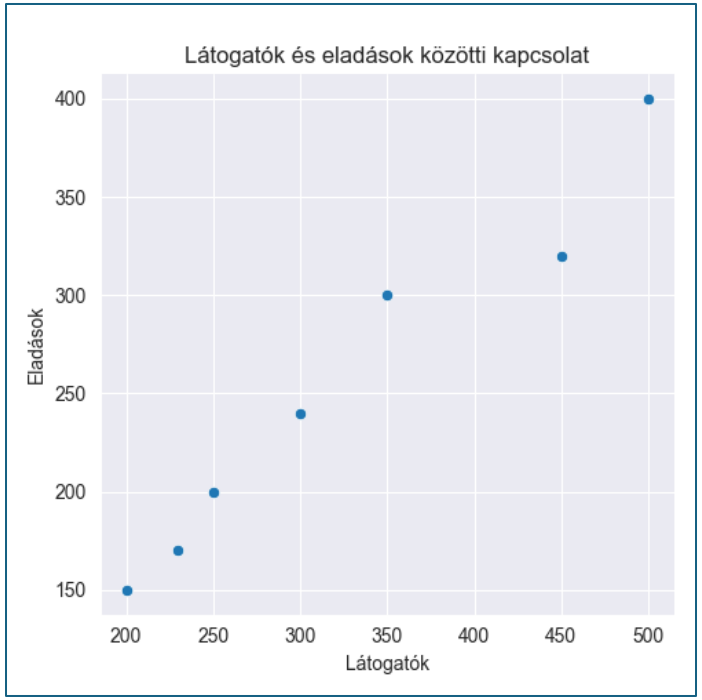


Stílusok és esztétikai testreszabások

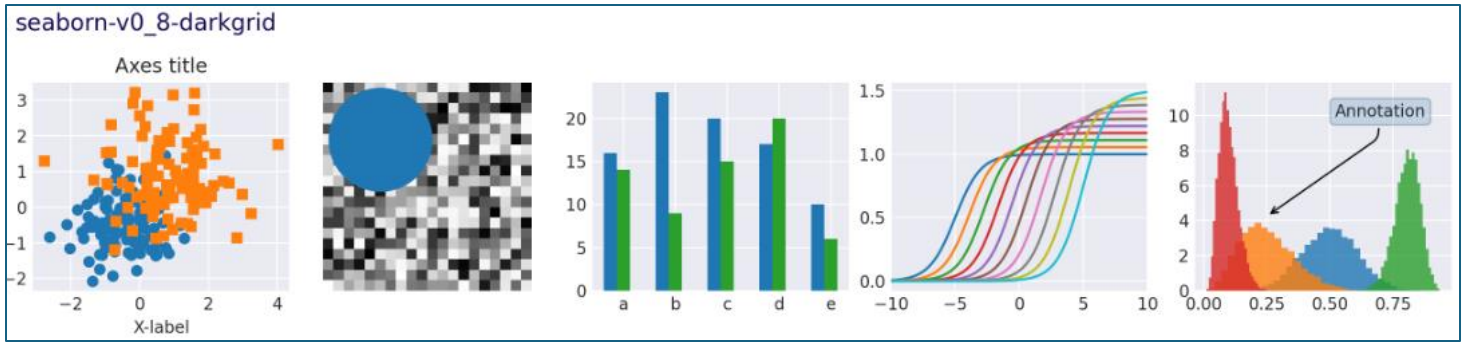
A Seaborn beépített stílusai lehetővé teszik az ábrák gyors testreszabását. Például beállíthatjuk a sötét hátteret, rácsozást vagy egyéb esztétikai elemeket.

Stílusbeállítások használata:

```
sns.set_style('darkgrid')
plt.figure(figsize=(10, 6))
# relplot használata stílusban
sns.relplot(x='Látogatók', y='Eladások', data=adatok, kind='scatter')
plt.title('Látogatók és eladások közötti kapcsolat')
plt.show()
```



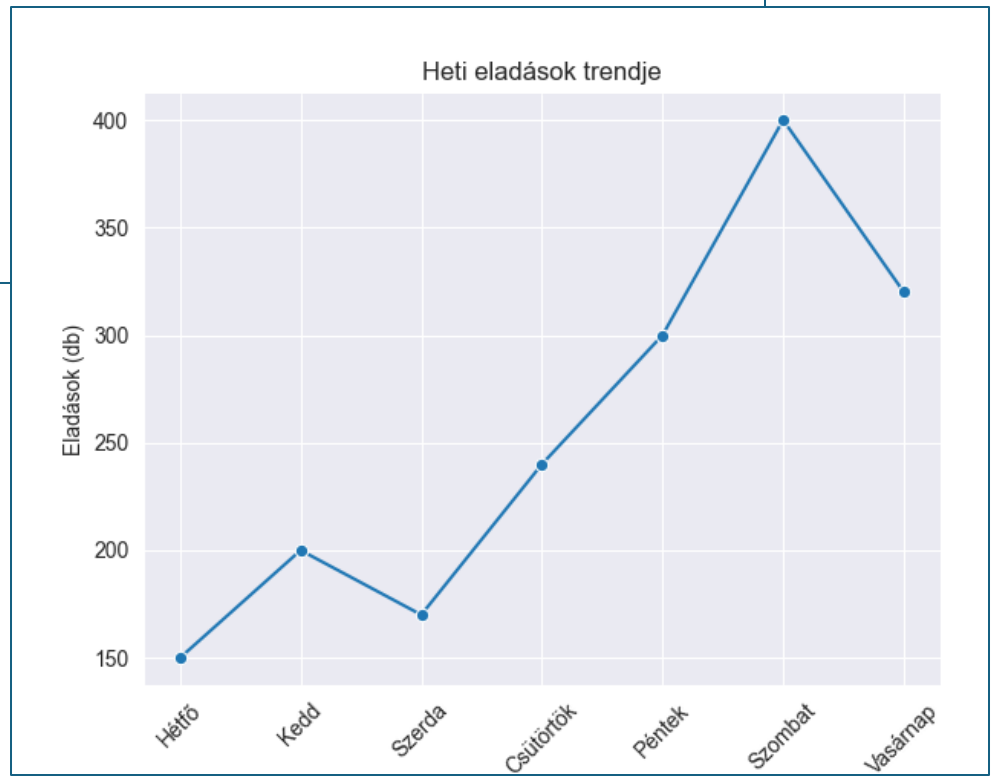
Beépített Seaborn stílusok változatai:
[Style sheets reference — Matplotlib 3.9.2 documentation]



Adatcsoportosítás és trendek ábrázolása

Példa: Csoportosítsuk az adatokat napok szerint, készítsünk vonaldiagramot és vizsgáljuk meg a trendeket.

```
sns.set_style('darkgrid')
# Csoportosított relplot
sns.relplot(x='Nap', y='Eladások', data=adatok, kind='line', marker='o')
plt.title('Heti eladások trendje')
plt.xlabel('Napok')
plt.ylabel('Eladások (db)')
plt.xticks(rotation=45)
plt.show()
```



A **kind='line'** paraméterrel vonaldiagramot hozzunk létre, az adatok változásának bemutatására az idő függvényében.
A **marker='o'** beállítás kiemeli az adatpontokat a diagramon.

5. Grafikonok készítése gyakorlati példák alapján

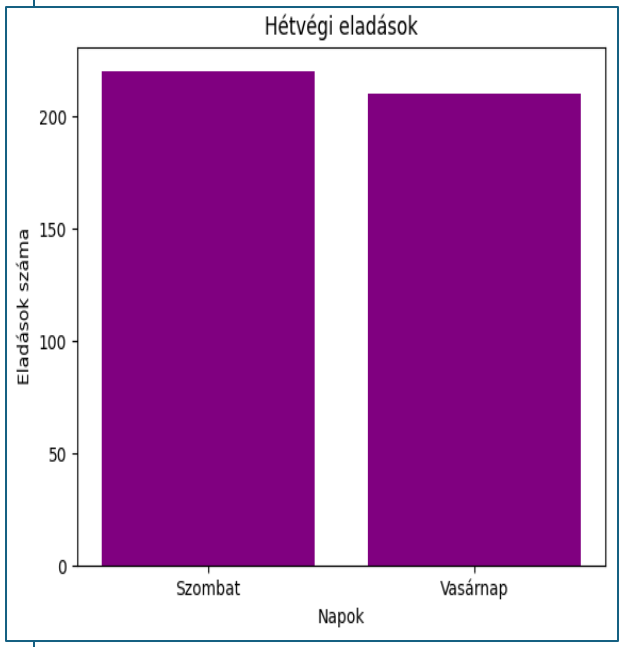
1. **Példa: Gazdasági adatok** vizualizálására készítsünk egy vonaldiagramot, amely egy ország GDP-jének alakulását ábrázolja az évek során.

```
import matplotlib.pyplot as plt
import pandas as pd
# Gazdasági adatok (például éves GDP)
evek = [2015, 2016, 2017, 2018, 2019, 2020, 2021]
gdp = [800, 850, 900, 950, 1000, 970, 1100]  # Milliárd USD
plt.plot(evek, gdp, marker='o', linestyle='-', color='green')
plt.title('Ország GDP-je az évek során')
plt.xlabel('Évek')
plt.ylabel('GDP (Milliárd USD)')
plt.grid(True)
plt.show()
```

5. Példa: Adatok szűrése és különböző szempontok szerinti ábrázolás

Oszlopdiagramot készítünk, amely egy bolt eladásait mutatja a hét különböző napjain, csak a hétvégi adatokat kiemelve. Szűrés a DataFrame-ben:

```
adatok = pd.DataFrame({
   'Nap': ['Hétfő', 'Kedd', 'Szerda', 'Csütörtök', 'Péntek', 'Szombat', 'Vasárnap'],
   'Eladások': [120, 150, 130, 160, 170, 220, 210]
})

# Adatok szűrése csak hétvégi napokra
hetvegi_adatok = adatok[adatok['Nap'].isin(['Szombat', 'Vasárnap'])]

plt.bar(hetvegi_adatok['Nap'], hetvegi_adatok['Eladások'], color='purple')
plt.title('Hétvégi eladások')
plt.xlabel('Napok')
plt.ylabel('Eladások száma')
plt.show()
```



Az **isin()** metódussal szűrjük a DataFrame-t, majd a kiválasztott adatokat ábrázoljuk.

Köszönöm a figyelmet!