Repetisjon/oppgaver i BED-1304 Python-Lab

Markus J. Aase

Forelesning 7 - Matplotlib

Informasjon

Til nå har vi gått igjennom flere ting i Python-lab. Det inkluderer temaer som:

- Python basics
- Funksjoner (Innebygde funksjoner og de vi definerer selv)
- Lister
- Dictionary
- Tuples
- Pakker som `Pandas` og `NumPy`
- Logikk og løkker (`if/else`-setninger, `for`-løkker og `while`-løkker).

Videre skal vi fokusere på visualisering av data, et kjempeviktig tema for økonomer. Programmering og data er ikke særlig nyttig, hvis vi ikke kan visualisere det. Det skal vi gjøre ved hjelp av Python-biblioteket **matplotlib**.

Dette dokumentet er lagd for å repetere og teste forståelsen av kjernepensum i **BED-1304 Python-Lab**. Husk at dette er et supplement til forelesning og seminar.

God koding!

Plotting i Python

I Python kan vi bruke `matplotlib` til å visualisere data i form av grafer og diagrammer. Dette gjør det enklere å forstå sammenhenger og se mønstre i tallmateriale.

Som alltid, når vi begynner å arbeide med en pakke vi ikke har brukt før, må vi laste ned pakken:

```
!pip install matplotlib
```

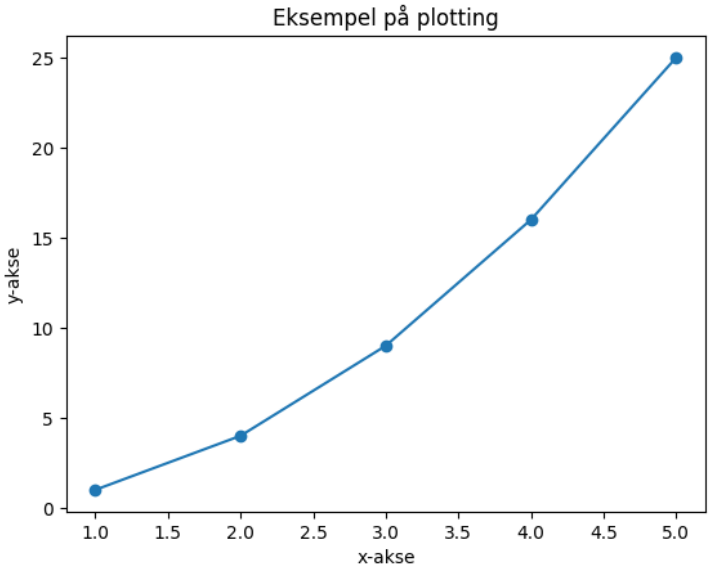
NB: Hvor noen kanskje må skrive %, ikke !.

Eksempel på linjeplott

Forklaring: Med `plt.plot()` kan vi tegne grafer. Vi sender inn to lister: én med x-verdier og én med y-verdier.

Eksempel 1:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

plt.plot(x, y, marker="o")
plt.xlabel("x-akse")
plt.ylabel("y-akse")
plt.title("Eksempel på plotting")
plt.show()
```



Figur 1: Plot til kodeblokken over.

Datavisualisering med Matplotlib

Introduksjon

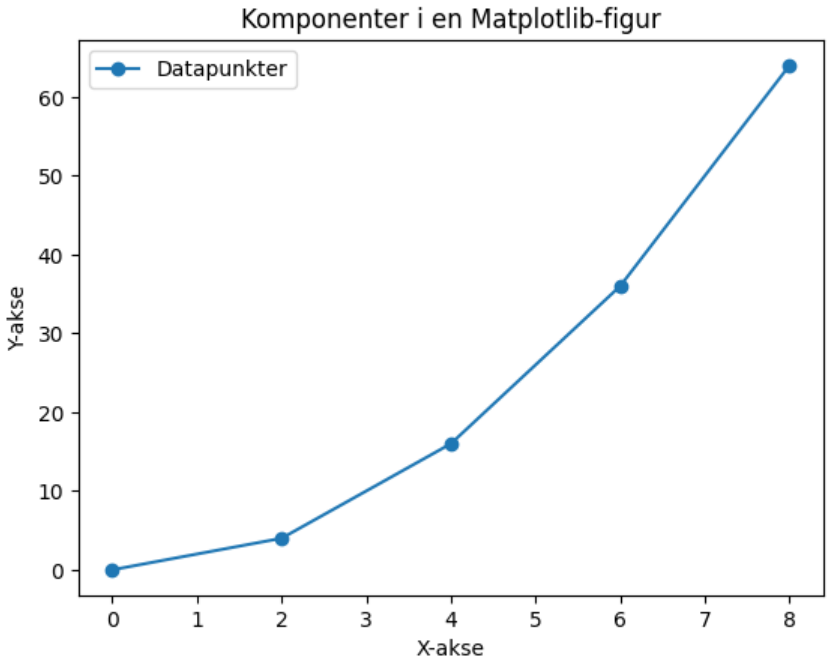
Matplotlib er et populært bibliotek i Python for datavisualisering. Det gir mulighet til å lage alt fra enkle linjediagrammer til mer avanserte figurer som histogrammer, punktdiagrammer og 3D-plott.

Eksempel 2: Figurkomponenter

Hver figur i Matplotlib består av flere elementer: *figure* (hele tegneflaten), *axes* (plottområdet), *axis* (x- og y-akser), samt titler og etiketter. Her er et eksempel, lignende til eksempel 1, men på en litt annen måte:

```
import matplotlib.pyplot as plt

# Samme koordinater
x = [0, 2, 4, 6, 8]
y = [0, 4, 16, 36, 64]

# Lager figur og akser
fig, ax = plt.subplots()
ax.plot(x, y, marker='o', label="Datapunkter")

# Tittel til plottet
ax.set_title("Komponenter i en Matplotlib-figur")
ax.set_xlabel("X-akse")
ax.set_ylabel("Y-akse")

plt.legend() # <- viser forklaringsboksen
plt.show()
```

Resultatet vises i figur 2.



Figur 2: Komponentene i en Matplotlib-figur: akser, etiketter, tittel og datapunkter.

Pandas, NumPy og Matplotlib

I dette eksemplet kombinerer vi tre viktige Python-bibliotek for dataanalyse: **Pandas** for å håndtere datasett, **NumPy** for å gjøre numeriske beregninger, og **Matplotlib** for å visualisere resultatene.

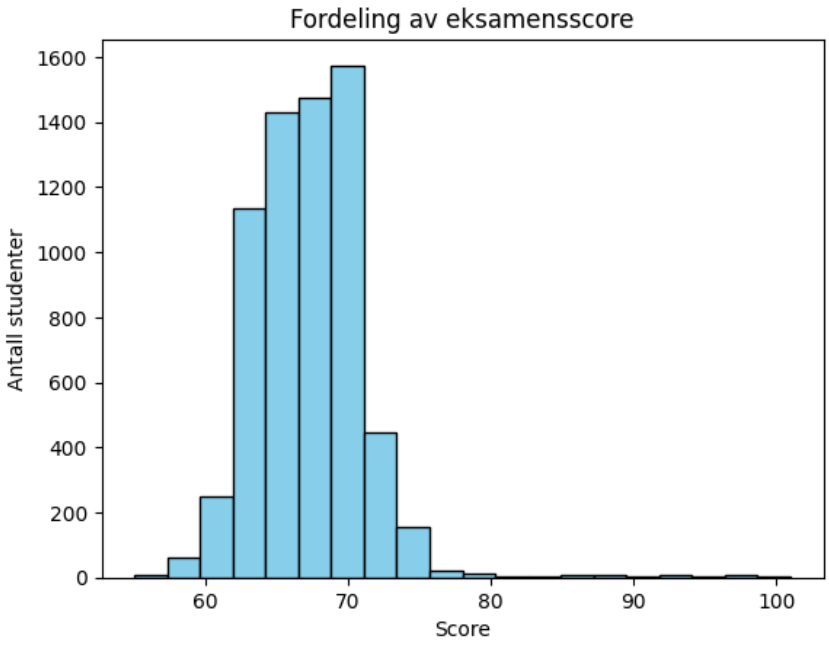
Datasettet som brukes i neste eksempel er `StudentPerformanceFactors.csv`, som inneholder ulike faktorer som potensielt kan påvirke elevers prestasjoner. Dette datasettet finner dere via hjemmesiden til kurset.

Eksempelkode

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Les inn datasettet ved hjelp av Pandas
df = pd.read_csv("StudentPerformanceFactors.csv")

# 1. df.head() til å gi oversikt
df.head()

# 2. NumPy's funksjon for å beregne gjennomsnittlig score
avg_score = np.mean(df['Exam_Score'])
print("Gjennomsnittlig eksamensscore:", avg_score)

# 3. Visualiser fordelingen av score med Matplotlib
plt.hist(df['Exam_Score'], bins=20, color='skyblue', edgecolor='black')
plt.title("Fordeling av eksamensscore")
plt.xlabel("Score")
plt.ylabel("Antall studenter")
plt.savefig("exam_scores.png")
plt.show()
```

Resultat

Histogrammet i figur 3 viser hvordan eksamensscore fordeler seg blant studentene. Gjennomsnittlig score ble beregnet ved hjelp av NumPy, mens selve visualiseringen ble laget med Matplotlib (som dere ser under her).



Figur 3: Fordeling av eksamensscore i datasettet.

Oppsummering

Matplotlib er et fleksibelt verktøy for datavisualisering i Python. Det er spesielt nyttig i rapportskriving fordi figurer kan eksporteres i høy kvalitet og enkelt settes brukes i rapporter/doku-

menter. I tillegg, gir det muligheten til å forklare trender i data som er vanskelig å se i tabulær form.

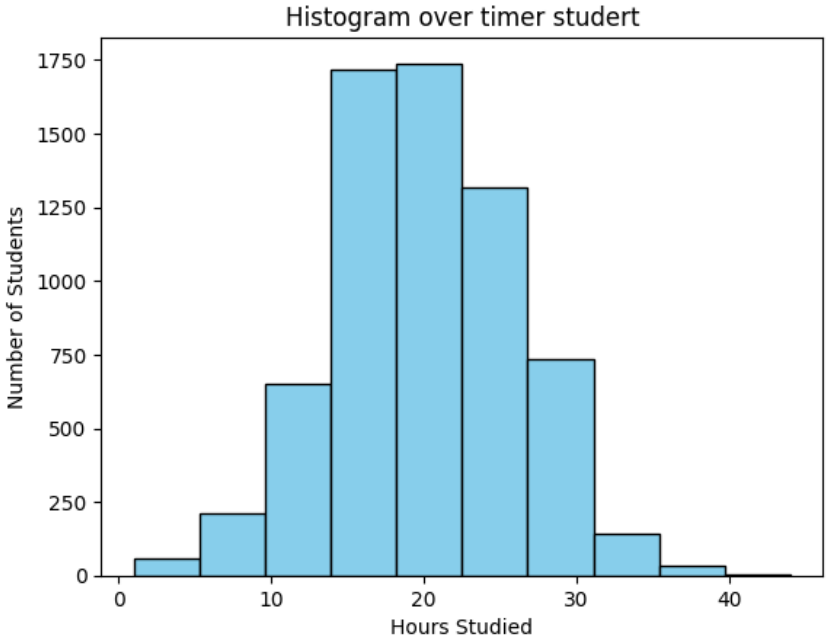
Andre diagrammer

Vi fortsetter med datasettet `StudentPerformanceFactors.csv`, og ser på ulike måter å visualisere dataen på.

1. Histogram over timer studert

Et histogram gir en oversikt over fordelingen av hvor mange timer studentene studerer, koden under generer plottet vi kan se i Figur 4.

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

# Les inn datasettet
df = pd.read_csv("StudentPerformanceFactors.csv")

# Lag histogram
plt.hist(df["Hours_Studied"], bins=10, color="skyblue", edgecolor="black")
plt.xlabel("Hours Studied")
plt.ylabel("Number of Students")
plt.title("Histogram over timer studert")
plt.show()
```



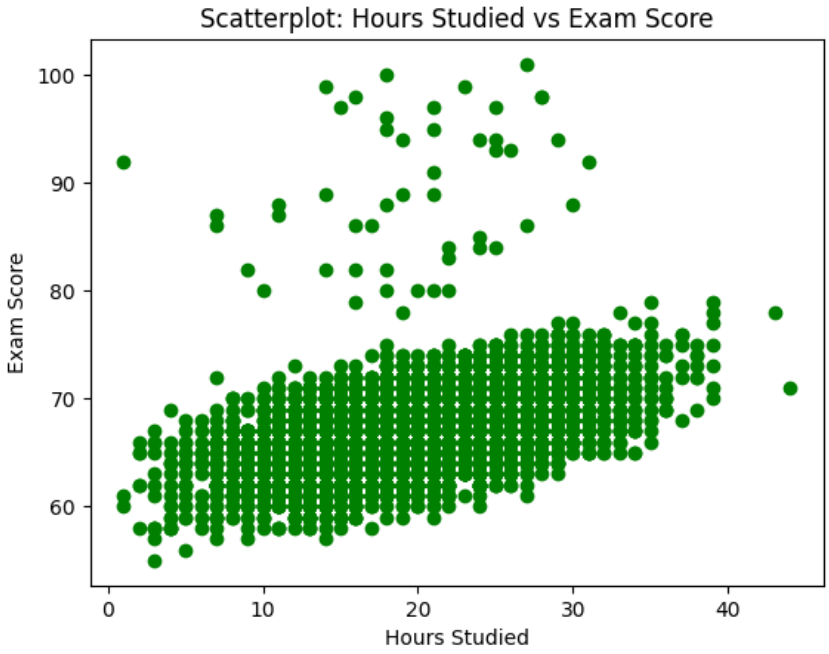
Figur 4: Histogram som viser hvor mye studentene studerer.

Tips: Prøv å gjenskap koden selv, og se hva hver enkelt linje gjør.

2. Scatterplot: Timer studert vs. eksamensresultat

Et scatterplot kan vise sammenhengen mellom antall timer studert og `Exam_Score`, plottet ser vi i 5.

```
# Lager et scatterplot
plt.scatter(df["Hours_Studied"], df["Exam_Score"], color="green")
plt.xlabel("Hours Studied")
plt.ylabel("Exam Score")
plt.title("Scatterplot: Hours Studied vs Exam Score")
plt.show()
```



Figur 5: Histogram som viser hvor mye studentene studerer.

Tenk selv: Still deg selv spørsmålet, *hva ser vi egentlig her*? Hva sier dette plottet oss?

3. Kakediagram: Fordeling av foreldreinvolvering

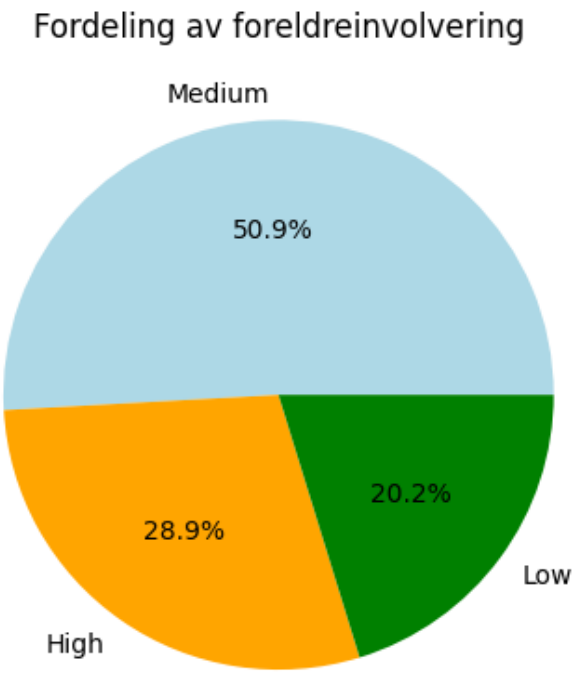
Først, så inspiserer vi kolonnen `Parental_Involvement` i datasettet vårt `df`.

```
df["Parental_Involvement"].unique()
```

Gjør at vi får ut følgende:
`array(['Low', 'Medium', 'High'], dtype=object)`

Med andre ord, det er en **kategorisk** variabel, med tre ulike verdier. Et kakediagram (pie chart) viser hvordan foreldreinvolvering er fordelt blant studentene. Diagrammet kan gi en rask oversikt over hvor mange som har lav, medium eller høy involvering.

```
# Lager et kakediagram
parent_counts = df["Parental_Involvement"].value_counts()
plt.pie(parent_counts, labels=parent_counts.index, autopct="%1.1f%%", colors=["
    lightblue","orange","green"])
plt.title("Fordeling av foreldreinvolvering")
plt.show()
```



Figur 6: Kakediagram som viser fordelingen av foreldreinvolvering blant studentene.

4. Matematiske funksjoner

En sentral del av matematikk og naturvitenskap er de trigonometriske funksjonene **cosinus** og **sinus**. Disse funksjonene beskriver forhold mellom kateter og hypotenusen i en rettvinklet trekant, og de er fundamentale i alt fra geometri til fysikk, signalbehandling, bølgefenomener, statistikk og datavitenskap..

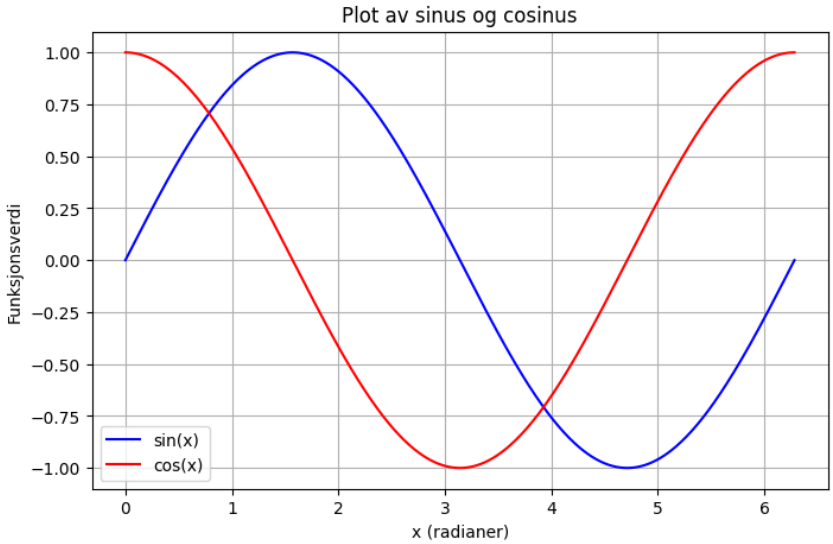
- $\sin(x)$ beskriver y-koordinaten til et punkt på enhetssirkelen (sirkelen med radius 1).
- $\cos(x)$ beskriver x-koordinaten til samme punkt.

Med andre ord: når vi beveger oss rundt en sirkel med radius 1, så er sinus og cosinus bare projeksjoner av dette punktet ned på henholdsvis y- og x-aksen. Dette gir funksjonene deres karakteristiske bølgeform.

Ved å plotte sinus og cosinus kan vi enkelt visualisere denne bølgeformen, og se hvordan de to funksjonene henger sammen — de er forskjøvet (faset) med $\frac{\pi}{2}$ radianer i forhold til hverandre. Dette er nok noe dere har hørt fra matematikken, men det hjelper veldig å visualisere hva dette faktisk betyr. Det kan vi bruke Python til!

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

# Lager en array med 500 verdier fra 0 til 2*pi
x = np.linspace(0, 2*np.pi, 500)

# Beregner sinus og cosinus
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plotter
plt.figure(figsize=(8,5))
```

```
plt.plot(x, y_sin, label="sin(x)", color="blue")
plt.plot(x, y_cos, label="cos(x)", color="red")
plt.title("Plot av sinus og cosinus")
plt.xlabel("x (radianer)")
plt.ylabel("Funksjonsverdi")
plt.legend()
plt.grid(True)
plt.show()
```



Figur 7: Visualisering av de trigonometriske funksjonene $\sin(x)$ og $\cos(x)$. Vi ser at de er like i form, men forskjøvet med en kvart periode ($\pi/2$).

Visualiseringer som dette er nyttige fordi de gjør det lettere å forstå abstrakte matematiske sammenhenger. Det er langt enklere å se faseforskyvningen og periodisiteten til sinus og cosinus når de tegnes ut, enn kun å lese definisjonene. Dette er et godt eksempel på hvordan matematikk og dataverktøy spiller sammen for å skape innsikt.

Økonomisk anvendelse - Tilbud og etterspørsel

Tilbud og etterspørsel er grunnleggende begreper i økonomi som beskriver hvordan prisen på en vare bestemmes. Tilbudet øker gjerne med prisen, mens etterspørselen reduseres når prisen stiger. Matematisk er det interessant fordi vi kan modellere pris og mengde med funksjoner og finne likevektspunktet.

1. Definere tilbud og etterspørsel som funksjoner

Først definerer vi funksjonene for tilbud og etterspørsel, ved hjelp av `def min_funksjon():`, som vi lærte om i starten av kurset.

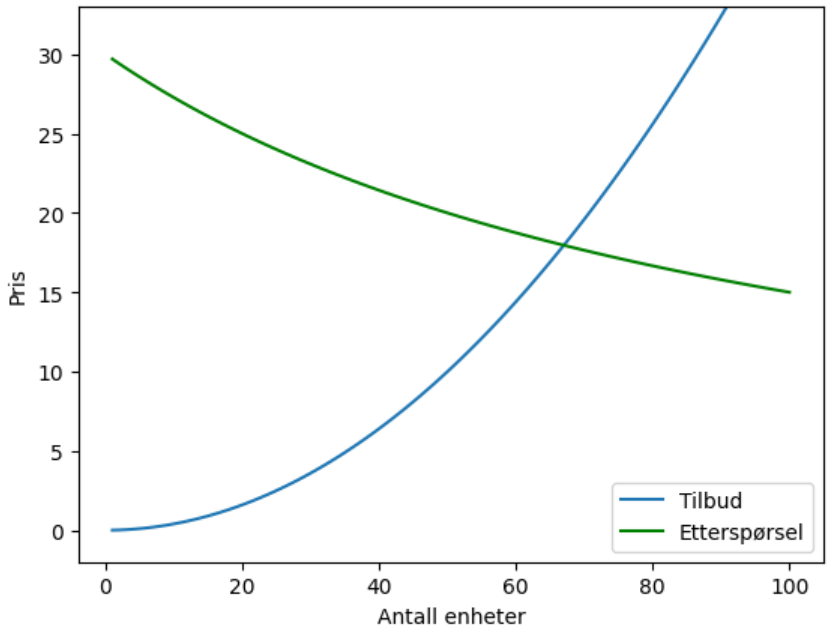
```
def supply(x):
    """Tilbudskurve: pris øker med kvadratet av antall enheter"""
    return (x**2) * (1/250)

def demand(x):
    """Etterspørselskurve: pris reduseres med økende antall enheter"""
    return 3000 / (100 + x)
```

Her representerer `x` antall enheter, `supply(x)` viser hvordan prisen øker med økt tilbud, og `demand(x)` viser hvordan prisen reduseres når flere ønsker varen.

2. Plotting av tilbud og etterspørsel

Vi kan visualisere disse funksjonene for å se hvor de krysser hverandre (likevekten):

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

# Tegner 100 punkter fra 1 til 100
x = np.linspace(1, 100, 100) # np.linspace(*fra*, *til*, *antall tall*)

# Plotter tilbudskurven
plt.plot(x, supply(x), label='Tilbud')

# Plotter etterspørselskurven
plt.plot(x, demand(x), color='green', label='Etterspørsel')

# Legger til forklaring, i nedre høyre hjørne
plt.legend(loc='lower right')

# Setter akselabels
plt.xlabel('Antall enheter')
plt.ylabel('Pris')

# Setter y-akseområde for å se likevekten tydelig
plt.ylim(-2, 33)

# Her kan du legge til plt.show() når du kjører koden
plt.show()
```



Figur 8: Tilbud- og etterspørselskurven.

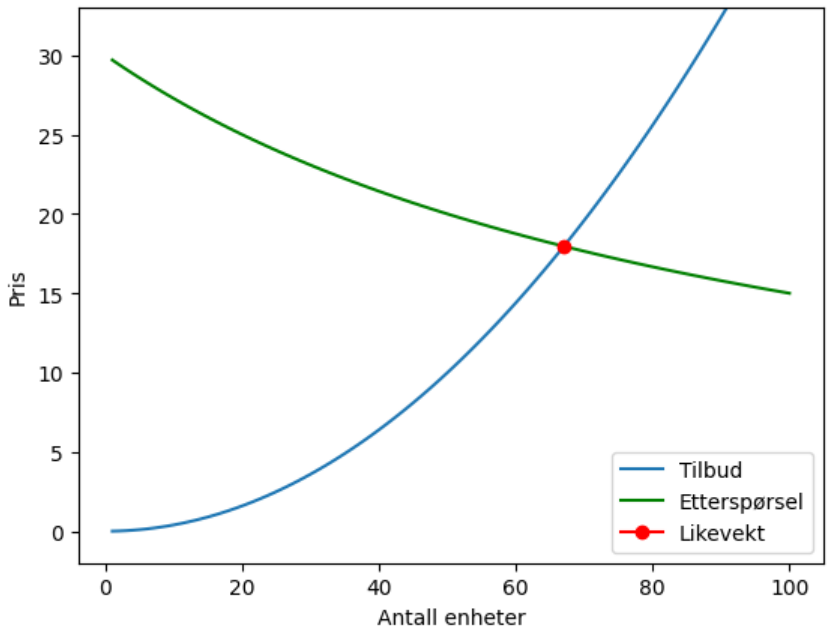
Som vi ser i Figur 8 så er likevekten (altså, hvor linjene krysses) rundt antall enheter er lik 67, og til en pris rundt 18.

Hvordan vi løser dette, helt nøyaktig, skal vi lære om senere i kurset ved hjelp av pakken `SymPy`. Men, vi kan likevel **"highlighte"** hvor likevekten er ved hjelp av `matplotlib`.

3. Ekstra: Viser hvor tilbud = etterspørsel

Vi kan finne omtrent hvor kurvene krysser hverandre. Som vi så i plottet over, tror vi at det er rundt hvor `Antall enheter` ≈ 67.

```
# Plotter tilbudskurven
plt.plot(x, supply(x), label='Tilbud')
# Plotter etterspørselskurven
plt.plot(x, demand(x), color='green', label='Etterspørsel')
# Plotter likevekten (der det ser ut som den er), rundt 67. Prøv andre verdier selv!
plt.plot(67, supply(67), marker='o', color='red', label='Likevekt')

# Legger til forklaring, i nedre høyre hjørne
plt.legend(loc='lower right')

# Setter akselabels
plt.xlabel('Antall enheter')
plt.ylabel('Pris')

# Her kan du legge til plt.show() når du kjører koden
plt.show()
```



Figur 9: Tilbud- og etterspørselskurven.

Her ser vi at likevekten ser ut til å være cirka ved `Antall enheter` ≈ 67.
Dette gjør det tydelig hvor markedet *balanserer*, og viser hvordan matematikk og økonomi henger sammen på en visuell måte.

Del 1: Flervalgsoppgaver - Visualisering og datastruktur

1. Hva vil følgende kode printe/utføre?

```
import pandas as pd

df = pd.DataFrame({
    "Navn": ["Alice", "Bob", "Charlie"],
    "Poeng": [10, 15, 12]
})
print(df["Poeng"].mean())

```

   a) 10
   b) 12.33
   c) 15
   d) Feilmelding

2. Hvilken funksjon brukes for å lage et linjediagram i matplotlib?

   a) plt.bar()
   b) plt.plot()
   c) plt.scatter()
   d) plt.hist()

3. Hva gjør denne koden?

```
import matplotlib.pyplot as plt

x = [1, 2, 3]
y = [4, 5, 6]
plt.scatter(x, y)
plt.show()

```

   a) Lager et linjediagram
   b) Lager et scatterplot
   c) Lager et histogram
   d) Lager et kakediagram

4. Hva vil følgende kode gjøre?

```
import numpy as np
x = np.arange(0, 10, 2)
print(x)
```

   a) $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$
   b) $[0, 2, 4, 6, 8]$
   c) $[2, 4, 6, 8, 10]$
   d) Feilmelding

5. Hvilket av disse er korrekt måte å lage et kakediagram fra en dictionary?

```
import matplotlib.pyplot as plt
data = {"Røde": 5, "Blå": 3, "Grønn": 7}
# Alternativer:
# A: plt.pie(data.values(), labels=data.keys())
# B: plt.plot(data)
# C: plt.bar(data)
# D: plt.scatter(data)
```

a) A
b) B
c) C
d) D

6. Hva gjør funksjonen nedenfor?

```
def dobbel(x):
    return x * 2

print(dobbel(5))
```

a) Printer 5
b) Printer 10
c) Printer 'dobbel'
d) Gir `None`

7. Hva vil følgende kode gjøre?

```
import matplotlib.pyplot as plt
verdier = [3, 7, 2, 5]
plt.bar(range(len(verdier)), verdier)
plt.show()

```

a) Lager et linjediagram
b) Lager et stolpediagram
c) Lager et scatterplot
d) Lager et kakediagram

8. Hva gjør koden nedenfor?

```
frukter = ["eple", "banan", "appelsin"]
for f in frukter:
    print(f.upper())

```

a) Printer fruktnavnene med store bokstaver
b) Teller antall frukter
c) Legger til fruktnavn i en liste
d) Ingenting

Del 2: Hva blir output? - Visualisering og datastruktur

1.
```
import matplotlib.pyplot as plt
x = [1,2,3,4]
y = [2,4,6,8]
plt.plot(x, y)
plt.title("Ukjent tittel")
plt.show()
```

2.
```
import numpy as np
a = np.array([1,2,3])
print(a*3)
```

3.
```
def kvadrat(x):
    return x**2

print(kvadrat(6))
```

4.
```
verdier = [5,2,7,10]
plt.hist(verdier, bins=8, color="purple")
plt.show()
```

Del 3 - Hva vi har lært til nå

Her vil dere bli utfordret på mye av det vi har lært til nå.

Oppgave

I denne oppgaven skal du bruke alt vi har lært til nå. Du skal jobbe med boligprisdata for Oslo, hentet fra SSB, `Median_totalpris_boliger_Oslo.csv`. Fila finner dere på hjemmesiden til kurset.

1. **Lag en funksjon for innlasting av data** Skriv en **funksjon** som bruker `pandas` til å laste inn CSV-filen.
2. **Databehandling med `pandas`** Lag en funksjon som tar inn DataFrame-en og gjør følgende:
   - Finn gjennomsnittlig boligpris per år (kolonnevis).
   - Finn gjennomsnittlig boligpris per bydel (radvis).
3. **Visualisering med `matplotlib`** Lag en eller flere funksjoner som lager følgende plots:
   (a) Et funksjon som tar inn navnet til en bydel, og lager et linjediagram som viser prisutviklingen til den bydelen fra 2018–2023.
   (b) Et stolpediagram som sammenligner prisene for alle bydeler i et valgt år.
4. **Ekstra utfordring:** Bruk funksjonene dine til å finne ut hvilken bydel som har hatt størst prosentvis vekst i boligpris fra 2018 til 2023. Visualiser dette i et eget plot.

Løsningsforslag Del 1: Flervalgsoppgaver

1. Gjennomsnittet av $[10, 15, 12]$ er $\frac{10+15+12}{3} = 12.33$. **Riktig svar: b) 12.33**

2. For linjediagram i matplotlib bruker vi `plt.plot()`. **Riktig svar: b) plt.plot()**

3. Koden lager et scatterplot av punktene (1,4), (2,5), (3,6). **Riktig svar: b) Lager et scatterplot**

4. `np.arange(0, 10, 2)` gir en array fra 0 til 10 (ikke med 10), med steg 2: $[0, 2, 4, 6, 8]$. **Riktig svar: b) [0,2,4,6,8]**

5. For kakediagram bruker vi `plt.pie()`. **Riktig svar: a) A**

6. Funksjonen returnerer input multiplisert med 2. For `dobbel(5)` gir dette 10. **Riktig svar: b) Printer 10**

7. Koden lager et stolpediagram (`plt.bar`). **Riktig svar: b) Lager et stolpediagram**

8. `f.upper()` gjør teksten om til store bokstaver. Koden printer: `EPLE`, `BANAN`, `APPELSIN`. **Riktig svar: a) Printer fruktnavnene med store bokstaver**

Løsningsforslag Del 2: Hva blir output?

1.
```
    import matplotlib.pyplot as plt
    x = [1,2,3,4]
    y = [2,4,6,8]
    plt.plot(x, y)
    plt.title("Ukjent tittel")
    plt.show()

```

Dette gir et linjediagram med punktene (1,2), (2,4), (3,6), (4,8) og tittelen *"Ukjent tittel"* på toppen.

2.
```
    import numpy as np
    a = np.array([1,2,3])
    print(a*3)

```

Her multipliseres hvert element med 3. Output blir: `[3 6 9]`

3.
```
    def kvadrat(x):
        return x**2

    print(kvadrat(6))

```

Her returneres kvadratet av 6, altså $6^2 = 36$. Output: `36`

4.
```
    verdier = [5,2,7,10]
    plt.hist(verdier, bins=8, color="purple")
    plt.show()

```

Dette lager et histogram med 8 bins og lilla farge, som viser fordelingen av verdiene $[5, 2, 7, 10]$.

Løsningsforslag Del 3

Oppgave 1

Vi laster inn dataen ved hjelp av Pandas, og da kan vi lage en egen funksjon.

```
import pandas as pd

# Definerer funksjonen vår
def last_data(filbane):
    data = pd.read_csv(filbane)
    return data

# Kaller på funksjonen
df = last_data('data/Median_totalpris_boliger_Oslo.csv')
df.head()
```

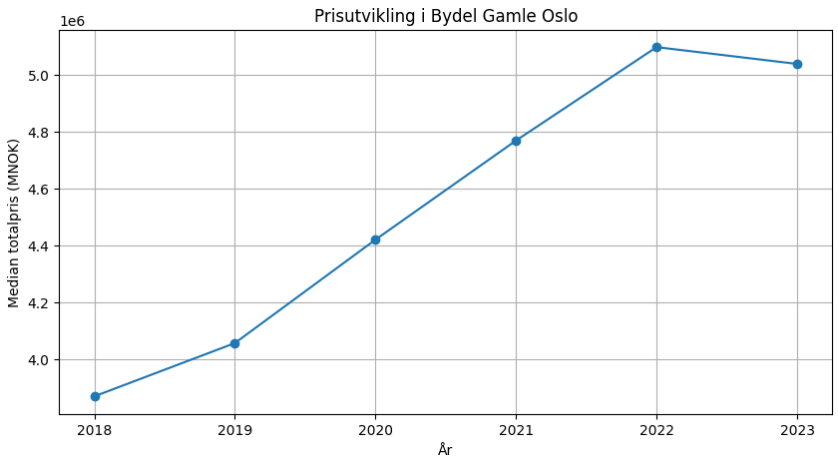
Oppgave 2

```
import pandas as pd

# Finn gjennomsnittlig boligpris per år (kolonnevis)
gjennomsnitt_per_aar = df.mean(numeric_only=True).to_frame(name="
    Gjennomsnittpris")
```

```python
gjennomsnitt_per_aar.index.name = "År"

print("Gjennomsnittlig boligpris per år:")
print(gjennomsnitt_per_aar)
print()

# Finn gjennomsnittlig boligpris per bydel (radvis)
gjennomsnitt_per_bydel = df.set_index("Geografi").mean(axis=1, numeric_only=True).to_frame(name="Gjennomsnittpris")
gjennomsnitt_per_bydel.index.name = "Bydel"

print("Gjennomsnittlig boligpris per bydel:")
print(gjennomsnitt_per_bydel)
```

Oppgave 3a

```python
def plot_bydel(bydel):
    # Plot prisutvikling for bydel
    priser = df[df["Geografi"] == bydel].iloc[0, 1:]  # Henter prisene for valgt 'bydel'
    priser.index = priser.index.astype(int)  # Konverterer kolonneindeks til int for plotting

    plt.figure(figsize=(10, 5))
    plt.plot(priser.index, priser.values, marker='o')
    plt.title(f"Prisutvikling i {bydel}")
    plt.xlabel("År")
    plt.ylabel("Median totalpris (MNOK)")

    plt.grid(True)
    plt.show()

# Her kan vi skrive inn "Bydel Gamle Oslo" eller "Bydel Stovner" og så videre.
plot_bydel("Bydel Gamle Oslo")
```

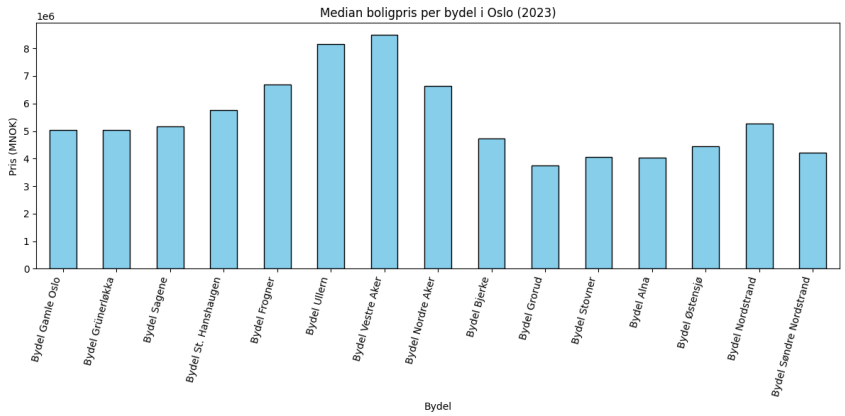


Figur 10: Plot fra oppgave 3a.

Enkel øvelse: gå i gjennom løsningsforslaget over og bli kjent med koden. Hva gjør hver enkelt linje?

Oppgave 3b

```python
import matplotlib.pyplot as plt

# Lager en funksjon som tar inn data, og år (som streng)
def plott_bydelspriser(df, år):
    if år not in df.columns:
        raise ValueError(f"Året {år} finnes ikke i datasettet. Tilgjengelige år: {list(df.columns[1:])}")

    # Henter kun kolonnene vi trenger
    data = df[["Geografi", år]].set_index("Geografi")

    # Fjerner totalen for Oslo
    data = data.drop("Oslo i alt", errors="ignore")

    # Plot
    plt.figure(figsize=(12,6))
    data[år].plot(kind="bar", color="skyblue", edgecolor="black")

    plt.title(f"Median boligpris per bydel i Oslo ({år})")
    plt.ylabel("Pris (MNOK)")
    plt.xlabel("Bydel")
    plt.xticks(rotation=75, ha="right")
    plt.tight_layout()
    plt.show()

# Eksempelbruk
plott_bydelspriser(df, "2023")
```



Figur 11: Plot fra oppgave 3b.

Enkel øvelse: gå i gjennom løsningsforslaget over og bli kjent med koden. Hva gjør hver enkelt linje?