```
# %% [markdown]
# # Mappeoppgave 2

# %% [markdown]
# #### Informasjon om oppgaven
# Når du besvarer oppgaven, husk:
# - les oppgaveteksten nøye
# - kommenter koden din
# - sett navn på akser og lignende i figurene
# - skriv hvor du har hentet kodesnutter fra, hvis du gjør det
# - bruk engelske variabelnavn og vær konsistent med hvordan du bruker store og
små bokstaver
# - bruk mest mulig funksjoner for ting som kan repeteres
# - En kort kode kan være en bra kode, så ikke gjør det mer komplisert enn det
spørres om.
#
# Du kan få full pott uten å svare på oppgaven som er markert "ekstrapoeng". Du
blir likevel belønnet for denne (dvs. hvis du har noen feil og får 45 poeng
totalt, så kan du få en høyere poengsum hvis du også har svart på "ekstrapoeng".

#
# #### Innlevering av oppgavene
# Du skal levere begge mappene samtidig (det vil si denne oppgaven og mappe 1).
Innleveringsfristen er 6 desember kl 13:00.
# Begge oppgavene skal leveres i github (som jupyter-fil) og wiseflow (som PDF).
Bruk navnet  "SOK-1003-eksamen-2022-mappe2" på filene.
# - For github: Husk å gi meg (brukernavn "okaars") tilgang til
github-reposetoriet deres. Hvis dere har satt reposetoriet til public (anbefales
ikke), må dere dele lenken til dette på ole.k.aars@uit.no
# - For wiseflow: En person fra hver gruppe (for hver mappeoppgave), leverer
inn. Ved innlevering kan du krysse av hvem som er på gruppen din
#
# Se generell informasjon om hvordan man leverer oppgaven <th><a
href='https://uit-sok-1003-h22.github.io/semesteroppgave.html'>her</a>.
#
# <b> NB!:</b> En person fra gruppa må <a
href='https://docs.google.com/forms/d/e/1FAIpQLSeljUukzUU5d-VbxyM4C0x4WUplhUKBCU
5wIpPZl_bP5kN71A/viewform?usp=sf_link'> fylle ut dette skjemaet </a>  for å
melde om hvem som er på gruppa. Dere vil i etterkant motta en epost om tidspunkt
for presentasjon.
#
# #### Presentasjon
# Presentasjonen innebærer en kort gjennomgang av oppgaven (10-15 min)
etterfulgt av kommentarer fra meg (10-15 min). Alle gruppemedlemmer skal bidra
til presentasjonen. Det er anbefalt å laste opp besvarelsen på github forut for
presentasjonen (helst to dager før) slik at jeg har mulighet til å lese gjennom.
Dere vil ha mulighet til å endre besvarelsen etter presentasjonen, frem til
endelig innlevering 6 desember.
#
#
#

# %% [markdown]
# ### Oppgave 1 (10 poeng)
```

```
# %% [markdown]
# a) Vi skal spille et spill der vi kaster en terning 6 ganger. Lag en funksjon
med "for-løkke" som printer alle terningene som har blitt kastet. Du kan bruke
`np.random.randint()` til å lage tilfeldige tall

# %% [markdown]
#
-------------------------------------------------------------------------------
----------------------
# Fra student:
# Har brukt 'Github copilot' som hjelp
#
-------------------------------------------------------------------------------
----------------------

# %%
import numpy as np
#definer terningkast
def game():
     dice = np.random.randint(1, 7)
     return dice

#definer antall kast
for x in range(6):
    #print terningkast
    print(game())

# %% [markdown]
# b) Juster den samme funksjonen slik at den lagrer tallene i en liste før den
printer ut selve listen. Dere kan kalle denne listen for `lot_numbers`. Dere kan
vurdere å bruke `append()` som del av funksjonen.

# %%
#Tom liste
lot_numbers = []
#Antall terningkast
for x in range(6):
    #Legg til terningkast i listen
    lot_numbers.append(game())
#print listen
print(lot_numbers)

# %% [markdown]
# c) Juster den samme funksjonen slik at den har to argument. Disse argumentene
er to terningverdier som du "tipper" blir kastet. Bruk `if`, `else` og `elif`
til å generere vinnertall. Resultatet fra funksjonen skal printe ut ulike
setninger avhengig av om man får 0, 1 eller 2 rette. Setningene velger du selv,
men de skal inneholde tallene som du tippet, og tallene som ble trukket.

# %%
#Tom liste
lot_numbers_2 = []
#Liste med tall vi tipper på
```

```
lot_list = [5,5,1,4,3,2]

#Antall terningkast
for x in range(6):
    #Legg til terningkast i listen
    lot_numbers_2.append(game())

#for loop for å sammenligne gjetninger og terningkast
for x in range(6):
    #hvis gjetning og terningkast er like
    if lot_numbers_2[x] == lot_list[x]:
        print("Guessed " + str(lot_list[x]) + ", rolled " +
str(lot_numbers_2[x]) + "; Winner, winner, chicken dinner!")
        #hvis gjetning og terningkast er ulike
    else:
        print("Guessed " + str(lot_list[x]) + ", rolled " +
str(lot_numbers_2[x]) + "; Loser!")


# %% [markdown]
# ### Oppgave 2 (10 poeng)
# a) Du har nå begynt å spille lotto i stedet, og satser alt på ett vinnertall.
Lag en while-løkke som printer ut tall helt til du har trukket riktig tall (som
du definerer selv). For enkelthets skyld kan du begrense utfallsrommet av
trekningene til mellom 0-30.

# %%
#tall vi gjetter på
tall = 8
#tall vi ruller
roll = np.random.randint(0, 31)

#while loop som kjører så lenge vi ikke har rullet riktig tall
while roll != tall:
    #printe ut tall vi ruller
    print("You guessed " + str(tall) + ", but rolled " + str(roll) + ". Try
again!")
    #tall vi ruller
    roll = np.random.randint(0, 31)
if roll == tall:
    #printe ut tall som er riktig
    print("You guessed " + str(tall) + ", and rolled " + str(roll) + ". You
win!!!")



# %% [markdown]
# b) Lag et plot av den while-løkken du nettopp lagde. Man blir belønnet om man;

# - bruker `scatter`;
# - lager plottet dynamisk (dvs at hver trekning vises hver for seg, og at
x-aksen endrer seg etter en gitt verdi);
# - viser hvor når siste trekningen blir gjort (dvs at den vises kun når du har
trukket vinnertallet).
```

```
#
# Avhengig av hvordan du lager figuren din kan du får bruk for å importere
pakkene `Ellipse`, `display`, `clear_output`.

# %%
#pakker som du kan få bruk for
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from IPython.display import display, clear_output
import time

trekk = []
tall_2 = 8
roll_2 = np.random.randint(0, 31)
length = []

#while loop som kjører så lenge vi ikke har rullet riktig tall
while roll_2 != tall_2:
    #printe ut tall vi ruller
    print("You guessed " + str(tall_2) + ", but rolled " + str(roll_2) + ". Try
again!")
    #legge til tall vi ruller i en liste
    trekk.append(roll_2)
    #Lager liste over hvor mange trekk det har tatt
    length.append(len(trekk)-1)
    #tall vi ruller
    roll_2 = np.random.randint(0, 31)
    #Størrelse på figuren
    plt.subplots(figsize=(18, 6))
    #linje som viser hvor vi gjetter
    #plt.axhline(y = 8, color = 'r', linestyle = '-')
    #navn på x-aksen
    plt.xlabel("Antall tall trukket")
    #navn på y-aksen
    plt.ylabel("Trukket tall")
    #tittel på figuren
    plt.title("Lotto", fontsize=20)
    #plotter tall vi ruller
    plt.scatter(length, trekk, s = 25, color = "red",)
    plt.scatter(length[-1], trekk[-1], s = 100, color = "blue",)
    plt.plot(trekk)
    #viser plot
    plt.show()
    #setter en pause på 1 sekund
    time.sleep(0.175)
    #clearer plot
    clear_output(wait=True)
if roll_2 == tall_2:
    #printe ut tall som er riktig
    print("You guessed " + str(tall_2) + ", and rolled " + str(roll_2) + ". You
win!!!")
    #legger til tall vi ruller i en liste
    trekk.append(roll_2)
    #Lager liste over hvor mange tall vi har trukket
```

```
    length.append(len(trekk)-1)
    #Figur størrelse
    plt.subplots(figsize=(18, 6))
    #linje som viser hvilket tall vi gjetter på
    plt.axhline(y = 8, color = 'r', linestyle = '-')
    #plt.scatter(length[x], trekk[x], color = "red")
    #navn på x aksen
    plt.xlabel("Antall tall trukket")
    #navn på y aksen
    plt.ylabel("Trukket tall")
    #tittel på figur
    plt.title("Lotto", fontsize = 20)
    #plotter tall vi ruller
    plt.scatter(length, trekk, s = 25, color = "red",)
    plt.scatter(length[-1], trekk[-1], s = 200, color = "blue")
    plt.plot(trekk)
    #lager en pil og skriver "Winner" så vi vet hvor vinner tallet er. Fikk
hjelp fra https://www.geeksforgeeks.org/how-to-add-text-to-matplotlib/
    plt.annotate('Winner!', xy=(length[-1], trekk[-1]), xytext=(length[-1]+1.5,
trekk[-1]+1), fontsize=12,
            arrowprops=dict(facecolor='green', shrink=0.05))
    #Bruker legende til å utrope vinnertallet
    plt.legend(['The winning number is ' + str(tall_2) + ', you won!!!'],
loc='upper center', fontsize = 25, frameon = True, edgecolor = "black")
    #viser plot
    plt.show()

# %% [markdown]
# c) Ekstrapoeng: gjør det samme som i (b), men lag et histogram som vises ved
siden av. Dette histogrammet skal vise hvor mange ganger de ulike tallene ble
trekt. Bruk `plt.hist` til dette. Husk at du må definere figur og akseobjekt
først.

# %%
trekk_2 = []
tall_3 = 8
roll_3 = np.random.randint(0, 31)
length_2 = []

fig3, (ax1,ax2) = plt.subplots(1, 2, figsize=(18, 6))

#while loop som kjører så lenge vi ikke har rullet riktig tall
while roll_3 != tall_3:


    #printe ut tall vi ruller
    print("You guessed " + str(tall_3) + ", but rolled " + str(roll_3) + ". Try
again!")
    #legge til tall vi ruller i en liste
    trekk_2.append(roll_3)
    #Lager liste over hvor mange trekk vi har gjort
    length_2.append(len(trekk_2)-1)
    #tall vi ruller
    roll_3 = np.random.randint(0, 31)
```

```
    ax1.scatter(length_2, trekk_2, s = 25, color = "red",)
    ax1.scatter(length_2[-1], trekk_2[-1], s = 100, color = "blue",)
    ax1.plot(trekk_2)
    ax1.set_xlabel("Antall tall trukket")
    ax1.set_ylabel("Trukket tall")
    ax1.set_title("Lotto", fontsize=15)
    ax1.axhline(y = 8, color = 'r', linestyle = '-')

    ax2.hist(trekk_2, bins = 30, color = "red")
    ax2.set_xlabel("Trukket tall")
    ax2.set_ylabel("Antall ganger trukket")
    ax2.set_title("Lotto histogram", fontsize = 15)
    ax2.set_xticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30])
    #viser plot

    #setter en pause
    time.sleep(0.175)
    #clearer plot
    clear_output(wait=True)




if roll_3 == tall_3:
    #printe ut tall som er riktig
    print("You guessed " + str(tall_3) + ", and rolled " + str(roll_3) + ". You
win!!!")
    #legger til tall vi ruller i en liste
    trekk_2.append(roll_3)
    #Lager liste over hvor mange trekk vi har gjort
    length_2.append(len(trekk_2)-1)
    ax1.scatter(length_2, trekk_2, s = 25, color = "red",)
    ax1.scatter(length_2[-1], trekk_2[-1], s = 200, color = "blue")
    ax1.plot(trekk_2)
    ax1.set_xlabel("Antall tall trukket")
    ax1.set_ylabel("Trukket tall")
    ax1.set_title("Lotto", fontsize = 15)
    ax1.axhline(y = 8, color = 'r', linestyle = '-')

    ax2.hist(trekk_2, bins = 31, color = "red", edgecolor = "black")
    ax2.set_xlabel("Trukket tall")
    ax2.set_ylabel("Antall ganger trukket")
    ax2.set_title("Lotto histogram", fontsize = 15)
    ax2.set_xticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30])
    #lager en pil og skriver "Winner" så vi vet hvor vinner tallet er. Fikk
hjelp fra https://www.geeksforgeeks.org/how-to-add-text-to-matplotlib/
    ax1.annotate('Winner!', xy=(length_2[-1], trekk_2[-1]),
xytext=(length_2[-1]+1.5, trekk_2[-1]+1), fontsize=12,
            arrowprops=dict(facecolor='green', shrink=0.05))

    #Bruker legende til å utrope vinnertallet
```

```
    ax1.legend(['The winning number is ' + str(tall_3) + ', you won!!!'],
loc='upper center', fontsize = 15, frameon = True, edgecolor = "black")
```

```
# %% [markdown]
# ### Oppgave 3 (20 poeng)

# %% [markdown]
# En bedrift produserer biler. Produktfunksjonen til bedriften defineres slik
$f(L, a, R) = 2RL^{a}$, hvor:
# - `L` er arbeidskraft,
# - `a` er produktiviteten til arbeiderne og
# - `R` er antall robotmaskiner
#
# a) Lag en formel for produktfunksjonen til bedriften og plot den grafisk med
ulike verdier av `L` på x-aksen. Anta `a=0.6` og `R=2`

# %%
#definer produktfunksjonen
def f(L,a,R):
    return (2*R)*L**a

# %%
#kodehjelp fra forelesning 5, sympy
x = np.linspace(0,5,100)

plt.subplots(figsize=(18, 6))

plt.plot(x,f(x,0.6,2), label = "produksjon", color = "blue")
plt.xlabel('Arbeidskraft')
plt.ylabel('Produksjon')
plt.title('Produksjon som funksjon av arbeidskraft')
plt.legend(loc = "upper left", edgecolor = "black", frameon = True)
plt.show()

# %% [markdown]
# b) anta at profittfunksjonen til denne bedriften er ${profit = f(L, a,
R)p-wL-cR-K}$, hvor
# - `w` er månedslønnen til arbeiderne,
# - `c` er kostnaden for robotmaskinene
# - `K` er faste kostnader
# - `p` er utsalgsprisen på bilene.
#
# Anta `a=0.6`,`R=6`, `p=300 000`, `w=100 000`, `c= 1 000 000` og `K=90 000
000`. Plot profittfunksjonen figurativt for antall arbeidere (`L`) mellom 0 og
10 000. Vis profitten i millioner (dvs at du må dele på 1 000 000)

# %%
#definer profittfunksjonen
def profit(L,a,R,p,w,c,K):
    return ((f(L,a,R)*p)-(w*L)-(c*R)-K)
```

```
# %%
#Kodehjelp fra forelesning 5, sympy

#Definerer verdiene til variablene i profittfunksjonen
a1 = 0.6
R1 = 6
p1 = 300000
w1 = 100000
c1 = 1000000
K1 = 90000000
#for å dele på 1 million
m = 1000000

#creating the plot
x = np.linspace(0,10000,100)
#lag figur for senere bruk
fig,ax=plt.subplots(figsize=(18, 6))
ax.set_ylabel('kroner i millioner')
ax.set_xlabel('Antall ansatte')

#plotting the function
plt.plot(x,profit(x,a1,R1,p1,w1,c1,K1)/m,label='profitt')
plt.title('Profitt som funksjon av antall ansatte')
ax.legend(loc='upper right',frameon=True, edgecolor='black')
plt.show()

# %% [markdown]
# c) Plot profitttfunksjonen for antall robostmaskiner `R=[3, 6, 9]` i samme
plot (dvs at tre profittfunksjoner vises sammen). Bruk av "for loops" for å
gjøre dette belønnes

# %%
#Liste med antall robotmaskiner
R1 = [3, 6, 9]

plt.subplots(figsize=(18, 6))

#For loop som kjører gjennom antall robotmaskiner
for i in range(3):
    plt.ylabel('kroner i millioner')
    plt.xlabel('Antall ansatte')

    #plotter profittfunksjonen, med forskjellige robotmaskiner. R[i] er antall
robotmaskiner
    plt.plot(x,profit(x,a1,R1[i],p1,w1,c1,K1)/m,label='profitt robotmaskiner x'
+ str(R1[i]), color = 'C' + str(i))
    plt.title('Profitt som funksjon av antall ansatte')
    plt.legend(loc='lower left',frameon=True, edgecolor='black')
plt.show()

# %% [markdown]
# d) finn profittmaksimum og optimal antall arbeidere ved hjelp av derivasjon
med samme forutsetninger som i (1b). Bruk `sympy`-pakken til dette
```

```
# %%
#Mer hjelp fra forelesning 5, sympy
#-----------------------------------------
#pakker du kan få bruk for
import sympy as sp
from sympy.solvers import solve

#Definerer symboler
L,a,R,p,w,c,K=sp.symbols("L a R p w c K")
#viser profittfunksjonen analytisk, ganger med 1 million for å få original verdi
profit(L,a,R,p,w,c,K)
#Deriverer profittfunksjonen
d_profitt=sp.diff(profit(L,a,R,p,w,c,K),L)
#Setter den deriverte lik null
foc=sp.Eq(d_profitt,0)
#Løser førsteordensbetingelsen
L_max=solve(foc,L)[0]
#Finner analytisk løsning for maksimal profitt
profit_max=profit(L_max,a,R,p,w,c,K)
#Setter inn verdiene til profittfunksjonen
num_dict={a:0.6,R:6,p:300000,w:100000,c:1000000,K:90000000}
#Regner ut arbeidskraft som kreves for å nå maksimal profitt
L_max.subs(num_dict)
#Viser alle verdiene for maksimal profitt samlet
num_dict[L]=L_max.subs(num_dict)
#Viser profittmaksimum
profit_max_num=float(profit(L,a,R,p,w,c,K).subs(num_dict))

#Pen tabell som viser arbeidskraft, maksimal profitt og analytisk løsning
from IPython.display import Markdown

tbl=f"""
|                                | Desimalverdi                                |
Analytisk verdi              |
| :------------------------------| :-------------------------------------------|
:----------------------------|
| Optimal mengde arbeidskraft:     | ${np.round(float(num_dict[L]),1)}$
|${sp.latex(L_max)}$            |
| Maksimal profitt                 | ${np.round(float(profit_max_num),1)}$
|${sp.latex(profit_max)}$       |

"""
display(Markdown(tbl))

# %% [markdown]
# e) vis figurativt med bruk av `fill_between` arealet hvor man taper penger (i
rødt) og hvor man tjener penger (i grønt). Marker også profittmaksimum og antall
arbeidere i profittmaksimum - gjerne ved bruk av `vlines`. Bruk ellers samme
forutsetninger for argumentene som i oppgave (1b)

# %%
#Definerer verdiene til variablene i profittfunksjonen igjen fordi de ble
overskrevet
```

```
a1 = 0.6
R1 = 6
p1 = 300000
w1 = 100000
c1 = 1000000
K1 = 90000000
#for å dele på 1 million
m = 1000000

plt.subplots(figsize=(18, 6))
plt.plot(x,profit(x,a1,R1,p1,w1,c1,K1)/m,label='profittfunksjon', color =
'blue')
plt.title('Profitt som funksjon av antall ansatte')
plt.axhline(y=profit_max_num/m, color='green', linestyle='-', label =
'profittmaksimum')
plt.axvline(x=float(L_max.subs(num_dict)), color='green', linestyle='--', label
= 'optimal arbeidskraft')
plt.fill_between(x,profit(x,a1,R1,p1,w1,c1,K1)/m, where =
(profit(x,a1,R1,p1,w1,c1,K1)/m >= 0), color = 'green', alpha = 0.35, interpolate
= True, label = 'hvor vi er i profitt')
plt.fill_between(x,profit(x,a1,R1,p1,w1,c1,K1)/m, where =
(profit(x,a1,R1,p1,w1,c1,K1)/m <= 0.), color = 'red', alpha = 0.35, interpolate
= True, label = 'hvor vi er i tap')
plt.legend(loc='lower center',frameon=True, edgecolor='black')
plt.show()


# %% [markdown]
# f) Plot nå to figurer sammen der du viser hva optimal antall arbeidere gir i
profitt (slik som i (2e)) og produksjon av antall biler (som du får fra
produktfunksjonen). Marker optimum med vlines. Ha grafen med profittfunksjonen
over grafen med produktfunksjonen. Du kan bruke `fig, (ax1, ax2) =
plt.subplots(2)` når du skal gjøre dette. <br>
#
# <b> Hint: </b> Du kan finne antall biler som blir produsert ved å bruke antall
arbeidere i profittmaksimum, i produktfunksjonen.
#

# %%

fig2, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))
ax1.plot(x,profit(x,a1,R1,p1,w1,c1,K1)/m,label='profittfunksjon', color =
'blue')
ax1.set_title('Profitt som funksjon av antall ansatte')
ax1.set_ylabel('kroner i millioner')
ax1.set_xlabel('Antall ansatte')
ax1.axhline(y=profit_max_num/m, color='green', linestyle='-', label =
'profittmaksimum')
ax1.axvline(x=float(L_max.subs(num_dict)), color='green', linestyle='--', label
= 'optimal arbeidskraft')
ax1.fill_between(x,profit(x,a1,R1,p1,w1,c1,K1)/m, where =
(profit(x,a1,R1,p1,w1,c1,K1)/m >= 0), color = 'green', alpha = 0.35, interpolate
= True, label = 'hvor vi er i profitt')
ax1.fill_between(x,profit(x,a1,R1,p1,w1,c1,K1)/m, where =
```

```
(profit(x,a1,R1,p1,w1,c1,K1)/m <= 0.), color = 'red', alpha = 0.35, interpolate
= True, label = 'hvor vi er i tap')
ax1.legend(loc='lower center',frameon=True, edgecolor='black')


ax2.plot(x,f(x,0.6,2), label='produksjon', color = 'blue')
ax2.set_title('Biler produsert som funksjon av arbeidskraft')
ax2.set_ylabel('Antall biler produsert')
ax2.set_xlabel('Antall ansatte')
ax2.axvline(x=float(L_max.subs(num_dict)), color='green', linestyle='--', label
= 'optimal arbeidskraft')
ax2.axhline(y=float(f(L_max.subs(num_dict),0.6,2)), color='green',
linestyle='-', label = 'optimal produksjon')
ax2.legend(loc='lower center',frameon=True, edgecolor='black')

plt.show()

# %% [markdown]
# ### Oppgave 4 (10 poeng)
# I denne oppgaven skal vi hente ut et datasett fra eurostat på investeringer i
hosholdningen. Bruk koden under til å hente ut dataene.
# <br><b>NB!:</b> Husk at dere må ha innstallert pakken `eurostat`. Dette gjør
dere med å åpne "Terminal" og kjøre  `pip install eurostat`.

# %%
import eurostat
import pandas as pd

inv_data = eurostat.get_data_df('tec00098')
#stod country fra før så bare lot det stå...
inv_data.columns = ['freq', 'unit',   'sector',  'na_item', 'country'] +
list(range(2010, 2022)) #v2

# %% [markdown]
# a) Bytt navn på kolonnen "geo\\time" til "country" ved bruk av en av kodene
under. Fjern så alle kolonner utenom "country" og alle årstallene.
# <br> <b>NB!:</b> Noen vil få en ekstra første kolonne som heter "freq" eller
noe annet. Da må dere bruke versjon 2 av koden under.

# %%
#drop kolonner som ikke er i bruk
df = inv_data.drop(['freq', 'unit', 'sector', 'na_item'], axis=1)
#printer dataframe
print(df)

# %% [markdown]
# b) fjern radene med nan verdi. Sett deretter indeksen til "country". <br> <b>
Hint: </b> En metode er å bruke `isna()` og `any()` over radaksene (dvs.
`axis=1`)

# %%
#drop nan
df = df.dropna()
#set index til country
```

```
df.set_index('country', inplace=True)

print(df)

# %% [markdown]
# c) Lag et nytt datasett hvor du kun har med de nordiske landene (dvs. "NO", "SE", "DK", "FI"). Det kan være nyttig å bruke `isin` til dette. Bytt så om på kolonner og rader ved hjelp av `transpose`.

# %%
#liste for landene jeg vll ha
nordic = ['NO', 'SE', 'FI', 'DK']

#velg ut nordiske land
df_nordic = df.filter(nordic, axis=0)
#Transponer dataframe slik at år er kolonner
df_nordic = df_nordic.transpose()

print(df_nordic)

# %% [markdown]
# d) Lag en ny kolonne som du kaller "mean". Denne skal være gjennomsnittet av alle de nordiske landene for hvert av årene (dvs at du må ta gjennomsnittet over radene). Plot så dette og kall y-aksen for "investering"

# %%
mean = df_nordic.mean(axis=1)

df_nordic['Mean'] = mean

print(df_nordic)

tick_lab = [x for x in range(2010, 2022)]

plt.subplots(figsize=(18, 6))
plt.plot(df_nordic.index, df_nordic['Mean'], color = 'red')
plt.scatter(df_nordic.index, df_nordic['Mean'], color = 'red')
plt.xlabel("År")
plt.ylabel("Investering")
plt.xticks(tick_lab)
plt.yticks(np.arange(8.5, 10.75, 0.25))
plt.title("Investering i nordiske land - Gjennomsnitt", fontsize = 20)
plt.legend(['Gjennomsnitt'])
plt.show()
```