Mappeoppgave 2

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 her.

NB!: En person fra gruppa må fylle ut dette skjemaet 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.

Oppgave 1 (10 poeng)

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

In [1]:
```
import numpy as np # Importerer pakke.

for a in range(6): # Terningen kastets seks ganger
    dice = np.random.randint(1, 7) # Terningen har seks sider
    print(dice) # Printer ut
```

```
4
6
4
6
5
5
```

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.

In [2]:
```
def dice_list(throws): # Definerer en funksjon for kast
    lot_numbers = []  # Lager en liste lot_numbers
    for b in range(throws):
        dice= np.random.randint(1,7) # Terningen har seks sider
        lot_numbers.append(dice) # Legger inn i listen
    return lot_numbers

dice_list(10)
```

Out[2]:
```
[5, 1, 6, 2, 2, 2, 4, 1, 3, 4]
```

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.

In [3]:
```
def guesses(guess1,guess2): # Definerer en funksjon slik at jeg kan gjette verdier.
    lot_numbers = [] # Listen som resultatene printes ut i.
    for a in range(6):
        dice = np.random.randint(1,7) # Samme terning som over.
        lot_numbers.append(dice) # Legger inn i listen.
    wrong_or_right = ["right" if i == guess1 else "right" if i == guess2 else "wrong" for i in lot_numbers]
    if wrong_or_right.count("right") == 0: # Null rett:
        print("You guessed " + str(guess1)+ " and " + str(guess2) + ", and got 0 right. Better luck next time! The numbers in the
    elif wrong_or_right.count("right") == 1: # En rett:
        print("You guessed " + str(guess1)+ " and " + str(guess2) + ", you got 1 of 6 right. Decent! The numbers in the lot are:
    elif wrong_or_right.count("right") == 2: # To rett:
        print("You guessed " + str(guess1)+ " and " + str(guess2) + ", you got 2 of 6 right. Very nice! The numbers in the lot ar
    else: # Mer enn to rette:
        print("You guessed " + str(guess1)+ " and " + str(guess2) + ", you got more than 2 right. Congratulations! The numbers ir

guesses(5,2) # Legger inn tallene som gjettes.
```

```
You guessed 5 and 2, you got 2 of 6 right. Very nice! The numbers in the lot are: [3, 4, 3, 2, 2, 6].
```

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.

In [4]:
```
total_draws_list = [] # Hvor mange trekk krevdes før vinnertall

drawn_numbers = [] # Liste over alle tall som har blitt trukket
```

In [37]:
```
import random

prize_number = 15 # Dette er vinnertallet

lotto_numbers = list(range(0,31)) # Lager en liste med tall fra 0 til 30

randomizer = random.choice(lotto_numbers) # Lager en komando som tilfeldig trekker et tall fra lotto_numbers
```

```
total_draws = 0 # Antall trekk før vinnertallet

while randomizer != prize_number: # Løkken vil kjøre hvis trukket tall ikke er lik vinnertallet
    total_draws +=1 # Teller hvor mange ganger tall trekkes før vinnertallet trekkes
    lotto_numbers.remove(randomizer) # Fjerner tall som allerede er trukket
    drawn_numbers.append(randomizer) # Lagrer alle tall som har blitt trukket for oppgave c
    randomizer = random.choice(lotto_numbers) # trekker et tall fra lotto_number
    if randomizer == prize_number: # loopen brytes hvis trukket tall er lik vinnertallet
        drawn_numbers.append(randomizer) # legger til det siste tallet i drawn_numbers

total_draws_list.append(total_draws) # lagrer hvor mange trekk som krevdes

print(f"""JACKPOT!! Gratulerer, du vant!
Det tok {total_draws} trekk før du trakk vinnertallet {prize_number}""")
```

```
JACKPOT!! Gratulerer, du vant!
Det tok 6 trekk før du trakk vinnertallet 15
```

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

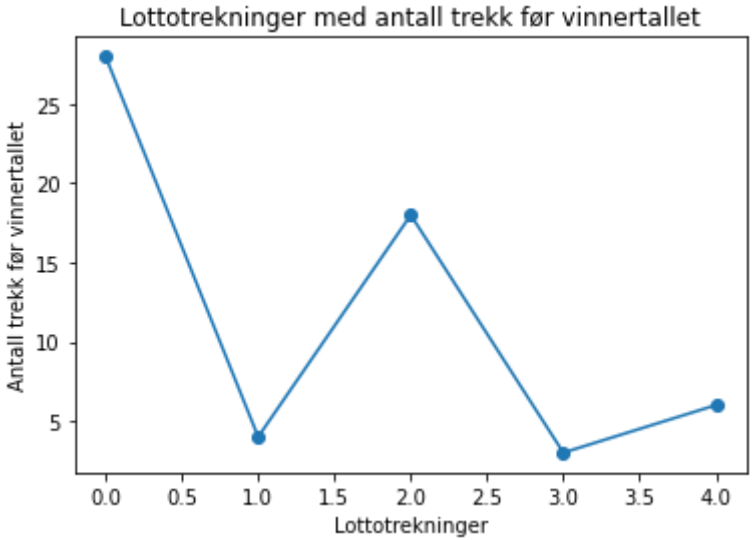
- bruker `scatter`;
- lager plottet dynamisk (dvs at hver trekning vises hver for seg, og at x-aksen endrer seg etter en gitt verdi);
- viser hvor når siste trekningen blir gjort (dvs at den vises kun når du har trukket vinnertallet).

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

In [6]:
```
#pakker som du kan få bruk for
from matplotlib.patches import Ellipse
from IPython.display import display, clear_output
from matplotlib import pyplot as plt
```

In [38]:
```
draw_count = list(range(0,len(total_draws_list))) # lager en liste over alle lotto trekningene

plt.scatter(draw_count,total_draws_list) # Plotter antall lottotrekninger på x-aksen og antall trekk før vinnertallet på y-aksen
plt.plot(draw_count,total_draws_list) # Plotter punkter også
plt.xlabel('Lottotrekninger') # X-akse label
plt.ylabel('Antall trekk før vinnertallet') # y-akse label
plt.title('Lottotrekninger med antall trekk før vinnertallet')
```

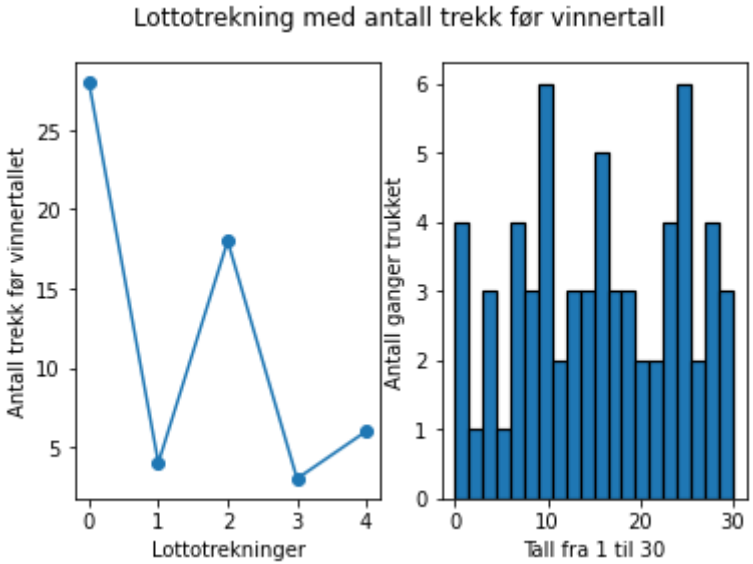
Out[38]:
```
Text(0.5, 1.0, 'Lottotrekninger med antall trekk før vinnertallet')
```



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.

In [39]:
```
fig, (ax1,ax2) = plt.subplots(1, 2) # setter opp subplots

draw_count = list(range(0,len(total_draws_list))) # en liste over alle lotto trekningene

ax1.scatter(draw_count,total_draws_list) # plotter antall lottotrekninger på x-aksen og antall trekk før vinnertallet på y-aksen
ax1.plot(draw_count,total_draws_list) # plotter punkter
ax1.set_xlabel('Lottotrekninger') # x-akse label
ax1.set_ylabel('Antall trekk før vinnertallet') #y-akse label
ax2.hist(drawn_numbers, bins=20, edgecolor='black',linewidth=1.2) # plotter histogram for antall ganger ulike tall har blitt trul
ax2.set_xlabel('Tall fra 1 til 30') # x-akse label
ax2.set_ylabel('Antall ganger trukket') # y-akse label
fig.suptitle('Lottotrekning med antall trekk før vinnertall')
```

Out[39]:
```
Text(0.5, 0.98, 'Lottotrekning med antall trekk før vinnertall')
```



Oppgave 3 (20 poeng)

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`

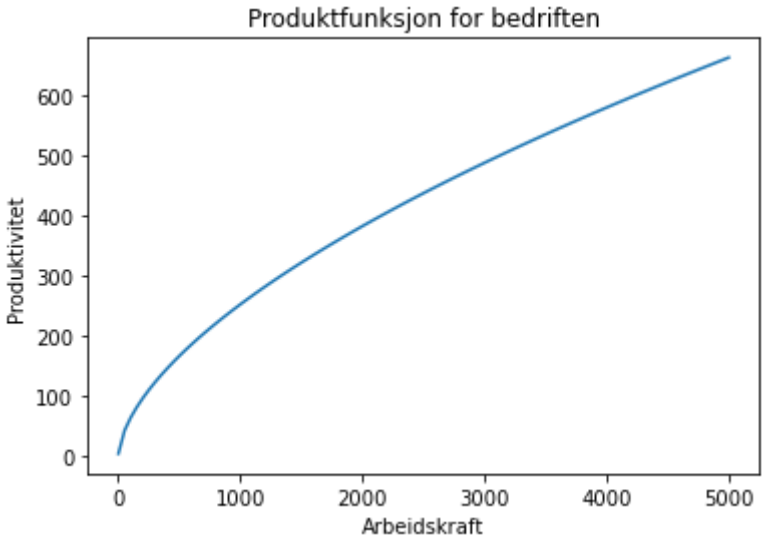
In [9]:
```
import numpy as np # Importerer pakker
from matplotlib import pyplot as plt
```

In [10]:
```
L = np.linspace(1,5000,100) # Ulike verdier for L. Bruker linspace.

def produkt(L,a,R): # Definerer en produktfunksjon.
    return 2*R*L**a
```

In [11]:
```
plt.plot(L,produkt(L,0.6,2)) # Plotter produktfunksjon, med labels og tittel.
plt.xlabel("Arbeidskraft")
```

```
plt.ylabel("Produktivitet")
plt.title("Produktfunksjon for bedriften")
```

Out[11]:
```
Text(0.5, 1.0, 'Produktfunksjon for bedriften')
```



b) anta at profittfunksjonen til denne bedriften er ${profit = f(L, a, R)p-wL-cR-K}$, hvor

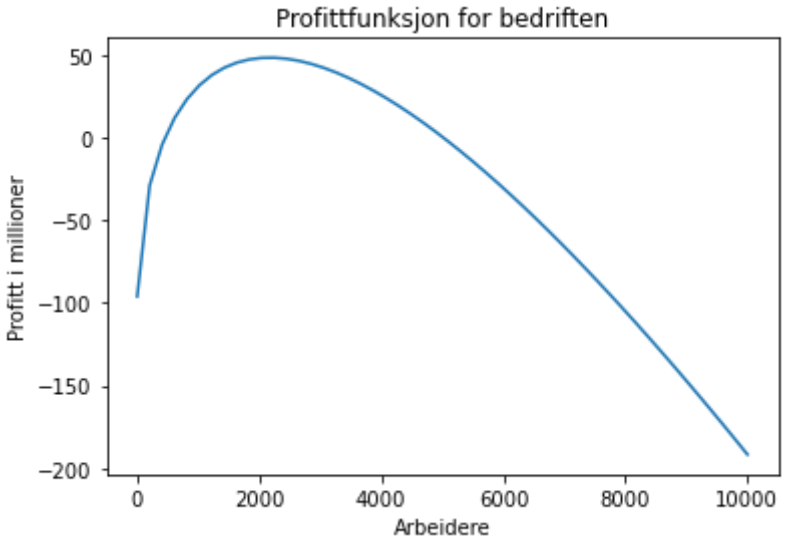
- `w` er månedslønnen til arbeiderne,
- `c` er kostnaden for robotmaskinene
- `K` er faste kostnader
- `p` er utsalgsprisen på bilene.

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)

In [12]:
```
a = 0.6 # Definerer verdier som skal brukes i profittfunksjon.

R = 6

p = 300000

w = 100000
```

```
c = 1000000

K = 90000000

L= np.linspace(0,10000) # L er mellom 0 og 10 000.

def profitt (a,p,w,L,c,R,K):
    return (produkt(L,a,R)*p-w*L-c*R-K)/(1000000) # Definerer profittfunksjon sammen med produktfunksjon.

fig,ax=plt.subplots() # Bruker subplots
plt.plot(L,profitt (a,p,w,L,c,R,K)) # Plotter funksjonen og gir passende labels og tittel.
plt.xlabel("Arbeidere")
plt.ylabel("Profitt i millioner")
plt.title("Profittfunksjon for bedriften")
```

Out[12]:
```
Text(0.5, 1.0, 'Profittfunksjon for bedriften')
```

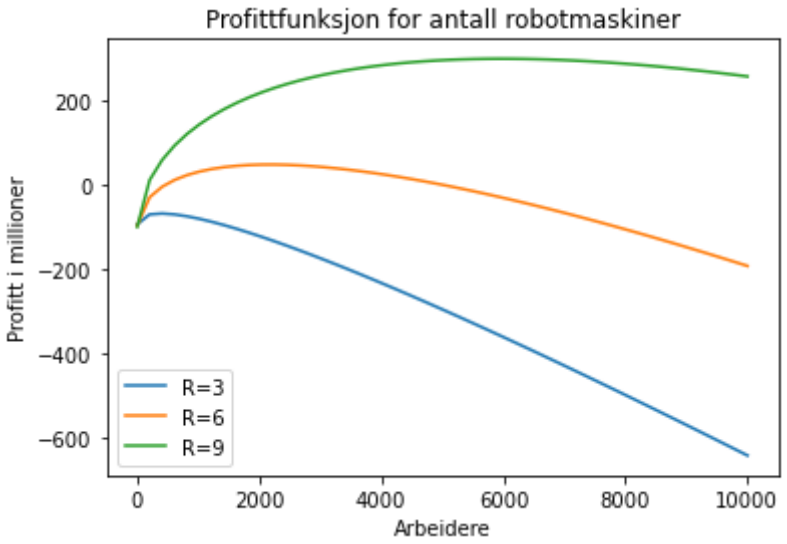


c) Plot profitttfunksjonen for antall robostmaskiner `R=[3, 6, 9]` i samme plot (dvs at tre profittfunksjoner vises sammen). Bruk av "for loops" for å gjøre dette belønnes

In [13]:
```
R=[3, 6, 9] # Legger antall robotmaskinser inn i en liste.

fig,ax=plt.subplots() # Bruker subplots til å plotte de tre funksjonene sammen.
plt.plot(L,profitt (a,p,w,L,c,R[0],K),label='R=3') # 3 antall maskiner
plt.plot(L,profitt (a,p,w,L,c,R[1],K),label='R=6') # 6 antall maskiner
```

```
plt.plot(L,profitt (a,p,w,L,c,R[2],K),label='R=9') # 9 antall maskiner
plt.plot

plt.xlabel("Arbeidere") # Plotter funksjonene med ulike antall roboter med passende tittel og labels.
plt.ylabel("Profitt i millioner")
plt.title("Profittfunksjon for antall robotmaskiner")
ax.legend(loc='lower left')
```

Out[13]: <matplotlib.legend.Legend at 0x7ff3d7d0c490>



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

In [14]:
```
#pakker du kan få bruk for
import sympy as sp
from sympy.solvers import solve

L,a,w,p,c,K,R = sp.symbols("L a w p c K R") # Bruker sympy for å få symboler.

profitt(a,p,w,L,c,R,K) # Profittfunksjonen med symboler.
```

Out[14]: $\displaystyle - \frac{K}{1000000} - \frac{L w}{1000000} + \frac{L^{a} R p}{500000} - \frac{R c}{1000000}$

In [15]:
```
d_profitt=sp.diff(profitt(a,p,w,L,c,R,K),L) # Deriverer profittfunksjonen med hensyn på L.
d_profitt
```

Out[15]: $\displaystyle - \frac{w}{1000000} + \frac{L^{a} R a p}{500000 L}$

In [16]:
```
foc=sp.Eq(d_profitt,0) # Setter opp som en ligning og kaller den foc.
foc
```

Out[16]: $\displaystyle - \frac{w}{1000000} + \frac{L^{a} R a p}{500000 L} = 0$

In [17]:
```
from sympy.solvers import solve
L_max=solve(foc,L)[0] # Henter ut resultatet for den største verdien av L.
L_max
```

Out[17]: $\displaystyle \left(\frac{2 R a p}{w}\right)^{- \frac{1}{a - 1}}$

In [18]:
```
profit_max= profitt(L_max,a,w,p,R,K,K) # Setter inn L_max for å få ut utrykket for profittmaksimum.
profit_max
```

Out[18]: $\displaystyle - \frac{K R}{1000000} + \frac{K a p^{\left(\frac{2 R a p}{w}\right)^{- \frac{1}{a - 1}}}}{500000} - \frac{K}{1000000} - \frac{p w}{1000000}$

In [19]:
```
num_dict={a:0.6,R:6,p:300000,w:100000,c:1000000,K:90000000} # Legger inn verdier ved dict.

L_max.subs(num_dict) # Finner optimal antall arbeidere
```

Out[19]: $\displaystyle 2168.37493200784$

In [20]:
```
num_dict[L]=L_max.subs(num_dict)
num_dict
```

Out[20]:
```
{a: 0.6,
 R: 6,
 p: 300000,
 w: 100000,
 c: 1000000,
 K: 90000000,
 L: 2168.37493200784}
```

In [21]:
```
profit_max_num=float(profitt(a,p,w,L,c,R,K).subs(num_dict))
profit_max_num # Finner profittmaksimum.
```

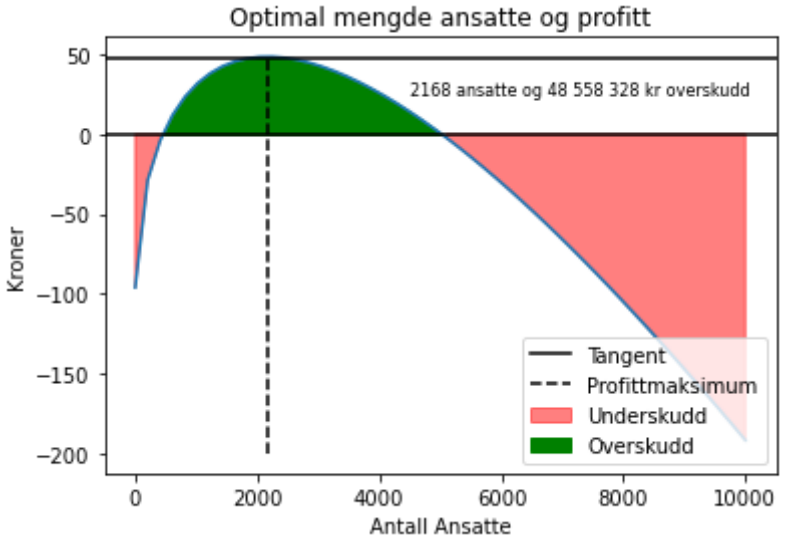
Out[21]:
```
48.55832880052253
```

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)

In [22]:
```
a = 0.6

R = 6

p = 300000

w = 100000

c = 1000000

K = 90000000

L= np.linspace(0,10000)

x=L # Legger L som x
y=profitt (a,p,w,L,c,R,K) # legger profittfunksjonen som y

plt.plot(x,y) # plotter grafen
plt.ylabel('Kroner') # label for y akse
plt.xlabel('Antall Ansatte') # label for x akse
plt.axhline(y=48,color="black",label = "Tangent") # Tegner tangenten i for den deriverte = 0

plt.vlines(x=2168,linestyles="dashed",
                          ymin=-200,
                          ymax=48.5,
                          color="black",
                          label="Profittmaksimum") # Tegner linjen for profittmaksimum
plt.fill_between(x,y,
                 where=y<0,
                 color="red",
                 label="Underskudd",
                alpha=0.5) # Fyller området under y = 0

plt.fill_between(x,y,
                 color="green",
                 where=y>0,
                 interpolate=True,
                 label="Overskudd") # Fyller området over y = 0
```

```
plt.axhline(y=0,color="black") # Tegner y = 0

plt.text(4500,25,"2168 ansatte og 48 558 328 kr overskudd",fontsize=8) # Tekst som markerer hvor mye overskudd og ansatte

plt.legend(loc="lower right") # Lager en legend

plt.title("Optimal mengde ansatte og profitt") # lager title
```

Out[22]:
```
Text(0.5, 1.0, 'Optimal mengde ansatte og profitt')
```



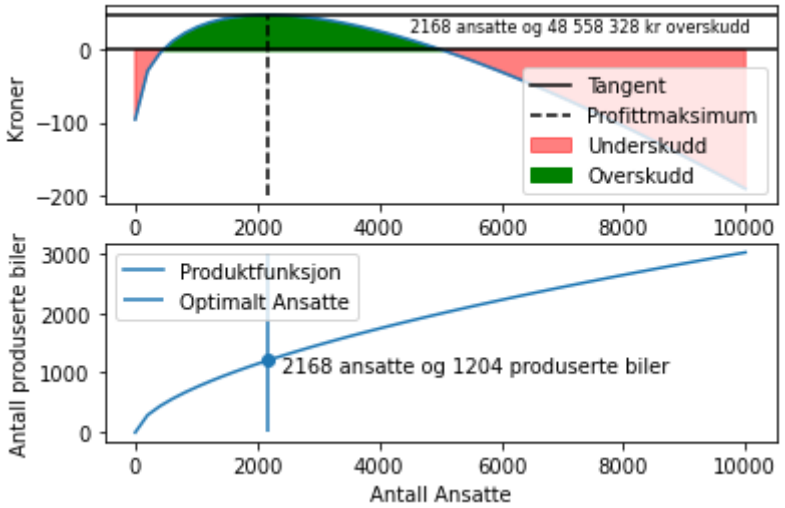
f) Plot nå to figurer sammen der du viser hva optimal antall arbeidere gir i profitt (slik som i (2e)) og produksjon av antall biler (som du får fra produktfunksjonen). Marker optimum med vlines. Ha grafen med profittfunksjonen over grafen med produktfunksjonen. Du kan bruke `fig, (ax1, ax2) = plt.subplots(2)` når du skal gjøre dette.

Hint: Du kan finne antall biler som blir produsert ved å bruke antall arbeidere i profittmaksimum, i produktfunksjonen.

In [23]:
```
fig, (ax1, ax2) = plt.subplots(2) # setter opp subplots

# Figur 1

ax1.plot(x,y) # plotter grafen
ax1.set_ylabel('Kroner')
ax1.set_xlabel('Antall Ansatte')
```

```
ax1.axhline(y=48,color="black",label = "Tangent") # Tegner tangenten i for den deriverte = 0

ax1.vlines(x=2168,linestyles="dashed",
                          ymin=-200,
                          ymax=48.5,
                          color="black",
                          label="Profittmaksimum") # Tegner linjen for profittmaksimum
ax1.fill_between(x,y,
                 where=y<0,
                 color="red",
                 label="Underskudd",
                alpha=0.5) # Fyller området under y = 0

ax1.fill_between(x,y,
                 color="green",
                 where=y>0,
                 interpolate=True,
                 label="Overskudd") # Fyller området over y = 0

ax1.axhline(y=0,color="black") # Tegner x = 0

ax1.text(4500,25,"2168 ansatte og 48 558 328 kr overskudd",fontsize=8) # Tekst som markerer hvor mye overskudd og ansatte

ax1.legend(loc="lower right") # Lager en legend

# Figur 2

ax2.plot(x,produkt(L,a,R), label="Produktfunksjon") # tegner produktfunksjonen
ax2.vlines(x=2168,ymax=0,ymin=3000, label="Optimalt Ansatte") # tegner linje for optamale antall ansatte
ax2.scatter(x=2168,y=1204) # lager punkt i skjæringspunktet
ax2.text(2400, 1000,"2168 ansatte og 1204 produserte biler") # legger ved tekst for å markere antall
ax2.legend() # tegner legende
ax2.set_ylabel("Antall produserte biler") #skriver y-akse label
ax2.set_xlabel("Antall Ansatte") #skriver x-akse label
```

Out[23]:
```
Text(0.5, 0, 'Antall Ansatte')
```



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.
NB!: Husk at dere må ha innstallert pakken `eurostat`. Dette gjør dere med å åpne "Terminal" og kjøre `pip install eurostat`.

In [24]:
```
import eurostat

inv_data = eurostat.get_data_df('tec00098')
```

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.
NB!: Noen vil få en ekstra første kolonne som heter "freq" eller noe annet. Da må dere bruke versjon 2 av koden under.

In [25]:
```
inv_data.columns = ['freq', 'unit',   'sector',  'na_item', 'country'] + list(range(2010, 2022)) #v2
```

In [26]:
```
# Her fjerner vi "freq", "unit", "sector" og "na_item" ved bruk av drop funksjonen.
# Vi spesifiserer ved axis=1 at vi vil droppe fra x-aksen.

inv_data_1 = inv_data.drop(['freq', 'unit', 'sector', 'na_item'], axis=1)

inv_data_1
```

Out[26]:

	country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
0	AT	8.44	8.60	8.47	8.58	8.30	8.36	8.29	8.69	8.81	9.03	9.21	9.99
1	BE	9.82	9.45	9.37	9.03	9.45	9.31	9.33	9.28	9.35	9.78	9.15	9.94
2	CH	6.83	6.48	6.29	6.31	6.20	6.26	6.13	6.07	6.06	5.69	5.42	NaN
3	CY	13.72	10.73	8.74	7.53	7.15	6.69	8.02	8.97	11.30	12.98	13.12	13.26
4	CZ	11.14	9.82	8.74	8.77	8.88	8.84	9.18	7.86	9.00	9.45	9.34	9.24
5	DE	8.72	9.51	9.61	9.56	9.65	9.32	9.62	9.48	9.68	9.71	9.95	10.55
6	DK	8.50	8.51	7.87	7.35	7.60	7.50	7.46	8.10	8.33	8.57	8.69	9.24
7	EA19	9.30	9.21	8.80	8.37	8.24	8.09	8.35	8.52	8.71	8.77	8.53	9.51
8	EE	6.14	6.47	6.84	7.52	7.79	7.94	8.52	8.99	8.68	9.26	9.94	10.04
9	EL	9.12	8.54	6.28	4.90	3.16	2.72	2.75	2.69	2.44	2.59	2.97	3.40
10	ES	9.66	7.85	6.48	4.84	4.72	4.57	4.60	5.17	5.42	5.54	5.34	6.68
11	EU27_2020	9.07	8.95	8.56	8.12	8.06	7.97	8.22	8.42	8.54	8.60	8.33	9.19
12	EU28	8.44	8.34	7.94	7.67	7.65	7.55	7.85	8.16	8.24	8.39	NaN	NaN
13	FI	11.66	12.18	12.12	11.49	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50
14	FR	9.31	9.46	9.31	9.27	9.07	8.88	8.99	9.41	9.47	9.62	8.57	9.95
15	HR	5.96	5.66	5.53	5.01	4.78	4.91	4.96	5.01	5.35	6.37	6.40	6.45
16	HU	6.67	5.11	4.80	4.92	5.17	5.68	5.86	6.53	7.19	7.52	8.87	8.30
17	IE	6.01	4.96	4.12	4.29	4.41	4.72	5.26	5.50	6.27	4.92	4.27	5.29
18	IS	4.45	4.58	4.81	5.19	5.57	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	IT	10.33	9.82	9.20	8.42	7.78	7.58	7.68	7.74	7.77	7.64	7.20	8.69
20	LT	4.73	5.11	4.70	5.56	6.04	6.68	6.89	6.51	6.78	7.09	7.04	7.39
21	LU	9.69	11.26	10.43	11.34	12.75	12.34	12.48	12.35	11.45	10.34	9.71	11.33
22	LV	3.56	4.92	6.17	4.62	4.98	5.76	4.87	4.83	5.68	5.85	5.30	4.62
23	NL	10.05	9.54	8.47	7.46	8.10	9.08	10.60	10.80	11.52	12.15	12.22	12.96

	country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
24	NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
25	PL	7.99	7.92	8.24	7.59	7.86	7.88	7.52	6.88	5.84	5.93	5.18	6.59
26	PT	6.27	5.81	5.07	4.37	4.51	4.50	4.71	5.05	5.48	5.65	5.69	6.10
27	RS	NaN	NaN	NaN	NaN	NaN	2.63	3.25	3.17	3.21	3.35	2.97	NaN
28	SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
29	SI	6.92	6.30	5.78	5.50	5.65	5.81	5.85	6.28	6.51	6.33	5.64	6.20
30	SK	6.32	7.00	6.65	7.08	6.39	6.10	6.75	6.71	6.76	6.79	6.93	7.04
31	UK	4.96	5.02	4.95	5.27	5.67	5.84	6.16	6.77	6.62	6.81	NaN	NaN

In [27]:
```
# Vi bruker dropna for å fjerne alle radene med nan verdier.

inv_data_1 = inv_data_1.dropna()

inv_data_1
```

Out[27]:

	country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
0	AT	8.44	8.60	8.47	8.58	8.30	8.36	8.29	8.69	8.81	9.03	9.21	9.99
1	BE	9.82	9.45	9.37	9.03	9.45	9.31	9.33	9.28	9.35	9.78	9.15	9.94
3	CY	13.72	10.73	8.74	7.53	7.15	6.69	8.02	8.97	11.30	12.98	13.12	13.26
4	CZ	11.14	9.82	8.74	8.77	8.88	8.84	9.18	7.86	9.00	9.45	9.34	9.24
5	DE	8.72	9.51	9.61	9.56	9.65	9.32	9.62	9.48	9.68	9.71	9.95	10.55
6	DK	8.50	8.51	7.87	7.35	7.60	7.50	7.46	8.10	8.33	8.57	8.69	9.24
7	EA19	9.30	9.21	8.80	8.37	8.24	8.09	8.35	8.52	8.71	8.77	8.53	9.51
8	EE	6.14	6.47	6.84	7.52	7.79	7.94	8.52	8.99	8.68	9.26	9.94	10.04
9	EL	9.12	8.54	6.28	4.90	3.16	2.72	2.75	2.69	2.44	2.59	2.97	3.40
10	ES	9.66	7.85	6.48	4.84	4.72	4.57	4.60	5.17	5.42	5.54	5.34	6.68
11	EU27_2020	9.07	8.95	8.56	8.12	8.06	7.97	8.22	8.42	8.54	8.60	8.33	9.19
13	FI	11.66	12.18	12.12	11.49	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50
14	FR	9.31	9.46	9.31	9.27	9.07	8.88	8.99	9.41	9.47	9.62	8.57	9.95
15	HR	5.96	5.66	5.53	5.01	4.78	4.91	4.96	5.01	5.35	6.37	6.40	6.45
16	HU	6.67	5.11	4.80	4.92	5.17	5.68	5.86	6.53	7.19	7.52	8.87	8.30
17	IE	6.01	4.96	4.12	4.29	4.41	4.72	5.26	5.50	6.27	4.92	4.27	5.29
19	IT	10.33	9.82	9.20	8.42	7.78	7.58	7.68	7.74	7.77	7.64	7.20	8.69
20	LT	4.73	5.11	4.70	5.56	6.04	6.68	6.89	6.51	6.78	7.09	7.04	7.39
21	LU	9.69	11.26	10.43	11.34	12.75	12.34	12.48	12.35	11.45	10.34	9.71	11.33
22	LV	3.56	4.92	6.17	4.62	4.98	5.76	4.87	4.83	5.68	5.85	5.30	4.62
23	NL	10.05	9.54	8.47	7.46	8.10	9.08	10.60	10.80	11.52	12.15	12.22	12.96
24	NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
25	PL	7.99	7.92	8.24	7.59	7.86	7.88	7.52	6.88	5.84	5.93	5.18	6.59
26	PT	6.27	5.81	5.07	4.37	4.51	4.50	4.71	5.05	5.48	5.65	5.69	6.10

	country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
28	SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
29	SI	6.92	6.30	5.78	5.50	5.65	5.81	5.85	6.28	6.51	6.33	5.64	6.20
30	SK	6.32	7.00	6.65	7.08	6.39	6.10	6.75	6.71	6.76	6.79	6.93	7.04

b) fjern radene med nan verdi. Sett deretter indeksen til "country".
Hint: En metode er å bruke `isna()` og `any()` over radaksene (dvs. `axis=1`)

In [28]:
```
# Vi bruker set_index for å gjøre country til indeksen.

inv_data_1 = inv_data_1.set_index('country')

inv_data_1
```

Out[28]:

country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
AT	8.44	8.60	8.47	8.58	8.30	8.36	8.29	8.69	8.81	9.03	9.21	9.99
BE	9.82	9.45	9.37	9.03	9.45	9.31	9.33	9.28	9.35	9.78	9.15	9.94
CY	13.72	10.73	8.74	7.53	7.15	6.69	8.02	8.97	11.30	12.98	13.12	13.26
CZ	11.14	9.82	8.74	8.77	8.88	8.84	9.18	7.86	9.00	9.45	9.34	9.24
DE	8.72	9.51	9.61	9.56	9.65	9.32	9.62	9.48	9.68	9.71	9.95	10.55
DK	8.50	8.51	7.87	7.35	7.60	7.50	7.46	8.10	8.33	8.57	8.69	9.24
EA19	9.30	9.21	8.80	8.37	8.24	8.09	8.35	8.52	8.71	8.77	8.53	9.51
EE	6.14	6.47	6.84	7.52	7.79	7.94	8.52	8.99	8.68	9.26	9.94	10.04
EL	9.12	8.54	6.28	4.90	3.16	2.72	2.75	2.69	2.44	2.59	2.97	3.40
ES	9.66	7.85	6.48	4.84	4.72	4.57	4.60	5.17	5.42	5.54	5.34	6.68
EU27_2020	9.07	8.95	8.56	8.12	8.06	7.97	8.22	8.42	8.54	8.60	8.33	9.19
FI	11.66	12.18	12.12	11.49	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50
FR	9.31	9.46	9.31	9.27	9.07	8.88	8.99	9.41	9.47	9.62	8.57	9.95
HR	5.96	5.66	5.53	5.01	4.78	4.91	4.96	5.01	5.35	6.37	6.40	6.45
HU	6.67	5.11	4.80	4.92	5.17	5.68	5.86	6.53	7.19	7.52	8.87	8.30
IE	6.01	4.96	4.12	4.29	4.41	4.72	5.26	5.50	6.27	4.92	4.27	5.29
IT	10.33	9.82	9.20	8.42	7.78	7.58	7.68	7.74	7.77	7.64	7.20	8.69
LT	4.73	5.11	4.70	5.56	6.04	6.68	6.89	6.51	6.78	7.09	7.04	7.39
LU	9.69	11.26	10.43	11.34	12.75	12.34	12.48	12.35	11.45	10.34	9.71	11.33
LV	3.56	4.92	6.17	4.62	4.98	5.76	4.87	4.83	5.68	5.85	5.30	4.62
NL	10.05	9.54	8.47	7.46	8.10	9.08	10.60	10.80	11.52	12.15	12.22	12.96
NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
PL	7.99	7.92	8.24	7.59	7.86	7.88	7.52	6.88	5.84	5.93	5.18	6.59

country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
PT	6.27	5.81	5.07	4.37	4.51	4.50	4.71	5.05	5.48	5.65	5.69	6.10
SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
SI	6.92	6.30	5.78	5.50	5.65	5.81	5.85	6.28	6.51	6.33	5.64	6.20
SK	6.32	7.00	6.65	7.08	6.39	6.10	6.75	6.71	6.76	6.79	6.93	7.04

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`.

In [29]:
```
# Vi bruker filter, og velger de nordiske landene og spesifiserer fra axis 0.
# vi velger fra akse 0 fordi akse 0 er y-aksen.

inv_data_scandi = inv_data_1.filter(items=['NO', 'SE', 'DK', 'FI'], axis=0)

inv_data_scandi
```

Out[29]:

country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
DK	8.50	8.51	7.87	7.35	7.60	7.50	7.46	8.10	8.33	8.57	8.69	9.24
FI	11.66	12.18	12.12	11.49	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50

In [30]:
```
# Vi bruker np.transpose for å bytte om på kolonnene og radene.

inv_data_scandi_1 = np.transpose(inv_data_scandi)

inv_data_scandi_1
```

Out[30]:

country	NO	SE	DK	FI
2010	9.71	5.86	8.50	11.66
2011	10.98	5.48	8.51	12.18
2012	11.74	4.51	7.87	12.12
2013	12.32	4.61	7.35	11.49
2014	11.82	4.69	7.60	10.88
2015	11.35	5.69	7.50	10.49
2016	12.34	6.15	7.46	11.57
2017	13.02	6.77	8.10	12.09
2018	12.22	6.24	8.33	12.50
2019	11.89	5.88	8.57	12.25
2020	11.29	6.48	8.69	11.91
2021	11.11	6.84	9.24	12.50

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"

In [31]:
```
# Vi lager gjennomsnitt av verdiene fordelt på landene for hvert år.
# Så legger vi til en ny column i dataframen for mean.

mean = inv_data_scandi_1.mean(axis=1)

inv_data_scandi_1['Mean'] = mean
# Kode hentet fra: https://www.geeksforgeeks.org/adding-new-column-to-existing-dataframe-in-pandas/

inv_data_scandi_1
```

Out[31]:

country	NO	SE	DK	FI	Mean
2010	9.71	5.86	8.50	11.66	8.9325
2011	10.98	5.48	8.51	12.18	9.2875
2012	11.74	4.51	7.87	12.12	9.0600
2013	12.32	4.61	7.35	11.49	8.9425
2014	11.82	4.69	7.60	10.88	8.7475
2015	11.35	5.69	7.50	10.49	8.7575
2016	12.34	6.15	7.46	11.57	9.3800
2017	13.02	6.77	8.10	12.09	9.9950
2018	12.22	6.24	8.33	12.50	9.8225
2019	11.89	5.88	8.57	12.25	9.6475
2020	11.29	6.48	8.69	11.91	9.5925
2021	11.11	6.84	9.24	12.50	9.9225

In [32]:
```
# Vi lager en liste med årstallene som blir brukt i dataframen, for å kunne enkelt sette det inn i et plot.

årstall = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021]
```

In [33]:
```
# Vi lager et plot hvor vi putter in investeringen til de nordiske landene og mean mellom dem for årene 2010-2021.

fig, ax = plt.subplots()
ax.plot(årstall, inv_data_scandi_1['NO'], label = 'Norge', color = 'green')
ax.plot(årstall, inv_data_scandi_1['SE'], label = 'Sverige', color = 'black')
ax.plot(årstall, inv_data_scandi_1['DK'], label = 'Danmark', color = 'red')
ax.plot(årstall, inv_data_scandi_1['FI'], label = 'Finland', color = 'blue')
ax.plot(årstall, inv_data_scandi_1['Mean'], label = 'Mean', color = 'purple')
ax.legend(bbox_to_anchor=(1.04, 0.5), loc="center left")
# Kode hentet fra: https://stackoverflow.com/questions/4700614/how-to-put-the-legend-outside-the-plot
ax.set_title('Investering i nordiske land fra 2010-2021')
ax.set_ylabel('Investering')
ax.set_xlabel('Årstall')
plt.show
```

Out[33]: <function matplotlib.pyplot.show(close=None, block=None)>