Mappe 2_oppgaver

November 30, 2022

1 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.

1.0.1 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

[2]:
```
#Laster ned numpy
import numpy as np
#Lager terningkast
np.random.seed(10)
terningkast = range(1,7)
for count in terningkast:
    print(np.random.randint(1,7))
```

```
2
6
5
1
2
4
```

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.

[3]:
```
import random
#Tom liste
lot_numbers=[]

#Seed og løkke
np.random.seed(10)
for i in range(1,7):
    terningkast = np.random.randint(1,7)
    lot_numbers.append(terningkast)
print(lot_numbers)
```

```
[2, 6, 5, 1, 2, 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.

[5]:
```
#Tom liste
lot_numbers=[]
#Definerer a og b
a = 3
b = 4
guessed_number=[a,b]
np.random.seed(10)
#For-løkke med if-setninger
for i in range(1,7):
    terningkast = np.random.randint(1,7)
    lot_numbers.append(terningkast)
```

```
if guessed_number in lot_numbers:
        print("Gratulerer, begge riktig")
elif a or b in lot_numbers:
        print("Gratulerer, èn riktig")
else:
        print("Dessverre, ingen riktige")

print(lot_numbers)
```

```
Gratulerer, èn riktig
[2, 6, 5, 1, 2, 4]
```

1.0.2 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.

[8]:
```
winning_number=7 #Setter vinntertall
lotto_number=np.random.randint(0,30) #Tall jeg trekker fra
guess_count=0 #Starte å telle trekningene fra
guessed_numbers=[]

#While-løkke som går så lenge vinnertallet ikke er trukket enda
while lotto_number!=winning_number:
    guess_count += 1
    lotto_number = np.random.randint(0,30)
    guessed_numbers.append(lotto_number)

guessed_numbers
```

[8]:
```
[28, 1, 20, 26, 0, 24, 12, 5, 4, 7]
```

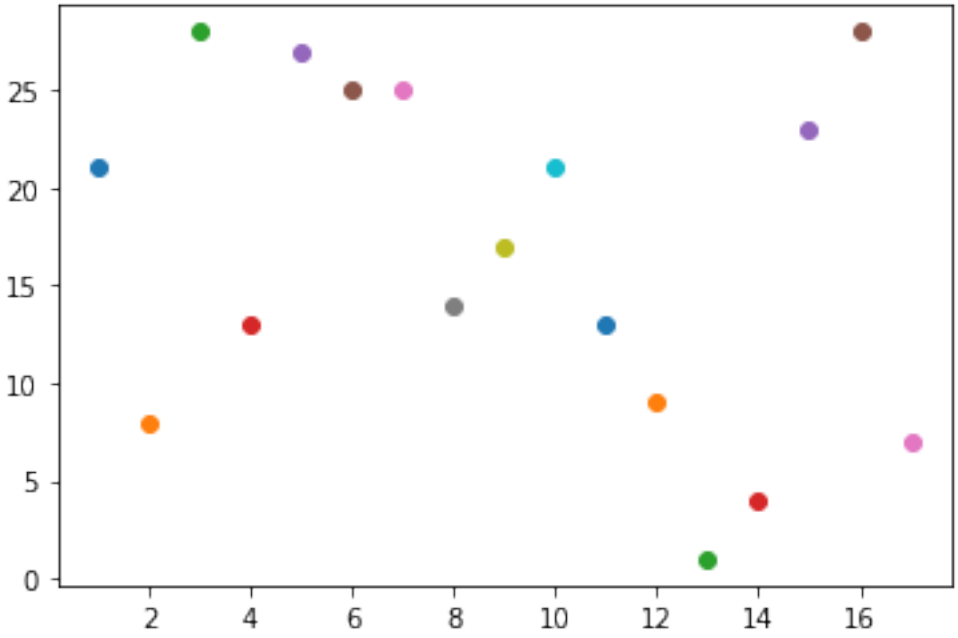
b) Lag et plot av den while-løkken du nettopp lagde. Man blir belønnet om man;

- 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`.

[10]:
```
#pakker som du kan få bruk for
from matplotlib.patches import Ellipse
from IPython.display import display, clear_output
import matplotlib.pyplot as plt

winning_number=7 #Setter vinntertall
lotto_number=np.random.randint(0,30) #Tall jeg trekker fra
```

```
guess_count=0 #Starte å telle trekningene fra
guessed_numbers=[]

#While-løkke som går så lenge vinnertallet ikke er trukket enda
while lotto_number!=winning_number:
    guess_count += 1
    lotto_number = np.random.randint(0,30)
    guessed_numbers.append(lotto_number)
    plt.scatter(guess_count, lotto_number)
```



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.

[27]:
```
winning_number=7 #Setter vinntertall
lotto_number=np.random.randint(0,30) #Tall jeg trekker fra
guess_count=0 #Starte å telle trekningene fra
guessed_numbers=[]


#While-løkke som går så lenge vinnertallet ikke er trukket enda
while lotto_number!=winning_number:
    guess_count += 1
    lotto_number = np.random.randint(0,30)
    guessed_numbers.append(lotto_number)
    print(guess_count, lotto_number)
```

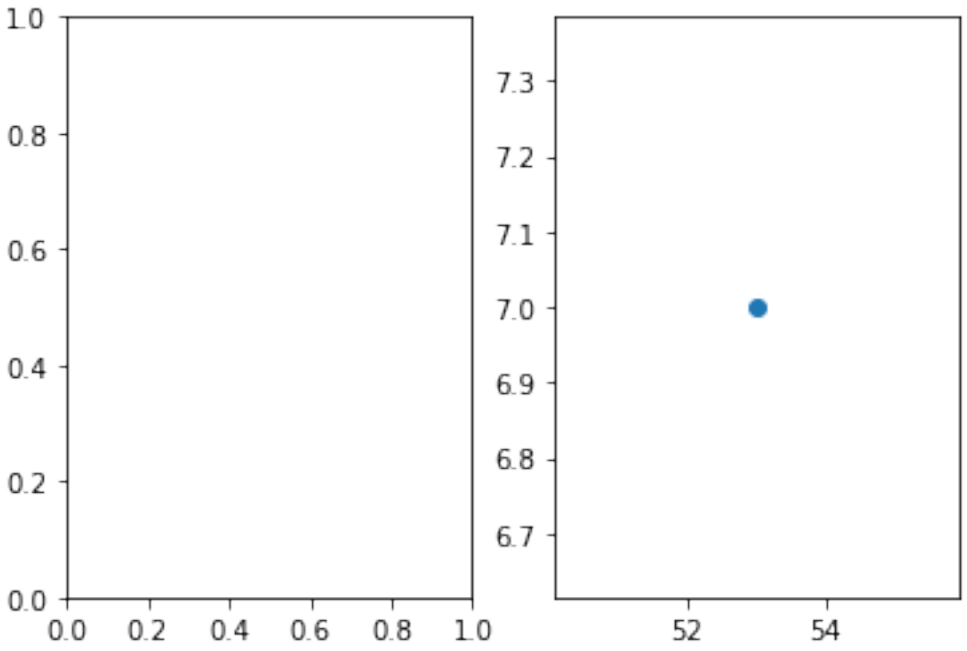
```
fig, (ax1, ax2) = plt.subplots(1,2)
ax1.plot(plt.scatter(guess_count,lotto_number))
ax2.plot(plt.hist(guess_count, lotto_number))
```

```
1 23
2 17
3 29
4 5
5 18
6 19
7 20
8 5
9 23
10 12
11 6
12 29
13 8
14 16
15 10
16 4
17 22
18 19
19 21
20 9
21 3
22 24
23 3
24 16
25 11
26 9
27 22
28 18
29 19
30 9
31 17
32 29
33 10
34 4
35 0
36 27
37 21
38 0
39 10
40 8
41 17
```

```
42 28
43 13
44 3
45 22
46 28
47 0
48 0
49 6
50 5
51 15
52 27
53 7
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [27], in <cell line: 14>()
     12     print(guess_count, lotto_number)
     13 fig, (ax1, ax2) = plt.subplots(1,2)
---> 14 ax1.plot(plt.scatter(guess_count,lotto_number))
     15 ax2.plot(plt.hist(guess_count, lotto_number))

File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/axes/_axes.py:1634, in Axes.plot(self, scalex, scaley, data, *args
 ↪**kwargs)
   1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
-> 1634     self.add_line(line)
   1635 self._request_autoscale_view(scalex=scalex, scaley=scaley)
   1636 return lines

File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/axes/_base.py:2288, in _AxesBase.add_line(self, line)
   2285 if line.get_clip_path() is None:
   2286     line.set_clip_path(self.patch)
-> 2288 self._update_line_limits(line)
   2289 if not line.get_label():
   2290     line.set_label(f'_child{len(self._children)}')

File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/axes/_base.py:2311, in _AxesBase._update_line_limits(self, line)
   2307 def _update_line_limits(self, line):
   2308     """
   2309     Figures out the data limit of the given line, updating self.dataLim
   2310     """
-> 2311     path = line.get_path()
   2312     if path.vertices.size == 0:
   2313         return
```

```
File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/lines.py:999, in Line2D.get_path(self)
    997 """Return the `~matplotlib.path.Path` associated with this line."""
    998 if self._invalidy or self._invalidx:
--> 999     self.recache()
   1000 return self._path

File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/lines.py:657, in Line2D.recache(self, always)
    655 if always or self._invalidy:
    656     yconv = self.convert_yunits(self._yorig)
--> 657     y = _to_unmasked_float_array(yconv).ravel()
    658 else:
    659     y = self._y

File /usr/local/Miniconda3-py39_4.10.3-Linux-x86_64/lib/python3.9/site-packages
 ↪matplotlib/cbook/__init__.py:1298, in _to_unmasked_float_array(x)
   1296     return np.ma.asarray(x, float).filled(np.nan)
   1297 else:
-> 1298     return np.asarray(x, float)

TypeError: float() argument must be a string or a number, not 'PathCollection'
```



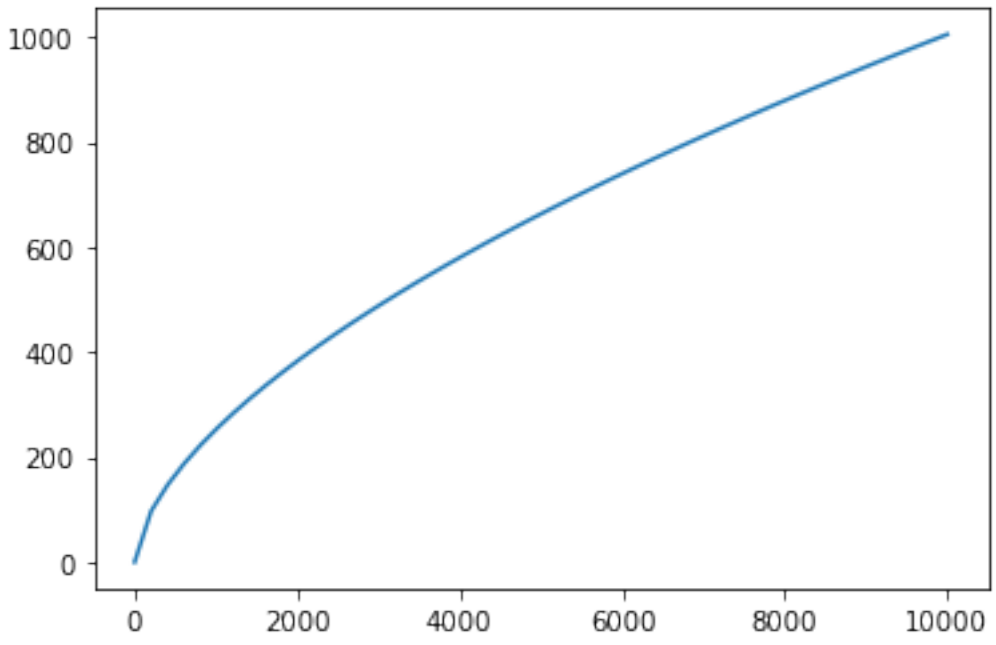
1.0.3 Oppgave 3 (20 poeng)

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`

[28]:
```
##oppgave. 3a
#laster ned aktuelle pakker
import sympy as sp

L = np.linspace(0,10000)
#fyller utgitte verdier i funksjone og kaller den for prodfunk
a=0.6
R=2

def produktfunksjon(L,a,R):
    return (2*R*L**a)

plt.plot(L,produktfunksjon(L,a,R))
plt.show
```

[28]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



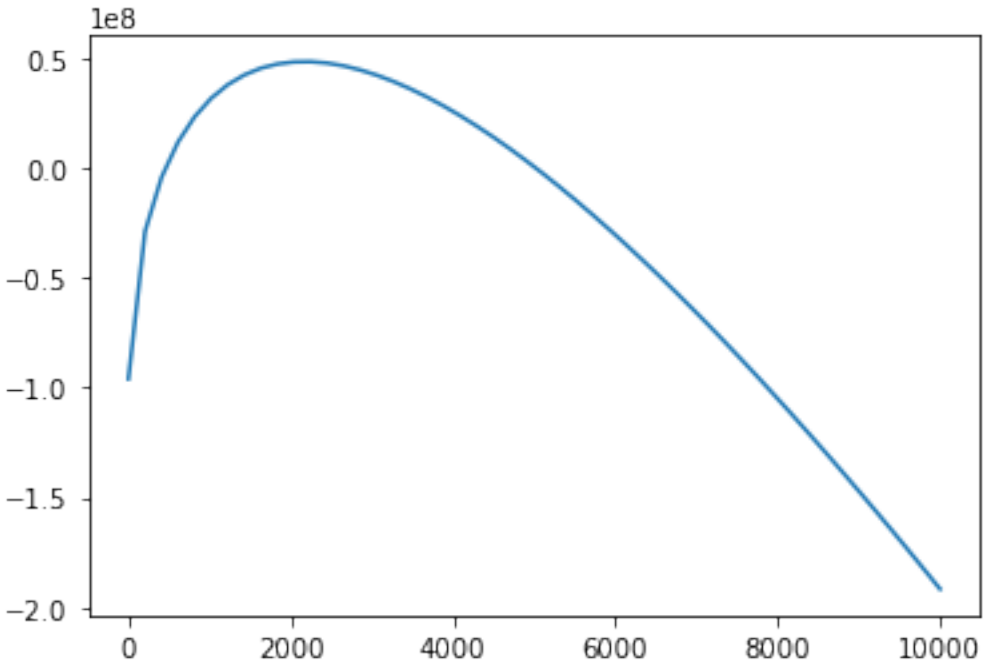
b) anta at profittfunksjonen til denne bedriften er $profit = f(L, a, R)p - wL - cR - K$, hvor

- `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)

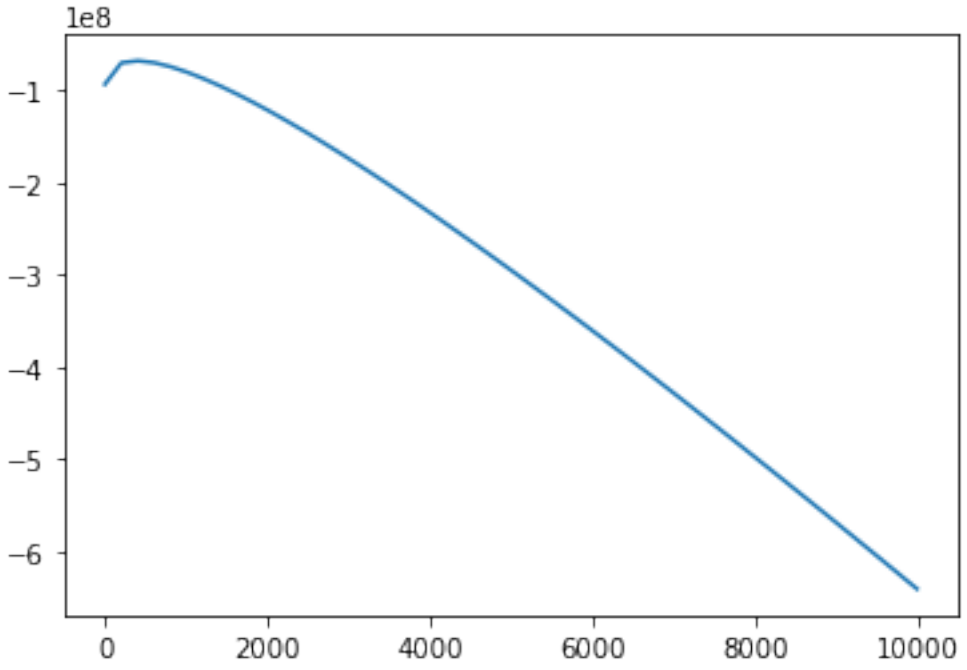
[31]:
```
#opgave. 3b #
L = np.linspace(0,10000)

R=6
p=300000
w=100000
c=1000000
K=90000000

def profittfunksjon(L,a,R,p,w,c,K):
    return (2*R*L**a)*p-w*L-c*R-K

plt.plot(L,profittfunksjon(L,a,R,p,w,c,K))
plt.show
```

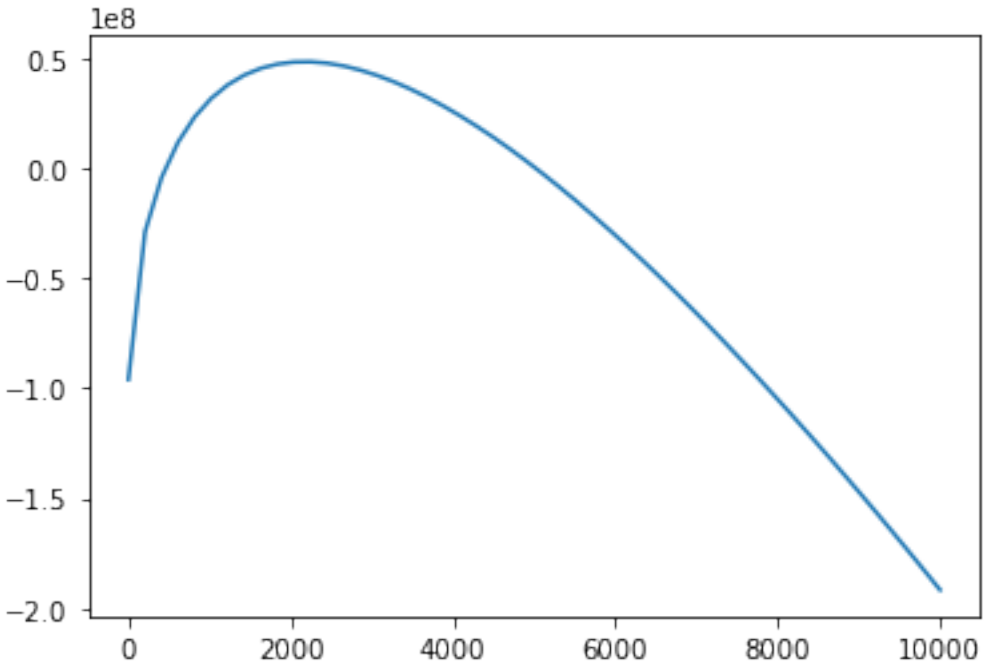
[31]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

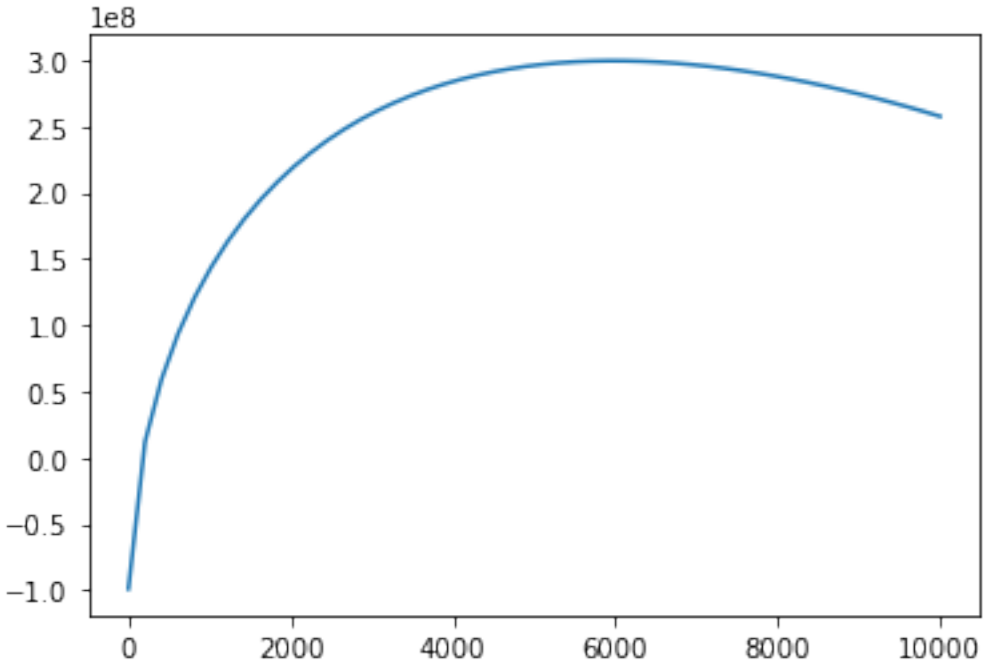


c) Plot profitttfunksjonen for antall robostmaskiner `R=[3, 6, 9]` i samme plot (dvs at tre profittfunksjoner vises sammen). Bruk av "for loops" for å gjøre dette belønnes

[32]:
```
#oppgave. 3c #
for R in [3, 6, 9]:
    plt.plot(L,profittfunksjon(L,a,R,p,w,c,K))
    plt.show()
```



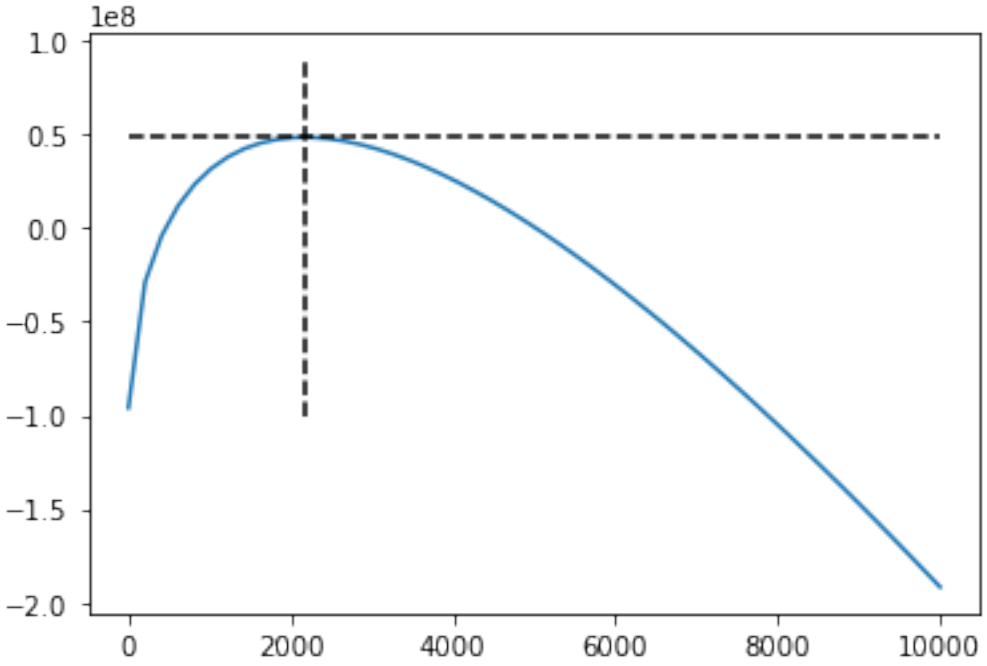




d) finn profittmaksimum og optimal antall arbeidere ved hjelp av derivasjon med samme forutsetninger som i (1b). Bruk `sympy`-pakken til dette

[33]:
```
#oppgave. 3d #pakker du kan få bruk for
import sympy as sp
from sympy.solvers import solve

R=6
L = np.linspace(0,10000)

plt.plot(L,profittfunksjon(L,a,R,p,w,c,K))

#plotter stiplede linjer for å vise mest profitabelt antall arbeidere
plt.vlines(2168, -100000000, 90000000, linestyles ="dashed", colors ="k")
plt.hlines(49000000, 0, 10000, linestyles ="dashed", colors ="k")

plt.show
```

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

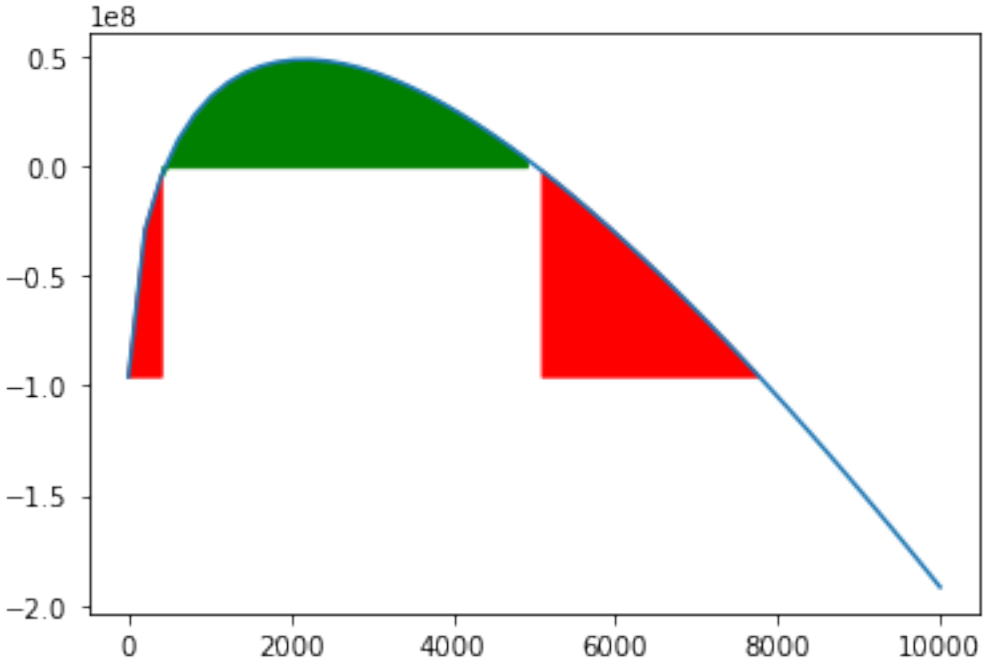


e) vis figurativt med bruk av `fill_between` arealet hvor man taper penger (i rødt) og hvor man tjener penger (i grønt). Marker også profittmaksimum og antall arbeidere i profittmaksimum - gjerne ved bruk av `vlines`. Bruk ellers samme forutsetninger for argumentene som i oppgave (1b)

[34]:
```
#oppgave. 3e

plt.plot(L,profittfunksjon(L,a,R,p,w,c,K))
```

```
#farger inn der det er profitt og ikke ved hjelp av fill.between
plt.fill_between(L,profittfunksjon(L,a,R,p,w,c,K),0,where=(L>400) &␣
 ↪(L<=5100),color='g')
plt.
 ↪fill_between(L,profittfunksjon(L,a,R,p,w,c,K),-96000000,where=(profittfunksjon(L,a,R,p,w,c,
 ↪& (profittfunksjon(L,a,R,p,w,c,K)<=0),color='r')

plt.show
```

[34]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



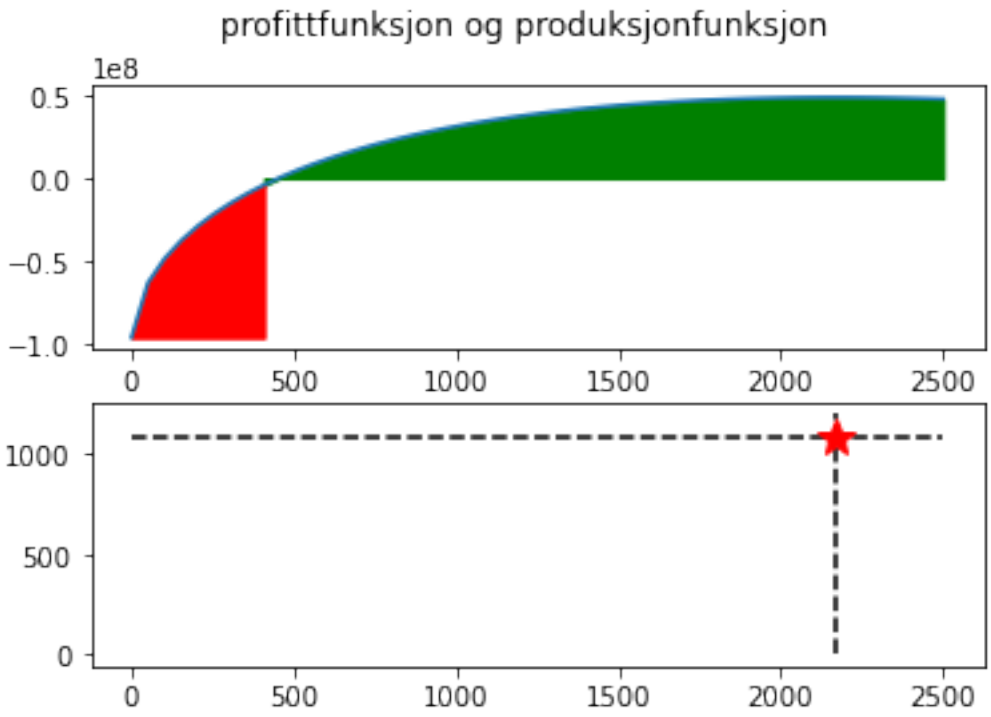
f) Plot nå to figurer sammen der du viser hva optimal antall arbeidere gir i profitt (slik som i (2e)) og produksjon av antall biler (som du får fra produktfunksjonen). Marker optimum med vlines. Ha grafen med profittfunksjonen over grafen med produktfunksjonen. Du kan bruke `fig, (ax1, ax2) = plt.subplots(2)`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

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

[35]:
```
#oppgave. 3f #definerer nye verdier for L slik at figuren blir ryddig
L = np.linspace(0,2500)

fig, (ax1, ax2) = plt.subplots(2)
fig.suptitle('profittfunksjon og produksjonfunksjon')
```

```
#farger inn der det er profitt og ikke ved hjelp av fill.between
ax1.fill_between(L,profittfunksjon(L,a,R,p,w,c,K),0,where=(L>400) &␣
 ↪(L<=5100),color='g')
ax1.
 ↪fill_between(L,profittfunksjon(L,a,R,p,w,c,K),-96000000,where=(profittfunksjon(L,a,R,p,w,c,
 ↪& (profittfunksjon(L,a,R,p,w,c,K)<=0),color='r')

#plotter profittfuksjonen
ax1.plot(L,profittfunksjon(L,a,R,p,w,c,K))

#plotter stiplede linjer på produktfunksjonen får å vise...
ax2.vlines(2168, 0, 1200, linestyles ="dashed", colors ="k")
ax2.hlines(1080, 0, 2500, linestyles ="dashed", colors ="k")

#legger til et punkt ved mest profitabelt antall biler
ax2.plot(2168, 1080, marker='*', color='r', ms=15)

#plotter produktfunksjonen
ax2.plot(L, produktfunksjon(L_with_optimal,a,R))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [35], in <cell line: 22>()
     19 ax2.plot(2168, 1080, marker='*', color='r', ms=15)
     21 #plotter produktfunksjonen
---> 22 ax2.plot(L, produktfunksjon(L_with_optimal,a,R))

NameError: name 'L_with_optimal' is not defined
```



1.0.4 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`.

[1]:
```
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.

[ ]:
```
inv_data.columns = ['unit',   'sector',  'na_item', 'country'] +␣
 ↪list(range(2010, 2022)) #v1
```

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

[36]:
```
#oppgave. 4
import eurostat

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

```
inv_data
```

[36]:

	freq	unit	sector	na_item	geo\TIME_PERIOD	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
0	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	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	A	PC	S14_S15	IRG_S14_S15	ES	9.66	7.85	6.48									
11	A	PC	S14_S15	IRG_S14_S15	EU27_2020	9.07	8.95	8.56									
12	A	PC	S14_S15	IRG_S14_S15	EU28	8.44	8.34	7.94									
13	A	PC	S14_S15	IRG_S14_S15	FI	11.66	12.18	12.12									
14	A	PC	S14_S15	IRG_S14_S15	FR	9.31	9.46	9.31									
15	A	PC	S14_S15	IRG_S14_S15	HR	5.96	5.66	5.53									
16	A	PC	S14_S15	IRG_S14_S15	HU	6.67	5.11	4.80									
17	A	PC	S14_S15	IRG_S14_S15	IE	6.01	4.96	4.12									
18	A	PC	S14_S15	IRG_S14_S15	IS	4.45	4.58	4.81									
19	A	PC	S14_S15	IRG_S14_S15	IT	10.33	9.82	9.20									
20	A	PC	S14_S15	IRG_S14_S15	LT	4.73	5.11	4.70									
21	A	PC	S14_S15	IRG_S14_S15	LU	9.69	11.26	10.43									
22	A	PC	S14_S15	IRG_S14_S15	LV	3.56	4.92	6.17									
23	A	PC	S14_S15	IRG_S14_S15	NL	10.05	9.54	8.47									
24	A	PC	S14_S15	IRG_S14_S15	NO	9.71	10.98	11.74									
25	A	PC	S14_S15	IRG_S14_S15	PL	7.99	7.92	8.24									
26	A	PC	S14_S15	IRG_S14_S15	PT	6.27	5.81	5.07									
27	A	PC	S14_S15	IRG_S14_S15	RS	NaN	NaN	NaN									
28	A	PC	S14_S15	IRG_S14_S15	SE	5.86	5.48	4.51									
29	A	PC	S14_S15	IRG_S14_S15	SI	6.92	6.30	5.78									
30	A	PC	S14_S15	IRG_S14_S15	SK	6.32	7.00	6.65									
31	A	PC	S14_S15	IRG_S14_S15	UK	4.96	5.02	4.95									

10	4.84	4.72	4.57	4.60	5.17	5.42	5.54	5.34	6.68
11	8.12	8.06	7.97	8.22	8.42	8.54	8.60	8.33	9.19
12	7.67	7.65	7.55	7.85	8.16	8.24	8.39	NaN	NaN
13	11.49	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50
14	9.27	9.07	8.88	8.99	9.41	9.47	9.62	8.57	9.95
15	5.01	4.78	4.91	4.96	5.01	5.35	6.37	6.40	6.45
16	4.92	5.17	5.68	5.86	6.53	7.19	7.52	8.87	8.30
17	4.29	4.41	4.72	5.26	5.50	6.27	4.92	4.27	5.29
18	5.19	5.57	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	8.42	7.78	7.58	7.68	7.74	7.77	7.64	7.20	8.69
20	5.56	6.04	6.68	6.89	6.51	6.78	7.09	7.04	7.39
21	11.34	12.75	12.34	12.48	12.35	11.45	10.34	9.71	11.33
22	4.62	4.98	5.76	4.87	4.83	5.68	5.85	5.30	4.62
23	7.46	8.10	9.08	10.60	10.80	11.52	12.15	12.22	12.96
24	12.32	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
25	7.59	7.86	7.88	7.52	6.88	5.84	5.93	5.18	6.59
26	4.37	4.51	4.50	4.71	5.05	5.48	5.65	5.69	6.10
27	NaN	NaN	2.63	3.25	3.17	3.21	3.35	2.97	NaN
28	4.61	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
29	5.50	5.65	5.81	5.85	6.28	6.51	6.33	5.64	6.20
30	7.08	6.39	6.10	6.75	6.71	6.76	6.79	6.93	7.04
31	5.27	5.67	5.84	6.16	6.77	6.62	6.81	NaN	NaN

[37]:
```
#oppgave. 4a
inv_data.columns = ['freq', 'unit',  'sector',  'na_item', 'country'] +␣
 ↪list(range(2010, 2022)) #v2
inv_data
```

[37]:

	freq	unit	sector	na_item	country	2010	2011	2012	2013	\
0	A	PC	S14_S15	IRG_S14_S15	AT	8.44	8.60	8.47	8.58	
1	A	PC	S14_S15	IRG_S14_S15	BE	9.82	9.45	9.37	9.03	
2	A	PC	S14_S15	IRG_S14_S15	CH	6.83	6.48	6.29	6.31	
3	A	PC	S14_S15	IRG_S14_S15	CY	13.72	10.73	8.74	7.53	
4	A	PC	S14_S15	IRG_S14_S15	CZ	11.14	9.82	8.74	8.77	
5	A	PC	S14_S15	IRG_S14_S15	DE	8.72	9.51	9.61	9.56	
6	A	PC	S14_S15	IRG_S14_S15	DK	8.50	8.51	7.87	7.35	
7	A	PC	S14_S15	IRG_S14_S15	EA19	9.30	9.21	8.80	8.37	
8	A	PC	S14_S15	IRG_S14_S15	EE	6.14	6.47	6.84	7.52	
9	A	PC	S14_S15	IRG_S14_S15	EL	9.12	8.54	6.28	4.90	
10	A	PC	S14_S15	IRG_S14_S15	ES	9.66	7.85	6.48	4.84	
11	A	PC	S14_S15	IRG_S14_S15	EU27_2020	9.07	8.95	8.56	8.12	
12	A	PC	S14_S15	IRG_S14_S15	EU28	8.44	8.34	7.94	7.67	
13	A	PC	S14_S15	IRG_S14_S15	FI	11.66	12.18	12.12	11.49	
14	A	PC	S14_S15	IRG_S14_S15	FR	9.31	9.46	9.31	9.27	
15	A	PC	S14_S15	IRG_S14_S15	HR	5.96	5.66	5.53	5.01	
16	A	PC	S14_S15	IRG_S14_S15	HU	6.67	5.11	4.80	4.92	
17	A	PC	S14_S15	IRG_S14_S15	IE	6.01	4.96	4.12	4.29	

18	A	PC	S14_S15	IRG_S14_S15	IS	4.45	4.58	4.81	5.19
19	A	PC	S14_S15	IRG_S14_S15	IT	10.33	9.82	9.20	8.42
20	A	PC	S14_S15	IRG_S14_S15	LT	4.73	5.11	4.70	5.56
21	A	PC	S14_S15	IRG_S14_S15	LU	9.69	11.26	10.43	11.34
22	A	PC	S14_S15	IRG_S14_S15	LV	3.56	4.92	6.17	4.62
23	A	PC	S14_S15	IRG_S14_S15	NL	10.05	9.54	8.47	7.46
24	A	PC	S14_S15	IRG_S14_S15	NO	9.71	10.98	11.74	12.32
25	A	PC	S14_S15	IRG_S14_S15	PL	7.99	7.92	8.24	7.59
26	A	PC	S14_S15	IRG_S14_S15	PT	6.27	5.81	5.07	4.37
27	A	PC	S14_S15	IRG_S14_S15	RS	NaN	NaN	NaN	NaN
28	A	PC	S14_S15	IRG_S14_S15	SE	5.86	5.48	4.51	4.61
29	A	PC	S14_S15	IRG_S14_S15	SI	6.92	6.30	5.78	5.50
30	A	PC	S14_S15	IRG_S14_S15	SK	6.32	7.00	6.65	7.08
31	A	PC	S14_S15	IRG_S14_S15	UK	4.96	5.02	4.95	5.27

	2014	2015	2016	2017	2018	2019	2020	2021
0	8.30	8.36	8.29	8.69	8.81	9.03	9.21	9.99
1	9.45	9.31	9.33	9.28	9.35	9.78	9.15	9.94
2	6.20	6.26	6.13	6.07	6.06	5.69	5.42	NaN
3	7.15	6.69	8.02	8.97	11.30	12.98	13.12	13.26
4	8.88	8.84	9.18	7.86	9.00	9.45	9.34	9.24
5	9.65	9.32	9.62	9.48	9.68	9.71	9.95	10.55
6	7.60	7.50	7.46	8.10	8.33	8.57	8.69	9.24
7	8.24	8.09	8.35	8.52	8.71	8.77	8.53	9.51
8	7.79	7.94	8.52	8.99	8.68	9.26	9.94	10.04
9	3.16	2.72	2.75	2.69	2.44	2.59	2.97	3.40
10	4.72	4.57	4.60	5.17	5.42	5.54	5.34	6.68
11	8.06	7.97	8.22	8.42	8.54	8.60	8.33	9.19
12	7.65	7.55	7.85	8.16	8.24	8.39	NaN	NaN
13	10.88	10.49	11.57	12.09	12.50	12.25	11.91	12.50
14	9.07	8.88	8.99	9.41	9.47	9.62	8.57	9.95
15	4.78	4.91	4.96	5.01	5.35	6.37	6.40	6.45
16	5.17	5.68	5.86	6.53	7.19	7.52	8.87	8.30
17	4.41	4.72	5.26	5.50	6.27	4.92	4.27	5.29
18	5.57	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	7.78	7.58	7.68	7.74	7.77	7.64	7.20	8.69
20	6.04	6.68	6.89	6.51	6.78	7.09	7.04	7.39
21	12.75	12.34	12.48	12.35	11.45	10.34	9.71	11.33
22	4.98	5.76	4.87	4.83	5.68	5.85	5.30	4.62
23	8.10	9.08	10.60	10.80	11.52	12.15	12.22	12.96
24	11.82	11.35	12.34	13.02	12.22	11.89	11.29	11.11
25	7.86	7.88	7.52	6.88	5.84	5.93	5.18	6.59
26	4.51	4.50	4.71	5.05	5.48	5.65	5.69	6.10
27	NaN	2.63	3.25	3.17	3.21	3.35	2.97	NaN
28	4.69	5.69	6.15	6.77	6.24	5.88	6.48	6.84
29	5.65	5.81	5.85	6.28	6.51	6.33	5.64	6.20
30	6.39	6.10	6.75	6.71	6.76	6.79	6.93	7.04

```
31   5.67   5.84   6.16   6.77   6.62   6.81    NaN    NaN
```

[38]:
```
inv_data=inv_data.drop(inv_data.columns[[0,1,2,3]], axis = 1)
inv_data
```

[38]:

	country	2010	2011	2012	2013	2014	2015	2016	2017	2018	\
0	AT	8.44	8.60	8.47	8.58	8.30	8.36	8.29	8.69	8.81	
1	BE	9.82	9.45	9.37	9.03	9.45	9.31	9.33	9.28	9.35	
2	CH	6.83	6.48	6.29	6.31	6.20	6.26	6.13	6.07	6.06	
3	CY	13.72	10.73	8.74	7.53	7.15	6.69	8.02	8.97	11.30	
4	CZ	11.14	9.82	8.74	8.77	8.88	8.84	9.18	7.86	9.00	
5	DE	8.72	9.51	9.61	9.56	9.65	9.32	9.62	9.48	9.68	
6	DK	8.50	8.51	7.87	7.35	7.60	7.50	7.46	8.10	8.33	
7	EA19	9.30	9.21	8.80	8.37	8.24	8.09	8.35	8.52	8.71	
8	EE	6.14	6.47	6.84	7.52	7.79	7.94	8.52	8.99	8.68	
9	EL	9.12	8.54	6.28	4.90	3.16	2.72	2.75	2.69	2.44	
10	ES	9.66	7.85	6.48	4.84	4.72	4.57	4.60	5.17	5.42	
11	EU27_2020	9.07	8.95	8.56	8.12	8.06	7.97	8.22	8.42	8.54	
12	EU28	8.44	8.34	7.94	7.67	7.65	7.55	7.85	8.16	8.24	
13	FI	11.66	12.18	12.12	11.49	10.88	10.49	11.57	12.09	12.50	
14	FR	9.31	9.46	9.31	9.27	9.07	8.88	8.99	9.41	9.47	
15	HR	5.96	5.66	5.53	5.01	4.78	4.91	4.96	5.01	5.35	
16	HU	6.67	5.11	4.80	4.92	5.17	5.68	5.86	6.53	7.19	
17	IE	6.01	4.96	4.12	4.29	4.41	4.72	5.26	5.50	6.27	
18	IS	4.45	4.58	4.81	5.19	5.57	NaN	NaN	NaN	NaN	
19	IT	10.33	9.82	9.20	8.42	7.78	7.58	7.68	7.74	7.77	
20	LT	4.73	5.11	4.70	5.56	6.04	6.68	6.89	6.51	6.78	
21	LU	9.69	11.26	10.43	11.34	12.75	12.34	12.48	12.35	11.45	
22	LV	3.56	4.92	6.17	4.62	4.98	5.76	4.87	4.83	5.68	
23	NL	10.05	9.54	8.47	7.46	8.10	9.08	10.60	10.80	11.52	
24	NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22	
25	PL	7.99	7.92	8.24	7.59	7.86	7.88	7.52	6.88	5.84	
26	PT	6.27	5.81	5.07	4.37	4.51	4.50	4.71	5.05	5.48	
27	RS	NaN	NaN	NaN	NaN	NaN	2.63	3.25	3.17	3.21	
28	SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24	
29	SI	6.92	6.30	5.78	5.50	5.65	5.81	5.85	6.28	6.51	
30	SK	6.32	7.00	6.65	7.08	6.39	6.10	6.75	6.71	6.76	
31	UK	4.96	5.02	4.95	5.27	5.67	5.84	6.16	6.77	6.62	

	2019	2020	2021
0	9.03	9.21	9.99
1	9.78	9.15	9.94
2	5.69	5.42	NaN
3	12.98	13.12	13.26
4	9.45	9.34	9.24
5	9.71	9.95	10.55
6	8.57	8.69	9.24

```
7    8.77   8.53   9.51
8    9.26   9.94  10.04
9    2.59   2.97   3.40
10   5.54   5.34   6.68
11   8.60   8.33   9.19
12   8.39    NaN    NaN
13  12.25  11.91  12.50
14   9.62   8.57   9.95
15   6.37   6.40   6.45
16   7.52   8.87   8.30
17   4.92   4.27   5.29
18    NaN    NaN    NaN
19   7.64   7.20   8.69
20   7.09   7.04   7.39
21  10.34   9.71  11.33
22   5.85   5.30   4.62
23  12.15  12.22  12.96
24  11.89  11.29  11.11
25   5.93   5.18   6.59
26   5.65   5.69   6.10
27   3.35   2.97    NaN
28   5.88   6.48   6.84
29   6.33   5.64   6.20
30   6.79   6.93   7.04
31   6.81    NaN    NaN
```

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`)

[39]:
```
#oppgave. 4b
inv_data=inv_data.dropna()
inv_data.set_index('country', inplace=True)
inv_data
```

[39]:
```
            2010   2011   2012   2013   2014   2015   2016   2017   2018  \
country
AT          8.44   8.60   8.47   8.58   8.30   8.36   8.29   8.69   8.81
BE          9.82   9.45   9.37   9.03   9.45   9.31   9.33   9.28   9.35
CY         13.72  10.73   8.74   7.53   7.15   6.69   8.02   8.97  11.30
CZ         11.14   9.82   8.74   8.77   8.88   8.84   9.18   7.86   9.00
DE          8.72   9.51   9.61   9.56   9.65   9.32   9.62   9.48   9.68
DK          8.50   8.51   7.87   7.35   7.60   7.50   7.46   8.10   8.33
EA19        9.30   9.21   8.80   8.37   8.24   8.09   8.35   8.52   8.71
EE          6.14   6.47   6.84   7.52   7.79   7.94   8.52   8.99   8.68
EL          9.12   8.54   6.28   4.90   3.16   2.72   2.75   2.69   2.44
ES          9.66   7.85   6.48   4.84   4.72   4.57   4.60   5.17   5.42
EU27_2020   9.07   8.95   8.56   8.12   8.06   7.97   8.22   8.42   8.54
FI         11.66  12.18  12.12  11.49  10.88  10.49  11.57  12.09  12.50
```

FR	9.31	9.46	9.31	9.27	9.07	8.88	8.99	9.41	9.47
HR	5.96	5.66	5.53	5.01	4.78	4.91	4.96	5.01	5.35
HU	6.67	5.11	4.80	4.92	5.17	5.68	5.86	6.53	7.19
IE	6.01	4.96	4.12	4.29	4.41	4.72	5.26	5.50	6.27
IT	10.33	9.82	9.20	8.42	7.78	7.58	7.68	7.74	7.77
LT	4.73	5.11	4.70	5.56	6.04	6.68	6.89	6.51	6.78
LU	9.69	11.26	10.43	11.34	12.75	12.34	12.48	12.35	11.45
LV	3.56	4.92	6.17	4.62	4.98	5.76	4.87	4.83	5.68
NL	10.05	9.54	8.47	7.46	8.10	9.08	10.60	10.80	11.52
NO	9.71	10.98	11.74	12.32	11.82	11.35	12.34	13.02	12.22
PL	7.99	7.92	8.24	7.59	7.86	7.88	7.52	6.88	5.84
PT	6.27	5.81	5.07	4.37	4.51	4.50	4.71	5.05	5.48
SE	5.86	5.48	4.51	4.61	4.69	5.69	6.15	6.77	6.24
SI	6.92	6.30	5.78	5.50	5.65	5.81	5.85	6.28	6.51
SK	6.32	7.00	6.65	7.08	6.39	6.10	6.75	6.71	6.76

	2019	2020	2021
country			
AT	9.03	9.21	9.99
BE	9.78	9.15	9.94
CY	12.98	13.12	13.26
CZ	9.45	9.34	9.24
DE	9.71	9.95	10.55
DK	8.57	8.69	9.24
EA19	8.77	8.53	9.51
EE	9.26	9.94	10.04
EL	2.59	2.97	3.40
ES	5.54	5.34	6.68
EU27_2020	8.60	8.33	9.19
FI	12.25	11.91	12.50
FR	9.62	8.57	9.95
HR	6.37	6.40	6.45
HU	7.52	8.87	8.30
IE	4.92	4.27	5.29
IT	7.64	7.20	8.69
LT	7.09	7.04	7.39
LU	10.34	9.71	11.33
LV	5.85	5.30	4.62
NL	12.15	12.22	12.96
NO	11.89	11.29	11.11
PL	5.93	5.18	6.59
PT	5.65	5.69	6.10
SE	5.88	6.48	6.84
SI	6.33	5.64	6.20
SK	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`.

[40]:
```
#oppgave. 4c
#bruker isin for å finne noridkse landene
nordisk=inv_data.index.isin(['NO', 'SE', 'DK', 'FI'])
nordisk
```

[40]:
```
array([False, False, False, False, False,  True, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False,  True, False, False,  True, False, False])
```

[41]:
```
#gjør om datasettet slik at det kun inneholder nordiske landene
inv_data= inv_data[nordisk==True]
inv_data
```

[41]:

country	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
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
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

[42]:
```
#bruker transpose for å bytte om på kolonner og rader
inv_data=inv_data.transpose()
inv_data
```

[42]:

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

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”

[43]:
```
#oppgave. 4d
#bruker mean som tar gjennomsnittet av radene
inv_data['mean'] = inv_data.mean(axis=1)
inv_data
```

[43]:

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

[44]:
```
ax=inv_data.plot()
plt.xlabel('år')
plt.ylabel('investering')
ax.legend(loc='lower right',frameon=False)
```

[44]:
```
<matplotlib.legend.Legend at 0x7f5f191534c0>
```