```
# %% [markdown]
# # Mappeoppgave 1
# ## <u>Beskrivelse
# Les oppgaveteksten nøye. Se hvordan man leverer oppgaven <th><a
href='https://uit-sok-1003-h22.github.io/semesteroppgave.html'>her</a> </th> og
<th><a href='https://uit-sok-1003-h22.github.io/github.html'>her</a>. Husk at
den skal leveres både som jupyter-fil og som PDF. Kommenter kodene du skriver i
alle oppgaver og vær nøye på å definere aksene mm i figurer. I noen av
oppgavetekstene står det hint, men det betyr ikke at de ikke kan løses på andre
måter
#
# ##### For å hente denne filen til Jupyter gjør du slik:
#
# <ol>
# <li>Åpne et "terminalvindu"
# <li>Gå til hjemmeområdet ditt
#
#
#     [user@jupty02 ~]$ cd
# <li>Lag en ny mappe på ditt hjemmeområde ved å skrive inn i terminalvinduet
#
#     [user@jupty02 ~]$ mkdir SOK-1003-eksamen-2022-mappe1
#
# <li>Gå så inn i den mappen du har laget ved å skrive
#
#     [user@jupty02 ~]$ cd SOK-1003-eksamen-2022-mappe1
# <li> Last ned kursmateriellet ved å kopiere inn følgende kommando i
kommandovinduet:
#
#     [user@jupty02 sok-1003]$ git clone
https://github.com/uit-sok-1003-h22/mappe/
# </ol>
# <br> Oppgi gruppenavn m/ medlemmer på epost ole.k.aars@uit.no innen 7/10, så
blir dere satt opp til tidspunkt for presentasjon 19/10.
# <br>Bruk så denne filen til å gjøre besvarelsen din. Ved behov; legg til flere
celler ved å trykke "b"
# </br>
# <hr>

# %% [markdown]
# ## <u> Oppgavene
#
# ### Oppgave 1 (5 poeng)
# a) Lag en kort fortelling i en python kode som inkluderer alle de fire typer
variabler vi har lært om i kurset. Koden skal kunne kjøres med print(). Koden
burde inneholde utregninger av elementer du har definert
#

# %%
A = 56
B = 36.58
C = 'There once was '
D = ' workers, they had meetings at three. On average only '
E = ' workers showed up. '
```

```
F = B > A/2

if F == True:
    G = 'More than half of the workers showed up.'
else:
    G = 'Less more than half of the workers showed up.'



Story = C + str(A) + D + str(B) + E + G

print(Story)

# %% [markdown]
# ### Oppgave 2 (10 poeng)

# %% [markdown]
# Leieprisene i landet har steget de siste månedene. Ved å bruke realistiske
tall<br>
# a) Lag tilbuds og etterspørselsfunksjoner for leie av bolig (Bruk av
ikke-lineære funksjoner belønnes). <br>
#  <br> Definer funksjonene slik at det er mulig å finne en likevekt
#
#
#
#

# %%
import matplotlib.pyplot as plt
import numpy as np


def Leie(q):
    return (q**2)

def Utleie(q):
    return (q - 250)**2

q = np.linspace(0, 250, 100000)


# %% [markdown]
# b) Vis at disse er henholdvis fallende og stigende, ved bruk av
# - Regning
# - figurativt (matplotlib)
# Husk å markere aksene tydelig og at funskjonene er definert slik at linjene
krysser

# %%
print("--------------------")
print("Viser at tilbudskurven er stigende")
print(Leie(0), "kr,", Leie(25), "kr,", Leie(50), "kr,", Leie(75), "kr,",
Leie(100), "kr,", Leie(125), "kr,", Leie(150), "kr,", Leie(175), "kr,",
Leie(200), "kr,", Leie(225), "kr,", Leie(250), "kr.")
```

```
print("-------------------")
print("Viser at etterspørselskurven er fallende")
print(Utleie(0), "kr,", Utleie(25), "kr,", Utleie(50), "kr,", Utleie(75), "kr,",
Utleie(100), "kr,", Utleie(125), "kr,", Utleie(150), "kr,", Utleie(175), "kr,",
Utleie(200), "kr,", Utleie(225), "kr,", Utleie(250), "kr.")
print("-------------------")

plt.subplots(figsize=(18, 6))
plt.plot(q, Leie(q), label = "Tilbudskurve")
plt.plot(q, Utleie(q), label = "Etterspørselskurve")
plt.title("Leie og Utleie")
plt.legend(frameon = False)
plt.xlabel("Utleieboliger")
plt.ylabel("Utleiepris")

# Definition of tick_val and tick_lab
tick_val = np.arange(0, 275, 25)
tick_lab = [x for x in range(0, 275, 25)]

tick_valy = np.arange(0, 75000, 5000)
tick_laby = [x for x in range(0, 75000, 5000)]

#Komma i y-aksen
ylabels = [f'{label:,}' for label in tick_laby]

# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)
plt.yticks(tick_valy, ylabels)

plt.show()

# %% [markdown]
# c) Kommenter funksjonene og likevekten. Vis gjerne figurativt hvor likevekten
er ved bruk av scatter

# %%
plt.subplots(figsize=(18, 6))
plt.plot(q, Leie(q), label = "Leiekurve", color = "red")
plt.plot(q, Utleie(q), label = "Utleiekurve", color = "blue")
plt.title("Leie og Utleie")
plt.legend(frameon = False)
plt.xlabel("Utleieboliger")
plt.ylabel("Utleiepris")

plt.scatter(125, 15666, color = "black")

# Definition of tick_val and tick_lab
tick_val = np.arange(0, 275, 25)
tick_lab = [x for x in range(0, 275, 25)]

tick_valy = np.arange(0, 75000, 5000)
tick_laby = [x for x in range(0, 75000, 5000)]
```

```
#Komma i y-aksen
ylabels = [f'{label:,}' for label in tick_laby]

# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)
plt.yticks(tick_valy, ylabels)

plt.show()

# %% [markdown]
# ### Oppgave 3 (15 poeng)

# %% [markdown]
# SSB har omfattende data på befolkningsutvikling
(https://www.ssb.no/statbank/table/05803/tableViewLayout1/). Disse dataene skal
du bruke i de neste deloppgavene.
#
# a) lag lister av følgende variabler: "Befolkning 1. januar", "Døde i alt",
"Innflyttinger" og "Utflyttinger". Velg selv variabelnavn når du definerer dem i
python. Første element i hver liste skal være variabelnavnet. Bruk tall for
perioden 2012-2021. Lag så en liste av disse listene. Du kan kalle den "ssb".
# <br><br>
# <b>Hint:</b> når du skal velge variabler på SSB sin nettside må du holde inne
ctrl for å velge flere variabler.
#

# %%
import numpy as np

ssb =   [[  "Befolkning",           4985870,    5051275,        5109056,
5165802,        5213985,        5258317,        5295619,        5328212,
5367580,        5391369],
        [   "Døde i alt",           41992,          41282,          40394,
   40727,           40726,          40774,          40840,          40684,
   40611,           42002],
        [   "Innflyttinger",    78570,      75789,          70030,
67276,      66800,          58192,          52485,          52153,
38071,      53947],
        [   "Utflyttinger",         31227,          35716,          31875,
   37474,           40724,          36843,          34382,          26826,
   26744,           34297]]


# %% [markdown]
# b) konverter "ssb" til en numpy matrise og gi den et nytt navn

# %%
ssb_np = np.array(ssb)

# %% [markdown]
# c) Putt alle tallene inn i en egen matrise og konverter disse til int

# %%
Befolkning = ssb_np[0,1:]
```

```
Døde = ssb_np[1,1:]
Innflytt = ssb_np[2,1:]
Utflytt = ssb_np[3,1:]

int_Befolkning = Befolkning.astype(int)
int_Døde = Døde.astype(int)
int_Innflytt = Innflytt.astype(int)
int_Utflytt = Utflytt.astype(int)

print(ssb_np)



# %% [markdown]
# d) vis befolkningsutviklingen grafisk for de gjeldene årene ved bruk av
matplotlib, og mer spesifikt "fig, ax = plt.subplots()". Vis befolkning på
y-aksen i millioner

# %%
import matplotlib.pyplot as plt

m = 1000000

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

ax.plot(int_Befolkning/m, label="Befolkning i millioner")

plt.legend(loc='upper center', frameon = True, edgecolor = "black")

plt.title('Befolkningsvekst 2012-2021')
plt.xlabel('År')
plt.ylabel('Befolkning i millioner')

# Definition of tick_val and tick_lab
tick_val = np.arange(0, 10, 1)
tick_lab = [x for x in range(2012, 2022)]


# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)

#Limit
plt.xlim(0, 10)
plt.ylim(4.9, 5.6)

plt.show()

# %% [markdown]
# e) Lag det samme plottet ved bruk av oppslag. Hva er fordelen med dette?

# %%
ssb_dict=dict()
ssb_dict['Befolkning']=int_Befolkning[:]
ssb_dict['Døde']=int_Døde[:]
```

```
ssb_dict['Innflyttet']=int_Innflytt[:]
ssb_dict['Utflyttet']=int_Utflytt[:]


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

ax.plot(ssb_dict['Befolkning']/m, label="Befolkning i millioner")

plt.legend(loc='upper center', frameon = True, edgecolor = "black")

plt.title('Befolkningsvekst 2012-2021')
plt.xlabel('År')
plt.ylabel('Befolkning i millioner')

# Definition of tick_val and tick_lab
tick_val = np.arange(0, 10, 1)
tick_lab = [x for x in range(2012, 2022)]

# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)

#Limit
plt.xlim(0, 10)
plt.ylim(4.9, 5.6)

plt.show()

# Fordelen er at det er enklere å referere til data, legge til ny data.


# %% [markdown]
# f) Hva er den relative befolkningstilveksten utenom fødsler (dvs. innvandring/utvandring)? Definer en ny array og legg den til i oppslaget du laget i oppgaven tidligere. Kall den "rel_immigration". Plot denne sammen med grafen du laget i (d).

# %%
from cProfile import label

#For å dele på tusen
t = 1000


ssb_dict['rel_immigration']=int_Innflytt[:] - int_Utflytt[:]

fig, ax1 = plt.subplots(figsize=(18, 6))

ax2 = ax1.twinx()

ax1.plot(ssb_dict['Befolkning']/m, color='red', label = 'Befolkning i millioner')

ax2.plot(ssb_dict['rel_immigration']/t, color='blue', label = 'Netto innflytting i tusen')
```

```
ax1.legend(loc='upper center', frameon = True, edgecolor = "black")
ax2.legend(loc=(0.4295,0.875), frameon = True, edgecolor = "black")
plt.title('Befolkningsvekst og relativ innvandring 2012-2021')

tick_val = np.arange(0, 10, 1)
tick_lab = [x for x in range(2012, 2022)]


plt.xticks(tick_val, tick_lab)


ax1.set_xlim(0, 10)
ax1.set_ylim(4.9, 5.6)
ax2.set_ylim(0, 60)

ax1.set_xlabel("År")
ax1.set_ylabel("Befolkning i millioner")
ax2.set_ylabel("Relativ Innvandring i tusen")




plt.show()

# %% [markdown]
# g) ekstrapoeng. Kan plotte de samme tallene (dvs "rel_immigration" og
"befolkning" sammen med år) i to figurer ved siden av hverandre ved bruk av
"fig, (ax1, ax2) = plt.subplots(1, 2)". Gi grafene ulik farge
#

# %%
#For å dele på tusen
t =1000

fig, (ax1, ax2) = plt.subplots(1,2 , figsize=(18,6), sharex=True)

ax1.plot(ssb_dict['Befolkning']/m, color='red', label = 'Befolkning i
millioner')

ax2.plot(ssb_dict['rel_immigration']/t, color='blue', label = 'Netto innflytting
i tusen')


tick_val = np.arange(0, 10, 1)
tick_lab = [x for x in range(2012, 2022)]

plt.xticks(tick_val, tick_lab)

ax1.set_title("Befolkningsvekst 2012-2021")
ax2.set_title("Relativ innvandring 2012-2021")

ax1.set_xlim(0, 10)
```

```
ax1.set_ylim(4.9, 5.6)
ax2.set_ylim(0, 60)

ax1.set_xlabel("År")
ax2.set_xlabel("År")
ax1.set_ylabel("Befolkning i millioner")
ax2.set_ylabel("Relativ Innvandring i tusen")

ax1.legend(loc='upper center', frameon = True, edgecolor = "black")
ax2.legend(loc='upper center', frameon = True, edgecolor = "black")

plt.show()


# %% [markdown]
# ## Oppgave 4 (20 poeng)

# %% [markdown]
# Et lån består som regel av et månedlig terminbeløp. Dette beløpet er summen av avdrag (nedbetalingen på lånet) og renter. Vi antar månedlig forrenting i alle oppgavene. Dvs. at det er 12 terminer i hvert år.<br><br>
# a) Lag en funksjon som regner ut hvor mye lånet "x" koster deg i renteutgifter for "t" terminer med årlig rente "r" for et serielån. <br> <br>  Siden dette er et serielån, så vil avdragene være like hver måned men renteutgiftene reduseres i takt med avdragene. Renteutgiftene for en gitt termin "t" vil derfor være den årlige renten "r" (delt på antall forrentinger "f") på gjenværende beløp på det tidspunktet.   $renteutgifter_{t} = (x-a*(t-1))*{r/f}$   <br>  <br>
# Det vil si at renteutgiftene første termin er <br>
# $renteutgifter_{1} = (x-a*0)*{r/f}$, og andre termin er <br>
# $renteutgifter_{2} = (x-a*1)*{r/f}$ osv..<br> <br>
# Siden vi er ute etter den totale kostnaden i svaret, må du summere renteutgiftene over alle terminer, det vil si $\sum_{t=1}^N(x-a*(t-1))*{r/f}$. Dette betyr egentlig bare $renteutgifter_{1} + renteutgifter_{2},...,+renteutgifter_{t}$
#
#
# <br>
# <b>Hint:</b> siden terminbeløpet varierer for hver måned (pga at rentene endres), må alle enkeltperioder summeres. Det kan være nyttige å bruke funksjonen np.arange() til dette. Mao, det er ikke nødvendig å bruke sigma ($\sum_{t=1}^N$) i formelen til dette
#

# %%
import numpy as np

#Prinsipal
x = 2500000

#Rente per år
r = 0.025

#Terminer per år
f = 12
```

```
#Antall år
n = 25

#Terminer
t = f*n

#Avdrag per termin
a = x/t

#Resterende prinsipal per termin
RestPerMnd = np.arange(x, 0, -a)

#Rentebataling per termin
rPerMnd = RestPerMnd * r/f

#Totale renter betalt over lånets løpetid
rTotal = sum(rPerMnd)
formTotal= "{:,.2f}".format(rTotal)


print("Totalt rentebeløp betalt er", formTotal, "kr")


# %% [markdown]
# b) regn ut hvor mye lånet koster deg med henholdsvis 10, 20 og 30 års
tilbakebetaling. Anta 1 000 000 kr lånebeløp med 3% rente

# %%
import numpy as np

#Prinsipal
x = 1000000

#Rente per år
r = 0.03

#Terminer per år
f = 12

#Antall år
n1 = 10
n2 = 20
n3 = 30

#Terminer
t1 = f*n1
t2 = f*n2
t3 = f*n3

#Avdrag per termin
a1 = x/t1
a2 = x/t2
a3 = x/t3
```

```
#Resterende prinsipal per termin
RestPerMnd1 = np.arange(x, 0, -a1)
RestPerMnd2 = np.arange(x, 0, -a2)
RestPerMnd3 = np.arange(x, 0, -a3)

#Rentebataling per termin
rPerMnd1 = RestPerMnd1 * r/f
rPerMnd2 = RestPerMnd2 * r/f
rPerMnd3 = RestPerMnd3 * r/f

#Totale renter betalt over lånets løpetid
rTotal1 = sum(rPerMnd1)
formTotal1 = "{:,.2f}".format(rTotal1)

rTotal2 = sum(rPerMnd2)
formTotal2 = "{:,.2f}".format(rTotal2)

rTotal3 = sum(rPerMnd3)
formTotal3 = "{:,.2f}".format(rTotal3)


print("Totalt rentebeløp betalt over 10 år er", formTotal1, "kr")
print("Totalt rentebeløp betalt over 20 år er", formTotal2, "kr")
print("Totalt rentebeløp betalt over 30 år er", formTotal3, "kr")


# %% [markdown]
# c) Vis hva det samme lånet koster som annuitetslån, dvs differansen mellom
alle terminbeløp og lånebeløp.<br> <br> Annuitetslån gir like terminbeløp hver
måned, men renten utgjør en større del av dette beløpet i starten. Terminbeløpet
for et annuitetslån er definert ved formelen:
# $T = x*\frac{r/f} {(1-(1+(r/f))^{-t})}$, hvor x=lånebeløp, r = årlig rente, t
= terminer, f= antall forrentinger
#
#
#
#

# %%
import numpy as np

#Prinsipal
x = 1000000

#Rente per år
r = 0.03

#Terminer per år
f = 12

#Antall år
n1 = 10
n2 = 20
```

```
n3 = 30

#Terminer
t1 = f*n1
t2 = f*n2
t3 = f*n3

#Avdrag per termin
a1 = x*(r/f)/(1-(1+r/f)**(-t1))
a2 = x*(r/f)/(1-(1+r/f)**(-t2))
a3 = x*(r/f)/(1-(1+r/f)**(-t3))

#Avdrag og renter totalt
aTotal1 = a1 * t1
aTotal2 = a2 * t2
aTotal3 = a3 * t3


#Totale renter betalt
rTotal1 = aTotal1 - x
formrTotal1 = "{:,.2f}".format(rTotal1)

rTotal2 = aTotal2 - x
formrTotal2 = "{:,.2f}".format(rTotal2)

rTotal3 = aTotal3 - x
formrTotal3 = "{:,.2f}".format(rTotal3)

print("Totalt rentebeløp betalt over 10 år er", formrTotal1, "kr")
print("Totalt rentebeløp betalt over 20 år er", formrTotal2, "kr")
print("Totalt rentebeløp betalt over 30 år er", formrTotal3, "kr")

# %% [markdown]
# c) Vis hvordan utviklingen i rentekostnader og avdrag på terminer for serielån grafisk ved hjelp av stackplot funksjonen i matplotlib. Anta et bankinnskudd x = 1 000 000 kr, årlig rente r=3% og antall terminer t = 240 (det vil si 20 år). Siden vi må vise utviklingen per termin, husk at "t" også definerer hvilken måned vi er i. Dvs, hvis t=15, har det gått 1 år og 3 mnd med terminer. Se forøvrig relevante formler i oppgave (a)
#
# <br>
#
# <b>Hint1:</b> Siden avdragene er like for alle måneder, kan det være lurt å definere det månedlige avdraget som en liste og gange det med antall perioder.
# <b>Hint2:</b> Siden vi er ute etter både rentekostnader og avdrag hver for seg, kan det være lurt å definere en funksjon for hver av dem.
#

# %%
import numpy as np
import matplotlib.pyplot as plt

#Prinsipal
x = 1000000
```

```
#Rente per år
r = 0.03

#Terminer per år
f = 12

#Antall år
n = 20

#Terminer
t = f*n

#Avdrag per termin
a = x/t

#Avdrag per termin som liste
aPerMnd = [a] * t

#Terminer som år
tPerÅr = np.arange(0, 240, 12)

#Resterende prinsipal per termin
RestPerMnd = np.arange(x, 0, -a)

#Rentebataling per termin
rPerMnd = RestPerMnd * r/f

#Stackplot
plt.subplots(figsize=(18, 6))
plt.stackplot(range(t), aPerMnd, rPerMnd, labels=['Avdrag', 'Renter'], colors
=['y', 'b'])

plt.legend(loc='upper center', frameon = True, edgecolor = "black")

plt.title('Avdrag og renter over 20 år (240 terminer)')
plt.xlabel('År')
plt.ylabel('Beløp i kr')

# Definition of tick_val and tick_lab
tick_val = np.arange(0, 240.00000001, 12)
tick_lab = [x for x in range(0, 21)]

tick_valy = np.arange(0, 7500, 500)
tick_laby = [x for x in range(0, 7500, 500)]

#Komma i y-aksen
ylabels = [f'{label:,}' for label in tick_laby]

# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)
plt.yticks(tick_valy, ylabels)

#Limit
```

```
plt.xlim(0, 240)
plt.ylim(0, 7000)

plt.show()
```