In [1]:
```
# Gruppe 18
# Gustav Gabrielsen, Nesta Ndungu, Johannes Pedersen
```

Mappeoppgave 1

Beskrivelse

Les oppgaveteksten nøye. Se hvordan man leverer oppgaven her og her. 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:

1. Åpne et "terminalvindu"
2. Gå til hjemmeområdet ditt

   `[user@jupty02 ~]$ cd`
3. Lag en ny mappe på ditt hjemmeområde ved å skrive inn i terminalvinduet

   `[user@jupty02 ~]$ mkdir SOK-1003-eksamen-2022-mappe1`
4. Gå så inn i den mappen du har laget ved å skrive

   `[user@jupty02 ~]$ cd SOK-1003-eksamen-2022-mappe1`
5. 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/`

   Oppgi gruppenavn m/ medlemmer på epost o.k.aars@uit.no innen 7/10, så blir dere satt opp til tidspunkt for presentasjon 19/10.
   Bruk så denne filen til å gjøre besvarelsen din. Ved behov; legg til flere celler ved å trykke "b"

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

In [2]:
```
# Lager fire variabler som skal brukes i fortellingen.
```

```
wands = 4                # Int/heltall
cost_wands = 19.99       # Float
prophecy = True          # Bool
eureka = "EUREKA!!!"     # String

print(f"Per og Pål dro på en butikk, der møtte de en mystisk trollmann som solgte 
     f"En tryllestav kostet {cost_wands} kroner. Per og Pål var usikre om dem hadd
     f"Derfor spurte de om trollmannen kunne telle hvor mange penger de hadde. Tro
      f"Til slutt ropte trollmannen {eureka}. Han fortalte at: The Prophecy is {pr
      f"Per og Pål hadde akkurat {cost_wands*4} kroner i pengeboken. Dermed kjøpte
     )
```

```
Per og Pål dro på en butikk, der møtte de en mystisk trollmann som solgte 4 trylle
staver.
En tryllestav kostet 19.99 kroner. Per og Pål var usikre om dem hadde råd til alle
tryllestavene.
Derfor spurte de om trollmannen kunne telle hvor mange penger de hadde. Trollmanne
n telte myntene.
Til slutt ropte trollmannen EUREKA!!!. Han fortalte at: The Prophecy is True.
Per og Pål hadde akkurat 79.96 kroner i pengeboken. Dermed kjøpte de alle tryllest
avene fra trollmannen.
```

Oppgave 2 (10 poeng)

Leieprisene i landet har steget de siste månedene. Ved å bruke realistiske tall

a) Lag tilbuds og etterspørselsfunksjoner for leie av bolig (Bruk av ikke-lineære funksjoner belønnes).

Definer funksjonene slik at det er mulig å finne en likevekt

In [3]:
```
# Definerer en funkskjon for tilbud.
def supply(x):
    return 2500*x
# Så definerer vi en funksjon for etterspørsel.
def demand(x):
    return 50000 - 2500*x
```

b) Vis at disse er henholdvis fallende og stigende, ved bruk av

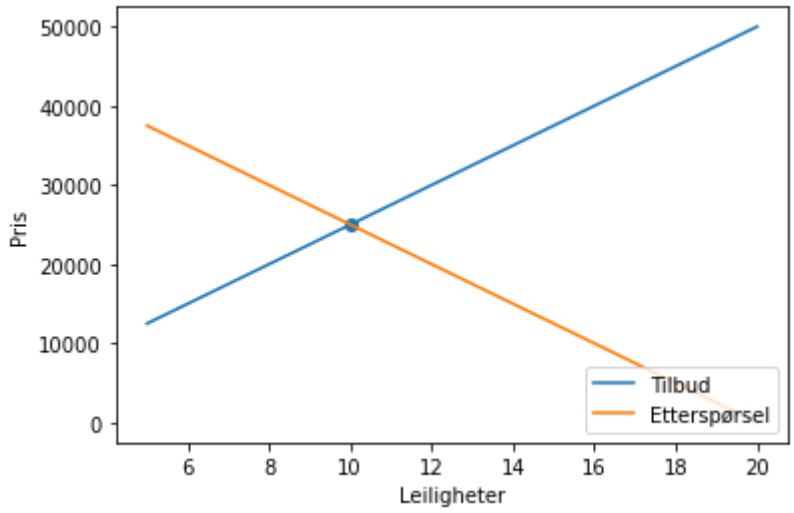
- Regning
- figurativt (matplotlib) Husk å markere aksene tydelig og at funskjonene er definert slik at linjene krysser

In [4]:
```
print("Ser at tilbusfunksjonen er økende.")
print(f"Setter inn for 1:")
print(supply(1))
print(f"Setter inn for 2:")
print(supply(2))
print("Ser at etterspørselen er fallende")
print("Setter inn for 1")
print(demand(1))
print("Setter inn for 2")
print(demand(2))
print("Ser at etterspørselen er fallende")
```

```
Ser at tilbusfunksjonen er økende.
Setter inn for 1:
2500
Setter inn for 2:
5000
Ser at etterspørselen er fallende
Setter inn for 1
47500
Setter inn for 2
45000
Ser at etterspørselen er fallende
```

In [5]:
```
import numpy as np
from matplotlib import pyplot as plt # Importerer pakker.
x = np.linspace(5,20) # Definerer x - aksen.
plt.plot(x,supply(x), label="Tilbud") # Tilbud
plt.plot(x,demand(x),label="Etterspørsel") # Ettersørsel
plt.legend(loc="lower right") # Legend og labels.
plt.ylabel('Pris')
plt.xlabel("Leiligheter")
plt.scatter(10,25000) # Markerer likevektspunkt.
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x7f3b128c2e20>
```



c) Kommenter funksjonene og likevekten. Vis gjerne figurativt hvor likevekten er ved bruk av scatter

In [6]:
```
# Tilbudet øker ved kvantum av leiligher, og etterspørselen synker.
# Har markert likevekt i figuren ved bruk av scatter, her møtes etterspørsel og ti

# Printer ut.
print(f"Likevekt er ved 25000 leiligheter.")
print(f"Det er det punktet hvor tilbudet møter etterspørselen")
```

```
Likevekt er ved 25000 leiligheter.
Det er det punktet hvor tilbudet møter etterspørselen
```

Oppgave 3 (15 poeng)

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".

Hint: når du skal velge variabler på SSB sin nettside må du holde inne ctrl for å velge flere variabler.

In [7]:
```
import numpy as np
import pandas as pd

# Lager lister med variabelene i perioden 2012 - 2021.

årstall = ["årstall", 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021]
befolkning = ["befolkning", 4985870, 5051275, 5109056, 5165802, 5213985, 5258317, 
døde = ["døde", 41992, 41282, 40394, 40727, 40726, 40774, 40840, 40684, 40611, 420
innflyttning =["innflyttning", 78570, 75789, 70030, 67276, 66800, 58192, 52485, 52
utflyttning = ["utflyttning", 31227, 35716, 31875, 37474, 40724, 36843, 34382, 268

ssb = [årstall, befolkning, døde, innflyttning, utflyttning] # Legger listene samm

print(ssb) # Printer ut listen.
```

```
[['årstall', 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021], ['befolk
ning', 4985870, 5051275, 5109056, 5165802, 5213985, 5258317, 5295619, 5328212, 536
7580, 5391369], ['døde', 41992, 41282, 40394, 40727, 40726, 40774, 40840, 40684, 4
0611, 42002], ['innflyttning', 78570, 75789, 70030, 67276, 66800, 58192, 52485, 52
153, 38071, 53947], ['utflyttning', 31227, 35716, 31875, 37474, 40724, 36843, 3438
2, 26826, 26744, 34297]]
```

b) konverter "ssb" til en numpy matrise og gi den et nytt navn

In [8]:
```
np_ssb = np.array(ssb) # Konverterer til array
print(np_ssb) # Printer ut.
```

```
[['årstall' '2012' '2013' '2014' '2015' '2016' '2017' '2018' '2019'
  '2020' '2021']
 ['befolkning' '4985870' '5051275' '5109056' '5165802' '5213985'
  '5258317' '5295619' '5328212' '5367580' '5391369']
 ['døde' '41992' '41282' '40394' '40727' '40726' '40774' '40840' '40684'
  '40611' '42002']
 ['innflyttning' '78570' '75789' '70030' '67276' '66800' '58192' '52485'
  '52153' '38071' '53947']
 ['utflyttning' '31227' '35716' '31875' '37474' '40724' '36843' '34382'
  '26826' '26744' '34297']]
```

c) Putt alle tallene inn i en egen matrise og konverter disse til int

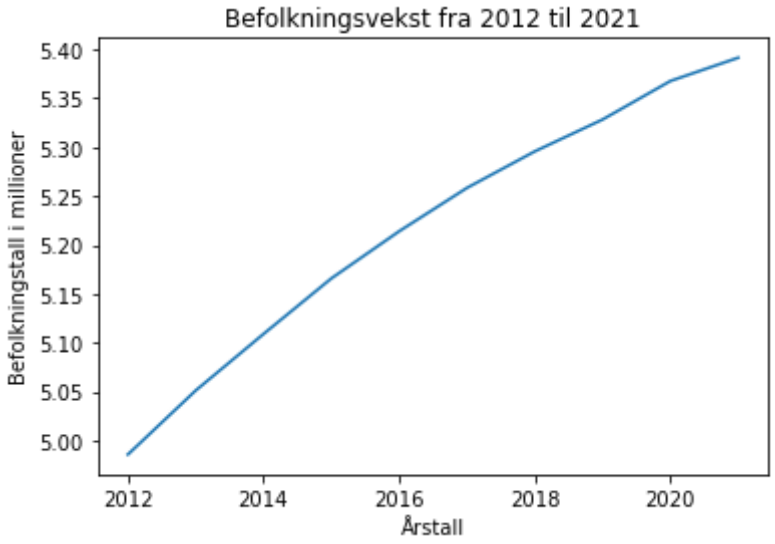
In [9]:
```
# Fjerner første rad slik at vi bare får tall verdier.
np_ssb2 = np_ssb[:,1:]

# Gjør matrisen om til int.
np_ssb_int = np_ssb2.astype(int)

# Printer ut:
print(np_ssb_int)
```

```
[[   2012    2013    2014    2015    2016    2017    2018    2019    2020
     2021]
 [4985870 5051275 5109056 5165802 5213985 5258317 5295619 5328212 5367580
  5391369]
 [  41992   41282   40394   40727   40726   40774   40840   40684   40611
    42002]
 [  78570   75789   70030   67276   66800   58192   52485   52153   38071
    53947]
 [  31227   35716   31875   37474   40724   36843   34382   26826   26744
    34297]]
```

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

In [10]:
```
fig, ax = plt.subplots()  # Bruker subplots
ax.plot(np_ssb_int[0,:],np_ssb_int[1,:]/1000000) # Bruker sliceing for å hente dat
ax.set_title("Befolkningsvekst fra 2012 til 2021") # Legger til tittel og labels:
ax.set_ylabel("Befolkningstall i millioner")
ax.set_xlabel("Årstall")
```

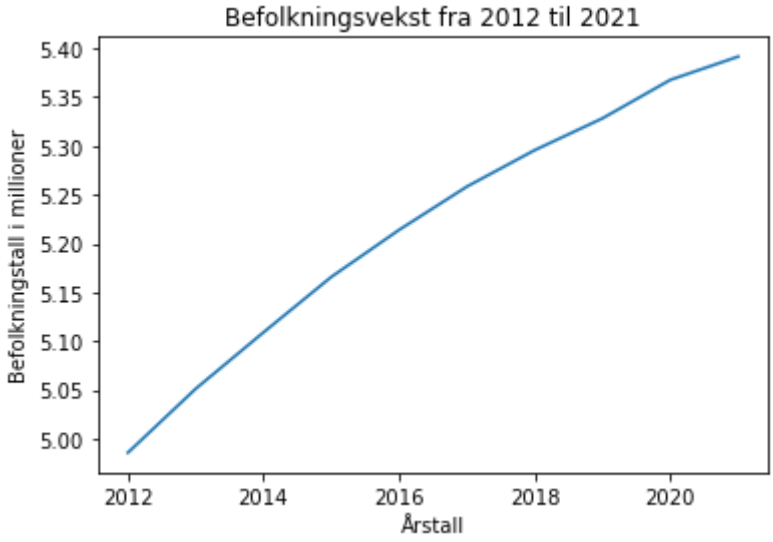
Out[10]:
```
Text(0.5, 0, 'Årstall')
```



e) Lag det samme plottet ved bruk av oppslag. Hva er fordelen med dette?

In [11]:
```
# Bruker slice til å lage oppslag.
a=dict()
a['årstall']=np_ssb_int[0,:]
a['befolkning']=np_ssb_int[1,:]/1000000 # Deler på 1 million.
a['døde']=np_ssb_int[2,:]
a['innflytning']=np_ssb_int[3,:]
a['utflyttning']=np_ssb_int[4,:]

# Dette gjør koden lesbar og mer håndterlig. Man slipper å velge ut kolonner og kar

fig, ax = plt.subplots()

# Bruker nå oppslagene til å plotte figuren.
ax.plot(a['årstall'],a['befolkning'])
ax.set_ylabel("Befolkningstall i millioner")
ax.set_xlabel("Årstall")
ax.set_title("Befolkningsvekst fra 2012 til 2021")
```

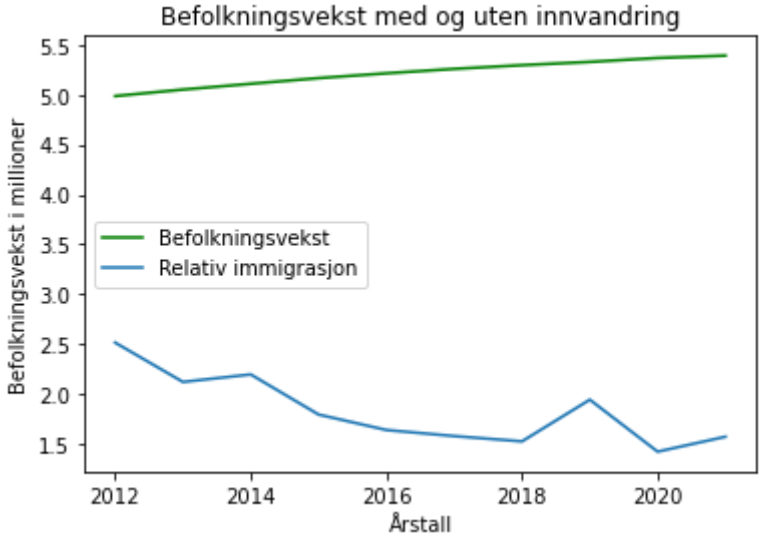
Out[11]:
```
Text(0.5, 1.0, 'Befolkningsvekst fra 2012 til 2021')
```



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).

In [12]:
```
rel_immigration = a['innflytning'] / a['utflyttning']
# Deler innvandring på utvandring og kaller den for rel_immigration.

fig, ax = plt.subplots()
ax.plot(a['årstall'],a['befolkning'], label = "Befolkningsvekst", color = "green")
ax.plot(a['årstall'],rel_immigration, label = "Relativ immigrasjon")
ax.set_ylabel("Befolkningsvekst i millioner") # Gir aksene passende labels.
ax.set_xlabel("Årstall")
ax.set_title("Befolkningsvekst med og uten innvandring") # Gir figuren en passende
ax.legend(loc = "center left")
```

Out[12]:
```
<matplotlib.legend.Legend at 0x7f3b08b5f7f0>
```

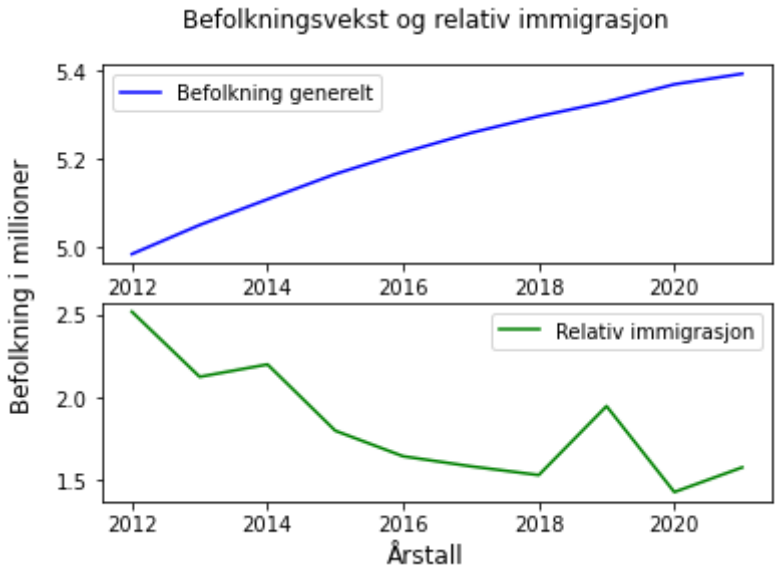


g) ekstrapoeng. Kan plotte de samme tallene (dvs "rel_immigration" og "befolkning" sammen med år) i to figurer ved siden av hverandre ved bruk av "fig, (ax1, ax2) = plt.subplots(1, 2)". Gi grafene ulik farge

In [13]:
```
fig, (ax1, ax2) = plt.subplots(2) # Bruker subplots.
# fortsetter å bruke oppslag når vi skal plott figuren.
ax1.plot(a['årstall'],a['befolkning'], label = 'Befolkning generelt', color = 'blu
ax2.plot(a['årstall'],rel_immigration, label = 'Relativ immigrasjon', color = 'gre
```

```
ax1.legend() # Lager legends.
ax2.legend()
fig.suptitle('Befolkningsvekst og relativ immigrasjon') # Lager en tittel og gir al
fig.supxlabel('Årstall')
fig.supylabel('Befolkning i millioner')
```

Out[13]:
```
Text(0.02, 0.5, 'Befolkning i millioner')
```



Oppgave 4 (20 poeng)

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.

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.

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. $renteutgifter_{t} = (x-a*t)*{r/f}$

Siden vi er ute etter den totale kostnaden i svaret, må du summere renteutgiftene over alle terminer, det vil si $\sum_{t=1}^N(x-a*t)*{r/f}$

Hint: 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.

In [14]:
```
# Skriver ned hva de ulike boksavene betyr slik at de er enklere og jobbe med.
# x = lånet
# t = terminer
# r = rente
# a = avdrag
# f = forrentning

# Funksjonen skal vise hvor mye lånet koster deg i renteutgifter etter antall term
```

```
# Bruker np.arrange funksjonen slik at jeg får renteutgiftene ut i en liste.

def renteutgifter(x,a,t,r,f):
    return (x-a*np.arange(t-1))*r/f
```

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

In [15]:
```
forrentning = 12 # Vi vet at lånet skal forrentes hver måned.
lånebeløp = 1000000 # Vi vet også lånebeløpet.

# Siden lånet skal nedbetales i 10, 20 og 30 år er det ulike avdragsbeløp. Lager vi
def avdrag(x,t):
    return x/t

# Definerer en funksjon for å finne antall terminer.
def terminer(forrentning,år):
    return år*forrentning

# Bruker funksjonen for å finne antall terminer.
print(f"Bruker funksjonen for å finne antall terminer ved ulik nedbetalingsplan:")
print(terminer(forrentning,10))
# Gjør det samme med 20 og 30 års nedbetaling:
print(terminer(forrentning,20))
print(terminer(forrentning,30))


# Bruker funksjonen til å finne avdragene.
print("Bruker funksjon for avdrag til å finne avdragsbeløpene:")
print(avdrag(1000000,120)) # Avdrag 10 års nedbetaling
print(avdrag(1000000,240)) # Avdrag 20 års nedbetaling
print(avdrag(1000000,360)) # Avdrag 30 års nedbetaling
```

```
Bruker funksjonen for å finne antall terminer ved ulik nedbetalingsplan:
120
240
360
Bruker funksjon for avdrag til å finne avdragsbeløpene:
8333.333333333334
4166.666666666667
2777.777777777778
```

In [16]:
```
# Nå som vi vet avdrag og antall terminer setter vi inn i funksjonen for renteutgi
utgifter10 = (renteutgifter(1000000,8333,121, 0.03,12)) # Legger inn i funksjonen r
print(f"Et Lån med på 1 million med 3 % rente og 10 års nedbetaling koster totalt:"
print(sum(utgifter10)) # Summerer utgiftene

utgifter20 = (renteutgifter(1000000,4166,241, 0.03,12)) # Legger inn i funksjonen r
print(f"Et lån på 1 million med 3 % rente og 20 års nedbetaling koster totalt:")
print(sum(utgifter20)) # Summerer utgiftene.

utgifter30 = (renteutgifter(1000000,2777,361, 0.03,12)) # Legger inn i funksjonen r
print(f"Et lån på 1 million med 3 % rente og 30 års nedbetaling koster totalt:")
print(sum(utgifter30))  # Summerer utgiftene.
```

```
Et Lån med på 1 million med 3 % rente og 10 års nedbetaling koster totalt:
151255.94999999998
Et lån på 1 million med 3 % rente og 20 års nedbetaling koster totalt:
301297.8
Et lån på 1 million med 3 % rente og 30 års nedbetaling koster totalt:
451375.64999999997
```

c) Vis hva det samme lånet koster som annuitetslån, dvs differansen mellom alle

terminbeløp og lånebeløp.

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

In [17]:
```
# Definerer funksjonen for annuitetslån
def annuitetslån(x,r,f,t):
    return x*((r/f)/(1-(1+(r/f))**(-t)))

# Legger inn verdier for å finne terminbeløp ved 10 årsnedbetaling.
# Vi vet allerede antall terminer fra forrige oppgave.
print(f"Anniutetslån med 10år/120terminer koster {annuitetslån(1000000,0.03,12,120
# Finner terminbeløp og definerer dette:
terminbeløp = 9656

# Lager en funksjon for kostand av annuitetslånet:
def kostnad_annuitetslån(terminbeløp,terminer,lånebeløp):
    return (terminbeløp*terminer)-lånebeløp

#Printer ut hva lånet koster totalt:
print("Lånet koster totalt:")
print(kostnad_annuitetslån(terminbeløp,120,lånebeløp))
print("Det vil si at et annuiteslån er dyrere enn et serielån ved 10 årsnedbetaling
```

```
Anniutetslån med 10år/120terminer koster 9656.07446983913 per termin.
Lånet koster totalt:
158720
Det vil si at et annuiteslån er dyrere enn et serielån ved 10 årsnedbetaling.
```

d) 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)

Hint1: 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. </br> **Hint2:** Siden vi er ute etter både rentekostnader og avdrag hver for seg, kan det være lurt å definere en funksjon for hver av dem. </br>

In [18]:
```
from matplotlib import pyplot as plt

# Vi har laget funksjoner for avdrag og antall terminer så vi kan legge svarende de
avdrag = [4166] * 240 # Liste med avdrag ganger det med antall terminer
termin240 = list(range(0,240)) # Lager en liste for alle terminene.

# Vi vet fra tidligere hva avdraget er og i hvor mange terminer lånet skal nedbetal
# Legger dette inn i funksjonen for serielån.
renteutgifter2 = renteutgifter(1000000,4166,241, 0.03, 20)

# Så kan vi plotte serielånet.
plt.stackplot(termin240, avdrag, renteutgifter2) # Bruker stackplot til å illustrer
# Plottet illusterer et serielån, der avdragene er like mens renteutgiftene synker
# Et serielån dyrest i starten, da renteutgiftene er store.
# Serielånet er billigere men mindre forutsigbart enn et annuitetslån.
```

Out[18]: [<matplotlib.collections.PolyCollection at 0x7f3b089c2610>,
<matplotlib.collections.PolyCollection at 0x7f3b089c2a00>]