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"

Før jeg begynner skal jeg laste inn alle pakkene jeg trenger for å løse oppgavene. Dette inkluderer numpy og pyplot fra matplotlib

In [1]:
```
# Laster inn pakker og gir disse forkortelser.
import numpy as np
from matplotlib import pyplot as plt
```

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]:
```
# Definerer de fire (bool, int, float og str) variabler jeg skal bruke i historien

# bool (Sann/Usann).
brødrene_liker_sjokolade = True
søstrene_liker_sjokolade = False

# int (heltall).
antall_brødre = 3
antall_søstre = 4
alder_på_mor = 42

# float (desimaltall).
kort_rute = 32.3e-1
lang_rute = 1.58e+2
antall_sjokolader = 1.05e+1

# str (tekst).
k_r = "kort rute"
l_r = "lang rute"
slutt = "Snipp snapp snute."

# Lager historien med bruk av variabelene.
print(
    f"Det var en gang {antall_brødre} brødre og {antall_søstre} søstre. \n"
    f"Alle {antall_brødre + antall_søstre} skulle på skogstur i indre Troms. \n"
    f"Før de kunne sette i mars inn i skogen skulle moren deres som var {alder_på_
    f"Når brødrene åpnet sekken så moren at de hadde pakket med seg {antall_sjokol
    f"Hun visste jo at det var {brødrene_liker_sjokolade} at brødrene likte sjokol
    f"De hadde fortsatt {int(antall_sjokolader - (antall_sjokolader - antall_brødr
    f"Så skulle moren sjekke søstrene sine sekker. \n"
    f"Når søstrene åpnet sekken så moren at de hadde de hadde pakket med seg {int(
    f"Hun visste jo at det var {søstrene_liker_sjokolade} at søstrene likte sjokol
    f"Til slutt ga moren ut en {k_r} og en {l_r}. \n"
    f"Disse var henholdsvis {kort_rute} km og {int(lang_rute)} km lang. Den totale
    f"{slutt} \n" )
```

```
Det var en gang 3 brødre og 4 søstre.
Alle 7 skulle på skogstur i indre Troms.
Før de kunne sette i mars inn i skogen skulle moren deres som var 42 år gammel sje
kke om de hadde fått med seg det viktigste.
Når brødrene åpnet sekken så moren at de hadde pakket med seg 3.5 sjokolader hver.
Hun visste jo at det var True at brødrene likte sjokolade men hun tok bort 7.5 sjo
kolader.
De hadde fortsatt 3 sjokolader igjen.
Så skulle moren sjekke søstrene sine sekker.
Når søstrene åpnet sekken så moren at de hadde de hadde pakket med seg 0 sjokolade
r.
Hun visste jo at det var False at søstrene likte sjokolade.
Til slutt ga moren ut en kort rute og en lang rute.
Disse var henholdsvis 3.23 km og 158 km lang. Den totale lengden ble 161.23 km.
Snipp snapp snute.
```

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 etterspørsels funksjonen.
def etterspørsel(x):
    return 3.0e+6/(120+x)

# Definerer tilbuds funksjonen.
def tilbud(x):
    return (x**2)*(10/250)+5000
```

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

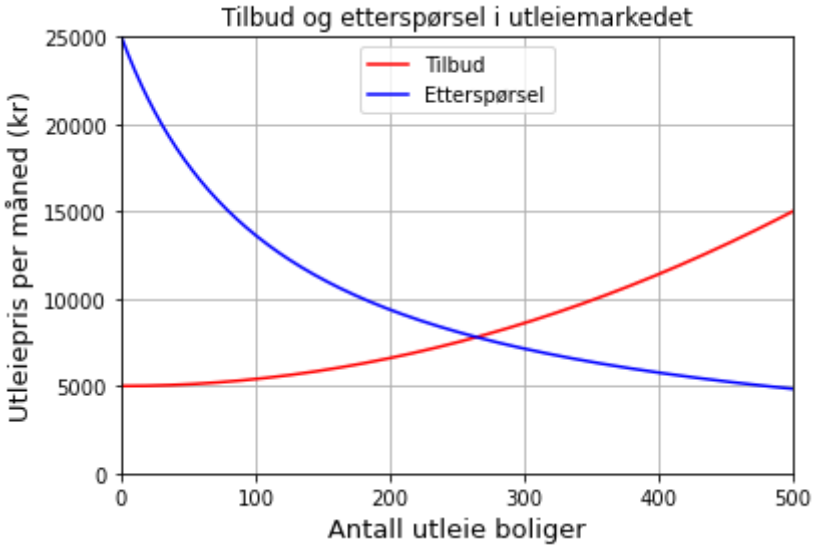
In [4]:
```
# Viser at etterspørsel og tilbud er fallende og stigende med bruk av regning.

# Etterspørsel, x = 1, 10 og 100.
print("\033[1m" + "Etterspørsel:" + "\033[0m")
print(round(3.0e+6/(120+1),2))
print(round(3.0e+6/(120+10),2))
print(round(etterspørsel(100),2))

# Lager ett tomrom i "output" så det blir penere.
print()

# Tilbud, x = 1, 10 og 100.
print(("\033[1m" + "Tilbud:" + "\033[0m"))
print(round((1**2)*(10/250),2))
print(round((10**2)*(10/250),2))
print(round(tilbud(100),2))
```

```
Etterspørsel:
24793.39
23076.92
13636.36

Tilbud:
0.04
4.0
5400.0
```

In [5]:
```
# Viser at etterspørsel og tilbud er fallende og stigende med bruk av matplotlib.

# Tegner 500 punkter mellom intervallen 0 til 500.
x = np.linspace(0,500,500)

# Definerer grafen til tilbud og etterspørsel.
plt.plot(x, tilbud(x), label = "Tilbud", color = "red")
plt.plot(x, etterspørsel(x), label = "Etterspørsel", color = "blue")

# Gir figuren og aksene navn.
plt.title("Tilbud og etterspørsel i utleiemarkedet")
plt.ylabel("Utleiepris per måned (kr)", fontsize = 13)
plt.xlabel("Antall utleie boliger", fontsize = 13)

# Lager en grid i figuren.
plt.grid()
```

```
# Definerer hvilke verdier jeg ønsker å se på y-aksen og x-aksen.
plt.ylim(0,25000)
plt.xlim(0,500)

# Legger til "legend".
plt.legend(loc = "upper center", frameon = True)
```

Out[5]:
```
<matplotlib.legend.Legend at 0x7fdfc0764b50>
```

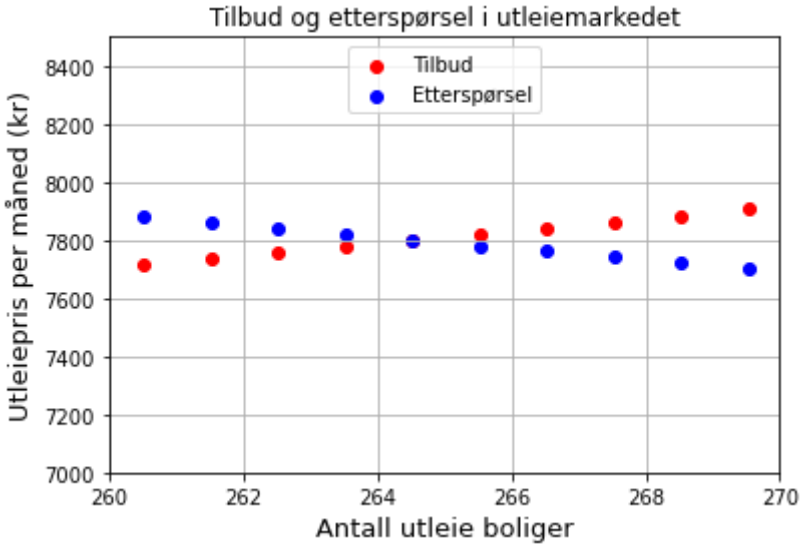


c) Kommenter funksjonene og likevekten. Vis gjerne figurativt hvor likevekten er ved bruk av scatter

Funksjonene viser utleiepris per måned i kroner på y-aksen og hvor mange boliger som er til utleie på x-aksen. Etterspørselfunksjonen er synkende fordi lavere utleiepris gir flere leietakere. Tilbudsfunksjonen er økende fordi høyere utleiepris gir selskaper større grunn til å drive med utleie.

In [6]:
```
# Tegner opp samme plott bortsettfra at jeg lager det med scatter (punkter) og "zoo

# Tegner 500 punkter mellom intervallen 0 til 500.
x = np.linspace(0,500,500)

# Definerer grafen til tilbud og etterspørsel med bruk av scatter (punkter).
plt.scatter(x, tilbud(x), label = "Tilbud", color = "red")
plt.scatter(x, etterspørsel(x), label = "Etterspørsel", color = "blue")

# Gir figuren og aksene navn.
plt.title("Tilbud og etterspørsel i utleiemarkedet")
plt.ylabel("Utleiepris per måned (kr)", fontsize = 13)
plt.xlabel("Antall utleie boliger", fontsize = 13)

# Lager en grid i figuren.
plt.grid()

# Definerer hvilke verdier jeg ønsker å se på y-aksen og x-aksen, "zoomer" inn.
plt.ylim(7000,8500)
plt.xlim(260,270)

# Legger til "legend".
plt.legend(loc = "upper center", frameon = True)
```

Out[6]:
```
<matplotlib.legend.Legend at 0x7fdfb8613700>
```



På figuren over kan vi se likevekten i funksjonene. Likevekt = ca. 264,5

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]:
```
# Lager lister med data hentet fra SSB.
år            = ["år", 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021]
død           = ["Døde i alt", 41992, 41282, 40394, 40727, 40726, 40774, 40840, 40
utflyttinger  = ["Utflyttinger", 31227, 35716, 31875, 37474, 40724, 36843, 34382,
innflyttinger = ["Innflyttinger", 78570, 75789, 70030, 67276, 66800, 58192, 52485,
befolkning    = ["Befolkning 1.januar", 4985870, 5051275, 5109056, 5165802, 521398

# Legger listene jeg laget inn i en egen liste.
ssb           = [år, befolkning, død, innflyttinger, utflyttinger]
```

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

In [8]:
```
# Konverterer "ssb" listen til en numpy matrise (array).
ssb_np = np.array(ssb)
```

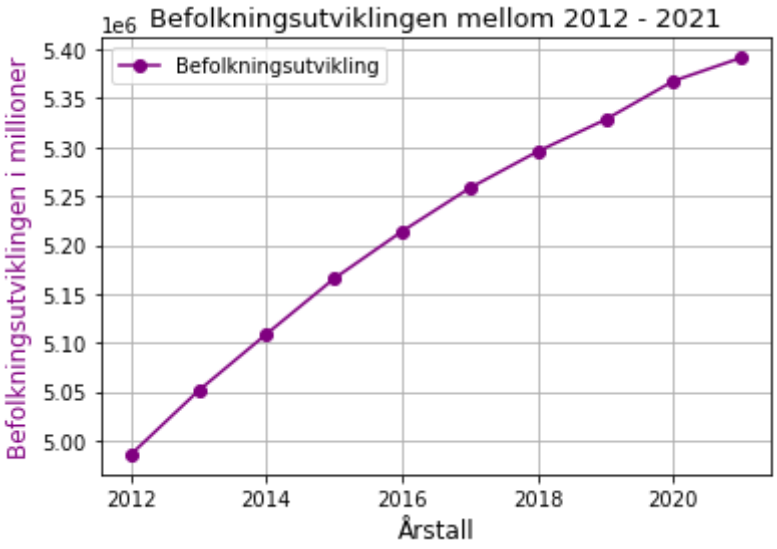
c) Putt alle tallene inn i en egen matrise og konverter disse til int

In [9]:
```
# Legger alle tallene fra "ssb_np" i en egen matrise (array) og konverterer disse
ssb_tall = np.array(ssb_np[:,1:], dtype = int)
```

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]:
```
# Lager en figur som viser befolkningsutviklingen grafiskt mellom periode 2012 til
fig, ax = plt.subplots()
ax.plot(ssb_tall[0,:], ssb_tall[1,:],  label = "Befolkningsutvikling",    color = '
plt.title("Befolkningsutviklingen mellom 2012 - 2021", fontsize = 13)
ax.set_ylabel("Befolkningsutviklingen i millioner", color = "purple", fontsize = 1
ax.set_xlabel("Årstall", fontsize = 12)
plt.grid()
ax.legend(loc = "upper left", frameon = True)
```

Out[10]:
```
<matplotlib.legend.Legend at 0x7fdfb864d2b0>
```



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

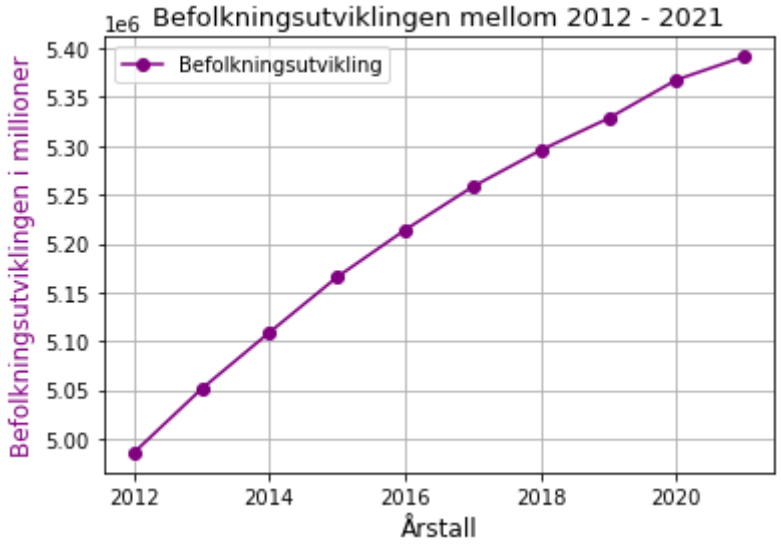
In [11]:
```
# Lager ett oppslag (dict) som inneholder.
ssb_dict = dict()
ssb_dict["År"]            = ssb_tall[0,:]
ssb_dict["Død"]           = ssb_tall[2,:]
ssb_dict["Befolkning"]    = ssb_tall[1,:]
ssb_dict["Utflyttinger"]  = ssb_tall[4,:]
ssb_dict["Innflyttinger"] = ssb_tall[3,:]
```

Fordelere med oppslag

Fordelen med å bruke oppslag er at det gjør koden mer lesbar. Istedefor å bruke f.eks "ssb_tall[0,:]" kan vi bruke "ssb_dict["år"].
Dette gjør det også enklere/raskere å hente, redigere og legge til data siden man jobber med ord (keys) framfor tall.

In [12]:
```
# Lager samme figur som i oppgave d bare at jeg bruker variablene fra oppslaget.
fig,ax=plt.subplots()
plt.title("Befolkningsutviklingen mellom 2012 - 2021", fontsize = 13)
plt.grid()
ax.set_ylabel("Befolkningsutviklingen i millioner", color = "purple", fontsize = 1
ax.set_xlabel("Årstall", fontsize = 12)
ax.plot(ssb_dict["År"], ssb_dict["Befolkning"],  label = "Befolkningsutvikling",
ax.legend(loc = "upper left", frameon = True)
```

Out[12]:
```
<matplotlib.legend.Legend at 0x7fdfb85ca130>
```

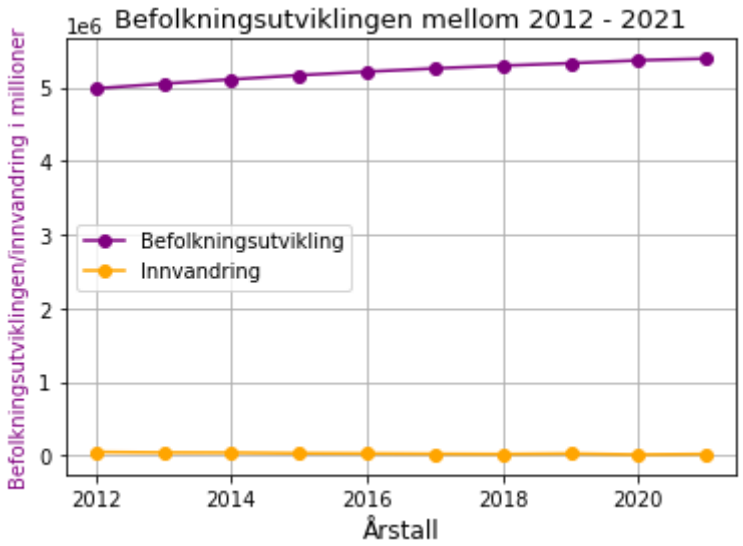


f) Hva er den relative befolkningstilveksten utenom fødsler (dvs. innvandring/utvandring)? Definer en ny array og legg den til i oppslaget du laget i oppgaven tidligere. Kall den "rel_immigration". Plot denne sammen med grafen du laget i (d).

In [13]:
```
# Definerer en ny variabel i oppslaget.
ssb_dict["rel_immogration"] = ssb_dict["Innflyttinger"] - ssb_dict["Utflyttinger"]
```

In [14]:
```
# Leger en figur som viser befolkningsutviklingen og invandringen.
fig,ax=plt.subplots()
plt.title("Befolkningsutviklingen mellom 2012 - 2021", fontsize = 13)
plt.grid()
ax.set_ylabel("Befolkningsutviklingen/innvandring i millioner", color = "purple", 
ax.set_xlabel("Årstall", fontsize = 12)
ax.plot(ssb_dict["År"], ssb_dict["Befolkning"],        label = "Befolkningsutvikli
ax.plot(ssb_dict["År"], ssb_dict["rel_immogration"],   label = "Innvandring",
ax.legend(loc = "center left", frameon = True)
```

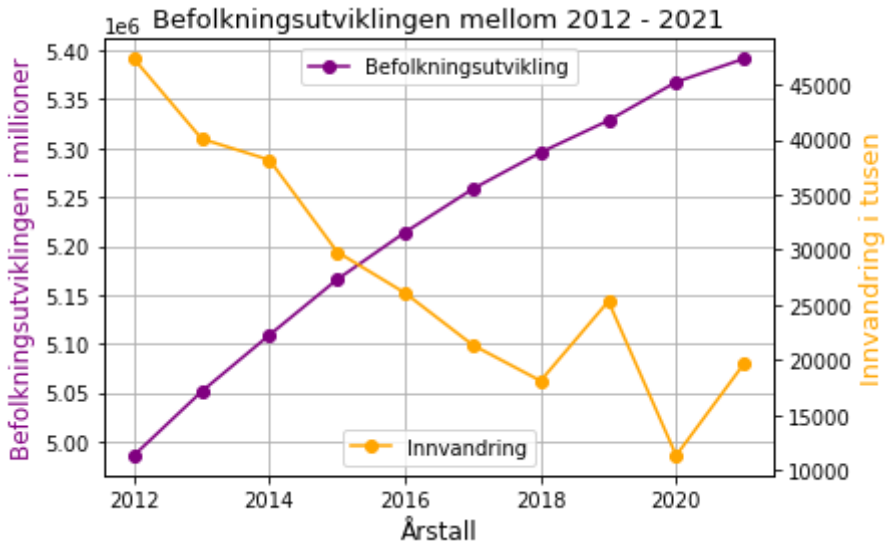
Out[14]:
```
<matplotlib.legend.Legend at 0x7fdfb84e1250>
```



In [15]:
```
# Vanskelig å få noen reel informasjon ut av grafen over så lager en ny graf med to

# Lager og definerer y-akse(1) som befolkningsutviklingen i millioner.
fig,ax=plt.subplots()
plt.title("Befolkningsutviklingen mellom 2012 - 2021", fontsize = 13)
plt.grid()
ax.set_ylabel("Befolkningsutviklingen i millioner", color = "purple", fontsize = 1
```

```
ax.set_xlabel("Årstall", fontsize = 12)
ax.plot(ssb_dict["År"], ssb_dict["Befolkning"],  label = "Befolkningsutvikling",  c
ax.legend(loc = "upper center", frameon = True)

# Lager og definerer y-akse(2) som innvandringen i tusener.
ax2 = ax.twinx()
ax2.plot(ssb_dict["År"], ssb_dict["rel_immogration"],   label = "Innvandring",   co
ax2.set_ylabel("Innvandring i tusen", color = "orange", fontsize = 13 )
ax2.legend(loc = "lower center", frameon = True)
```

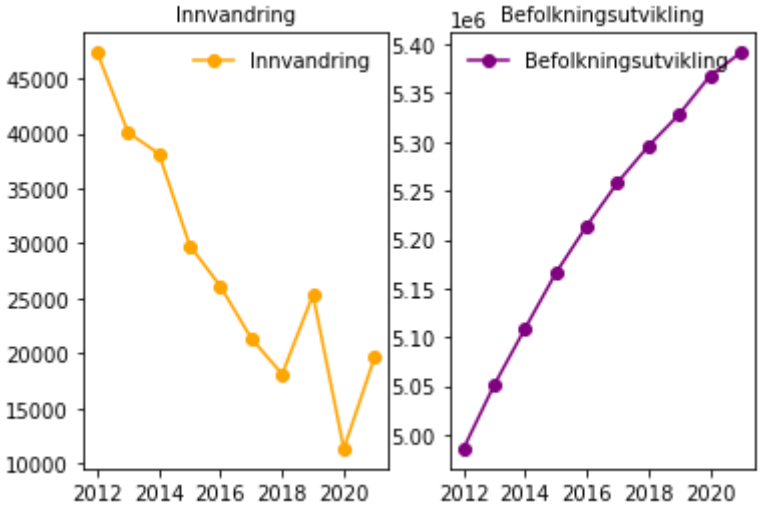
Out[15]:
```
<matplotlib.legend.Legend at 0x7fdfb848efd0>
```



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 [16]:
```
# En alternativ måte å vise to y-akser samtidig er med to figurer ved siden av hver

# Definerer at jeg vil ha to individuelle figurer ved siden av hverandre og mater
fig,(ax1, ax2) = plt.subplots(1, 2)
ax1.plot(ssb_dict["År"], ssb_dict["rel_immogration"], label = "Innvandring", color
ax1.set_title("Innvandring", fontsize = 10)
ax1.legend(loc = "upper right", frameon = False)

# Mater informasjon inn i figur 2.
ax2.plot(ssb_dict["År"], ssb_dict["Befolkning"], label = "Befolkningsutvikling", co
ax2.set_title("Befolkningsutvikling", fontsize = 10)
ax2.legend(loc = "upper left", frameon = False)
```

Out[16]:
```
<matplotlib.legend.Legend at 0x7fdfb84e1dc0>
```



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 [17]:
```
# Definerer funksjonen "S_lån" som har tre inputs: lån, rente, år.
def S_lån(lån,rente,år):
    terminer = 12
    måneder = år * terminer
    avdrag = lån / måneder
    rentesats = rente / 100

    # Legger alle de aktuelle månedene inn i en egen matrise.
    måneder_arange = np.arange(måneder)

    # Bruker formelen i oppgaven til å først regne ut renteutgiften i hver måned fø
    renteutgift = (lån - avdrag * måneder_arange) * (rentesats / terminer)
    total_renteutgift = np.sum(renteutgift)
    return total_renteutgift
```

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 [18]:
```
# Printer ut en "str" med svar på oppgaven i avrunede "float".
print(f"Ett serielån på 1.000.000 med 3% rente over 10 år vil koste deg {round(S_l
print(f"Ett serielån på 1.000.000 med 3% rente over 20 år vil koste deg {round(S_l
print(f"Ett serielån på 1.000.000 med 3% rente over 30 år vil koste def {round(S_l
```

```
Ett serielån på 1.000.000 med 3% rente over 10 år vil koste deg 151250 kr.
Ett serielån på 1.000.000 med 3% rente over 20 år vil koste deg 301250 kr.
Ett serielån på 1.000.000 med 3% rente over 30 år vil koste def 451250 kr.
```

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 [19]:
```
# Definerer funksjonen "A_lån" som har tre inputs: lån, rente, år.
def A_lån(lån,rente,år):
    terminer = 12
    måneder = år * terminer
    rentesats = rente / 100

    # Definerer terminbeløpet.
    terminbeløp = lån * ((rentesats / terminer) / (1-(1+(rentesats / terminer))**-
    # Summerer alle terminbeløpene og trekker fra lånet for å finne renteutgifter.
    terminbeløp_total = terminbeløp * måneder
    renteutgifter = terminbeløp_total - lån
    return renteutgifter
```

In [20]:
```
# Printer ut en "str" med svar på oppgave i avrunede "float".
print(f"Ett annuitetslån på 1.000.000 med 3% rente over 10 år vil koste deg {round
print(f"Ett annuitetslån på 1.000.000 med 3% rente over 20 år vil koste deg {round
print(f"Ett annuitetslån på 1.000.000 med 3% rente over 30 år vil koste deg {round
```

```
Ett annuitetslån på 1.000.000 med 3% rente over 10 år vil koste deg 158729 kr.
Ett annuitetslån på 1.000.000 med 3% rente over 20 år vil koste deg 331034 kr.
Ett annuitetslån på 1.000.000 med 3% rente over 30 år vil koste deg 517775 kr.
```

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)

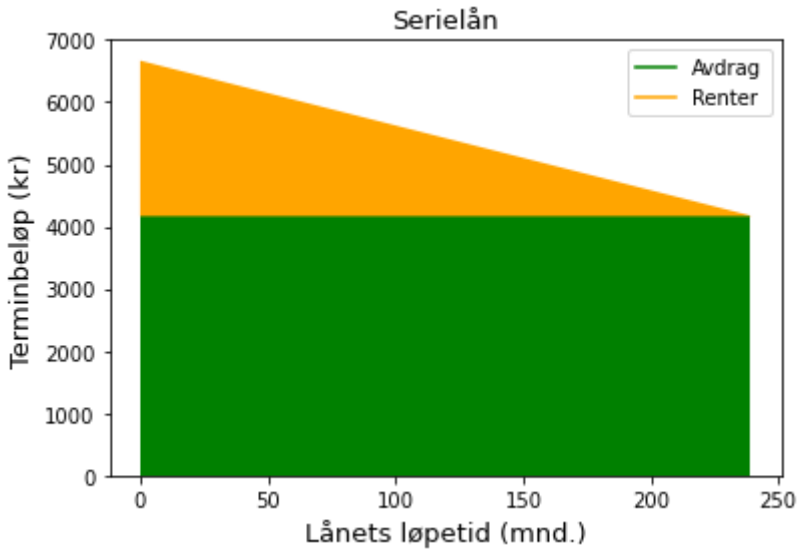
Hint1: Siden avdragene er like for alle måneder, kan det være lurt å definere det månedlige avdraget som en liste og gange det med antall perioder. </br> **Hint2:** Siden vi er ute etter både rentekostnader og avdrag hver for seg, kan det være lurt å definere en funksjon for hver av dem. </br>

In [21]:
```
# Definerer funksjonen "avdrag" med to inputs: lån og år.
def avdrag(lån, år):
    terminer = 12
    måneder = år * terminer
    avdrag = lån / måneder
```

```
    return avdrag

# Definerer funksjonen "rente" med tre inputs: lån, rente og år.
def rente(lån, rente, år):
    terminer = 12
    måneder = år * terminer
    rentesats = rente / 100
    avdrag = lån / måneder
    måneder_arange = np.arange(måneder)
    renteutgifter = (lån - avdrag * måneder_arange) * (rentesats / terminer)
    return renteutgifter

# Lager en liste som inneholder det samme avdraget 240 (antall terminer) ganger.
avdrag_liste = [avdrag(1000000,20)]
avdrag_liste_240 = avdrag_liste * 240

# Lager en variabel som inneholder alle tall fra 0 til 240 (antall terminer).
terminer = range(240)

# Definerer to åpene plot slik at jeg kan gi de en "legend" seinere.
plt.plot([],[],color = "green", label = "Avdrag")
plt.plot([],[],color = "orange", label = "Renter")

# Bruker stackplot til å tegne figuren, dette blir puttet inn i de åpne plottene ov
plt.stackplot(terminer, avdrag_liste_240, rente(1000000, 3, 20), colors = ["green",

# Legger til "legend".
plt.legend()

# Gir figuren og aksene navn.
plt.title("Serielån", fontsize = 13)
plt.xlabel("Lånets løpetid (mnd.)", fontsize = 13)
plt.ylabel("Terminbeløp (kr)", fontsize = 13)
```

Out[21]:

```
Text(0, 0.5, 'Terminbeløp (kr)')
```



Kildeliste: SSB, (2022), Endring i befolkningen i løpet av året 1735 - 2022, SSB, hentet 10.10.2022 https://www.ssb.no/statbank/table/05803/tableViewLayout1/ CMDLINE, (25.10.2019), How to Make a Plot with Two Different Y-axis in Python with Matplotlib, CMDLINE, hentet 13.10.2022 https://cmdlinetips.com/2019/10/how-to-make-a-plot-with-two-different-y-axis-in-python-with-matplotlib/ Sparebank1, Hva er forskjellen på annuitetslån og serielån?, Sparebank1, hentet 14.10.2022

https://www.sparebank1.no/nb/bank/privat/kundeservice/lan/Hva-er-forskjellen-pa-annuitetslan-og-serielan.html