Bonusopppgaver

I denne oppgavesamlingen finner du et knippe med bonusoppgaver for dag 2 dersom du vil ha litt mer å bryne deg på. God koding!

Løkker

Oppgave 1 *Kjøpe telefon på kreditt*

Hallgeir har veldig lyst på en ny telefon som koster 2999 kroner. Problemet er bare at han har brukt opp sparepengene sine på andre ting. For å få kjøpt telefonen skaffer Hallgeir et kredittkort som har 30% rente. Etter å ha kjøpt telefonen går det tre år før Hallgeir betaler tilbake kreditten. I denne oppgaven skal vi undersøke hvor mye Hallgeir blir nødt til å betale da.

a) Opprett variablene `renter`, `originalpris` og `antall_år` og gi med verdiene `30`, `2999` og `3`

b) Regn ut vekstfaktoren til renta og lagre den i en variabel `rentevekstfaktor`

c) Opprett en variabel, `lån`. Denne variabelen skal holde orden på hvor stort lånet til Hallgeir er. Til å begynne med er lånet like stort som prisen på telefonen. Sett altså `lån`-variabelen til å ha verdien `2999`

d) Bruk en `for`-løkke til å simulere hvordan lånet vokser for hvert år.
Hint: for hvert år skal lånet ganges med vekstfaktoren du regnet ut i oppgave b)

e) Oppdater programmet ditt til å skrive ut størrelsen på lånet for hvert år. Hvor mye skylder Hallgeir etter tre år?

f) Hvor mye ekstra kostet telefonen i forhold til originalprisen?

g) Gå inn på `https://kredittkort.com/` og se hvilket kort som kommer øverst og noter deg renta. Endre renta i programmet ditt til å matche denne renta. Gå inn på `https://www.prisjakt.no/category.php?k=103` og se hvilken telefon som er mest populær. Endre orginalprisen i programmet ditt til å matche prisen til den telefonen. Kjør programmet nå. Hva ville denne telefonen og dette kredittkortet kostet Hallgeir?

Løsning oppgave 1 *Kjøpe telefon på kreditt*

a)

```
renter = 30
originalpris = 2999
antall_år = 3
```

b)

```
rentevekstfaktor = 1 + renter/100
```

c)

```
lån = originalpris
```

d)

```
for år in range(antall_år):
    lån *= rentevekstfaktor

print(f"Hallgeir skylder {lån:.2f} kroner etter {
   antall_år} år")
```

```
Hallgeir skylder 6588.80 kroner etter 3 år
```

e) Det kostet 3589.80 kroner mer enn originalprisen å kjøpe telefonen på kreditt.

f) Det kredittkortet med lavest rente har en rente på $23,1\%$

g) Den vanligste telefonen er en iPhone 11 64GB og koster 7990 kroner. Om vi bruker disse tallene i koden får vi at telefonen koster $14904,62$ kroner, det er nesten dobbelt så mye som originalprisen!

Oppgave 2 *Hangman*

I denne oppgaven skal du lage Python kode for spillet "hangman". Deloppgavene leder deg trinn for trinn gjennom hvordan du kan bygge opp funksjonaliteten til du har et komplett hangman spill.

Hangman trenger et maks antall gjett og et hemmelig ord.

a) lag to variabler, `gjett` som har verdien verdien `10` og `hemmelig_ord` som har et hemmelig ord som verdi, for eksempel `'hemmelig'`.

Kanskje det viktigste biten av et spill med hangman er at spilleren gjetter en bokstav. Dersom bokstaven ikke er i det hemmelige ordet bruker spilleren opp et av sine gjett. La oss begynne med å implementere dette. Først trenger vi en løkke som fortsetter helt til antall gjett er brukt opp.

b) Lag en `while`-løkke som fortsetter så lenge `gjett`>0. **OBS:** Hvis du kjører koden uten å oppdatere `gjett`, vil du få en uendelig løkke

c) Inne i løkka skal du spørre brukeren om å gjette en bokstav og lagre gjettet i en variabel, `bokstavgjett`

d) Bruk en betingelse for å sjekke om `bokstavgjett` er i det hemmelige ordet. Hvis bokstaven er riktig skriv ut beskjed til brukeren om at de gjettet en riktig bokstav. Hvis ikke, skal variabelen `gjett` reduseres med en.

e) Skriv ut hvor mange `gjett` brukeren har igjen for hver runde i løkka

Nå har vi nesten et fungerende hang-man spill, men det mangler noen ting. Vi vil jo for eksempel at spilleren skal kunne se hvilke bokstaver de har fått til. De tre neste deloppgavene forklarer hvordan:

f) Lag en variabel, `gjettede_bokstaver`, i toppen av programmet, som inneholder en tom streng, `''`. Hver gang spilleren gjetter en bokstav skal bokstaven legges til i denne variabelen med `gjettede_bokstaver +=bokstavgjett`.

g) I starten av `while`-løkka: Bruk en `for`-løkke til å løkke igjennom hver bokstav i det hemmelige ordet.

h) For hver bokstav i det hemmelige ordet skal du sjekke om den bokstaven er blant de gjettede bokstavene, altså om en bokstav er i `gjettede_bokstaver`. Bokstaver som er gjettet skal skrives ut med `print(bokstav, end='')`. Bokstaver som ikke ennå er gjettet skal skrives ut som en strek med `print('_', end='')` (`end=' '` endrer `print` kommandoen slik at den avslutter med mellomrom istedenfor linjeskift).

i) Test programmet ditt. Finner du noen mangler? Hva skjer dersom du gjetter hele ordet før du har brukt opp dine 10 gjett?

j) Lag en variabel `antall_manglende_bokstaver` som skal holde orden på hvor mange bokstaver som fortsatt ikke er gjettet. `antall_manglende_bokstaver` skal settes til 0 før `for`-løkka som løkker over ordet og økes med 1 hver gang en bokstav i det hemmelige ordet er ukjent.

k) Etter `for`-løkka skal du bruke en betingelse som skjekker om `antall_manglende_bokstaver` er 0 og i såfall skrive ut en vinnerbeskjed og bryte ut av løkka med `break`.

l) **Bonusoppgave:** i hangman-spill er det også vanlig å gradvis tegne en tegning av en "hangman" for hver gang spilleren gjetter feil. Dette kan du for eksempel få til ved å lagre tegninger i en liste og vise frem riktig tegning basert på hvor mange gjett spilleren har igjen. For å lage tegningene kan du basere deg på ascii-tegningen under eller lage din egen.

```
"""

--------
 |/    |
 |    (_)
 |    /|\
 |     |
 |    / \
 |
 |___
 """
```

Løsning oppgave 2 *Hangman*

a)

```
gjett = 10
hemmelig_ord = 'hemmelig'
```

b)

```
while gjett>0:
    pass
```

c)

```
while gjett>0:
    bokstavgjett = input("Gjett en bokstav")
```

d)

```
while gjett>0:
    bokstavgjett = input("Gjett en bokstav")
    if bokstavgjett in hemmelig_ord:
        print("Riktig!")
    else:
        print("Feil")
        gjett -= 1
```

e)

```
while gjett>0:
    print(f"Du har {gjett} gjett igjen")
    bokstavgjett = input("Gjett en bokstav")
    if bokstavgjett in hemmelig_ord:
        print("Riktig!")
    else:
        print("Feil")
        gjett -= 1
```

f)

```
gjett = 10
hemmelig_ord = 'hemmelig'
gjettede_bokstaver = ''
while gjett>0:
    print(f"Du har {gjett} gjett igjen")
    bokstavgjett = input("Gjett en bokstav")
    if bokstavgjett in hemmelig_ord:
        print("Riktig!")
        gjettede_bokstaver += bokstavgjett
    else:
        print("Feil")
        gjett -= 1
```

g)

```
gjett = 10
hemmelig_ord = 'hemmelig'
gjettede_bokstaver = ''
while gjett>0:
    for bokstav in hemmelig_ord:
        if bokstav in gjettede_bokstaver:
            print(bokstav, end=' ')
        else:
            print('_', end=' ')
    print()
    print(f"Du har {gjett} gjett igjen")
    bokstavgjett = input("Gjett en bokstav")
    if bokstavgjett in hemmelig_ord:
        print("Riktig!")
        gjettede_bokstaver += bokstavgjett
    else:
        print("Feil")
        gjett -= 1
```

h) Den viktigste mangelen er at programmet vil fortsette å spørre om nye gjett selv om spilleren har gjettet hele ordet riktig! Det går altå ikke ann å vinne!

i)

```
gjett = 10
hemmelig_ord = 'hemmelig'
gjettede_bokstaver = ''
while gjett>0:
    antall_manglende_bokstaver = 0
    for bokstav in hemmelig_ord:
        if bokstav in gjettede_bokstaver:
            print(bokstav, end=' ')
        else:
            print('_', end=' ')
            antall_manglende_bokstaver += 1
    print()
    if antall_manglende_bokstaver == 0:
        print("Du klarte det!")
        break
    print(f"Du har {gjett} gjett igjen")
    bokstavgjett = input("Gjett en bokstav")
    if bokstavgjett in hemmelig_ord:
        print("Riktig!")
        gjettede_bokstaver += bokstavgjett
    else:
        print("Feil")
        gjett -= 1
```

j)

```
gjett = 10
hemmelig_ord = 'hemmelig'


hangman = [
"""
```

```
  ___
 """
,
"""
  |
  |
  |
  |
  |
  |
  |___
 """
,
"""
________
 |/
 |
 |
 |
 |
 |
 |___
 """
,
"""
________
 |/    |
 |
 |
 |
 |
 |
 |___
 """
,
"""
________
 |/    |
```

```
  |    (_)
  |
  |
  |
  |
  |___
  """
,
"""

________
 |/    |
 |    (_)
 |     |
 |     |
 |
 |
 |___
 """
,
"""

________
 |/    |
 |    (_)
 |    /|
 |     |
 |
 |
 |___
 """
,
"""

________
 |/    |
 |    (_)
 |    /|\
 |     |
 |
 |
 |___
 """
```

```
,
"""

________
 |/    |
 |    (_)
 |    /|\
 |     |
 |    /
 |
 |___
 """
,
"""

________
 |/    |
 |    (_)
 |    /|\
 |     |
 |    / \
 |
 |___
 """]


gjettede_bokstaver = ''
while gjett>0:
    antall_manglende_bokstaver = 0
    for bokstav in hemmelig_ord:
        if bokstav in gjettede_bokstaver:
            print(bokstav, end=' ')
        else:
            print('_', end=' ')
            antall_manglende_bokstaver += 1
    print()
    if antall_manglende_bokstaver == 0:
        print("Du klarte det!")
        break
    print(f"Du har {gjett} gjett igjen")
    print(hangman[10-gjett])
    bokstavgjett = input("Gjett en bokstav")
```

```
133     if bokstavgjett in hemmelig_ord:
134         print("Riktig!")
135         gjettede_bokstaver += bokstavgjett
136     else:
137         print("Feil")
138         gjett -= 1
```

Oppgave 3 *Wokoppskrift*

I denne oppgaven skal vi øve på å bruke en funksjon vi ikke har sett på før, nemlig `choice` fra `random`-biblioteket. Det å finne ut og forstå hva nye funksjoner er en viktig ferdighet for en programmerer. Derfor er det lurt å øve på å slå opp i dokumentasjonen.

Nå skal vi lage et program som kan autogenerere wok-oppskrifter, inspirert av denne nettsiden: `https://www.frenchguycooking.com/stir-fry-generator`. For å lage en god wok kan man kombinere en karbohydratkilde, grønnsaker og en proteinkilde. La oss derfor starte med å lage en liste over mulige karbohydratkilder.

a) Lag en liste, `karbohydratkilder` med minst tre karbohydratkilder, f.eks. ris, eggnuddel og risnuddel.

b) Importer funksjonen `choice` fra `random` biblioteket ved å skrive `from random import choice`.

c) Bruk `help` funksjonen for å lese dokumentasjonen til `choice` ved å skrive `help(choice)` i terminalvinduet. Hva tror du denne funksjonen gjør?

d) Bruk choice-funksjonen for å hente ut en tilfeldig karbohydratkilde fra `karbohydratkilder`-lista. Lagre karbohydratkilden i en variabel du kaller `karbohydratkilde` og skriv denne karbohydratkilden ut til brukeren

e) Kjør programmet mange ganger, får du samme utskrift hver gang? Hvorfor får du denne oppførselen?

f) Lag en liste `grønnsaker` som inneholder minst fem grønnsaker du kunne tenkt deg i en wok. For eksempel brokkoli, gulrot, spinat, vårløk og erter. Lag også en liste `proteinkilder` som inneholder noen proteinkilder, f.eks tofu, kylling, storfe og scampi.

g) Bruk `choice` funksjonen til å hente ut to tilfeldige grønnsaker og en proteinkilde, lagre disse grønnsakene i variablene `grønnsak1`, `grønnsak 2` og `proteinkilde`.

h) Skriv ut en fin beskjed til brukeren som beskriver woken du skal ha til middag. F.eks. `eggnuddel-wok med spinat, gulrot og tofu`

Løsning oppgave 3 *Wokoppskrift*

a)

```
karbohydratkilder = ["ris", "eggenuddel", "
    risnuddel"]
```

b)

```
from random import choice
```

c)

```
help(choice)
```

d)

```
karbohydratkilde = choice(karbohydratkilder)
print(karbohydratkilde)
```

e) Vi får forskjellig resultat siden choice-funksjonen henter ut tilfeldige element fra karbohydrat-lista.

f)

```
grønnsaker = ["brokkoli", "gulrot", "spinat", "vå
    rløk", "erter"]
proteinkilder = ["tofu", "kylling", "storfe", "
    scampi"]
```

g)

```python
grønnsak1 = choice(grønnsaker)
grønnsak2 = choice(grønnsaker)
proteinkilde = choice(proteinkilder)
```

h)

```python
from random import choice

karbohydratkilder = ["ris", "eggenuddel", "risnuddel"]
grønnsaker = ["brokkoli", "gulrot", "spinat", "vårløk", "erter"]
proteinkilder = ["tofu", "kylling", "storfe", "scampi"]


karbohydratkilde = choice(karbohydratkilder)
grønnsak1 = choice(grønnsaker)
grønnsak2 = choice(grønnsaker)
proteinkilde = choice(proteinkilder)

print(f"{karbohydratkilde}-wok med {grønnsak1}, {grønnsak2} og {proteinkilde}")
```

Oppgave 4 *Slå opp i dokumentasjonen*

I denne oppgaven skal du teste ut en smakebit på hvordan man kan lage sin helt egne funksjon i Python. Hvis du vil vite mer om dette kan du lese kapittel 8 i kompendiet (https://github.com/kodeskolen/tekna_agder_h20_1/blob/main/kompendium.pdf) Fra før av har du kanskje brukt ferdiglagde funksjoner som print, input og float, men man kan også lage egne funksjoner.

a) Lim inn koden under inn i et Python script og kjør den. Hva får du ut i terminalen?

```python
def hils_på(navn)
```

```
2      """funksjon som tar inn et navn og skriver ut en
           hilsen til det navnet"""
3      print(f'Hei på deg, {navn}!')
4
5  hils_på("Sofie")
```

b) Endre `'Sofie'` til `'Adla'` i `hils_på('Sofie')`. Hva får du ut til terminalen nå?

c) Basert på disse forsøkene, hva tror du funksjonen gjør?

d) Hva tror du er poenget med teksten inne i `"""` på linje 2?

e) Bruk `help(hils_på)` til å skrive ut dokumentasjonen til funksjonen

f) Modifiser koden for å lage en funksjon `si_farvel(navn)` som tar inn et navn og skriver ut en farvelbeskjed (f.eks `f'Ha en fin dag, navn!'`).

Løsning oppgave 4 *Slå opp i dokumentasjonen*

a) Vi får ut `Hei på deg, Sofie!`.

b) Vi får ut `Hei på deg, Adla!`.

c) Funksjonen tar inn et navn og skriver ut hilsenen "Hei på deg, NAVN", hvor NAVN byttes ut med input-navnet.

d) `"""`-strengen på linje 2 brukes for å lage dokumentasjonen til funksjonen. En slik kommentarstreng kalles gjerne for en dokumentasjonsstreng, eller en docstring.

e) Når vi bruker `help`-funksjonen på en funksjon så printes dokumentasjonsstrengen til funksjonen ut.

f)

```
1  def si_farvel(navn):
2      """Funksjon som tar inn et navn og skriver ut
           en farvell-beskjed til det navnet."""
```

```
3       print(f'Ha en fin dag, {navn}')
```

Plotting

Oppgave 5 *Plotte lineære funksjoner*

a) Lag et program som plotter funksjonen

$$f(x) = ax + b$$

for to gitte verdier `a` og `b`.

b) Legg programmet ditt over inn i en funksjon som tar inn to argumenter `a, b` og plotter den generelle funksjonen gitt disse verdiene.

Kall funksjonen flere ganger med ulike parametere, og legg `show()` utenom funksjonen (helt til slutt i programmet ditt). Slik får vi frem flere plott i samme vindu. Prøv deg frem med ulike parametere for `a` og `b`.

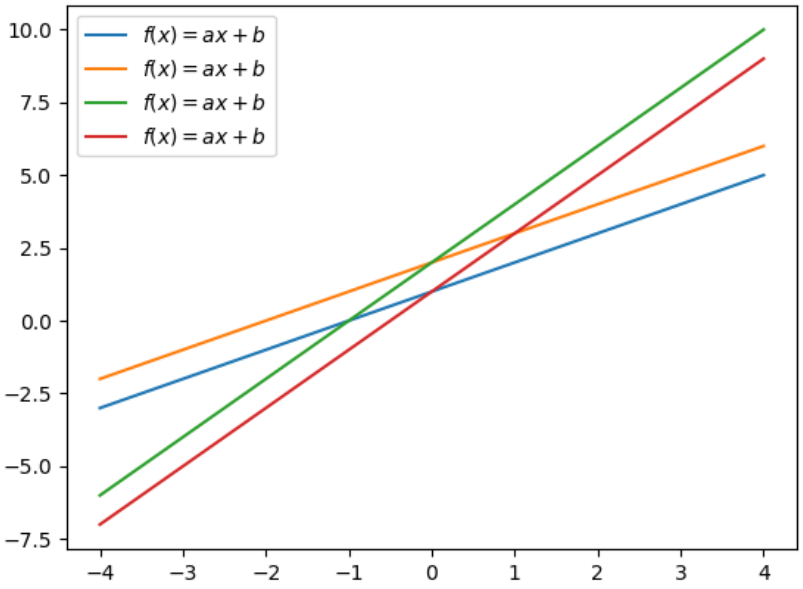
Løsning oppgave 5 *Plotte lineære funksjoner*

Kode:

```
1  from matplotlib.pyplot import *
2
3  def plott_lineær_funksjon(a, b):
4      xverdier = []
5      yverdier = []
6
7      for x in range(-4, 5):
8          xverdier.append(x)
9          yverdier.append(a*x + b)
10
11     plot(xverdier, yverdier, label=rf"$f(x) = ax + b$"
           )
12
13 plott_lineær_funksjon(1, 1)
```

```
14 plott_lineær_funksjon(1, 2)
15 plott_lineær_funksjon(2, 2)
16 plott_lineær_funksjon(2, 1)
17
18
19 legend()
20 show()
```

Plott:



Oppgave 6 *Plotte kvadrattall og kubikktall*

Lag et program som plotter den lineære funksjonen

$$f(x) = x$$

den kvadratiske funksjonen

$$f(x) = x^2$$

og den kubiske funksjonen

$$f(x) = x^3$$

i samme plott.

Legg programmet over inn i en funksjon som tar to argumenter, `xmin` og `xmax` som angir hvilke x-verdier vi plotter funksjonene over. Prøv deg frem med ulike grenser. Hva har valget av x-verdier å si?

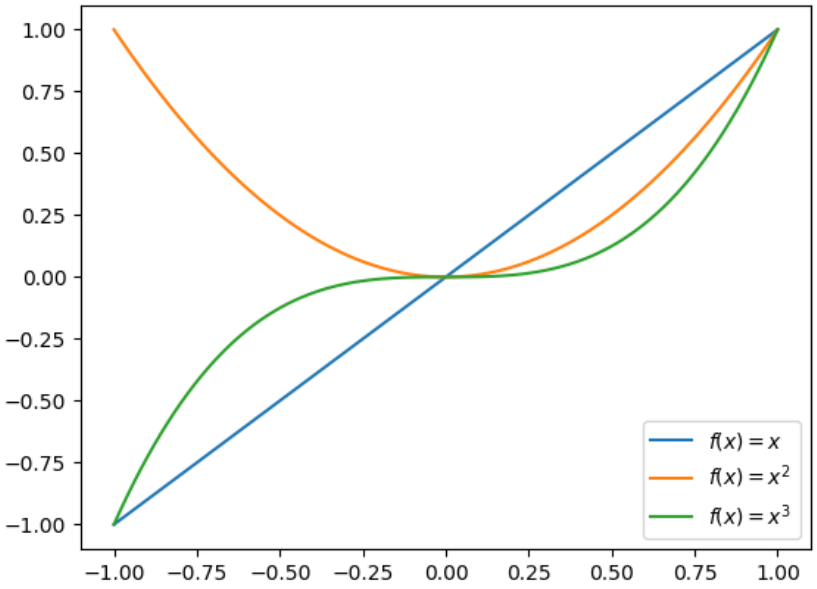
Løsning oppgave 6 *Plotte kvadrattall og kubikktall*

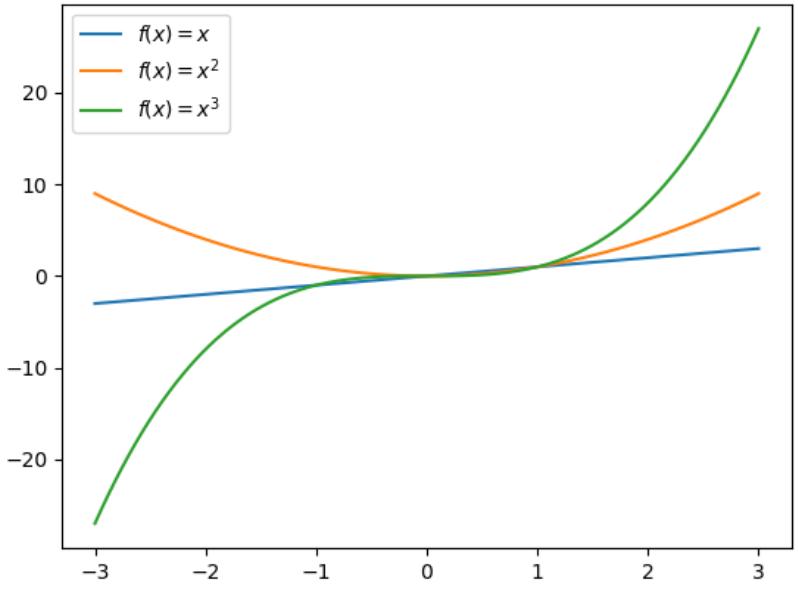
Kode:

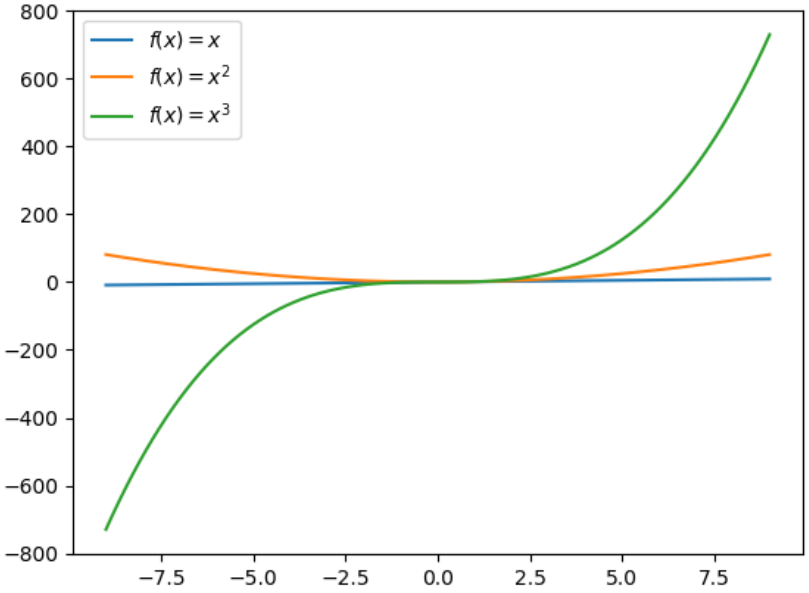
```
from matplotlib.pyplot import *

def sammenlign(xmin, xmax):

    xverdier = []
    kvadratisk = []
    kubisk = []

    for x in arange(xmin, xmax, 0.01):
        xverdier.append(x)
        kvadratisk.append(x**2)
        kubisk.append(x**3)

    plot(xverdier, lineær, label='$f(x) = x$')
    plot(xverdier, kvadratisk, label=r'$f(x) = x^2$')
    plot(xverdier, kubisk, label=r'$f(x) = x^3$')
    legend()

    show()

    #eventuelt lagre + slett plott for å gjøre klar
       for neste:
    #savefig(f"lineær_kvadratisk_kubisk_{xmin}_{xmax}.
       png")
    #clf()

sammenlign(-1, 1)
sammenlign(-3, 3)
```

```
27 sammenlign(-9, 9)
```

Plott:







Ulike valg av x-verdier fremhever ulike ting. Mellom -1 og 1 kan man se

hvordan verdiene endrer seg for veldig små verdier. For større intervall får man bedre forståelse for hvor mye mer kvadratiske og kubiske funksjoner øker.

Oppgave 7 *Spare med BSU-konto*

Vigdis setter inn 1000 kroner på BSU med 3,5 % rente.

a) Opprett variabel `pengemengde = 1000` og en variabel `antall_år = 0`.

b) Opprett en `while`-løkke som løkker så lenge pengemengde ikke har doblet. For hver runde skal du øke pengemengde med 3,5 % og `antall_år` med 1.

c) Skriv ut hvor mange år det tar å doble pengemengden og hvor mye penger Vigdis har da.

d) For å plotte pengeutviklingen trenger vi en liste over pengemengden for hvert år og en liste over år. Oppdater programmet ditt til å opprette en tom liste `pengemende_liste` og en tom liste `år_liste` for løkka.

e) Oppdater programmet til å legge til verdien til `pengemengde` i slutten av `pengemengde_liste` med `append` for hver runde i løkka.

f) Oppdater programmet til å legge til verdien til `antall_år` i slutten av `år_liste` med `append` for hver runde i løkka.

g) Plot pengeutviklingen mot årene med `plot`.

h) Legg på merkelapper på x og y aksen og en tittel til plottet.

i) Endre renta til 4,1 %. Hvordan endrer plottet seg?

Løsning oppgave 7 *Spare med BSU-konto*

a)–c)

```
pengemengde = 1000
antall_år = 0


while pengemengde < 2*1000:
    pengemengde *= 1.035
    antall_år += 1

print(f"Etter {antall_år} har du {pengemengde:.2f}
    kr på konto")
```

Pass på at du i `while`-løkka alltid sammenligner med 2 ganger **start-summen** og at den ikke oppdateres i motsetning til `pengemengde` som oppdateres hver gang løkka kjøres. På den siste linjen printer vi ut hva pengemengden er og hvor mange år som er gått.

d)

```

pengemengde_liste = []
år_liste = []

pengemengde = 1000
antall_år = 0
rente = 0.035
```

e) - f)

```

pengemengde_liste = []
år_liste = []

pengemengde = 1000
antall_år = 0
rente = 0.035


while pengemengde < 2*1000:
```

```
    pengemengde_liste.append(pengemengde)
    år_liste.append(antall_år)
    pengemengde *= (1 + rente)
    antall_år += 1
```

g)

```
from matplotlib.pyplot import *

pengemengde_liste = []
år_liste = []

pengemengde = 1000
antall_år = 0
rente = 0.035


while pengemengde < 2*1000:
    pengemengde_liste.append(pengemengde)
    år_liste.append(antall_år)
    pengemengde *= (1 + rente)
    antall_år += 1


print(f"Etter {antall_år} år har du {pengemengde:.2f} kr på konto")

plot(år_liste, pengemengde_liste)
show()
```

Her er de viktigste endringene i første og siste linjer i programmet.

h)

```
from matplotlib.pyplot import *

pengemengde_liste = []
år_liste = []

pengemengde = 1000
antall_år = 0
rente = 0.035

```

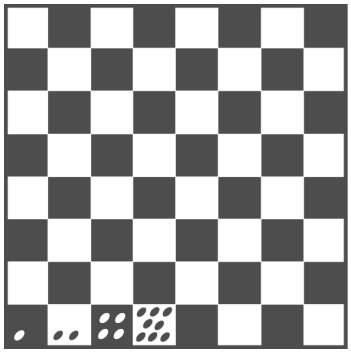
```
10
11  while pengemengde < 2*1000:
12      pengemengde_liste.append(pengemengde)
13      år_liste.append(antall_år)
14      pengemengde *= (1 + rente)
15      antall_år += 1
16
17
18  print(f"Etter {antall_år} år har du {pengemengde:.
        2f} kr på konto")
19
20  plot(år_liste, pengemengde_liste)
21  xlabel("Antall år på konto")
22  ylabel("Penger i kroner")
23  title("Penger på BSU - konto")
24  show()
```

i) Endre rentevariablen til 0.041 og se hvordan endringen ser ut.

Oppgave 8 *Sjakk og riskornvekst*

Det finnes en nesten tusen år gammel legende om oppfinnelsen av sjakk som går slik: En veldig smart mann finner opp spillet sjakk og viser det til keiseren sin. Keiseren blir så imponert at han erklærer at oppfinneren kan velge sin egen belønning. Oppfinneren svarer at han er en ydmyk mann og ønsker kun ris. Og siden det er sjakk han blir belønnet for vil han ha et riskorn for den første ruta i brettet, to for den andre, fire for den tredje og så videre. Han ønsker altså at mengden ris skal dobles for hver rute i brettet.

Kongen synes dette er en beskjeden belønning og aksepterer den på stedet. Men når han forteller det til sin kasserer får han beskjed om at hele keiserdømmet vil gå konkurs!



a) Bruk en for-løkke til å finne ut hvor mange riskorn oppfinneren ba om. Er du enig med kassererens fortvilelse?

b) Det er omtrent 50 000 riskorn i et kilo med ris. Bruk en `while`-løkke til å finne ut hvor mange ruter man må belønne oppfinneren for for at han skal få minst et kilo med ris.

c) Lag et plot over veksten av riskorn. La x-aksen være antall sjakkruter og y-aksen være antall riskorn. Gjør plottet pent ved å gi navn til aksene med `xlabel(...)`/`ylabel(...)`), legg på en tittel med `title(...)` og et rutenett med `grid()`.

Løsning oppgave 8 *Sjakk og riskornvekst*

a)

```
antall_riskorn = 0

for i in range(64):
    antall_riskorn += 2**i

print(f"Antall riskorn: {antall_riskorn}")
```

b)

```
riskorn1kg = 50000
antall_riskorn = 0
antall_ruter = 0

while antall_riskorn < riskorn1kg:
```

```
    antall_riskorn += 2**antall_ruter
    antall_ruter += 1

print(f"Antall ruter: {antall_ruter}")
```

c)

```
from pylab import *

xer = range(1, 65)
yer = []
antall_riskorn = 0

for i in range(64):
    antall_riskorn += 2**i
    yer.append(antall_riskorn)

plot(xer, yer)
xlabel("Ruter")
ylabel("Antall riskorn")
legend(["Riskorn vs ruter"])
title("Legende")
grid()
show()
```