Flere oppgaver for funksjoner og plotting
Avansert kurs ved Folkeuniversitetet

Funksjoner

Oppgave 1 *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?

```
1 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 1 *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}')
```

Oppgave 2 *Enkle funksjoner*

a) Lag en funksjon `pluss(a,b)` som tar inn to tall a og b og returnerer summen av dem.

b) Lag en funksjon `minus(a,b)` som tar inn to tall a og b og returnerer differansen av dem.

c) Lag en funksjon `kalkulator(operasjon,a,b)` som tar inn operasjon

som enter er ”pluss"” eller ”"minus"” (i form av en tekststring), og to tall, og regner ut resultatet.

Løsning oppgave 2 *Enkle funksjoner*

```
def pluss(a,b):
    return a+b

def minus(a,b):
    return a-b

def kalkulator(operator,a,b):
    if operator == "pluss":
        return a+b
    if operator == "minus":
        return a-b
```

Oppgave 3 *«Vi har ei gammel tante»*

Dette er teksten til sangen «Vi har ei gammel tante»:

Vi har ei gammel tante som heter Monica
Og når a' går på torvet
Vi hermer etter a'!
For sånn svaier hatten, og hatten svaier sånn
Og sånn svaier hatten, og hatten svaier sånn!

Vi har ei gammel tante som heter Monica
Og når a' går på torvet
Vi hermer etter a'!
For sånn svaier fjøra, og fjøra svaier sånn!
Og sånn svaier fjøra, og fjøra svaier sånn!

Vi har ei gammel tante som heter Monica

Og når a' går på torvet
Vi hermer etter a'!
For sånn svinger veska, og veska svinger sånn!
Og sånn svinger veska, og veska svinger sånn!

Vi har ei gammel tante som heter Monica
Og når a' går på torvet
Vi hermer etter a'!
For sånn svinger skjørtet, og skjørtet svinger sånn!
Og sånn svinger skjørtet, og skjørtet svinger sånn!

Vi har ei gammel tante som heter Monica
Og når a' går på torvet
Vi hermer etter a'!
For sånn svinger rumpa, og rumpa svinger sånn!
Og sånn svinger rumpa, og rumpa svinger sånn!

a) Lag en liste med ulike elementer som kan svaie, f.eks. "fjøra".

b) Lag en funksjon som printer ut teksten til ett vers i sangen, der argumentene er første element i listen du lagde i a). Du kan skrive ut flere linjer på én gang, som tar med linjeskift, ved å bruke tre anførselstegn, slik:

```
print("""
...
""")
```

c) Lag en løkke som går over alle elementene i listen, og kaller på funksjonen som printer ut verset med det gitte elementet som argument.

d) **Bonusoppgave: Remix** For å få ett tilfeldig element fra listen kan du bruke funksjonen `choice` fra `random`-biblioteket. Bruk `choice` til å skrive ut et gitt antall vers der du hver gang velger ut et tilfeldige element som skal svaie.

Løsning oppgave 3 *«Vi har ei gammel tante»*

```
svaie_elementer = [''rumpa'', ''fjøra'', ''hatten'', ''nesa'', ''veska'', ''hestehalen'', ''paraplyen'', ''øyevippa'', ''handleposen'', ''humøret'', ''jakka'', ''skjerfet'']
```

```
import random

svaieelementer = ["rumpa", "fjøra", "hatten", "nesa", "veska", "hestehalen", "paraplyen", "øyevippa", "handleposen", "humøret", "jakka", "skjerfet", "smilet", "skjørtet"]

def print_monica_vers(element):
    print(f"""Vi har ei gammel tante som heter Monica
    Og når  a   går på torvet
    Vi hermer etter  a  !
    For sånn svaier {element}, og {element} svaier sånn
    Og sånn svaier {element}, og {element} svaier sånn
    !""")

monica_vers(svaieelementer[0])

for element in svaieelementer:
    print_monica_vers(element)

for _ in range(12):
    print_monica_vers(random.choice(svaieelementer))
```

Oppgave 4 *Statistiske egenskaper*

I denne oppgaven skal vi beregne ulike typer beskrivende statistikk.

Start med å lage et program med et par lister over (f.eks.) målinger vi har utført:

```
forsøk1 = [5.4, 3.2, 5.6, 5.8, 3.9, 4.5, 7.3, 8.4]
forsøk2 = [5, 4, 7, 3, 7, 4, 6, 3, 4, 5]
forsøk3 = [32.3, 45.1, 33.4, 23.1, 65.3, 45.6]
forsøk4 = [3, 4, 4, 5, 2, 3, 5, 2, 6, 3, 4, 4, 5]
```

For hver av deloppgavene under, prøv å gi hver av disse som argument – regn gjerne litt på det på papir også for å teste om programmet ditt fungerer som det skal.

a) Lag en funksjon `gjennomsnitt` som regner ut *gjennomsnittet* av tallene i listen, det vil si summen av alle delt på antall tall.

b) Lag en løkke som går igjennom alle forsøkene, og så skriver ut gjennomsnittet av elementene i hvert forsøk ved hjelp av funksjonen du lagde i forrige deloppgave.

c) Lag en funksjon `median` som regner ut *medianen* av tallene i listen, det vil si det midterste tallet, eller gjennomsnittet av de to midteste tallene dersom vi har et partall antall elementer, når listen er sortert,

d) Lag en funksjon `typetall` som regner ut *typetallet* av tallene i listen, det vil si det tallet det forekommer flest av.

e) Sammenlign de ulike forsøkene. Er noen av verdiene (dvs. gjennomsnitt, median eller typetall) sammenfallende? Hva tror du det kommer av?

Løsning oppgave 4 *Statistiske egenskaper*

```
def gjennomsnitt(liste):
    sum_tall = 0

    for tall in liste:
        sum_tall += tall

    return sum_tall/len(liste)

```

```
def median(liste):
    liste.sort()

    if len(liste)%2 == 1:     # odde antall elementer
        midterste_index = int(len(liste)/2)
        return liste[midterste_index]

    else:                     # likt antall elementer
        index2 = int(len(liste)/2)
        index1 = index2 - 1
        return (liste[index1] + liste[index2])/2


def typetall(liste):
    flest_forekomster = 0
    typetall = 0

    for tall_orgiginal in liste:
        antall_forekomster = 0
        for tall_sammenlign in liste:
            if tall_orgiginal==tall_sammenlign:
                antall_forekomster += 1

        if antall_forekomster > flest_forekomster:
            flest_forekomster = antall_forekomster
            typetall = tall_orgiginal

    return typetall

forsøk1 = [5.4, 3.2, 5.6, 5.8, 3.9, 4.5, 7.3, 8.4]
forsøk2 = [5, 4, 7, 3, 7, 4, 6, 3, 4, 5]
forsøk3 = [32.3, 45.1, 33.4, 23.1, 65.3, 45.6]
forsøk4 = [3, 4, 4, 5, 2, 3, 5, 2, 6, 3, 4, 4, 5]

for målinger in [forsøk1, forsøk2, forsøk3, forsøk4]:
    print("Gjennomsnitt, median, typetall: ", \
            gjennomsnitt(målinger), median(målinger),
               typetall(målinger))
```

Oppgave 5 *Fakultet med rekursjon*

Rekursive funksjoner er funksjoner som kaller på seg selv. I matematikken finnes det mange eksempler på rekursive sammenhenger. I denne oppgaven skal vi se på hvordan vi kan beskrive fakultet ved hjelp av rekursjon. Fakultet er definert som

$$\text{tall!} = \text{fakultet}(\text{tall}) = \text{tall} \cdot (\text{tall} - 1) \cdot (\text{tall} - 2) \cdot \ldots \cdot 1$$

Dette kan vi beskrive rekursivt:

$$\text{tall!} = \text{fakultet}(\text{tall}) = \text{tall} \cdot \text{fakultet}(\text{tall} - 1)$$

a) Lag en funksjon `fakultet(tall)`. Dersom tall er `0` skal funksjonen bare returnere `1`. Dersom `tall` er større enn `0` skal funksjonen "kalle på seg selv" og returnere `tall*fakultet(tall-1)`

b) Bruk funksjonen til å regne ut fakultet av 10, 1, og 13

Løsning oppgave 5 *Fakultet med rekursjon*

a)

```
def fakultet(tall):
    if (tall == 0):
        return 1
    else:
        return tall * fakultet(tall-1)
```

b)

```
print(fakultet(5))
```

```
120
```

Oppgave 6 *Elekstrisitet: Strøm – spenning – resistans*

Viktige egenskaper i en elektrisk krets er spenningen U, målt i volt, strømmen I, målt i ampere, og motstanden/resistansen R, målt i ohm (Ω). Forholdet mellom disse enhetene er gitt ved *Ohms lov*: $U = R \cdot I$

a) Lag tre funksjoner som regner ut strøm, spenning og resistansen i kretsen gitt at man vet de to andre størrelsene.

b) Bruk funksjonene dine til å regne ut:

- Spenningen når strømmen er 10 A og motstanden 1.7 Ω
- Resistansen når spenningen er 230 V og strømmen er 20 A
- Strømmen når spenningen er 5 V og motstanden er 200 Ω

c) Når flere motstander er koblet i seriekobling, blir den totale motstanden lik summen av alle motstandene i serien. $R_{tot} = R_1 + R_2 + \cdots + R_N$. Lag en funksjon `seriekobling` som tar inn en liste med motstander og returnerer den totale motstanden.

d) Dersom spenningen er 5 V, og motstandene på 10, 5, 2 og 11 Ω er koblet i serie, hva blir strømmen da?

e) **Utfordring:** Når motstander er koblet i parallell, er sammenhengen mellom dem $\frac{1}{R_{tot}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_N}$. Lag en funksjon `parallellkobling` som regner ut den totale motstanden i kretsen. Hva blir strømmen nå, dersom motstandene og spenningen er den samme?

Løsning oppgave 6 *Elekstrisitet: Strøm – spenning – resistans*

a) Vi definerer funksjonene:

```
1  def spenning(I,R):
2    return R*I
3
4  def strøm(U,R):
5    return U/R
6
```

```
def motstand(U,I):
  return U/I
```

b) 17 V, 11.5 Ω, og 25 mA

c)

```
def seriekobling(motstander):
  tot_motstand = 0
  for motstand in motstander:
    tot_motstand += motstand
  return tot_motstand
```

d) Vi bruker funksjonene vi har laget:

```
motstander  = [10,5,2,11]
R = seriekobling(motstander)
U = 230
print(f"Strømmen er {strøm(U,R):.2f} A")

>>> Strømmen er 8.21 A
```

e) Vi lager en funksjon som er ganske lik som for seriekobling, men må dele på 1 i alle ledd:

```
def parallellkobling(motstander):
  tot_motstand_inv = 0
  for motstand in motstander:
    tot_motstand_inv += 1/motstand
  tot_motstand = 1/tot_motstand_inv
  return tot_motstand


motstander  = [10,5,2,11]
R = parallellkobling(motstander)
U = 230
print(f"Strømmen er {strøm(U,R):.2f} A")

>>> Strømmen er 204.91 A
```

Array og plotting

Oppgave 7 *Plotte en annengradsfunksjon*

a) Definer en funksjon for den matematiske funksjonen

$$f(x) = x^2 - 5x + 9$$

b) Bruk funksjonen `linspace` for å generere x-verdier mellom 0 og 5. Lagre disse i en `array`. Finn tilsvarende y-verdier ved å sende x-verdiene inn som et argument (du kan sende med hele arrayet som ett argument). Skriv ut begge listene.

c) Plot $f(x)$ mellom $x = 0$ og $x = 5$. Prøv å endre antall x-verdier du valgte i (b) og se hvordan det endrer plottet (du kan ta bort print-delen her hvis du vil).

d) Legg til en tittel til plottet og sett navn på aksene.

Løsning oppgave 7 *Plotte en annengradsfunksjon*

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2 - 5*x + 9

def plott_f(n):
    xer = np.linspace(0, 5, n)
    yer = f(xer)

    for (x, y) in zip(xer, yer):
        print(f"(x, y): ({x}, {y})")

    plt.plot(xer, yer)
    plt.title(f"f(x) for x mellom 0 og 5, n = {n}")
    plt.xlabel("x")
```

```
    plt.ylabel("y")
    plt.show()

plott_f(5)
plott_f(10)
plott_f(50)
```

Oppgave 8 *Plotte annengradsfunksjon*

En funksjon er definert slik:

$$f(x) = x^2 + 3x - 10$$

I denne oppgaven skal vi bruke Python til å plotte funksjonen for $x = -5, \ldots, 5$

a) Definer en funksjon `f(x)` som tar inn en x-verdi og returnerer $f(x)$ slik den er definert over.

b) Bruk `arange` fra `numpy` (eller `pylab`) til å opprette en array, `x_verdier` som inneholder verdier fra -5 til 5 med steglengde 0.5.

c) Bruk funksjonen `f` du definerte i *a)* til å regne ut tilhørende y-verdier og lagre dem i en variabel, `y_verdier`.

d) Bruk `plot` og `show` fra `matplotlib.pyplot` (eller `pylab`) til å plotte funksjonen med `x_verdier` på x-aksen og `y_verdier` på y-aksen.

e) Bruk `xlim` til å endre plotteområde til mellom -5 og 5.

Løsning oppgave 8 *Plotte annengradsfunksjon*

a)

```
def f(x):
  return x**2 + 3*x - 10
```

b)

```
from numpy import linspace
x_verdier = linspace(-5, 5, 101)
```

c)

```
y_verdier = f(x_verdier)
```

d)

```
from matplotlib.pyplot import plot, show
plot(x_verdier, y_verdier)
show()
```

e)

```
from matplotlib.pyplot import plot, show, xlim
plot(x_verdier, y_verdier)
xlim(-5, 5)
show()
```

Oppgave 9 *Bakterievekst*

En kultur med bakterier inneholder 1100 bakterier. Hver time øker mengden bakterier med 3 %. I denne oppgaven skal vi lage et plot over populasjonsutviklingen til bakteriene over et døgn.

a) Opprett en variabel `bakteriemengde` og sett den til å inneholde 1100 bakterier.

b) Opprett en for-løkke som løkker over 24 timer og oppdaterer `bakteriemengde` hver time slik at den nye verdien til `bakteriemengde` er den gamle verdien til `bakteriemengde` + 3 % av den gamle verdien av `bakteriemengde`.

c) Skriv ut hvor mange bakterier som er i kulturen etter et døgn.

d) For å plotte bakterieveksten trenger vi en liste over bakteriemengder for hver time og en liste over timer. Oppdater programmet ditt til å lage en liste `bakteriemengde_liste` som inneholder variabelen bakteriemengde og en tom liste `time_liste` som inneholder verdien 0 (0 timer har gått) før løkka.

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

f) Oppdater programmet til å legge til hvilken time som blir den neste i slutten av `time_liste` med `append` for hver runde i løkka.

g) Plot bakterieveksten mot timene med `plot`.

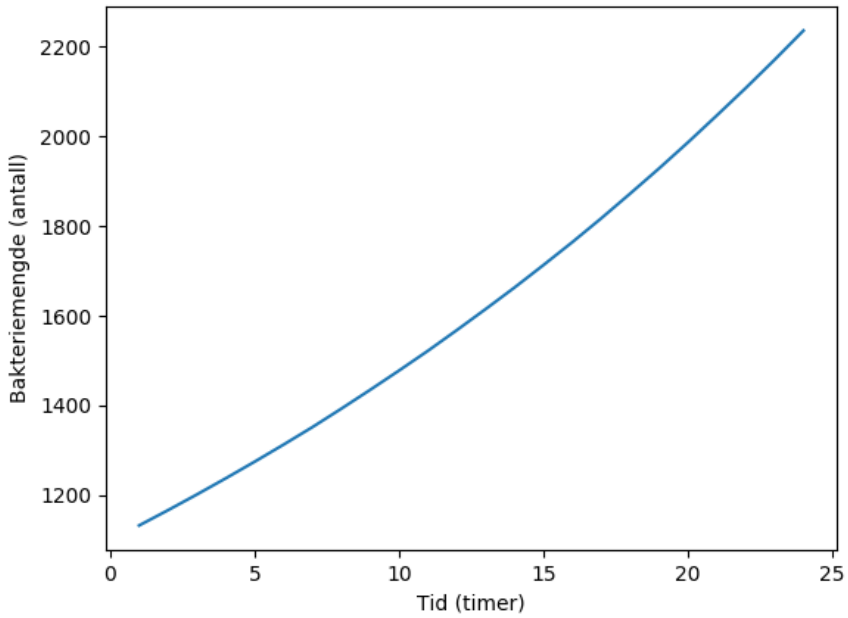
h) Legg på merkelapper på x og y aksen og en tittel til plottet.

i) Øk antall timer til 300 og se på plottet ditt. Er det noen svakheter med denne modellen? Hva kommer de av? Er dette realistisk i forhold til hva som skjer i naturen?

Løsning oppgave 9 *Bakterievekst*

a)–c)

```
bakteriemengde = 1100

for time in range(24):
    bakteriemengde += round(0.03*bakteriemengde)

print(f"Etter {time+1} timer er det {
   bakteriemengde} bakterier i kulturen.")
```

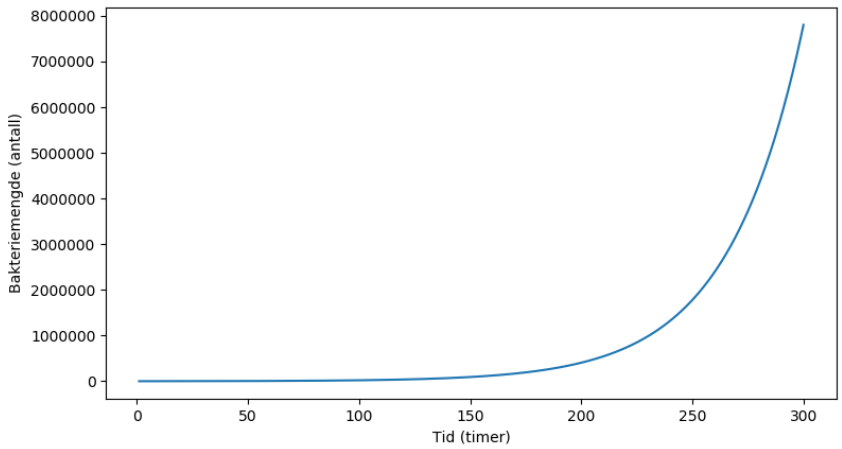


d)–f)

```
from matplotlib.pyplot import *  # Vi kan også importere fra pylab

bakteriemengde = 1100

bakteriemengde_liste = []
time_liste = []

for time in range(24):
    bakteriemengde += round(0.03*bakteriemengde)

    bakteriemengde_liste.append(bakteriemengde)
    time_liste.append(time+1)

print(f"Etter {time+1} timer er det {bakteriemengde} bakterier i kulturen.")

plot(time_liste, bakteriemengde_liste)
xlabel("Tid (timer)")
ylabel("Bakteriemengde (antall)")

```

```
show()
```

g) Plott:



Kode:

```
from matplotlib.pyplot import *  # Vi kan også importere fra pylab

bakteriemengde = 1100

bakteriemengde_liste = []
time_liste = []

for time in range(300):
    bakteriemengde += round(0.03*bakteriemengde)

    bakteriemengde_liste.append(bakteriemengde)
    time_liste.append(time+1)

print(f"Etter {time+1} timer er det {bakteriemengde} bakterier i kulturen.")

plot(time_liste, bakteriemengde_liste)
xlabel("Tid (timer)")
ylabel("Bakteriemengde (antall)")

```

```
20 show()
```

Oppgave 10 *Arealberegninger*

I denne oppgaven skal vi plotte ulike funksjoner som beregner areal, og sammenligne arealutviklingen som en funksjon av sidelengde/radius.

a) Hvis du har en kvadrat med en gitt sidelengde, en sirkel med den samme sidelengden som diameter og en likesidet trekant med samme sidelengde, hvilket har størst areal? Gjett fort uten å regne på noe!

b) Lag en funksjon `regn_ut_areal_kvadrat` som tar inn ett argument, `sidelengde` og returnerer arealet av et kvadrat med denne sidelengden – altså `sidelengde**2`.

c) Lag et array `lengdeverdier` med verdier mellom `0` og `10`. Velg selv hvor mange verdier du vil jobbe med.

d) Kall på funksjonen `regn_ut_areal_kvadrat` med `lengdeverdier` som input. Lagre resultatet i et nytt array, `areal_kvadrat`.

e) Plott `areal_kvadrat` mot `lengdeverdier` ved hjelp av `plot`-funksjonen.

f) Lag en ny funksjon, `regn_ut_areal_sirkel` som også tar inn ett argument, `diameter`, og returnerer arealet av en sirkel med gitt diameter (husk å regne om til radius).

g) Kall på funksjonen med `lengdeverdier` (altså den samme array som vi brukte tidligere), lagre resultatene i en ny array og plott dette i samme plott.

h) Legg på merkelapper for hver av de, ved hjelp av `label` og `legend`.

i) Lag enda en ny funksjon, `regn_ut_areal_trekant` som også tar inn ett argument, `sidelengde`, og returnerer arealet av en likesidet trekant. Arealet av en likesidet trekant med sidelengde s er gitt ved

$$\frac{\sqrt{3}}{4}s^2$$

Kall igjen på funksjonen med `lengdeverdier`, lagre resultatene i en ny array og plott dette i samme plott.

j) Se på plottet og se hvilket arealplott som vokser raskest. Stemmer det med gjetningen din fra første deloppgave?

Løsning oppgave 10 *Arealberegninger*

```
from pylab import *

def regn_ut_areal_kvadrat(sidelengde):
    return sidelengde**2

def regn_ut_areal_sirkel(diameter):
    radius = diameter/2*2
    return pi*radius**2

def regn_ut_areal_trekant(sidelengde):
    return sqrt(3)/4*sidelengde**2

lengdeverdier = np.linspace(0, 10, 101)

areal_kvadrat = regn_ut_areal_kvadrat(lengdeverdier)
areal_sirkel = regn_ut_areal_sirkel(lengdeverdier)
areal_trekant = regn_ut_areal_trekant(lengdeverdier)

plt.plot(lengdeverdier, areal_kvadrat, label="Kvadrat")
plt.plot(lengdeverdier, areal_sirkel, label="Sirkel")
plt.plot(lengdeverdier, areal_trekant, label="Trekant")

plt.legend()
plt.show()
```

Oppgave 11 *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 seg, dvs. så lenge den er under `2000`. 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 11 *Spare med BSU-konto*

a)–c)

```
1  pengemengde = 1000
2  antall_år = 0
3
4  while pengemengde < 2*1000:
5      pengemengde *= 1.035
6      antall_år += 1
7
8  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)

```
1
2  pengemengde_liste = []
3  år_liste = []
4
5  pengemengde = 1000
6  antall_år = 0
7  rente = 0.035
```

e) - f)

```
1
2  pengemengde_liste = []
3  år_liste = []
4
5  pengemengde = 1000
6  antall_år = 0
7  rente = 0.035
8
9
10 while pengemengde < 2*1000:
11     pengemengde_liste.append(pengemengde)
12     år_liste.append(antall_år)
13     pengemengde *= (1 + rente)
14     antall_år += 1
```

g)

```
1  from matplotlib.pyplot import *
2
3  pengemengde_liste = []
4  år_liste = []
5
6  pengemengde = 1000
7  antall_år = 0
8  rente = 0.035
```

```
9
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  show()
```

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

h)

```
1   from matplotlib.pyplot import *
2
3   pengemengde_liste = []
4   år_liste = []
5
6   pengemengde = 1000
7   antall_år = 0
8   rente = 0.035
9
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 12 *Innhegning*

Du har 100 meter gjerde og vil lage en rektangulær innhegning. Vi skal se på hvor stort areal denne innhegningen vil ha.

a) Tegn opp et rektangel på papir. Hvis du sier at den ene siden er x meter lang, hvor lang blir da de tre andre sidene i rektangelet? Skriv det opp på arket.

b) Skriv ut uttrykket for arealet til hele innhegningen.

c) Hvor stor kan x maksimalt være?

Du skal nå lage et Python-program som plotter arealet av innhegningen som en funksjon av x.

d) Fyll inn i skjelettkoden under for å lage programmet:

```
1 from pylab import *
2
3 x = arange(0, ..., 0.1)
4 A = ...
5
6 plot(x, A)
7 xlabel('x')
8 ylabel('Areal')
9 show()
```

e) Kjør programmet og se på figuren som tegnes. Hvilket valg av x er det som gir størst mulig areal? Hva slags innhegning er det vi ender opp med å lage?

Løsning oppgave 12 *Innhegning*

a) En av de resterende siden vil ha lengde x og to av de resterende sidene vil ha lengde $(100\text{ m} - 2x)/2 = 50\text{ m} - x$.

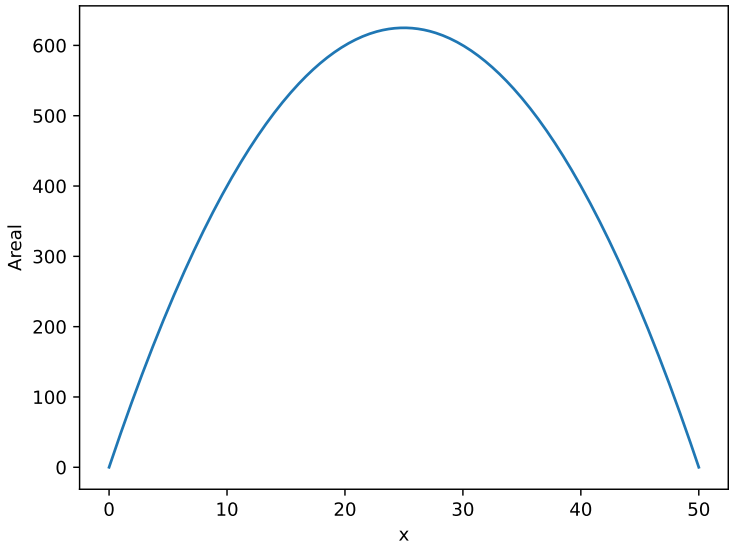
b) Rektangelet vil ha et areal på $A = x(50\text{ m} - x)$.

c) x kan maksimalt være 50 m.

d)

```
from pylab import *

x = arange(0, 50.1, 0.1)
A = x * (50 - x)

plot(x, A)
xlabel('x')
ylabel('Areal')
show()
```

e) Av grafen under ser vi at vi får maksimalt areal når $x = 25$ m, altså når rektangelet er et kvadrat.


600
500
400
300
200
100
0
Areal
0
10
20
30
40
50
x