Bonus-oppgaver

Her er et knippe bonusoppgaver for de som vil ha litt mer å bryne seg på. Disse oppgavene er ikke sortert på nivå, men etter fag. Samlingen er fin som ekstra programmeringstrening eller som inspirasjon til realfagsrelevante Python-oppgaver.

1 Matematikk

Oppgave 1 *Annengradsfunksjon*

En annengradsfunksjon er definert slik:

$$f(x) = x^2 + 0.3x - 1$$

a) Definer funksjonen som en Python-funksjon, `f(x)`, som tar inn en x-verdi og returnerer tilhørende y-verdi.

b) Test funksjonen din ved å kalle på den med følgende x verdier og skriv ut resultatet til terminalen:

1. $x = 0$
2. $x = 1$
3. $x = -0.4$

Løsning oppgave 1 *Annengradsfunksjon*

a)

```
1  def f(x):
2    return x**2 + 0.3*x - 1
```

b)

1.

```python
x = 0
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=0, f(x)=-1.0
```

2.

```python
x = 1
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=1, f(x)=0.30000000000000004
```

3.

```python
x = -0.4
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=-0.4, f(x)=-0.96
```

Oppgave 2 *Arealet av en trekant*

Arealet av et rektangel er gitt ved:

$$\text{lengde} \times \text{høyde}$$

a) Lag en funksjon, `firkantareal`, som tar inn to tall, `lengde` og `høyde` og returnerer arealet av et rektangel med tilsvarende lengde og høyde

b) Bruk funksjonen til å finne arealet av et rektangel med høyde 31 og lengde 43

Arealet av en trekant er halvparten av arealet til et rektangel som har samme grunnlinje og høyde:

$$\frac{\text{lengde} \times \text{høyde}}{2}$$

I denne oppgaven skal du bruke funksjonen, `firkantareal`, du lagde i oppgave *a)* til å regne ut trekantareal i stedet

c) Lag en funksjon, `firkantareal`, som tar inn lengden og høyden til en trekant og returnerer arealet til trekanten. Funksjonen skal kalle på funksjonen du lagde i oppgave *a)*

d) Bruk funksjonen til å finne arealet av en trekant med høyde 14 og lengde 26

Løsning oppgave 2 *Arealet av en trekant*

a)

```
def firkantareal(lengde, høyde):
  return lengde*høyde
```

b)

```
firkantlengde = 31
firkanthøyde = 43

print(f'Areal: {firkantareal(firkantlengde,
   firkanthøyde)}')
```

```
Areal: 1333
```

c)

```
def trekantareal(lengde, høyde):
  return firkantareal(lengde, høyde)/2
```

d)

```
trekantlengde = 14
trekanthøyde = 26

print(f'Areal: {trekantareal(trekantlengde,
   trekanthøyde)}')
```

Oppgave 3 *Fibonacci rekken*

Fibonacci rekken er en kjent tallfølge som naturlig oppstår mange steder i naturen. Tallfølgen går som følger:

$$1, 1, 2, 3, 5, 8, 13, 21, ... \tag{1}$$

Vi kan skrive Fibonacci rekken som en rekursiv tallfølge etter denne formelen:

$$a_n = a_{n-1} + a_{n-2}, \tag{2}$$

hvor $a_0 = 1$ og $a_1 = 1$.

En interessant egenskap ved Fibonacci rekken er at forholdet mellom to etterfølgende tall i følgen går mot det gyldne snitt, $\phi = \frac{1+\sqrt{5}}{2} \approx 1.62$. Det vil si at

$$\frac{a_n}{a_{n-1}} \to \phi. \tag{3}$$

a) Du skal nå lage et program som skriver ut de første 10 tallene i Fibonacci rekka. Start med å opprette en variabel `forrige_tall =1` og en variabel `fibonacci_tall =1`.

b) Skriv så ut `forrige_tall` og `fibonacci_tall` til brukeren av programmet.

c) Bruk en `for`-løkke som repeteres 8 ganger (10 tall - 2 start-tall) som skriver ut de resterende 8 tallene i Fibonacci rekka. HINT: Her kan det være lurt å opprette det nye Fibonacci tallet i en variabel `nytt_fibonacci_tall` før du oppdaterer verdien til `forrige_tall` og `fibonacci_tall`.

d) Endre programmet slik at du bruker input til å be brukeren om hvor mange Fibonacci tall du skal skrive ut til skjermen.

e) Oppdater programmet slik at du og skriver ut forholdet mellom `forrige_tall` og `fibonacci_tall`. Blir dette forholdet ca lik 1.62?

f) Endre start-tallene fra $a_0 = 1$ og $a_1 = 1$ til noe annet (f.eks $a_0 = 2$ og $a_1 = 1$), hvordan endrer det oppførselen til rekka?

Løsning oppgave 3 *Fibonacci rekken*

a)

```
forrige_tall = 1
fibonacci_tall = 1
```

b)

```
print(forrige_tall)
print(fibonacci_tall)
```

c)

```
for tallnummer in range(8):
    nytt_fibonacci_tall = fibonacci_tall +
       forrige_tall
    forrige_tall = fibonacci_tall
    fibonacci_tall = nytt_fibonacci_tall
    print(fibonacci_tall)
```

d)

```
antall_tall = int(input('Hvor mange Fibonacci tall
    ønsker du? '))

forrige_tall = 1
fibonacci_tall = 1
print(forrige_tall)
print(fibonacci_tall)
```

```
7
8  for tallnummer in range(antall_tall - 2):
9      nytt_fibonacci_tall = fibonacci_tall +
           forrige_tall
10     forrige_tall = fibonacci_tall
11     fibonacci_tall = nytt_fibonacci_tall
12     print(fibonacci_tall)
```

e)

```
1  for tallnummer in range(antall_tall - 2):
2      nytt_fibonacci_tall = fibonacci_tall +
           forrige_tall
3      forrige_tall = fibonacci_tall
4      fibonacci_tall = nytt_fibonacci_tall
5      forhold = fibonacci_tall / forrige_tall
6      print(f'Fibonacci tall: {fibonacci_tall},
           forhold mellom nåværende og forrige
           Fibonacci tall: {forhold}')
```

f) Vi ser at forholdet fortsatt går mot det gyldne snitt. I tillegg, hvis vi bruker $a_0 = 2$ og $a_1 = 1$ får vi de mindre kjente, men mer interessante *Lucas tallene*. Du kan lære mer om Lucas Tallene i denne fine YouTube videoen https://www.youtube.com/watch?v=PeUbRXnbmms av Numberphile (Brady Haran) og Matt Parker.

```
1  antall_tall = int(input('Hvor mange Fibonacci tall
       ønsker du? '))
2
3  forrige_tall = 1
4  fibonacci_tall = 1
5  print(forrige_tall)
6  print(fibonacci_tall)
7
8  for tallnummer in range(antall_tall - 2):
9      nytt_fibonacci_tall = fibonacci_tall +
           forrige_tall
10     forrige_tall = fibonacci_tall
11     fibonacci_tall = nytt_fibonacci_tall
```

Oppgave 4 *Fakultet*

Si at du har 5 forskjellige små skulpturer du skal sette opp på rekke. Hvor mange ulike måter kan du gjøre dette på? For den første har du 5 valg, deretter har du 4 valg, så 3, og så videre. Dermed blir antall kombinasjoner til

$$5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

I matematikken skriver vi dette mer kompakt som 5!, som vi kaller *fakultet*. Å rangere n ting kan altså gjøres på $n!$ måter, som blir

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n.$$

For små tall er det enkelt å regne ut fakultet for hånd, men dersom n begynner å bli litt større vil dette ta fryktelig lang tid.

a) Du skal nå skrive et program som kan regne ut fakultet for oss. Du kan enten prøve å gjøre dette helt selv, eller følge vår steg-for-steg"instrukser under.

- Spør brukeren om hva n er, lagre svaret i en variabel. Husk å konvertere svaret med `int(input())`.
- Lag en variabel `fakultet`, som du gir verdien 1.
- Lag en for-løkke som går fra 1 til n.
- For hver iterasjon av løkka, gang `fakultet`-variabelen din med tallene du løkker over.
- Skriv ut det endelige svaret så brukeren kan se det.

b) Test at programmet ditt gir $5! = 120$.

c) En vanlig kortstokk har 52 unike kort. Hvor mange mulige rekkefølger kan vi få om vi stokker en kortstokk?

d) I noen kortspill legger man til 2 jokere i kortstokken, slik at det nå er 54 kort i kortstokken. Hvor mange måter blir det nå å stokke kortstokken på?

Løsning oppgave 4 *Fakultet*

a)

```python
n = int(input("Velg n: "))

fakultet = 1
for i in range(1, n+1):
        fakultet *= i

print(f"{n}! = {fakultet}")
```

b)

```
5! = 120
```

c)

```
52! = 120! =
669502913449127057588118054090372586752746333138029810295671352301
```

1.1 Naturfag

Oppgave 5 *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) Bruke 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 spennignen er 5 V og motstanden er 200 Ω

Løsning oppgave 5 *Elekstrisitet: Strøm – spenning – resistans*

a) Vi definerer funksjonene:

```
def spenning(I,R):
  return R*I

def strøm(U,R):
  return U/R

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

b) 17 V, 11.5 Ω, og 25 mA

Oppgave 6 *Vekstfaktor*

I en petriskål lever en bakteriekultur som består av 1000 bakterier som formerer seg raskt nok til at mengden bakterier øker med 50% hver time. I denne opppgaven skal vi bruke plotting og python til å modellere og visualiserer veksten i bakteriekulturen over 10 timer

a) Opprett en variabel, `vekstfaktor`, som har vekstfaktoren tilhørende 50%

b) Bruk `arange` fra `pylab` til å opprette en array, `timer`, med tallene fra 1 til 10

Nå vil vi regne ut hvor mange bakterier som er i petri-skålen hver time. For å gjøre det, må vi først opprette en tom array og bruke en løkke til å "fylle" inn antall bakterie hver time. Men først, hva er egentlig en tom array? Vel, vi kan jo bruke en array hvor alle elementene er lik 0. Dette er det en funksjon for i `pylab` fra før av, og den heter `zeros`. Hvis vi skriver `ti_nuller =zeros(10)`, vil vi få en variabel `ti_nuller` som består av ti element som alle er lik null.

c) Bruk `zeros` fra `pylab` til å opprette en tom array, `bakteriemengde` med

10 elementer. Husk å importere `zeros` først!

Det neste steget er å endre verdiene til hvert element i arrayen vårt. Dette kan vi gjøre med *indeksering*. En array består av mange tall som er etter hverandre, og hvis du vil ha ut et tall fra en array bruker vi klammeparanteser. Hvis vi har en array-variabel, `x`, kan rett og slett skrive `x[0]` for å hente ut det første elementet i arrayen. Tilsvarende kan vi skrive `x[1]` for å hente ut det andre elementet i arrayen, osv. Under har vi et eksempel

```
from pylab import arange

tallrekke = arange(10)*2
første_tall = tallrekke[0]
femte_tall = tallrekke[4]

print(første_tall)
print(femte_tall)
```

```
0
8
```

Tilsvarende, kan vi starte med en tallrekke, og endre enkeltelement! Se eksempelet under.

```
from pylab import arange

tallrekke = arange(4)*2
print(tallrekke)

tallrekke[0] = -1
tallrekke[2] = 0

print(tallrekke)
```

```
[0 2 4 6]
[-1  2  0  6]
```

Vi ser her at vi bruker `tallrekke[i] =x` for å sette verdien til element nummer $i + 1$ i arrayet lik x.

d) Bruk array-indeksering (klammeparanteser) til å sette det første elementet (element nummer 0) i `bakteriemengde` til `1000` som er startverdien.

e) Bruk en løkke `for time in range(1,10)` til å løkke igjennom alle timene fra 1 til 9

f) Bruk arrayindeksering til å sette bakteriemengden for tid `time` til å være lik vekstfart multiplisert med bakteriemengden for tid `time-1`. **Hint:** `bakteriemengde[time-1]`

g) Bruk `plot` og `show` funksjonene fra `pylab` til å plotte bakteriemengde på yaksen og timer på y-aksen

h) Bruk `xlabel` og `ylabel` funksjonene fra `pylab` til å legge merkelapper på x og y-aksen

i) Refleksjonsoppgave: Hva vil skje med bakteriene etterhvert som tiden går? Hva synes du om denne modellen? Er det noen svakheter ved en slik modell?

Løsning oppgave 6 *Vekstfaktor*

a)

```
1 vekstfaktor = 1.5
```

b)

```
1 from pylab import arange
2
3 timer = arange(10)
```

c)

```
1 from pylab import arange, zeros
2
3 bakteriemengder = zeros(10)
```

d)

```
bakteriemengder[0] = 1000
```

e)

```
for time in range(1, 10):
    bakteriemengder[time] = bakteriemengder[time-1
       ]*vekstfaktor
```

f)

```
from pylab import plot, show

plot(timer, bakteriemengder)
show()
```

g)

```
from pylab import plot, show, xlabel, ylabel

plot(timer, bakteriemengder)
xlabel("Timer")
ylabel("Antall bakterier")
show()
```

1.2 Biologi

Oppgave 7 *Hvor mye antibiotika i blodet*

Legen din har gitt deg 150 mg antibiotika, som du skal ta hver 24 time. Vi antar at all antibiotika går inn i blodet ditt, og at leveren fjerner 60% av antibiotikanivået i blodet hver 12 time. Likningen for hvor mye antibiotika du har i blodet etter n doser:

$$x_n = 0.40x_{n-1} + 150 \tag{4}$$

a) Bruk en for-løkke til å skrive ut antibiotikainnholdet i blodet ditt for

de første 14 døgnene.

b) Etter 14 døgn slutter du på antibiotikakuren. Skriv ut mengden antibiotika du har i blodet hvert døgn de neste to ukene.

Løsning oppgave 7 *Hvor mye antibiotika i blodet*

a)

```
antibiotikamengde = 0

for dag in range(15):
    antibiotikamengde = antibiotikamengde*0.4 + 150
    print(f'Du har {antibiotikamengde} mg antibiotika i kroppen etter {dag+1} dager')
```

b)

```
antibiotikamengde = 0

for dag in range(15):
    antibiotikamengde = antibiotikamengde*0.4 + 150
    print(f'Du har {antibiotikamengde} mg antibiotika i kroppen etter {dag+1} dager')

for dag in range(15):
    antibiotikamengde = antibiotikamengde*0.4
    print(f'Du har {antibiotikamengde} mg antibiotika i kroppen etter {dag+1} dager')
```

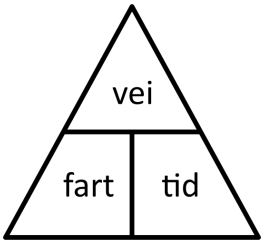
1.3 Fysikk

Oppgave 8 *Vei-fart-tid-kalkulator*

En viktig formel for å beskrive bevegelse er vei-fart-tid formelen,

$$\text{vei} = \text{tid} \cdot \text{fart}$$

Egentlig er det tre formler i ett, for vi kan stokke om på den avhengig av hva vi ønsker å regne ut. Av og til tegnes vei-fart-tid formelen som en pyramide:



a) Lag en funksjon som bruker vei-tid-fart-formelen gitt over, kall denne `vei(fart, tid)`. Vi vil at fart skal oppgis i kilometer i timen, vei i antall kilometer og tid i antall minutter – husk å regne om tiden fra antall timer til antall minutter.

b) Skriv opp uttrykket for å regne ut fart, gitt vei og tid. Lag en funksjon som bruker formelen, kall denne `fart(vei, tid)`.

c) Skriv opp uttrykket for å regne ut tid, gitt vei og fart. Lag en funksjon som bruker formelen, kall denne `tid(vei, fart)`.

d) Du skal nå lage en vei-fart-tid-kalkulator. Når dette programmet kjører skal det først spørre brukeren om de ønsker å regne ut vei, fart, eller tid. Avhengig av svaret skal så programmet spørre om de to andre størrelsene, og så regne ut og skrive ut svaret.

Løsning oppgave 8 *Vei-fart-tid-kalkulator*

a)

```
1 def fart(vei, tid):
2     tid_i_timer = tid/60
3     return vei/tid_i_timer
```

b)

```
def vei(fart, tid):
    tid_i_timer = tid/60
    return tid_i_timer*fart
```

c)

```
def tid(vei, fart):
    tid_i_timer = vei/fart
    return 60*tid_i_timer
```

d)

```
def input_fart():
    return float(input("Hvor fort, i km/t? "))

def input_vei():
    return float(input("Hvor langt, i km? "))

def input_tid():
    return float(input("Hvor lenge, i minutter? ")
        )

regnut = input("Skriv inn 'f' for å regne ut fart, " + \
        "'v' for å regne ut vei " + \
        "og 't' for å regne ut tid:\n")

if regnut=="f":
    print("Regner ut fart!")
    v = input_vei()
    t = input_tid()
    f = fart(v, t)

    print(f"Fart: {f} km / t")

elif regnut=="v":

    print("Regner ut vei!")
    f = input_fart()
    t = input_tid()
    v = vei(f, t)
```

```
28
29      print(f"Vei: {v} km")
30
31  elif regnut=="t":
32
33      print("Regner ut tid!")
34      v = input_vei()
35      f = input_fart()
36      t = tid(v, f)
37
38      print(f"Tid: {t} minutter")
39  regnut = input("Skriv inn 'f' for å regne ut fart,
        " + \
40          "'v' for å regne ut vei " + \
41          "og 't' for å regne ut tid:\n")
42
43  if regnut=="f":
44      print("Regner ut fart!")
45      v = input_vei()
46      t = input_tid()
47      f = fart(v, t)
48
49      print(f"Fart: {f} km / t")
50
51  elif regnut=="v":
52
53      print("Regner ut vei!")
54      f = input_fart()
55      t = input_tid()
56      v = vei(f, t)
57
58      print(f"Vei: {v} km")
59
60  elif regnut=="t":
61
62      print("Regner ut tid!")
63      v = input_vei()
64      f = input_fart()
65      t = tid(v, f)
66
```

```
67      print(f"Tid: {t} minutter")
68
69  else:
70      print("Ukjent alternativ   prøv igjen!")
```

Oppgave 9 *Hvor mye koster morgendusjen din?*

For å varme opp vann trenger vi energi, og i Norge kommer den energien vanligvis fra strøm. Enheten som brukes for å måle hvor mye strøm vi har brukt er gjerne kilowattimer (kWh), som representerer at du har brukt 1000 Watt i en time.

Til vanlig måler vi energiforbruk i Joule (J), så derfor kan det være nyttig å ha en funksjon som oversetter energi fra kWh til J og motsatt. Det kan vi gjøre med denne formelen

$$[\text{kWh}] = 3.6[\text{MJ}], \tag{5}$$

hvor MJ er megajoule, eller 10^6 J.

Når vi varmer opp vann, så bruker vi 4.2 J per liter (L) vann. For å gjøre enhetsomregning lettere er det derfor vanlig å snakke om enheten *kalorier* (cal). En kalori representerer mengden energi som skal til for å varme opp en liter vann en grad Celcius (°C). Altså er en kalori gitt ved formelen

$$[\text{cal}] = 4.2[\text{kJ}], \tag{6}$$

hvor kJ er kilojoule, eller 1000 J.

a) Lag en funksjon `kWh_til_joule(energi_i_kWh)` som oversetter en energimengde oppgitt i kWh til en energimengde oppgitt i Joule.

b) Lag en funksjon `joule_til_kWh(energi_i_J)` som oversetter en energimengde oppgitt i Joule til en energimengde oppgitt i kWh.

c) Lag en funksjon `kalori_til_joule(energi_i_cal)` som oversetter en energimengde oppgitt i kalorier til en energimengde oppgitt i Joule.

d) Lag en funksjon `kalori_til_kWh(energi_i_cal)` som oversetter en energimengde oppgitt i kalorier til en energimengde oppgitt i kWh. (Tips: Bruk `joule_til_kWh` og `kalori_til_joule` funksjonene du allerede har laget).

e) Når vi dusjer bruker vi fort 60 L vann, og dette vannet blir gjerne varmet opp 35 grader. Bruk disse tallene for å estimere hvor mye energi (i kWh) du bruker hver gang du dusjer.

f) En tommelfingerregel er at hver kWh med energi vi kjøper koster ca 1 krone. Hvor mye penger bruker du da på å dusje hvis du dusjer hver dag i et år?

Løsning oppgave 9 *Hvor mye koster morgendusjen din?*

a)

```
1 def kWh_til_joule(energi_i_kWh):
2     return energi_i_kWh*1_000_000*3.6
```

b)

```
1 def joule_til_kWh(energi_i_joule):
2     return energi_i_joule/(1_000_000*3.6)
```

c)

```
1 def kalori_til_joule(energi_i_cal):
2     return energi_i_cal*4.2*1000
```

d)

```
1 def kalori_til_kWh(energi_i_cal):
2     energi_i_joule = kalori_til_joule(energi_i_cal
          )
3     return joule_til_kWh(energi_i_joule)
```

e) Vi må bruke ca 2.5 kWh energi for å varme opp vannet til en dusj.

f) Et år med dusjing daglig koster ca 900 kroner.

1.4 Kjemi

Oppgave 10 *Regne ut pH*

pH-verdien til en væske eller løsning er et mål på hvor sur væsken er, det vil si konsentrasjonen av hydrogenioner $[H^+]$. Formlene for å regne med pH er gitt ved

$$\text{pH} = -\log_{10}([H^+]), \qquad [H^+] = 10^{-\text{pH}}.$$

a) Bruk et Python program til å regne ut konsentrasjonen av hydrogenioner for en væske med pH 3.6.

b) Bruk et Python program til å regne ut pH verdien til en væske med konsentrasjon av hydrogenioner lik $4.7 \cdot 10^{-5}$ mol/L. **Hint:** for å regne ut 10-logaritmen kan bruke funksjonen `log10` som finnes i `math`, `pylab` og `numpy`

c) Cola har en pH på ca 2.5, nøytralt vann har en pH på 7. Hvor mange ganger fler hydrogenioner er det i cola sammenlignet med vann?

Løsning oppgave 10 *Regne ut pH*

a) Merk at variabelnavn i Python bare kan inneholder bokstaver, understrek (_) og tall, så vi kan ikke kalle konsentrasjonen $[H^+]$ i koden vår, vi velger derfor isteden å kalle den `konsentrasjon`. Vi velger også å skrive ut med stilen `:.2g`, bokstaven gbetyr at konsentrasjonen skrives ut som et desimaltall om det er passende, avhengig av størrelsesorden.

```python
pH = 3.6
konsentrasjon = 10**-pH
print(f"pH = {pH:.1f} ---> [H+] = {konsentrasjon:.2g} mol/L")
```

```
pH = 3.6 --->  [H+] = 0.00025 mol/L
```

b) For å regne ut konsentrasjonen trenger vi briggisk logaritme, altså logaritme av base 10. I matematikk skrives denne ofte bare som log, men i programmering er log den naturlige logaritmen. Da bruker vi istedet funksjonen `log10`, for å tydeliggjøre basen.

```python
from pylab import log10

konsentrasjon = 4.7e-5 # mol/L
pH = -log10(konsentrasjon)
print(f"[H+] = {konsentrasjon:.2g} mol/L ---> pH = {pH:.1f}")
```

```
[H+] = 4.7e-05 mol/L ---> pH = 4.3
```

c) Vi kan nå enten kjøre programmet vårt fra første deloppgave to ganger:

```
pH = 2.5 --->  [H+] = 0.0032
pH = 7.0 --->  [H+] = 1e-07
```

Og så dele det ene svaret på det andre. Eller vi kan skrive et nytt lite program som gjør hele beregningen:

```python
pH_cola = 2.5
pH_vann = 7.0

konsentrasjon_cola = 10**-pH_cola
konsentrasjon_vann = 10**-pH_vann

forhold = konsentrasjon_cola/konsentrasjon_vann

print(f"Cola har omtrent {forhold:.0f} ganger fler hydrogenioner enn vann.")
```

```
Cola har omtrent 31623 ganger fler hydrogenioner
   enn vann.
```

Oppgave 11 *Konsentrasjonskalkulator*

Når man gjør eksperimenter i kjemien må man ofte lage løsninger med en bestemt molar konsentrasjon.

Si at vi ønsker å lage en løsning av et stoff X, som veier M g/mol. Vi ønsker å lage en løsning av volum V mL av en gitt konsentrasjon c mol/L.

a) Lag et program som regner ut hvor mange gram av stoff X man må veie ut og løse opp, oppgitt med en nøyaktighet på 0.01 gram.

Hint: For å skrive ut en variabel med 3 desimaler bruk print-formatering med `{m:.2f}`.

b) Test programmet ved å finne mengden bordsalt (NaCl) du trenger for å lage 100 mL løsning med 2.5 mol/L.

c) Ved 25°C klarer man å løse opptil 35.7 gram NaCl per 100 mL vann, etter dette er løsningen metter. Øk konsentrasjonen c i programmet ditt til du er omtrent på denne grensa, hva slags konsentrasjon svarer dette til?

Løsning oppgave 11 *Konsentrasjonskalkulator*

a) Programmet kan se ut som følger. Det viktigste i denne oppgaven er å holde styr på alle enhetene, da er kommentarer et godt virkemiddel.

```
1 stoff = "NaCl"
2 M = ... # Molar masse av stoffet i g/Mol
3 V = ... # Volum i mL
4 c = ... # Ønsket konsentrasjon i mol/L
5
6 m = M*V*c # Masse av stoffet i mg
7 m /= 1000 # Regner om fra mg til g
```

```
8
9  # Skriv ut løsningen
10 print(f"{m:.3f} gram {stoff}")
11 print(f"Løst i {V} mL vann")
12 print(f"Gir en {c} mol/L løsning")
```

b) Når vi setter inn verdiene oppgitt i oppgaven får vi følgende output

```
14.61 gram NaCl
Løst i 100 mL vann
Gir en 2.5 mol/L løsning
```

Dette kan man sjekke høres rimelig ut ved å dobbeltsjekke beregningen for hånd, eller for eksempel å bruke en online konsentrasjonskalkulator.

c) Utifra svaret fra forrige oppgave ser vi at 14.6 gram gir en 2.5 mol/L løsning. Vi trenger altså litt mer en dobbelt så masse for å treffe metningspunktet. Prøver vi oss frem litt finner vi 6.1 mol/L som et godt estimat

```
35.648 gram NaCl
Løst i 100 mL vann
Gir en 6.1 mol/L løsning
```

Denne oppgaven er ment for å vise hvordan et program som allerede er skrevet kan brukes for å utforske andre problemstillinger en de man originalt var ute etter å løse. Dette spørsmålet kan også enkelt besvares med penn og papir, eller ved å skrive ny kode, men det hadde nok tatt lengre tid.