Oppgaver – Dag 3

I dette heftet finner du oppgaver som hører til dag 3 av Kodeskolens programmeringskurs for lærere.

Tema for andre dag er *funksjoner, plot og sammensatte eksempler*. Vi arbeider med oppgavene under kurset, og det du ikke rekker å bli ferdig med er lekse til neste gang. Vi anbefaler at du prøver på ihvertfall et par av oppgavene til hvert tema. Oppgaver markert som bonusoppgaver er litt vanskeligere og du velger selv om du har lyst til å prøve deg på dem. Dersom du står fast er det bare å spørre. I tillegg anbefaler vi å lese i kompendiet hvis det er noen temaer du synes er spesielt vanskelig.

God koding!

Funksjoner

Oppgave 1 *Min første funksjon*

a) Definer en funksjon `min_funksjon` som printer `'Min første funksjon'`. Prøv deretter å kjøre programmet ditt, ikke glem å kalle på funksjonen.

b) Utvid funksjonen din til å ta inn et argument, `mitt_argument`. La også funksjonen din printe ut argumentet. Test funksjonen din ved å kalle på funksjonen med tekststrengen `'Mitt argument'`.

c) Vi ønsker nå å la funksjonen returnere en tekstreng som vi vil printe senere. Da kan vi bruke nøkkelordet `return` og returnere tekststrengen vi vil ha. La funksjonen returnere en valgfri tekststreng.

d) Hvis vi kaller på funksjonen nå på følgende vis `min_tekst =min_funksjon ('Mitt argument')` vil variabelen `min_tekst` inneholde resultatet av

funksjonen vår. Print variabelen og se hva den inneholder. Prøv deretter å endre hva funksjonen returnerer og se på hva som deretter printes.

Løsning oppgave 1 *Min første funksjon*

a)

```
def min_funksjon():
    print('Min første funksjon')
min_funksjon()
```

Det er viktig å få med seg parentesene etter navnet på funksjonen slik at Python vet at den skal utføre funksjonen.

b)

```
def min_funksjon(mitt_argument):
    print('Min første funksjon')
    print(mitt_argument)

min_funksjon('Mitt argument')
```

c)

```
def min_funksjon(mitt_argument):
    print('Min første funksjon')
    print(mitt_argument)
    return 'Min tekst'

min_funksjon('Mitt argument')
```

d)

```
def min_funksjon(mitt_argument):
    print('Min første funksjon')
    print(mitt_argument)
    return 'Min tekst'

min_tekst = min_funksjon('Mitt argument')
print(min_tekst)
```

Variabelen `min_tekst` vil alltid inneholde variabelen som returneres av funksjonen.

Oppgave 2 *Enkle funksjoner*

a) Lag en funksjon `kvadrat(x)` som tar inn et tall x og returnerer kvadratet x^2.

b) Gjenta oppgaven over, men med funksjonen `kubikk()` som returerer x^3

c) Lag en funksjon `hei(navn)` som tar inn en variabel `navn` og skriver ut en hilsen til navnet.

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

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

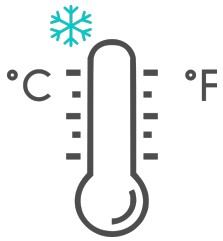
f) **Bonusoppgave:** Lag en funksjon `kalkulator(operasjon,a,b)` som tar inn operasjon som enter er "pluss" eller "minus", og to tall, og regner ut resultatet.

Løsning oppgave 2 *Enkle funksjoner*

```
1  def kvadrat(x):
2      return x**2
3
4  def kubikk(x):
5      return x**3
6
7  def hei(navn):
8      print(f"Hei på deg, {navn}!")
9
10 def pluss(a,b):
11     return a+b
12
```

```python
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 *Konvertering av temperatur*



I Norge oppgir vi temperaturer i celsius, men i USA bruker de ofte fahrenheit. Hvis du finner en kakeoppskrift fra USA kan det for eksempel stå at du skal bake kaken ved 350 grader. Da mener de altså 350°F. Vi vil nå lage et verktøy som kan konvertere denne temperaturen for oss, sånn at vi vet hva vi skal bake kaken på i celsius.

For å regne over fra fahrenheit til celsius bruker vi formelen:

$$C = \frac{5}{9}(F - 32).$$

der F er antall grader i fahrenheit, og C blir antall grader i celsius.

a) Lag et program som spør brukeren om en temperatur i antall grader fahrenheit, og skriver ut den tilsvarende temperaturen i antall grader celsius.

Husk å gjøre om svaret til et tall, med enten `int(input())` eller `float(input())`.

b) Bruk programmet ditt til å finne ut hvor mange grader Celsius 350°F svarer til. Virker det rimelig å skulle bake en kake på denne temperaturen?

c) Programmet du har lagd tar en temperatur i fahrenheit, og gjør om til celsius. Men hva om vi ønsker å gå motsatt vei? Om vi ønsker å lage et nytt program som gjør motsatt, så må vi først ha en formel for F. Klarer du å ta uttrykket

$$C = \frac{5}{9}(F - 32).$$

og løse for F?

d) Lag et nytt program som spør brukeren om en temperatur i antall grader celsius, og så skriver ut den tilsvarende temperaturen i fahrenheit.

e) Bruk funksjonen din til å finne frysepunktet og kokepunktet til vann i fahrenheit.

Løsning oppgave 3 *Konvertering av temperatur*

a) Her må vi først spørre om en temperatur, vi ønsker å kunne oppgi temperaturer som desimaltall, så da bruker vi `float` for å gjøre om svaret fra en tekstvariabel til en tallvariabel:

```
1 F = float(input("Antall grader i fahrenheit: "))
2 C = 5*(F - 32)/9
3 print(f"{F:g} fahrenheit = {C:g} celsius")
```

Her må man kjøre programmet man lagde i a.

```
1 Antall grader i fahrenheit: 350
2 350 fahrenheit = 176.667 celsius
```

Det virker som en rimelig temperatur å bake en kake på. Om vi hadde gjort a feil hadde vi kanskje fått en temperatur vi hadde reagert mer

på.

b) Dette er mer en matematikkoppgave, men vi kan først gange med 9 og dele med 5 på hver side

$$\frac{9}{5}C = F - 32,$$

så legge til 32 på begge sider

$$\frac{9}{5}C + 32 = F,$$

Så bare bytter vi sider:

$$F = \frac{9}{5}C + 32.$$

d) Her kan man enten begynne fra scratch, eller man kan kopiere over og endre programmet sitt. Uansett må man være forsiktig å ikke ha en parentes rundt $C + 32$, da dette hadde vært feil formel.

```
C = float(input("Antall grader i celsius: "))
F = 9/5*C + 32
print(f"{C:g} celsius = {F:g} fahrenheit")
```

e)

```
Antall grader i celsius: 0
0 celsius = 32 fahrenheit
```

```
Antall grader i celsius: 100
100 celsius = 212 fahrenheit
```

Oppgave 4 *Beste pizzapris*

Hallgeir har lyst til å kjøpe pizza og sjekker prisen på nettet. Der står det at pizzaen koster 200 kroner for stor pizza med diameter 40 cm og 110 for liten pizza med diameter 27 cm. Hallgeir har lyst til å velge den pizzaen som er billigst per kvadratcentimeter med pizza og han vet at arealet til en sirkel er gitt ved:

$$\text{Areal} = \pi \times \text{radius}^2$$

a) Opprett en variabel, `pizza_diameter` som skal ha verdien 40

b) For å bruke formelen for arealet av en sirkel trenger vi radiusen. Vi vet at diameteren til en sirkel er det dobbelte av radiusen. Opprett ennå en variabel, `pizza_radius`, som skal være halvparten av diameteren.

c) Regn ut arealet av sirkelen, lagre resultatet i en variabel, `pizza_areal` og skriv ut hvor mange kvadratcm med pizza den store pizzaen inneholder.

d) Regn ut hvor mye pizzaen koster i kroner per kvadratcentimeter med pizza. Skriv svaret ut til skjermen.

e) Opprett et funksjon `pizza_kvadratcentimeter_pris(pizza_diameter, pizza_pris)` som tar inn diameteren til en pizza og radiusen og returnerer prisen per kvadratcentimeter.

f) Bruk funksjonen du lagde i oppgaven over til å regne ut kvadratcentimeterprisen til den lille pizzaen

g) Hvilken pizza skal Hallgeir velge dersom han vil ha pizzaen som er billigst per kvadratcentimeter?

Løsning oppgave 4 *Beste pizzapris*

a)

```
1 pizza_diameter = 40
```

b)

```
1 pizza_radius = pizza_diameter/2
```

c)

```
1 pizza_areal = 3.14 * pizza_radius**2
2 print(pizza_areal)
```

d)

```
1 pizza_pris = 200
2 kr_per_kvadratcentimeter = pizza_pris/pizza_areal
3 print(kr_per_kvadratcentimeter)
```

e)

```
def pizza_kvadratcentimeter_pris(pizza_diameter,
    pizza_pris):
  pizza_radius = pizza_diameter/2
  pizza_areal = 3.14 * pizza_radius**2
  kr_per_kvadratcentimeter = pizza_pris/
      pizza_areal
  return kr_per_kvadratcentimeter
```

f)

```
print(pizza_kvadratcentimeter_pris(27, 110))
```

g)

```
#Pizzaen med diameter 40cm og pris 200 kr gir
    billigst kvadratcentimeter.
```

Oppgave 5 *Dele bamsemums med løkker*

Moren til Trine skal arrangere bursdagsfest og derfor gjør hun nå klar noen godteposer. Hun har kjøpt to pakker bamsemums og til sammen er det 64 bamsemums som skal fordeles likt på de 12 barna som skal være i bursdagen.

For å være sikker på at alle de 12 barna får like mange bamsemums tar moren til Trine 12 bamsemums ut av pakken og fordeler de i godteposer. Når bamsemumsen er fordelt i godteposene tar hun enda 12 bamsemums ut av pakken og fordeler de utover godteposene. Slik fortsetter moren til det er mindre enn 12 bamsemums igjen i pakken.

a) Skriv et program hvor du har en variabel `antall_bamsemums` og sett den lik `64`.

b) Lag en variabel `antall_bamsemums_i_hver_pose` og sett den lik `0`.

c) Lag en "Mens dette stemmer" (`while`) løkke hvor du trekker fra 12 fra `antall_bamsemums` hver iterasjon og øker `antall_bamsemums_i_hver_pose` med en hver iterasjon. Helt til det er mindre enn 12 bamsemums igjen.

d) Skriv ut hvor mange bamsemums som er i godteposene etter utdelingen.

e) Skriv ut hvor mange bamsemums moren til Trine ikke kunne legge i en godtepose.

f) Gjør programmet ditt om til en funksjon `dele` som tar inn antall bamsemums og antall barn og returnerer hvor mange bamsemums som er i godteposene

Løsning oppgave 5 *Dele bamsemums med løkker*

a) - e)

```
antall_bamsemums = 64
antall_bamsemums_i_hver_pose = 0

while antall_bamsemums >= 12:
    antall_bamsemums -= 12
    antall_bamsemums_i_hver_pose += 1

print(f"Etter utdelingen er det {
   antall_bamsemums_i_hver_pose} bamsemums i hver
   pose")
print(f"Moren til Trine har {antall_bamsemums}
   bamsemums igjen")
```

f)

```
def dele(antall_bamsemums,antall_barn):
    antall_bamsemums_i_hver_pose = 0

    while antall_bamsemums >= antall_barn:
        antall_bamsemums -= antall_barn
        antall_bamsemums_i_hver_pose += 1

    return antall_bamsemums_i_hver_pose
```

Oppgave 6 *Forstå andres kode*

En viktig ting å lære seg som programmerer er å kunne forstå hvordan kode skrevet av noen andre fungerer

a) Forklar hva denne koden gjør

```
def generer_tallrekke(n):
    tallrekke = ''
    for i in range(n):
        tallrekke += f'{i+1}'
    return tallrekke
```

b) Hva returneres hvis du kaller på funksjonen med `generer_tallrekke(5)`? Lek datamaskin og gå igjennom steg for steg for hånd.

c) Skriv av koden og kjør den for å se om du hadde rett.

d) Under har vi litt kode som genererer en tallpyramide, les koden og se om du forstår alle kodelinjene.

```
def tallpyramide(n):
    for i in range(n):
        tallrekke = generer_tallrekke(i)
        print(tallrekke)

tallpyramide(5)
```

```
1
12
123
1234
12345
```

e) Hvordan må du modifisere koden over slik at "tallpyramiden" kun består av 1-ere?

Løsning oppgave 6 *Forstå andres kode*

a) Funksjonen lager først en tom streng `''`. Så legger den til tall fra `1` opp til `n` og returnerer den ferdige tallrekka.

b) Den returnerer strengen `'12345'`.

d) Funksjonen `tallpyramide` kalles for `n = 5`. i funksjonen løkkes det fra `i = 0` til `i = 4` og og for hver gang kalles funksjonen `generer_tallrekke(i)`. Dette gir en nye tallrekke fra `1` til `i+1` hver gang. Det gjør at for hver runde i løkka blir tallrekka en lenger.

e) I funksjonen `generer_tallrekke()` kan endre vi linje 4 fra `tallrekke +=f'{i+1}'` til `tallrekke +='1'`

Oppgave 7 *Vinkelsummer*

I trekanter er alltid vinkelsummen 180 grader. Det betyr at dersom du kjenner to av vinklene i en trekant, kan du finne den siste ved følgende uttrykk:

$$\text{vinkel}_3 = 180 - \text{vinkel}_1 - \text{vinkel}_2$$

a) Lag en funksjon `finn_trekant_vinkel(vinkel1, vinkel2)` som tar inn to av vinklene i en trekant og returnerer den siste vinkelen.

b) Vinkelsummen for en mangekant er gitt ved:

$$\text{vinkelsum} = 180 \times (\text{antall kanter} - 2)$$

Lag en funksjon som tar inn antall kanter i en mangekant og returnerer vinkelsummen.

c) **Bonusoppgave:** Lag en funksjon som tar inn en liste av alle vinkler i en mangekant utenom en og returnerer den manglende vinkelen.
Hint 1: Du kan finne antall element i en liste med `len` funksjonen. Kan du fra lengden til listen finne ut hvor mange kanter mangekanten skal ha?
Hint 2: Kan du bruke funksjonen du lagde i forrige deloppgave til å finne hva vinkelsummen til mangekanten skal være?

Hint 3: Bruk en løkke til å finne summen av vinklene i lista.
Hint 4: Hva er da vinkelen som mangler?

Løsning oppgave 7 *Vinkelsummer*

a)

```
def finn_trekant_vinkel(vinkel1, vinkel2):
    return 180 - vinkel1 - vinkel2
```

b)

```
def mangekant_vinkelsum(antall_kanter):
    return 180 * (antall_kanter - 2)
```

c)

```
def finn_mangekant_vinkel(mangekant):
    vinkel = mangekant_vinkelsum(len(mangekant)+1)
    for kant in mangekant:
        vinkel = vinkel - kant
    return vinkel
```

Plot

Oppgave 8 *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 veriden 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 36 og se på plottet ditt. Er det noen svakheter med denne modellen? Hva kommer de av?

Løsning oppgave 8 *Bakterievekst*

a)–c)

```
1 bakteriemengde = 1100
```

```
2
3  for time in range(24):
4      bakteriemengde += round(0.03*bakteriemengde)
5
6  print(f"Etter {time+1} timer er det {
       bakteriemengde} bakterier i kulturen.")
```

d)–f)

```
1  from matplotlib.pyplot import *
2
3  bakteriemengde = 1100
4
5  bakteriemengde_liste = []
6  time_liste = []
7
8  for time in range(24):
9      bakteriemengde += round(0.03*bakteriemengde)
10
11     bakteriemengde_liste.append(bakteriemengde)
12     time_liste.append(time+1)
13
14 print(f"Etter {time+1} timer er det {
       bakteriemengde} bakterier i kulturen.")
15
16 plot(time_liste, bakteriemengde_liste)
17 xlabel("Tid (timer)")
18 ylabel("Bakteriemengde (antall)")
19
20 show()
```

g)

```
1  from matplotlib.pyplot import *
2
3  bakteriemengde = 1100
4
5  bakteriemengde_liste = []
6  time_liste = []
7
8  for time in range(36):
9      bakteriemengde += round(0.03*bakteriemengde)
```

```
10
11      bakteriemengde_liste.append(bakteriemengde)
12      time_liste.append(time+1)
13
14  print(f"Etter {time+1} timer er det {
        bakteriemengde} bakterier i kulturen.")
15
16  plot(time_liste, bakteriemengde_liste)
17  xlabel("Tid (timer)")
18  ylabel("Bakteriemengde (antall)")
19
20  show()
```

Oppgave 9 *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 `pengemendte_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 9 *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 **startsummen** 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)

```
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)

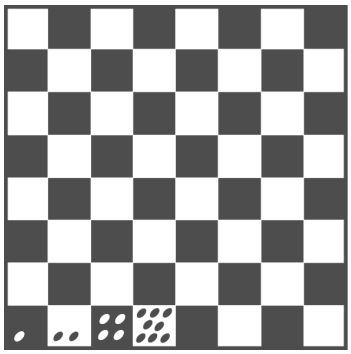
```
from matplotlib.pyplot import *

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)
xlabel("Antall år på konto")
ylabel("Penger i kroner")
title("Penger på BSU - konto")
show()
```

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

Oppgave 10 *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 10 *Sjakk og riskornvekst*

a)

```
antall_riskorn = 0

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

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

b)

```python
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)

```python
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()
```

Sammensatte eksempler

Oppgave 11 *Sjekke om et tall er perfekt*

Et talls ekte divisorer er alle tallets divisorer som er mindre enn tallet selv. For 12 for eksempel, er de ekte divisorene 1, 2, 3, 4 og 6. For noen veldig spesielle tall vil summen av de ekte divisorene være lik tallet selv—vi kaller disse tallene for *perfekte* tall.

Vi kan sjekke om 12 er et perfekt tall ved å summere sammen de ekte divisorene

$$1 + 2 + 3 + 4 + 6 = 16.$$

Ettersom at vi ikke får det samme tallet som vi startet med, er altså ikke 12 et perfekt tall. Det minste perfekte tallet er 6, fordi tallet har de ekte divisorene 1, 2 og 3, som blir til $1 + 2 + 3 = 6$.

Vi kan altså sjekke om et tall er perfekt ved første å finne alle de ekte divisorene, og så summere dem. Dette er enkelt og kjapt for små tall. Derimot blir det fort veldig slitsomt for større tall. La oss derfor skrive et program som gjør det for oss.

a) La oss først fokusere på å finne de ekte divisorene. Fyll inn i skjelettkoden under for å skrive et program som finner de ekte divisorene til et tall.

```
n = int(input("Hvilket tall skal jeg sjekke? ")

for d in range(...):
    if ...:
        print(...)
```

b) Sjekk om koden finner de riktige divisorene for tallet 12.

c) Nå vil vi endre koden, så den istedenfor å skrive ut alle divisorene, regner ut summen av dem. Til slutt sjekker vi om summen av divisorene er lik det originale tallet eller ei. Fyll inn koden for å få til dette.

```
n = int(input("Hvilket tall skal jeg sjekke? "))
divisorsum = 0

for d in range(...):
```

```
    if ...:
        ...

if ...:
    print(f"{n} er et perfekt tall!")
else:
    print(f"{n} er ikke et perfekt tall.")
```

d) Sjekk at koden din er enig i at 6 er perfekt, mens 12 ikke er det.

e) Hvilket at av disse tre tallene er perfekte? 4404, 8128, eller 9369?

Løsning oppgave 11 *Sjekke om et tall er perfekt*

a)

```
n = int(input("Hvilket tall skal jeg sjekke? "))

for d in range(1, n):
    if not n % d:
        print(d)
```

c)

```
n = int(input("Hvilket tall skal jeg sjekke? "
    ))

divisorsum = 0

for d in range(1, n):
    if not n % d:
        divisorsum += d

if divisorsum == n:
    print(f"{n} er et perfekt tall!")
else:
    print(f"{n} er ikke et perfekt tall.")
```

e) 8128.

Oppgave 12 *Finne perfekte tall*

I forrige oppgave laget vi et program som kunne sjekke om et tall var perfekt eller ikke. La oss nå utvide det til et program som automatisk finner perfekte tall. For å finne perfekte tall kan vi bruke en løkke som starter med å sjekke tallet 1, og så tallet 2, og så videre. Sånn kan vi fortsette til vi har funnet så mange perfekte tall vi ønsker.

a) La oss begynne med å prøve å finne de tre første perfekte tallene. Fyll inn i skjelettkoden under for å få til dette

```
n = 1
antall_funnet = 0

while ...:
    # Regn ut divisorsum av n
    divisorsum = 0
    for d in range(...):
        if ...:
        divisorsum += d

    # Sjekk om n er et perfekt tall
    if ...:
        antall_funnet += 1
        print(f"{n} er et perfekt tall", flush=
           True)
```

b) Hva er de tre første perfekte tallene?

c) Prøv å finne de fire første perfekte tallene med koden din, hvor lang tid tar det?

d) Om du nå prøver å finne de fem første perfekte tallene merker du kanskje at det fort tar veldig lang tid. Kan du tenke deg hvorfor?

e) Det femte perfekte tallet viser seg å være 33550336. Hvorfor tror du det tar så lang tid for programmet å finne dette tallet?

Løsning oppgave 12 *Finne perfekte tall*

a)

```
n = 1
antall_funnet = 0

while antall_funnet < 5:
    # Regn ut divisorsum av n
    divisorsum = 0
    for d in range(1,n):
        if n%d==0:
            divisorsum += d

    # Sjekk om n er et perfekt tall
    if n == divisorsum:
        antall_funnet += 1
        print(f"{n} er et perfekt tall", flush=
            True)
    n+=1
```

b) `6 er et perfekt tall 28 er et perfekt tall 496 er et perfekt tall`

c) Dette vil ta opptil 1-2 minutter. Tallet er 8128.

d) Hovedgrunnen til at dette tar såpass lang tid er at vi for hvert eneste tall deler det på alle tall som er mindre enn det tallet. Etterhvert som vi sjekker større og større tall blir antall operasjoner vi gjør også mye større. Kort sagt tar det lang tid fordi vi må gjøre ekstremt mange regneoperasjoner.

e) Samme som ovenfor.