Flere oppgaver

Her finner du flere relevante oppgaver – for alle de ulike temaene. Dersom du blir fort ferdig, kan du jobbe videre med disse, eller du kan bruke de for å øve deg mer på programmering i ettertid. Oppgavene her er ofte litt mer utfordrende enn de andre oppgavene vi har gitt i kurset, men vi håper at de også er mer interessante.

God koding!

1 Variabler og regning

Oppgave 1 *Regne mellom SI-enheter*

En millimeter er 0.01 centimeter. En mikrometer er 0.001 millimeter. En centimeter er 10 000 mikrometer.

Lag en variabel med din høyde i cm, og lag en ny variabel som gjør denne høyden til mm. Lag enda en ny variabel som gjør høyden i mm til µm. Til slutt, lag en variabel som på ny definerer din høyde i cm, men regnet fra µm. Print denne siste variabelen, har du regnet rett og fått riktig høyde i cm?

Løsning oppgave 1 *Regne mellom SI-enheter*

```
1 din_høyde_cm = 165
2 din_høyde_mm = din_høyde_cm * 10
3 din_høyde_um = din_høyde_mm * 10**3
4 din_høyde_nycm = din_høyde_um / 10**4
5
6 print(din_høyde_nycm)
```

Oppgave 2 *Konvertering av temperatur*

I Norge oppgir vi temperaturer i målestokken *Celsius*, men i USA bruker de ofte målestokken *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 ved 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 ved denne temperaturen?

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.

c) 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.

e) Bruk programmet ditt til å finne frysepunktet og kokepunktet til vann i Fahrenheit målestokken.

Løsning oppgave 2 *Konvertering av temperatur*

a)

```
fahrenheit = float(input('Fahrenheit: '))
celcius = (5/9)*(fahrenheit-32)

print(f'{fahrenheit} grader fahrenheit
   tilsvarer {celcius:.0f} celcius')
```

b)

```
Fahrenheit: 350
350.0 grader fahrenheit tilsvarer 177 celcius
```

177 ° C virker som en rimelig kakebaketemperatur

c)

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

d)

```
celcius = float(input('Celcius: '))
fahrenheit = (9/5)*celcius + 32

print(f'{celcius} grader celcius tilsvarer {
   fahrenheit:.0f} fahrenheit')
```

e)

```
Celcius: 0
0.0 grader celcius tilsvarer 32 fahrenheit
```

```
Celcius: 100
100.0 grader celcius tilsvarer 212 fahrenheit
```

Oppgave 3 *Jordkloden*

I denne oppgaven skal vi øve på å bruke Python som kalkulator, ved å regne litt på jordkloden. Husk at formelen for volumet av en kule er

$$V = \frac{4}{3}\pi R^3.$$

a) Jordkloden er tilnærmet en perfekt kule, og har en radius på 6371 km. Lag et kort program som først definerer en variabel `radius`, og deretter regner ut en variabel `volum`. Skriv til slutt ut svaret til brukeren med `print()`-funksjonen. La svaret være i km^3.

b) Endre programmet ditt så svaret istedet skrives ut i antall liter.

c) Den totale massen til jordkloden er omtrent $M = 5.972 \cdot 10^{24}$ kg. Regn ut hvor mange kg hver liter av jordkloden veier i gjennomsnitt. Virker svaret ditt rimelig?

Løsning oppgave 3 *Jordkloden*

a)

```
radiuskm = 6371
volumkm = (4/3)*3.14*(radiuskm**3)
print("Volumet til jorden er", volumkm, "
   kubikkkilometer.")
```

b)

```
radiusdm = 6371 * 10**4
volumdm = (4/3)*3.14*(radiusdm**3)
print("Volumet til jorden er", volumdm, "liter.")
```

c)

```
massejord = 5.972 * 10**24
vektperliter = massejord/volumdm
print("I gjennomsnitt veier hver liter av jorda",
   vektperliter, "kg.")
```

Oppgave 4 *Tolke feilmedlinger*

I denne oppgaven blir du presentert for noen av de vanligste feilmeldingene man kan få når man programmerer. Her trenger du ikke programmere noe, men les meldingen nøye, og beskriv hva du tror kan føre til en slik feilmelding. Finn også ut hvilken linje feilen er på.

a)

```
Traceback (most recent call last):
  File "test.py", line 6, in <module>
    print(f'Jeg er {alder} år gammel!')
NameError: name 'alder' is not defined
```

b)

```
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    sum = tall1 + tall2
TypeError: unsupported operand type(s) for +: 'int
   ' and 'str'
```

c)

```
  File "test.py", line 3
    navn = 'Jonas
                ^
```

```
SyntaxError: EOL while scanning string literal
```

Løsning oppgave 4 *Tolke feilmedlinger*

a) Her prøver vi å bruke variabelen `alder`, men den er ikke definert. Feilen er på linje 6.

b) Vi prøver å plusse sammen en variabel med type `int` og en med type `str`. Feilen er på linje 4.

c) Her slutter teksten uten at den har fått en avsluttende fnutt på høyre side. Feilen er på linje 3.

Betingelser

Oppgave 5 `and` *og* `or`

I denne oppgaven skal vi se på hvordan `and`- og `or`-operatorene fungerer og hvordan de brukes i betingelser.

a) Lag en variabel som inneholder tallet 5. Bruk en betingelse til å teste om tallet er mindre enn 10, og print isåfall ut en melding.

b) Bruk nøkkelordet `and` til å teste om tallet er mindre enn 10 og større enn 2.

c) Bruk nøkkelordet `or` til å teste om tallet er mindre enn 6 eller større enn 10.

Løsning oppgave 5 `and` *og* `or`

a)

```
1 tall = 5
2 if tall < 10:
```

```
    print(f"Tallet {tall} er mindre enn 10.")
```

```
tall = 5
if tall < 10 and tall > 2:
    print(f"Tallet {tall} er mindre enn 10 og stø
        rre enn 2.")
```

b)

```
tall = 5
if tall > 10 or tall < 6:
    print(f"Tallet {tall} er mindre enn 6 eller st
        ørre enn 10.")
```

Oppgave 6 *ABC-formelen*

ABC-formelen for å løse annengradsformler er som følger:

La a, b og c være reelle tall, hvor $a \neq 0$. Da har likningen $ax^2 + bx + c = 0$ løsningene

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

a) Lag et progam som finner løsningene til en annengradslikning med $a = 1$, $b = 2.5$ og $c = 1$.

b) Modifiser programmet ditt så det spør brukeren om verdier for a, b og c og skriver ut de tilhørende løsningene.

c) Dersom $b^2 - 4ac < 0$ har den tilhørende annengradslikningen ingen reell løsning. Bruk en `if`-betingelse til å informere brukeren om at det ikke finnes noen reell løsning dersom $b^2 - 4ac < 0$

d) Dersom $b^2 - 4ac = 0$ har likningen kun en løsning. Bruk en `elif` til å sjekke om dette er tilfelle og i så tilfelle informere brukeren om at det kun er en løsning

Løsning oppgave 6 *ABC-formelen*

a)

```python
from math import sqrt

a = 1
b = 2.5
c = 1
løsning1 = (-b + sqrt(b**2 - 4*a*c))/(2*a)
løsning2 = (-b - sqrt(b**2 - 4*a*c))/(2*a)

print(f'Løsningene for likningen {a}x^2 + {b}x + {c} = er {løsning1} og {løsning2}')
```

b)

```python
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

løsning1 = (-b + sqrt(b**2 - 4*a*c))/(2*a)
løsning2 = (-b - sqrt(b**2 - 4*a*c))/(2*a)

print(f'Løsningene for likningen {a}x^2 + {b}x + {c} = er {løsning1} og {løsning2}')
```

c)

```python
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

rot_del = b**2 - 4*a*c

if rot_del < 0:
    print(f' {a}x^2 + {b}x + {c} = 0 har dessverre ingen reelle løsninger')
else:
```

```
    løsning1 = (-b + sqrt(rot_del))/(2*a)
    løsning2 = (-b - sqrt(rot_del))/(2*a)
    print(f'Løsningene for likningen {a}x^2 + {b}x
        + {c} = 0 er {løsning1} og {løsning2}')
```

d)

```
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

rot_del = b**2 - 4*a*c

if rot_del < 0:
    print(f' {a}x^2 + {b}x + {c} = 0 har dessverre
        ingen reelle løsninger')
elif rot_del == 0:
    løsning = -b/(2*a)
    print(f'Løsningen for likningen {a}x^2 + {b}x
       + {c} = 0 er {løsning}')
else:
    løsning1 = (-b + sqrt(rot_del))/(2*a)
    løsning2 = (-b - sqrt(rot_del))/(2*a)
    print(f'Løsningene for likningen {a}x^2 + {b}x
        + {c} = 0 er {løsning1} og {løsning2}')
```

Oppgave 7 *Spisse og butte trekanter*

En trekant kalles *spiss* dersom alle de tre vinklene til trekanten er spisse (altså $< 90°$), tilsvarende kalles trekanten *butt* dersom én av de tre vinkelene er butt (altså $> 90°$). En trekant vil alltid enten være spiss, rettvinklet, eller butt.

Dersom vi kaller de tre sidelengdene a, b og c og lar c være den lengste av

disse kan vi vise at

$$a^2 + b^2 \begin{cases} > c^2 & \text{for spiss trekant,} \\ = c^2 & \text{for rettvinklet trekant,} \\ < c^2 & \text{for butt trekant.} \end{cases} \tag{1}$$

For rettvinklede trekanter er dette bare Pytagoras' læresetning. Men vi ser altså at likheten i Pytagoras' endres til ulikheter for spisse og butte trekanten.

Du skal nå lage et program der brukeren oppgir de tre sidelengdene i trekanten a, b og c og programmet skal si om trekanten er spiss, butt eller rettvinklet.

a) Forklar skjelettkoden under og fyll inn de bitene som mangler

```
1  print("Skriv inn de tre sidelengdene a, b og c der
       c skal være den lengste.")
2  a = float(input("Sidelengde a: "))
3  b = float(input("Sidelengde b: "))
4  c = float(input("Sidelengde c: "))
5
6  if c < a or c < b:
7      raise ValueError("c må være den lengste
           sidelengden.")
8
9  if ...:
10     print(...)
11
12 elif ...:
13     print(...)
14
15 elif ...:
16     print(...)
```

b) Vi gir nå sidelengdene til fem ulike trekanter. Hvilke to er spisse, hvilke to er butte og hvilken er rettvinklet?

- 4, 6, 8
- 3, 5, 5
- 5, 12, 13

- 6, 6, 8
- 9, 13, 17

Oppgave 8 *Hvilken skala?*

Et hus kan ha lengde 1 314 cm – eller, muligens på en mer hensiktsmessig skala, omtrent 13 meter. En celle kan være 0.000054 m lang – altså 54 μm. En fotballbane kan ha et areal på 6443399361 mm^2 – eller ca 6405 m^2.

I denne oppgaven skal vi, litt trinn for trinn, bygge opp et program for å konvertere *lengdemåleneheter* til en hensiktsmessig skala. Deretter vil vi gjerne utfordre dere på hvordan dere vil tenke for å utvikle et lignende program for å konvertere areal eller volum på lignende måte.

Vi anser at en målenhet er «hensiktsmessig» å bruke dersom målet kan oppgis med et tall større enn 0 foran kommaet, og at vi bruker størst mulig skala der dette er mulig. For å sjekke om man får et positivt tall foran kommaet (dvs. punktumet i Python) kan man konvertere tallet til et heltall ved hjelp av `int` og så sjekke om dette blir større enn 0.

a) Lag et Python-program som spør en bruker om hvor langt noe er. Lagre dette i en variabel. Husk å konvertere lengden til et desimaltall (ved hjelp av `float()`.

b) Spør deretter brukeren om hvilken målenhet dette er i. Her bør dere opplyse om hvilke mulige enheter det kan være – start med «m, cm eller mm».

c) Bruk betingelser, basert på målenheten brukeren gir, til å konvertere lengden brukeren har oppgitt til meter. Skriv ut svaret midlertidig og prøv noen ganger med regning for å sjekke at du har programmert dette riktig.

d) Regn ut hva lengden også blir i de andre enhetene (ut fra lengden i meter), og lagre disse i nye variable.

e) Bruk en betingelse for å først sjekke om det er hensiktsmessig å oppgi enheten i den største enheten (altså m). Hvis ja, skriv ut hva det blir

i meter – hvis ikke, sjekk om det er hensiktsmessig å oppgi den i den neste (cm), osv. Hvis det hverken er hensiktsmessig å oppgi målenheten i m eller cm, skriv ut svaret i den siste mulige (mm).

f) Skriv ut svaret på en fin måte. Bestem dere for antall desimaler dere vil oppgi i svaret. Dere kan formatere tallet ved å skrive et kolon, et punktum, antall desimaler og *f* etter variabelen i utskriften, altså f.eks. `print(f"... {lengde_meter:.2f} ...")`.

g) Utvid programmet deres til å også akseptere km og μm som mulige enheter.

h) Finn på fem objekter med lengde, i forskjellige størrelsesordener, og prøv om programmet gir forventet resultat for disse.

i) Nå har dere skrevet et program som kan konvertere ulike lengder til en skala som (muligens) er mer hensiktsmessig for det man vil måle. Hvordan ville dere utviklet et tilsvarende program for konvertering av areal? Av volum? Hva ville vært likt som i dette programmet, og hva ville vært annerledes? Hva ville vært de matematiske utfordringene?

Diskuter spørsmålene over med en kollega eller i en gruppe. Det er også mulig å programmere dette for å utfordre seg selv – eller kanskje elevene?

Løsning oppgave 8 *Hvilken skala?*

a)

```
1 lengde = float(input("Hva er lengden til objektet?
    "))
```

b)

```
1 målenhet = input("Hva er målenheten? m, cm eller
    mm?")
```

c)

```
1 if målenhet=="m":
2     lengde_meter = lengde
```

```
elif målenhet=="cm":
    lengde_meter = lengde/100
elif målenhet=="mm":
    lengde_meter = lengde/1000
else:
    print("Ukjent målenhet! Velg enten m, cm eller
        mm.")
```

d)

```
lengde_cm = lengde_meter*100
lengde_mm = lengde_cm*10
```

e)

```
if int(lengde_meter) > 0:
    print(f"Objektet er {lengde_meter:.2f} m langt
        .")
elif int(lengde_cm) > 0:
    print(f"Objektet er {lengde_cm:.2f} cm langt."
        )
else:
    print(f"Objektet er {lengde_mm:.2f} mm langt."
        )
```

f) Hele programmet ser nå slik ut:

```
lengde = float(input("Hva er lengden til objektet?
    "))
målenhet = input("Hva er målenheten? km, m, cm, mm
    eller um?")

if målenhet=="km":
    lengde_meter = lengde*1000
elif målenhet=="m":
    lengde_meter = lengde
elif målenhet=="cm":
    lengde_meter = lengde/100
elif målenhet=="mm":
    lengde_meter = lengde/1000
```

```
elif målenhet=="um":
    lengde_meter = lengde/1000000
else:
    print("Ukjent målenhet! Velg enten km, m, cm,
       mm eller um.")

lengde_km = lengde_meter/1000
lengde_cm = lengde_meter*100
lengde_mm = lengde_cm*10
lengde_um = lengde_mm*1000

if int(lengde_km) > 0:
    print(f"Objektet er {lengde_km:.2f} km langt."
       )
elif int(lengde_meter) > 0:
    print(f"Objektet er {lengde_meter:.2f} m langt
       .")
elif int(lengde_cm) > 0:
    print(f"Objektet er {lengde_cm:.2f} cm langt."
       )
elif int(lengde_mm) > 0:
    print(f"Objektet er {lengde_mm:.2f} mm langt."
       )
else:
    print(f"Objektet er {lengde_um:.2f} um langt."
       )
```

g) Vi forsøkte med 0.000001 km, 1000 m, 0.05 cm, 1500 mm og 20000 um, og fikk til svar (med to desimaler) 1.00 um, 1.00 km, 500 um, 1.5 m og 2.00 cm.

Når man tester slik bør man skrive inn tall slik at man kommer inn i alle betingelsene (alle if-testene), slik at mulige feil i alle disse kan fanges opp. Man vil helst teste både for programmeringsfeil og logiske feil.

For-løkker

Oppgave 9 *Trekanttall*

Et trekanttall er summen av tallrekken

$$1, 2, \ldots, n.$$

For eksempel så er

$$1 + 2 + 3 + 4 + 5 = 15,$$

så da er 15 et trekanttall. Siden det var summen av de 5 første tallene i tallrekka, så sier vi at det er det *femte* trekanttallet.

La oss si at vi vil vite hva det hundrede trekanttallet er, da må vi legge sammen alle tallene fra 1 til 100. Det blir fort slitsomt og kjedelig å gjøre for hånd. Så la oss bruke programmering.

For å gjøre det lettere å sjekke om programmet vårt, så startet vi med å prøve å regne ut summen fra 1 til 5.

a) Lag en løkke som skriver ut tallene

$$1, 2, 3, 4 \text{ og } 5$$

til skjermen.

b) Endre nå løkka så du isteden finner summen av tallene 1 til 5. Da må du først lage en variabel utenfor løkka, og for hvert tall, legge det til variabelen din. Husk at du kan legge noe til en varibel med `+=`.

c) Sammenlign svaret programmet ditt gir med det vi fant for hånd. Er de to like? Hvis de ikke er det så er det noe galt!

d) Hvis programmet ditt fungerte som forventet kan du nå endre sånn at du regner summen av de første 100 tallene

$$1 + 2 + 3 + \ldots + 100.$$

Det er en kjent matematiker, Carl Friedrich Gauss, som fikk denne oppgaven av sin mattelærer når han gikk på skolen på 1700-tallet. Læreren tenkte nok at dette skulle holde Gauss opptatt en god stund med å legge sammen tall etter tall. Gauss hadde ikke tilgang til en datamaskin, så

han kunne ikke automatisere jobben slik vi har gjort, men ha la merke til et mønster i tallene. Gauss la merke til at om vi starter på begge endene av rekka får vi et mønster

$$1 + 100 = 101, 2 + 99 = 101, 3 + 98 = 101, \ldots 50 + 51 = 101.$$

Fra dette mønsteret klarte Gauss å finne en formel for summen av tallene fra 1 til n, og uttrykket hans var

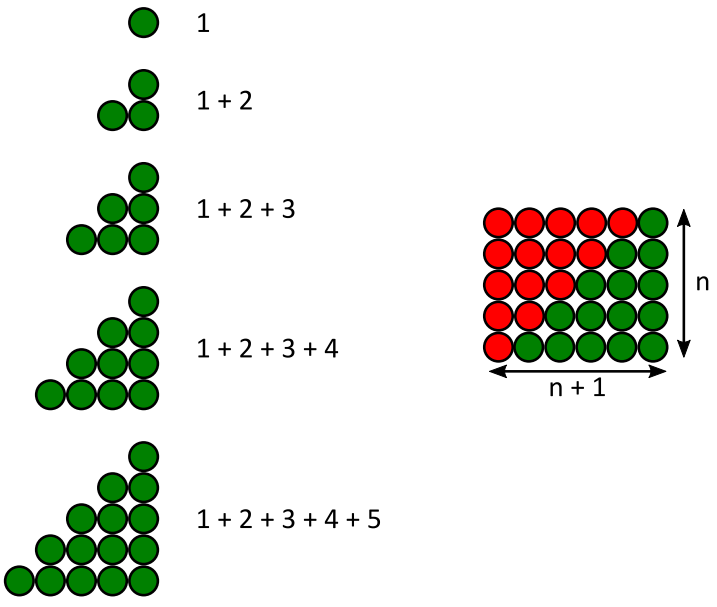
$$T_n = \frac{n(n+1)}{2}.$$

---

e) Bruk formelen til Gauss og sjekk at du får samme svar som programmet ditt for $n = 100$.

f) Gjør det samme for $n = 1000$, så $n = 1000000$ (én million).

For å skjønne hvorfor denne formelen er som den er, så kan det lønne seg å skjønne hvorfor de kalles trekanttall. Om vi tegner opp summene som antall baller, og tenger først 1, så 2, og så 3 bortover, sånn som dette:



Så ser vi at de ulike summene blir trekanter. Vi kan så gjøre om en slik trekant til en firkant ved å legge på like mange nye. Sånn som vist på høyre side av figuren. Denne firkanten har n baller i høyden, og $n+1$ baller bortover. Da er antall baller i hele firkanten $n(n+1)$. Men vi har jo doblet antall baller for å få en firkant, så om vi bare skal telle de grønne ballene må vi dele på to.

---

g) Hvordan kan kan du sjekke om et tall er et trekanttall? Skriv et program som sjekker om et heltall er et trekanttall, og hvis ja, skriver ut faktorene i dette. For eksempel bør programmet skrive ut `1 + 2 + 3 + 4 + 5` hvis det blir gitt input `15`.

Løsning oppgave 9 *Trekanttall*

a) Dette gjør vi med en `for`-løkke, og `range`-funksjonen:

```
1 for tall in range(1, 6):
2     print(tall)
```

Husk at `range` er fra-og-med, til (men ikke med), derfor skriver vi 1-6, for å få tallene 1, 2, 3, 4, og 5.

b)

```
1 total = 0
2
3 for tall in range(1, 6):
4     total += tall
5
6 print(total)
```

Her må vi både opprette `total` før løkka, og printe den ut etter løkka. Vi ser hvilke kodelinjer som hører til løkka fordi de har fått innrykk. I tilegg har vi lagt til blanke linjer for å skille dem litta fra løkka, men merk at dette er frivillig.

c) Svaret blir 15, som forventet.

d) Vi endrer programmet ved å endre hva løkka går til. For å gå opp til og

med hundre må vi skrive `range(1, 101)`. For å gjøre programmet vårt lettere å endre velger vi derimot å lage n som en variabel:

```
n = 100

total = 0
for tall in range(1, n+1):
    total += tall

print(total)
```

Det er nå rett-frem å endre programmet, bare ved å endre den første linja.

Svaret blir 5050

e) Vi fikk 5050 for $n = 100$, formelen gir

$$\frac{n(n+1)}{2} = \frac{100 \cdot 101}{2} = 5050.$$

Som altså er det samme, programmet vårt ser ut til å fungere.

f) For $n = 1000$ gir programmet vårt oss 500500. Formelen gir oss det samme. For $n = 1000000$ gir programmet vårt oss 500000500000, og formelen gir igjen det samme.

Oppgave 10 *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 10 *Kjøpe telefon på kreditt*

a)

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

b)

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

c)

```
lån = originalpris
```

d)

```python
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 11 *Tverrsum*

Tverrsummen av et tall er tallet du får dersom du plusser alle sifrene i tallet med hverandre.

a) Hvor mange 3-sifrede tall finnes det som har tverrsum 5? Løs for hånd.

b) Skriv et program som finner alle 3-sifrede tall med tverrsum 4. Start med å skrive en løkke for alle tall fra 1 (hvorfor fra 1?) til 9; i denne løkken lager du enda en løkke; og i denne løkken trenger du enda en. I den innerste løkken trenger du en betingelse, og til slutt en `print`.

c) Legg inn en teller i programmet ditt, som først er 0, og deretter øker hver gang du finner et tall med tverrsum 4. Hvor mange tall er det tilsammen? Stemmer det med det du fikk når du regnet for hånd?

d) Tror du det er flest 3-sifrede tall med tverrsum 4, eller flest 4-sifrede tall med tverrsum 3 – eller er det like mange? Utvid programmet ditt til å telle begge deler, og sjekk om du hadde rett.

Løsning oppgave 11 *Tverrsum*

a) Vi kan løse dette ved å gå frem systematisk. Vi kan starte med 103, deretter finne 112, osv.; til slutt får vi tallrekken *103, 112, 121, 130, 202, 211, 220, 301, 310, 400*. Dette utgjør 10 tall.

b)

```
tverrsum = 4

for i in range(1, 9):
  for j in range(9):
    for k in range(9):
        if (i + j + k) == tverrsum:
          print(f"{i}{j}{k}")
```

c) Tar bort print for å få en mer oversiktlig output:

```

teller = 0
tverrsum = 4

for i in range(1, 9):
  for j in range(9):
    for k in range(9):
        if (i + j + k) == tverrsum:
          teller += 1

print(f"Det er {teller} 3-sifrede tall med
   tverrsum {tverrsum}.")
```

d)

```

teller = 0
tverrsum = 4

for i in range(1, 9):
  for j in range(9):
    for k in range(9):
```

```
8           if (i + j + k) == tverrsum:
9             print(f"{i}{j}{k}")
10            teller += 1
11
12  print(f"Det er {teller} 3-sifrede tall med
       tverrsum {tverrsum}.")
13
14  teller = 0
15  tverrsum = 3
16
17  for i in range(1, 9):
18    for j in range(9):
19      for k in range(9):
20        for m in range(9):
21          for n in range(9):
22            for o in range(9):
23              if (i + j + k + m + n + o) == tverrsum
                    :
24                teller += 1
25
26  print(f"Det er {teller} 4-sifrede tall med
       tverrsum {tverrsum}.")
```

e) Det er faktisk like mange. Tror du det samme gjelder for antall 3-sifrede tall med tverrsum 5 / antall 5-sifrede tall med tverrsum 3?

Oppgave 12 *Fibonacci rekken*

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

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

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

$$a_n = a_{n-1} + a_{n-2}, \tag{3}$$

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

En interessant egenskap ved Fibonacci rekken er at forholdet mellom to etter-

følgende tall i følgen går mot det gyldne snitt, $\phi = \frac{1+\sqrt{5}}{2} \approx 1.62$. Det vil si at

$$\frac{a_n}{a_{n-1}} \to \phi. \tag{4}$$

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 12 *Fibonacci rekken*

a)

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

b)

```
1 print(forrige_tall)
2 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)

for tallnummer in range(antall_tall - 2):
    nytt_fibonacci_tall = fibonacci_tall +
       forrige_tall
    forrige_tall = fibonacci_tall
    fibonacci_tall = nytt_fibonacci_tall
    print(fibonacci_tall)
```

e)

```
for tallnummer in range(antall_tall - 2):
    nytt_fibonacci_tall = fibonacci_tall +
       forrige_tall
    forrige_tall = fibonacci_tall
    fibonacci_tall = nytt_fibonacci_tall
    forhold = fibonacci_tall / forrige_tall
    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.

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

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

for tallnummer in range(antall_tall - 2):
    nytt_fibonacci_tall = fibonacci_tall + forrige_tall
    forrige_tall = fibonacci_tall
    fibonacci_tall = nytt_fibonacci_tall
```