Oppgaver – Dag 2

I dette heftet finner du oppgaver som hører til dag 2 av Kodeskolens programmeringskurs for lærere.

Tema for andre dag er *variabler, regning, input, tester og løkker*. 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. 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!

Variabler og regning

Oppgave 1 *Ditt første program*

Det er vanlig når man lærer et nytt programmeringspråk og lage et lite "Hei, verden!" program. Et slikt program er kort og enkelt og skal brukes for å teste at alt fungerer som det skal

a) Bruk `print` til å lage et program som skriver ut bedskjeden "Hei, Python!" til vinduet. Kjør progammet og pass på at det fungerer som det skal.

Oppgave 2 *Finn tre feil!*

Her følger det tre programmer som ikke fungerer. Skriv programmet inn på din egen maskin og kjør det, les feilmeldingen, og prøv å tolke den. Når du skjønner hva som er galt, rett opp feilen og kjør programmet.

a)

```
print(Hva heter du?)
```

b)

```
navn = 'Sahid'
print('hei' navn)
```

c)

```
x = '3'
z = x + 2
print('x + y =', z)
```

Oppgave 3 *Vinkelsum*

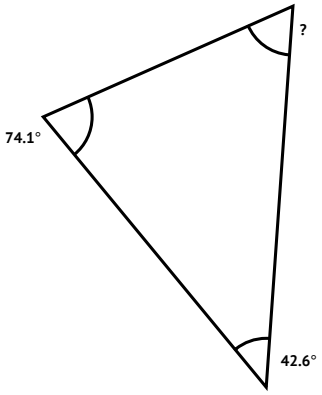
Vinkelsummen til en trekant er alltid 180°. I denne oppgaven skal du lage et program som regner ut den siste vinkelen, gitt at du kjenner til den første vinkelen.

a) Lag en variabel `første_vinkel` og sett den til å være 90°

b) Lag en variabel `andre_vinkel` og sett den til å være 60°

c) Regn ut hva den tredje vinkelen i trekanten må være og lagre svaret i en variabel `tredje_vinkel`

d) Bruk programmet ditt til å finne størrelsen på den ukjente vinkelen i figuren under:



Input

Oppgave 4 *Printing*

a) Lag et program som skriver ut teksten "Hei, Verden!", til skjermen.

b) Lag et program der du først lagrer navnet ditt i en variabel, og så få programmet ditt til å skrive ut en hilsen direkte til deg.

c) Hvis du bruker kommandoen `len()` på variabelen din får du ut antall bokstaver i navnet ditt. Endre programmet ditt så det også skriver ut denne informasjonen.

d) Husk at du kan bruke funksjonen `input()` til å stille brukeren et spørsmål. Bruk dette til å lage et program som spør brukeren om navnet dems, og deretter skriver ut en beskjed som bruker navnet de har oppgitt.

Oppgave 5 *Rabattkalkulator*

Noen ganger kan det være vanskelig å være sikker på hvor mye man skal betale for en vare på salg. Vi skal nå lage en rabattkalkulator.

a) Lag et program som først spør brukeren om: (1) originalprisen på en vare, og (2) hvor mange prosent rabatt vi får. Deretter skal programmet regne ut og skrive ut: (1) hva nyprisen er, og (2) hvor mange kroner rabatten utgjør.

b) Et par sko kostet originalt 750, men er satt ned 30%. Bruk programmet du skrev til å sjekke hvor mye du må betale for skoene nå.

c) Oppdater programmet ditt så det skriver ut resultatprisen med to desimaler

Oppgave 6 *Rabattkalkulator II*

I forrige oppgave regnet vi ut hvor mange kroner en viss rabatt utgjorde. La oss nå lage et program som istedet finner hvor mange prosent en rabatt vi egentlig får når en vare er satt ned.

a) Lag et program som spør brukeren om originalprisen på en vare, og så nyprisen. Deretter skal programmet regne og skrive ut hvor mange % rabatt prisendringen svarer til.

b) Prisen på en småis på kiosken er satt ned fra 25 kroner til 15 kroner, hele sommeren. Hvor mange % rabatt utgjør dette?

Oppgave 7 *Kvadrattall*

Kvadrattall er heltall som er blitt kvadrert, altså ganget med seg selv, eller opphøyet i annen. I Python kan du regne ut kvadratet av et tall `n` enten ved å skrive `n*n` eller `n**2`.

a) Skriv et program som spør brukeren om et tall, og deretter skriver ut kvadratet av tallet brukeren ga. Pass på at når vi spør om et tall må vi gjøre om svaret til fra brukeren til en tallvariabel ved å skrive `int` på følgende måte: `int(input())`. Dette er fordi "int" står for "integer", som er engelsk for heltall.

b) Bruk programmet ditt til å finne det minste tallet som har et kvadrat på over 1000 ved å prøve deg frem.

c) Import istedet kvadratrotfunksjonen `sqrt` fra `math` og bruk denne til å finne svaret.

Oppgave 8 *Brøkregning*

I denne oppgaven skal du undersøke brøken $\frac{7}{4}$

a) Begynn med å lage en variabel `teller` som du setter til verdien 7. Så lager du en variabel, `nevner` som du setter til verdien 4.

b) Oppdater programmet ditt så det skriver ut desimaltallverdien til brøken

c) Bruk heltalldivisjon (**Hint:** `//`) til å regne ut hvor mange ganger 7 går opp i 4 og skriv ut svaret i terminalen

d) Bruk modulo (**Hint:** `%`) til å regne ut hva resten blir når du deler 7 på 4. Skriv svaret ut i terminalen

e) Endre tallverdiene til heltall og nevner og kjør programmet på nytt.

Oppgave 9 *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

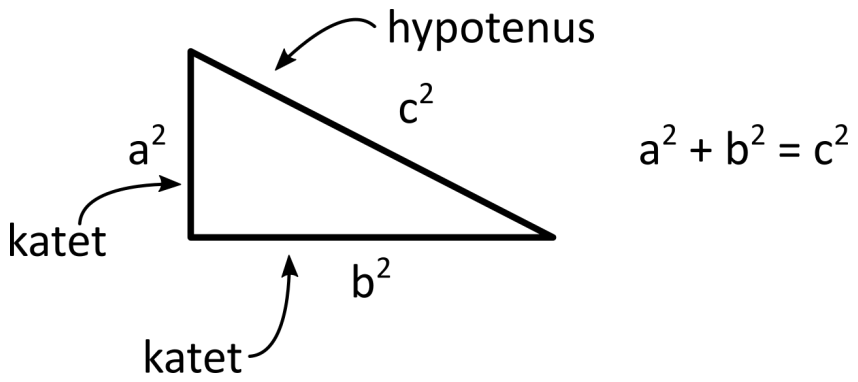
$$V = \frac{4}{3}\pi R^3.$$

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?

Oppgave 10 *Pytagoras*

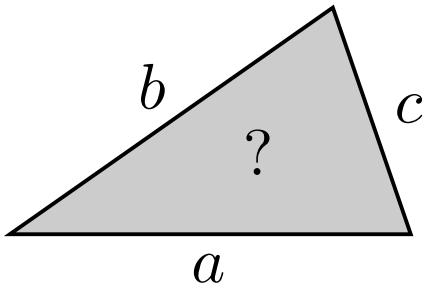


a) Skriv et program som spør brukeren om lengden på de to katetene i en rettvinklet trekant, og så skriver ut lengden på hypotenusen

b) Utvid programmet ditt, slik at det først spør brukeren om de vil finne en katet eller en hypotenus. Avhengig av hva brukeren svarer må du deretter spørre om lengden på de to kjente sidene og regne deg frem til den ukjente siden.

Oppgave 11 *Herons formel*

Herons formel er et matematisk uttrykk vi kan bruke for å finne arealet av en trekant, hvis vi kjenner lengdene på de tre sidene av trekanten. Herons formel fungerer for alle trekanter, de trenger ikke for eksempel å være rettvinklet.



For å bruke Herons formel må vi først regne ut et tall s, som er halve omkretsen til trekanten,

$$s = \frac{a+b+c}{2}.$$

Når vi har funnet s kan vi regne ut arealet til hele trekanten med formelen:

$$A = \sqrt{s(s-a)(s-b)(s-c)}.$$

a) Regn ut arealet av en trekant med sidelengder på 3, 4, og 5 for hånd, ved hjelp av Herons formel.

Vi skal nå skrive et program, som spør brukeren om lengdene på de tre sidene, og så regner ut arealet for oss.

b) Bruk `float(input(...))` for å spørre brukeren om lengden på de tre sidene av en trekant. Lagre svarene fra brukeren i tre variabler.

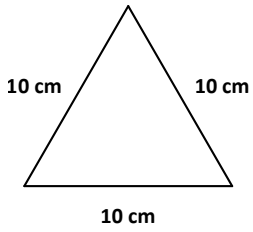
c) Regn så ut variabelen s, som var halve omkretsen til trekanten.

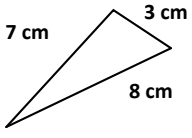
d) Regn så ut arealet A og skriv resultatet tilbake så brukeren kan lese det.

Om du tror du har klart å skrive hele programmet er det på tide å teste det

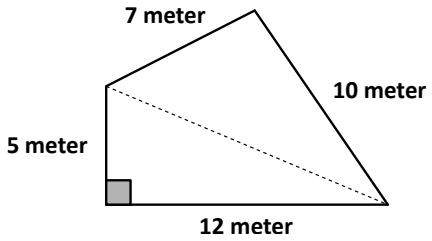
e) Bruk programmet ditt til å sjekke trekanten med sider på 3, 4, og 5. Får du samme svar som du gjorde for hånd?

f) Bruk programmet til å finne arealet av de to trekantene under.





g) En firkantet park er formet som figuren under. Klarer du å finne arealet av parken?


7 meter
10 meter
5 meter
12 meter

Lister

Oppgave 12 *Lister*

I denne oppgaven skal vi bli vant med lister. Start med å lage variabelen `min_liste =[3, 5, 3, 6, 8]`.

a) Hvordan henter du ut det første elementet fra `min_liste`?

b) Kan du hente ut det siste elementet fra `min_liste`?

c) Hva får du hvis du skriver `print(min_liste[3])`? Gjett først så prøv selv i Python.

d) Kan du bruke en funksjon for å måle hvor lang `min_liste` er?

e) Kan du legge til ett nytt tall, 2, på slutten av `min_liste`?

f) Kan du legge til ett nytt tall, 6, med indeks 3 i `min_liste`?

Oppgave 13 *Gjennomsnittshøyde*

I denne oppgaven skal du regne ut gjennomsnittshøyden til en skoleklasse. I klassen er det 10 elever, med høyden (i cm) 162, 151, 157, 163, 173, 172, 170, 154, 165 og 165.

a) Begynn med å lage en variabel `elevhøyde` som er en liste som inneholder høyden til alle elevene.

b) Skriv ut høyden til alle elevene (med `print` kommandoen)

c) Dobbeltsjekk at du skrev inn høyden til alle 10 elevene. Dette gjør du med å sjekke om lengden til lista er 10 (Hint: Husk `len` funksjonen). Lagre antallet elever i klassen i variabelen `antall_elever`

d) Bruk `sum` funksjonen til å sjekke hvor høy alle elevene i klassen ville vært om du stablet dem oppå hverandre. Lagre resultatet i en variabel `totalhøyde` og skriv ut denne variabelen.

e) Regn ut gjennomsnittshøyden ved å dele totalhøyden på antallet elever. Lagre resultatet i en variabel med navnet `gjennomsnittshøyde`.

f) Hvorfor er det lurt å lagre antallet elever i en egen variabel istedenfor å skrive den direkte inn når du regner ut gjennomsnittet?

Oppgave 14 *Månednummer med lister*

Vennen din sliter veldig med å huske hvilken måned som tilhører hvilket nummer og han ber deg om å lage et program som kan hjelpe han med å lære dette. Du skal derfor lage et program som tar et heltall in som input og printer ut hvilken måned det er. Her er et eksempel på hvordan programmet kan virke

```
Hvilket månednummer vil du vite navnet på? 5
Måned nummer 5 er Mai.
```

a) Start med å lage en variabel som heter `måneder`. Denne variabelen skal være en liste som inneholder alle månedsnavnene i rett rekkefølge.

b) Sjekk at du har husket alle månedene ved å skrive ut `måneder` og lengden til `måneder` på skjermen.

c) Lag en variabel `månednummer`. Be brukeren om hva denne skal være, men husk å gjør den om til heltall.

d) Skriv ut hvilken måned i `måneder` som har indeksen `månednummer`.

e) Mennesker teller jo fra 1, mens Python teller fra 0. Vi vil skrive ut månedene slik mennesker tenker, altså Januar er måned 1, ikke måned 0. For å få til dette må du trekke 1 fra `månednummer`. Gjør dette, skriv ut måneden som har indeksen `månednummer -1`.

f) Pynt litt på koden din slik at utskriften ser fin ut.

Tester

Oppgave 15 *Finn størst tall*

I denne oppgaven skal vi øve på `if`-tester ved å finne det største av to tall.

a) Be brukeren om å gi to tall og lagre dem i `tall1` og `tall2`

b) Dersom `tall1` er større enn `tall2` skal du gi bedskjed om at `tall1` er størst

c) Dersom `tall2` er større enn `tall2` skal du gi bedskjed om at `tall2` er størst

d) Dersom tallene er like skal brukeren få beskjed om det

Oppgave 16 *Vinkeltyper*

Vi kan dele vinkler inn i tre forskjellige typer:

1. En vinkel som er mindre enn 90 ° kalles en *spiss* vinkel.
2. En vinkel som er større enn 90 ° kalles en *stump* eller *butt* vinkel.
3. En vinkel som er akkurat 90 ° kalles en *rett* vinkel

Be brukeren om å gi en vinkel og bruk `if`, `elif` og `else` til å fortelle brukeren om vinkelen er *spiss*, *stump* eller *rett*

Løkker

Oppgave 17 *For-løkker for hånd*

For hver løkke, gå igjennom for hånd og forutsi hva som skrives ut. Etterpå kan du kjøre løkkene og se hva om du hadde rett:

a)

```
produkt = 0
for tall in range(1, 5):
    produkt *= tall

print(produkt)
```

b)

```
produkt = 1
for tall in range(1, 5):
    produkt *= tall

print(produkt)
```

c)

```
x = 1
for _ in range(10):
    x *= 2

print(x)
```

Oppgave 18 *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?

Oppgave 19 *While-løkker for hånd*

Gå igjennom løkkene under for hånd og forutsi hva de kommer til å skrive ut. Etterpå kan du kjøre dem og se om du hadde rett.

a)

```
i = 0
while i < 10:
    i += 2
    print(i)
```

b)

```
i = 2
while i < 10:
    i = i**2 + 1
    print(i)
```

c)

```
a = 0
b = 0
c = 0

while c < 10:
    a += 1
    b += a
    c += a + b

print(a)
print(b)
print(c)
```

Oppgave 20 *Brette ark*

Et vanlig ark er omtrent 0.1 mm tykt. Om vi bretter arket på midten dobbler vi tykkelsen av arket, så det er 0.2 mm tykt. Om vi bretter arket på nytt dobbler vi igjen tykkelsen, så det blir 0.4 mm tykt. Sånn kan vi fortsette å brette arket for å gjøre det tykkere. Om du prøver i praksis viser det seg nok fort at det er veldig vanskelig å brette arket noe særlig mer enn 6-7 ganger. Men om vi nå later som vi kunne brettet arket så mange ganger vi vil, er det ingen grense for hvor tykt arket kunne blitt.

Verdens høyeste byggning er Burk Khalifa i Dubai, som er 828 meter høyt. Vi

ønsker å finne ut hvor mange ganger vi må brette arket vårt, før det er like tykt som høyden av denne byggningen.

a) Diskuter med sidemannen hvordan dere kunne funnet ut av dette med penn og papir. Hva er ulempen med fremgangsmåten deres?

Vi skal nå løse problemet ved hjelp av et kort Pythonprogram. For å løse problemet bruker vi en løkke for å brette arket helt til vi har nådd den tykkelsen vi er ute etter.

b) Fyll inn skjelettkoden under for å finne antall brett vi trenger. Pass spesielt på at tykkelsen til arket er oppgitt i millimeter, mens byggningen er oppgitt i meter.

```
tykkelse = ...
antall_brett = ...

while ...:
    tykkelse *= ...
    antall_brett += ...

print(...)
```

c) Om du har klart å løse oppgaven. Finn antall brett som skal til før tykkelsen er like høyt som verdens høyeste fjell, Mount Everest, som er 8848 meter høyt.

d) Avstanden fra jorda til månen er ca 384400 km. Hvor mange ganger må arket brettes før det er like tykt som denne avstanden?

Oppgave 21 *Trekanttall*

Temaer: Løkker, lister 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

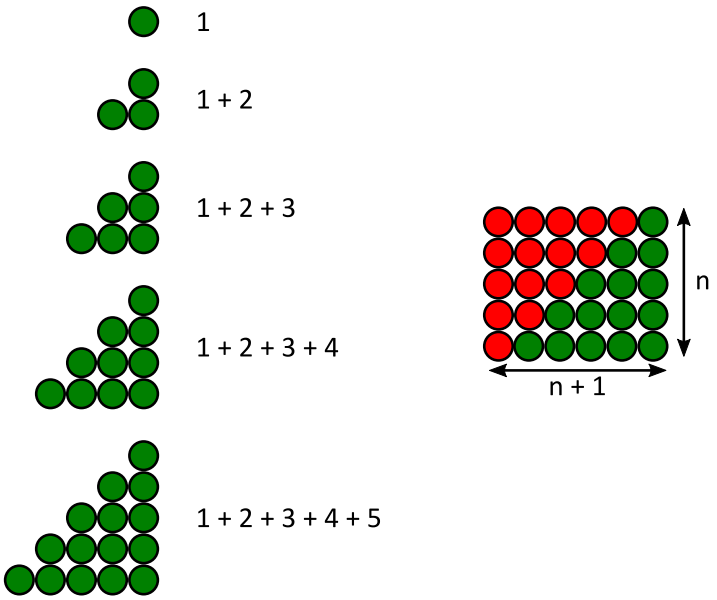
$$T_n = \frac{n(n+1)}{2}.$$

---

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`.

Mikrobit – litt av alt

Oppgave 22 *Programmering av Mikro:bit*

Finn frem eksempeloppgavene fra https://makecode.microbit.org igjen, samt https://python.microbit.org/. Nå skal vi programmere de igjen ved hjelp av Python. Finn frem oppgaven du lagde i går, eller finn frem tutorialen igjen, og lag følgende ved hjelp av Python:

a) Flashing Heart

b) Smiley Buttons

c) Dice

Oppgave 23 *Programmering av Mikro:bit – litt mer avansert*

Gå inn på https://makecode.microbit.org igjen, samt https://python.microbit.org/. Finn frem oppgaven du lagde i går, eller finn frem tutorialen igjen, og programmer følgende ved hjelp av Python:

a) Temperature

b) Rock Paper Scissors

c) Compass

Oppgave 24 *Mikro:bit – bildekavalkade*

I dette programmet skal vi få Mikro:bit-en til å vise frem ulike bilder. Husk at vi har mange bilder tilgjengeling, se https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html, eller man kan definere sine egne.

a) Lag et program der du lager en liste som inneholder ulike bilder. Hvis man trykker A skal Mikro:Bit-en vise det forrige bildet, og hvis man trykker B skal den vise det neste bildet. La Mikro:Bit-en starte med å vise frem det første bildet. Hva skal skje hvis man trykker seg til før det første bildet, eller etter det siste?

b) Endre programmet ditt til å hoppe til et tilfeldig bilde dersom man rister Mikro:bit-en.

Oppgave 25 *Mikro:bit – endre frekvens*

I denne oppgaven skal vi få Mikro:bit-en til å vise frem en slags animasjon ved å la den følge visse mønstre.

Ta utgangspunkt i følgende program, som viser frem en blinkende rombe:

```
from microbit import *

while True:
    for tall in range(10):
        figur = Image("00{i}00:0{i}0{i}0:{i}000{i}:0{i}0{i}0:00{i}00".format(i=tall))
        display.show(figur, delay=400)

    for tall in range(10):
        figur = Image("00{i}00:0{i}0{i}0:{i}000{i}:0{i}0{i}0:00{i}00".format(i=10-tall))
        display.show(figur, delay=400)
```

a) Endre programmet slik at det blinker med ulik frekvens. Prøv først *fort – sakte – fort – sakte ...* og deretter et mønster med varierende frekvens (f.eks. *sakte – middels – fort – sakte – middels – fort ...* og så enda flere nivåer. Bruk en for-loop for å iterere over de mulige alternativene.

b) Endre programmet slik at de ulike punktene har ulik intensitet. Kan du få til en rotasjon istedenfor at alle blinker med samme frekvens?