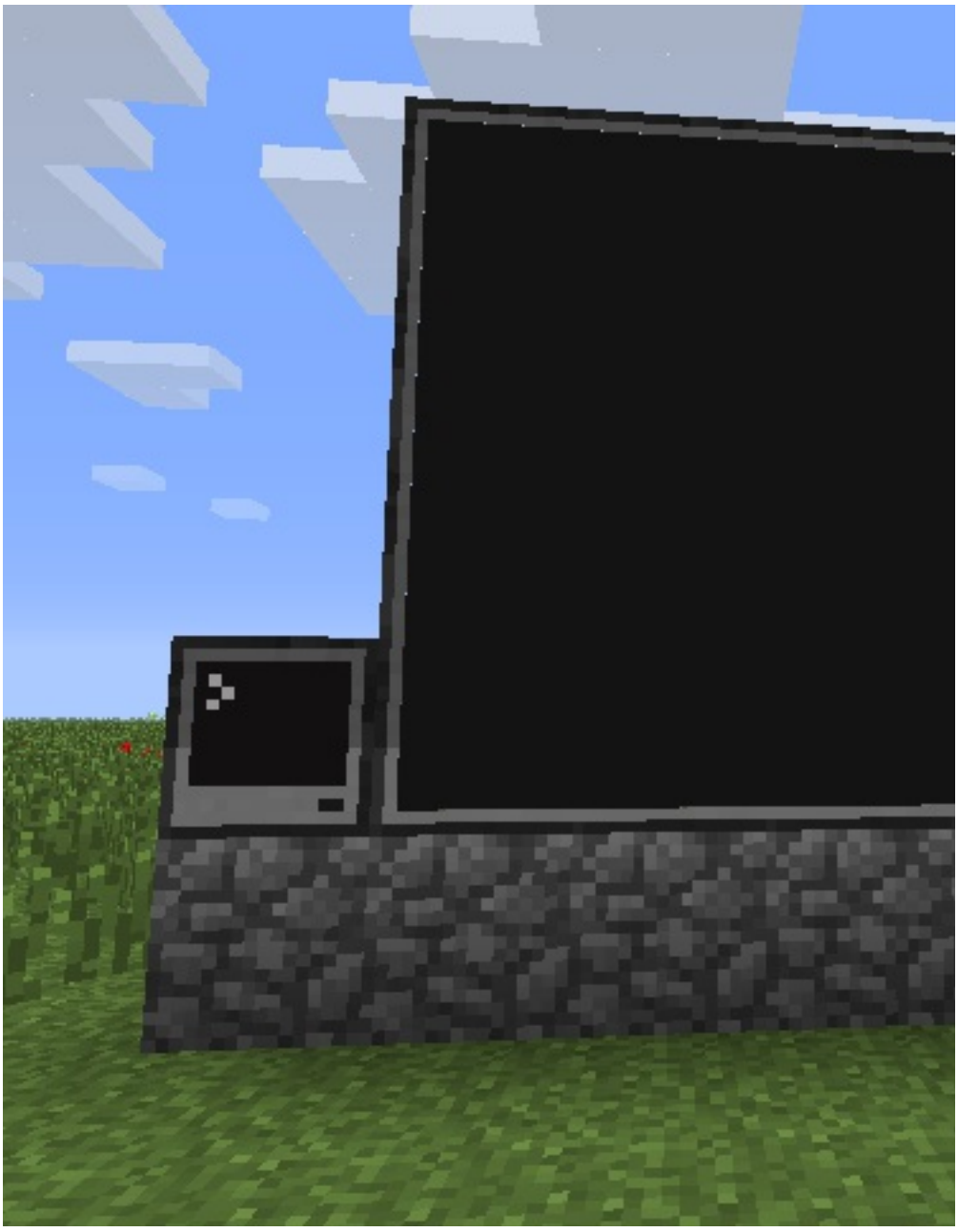
Sprettball

Introduksjon

Nå skal vi lære hvordan vi kan koble en skjerm til datamaskinen. Med kommunisere med verden rundt oss. Kanskje Minecraft-verden blir fyl

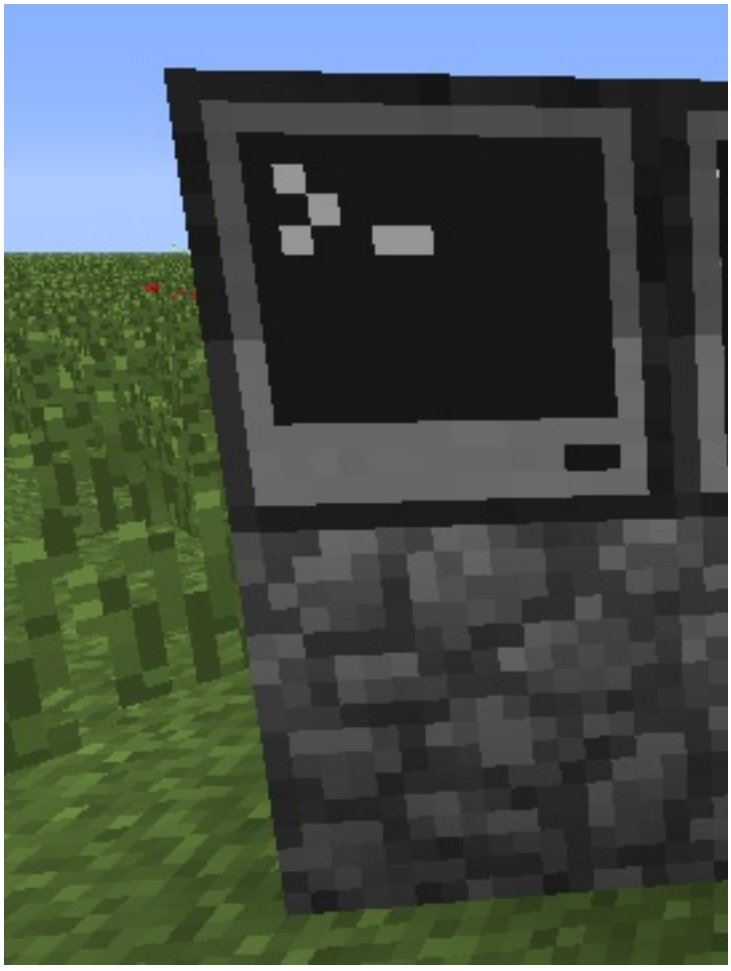


Steg 1: Koble en skjerm til

La oss som vanlig begynne helt enkelt. La oss koble en skjerm til en d

Sjekkliste

- [ ] Bygg en **Computer**. Du kan også bruke en **Advanced Comput**
- [ ] Bygg så en **Monitor** inntil datamaskinen du nettopp bygget, om



- [ ] Start datamaskinen. Den enkleste måten å skrive noe til skjerm
  skriver `help monitor` vil du kunne lese at `monitor` kan kjøre ar

  La oss for eksempel prøve programmet `hello`. Dette er et enke

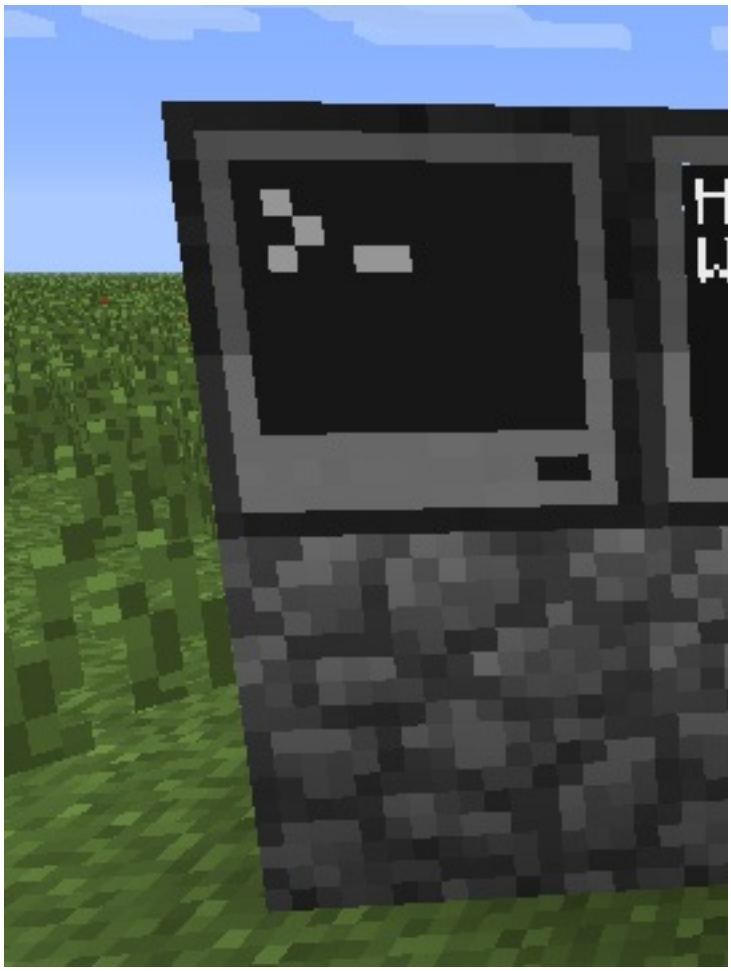
skriver bare teksten *Hello World!*. Kjør først programmet på data

```
> hello
Hello World!
```

Nå kan vi prøve å få dette til å kjøre på skjermen. Med skjermen
på bildet over) skriver vi bare

```
> monitor right hello
```

Du kan nå trykke `Esc` for å gå ut av datamaskinen og se på skj



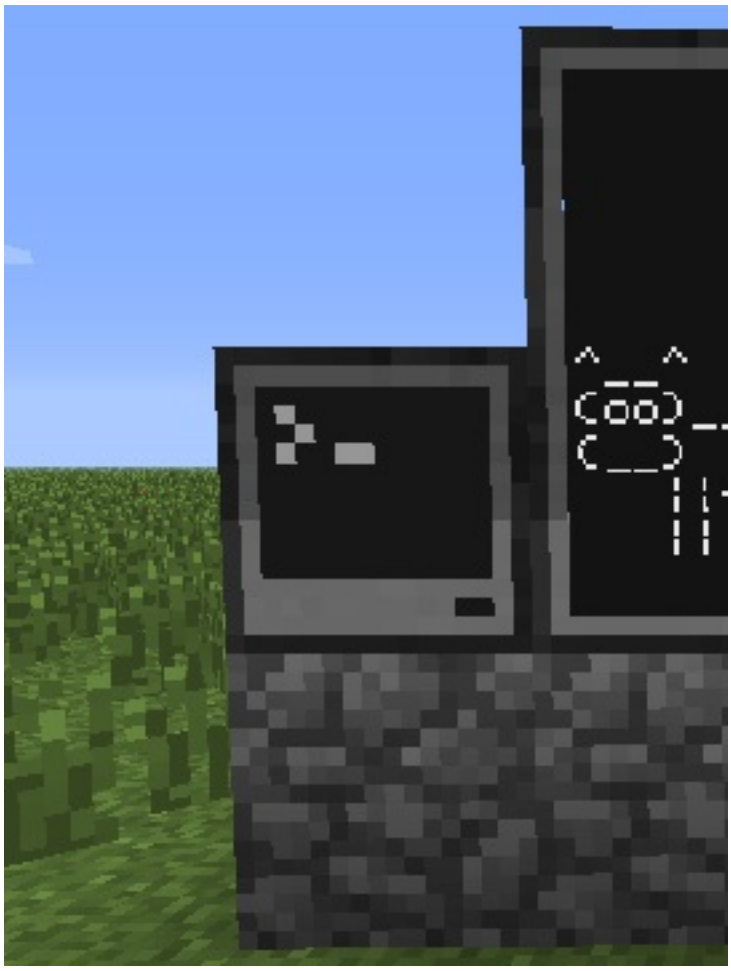
Kult! Da er vi igang med skjermene! Men - det ble jo veldig dårli
gjøre noe med. Om du setter en **Monitor** ved siden av den du a
større skjerm. På denne måten kan du lage en stor skjerm! Prøv

- [ ] Trikset med `monitor` fungerer selvsagt også med programmer
kaller `ku`:

```
print("^__^")
print("(oo)_______")
print("(__)       )")
print("   ||----W |")
print("   ||     ||")
```

Det er ikke så farlig om din ku ikke ser ut akkurat som her. Du k
at også vårt eget program kan skrives på skjemen.

- [ ] Nå kan vi kjøre programmet: `monitor right ku`.



Cowsay

Denne kuen er basert på et morsomt lite program som heter **Cows**
1990-tallet. En web-variant av programmet finnes nå på http://cow

Steg 2: En annen metode

I stedet for at vi bruker `monitor`-programmet for å skrive til skjermen
lager. Dette er ganske enkelt med et bibliotek som heter `peripheral`
skjermen er et tillegg til datamaskinen).

Sjekkliste

- [ ] Lag et enkelt program som heter `skjerm` og ser slik ut:

```
skjerm = peripheral.wrap("right")
skjerm.write("Heisann!")
```

Som vanlig kan du bytte ut *right* med for eksempel *left* eller *top*

- [ ] Kjør programmet ditt ved å skrive `skjerm`. Selv uten `monitor` s
datamaskinen.

- [ ] I stedet for `peripheral.wrap` kan vi bruke `peripheral.find`. D
datamaskinen skjermen er på. Endre programmet ditt som følge

```
local skjerm = peripheral.find("monitor")      -- endret
skjerm.write("Heisann!")
```

- [ ] Vi kan også sjekke om det er noen skjerm koblet til. Dette gjør v

```
local skjerm = peripheral.find("monitor")

if skjerm then                                 -- ny linj
    skjerm.write("Heisann!")
else                                           -- ny linj
    print("Ingen skjerm er koblet til")        -- ny linj
end                                            -- ny linj
```

Prøv å kjør dette nye programmet. Det kan hende du vil gjøre `m`
skjermen. Prøv også å koble fra (ødelegge) skjermen. Får du me
kjøre programmet ditt?

Steg 3: En ball faller over s

Vi skal nå begynne på en enkel animasjon som vi kan kjøre på skjerm

Sjekkliste

- [ ] Lag et nytt program som heter `sprettball`. Det begynner gans
vil kan du kopiere det ved å skrive `copy skjerm sprettball`. Pr

```
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.clear()
    skjerm.write("O")
else
    print("Ingen skjerm er koblet til")
end
```

Dette skal tegne en ball øverst på skjermen.

- [ ] Vi kan nå bruke en enkel *for*-løkke til å animere ballen. Endre ko

```
local skjerm = peripheral.find("monitor")

if skjerm then
    for rad = 1, 10 do                          -- ny linj
        skjerm.clear()
        skjerm.setCursorPos(3, rad)             -- ny linj
        skjerm.write("O")
        sleep(1)                                -- ny linj
    end                                         -- ny linj
else
    print("Ingen skjerm er koblet til")
end
```

For at vi skal rekke å se at ballen flytter seg har vi lagt inn en `s` sekund) mellom hver gang ballen flyttes.

- [ ] Dette er ikke så spennende enda, men før vi lager noe mer ava som gjør det enklere for oss å holde oversikten senere. Vi flytte funksjon `tegnBall`. Flytt kodelinjene rundt slik at de blir som d

```
function tegnBall(skjerm)                          -- ny linj
    for rad = 1, 10 do                             -- flytte
        skjerm.clear()                             -- flytte
        skjerm.setCursorPos(3, rad)                -- flytte
        skjerm.write("O")                          -- flytte
        sleep(1)                                   -- flytte
    end                                            -- flytte
end                                                -- ny linj

skjerm = peripheral.find("monitor")

if skjerm then
    tegnBall(skjerm)                               -- ny linj
else
    print("Ingen skjerm er koblet til")
end
```

Steg 4: Hvor stor er skjerm

En liten utfordring med skjermer er at de kan ha forskjellig størrelse. H hverandre slik at du får en veldig høy skjerm vil ikke ballen i `sprettba` Dette kan vi løse ved å bruke funksjonen `getSize`.

Sjekkliste

- [ ] Funksjonen `getSize` forteller oss hvor bred og hvor høy skjerm

over hele skjermen:

```
function tegnBall(skjerm)
    local bredde, hoyde = skjerm.getSize()     -- ny linj
    for rad = 1, hoyde do                      -- endret
        skjerm.clear()
        skjerm.setCursorPos(3, rad)
        skjerm.write("0")
        sleep(1)
    end
end
```

- [ ] Prøv å lag skjermen høyere eller lavere. Faller ballen hele veien

- [ ] Klarer du å sentrere ballen, slik at den faller nedover midt på sk

Steg 5: Sprettball

Nå vil vi få ballen til å oppføre seg mer som en sprettball. For å få til d
betegne hvor ballen er, mens `fartX` og `fartY` forteller hvor fort balle

Sjekkliste

- [ ] I tillegg til de nye variablene bytter vi ut *for*-løkken med en *whil*
Husk at du bruker `Ctrl-T` for å avslutte programmet.

```
function tegnBall(skjerm)
    local bredde, hoyde = skjerm.getSize()
    local X, Y = 1, 2                          -- ny linj
    local fartX, fartY = 1, 1                  -- ny linj

    while true do                              -- endret
        skjerm.clear()
```

```
        skjerm.setCursorPos(X, Y)                -- endret
        skjerm.write("O")
        sleep(1)

        X = X + fartX                            -- ny linj
        Y = Y + fartY                            -- ny linj
    end
end
```

Ballen vil nå bevege seg på skrå over skjermen. Ser du hvorfor?

Vi vil nå la ballen sprette når den treffer kanten. Dette gjør vi ve
Legg til et par tester nederst i funksjonen din:

```
function tegnBall(skjerm)
    local bredde, hoyde = skjerm.getSize()
    local X, Y = 1, 2
    local fartX, fartY = 1, 1

    while true do
        skjerm.clear()
        skjerm.setCursorPos(X, Y)
        skjerm.write("O")
        sleep(1)

        X = X + fartX
        Y = Y + fartY

        if X <= 1 or X >= bredde then            -- ny linj
            fartX = -fartX                       -- ny linj
        end                                      -- ny linj
        if Y <= 1 or Y >= hoyde then             -- ny linj
            fartY = -fartY                       -- ny linj
        end                                      -- ny linj
    end
end
```

Spretter ballen tilbake når den treffer kanten av skjermen? Lag

pausen mellom hver gang ballen flytter seg. For eksempel bytt

- [ ] Tilslutt kan vi lage en mer naturlig sprettball-bevegelse ved å ta gravitasjonen gjør at ballen faller stadig raskere ned mot bakke

```
function tegnBall(skjerm)
    local bredde, hoyde = skjerm.getSize()
    local X, Y = 1, 2
    local fartX, fartY = 1, 0                    -- endret
    local gravitasjon = 0.2                      -- ny linj

    while true do
        skjerm.clear()
        skjerm.setCursorPos(X, Y)
        skjerm.write("O")
        sleep(0.1)

        fartY = fartY + gravitasjon              -- ny linj
        X = X + fartX
        Y = Y + fartY

        if X <= 1 or X >= bredde then
            fartX = -fartX
        end
        if Y >= hoyde then                       -- endret
            fartY = -(fartY + gravitasjon)       -- endret
        end
    end
end
```

Vi endret også litt i sjekken om `Y` er slik at ballen skal sprette, s taket lengre.

Steg 6: Reklamebanner

Vi tar nå en liten pause fra sprettballen vår for å se på hvordan vi kan reklamebannere.

Sjekkliste

- [ ] Lag et nytt program som du kaller `reklame`. Vi begynner helt er

```
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.clear()
    skjerm.setCursorPos(1, 1)
    skjerm.write("ComputerCraft")
else
    print("Ingen skjerm er koblet til")
end
```

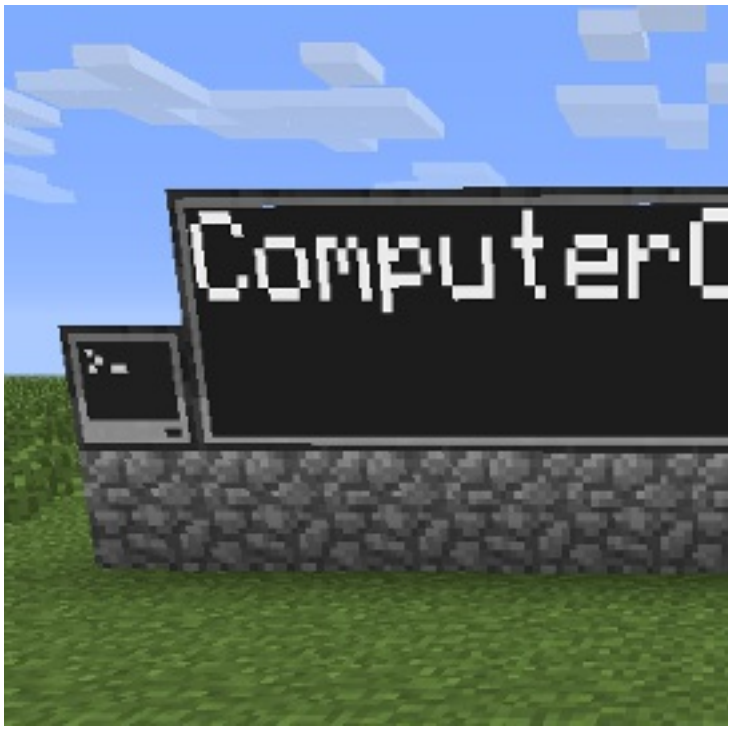
Kjør programmet. Skriver det til skjermen som det skal?

- [ ] Et problem hvis dette skal være et reklamebanner er at teksten skjermen for å kunne lese den.

For skjermer kan vi bruke `setTextScale` for å endre tekststørre

```
skjerm.setTextScale(3)
```

rett før linjen `skjerm.clear()`, og kjør programmet ditt på nytt.



Tallet `3` i `setTextScale(3)` indikererer størrelsen på teksten. H
største mulige teksten. Etter at vi har brukt `setTextScale` må v

- [ ] Prøv å endre verdien i `setTextScale`. Forandrer størrelsen på te
5? Eller mindre enn 0.5?

- [ ] Vi vil nå la programmet selv bestemme tekststørrelsen. Siden vi
teksten skal være så stor som mulig.

  En måte å gjøre dette på er å lage en løkke hvor vi tester alle te
  størrelse hvor all teksten får plass på skjermen.

  Legg inn denne funksjonen øverst i `reklame`-koden:

```
function brukStorTekst(skjerm, tekst)
    local lengde = #tekst

    for skala = 5, 0.5, -0.5 do
        skjerm.setTextScale(skala)
        skjerm.clear()
        bredde, hoyde = skjerm.getSize()
        if lengde <= bredde then
```

```
            break
        end
    end
end
```

Skjønner du hvordan denne koden fungerer? Vi bruker noen nye

Tegnet `#` brukes for å telle ting. For eksempel betyr `#tekst` ret
må vi vite når vi senere skal sjekke om skjermen er stor nok.

I *for*-løkken bruker vi tre tall i stedet for to som vanlig. Det siste
vi tar i løkken. Siden vi her ville telle ned fra 5 til 0.5 må vi bruk

Til slutt, `break` sier at vi vil avslutte *for*-løkken før den egentlig
tekststørrelser når vi finner en som passer. Tidligere har vi brukt
man fant skatten.

- [ ] Nå vil vi bruke denne funksjonen for å sette tekststørrelsen. Prø
kalles! Skal noen av de opprinnelige linjene slettes? Sjekk om p
skjermen (bygg eller ødelegg noen enkeltskjermer) og kjøre det

- [ ] En ting du kanskje ser er at du må skrive teksten som skal stå p
trenger den for å finne riktig størrelse på skjermen, og deretter
det lurt å i stedet lage en variabel. Med denne variabelen vil ko
slik ut:

```
local tekst = "ComputerCraft"
local skjerm = peripheral.find("monitor")

if skjerm then
    brukStorTekst(skjerm, tekst)
    skjerm.setCursorPos(1, 1)
    skjerm.write(tekst)
else
    print("Ingen skjerm er koblet til")
end
```

Prøv selv

Det er flere måter å gjøre `reklame`-programmet enda bedre på. He

Kan du midtstille teksten på skjermen? Du må da endre tallene i `se`
på skjermen.

Om du endrer størrelsen på skjermen blir teksten borte. Vi kan hell
når skjermen blir endret. Du kan bruke `os.pullEvent` til å lytte på
teksten på nytt når disse skjer.

Steg 7: En skikkelig ticker!

Dessverre er det en begrensning på hvor stor en skjerm kan være. Du
stor skjerm. Dette begrenser hvor lange tekster vi kan skrive, ihvertfa

Et alternativ for lengre tekster er å bruke en ticker, hvor vi animerer t

Sjekkliste

- [ ] Lag et nytt program som heter `ticker`. Vi begynner programm

```
local tekst = "Jeg er en lang tekst. Bytt meg gjerne ut!
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.setTextScale(5)
    skjerm.clear()

    skjerm.setCursorPos(1, 1)
    skjerm.write(tekst)
else
    print("Ingen skjerm er koblet til")
end
```

Her setter vi bare størrelsen på teksten fast til `5`, siden vi vil br
dette programmet?

I stedet for å prøve å skrive hele teksten til skjermen vil vi nå ba
vi igjen `skjerm.getSize()` for å finne ut hvor stor skjermen er.
biblioteket (*sub* betyr i denne sammenhengen *del*, den brukes t

```
local tekst = "Jeg er en lang tekst. Bytt meg gjerne ut!
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.setTextScale(5)
    skjerm.clear()
    local bredde, hoyde = skjerm.getSize()          -- ny

    local deltekst = string.sub(tekst, 1, bredde)   -- ny
    skjerm.setCursorPos(1, 1)
    skjerm.write(deltekst)                          -- er
else
    print("Ingen skjerm er koblet til")
end
```

I linjen `string.sub(tekst, 1, bredde)` sier vi at vi vil ta ut en
nummer `bredde`. Prøv å endre på disse tallene for å se effekter

Vi kan nå animere teksten ved hjelp av en *for*-løkke.

```
local tekst = "Jeg er en lang tekst. Bytt meg gjerne ut!
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.setTextScale(5)
    skjerm.clear()
    local bredde, hoyde = skjerm.getSize()
    local lengde = #tekst                          -- ny linj

    for i = 0, lengde - bredde do                  -- ny linj
```

```
        local deltekst = string.sub(tekst, i+1, i+bredde
                                    -- deltekstlinjen er e
        skjerm.setCursorPos(1, 1)
        skjerm.clear()                          -- ny linj
        skjerm.write(deltekst)
    end                                         -- ny linj
else
    print("Ingen skjerm er koblet til")
end
```

Skjønner du hvordan tellevariabelen `i` virker? Enkelt sagt teller kommet.

- [ ] Kjør programmet. Ser du animasjonen? Hvis du bare ser slutten animerer for raskt. Prøv å legg inn en `sleep`-kommando neders

- [ ] Hvis dette skal være en ticker som står og går vil vi at teksten s lag en `while true do`-løkke rundt *for*-løkken. Gjentas animasjo programmet ditt.

- [ ] Når animasjonen kommer til slutten av teksten, bare hopper de kan forbedre dette ved å legge på litt luft før og etter teksten.

  Hvor mye luft vi legger på vil ideelt sett avhenge av hvor stor sk repetere tekststrenger. For eksempel er `string.rep("Hei", 4)` luft rundt teksten før vi begynner å animere den.

```
local tekst = "Jeg er en lang tekst. Bytt meg gjerne ut!
local skjerm = peripheral.find("monitor")

if skjerm then
    skjerm.setTextScale(5)
    skjerm.clear()
    local bredde, hoyde = skjerm.getSize()
    local luft = string.rep(" ", bredde)      -- ny linj
    tekst = luft .. tekst .. luft             -- ny linj
```

```
    local lengde = #tekst
    while true do
        for i = 0, lengde - bredde do
            local deltekst = string.sub(tekst, i+1, i+br
            skjerm.setCursorPos(1, 1)
            skjerm.clear()
            skjerm.write(deltekst)
            sleep(0.2)
        end
    end
else
    print("Ingen skjerm er koblet til")
end
```

Pass på at det er et mellomromstegn i `string.rep(" ", bredde`