

# Lærerveiledning: Å programmere en historie

## La elevene simulere tilfeldige utfall i Scratch



### Introduksjon

I dette undervisningsopplegget skal elevene bruke det de kan om programmering og sannsynlighet til å lage en interaktiv og animert historiefortelling. I denne sammenhengen betyr interaktiv at man i løpet av historien vil få flere valg og at valgene man tar får betydning for historien. Noen av valgene skal kunne føre til minst 2 ulike hendelser der det er programmert inn en sannsynlighet for at hver hendelse skal inntreffe. For at historien skal være animert så må figurene og bakgrunnen man bruker til å fortelle historien bevege på seg.

Undervisningsopplegget kan inngå som en del av arbeidet med disse læreplanmålene i matematikk fra ungdomstrinnet:

- utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering
- simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering
- beregne og vurdere sannsynlighet i statistikk og spill
- utforske matematiske egenskaper og sammenhenger ved å bruke programmering

### Hensikt og læringsmål

Hensikten med dette undervisningsopplegget er å gi elevene trening i algoritmisk tenkning og programmering, samtidig som man lar dem jobbe kreativt og utforskende for å lage en egen historiefortelling som bygger på tilfeldige forsøk. Det vil hovedsakelig være fokus på at elevene skal lære mer om programmering i dette undervisningsopplegget. På enkelte tidspunkt vil de også være nødt til å bruke det de kan om sannsynlighet for å få programmet sitt til å fungere.

Undervisningsopplegget dekker ikke helheten av alle læreplanmålene over. Elevene vil for eksempel kun jobbe med deler av dette læreplanmålet, markert i fet skrift:

- **simulere utfall i tilfeldige forsøk** og beregne sannsynligheten for at noe skal inntreffe **ved å bruke programmering**

Hvis man skal arbeide med å beregne sannsynligheten for tilfeldige forsøk så vil det nok være mer egnet å bruke regneark eller skriftspråk som Python til å regne ut den relative frekvensen. I slutten av denne veiledningen er det lagt med et forslag til hvordan man kan bruke Scratch til å regne sannsynligheten for sammensatte hendelser.

Forslag til aktuelle læringsmål:

- Bruke variabler for å lage en enkel terning i Scratch
- Bruke meldinger og vilkår for å lage en animasjon til hvert terningkast
- Utforske hvordan man kan bruke det man har lært om meldinger og terningkast til å lage en interaktiv og animert historie.
- Bruke variabler for å programmere hva sannsynligheten for at hendelsene i historien skal inntreffe.

## Utstyr og programvare

I denne veiledningen er det tatt utgangspunkt i at elevene bruker <https://scratch.mit.edu/> når de programmerer. En fordel med å bruke Scratch er at det er enkelt å bruke og komme i gang med når man har liten erfaring med programmering. Man trenger kun en datamaskin eller et nettbrett som har tilgang til internett. Det er ingenting i veien for å bruke andre programmeringsspråk dersom både lærer og elever har hatt opplæring i dette tidligere. Undervisningsopplegget er bare testet ut på ungdomstrinnet, men kan gjennomføres med både yngre og eldre elever med små modifikasjoner.

Det kan være lurt å lage en egen lærerkonto for Scratch. Dette gir muligheten til å se elevenes kode så snart de har lagt den ut prosjektet sitt. På denne måten blir det enklere å holde oversikt underveis. Scratch godkjenner hver lærerkonto individuelt så det kan være lurt å ute i god tid før første programmeringsøkt med elevene. Her kan du bestille lærerkonto: <https://scratch.mit.edu/educators#teacher-accounts>.

## Forkunnskaper

Det er en fordel om både lærere og elever har jobbet litt med programmering tidligere og har lagd enkle programmer som består av variabler, løkker og vilkår. De to første delene av undervisningsopplegget er i større grad lærerstyrt og man kan nok enkelt gjennomføres selv med liten erfaring med programmering.

## Forslag til gjennomføring

### Del 1 – Lage en enkel terning

Gi elevene en utfordring om å finne ut hvordan de kan lage en egen terning eller en tilfeldig tallgenerator på egenhånd.

Det kan være lurt å presisere at de ikke nødvendigvis trenger å lage et program som ser ut som en fysisk terning, men at når man klikker på en knapp skal det tydelig fremgå hvilken verdi terningen viser.

I sin aller enkleste variant består terningkoden av kun tre ulike kodeblokker:



Det er laget en variabel som har fått navnet «terning». Denne variabelen vil bli satt til et tilfeldig tall fra 1 til 6 hver gang man klikker på flagget. Her er det valgt at variabelen alltid skal vises i venstre hjørnet av output-vinduet i Scratch slik at man alltid kan se det siste tallet man fikk på terningen.

### Del 2 – Videreutvikle terningen

I denne delen skal elevene videreutvikle terningkoden. Elevene skal bestemme hva som skal skje for hvert terningkast. Det vil sannsynligvis være nødvendig å bruke meldinger og vilkår for å få koden til å fungere.

Den aller enkleste varianten av en slik kode kan bestå av en hvis-blokk og en kodeblokk der en figur forteller hva man fikk på terningen.

Her er et eksempel på starten på en slik kode som består av kun ett vilkår:



Hvis man skal videreutvikle programmet enda mer vil man muligens være nødt til å benytte seg av flere vilkår og meldinger.

Her er et eksempel på en slik kode:



I koden ovenfor har eleven brukt ulikheter og 3 vilkår for å bestemme hvilken melding som skal sendes ut. Eleven har definert at hvis terningkast-variabelen viser 1 eller 2 skal så skal meldingen «tap» sendes. Hvis terningkast-variabelen viser 5 eller 6 så sendes meldingen «vinn». Dersom terningkast-variabelen ikke blir noen av disse tallene så sendes meldingen uavgjort. I dette tilfelle kan kun terningkast-variablene være tallene 1 til 6 så uavgjort sendes kun ut når terningkastet blir 3 eller 4. Meldingene som sendes ut brukes som et utgangspunkt i programmet for å velge hva figurene skal gjøre når de mottar de ulike meldingene.

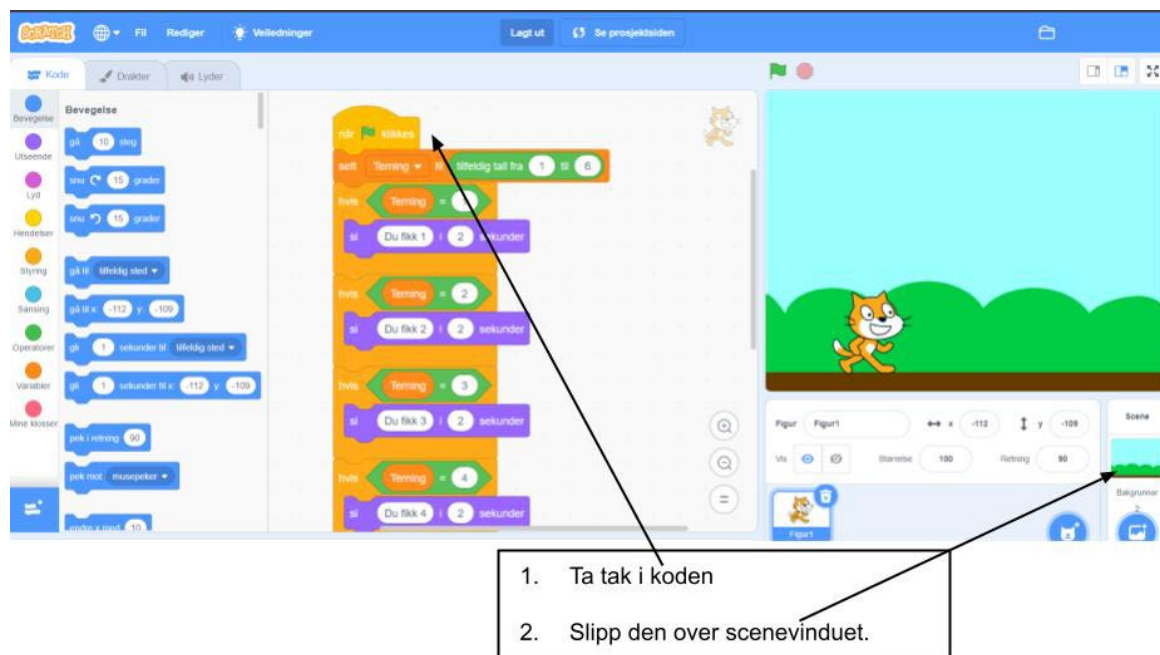
[Her er et eksempel på et program som er utviklet av en elev på 9. trinn](#) (lenke til Scratch)

### Tips – Legg hovedkoden på scenen

I Scratch lagres koden på hver figur man programmerer. Hvis man fjerner en av figurene man har programmert underveis så vil all koden som er på figuren bli slettet. Elevene kan ofte gjøre dette ved et uhell siden de legger terningkoden på katten som er den forhåndsvalgte figuren de møter når de begynner å programmere. Hvis de på et senere tidspunkt prøver å erstatte katten med en annen figur så vil hele koden deres bli borte. Det kan derfor være lurt å tipse elevene om å legge terningkoden på scenen. Da vil det være mindre risiko for at de sletter koden ved et uhell underveis.

## Å programmere en historie

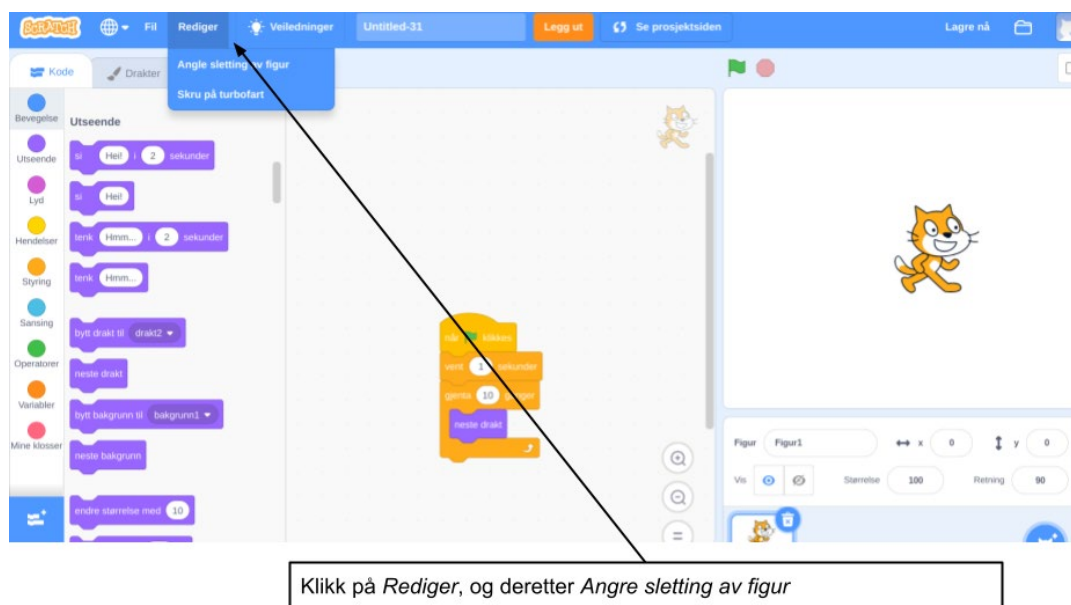
Her er et bilde som viser hvordan man flytter koden fra en figur til scenen:



Det er mulig å bruke nøyaktig samme metode for å kopiere en kode fra en figur til en annen.

For å holde oversikt kan det være mest ryddig om man kun legger koden som bare brukes av en figur på selve figuren. Dette kan være spesielt lurt hvis man jobber med større prosjekter som inneholder mange ulike figurer. «Felleskoder» som terningkoden og andre kodeblokker som kommuniserer med flere figurer bør legges på scenen.

Hvis man likevel skulle være så uheldig å slette en figur med mange kodeblokker på kan man heldigvis få figuren tilbake slik det er vist på bildet under:

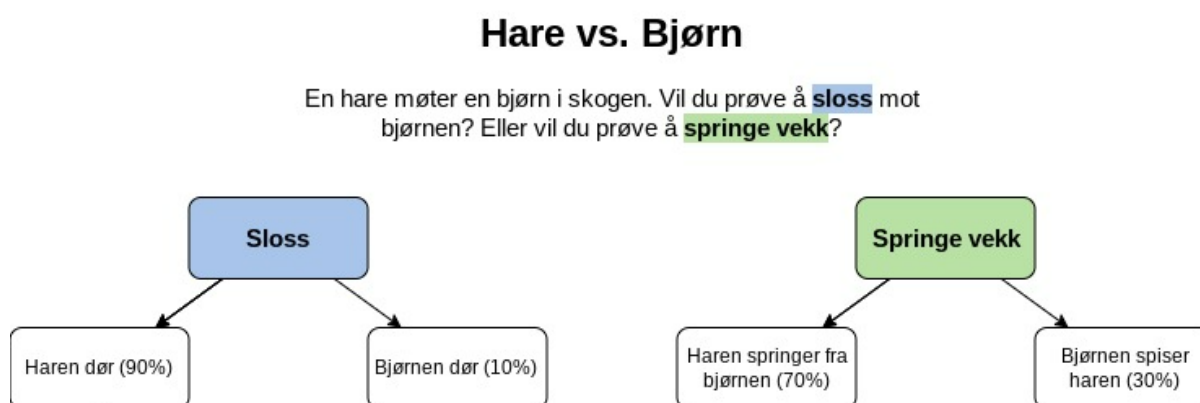


## Del 3 – Planlegge historien

Av erfaring så er det vanskelig for elevene å komme i gang uten en konkret plan til hva innholdet i historien skal være. Enkelte kan begynne å lage et program som ikke følger kriteriene til oppgaven i det hele tatt. Det kan derfor være lurt at de lager et flytskjema som gir en oversikt over alle hendelsene i historien sin. Det kan også være lurt å presentere et eksempel på en ferdig historie både som inspirasjon, men også for å konkretisere hvordan et sluttprodukt kan se ut. Hvis man velger å gjøre dette så bør man unngå å gi elevene tilgang til selve koden. Elevene vil nok ha mer igjen for å utforske Scratch på egenhånd i første omgang.

[Her er et eksempel på en ferdig historie som er laget av en lærer \(Hare vs. Bjørn\)](#)

Her er et flytskjema som viser alle mulige hendelser i historien i eksempelet over:



For enkelte elever kan det også være en fordel å utforske hvilke figurer og bakgrunner som finnes i Scratch som en del av planleggingsfasen. Det kan være enklere å lage en historie rundt en konkret figur de synes virket interessant. Når elevenes flytskjema er godkjent av lærer kan de begynne å programmere.

## Del 4 – Programmere historien

I denne delen bør det meste gå av seg selv så lenge elevene har lagt en god plan i del 3 og klarer å bruke det de har lært i del 1 og del 2.

Det vanligste stoppunktet for elevene er når de skal gjøre historien interaktiv og gi spilleren muligheten til å ta forskjellige valg underveis. Dette kan løses på flere måter, elevene må selv velge hvilken input som skal brukes for å gjennomføre valget.

Her er noen eksempler på vanlige løsninger:

Man kan klikke på figurene:



Man kan bruke tastaturet:



Man kan få figurene til å stille et spørsmål og la spilleren svare:



Her er noen eksempler på ferdige programmer som elever på 9. trinn har utviklet:

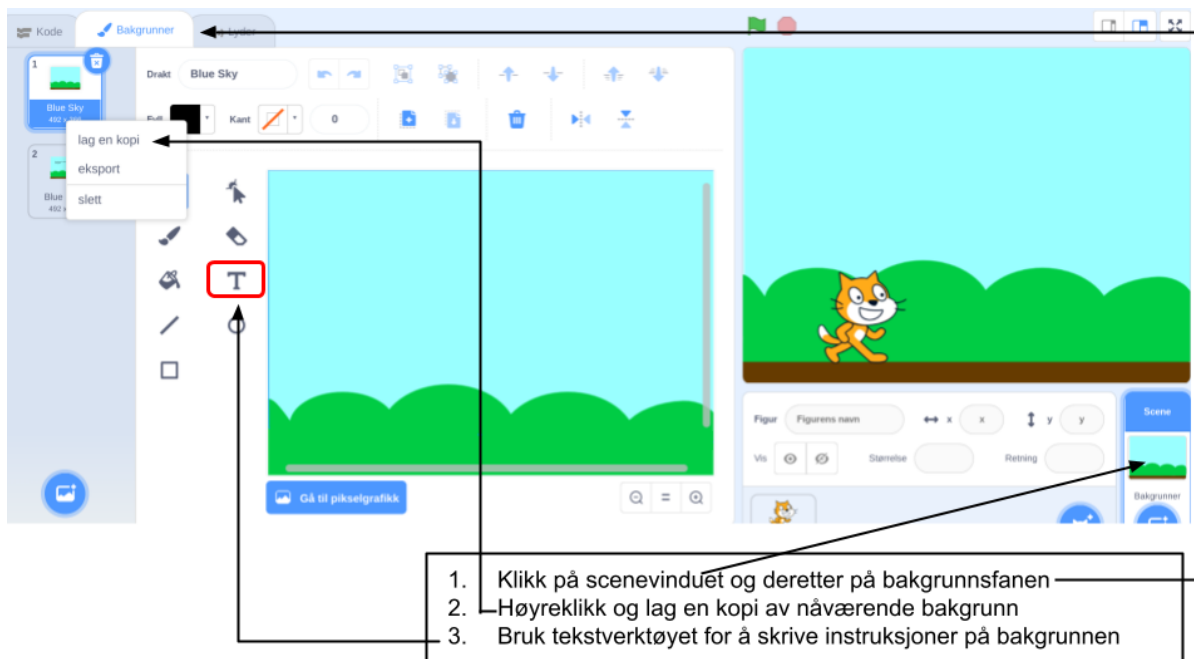
- [Eksempel 1 – Banan eller eple](#)  
 Dette programmet har brukt spørsmål og svar for at spilleren skal kunne ta valg underveis. Man får kun ett valg i løpet av historien, hver av valgene fører til 2 forskjellige hendelser.
- [Eksempel 2 – Sentre eller skyte?](#)  
 Dette programmet bruker klikking for at spilleren skal kunne ta valg underveis.

- [Eksempel 3 – Jobbas adventure](#)

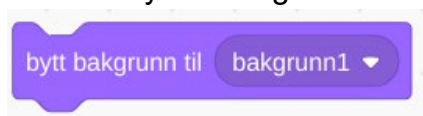
Dette eksempelet viser et program som har tatt en litt alternativ vinkling på oppgaven. Det er lagd som et mer tradisjonelt 2d-plattformspill og ikke en historie. Spilleren får flere valg underveis, valgene man tar fører til kun en bestemt hendelse. Sannsynlighet kommer inn ved at det er programmert inn en sannsynlighet for at figurene og banene endrer utseende for hver gang man dør eller starter et nytt spill.

## Tips – Redigere bakgrunner

Man vil være nødt til å gi informasjon om de ulike valgene i løpet av historien. Man kan få figurene til å gi informasjon ved å bruke kodeblokkene «snakke» eller «tenke». En annen metode er å lage en kopi av bakgrunnen å legge inn tekst på den. Husk at du i så fall må lage en kopi av den nåværende bakgrunnen.



Hvis man benytter seg av denne metoden så må man huske å legge inn i koden når det skal byttes bakgrunn:



Man kan redigere figurer og lage nye drakter til dem med den samme metoden.

## Del 5 – Presentasjon og prøve ut hverandres programmer

I den siste delen av dette undervisningsopplegget bør man legge til rette for at elevene viser frem hva de har laget til resten av klassen. Det kan ta mye tid hvis hver elev skal opp å presentere egen kode og forklare hvordan programmet deres fungerer. Det finnes flere lure måter å effektivisere dette på. Elevene kan for eksempel deles inn i grupper og prøve ut hverandres programmer. Man kan også lage en liten utstilling der halvparten av elevene viser frem og forklarer logikken i egen kode mens resten av klassen prøver ut programmene deres.

Det viktigste er uansett at man får satt av tid til at elevene får testet ut hverandres programmer og snakket om programmering. En annen artig vri er at man skal prøve å gjette hvilken sannsynlighet som er programmert inn for de ulike hendelsene i historien. Dette kan blant annet være en fin samtalestarter om store talls lov.

## Vurdering

Man får mye informasjon om elevenes kompetanse i samtalene som oppstår i løpet av undervisningen, spesielt i situasjonene der elevene oppsøker veiledning og hjelp til hvordan de skal komme seg videre med oppgaven. Hvis man har opprettet lærerkonto og elevene velger å «legge ut» prosjektet sitt så vil man som sagt ha mulighet til å se på koden til hver enkelt elev både før og etter undervisning.

Et av læreplanmålene som dette undervisningsopplegget bygger på handlet om å «utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering». Hvis man skal vurdere i hvor stor grad elevene har klart å *utforske*, *teste* og *forbedre* en algoritme så vil det være essensielt å gjennomføre god underveivurdering og få innsikt i elevens tankeprosess mens de jobber med oppgaven. Dette er noe som er vanskelig å vurdere ved å bare se på den ferdige koden. Samtalene, observasjonene og notatene man tar underveis blir derfor ekstra viktig for å danne seg et helhetlig bilde av elevenes kompetanse og arbeidsprosess. Det gjør det også lettere å gi læringsfremmende tilbakemeldinger og gode tips til hvordan elevene skal komme seg videre med oppgaven.

Hvis man ønsker å gjennomføre en mer formell sluttvurdering kan det være aktuelt å ha en samtale med hver elev der de forklarer logikken i egen kode og hva de har tenkt når de har laget programmet. En slik sluttvurdering kan gi læreren innsikt i elevens tekniske nivå når det kommer til programmering og egner seg nok bedre til å vurdere tre av de foreslåtte læringsmålene fra innledningen i denne veiledningen.

- Bruke variabler for å lage en enkel terning i Scratch
- Bruke meldinger og vilkår for å lage en animasjon til hvert terningkast
- Bruke variabler for å programmere hva sannsynligheten for at hver hendelse i historien skal inntreffe.

## Didaktiske betraktninger

I denne veiledningen finner man flere eksempler på ferdiglaget kode som man kan benytte seg av. Dette kan være til stor hjelp dersom man står helt fast, men det kan også være et hinder om man lar elevene kopiere den ukritisk. I store deler av dette undervisningsopplegget bør man gi elevene rom for å selv utforske hvilke muligheter Scratch har og finne ut hvordan det kan brukes som et verktøy for å lage det de ønsker. Dette krever tålmodighet, noe som kan være vanskelig å håndtere for elever som opplever at de har stanget hode i veggene i en lengre periode. Som lærer må man prøve å finne en balanse mellom å gi eleven rom til å utforske alene, gi elevene hint om hvordan de skal komme seg videre, eller vise frem konkrete løsningsforslag på et delproblem. I mange tilfeller kan det være nok å gi et lite hint om hvilke blokker elevene kan benytte seg av for å få koden til å fungere. Andre ganger kan det være nok å gi et hint om hvilke kategorier de finner de aktuelle blokkene i. Noen elever trenger kun hjelp til å feilsøke i koden og en liten forklaring på hvorfor koden deres ikke fungerte.

Det viktigste budskapet til elevene er nok at de får mest igjen for å prøve å lage en egen kode i stedet for å kopiere fra en medelev eller læreren. Hvis man velger å vise frem konkrete løsningsforslag så kan det være lurt å poengtere at en oppgave kan løses på flere måter og at de ikke nødvendigvis finnes en metode som er bedre enn en annen. Denne ideen kan styrkes ved at elever som har løst oppgaven på forskjellige måter presenterer hva de har gjort for resten av klassen.

## Videre arbeid med sannsynlighet

Hvis man skal beregne sannsynlighet ved hjelp av programmering er det nok mye lettere å regne ut den relative frekvensen til simuleringer som er gjort i regneark eller ved å bruke skriftspråk som Python. Man kan likevel bruke Scratch til å regne sannsynligheten av sammensatte hendelser. Hvis man ønsker en videreutvikling av dette undervisningsopplegget kan det være aktuelt å bruke Scratch til å regne ut sannsynligheten for alle hendelsene som inntraff under historien. Her er et forslag til hvordan en slik kode kan se ut.

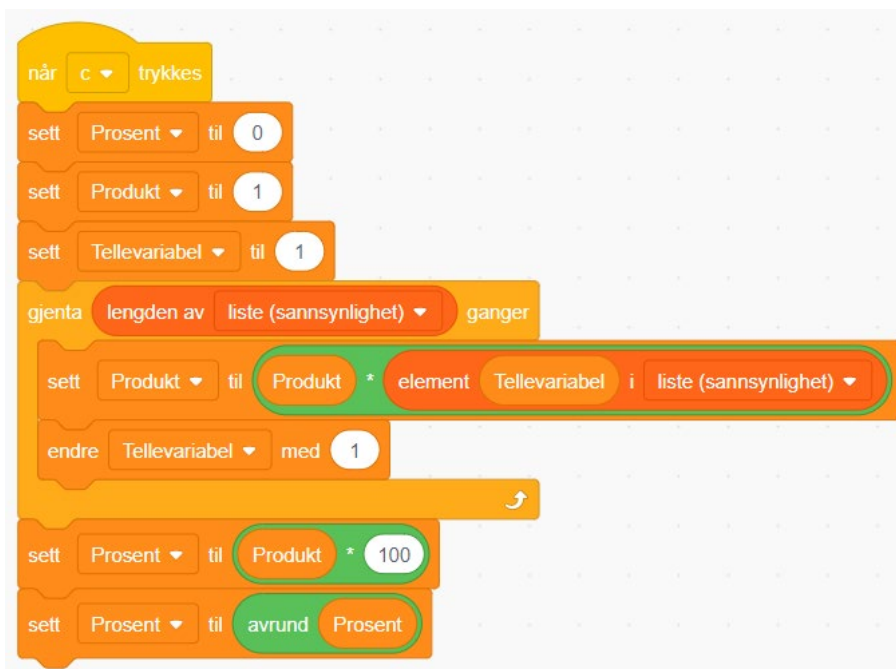
Man er først nødt til å lage en liste som lagrer sannsynligheten for hver hendelse som har inntruffet. Koden fra bildet under kommer fra historien om haren og bjørnen. Hvis man velger å slåss mot bjørnen har man 90% sjanse for å dø og 10% sjanse for å overleve (se flytskjema på side 6). Hvis haren dør så vil man at desimaltallet 0,9 skal lagres i listen, hvis haren overlever vil man at desimaltallet 0,1 skal lagres i listen.



Denne koden bruker meldinger som premisser for når desimaltallene skal blir lagt i listen. Hvordan dette løses vil være avhengig av hvordan resten av programmet er bygd opp.

Når man har kommet til slutten av historien ønsker man at alle desimaltallene i listen skal multipliseres med hverandre slik at man får regnet ut sannsynligheten for en rekke med sammensatte hendelser.

Her er et forslag til hvordan en slik kode kan se ut:

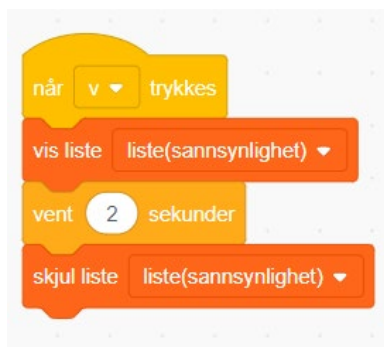


Til nå er sannsynligheten for de enkelte hendelsene gjennom historien lagret i en liste med navn «sannsynlighet». Koden over finner og presenterer den samlede sannsynligheten for disse hendelsene. Videre er koden forklart, med eksempel, blokk for blokk der vi antar at det er lagret to verdier i listen 0,3 og 0,9:

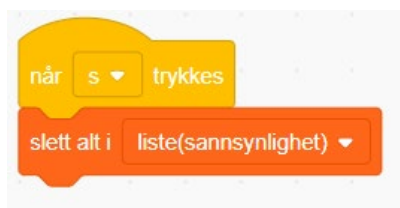
- Koden aktiveres når bruker trykker «c».
- Variabelen *Prosent* blir satt til 0
- Variabelen *Produkt* blir satt til 1
- Variabelen *Tellevariabel* blir satt til 1
- Løkken settes til å gjenta likt antall ganger som det er elementer i sannsynlighetslisten
- *Produkt*-variabelen settes til verdien av *Produkt* multiplisert med verdien fra listen lagret i posisjon lik *Tellevariabel*. I første gjennomgang er verdien av variablene *Produkt* og *Tellevariabelen* begge satt til 1. Altså vil tallet 1 bli multiplisert med verdien som ligger lagret i posisjon 1 i listen «sannsynlighet». La oss si at denne verdien er 0,3. Altså blir *Produkt* satt til  $1 * 0,3 = 0,3$ .
- *Tellevariabelen* økes med 1 til verdien 2, og løkken fortsetter om det er mer enn ett element i listen. I gjennomgang 2 er verdien 0,3 lagret i variabelen *Produkt*. Den verdien vil da multipliseres med verdien lagret i listens posisjon 2, siden *Tellevariabelen* nå er 2. La oss anta at denne verdien fra listen er 0,9. Den nye verdien i variabelen *Produkt* vil da settes til  $0,3 * 0,9 = 0,27$ .
- Når alle elementene i listen er multiplisert med hverandre tar løkken slutt. Variabelen *Prosent* blir satt til verdien i *Produkt* multiplisert med 100.
- Verdien i variabelen *Prosent* blir avrundet til nærmeste heltall.

Om listen «sannsynlighet» inneholder 2 elementer med verdi 0,3 og 0,9 vil altså koden gi en output på 27.

Denne koden krever nok mer forståelse for programmering enn det resten av dette undervisningsopplegget så langt har gjort. Det kan likevel være en utfordring for flinke elever å prøve å lage en egen kode eller forklare hvorfor denne koden fungerer.



Når man tester ut koden for å regne ut sannsynligheten av de sammensatte hendelsene kan det være greit å ha en knapp på tastaturet som viser alle elementene som ligger i listen. En slik kode kan se slik ut, da vises listen i kun 2 sekunder før den skjules igjen.

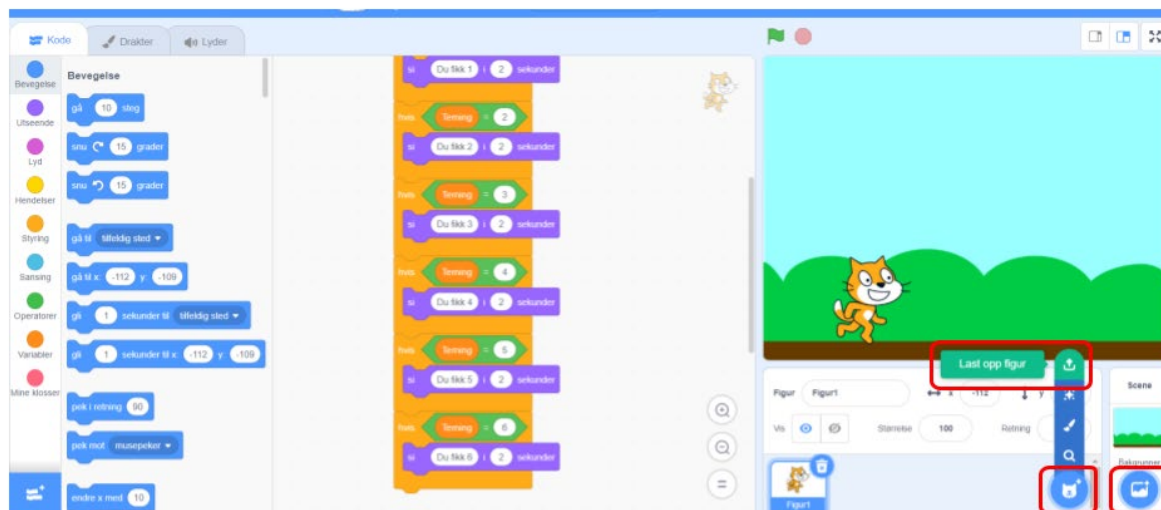


Det kan også være praktisk å ha en sletteknapp som sletter alle elementene i listen for hver gang man starter historien på nytt. Man kan bruke en egen knapp på tastaturet for dette eller bake det inn i selve koden som starter historien.

## Andre tips og triks

### Finne figurer eller bakgrunner på nett

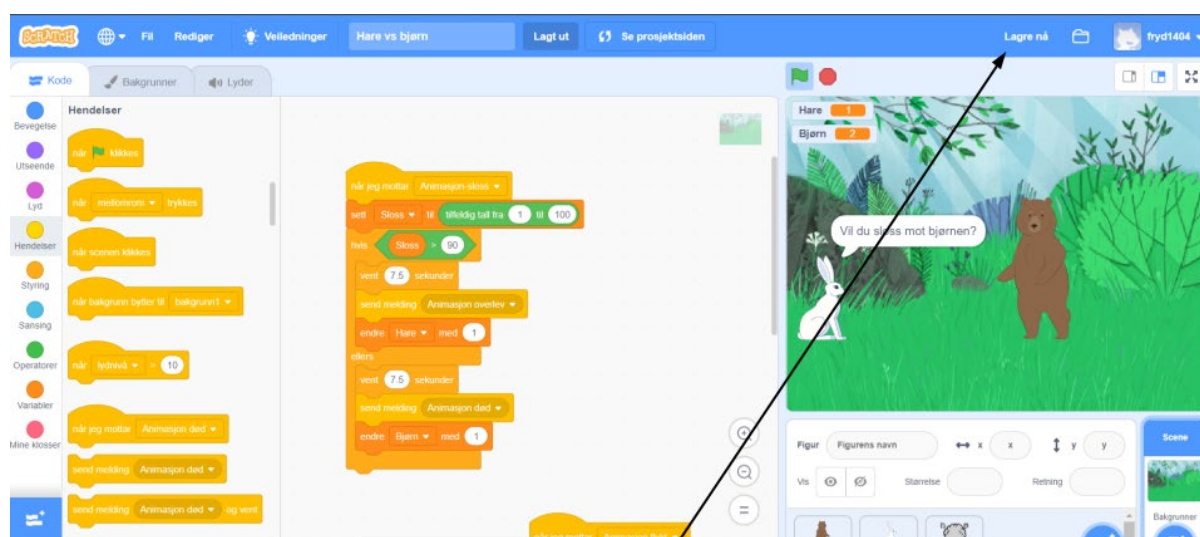
Scratch har et begrenset antall figurer og bakgrunner som man kan bruke når man starter å utvikle et nytt program. Det er lett å legge til nye figurer og bakgrunner ved å bruke knappene som er markert i rødt under.



Bakgrunnene man laster opp kan ofte lett tilpasses Scratch-vinduet. Det kan være vanskeligere å finne nye figurer. Ved å søke etter «scratch sprites» på google vil man finne scratch-prosjekter som er lagt ut med mange ferdiglagde figurer. Man kan enkelt låne disse figurene til sitt eget prosjekt ved å gå inn i prosjektet, høyreklikke på figuren man ønsker å låne, og deretter klikke på eksporter. Da vil figuren lastes ned til din datamaskin.

## Lagring og dårlig internett

Skoler med dårlige nettverk kan være utsatt for at elevene opplever at arbeidet deres går tapt underveis. Scratch lagrer som ofte det man jobber med jevnlig i bakgrunnen uten at man legger merke til det. Det er likevel en god vane å klikke på «lagre nå» før man avslutter en programmeringsøkt. På skoler med dårlig internett risikerer man at ingenting blir lagret automatisk.



Klikk på "lagre nå" før du avslutter. Hvis denne knappen er borte er programmet allerede automatisk lagret.

I enkelte tilfeller får man opp en oransje feilmelding når man klikke på denne knappen. Da vil man få en oppfordring om å laste ned programmet lokalt på maskinen. Dette er noe man bør oppfordre elevene til å gjøre for å være sikker på at arbeidet deres ikke går tapt.