Programmering i skolen

Notat fra Senter for IKT i utdanningen

SENTER
FOR IKT I
UTDANNINGEN

Programmering i skolen

Notat fra Senter for IKT i utdanningen

SENTER
FOR IKT I
UTDANNINGEN

Senter for IKT i utdanningen 2016

Senter for IKT i utdanningens notatserie har som målsetting å problematisere og diskutere ulike temaer som er sentrale i senterets virksomhet. Hensikten er å spre kunnskap og spore til nyansert debatt innenfor temaene.

Skrevet av:
Kristine Sevik m.fl.

Alle foto: Moment studio

Senter for IKT i utdanningen // 1. reviderte utgave 2016
ISBN 978-82-93378-40-2 (Trykt)
ISBN 978-82-93378-41-9 (PDF)

Innhold

side

1. Innledning

De siste årene har det vokst fram en internasjonal bevegelse for å fremme programmering i skolen. I 2013 ble nettstedet Code.org lansert, med visjon om at: «(…) every student in every school should have the opportunity to learn computer science»[1]. Code.org har blitt støttet av IT-selskaper som Google, Microsoft, Amazon og Facebook, og kjente personer som Bill Gates, Bill Clinton og Mark Zuckerberg. Siden 2013 har kodetimen (Hour of Code) blitt gjennomført i en rekke land, og etter å ha deltatt på kodetimen i 2014 ble president Obama kalt den første amerikanske presidenten som skrev en linje med kode.

Argumenter for programmering i skolen knyttes gjerne til nødvendige ferdigheter for det 21. århundre, fremtidige behov for kompetanse i næringslivet og evne til å forstå hvordan et stadig mer digitalisert samfunn fungerer.

EU har satt programmering på sin *Digital Agenda for Europe* og oppfordrer utdanningsministre i medlemslandene til å fremme programmering i skolen[2] . De begrunner satsingen med at programmering er en viktig ferdighet som fremmer kreativitet, lærer folk å samarbeide, lærer folk å jobbe sammen over geografiske avstander og å kommunisere via et felles språk.

I Norge påpekte NOU-utredningen *Hindre for digital verdiskaping* i 2013 manglende kompetanse i programmering i befolkningen, og et behov for å legge til rette for at barn og unge ikke bare er i stand til å bruke, men også skape digitalt innhold og digitale tjenester (NOU 2013:2). Sommeren 2015 leverte Ludvigsen-utvalget NOU-en *Fremtidens skole – Fornyelse av fag og kompetanser*, med understreking av hvordan skolen og skolefagene bør endres for å møte fremtidens kompetansebehov (NOU2015:8). Høsten 2016 leverte en ekspertgruppe oppnevnt av Utdanningsdirektoratet rapporten *Teknologi og programmering – et nytt fag i grunnskolen?*, med anbefaling om at det opprettes et nytt fag innen teknologi og programmering for å møte fremtidens behov og kompetanse.

Dypere forståelse av underliggende prosesser og systemer, evne til logisk tenkning og ferdigheter i å være skapende og produserende har alltid vært viktige i læring og utdanning. Utviklingen av det teknologirike samfunnet vi har i dag og i fremtiden, utfordrer både måten vi lærer på og hvilke kompetanser som blir viktige. Å skape meningsfulle læringsprosesser, utvikle skapende evner og dyp forståelse krever andre tilnærminger i teknologirike omgivelser. Å skape og produsere digitalt krever forståelse og kompetanse i programmering.

Senter for IKT i utdanningen mener kunnskapsdomenet programmering kan ha en plass i norsk skole, gjennom hele skoleløpet. Det er i hovedsak tre tilnærminger til å innføre programmering i skolen:

1. Programmering som (del av) eget IKT-fag
2. Programmering integrert i eksisterende fag
3. Programmering som fagovergripende kompetanse i flere fag

Tross stofftrengsel i skolen mener ekspertutvalget nedsatt av Utdanningsdirektoratet at det er tungtveiende grunner for et eget fag, og at det vil gi godt grunnlag for fornyelse av det faglige innholdet både i realfagene og i øvrige fag. Utvalget mener videre at faget bør være praktisk, og gi eleven mulighet til å tilegne seg grunnleggende teknologisk kompetanse.

Dette notatet utdyper senterets syn på programmering i skolen og hvordan det kan understøtte utvikling av sentrale kompetanser og ferdigheter hos elever. Enten i form av et eget fag eller ved å implementeres gjennom andre modeller.

2. Bakgrunn

Programmering i skolen er ingen ny idé. Allerede på slutten av sekstitallet ble programmeringsspråket LOGO utviklet med tanke på utdanning. Tanken var at det å lære programmering ville gjøre elevene bedre i blant annet problemløsning, en tankegang som i stor grad var basert på Seymour Paperts ideer (Papert 1993:a og Papert 1993:b). På åttitallet ble det gjennomført diverse forsøk med programmering i skolen, blant annet som del av EDB som valgfag. Dette medførte ikke den revolusjonen i pedagogikken enkelte hadde sett for seg, og programmering forble lenge en aktivitet for spesielt interesserte.

I dag er imidlertid situasjonen en helt annen. Verktøyene for å programmere og å lære programmering er lettere tilgjengelige, digital kompetanse står sentralt i læreplanene, og samfunnsutviklingen har gjort at vi hele tiden omgir oss med en rekke digitale produkter og enheter som vi interagerer med og tilpasser i ulik grad.

De siste årene har vi også sett fremveksten av en skaperbevegelse som kombinerer entusiasme for teknologi, robotikk og programmering med en gjør-det-selv-holdning og uformelle nettverk. Bevegelsen er nært knyttet til etablering av verksteder (Makerspace) der skapere med ulike interesser og kompetanse møtes og samarbeider. I Norge har en rekke slike skaperverksteder blitt etablert de siste årene, gjerne i tilknytning til biblioteker, vitensentre og museer[3].

Bevegelsen *Lær Kidsa Koding* har siden 2013 gått i bresjen for programmering som aktivitet for barn og unge. Lær Kidsa Koding har spilt en sentral rolle i å sette programmering på den utdanningspolitiske agendaen og har bidratt til etablering av en rekke kodeklubber over hele landet. I skrivende stund lister nettsiden deres opp 98 kodeklubber fra Hammerfest i nord til Kristiansand i sør[4]. Privatpersoner og entusiaster har stilt opp for lokalmiljøene sine og startet kodeklubber. I 2016 forventes det at opp mot 100 000 norske barn og skoleelever vil gjennomføre Kodetimen[5]. Dersom kunnskap om programmering vurderes til å være grunnleggende og viktig kompetanse for framtiden, er det likevel problematisk å la dette være avhengig av slike frivillige initiativ. Langt fra alle barn og unge har tilgang til kodeklubb der de bor, eller mulighet til å delta på fritiden. Kodetimen når riktignok ut til flere elever, men det gir bare en veldig grunnleggende smakebit til elevene som deltar.

Ved oppstart av skoleåret 2016–2017 ble det på 154 skoler fordelt på 55 kommuner igangsatt et pilotforsøk med programmering som valgfag. Pilotprosjektet skal gå over tre år, og skolene får støtte fra Utdanningsdirektoratet og Senter for IKT i utdanningen.[6] Senter for IKT i utdanningen har utviklet en ny MOOC (nettbasert kurs) i programmering for lærere.[7]

3. Begrepet programmering

Tradisjonelt har «programmering» vært den foretrukne termen for aktiviteten å skrive programkode, det vil si å lage instruksjoner til datamaskiner og andre digitale enheter om å utføre en oppgave. Programmering inkluderte det å skrive kode, men også beskrive hva programmet skulle gjøre og å designe løsninger.

De senere årene har ordet «koding» blitt stadig mer utbredt i dagligtale, spesielt når vi snakker om barn og unge som skal lære seg å programmere. Vi har fått bevegelser som Lær Kidsa Koding og Kodeklubben i Norge, og liknende frivillige initiativ finnes i en rekke europeiske land.[8] Det kan virke som om begrepet «koding» gir konnotasjoner til noe mer lekende og ufarlig for nybegynnere enn «programmering». «Koding» er muligens også et bedre begrep når vi snakker om visuelt orienterte verktøy for programmering, slik som Scratch[9], MIT App Inventor[10] og Code Studio[11], fremfor tekstbaserte programmeringsspråk.

I denne teksten vil vi gjennomgående benytte begrepet «programmering», da vi mener dette er mest entydig. Programmering, slik det er brukt i dette dokumentet, omfatter mer enn å bare skrive programkode som kan kjøres på en datamaskin, det inkluderer også prosessen med å komme fram til denne koden. Det vil si prosessen fra å identifisere et problem og tenke ut mulige løsninger på problemet, til å skrive kode som kan forstås av en datamaskin, og å feilsøke og kontinuerlig forbedre denne koden.

4. Programmering som kompetanse for det 21. århundre

Programmering eller programmering i skolen begrunnes ofte med at det er nødvendig kompetanse for å lære, arbeide og leve i dagens og morgendagens samfunn. Dette refereres ofte til som kompetanser og ferdigheter for det 21. århundre (*21st Century Skills*), som har mange ulike definisjoner og bestanddeler[13]. De fleste forsøk på å beskrive disse ferdighetene inneholder begreper som kreativitet, innovasjon, kritisk tenkning, metakognisjon (lære å lære), kommunikasjon, samarbeid, digital kompetanse, digital dannelse (literacy), medborgerskap, karriere og arbeidsliv.

Prosjektet *Assessment & Teaching of 21st Century Skills*[14] definerte ferdigheter for det 21. århundre som å høre til i fire brede kategorier (Binkley et al., 2012): hvordan vi tenker, hvordan vi jobber, verktøy vi bruker når vi jobber, og hvordan vi lever i verden (se Figur 1).

Figur 1: 21st Century Skills

4.1 Kompetanse for framtidens arbeidsmarked

Mange initiativ for å lære barn og unge programmering har sitt opphav i ønsket om å utdanne flere til et arbeidsliv som blir stadig mer avhengig av teknologisk kompetanse. Det er jevnlig medie-oppslag om at det i Norge er stor mangel på ingeniører, og at behovet bare vil øke i fremtiden.[15] I tillegg etterlyses bedre realfagskompetanse hos elever som går ut av grunnopplæringen.[16]

En undersøkelse fra Oxford universitet i 2013 anslo at nærmere halvparten av amerikanske jobber vil bli erstattet av maskiner i løpet av 10–20 år (Frey & Osborne 2013). En tilsvarende studie fra Finland konkluderte med at en tredjedel av finske jobber sannsynligvis vil bli erstattet av utstyr kontrollert av datamaskiner i samme tidsrom (The Research Institute of the Finnish Economy 2014). Ifølge Teknologirådet viser undersøkelser at i Sverige kan 53 % av jobbene forsvinne, og i Danmark er anslaget på 37 %.[17]

Fremtidens arbeidsmarked vil altså ha færre manuelle jobber, men det er ingen grunn til å tro at det vil bli behov for færre arbeidstakere i fremtiden. For eksempel er helse og omsorg en sektor i sterk vekst, og det er en utfordring å dekke det økende behovet for arbeidskraft. Fagfeltet eHelse handler om god utnyttelse av IKT for å oppnå helsepolitiske mål og vil bli stadig viktigere i tiden som kommer.[18] Behovet for teknologisk kompetanse handler om tilgang til personer som kan utvikle/skape morgendagens teknologi og maskiner, men like mye om personer som kan bruke dette utstyret. Dette krever en god forståelse av hvordan teknologi virker, og hvordan teknologi kan løse framtidige utfordringer.

Rapporten *Coding Our Future* (European Schoolnet 2015) slår fast at skolen skal utdanne framtidens arbeidsstyrke, men enda viktigere er det å utstyre dagens unge med den nødvendige kompetansen for å mestre og skape egne digitale teknologier, og å mestre og trives i samfunnet:
«We believe that teaching and learning how to code, in formal and non formal education settings, will play a significant role in this process.» (ibid.:4)

Mange unge drømmer om en fremtid i yrker som knapt fantes for 20 år siden (spillutvikler, visuelle effekter, 3D-design, robotikk), eller der teknologi har ført til at selve utøvelsen av yrket er endret (journalist, designer, musikkprodusent). Å bringe programmering, spillprogrammering og 3D-design inn i skolen kan gjøre at fagene oppleves som mer relevante for en del elever, og dermed motivere til økt innsats.

5. Fremtidens skole (Ludvigsen-utvalget)

I juni 2015 leverte Ludvigsen-utvalget sin utredning *Fremtidens skole* (NOU 2015:8) som beskriver hvordan skolen må fornyes for å møte fremtidens kompetansebehov. Utvalget anbefaler at følgende fire kompetanseområder vektlegges i skolens faglige innhold:

- fagspesifikk kompetanse
- kompetanse i å lære
- kompetanse i å kommunisere, samhandle og delta
- kompetanse i å utforske og skape

Samfunns- og teknologiutviklingen virker inn på alle fag, og de fire kompetansene må komme til uttrykk i alle skolefagene, skriver Ludvigsen-utvalget. Programmering kan knyttes opp mot alle de fire kompetanseområdene ved å bruke det inn mot et spesifikt fag eller i et tverrfaglig perspektiv i skolen. Programmering utfordrer elevene med ulike problemstillinger som bidrar til kritisk tenkning og resonnering. Det gir også elevene mulighet til å bruke sin kreativitet og fantasi til å skape noe digitalt ved å omsette en idé til en handling. I dette kapittelet vil vi kort gjøre rede for hvordan programmering kan komme til uttrykk i skolen, og knytte det opp mot de fire kompetanseområdene.

5.1 Fagspesifikk kompetanse

Fagspesifikk kompetanse er knyttet til sentrale fagområder som matematikk, naturfag, språk og estetiske fag. Dette er fagovergripende kompetanser som er relevante for mange ulike fag og kunnskapsområder. Ludvigsen-utvalget trekker frem matematikk som viktig i forbindelse med dette og anbefaler at dette faget styrkes i skolen (NOU 2015:8). Programmering kan bidra til å synliggjøre og styrke de sentrale fagområdene. Ved å for eksempel knytte programmering opp mot matematikk-faget kan man kombinere fagspesifikke ferdigheter med ferdigheter i programmering. Dette gir mulighet for å knytte teoretiske matematiske begreper opp mot en praktisk kontekst der elevene får en forståelse for kunnskap i en større sammenheng.

Programmering er i dag ikke knyttet til et skolefag, men i høyere utdanning er programmering et fag som krever fagspesifikk kompetanse i utøvelsen, slik som logikk, algoritmer og syntaks. Programmering krever mange av de samme ferdighetene som også er knyttet til fagspesifikk kompetanse i matematikk og andre realfag.

Utøvelsen av kunst- og håndverksfaget og musikkfaget baserer seg også i økende grad på digitale teknologier og programmeringsferdigheter. Vi har lenge hatt strikke- og veve-maskiner der mønsteret programmeres på ulike måter. Å komponere musikk er å programmere musikken, og mange musikere spiller ikke lenger instrumenter, men produserer og mikser musikken digitalt. Sonic Pi er et eksempel på verktøy som er utviklet for å lage musikk ved å programmere, til bruk i programmerings- og musikkfag i skolen.

Det er enklere og billigere enn noen gang å kombinere tradisjonelt håndverk med programmerbar teknologi. Ved å integrere programmerbare mikrokontrollere i produkter kan man for eksempel lage «smarte» klær, smykker, vesker og briller. Såkalte «wearables» spås å bli en trend innen utdanning i løpet av få år (New Media Consortium, 2015).

5.2 Kompetanse i å lære (metakognisjon)

Ludvigsen-utvalget anbefaler at det legges vekt på metakognisjon og selvregulert læring i alle fag (NOU 2015:8). Programmering som aktivitet kan være med på å styrke denne læringen. Når elevene programmerer, må de planlegge hva de skal lage og hvordan de skal lage det. Selve programmeringen gjør ideen og tankene om til en praktisk handling der det som er laget, for eksempel har blitt til et spill. I evalueringen av det som er laget, ser eleven på hva som fungerer og hva som kan gjøres videre for å forbedre eller videreutvikle det som er laget.

Programmering gir også en systematisk tilnærming til problemløsning. Programmering innebærer mye prøving og feiling og systematisk feilsøking (de-bugging). Prosessen handler både om å finne den beste løsningen og om å løse oppgaven med færrest mulig steg. Når vi har laget en løsning på et spesifikt problem, handler programmering også om å abstrahere løsningen slik at den kan brukes til å løse liknende problemer i fremtiden; det vil si å lage en algoritme for hvordan løse en type problemer.

5.2.1 Algoritmisk tankegang (Computational thinking)

Å undervise i programmering handler ikke bare om at elever skal lære seg å forstå datamaskiner eller få bedre kompetanse i hvordan de kan bruke digitale verktøy. Det handler også om å gi elevene gode verktøy for å løse oppgaver og problemer. På engelsk snakker man ofte om «computational thinking», og på svensk brukes «datalogiskt tänkande» om det samme. Vi har valgt å oversette dette til «algoritmisk tankegang» på norsk. Algoritmisk tankegang handler ikke om å tenke som en data-maskin, men er en problemløsningsprosess som innebærer å tenke som en informatiker når man skal løse et problem.

Algoritmisk tankegang innebærer å bryte ned store, komplekse problemer til mindre, mer håndterlige del-problemer. Det inkluderer å organisere og analysere data på en logisk måte og å lage fremgangs-måter (algoritmer) for å løse komplekse problemer. Det handler også om å lage abstraksjoner og modeller av den virkelige verden og å generalisere løsninger slik at de kan anvendes til å løse liknende problemer. Denne måten å arbeide på er sentral i programvareutvikling, men kan også brukes som metode i mange andre sammenhenger og fag (se for eksempel Stephenson & Barr 2011).

Å utvikle sin algoritmiske tankegang innebærer at eleven lærer å tilnærme seg problemer på en systematisk måte og foreslå løsninger som kan bruke datamaskiner til å løse (deler av) dem. Det vil si både å:

- tenke på hvilke steg som skal til for å løse et problem, og
- bruke sin teknologiske kompetanse for å få datamaskinen til å løse problemet.

Barefoot Programme[19] beskriver computational thinking som å bestå av seks konsepter og fem måter å jobbe på. De fem arbeidsmåtene eksperimenterende, skapende, feilsøkende, utholdende og samarbeidende samsvarer godt med Ludvigsen-utvalgets vurderinger av hvilke kompetanser som vil være viktige for norske elever i fremtiden.

Figur 2: Konsepter og arbeidsmåter som utgjør computational thinking. Bildet er hentet fra nettstedet til The Barefoot Programme. Bildet er publisert med en Open Government Licence (OGL) som tillater gjenbruk.

Programmering og algoritmisk tankegang kan beskrives som en kompetanse i å:

- abstrahere
- kunne gjennomgå informasjon systematisk
- lære å tilegne seg og forstå forskjellige representasjonsformer
- modulisere (dekomponere) problemer og problemstillinger
- resonnere i iterative og parallelle strukturer[20]

5.3 Samarbeid og kommunikasjon

Å kunne kommunisere, samhandle og delta er et viktig kompetanseområde i skolen og blir sett i sammenheng med ferdigheter og kompetanser for det 21. århundre *(21st Century Skills)*. Ludvigsen-utvalget anbefaler at samhandling og deltakelse rettes mot samarbeid om problemløsning, flerfaglige problemstillinger og deltakelse i faglige diskusjoner (NOU 2015:8). Programvareutvikling gjøres sjelden av enkeltpersoner alene, blant annet fordi oppgavene ofte er store og komplekse. Samarbeid er sentralt i arbeidet med programmering og kan ses i sammenheng med samarbeidslæring slik vi kjenner det i skolen. Programmering som et eget fag eller knyttet opp mot et fag gir muligheter for elever til å arbeide sammen om oppgaver der de drar nytte av hverandres erfaringer og kunnskap. Samarbeid om oppgaven legger til rette for samtaler og diskusjoner rundt utfordringer de møter, og skaper refleksjon og kritisk tenkning om hvordan dette skal løses for å komme videre. Oppsummering og vurdering av arbeidet i felleskap setter elevene i stand til å vurdere hva de har programmert, hvordan de har programmert, og hvorfor de har programmert det slik.

Programmering brukt som en del av samarbeidslæring kan gjøres på mange måter, men en arbeidsmetode som er mye brukt i programmering, «par-programmering», er et godt eksempel på hvordan dette kan gjøres i skolen. I «par-programmering» sitter to programmerere på én arbeidsstasjon og skaper kode sammen. Paret har hver sin rolle som byttes hyppig: «sjåføren» skriver kode mens «kartleseren» holder et mer overordnet blikk og kontrollerer at man beveger seg i riktig retning.[21]

5.4 Kompetanse i å utforske og skape

Det siste av de fire kompetanseområdene i NOU-en Fremtidens skole innebærer kompetanse i å utforske og skape. Utvalget skriver blant annet at en utforskende tilnærming til kunnskap er viktig for fremtidens skole. De understreker også at kritisk tenkning og problemløsning henger sammen med kreativitet og innovasjon (NOU 2015:8).

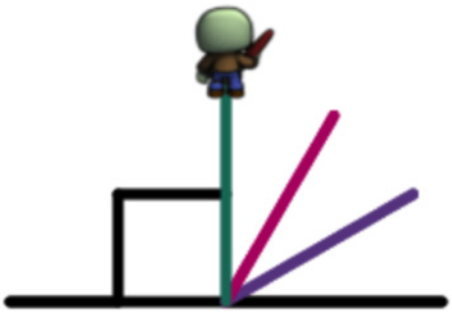
Seymour Papert (1993b:143) parafraserer antropologen Claude Levi-Strauss og bruker det franske, delvis uoversettelige ordet «bricolage» om en måte å lære og å løse problemer gjennom utprøving og lek. På norsk kan vi kanskje snakke om «fiksing og triksing» for noe av det samme. *Bricolage* handler om å arbeide kreativt med de verktøy og ressurser som til enhver tid er tilgjengelig, mye på samme måte som barn konstruerer lek. Improvisasjon og selvdrevet læring innenfor visse rammer er sentralt i Paperts konstruktivisme som tilnærminger til å løse problemer.

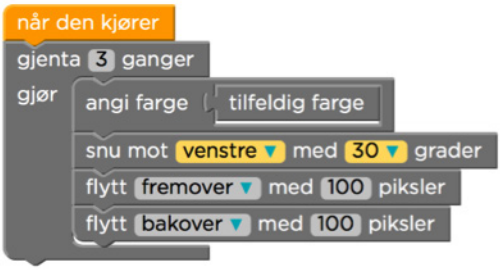
Analytisk tilnærming til problemløsning og bricolage ses ofte som motsatser, men ofte veksler vi mellom å improvisere med verktøyene som er tilgjengelig, og å være systematisk og analyserende. Papert (1993b) bruker den kreative kokken som eksempel, som både improviserer, smaker, tilpasser og måler opp nøyaktige mengder og regner ut forhold mellom ingrediensene. Programmering innebærer elementer av begge: Det er en kreativ prosess der man bruker tilgjengelige verktøy for å løse en oppgave, samtidig som det stiller store krav til systematikk, algoritmisk tenkning (steg-for-steg) og analyse. Programmering innebærer å skape noe nytt eller forbedre et eksisterende produkt. Det er en kreativ prosess som handler om å finne gode løsninger, og i mange tilfeller handler det også om å formulere problemet (hva er det jeg vil løse?).

Vi forbinder ofte programmering med å skape immaterielle produkter (programvare, apper, nettsider osv.), men det handler like gjerne om programmering av fysiske gjenstander og produkter (alt fra industriroboter og militære droner til leketøy og smarte klær). Skaperbevegelsen omfavner det kreativt lekende som tilnærming til produksjon av fysiske produkter.

6. Lav terskel, høyt tak og vide vegger – programmering og tilpasset opplæring

Programmering legger godt til rette for tilpasset opplæring. Fremveksten av brukervennlige verktøy for programmering gjør det enkelt å komme i gang med å lage et produkt som virker, og som eleven kan vise fram og være stolt av. Populære, gratis og nettbaserte dra-og-slipp-programmeringsverktøy, som Scratch og Code Studio, veileder den lærende gjennom programmeringens mysterier på morsomme og «ufarlige» måter. I skolen kan disse verktøyene gi alternative innganger til fagstoffet. For eksempel kan en oppgave være å programmere en figur til å fargelegge streker i en geometrisk figur. Dette kan gjøres visuelt i nettleseren eller ved å programmere, men i begge tilfeller må eleven jobbe med finne riktige vinkler og avstand (se Figur 3). Dette gir gode muligheter for nivåtilpassing.






```
for (var count = 0; count < 3; count++) {
  penColour(colour_random());
  turnLeft(30);
  moveForward(100);
  moveBackward(100);
}
```

Figur 3: Skjermdumper fra Code Studio som viser visuell og tekstbasert programmering for det samme.

Når elevene programmerer, får de umiddelbar tilbakemelding på om de har gjort rett (de ser om det virker etter intensjonen eller ikke). De fleste verktøyene for programmering, inkludert programvare som brukes av profesjonelle programmerere, vil også gi indikasjoner på hva som kan være feil.

Selv om det er forholdsvis enkelt å komme i gang med programmering (lav terskel), finnes det nesten ingen begrensninger på hvor avansert kode man kan skrive (høyt tak), og det er heller ingen begrensninger på hvor komplekse prosjekter man kan arbeide med (vide vegger). Dersom elevene

«når taket» i det verktøyet som brukes (for eksempel et lavterskel-verktøy for visuell programmering), kan de ganske enkelt bytte til andre verktøy og andre programmeringsspråk som gir større utfordringer og muliggjør mer avansert programmering.

Å jobbe med programmering kan gi en alternativ inngang til å jobbe med faglige problemstillinger for elever som strever, samtidig som det kan gi større faglige utfordringer til de elevene som trenger det.

6.1 Bygge bro mellom teori og praksis – dybdelæring og overflatelæring

Det sentrale poenget med kompetanse er ifølge Ludvigsen-utvalget å kunne anvende sine kunnskaper og ferdigheter til å løse og mestre utfordringer (NOU 2015:8). Det er viktig å kunne se sammenhengen mellom det man lærer og hvordan man kan bruke dette, og at ny kunnskap ses i sammenheng med den kompetansen man allerede har.

Programmering gir mulighet for å knytte teori opp mot en praktisk kontekst der elevene skaper teknologi. Programmering utfordrer elevene med ulike problemstillinger som krever kritisk tenkning og resonnering. Det er sjelden en løsning eller fasit på hvordan utfordringer skal løses. Det legger til rette for at elevene må bruke relevante strategier for å løse utfordringene de møter i arbeidet med programmeringen.

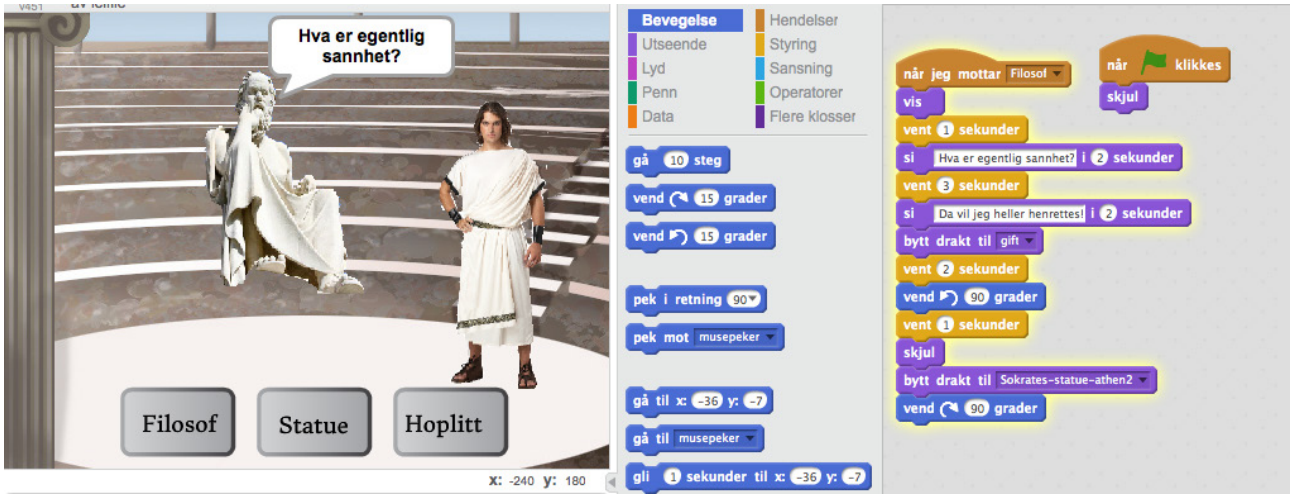
6.2 Praktisk eksempel – dybdelæring

Eksemplet nedenfor (Figur 4) er et eksempel på hvordan teori kan knyttes opp mot en praktisk kontekst, og hvordan kunnskaper og ferdigheter kan anvendes til å løse og mestre utfordringer. Oppgaven går ut på å lage et enkelt fotballspill, der poenget er å score på så mange straffespark som mulig. Et naturlig problem når man lager et slikt spill, er at det er for vanskelig å treffe mål. For å løse dette problemet kan elever benytte mange ulike strategier: Noen vil begynne med å redusere størrelsen på keeperen og eksperimentere med ulike størrelser til de finner noe som virker. Andre vil heller justere ballens hastighet og tilpasse den til keeperens hastighet. Disse løsningene vil gi to ganske forskjellige spill, men begge elevene har løst oppgaven. Herfra kan man diskutere fordeler og ulemper med de ulike løsningene, og hva man kan gjøre for å lage spillet enda bedre.

Figur 4: Scratch-program der flere variabler er avgjørende for spillets funksjonalitet og vanskelighetsgrad[22]

6.3 Praktisk eksempel - Fagspesifikk kompetanse

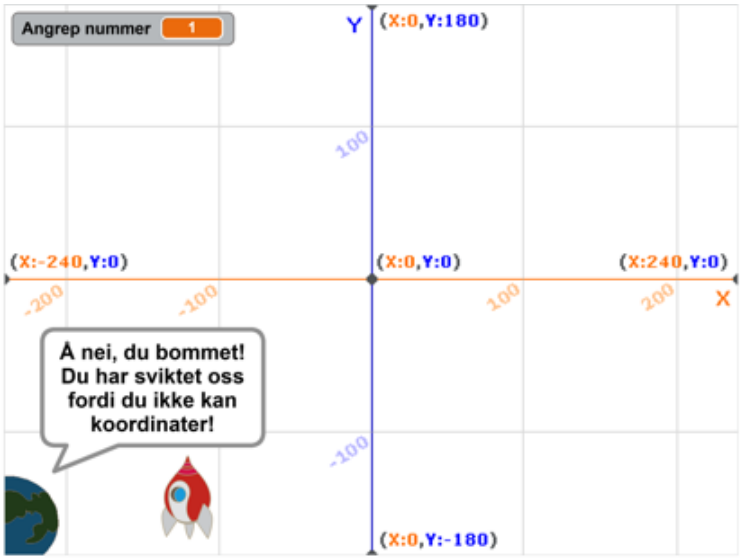
Selv om programmering i seg selv ikke nevnes i læreplanene, kan en rekke kompetansemål i flere fag nås ved hjelp av programmering. For eksempel kan Scratch brukes til å lage animerte og interaktive presentasjoner i alle fag. I tillegg til å være en spennende variasjon for elevene så kan dette lede dem til å forstå stoffet de presenterer på nye måter. Dette egner seg spesielt godt til innlæring av begreper, der begrepet kan illustreres ved hjelp av en animasjon.



Figur 5: Eksempel på presentasjon av begreper ved hjelp av animasjoner i Scratch[23]

Matematikk og fysikk er fag med kompetansemål som egner seg godt for innlæring gjennom programmering. For eksempel skal elevene i løpet av ungdomsskolen kunne «finne og diskutere sannsyn gjennom eksperimentering, simulering og berekning i daglegdagse samanhengar og spel». Simulering kan gjøres ved gjentatt kast av terninger, men man kan få en bedre forståelse av sannsynlighet ved å lage et program som gjør det for deg. Da kan man også lettere simulere et meget stort antall kast og studere hvor mange kast som skal til før gjennomsnittet nærmer seg den beregnede verdien for sannsynlighet.

Vinkler og koordinater er andre temaer som egner seg spesielt godt for programmering. Koordinater blir særlig fremtredende hvis elevene lager sine egne dataspill, der bevegelse gjerne representeres ved endring i koordinater. Dette betyr at elevene kan få innsikt i koordinater selv om spillet ikke eksplisitt trenger å handle om koordinater.



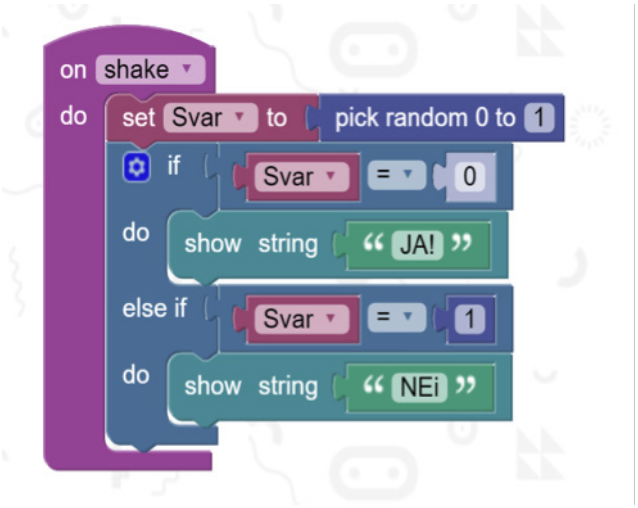
Figur 6: Eksempel på spill som lærer elevene koordinater, både i programmeringen og spillingen av spillet[24]

6.4 Praktisk eksempel – kompetanse i å kommunisere, samhandle og delta

I dette eksemplet får vi et lite innblikk i hvordan programmering kan brukes for å styrke og utvikle kompetansen i kommunikasjon og samhandling. Oppgaven i dette eksemplet er konstruert slik at elevene skal samarbeide («pair programming») om programmeringen, men også prøve ut programmet på hverandre når det er ferdig. Oppgaven går ut på å lage et program som kjøres på den lille, programmerbare datamaskinen micro:bit. Programmet lar micro:biten velge tilfeldig ja eller nei som svar når den blir ristet på. Så skal elevene stille hverandre ja- eller nei-spørsmål og la micro:biten svare for dem.[25]

Elevene lar micro:biten velge svar på ja- og nei-spørsmål.

I den første delen av oppgaven skal elevene programmere. De benyttet det blokkbaserte systemet til micro:bit.[26] Det blokkbaserte systemet fungerer slik at blokkene settes sammen til større blokker når de programmerer, litt som å bygge Lego. Elevene møtte på flere problemstillinger i programmerings-økten: Hvordan skulle de få micro:biten til å variere mellom å svare ja og nei, hvilke variabler skulle de bruke, og hvor lang tid skulle svaret vises på skjermen ? Det la til rette for samtaler og diskusjoner rundt utfordringene de møtte, og bidro til at elevene i fellesskap fant løsninger for å komme videre.



Figur 7: Blokkprogrammering av micro:bit

I den andre delen av oppgaven testet elevene programmet de hadde programmert, på hverandre. Det ble gjort på mange ulike måter. Noen elever stilte hverandre faglige spørsmål og brukte den programmerte micro:biten til å svare. Andre elever brukte den til å spørre hverandre om hobbyer og interesser. Noen elever laget gameshow og brukte den programmerte micro:biten til dette.

7. Programmering på agendaen i Europa

European Schoolnet (EUN) startet i 2014 *The Coding Initiative* på oppdrag fra EU-kommisjonen. Målet er å fremme programmering og informatikk i Europa, blant annet via nettstedet «all you need is {C<3DE}»[27] – et nettsted for elever, lærere og andre som vil utforske programmeringens muligheter. Siden 2014 har «CodeWeek» satt programmering på agendaen en hel uke gjennom en serie arrangementer i ulike land.

European Schoolnet publiserte i oktober 2015 rapporten *Computing our future. Computer programming and coding – Priorities, school curricula and initiatives across Europe (2015 update).* Rapporten gir en oversikt over status for programmering i skolen i 21 europeiske land. Rapporten omhandler hvordan ulike land prioriterer programmering i skolen, både som del av pensum og andre initiativ. 21 land svarte på undersøkelsen, og svært kort oppsummert ser vi at:

- 16 land svarer at programmering allerede er del av pensum på nasjonalt eller regionalt nivå. Det gjelder Bulgaria, Danmark, England, Estland, Frankrike, Irland, Israel, Litauen, Malta, Polen, Portugal, Slovakia, Spania, Ungarn og Østerrike. England innførte programmering som obligatorisk del av opplæringen allerede høsten 2014, mens Frankrike, Spania og Polen har besluttet innføring i løpet av det siste året.
- Finland og den belgiske regionen Flandern har konkrete planer om å innføre programmering i pensum. I Finland gjøres dette med nye læreplaner som trer i kraft høsten 2016.
- Norge, Nederland og den belgiske regionen Vallonia har ikke programmering som del av pensum i grunnskolen eller konkrete planer om å innføre dette. Spørsmålet om programmering i skolen debatteres i alle tre landene.

Valgfagsforsøket i Norge var ikke igangsatt da undersøkelsen ble gjennomført, og er derfor ikke med i rapporten.

Sveriges regjering har også varslet at programmering skal inn i grunnskolens læreplaner. I den forbindelse leverte Skolverket i april 2016 et forslag til nasjonal IT-strategi for grunnskolen. I dette forslaget foreslår de endringer i læreplanverket som styrker og tydeliggjør programmeringens plass i opplæringen.[28]

Fra 2014 til 2015 viser rapportene tre trender:[29]

1. Programmering blir i økende grad sett på som en tverrfaglig kompetanse som integreres i eksisterende fag (ref. Finland), ikke som eget fag.
2. Programmering innføres på stadig lavere trinn, helt ned i barneskole.
3. Økt oppmerksomhet på algoritmisk tankegang (computational thinking) når man velger å ta programmering inn i skolen.

Nedenfor vil vi kort beskrive hvordan programmering er innført eller planlegges innført i skoler i England, Finland og Estland.

7.1 England – programmering som del av eget IKT-fag

England innførte programmering gjennomgående fra første trinn høsten 2014, i forbindelse med at det ble skrevet nye læreplaner i en rekke fag. Programmering ble da en sentral del av det nye faget *Computing* som erstattet det gamle faget *ICT*. Bakgrunnen for det nye faget var blant annet sterk kritikk fra utdanningssektoren og arbeidslivet om at faget dreide seg om å lære kontorstøtteverktøy, fremfor forståelse av datamaskiner og -systemer.

Rapporten *Next Gen Skills* fra 2011 (Hope & Livingstone, 2011) var svært viktig for beslutningen om å etablere ett nytt IKT-fag. Rapporten var bestilt av daværende minister for kultur, kommunikasjon og kreative industrier. Bestillingen var å svare på hva slags kompetanse de store britiske næringene dataspill og visuelle effekter hadde behov for i fremtiden, samt gi anbefalinger om hvordan disse behovene burde møtes. Rapporten slo fast at det var sterk ubalanse mellom hva utdannings-systemet la vekt på, og det industrien hadde behov for:

> Whilst useful in teaching various proprietary office software packages, ICT fails to inspire children to study computer programming. (…) In a world where technology affects everything in our daily lives, so few children are taught such an essential STEM skill as programming. Bored by ICT, young people do not see the potential of the digital creative industries. (Hope & Livingstone, 2011:29)

Det nye faget *Computing* setter konkrete mål for elevenes programmeringskompetanse på de ulike trinnene, fra å forstå hva en algoritme er, til å kunne benytte minst to programmeringsspråk og designe gode løsninger. I forbindelse med det nye faget er det utviklet en rekke ressurser for å støtte lærere i overgangen, og gratis online kurs (MOOCs) tilbys fra en rekke universiteter. Det er også delt ut gratis micro:bit til alle elever på sjuende trinn i England. Dette er en liten, programmerbar data-maskin som blant annet har tastatur, skjerm, innebygd bevegelsessensor og bluetoothteknologi. Denne lille datamaskinen kan brukes til programmering av elever og lærere i faget Computing.[30]

Kompetansemål fra det engelske faget *Computing* inkluderer blant annet:[31]

- Å forstå hva algoritmer er og hvordan programmer virker, og lage og de-bugge enkle programmer på 1. og 2. trinn.
- Å designe, skrive og feilsøke programmer som utfører spesifikke oppgaver, inkludert å kontrollere eller simulere fysiske systemer. Løse problemer ved å dele dem opp i mindre del-problemer, bruke logisk resonnement for å forklare hvordan algoritmer fungerer, og jobbe med variabler (3.–6. trinn).
- Å designe, bruke og evaluere abstraksjoner av virkelige problemer og fysiske systemer. Forstå ulike sentrale algorimer innen faget, slik som sortering og søk, og sammenlikne hvordan de kan brukes til å løse et problem. Bruke minst to programmeringsspråk, hvorav minst ett tekstbasert, til å løse ulike problemer og designe og utvikle modulære programmer som benytter prosedyrer og funksjoner (7.–9. trinn).
- Å utvikle og anvende sin kompetanse innen analyse, problemløsning, design og *computational thinking* (10.–11. trinn).

7.2 Finland – programmering som del av matematikkfaget fra 2016

Høsten 2016 innførte Finland nye læreplaner der programmering er tatt med i flere fag. I for eksempel den nye læreplanen for matematikkfaget blir programmering beskrevet som en aktivitet og et mål fra 1. til 9. trinn under hovedområdet «Matematisk tenkning og matematiske metoder». Kompetansemål for det finske matematikkfaget inkluderer:[32]

- På 1. og 2. trinn skal elevene lage stegvise instruksjoner som så testes (algoritmer).
- På 3. til 6. trinn skal elevene planlegge og lage et program i et visuelt programmeringsmiljø (for eksempel Scratch).
- På 7. til 9. trinn skal elevene (videre-)utvikle sin algoritmiske tenkning og sine programmeringsferdigheter ved å lage enkle programmer. Elevenes anvendelse av prinsipper for algoritmisk tenkning og programmeringsferdigheter teller ved sluttvurderingen etter ungdomsskolen.

7.3 Estland – teknologi som tverrfaglig emne

Det har vært mange medieoppslag[33] om Estlands satsing på programmering i skolen, og det skjer mye spennende i estiske skoler. Estland har likevel ikke innført programmering som obligatorisk skolefag, og det finnes ikke et programmeringspensum for estiske skoler.

Estland har en tverrfaglig tilnærming til IKT og annen teknologi i skolen, blant annet et tverrfaglig emne i grunnskolen kalt «Teknologi og innovasjon», som krever at lærere bruker teknologi i undervisningen i alle fag. Det stilles ikke krav til hva slags teknologi som skal benyttes, det er opp til lærerne selv å bestemme.

Stiftelsen HITSA (The Information Technology Foundation for Education)[34] har som oppdrag å sikre at dem som går ut av estisk utdanning, har den nødvendige digitale kompetansen for å utvikle økonomien og samfunnet. Programmet ProgeTiger (programmering+tiger) ble lansert i 2012 og lanserte tanken om å undervise i programmering og robotikk i skolene.

HITSA har utviklet en rekke ressurser og læringsopplegg knyttet til programmering i skolen og driver også utstrakt kompetanseheving av lærere på området.

8. Programmering i norsk skole

Programmering er i skrivende stund ikke obligatorisk del av norsk grunnopplæring, med unntak av enkelte studieretninger i videregående skole. Høsten 2016 startet om lag 150 skoler fordelt på 53 kommuner opp med forsøk med programmering som valgfag på ungdomstrinnet. Det er laget en egen forsøkslæreplan for faget, som gjelder for skolene som deltar i forsøket. I læreplanen vektlegges at elevene gjennom faget skal utvikle sin algoritmiske tankegang.

Selv om programmering ikke er spesifikt skrevet inn med kompetansemål i læreplanene, har en rekke skoler funnet måter å innføre programmering i skolen på . I valgfaget Teknologi i praksis er programmering et naturlig verktøy å bruke for å nå mange av kompetansemålene. I realfag som matte, naturfag og fysikk kan programmering også innføres for å dekke kompetansemål som for eksempel koordinater, vinkler og sannsynlighet. Det er også vanlig at fag har kompetansemål som handler om å presentere noe digitalt, og her kan et programmeringsspråk som Scratch være et spennende alternativ til PowerPoint.

Selv om slik bruk av programmering i ulike fag forekommer, er det liten grunn til å tro at det foregår i stor utstrekning. Langt vanligere er det imidlertid at elever får en liten smakebit på programmering gjennom den nevnte kodetimen, der over 50 000 elever deltok i 2015. I tillegg kommer Kodeklubber som ofte arrangeres i samarbeid med skolen. SFO (Skolefritidsordning) er også en arena der det har blitt arrangert kodeklubber i nær forbindelse med skolen.[36]

I Oslo kommune vil det fra høsten 2017 bli gitt tilbud til alle fjerdeklassinger om å gå på kodeklubb etter skoletid.[37] Denne kodeklubben arrangeres i AKS-tiden (Oslos navn på SFO), men omfatter alle elever og er gratis å gå på. Instruktørene vil være ungdomsskoleelever som læres opp og lønnes av Oslo kommune.

Noen kommuner i Norge har valgt å tenke enda større og vil innføre programmering gjennomgående i grunnskolen.[38] Eksempelvis har Ulstein kommune planlagt å starte opp med programmering på alle trinn fra 1. til 10. høsten 2017, og vil gi nødvendig videreutdanning til alle berørte lærere i samarbeid med den lokale høgskolen.

8.1 Teknologi og programmering for alle

En ekstern gruppe satt ned av Utdanningsdirektoratet leverte høsten 2016 rapporten *Teknologi og programmering for alle*.[39] Her argumenterer de for at integrering av programmering i de etablerte skolefagene ikke er godt nok, og anbefaler at det opprettes et nytt obligatorisk fag som omhandler teknologi og programmering. «Teknologi er et eget kunnskapsområde med praktiske og kreative elementer, og elever trenger kompetanse i teknologi på egne premisser og ikke kun for å lære andre fag.» (ref. s. 75) Teknologi og programmering integrert i eksisterende fag vil ifølge rapporten risikere å bli systematisk nedprioritert, både fordi lærerne ikke opplever å ha nok kompetanse, og fordi det ikke passer naturlig inn i fagets kultur.

Hvis derimot elevene først lærer programmering i et eget fag, vil det være lettere å integrere det i de etablerte fagene: «Å trekke teknologi inn i det faglige innholdet i realfagene (og andre fag) kan synes ambisiøst, men dette er fullt mulig så sant alle elever får en praktisk innføring i programmering. Dette vil danne grunnlag for endret praksis, særlig i matematikk.» (ref. s. 76)

En slik bruk av ferdighetene fra det dedikerte teknologi- og programmeringsfaget inn i de etablerte fagene vil ikke være like praktisk gjennomførbart hvis teknologi og programmering hovedsakelig læres i valgfagene. En relatert bekymring er den tradisjonelle skjeve kjønnsfordelingen i teknologiorienterte valgfag.

9. Avslutning

Programmering integrert i undervisning i norske skoler kan gjøres på ulike måter: enten ved at det opprettes et eget fag der programmering inngår, ved at programmering integreres i eksisterende fag, eller ved at programmering defineres som en grunnleggende ferdighet (fagovergripende kompetanse). England er eksempel på et land som har gjort det første, mens Finland har valgt å integrere programmering i ordinære fag.

På kort sikt er det mest realistisk å jobbe med programmering innenfor eksisterende fag og læreplaner, eventuelt supplert med nye valgfag i ungdomsskole og videregående skole. På litt lengre sikt kan programmering knyttes til egne kompetansemål i fellesfagene, noe som vil sikre en både dypere forståelse og en bredere bruk.

Det er mange fag i skolen der programmeringskompetanse kan være nyttig og bidra til å gjøre faget mer relevant og motiverende; realfagene er åpenbare kandidater for programmering, men vi mener mulighetene er minst like store i de estetiske fagene. Programmering er et språk som muliggjør visuelle, musikalske og kunstneriske uttrykk, i tillegg til regneoperasjoner, algoritmer og måleinstrumenter.

På lengre sikt bør programmering, som *Teknologi og programmering for alle* -rapporten anbefaler, innføres som eget obligatorisk fag i grunnskolen. Dette vil sikre at alle elever får en grunnleggende opplæring i programmering, noe som igjen vil gi mulighet til faglig fornyelse av de øvrige skolefagene. Senter for IKT i utdanningen stiller seg bak denne anbefalingen.

Sluttnoter

1. code.org/about
2. Coding – the 21st century skill (Digital Agenda for Europe): https://ec.europa.eu/digital-agenda/en/coding-21st-century-skill
3. For eksempel Fellesverkstedet, Bitraf og Teknoteket i Oslo, Verkstedet ved Bergen Offentlige Bibliotek, Skaperrommet i Trondheim og mange flere. En oversikt over makerspaces i Norge finnes på: http://norwaymakers.org/kart
4. Oversikt over kodeklubber på Lær Kidsa Programmering: www.kidsakoder.no/kodeklubben/kodeklubboversikt/
5. Oversikt over deltakere på kodetimen: http://kidsakoder.no/kodetimen/
6. Forsøk med programmering som valgfag: http://www.udir.no/kvalitet-og-kompetanse/nasjonale-satsinger/realfagsstrategien/forsok-med-programmering-som-valgfag/
7. ProgrammeringsMOOC: https://iktsenteret.no/prosjekter/programmeringsmooc
8. Eksempler på frivillige initiativ for å undervise programmering inkluderer Coder Dojo som oppsto i Irland, og som nå finnes nå i 57 land (www.coderdojo.com), Code Club UK (www.codeclub.org.uk) og danske Coding Pirates (codingpirates.dk/).
9. Scratch fra MIT er gratis, finnes på norsk og kjører i nettleseren: https://scratch.mit.edu/
10. MIT App Inventor er et gratis verktøy for programmering av apper til mobil og nettbrett (Android), beregnet på nybegynnere: http://appinventor.mit.edu/
11. Code Studio er en samling nettbaserte læringsressurser fra Code.org for å lære og undervise programmering på alle nivå og for alle aldersgrupper. Alle ressursene er gratis tilgjengelige, og mye av materialet er oversatt til norsk: https://studio.code.org/
12. http://www.huffingtonpost.com/kiki-prottsman/coding-vs-programming-bat_b_7042816.html
13. Se for eksempel https://blogg.regjeringen.no/fremtidensskole/2013/12/10/32/
14. http://www.atc21s.org/
15. For eksempel http://e24.no/jobb/norge-mangler-10-000-ingenioerer/20328408
16. Se for eksempel Nasjonal strategi for realfag i barnehagen og grunnopplæringen (2015–2019).
17. Artikkel fra Teknologirådet om automatisering av arbeidsplasser: http://teknologiradet.no/norge-2030/halvparten-av-jobbene-kan-forsvinne/
18. Se for eksempel Nasjonal handlingsplan for e-helse 2014–2016.
19. The Barefoot Project: http://barefootcas.org.uk/
20. Teknologi og programmering for alle: http://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf
21. Se for eksempel Agile Alliance om parprogrammering: (http://guide.agilealliance.org/guide/pairing.html) og Barefoot project om computational thinking: http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/collaborating/

22. https://scratch.mit.edu/projects/109322386/
23. https://scratch.mit.edu/projects/119139272/
24. https://scratch.mit.edu/projects/24368107/
25. http://microbit.org
26. https://www.microbit.co.uk/blocks/contents
27. Det europeiske kodeinitiativet, eller «all you need is {C<3DE}» har som mål å fremme programmering og computational thinking på alle nivå i utdanningen: http://www.allyouneediscode.eu/
28. http://www.skolverket.se/publikationer?id=3621
29. http://goo.gl/GrgWIv
30. http://www.bbc.co.uk/programmes/articles/4hVG2Br1W1LKCmw8nSm9WnQ/the-bbc-micro-bit
31. https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study
32. De finske læreplanene for matematikkfaget i grunnskolen (på svensk): https://eperusteet.opintopolku.fi/#/sv/perusopetus/419550/sisallot/466344
33. Se for eksempel http://www.bbc.com/news/education-25648769 og http://www.nytimes.com/2014/03/24/world/europe/adding-coding-to-the-curriculum.html?_r=0
34. Mer informasjon om HITSA og ProgeTiger finnes her: http://www.hitsa.ee/it-education/educational-programmes/progetiger
35. Se artikkel om Espen Clausens prosjekt med ett år med programmering på Grinde skule: http://kidsakoder.no/2016/06/01/erfaringer-ar-koding/
36. Se artikkel om Torbjørn Skaulis prosjekt med ungdomsskoleelever som underviser fjerdeklasse på AKS: http://kidsakoder.no/2013/03/25/la-ungdommen-undervise-kidsa/
37. http://www.dn.no/talent/2016/02/04/1259/Torbjrn-Relsaksen/n-skal-alle-10ringer-i-oslo-f-gratis-kodekurs
38. http://kidsakoder.no/2015/10/30/teknologiskulen-i-ulstein-med-koding-pa-timeplanen/
39. http://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Referanser

Binkley, M., Erstad, O., Hermna, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). *Defining Twenty-First Century Skills.* In Griffin, P., Care, E., & McGaw, B. Assessment and Teaching of 21st Century Skills, Dordrecht, Springer.

Departement for Education (2013). *National curriculum in England: computing programmes of study.* Hentet fra: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

European Schoolnet (2014). *Computing our future Computer programming and coding - Priorities, school curricula and initiatives across Europe.* http://www.eun.org/publications/detail?publicationID=481

European Schoolnet (2015). *Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe (2015 update).*
http://www.eun.org/publications/detail?publicationID=661

Frey, C.B. og Osborne, M. (2013) T*he Future of Employment: How susceptible are jobs to computerisation?* http://www.oxfordmartin.ox.ac.uk/publications/view/1314 - søk foretatt 1.9.2015

Hope, A. og Livingstone, I. (2011) *Next Gen:Transforming the UK into the world's leading talent hub for the video games and visual effects industries.* http://www.nesta.org.uk/publications/next-gen - søk foretatt 30.9.2015

Johnson, L., Adams Becker, S., Estrada, V., Freeman, A., Kampylis, P., Vuorikari, R., and Punie, Y. (2014). *Horizon Report Europe: 2014 Schools Edition.* Luxembourg: Publications Office of the European Union. http://www.nmc.org/publication/nmc-horizon-report-europe-2014-schools-edition-2/

Johnson, L., Adams Becker, S., and Hall, C. (2015). *2015 NMC Technology Outlook for Scandinavian Schools: A Horizon Project Regional Report.* Austin, Texas: The New Media Consortium.http://www.nmc.org/publication/2015-nmc-technology-outlook-scandinavian-schools/

NOU 2013:2 (2013). *Hindre for digital verdiskaping. Utredning fra Digitutvalget oppnevnt i statsråd 24. juni 2011 Avgitt til Fornyings-, administrasjons- og kirkedepartementet 7. januar 2013.* Oslo: Administrasjons- og kirkedepartementet. Hentet fra http://www.regjeringen.no

NOU 2015:8 (2015). *Fremtidens skole — Fornyelse av fag og kompetanser. Utredning fra et utvalg oppnevnt ved kongelig resolusjon 21. juni 2013. Avgitt til Kunnskapsdepartementet 15. juni 2015.* Oslo: Kunnskapsdepartementet. Hentet fra http://www.regjeringen.no
Papert, Seymour (1993a). *Mindstorms: Children, computers and powerful Ideas* (2. utg.). New York: BasicBooks.

Papert, Seymour (1993a). *Mindstorms: Children, computers and powerful Ideas* (2. utg.). New York: Basic-Books.

Papert, Seymour (1993b). *The children's machine: rethinking school in the age of the computer*. New York: BasicBooks.

Stephenson, Chris; Valerie Barr (May 2011). "Defining Computational Thinking for K-12". CSTA Voice 7 (2): 3–4.

The Research Institute of the Finnish Economy (2014). *Computerization Threatens One Third of Finnish Employment.* http://www.etla.fi/julkaisut/computerization-threatens-finnish-employment/ - søk foretatt 30.9.2015

SENTER
FOR IKT I
UTDANNINGEN

www.iktsenteret.no