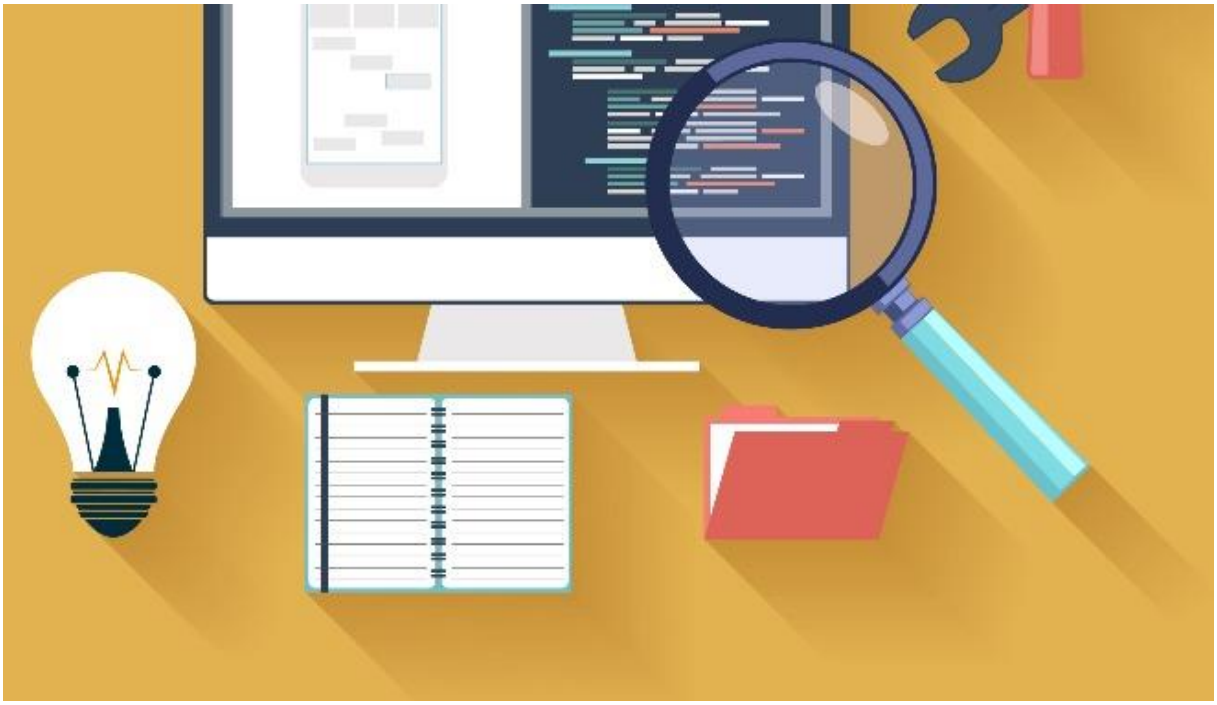
Realfaglig programmering

av Cathrine Wahlstrøm Tellefsen

Hentet fra https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/realfaglig-programmering/index.html

Med fagfornyelsen blir programmering obligatorisk i skolen. Realfaglig programmering handler om problemløsing, algoritmer, logisk tenkning og argumentasjon. Matematikk får et hovedansvar for opplæringen, men programmering kommer inn i flere fag. Kunnskap om numeriske metoder og realfaglig programmering åpner for å utforske fagene mer i dybden.



Realfaglig programmering dreier seg om programmering i matematikk og de naturvitenskapelige fagene. Det handler om å ta utgangspunkt i en realfaglig problemstilling og løse den ved hjelp av et dataprogram. Da må man bryte problemstillingen opp i sine enkeltdeler og formulere hver del svært presist i en logisk rekkefølge. I tillegg vil man i realfaglig programmering få bruk for kunnskap om numeriske metoder. Numeriske metoder er matematiske løsningsteknikker tilpasset datamaskinen. Disse teknikkene er ofte bygget rundt en enkel idé som repeteres mange ganger og har som regel langt mer generell gyldighet enn tilsvarende tradisjonelle, matematiske teknikker. Det betyr i praksis at numeriske metoder utvider den matematiske verktøykassa i betydelig grad og gir mange og mye muligheter for å

løse realfaglige problemstillinger. Samtidig har numeriske metoder også sine begrensninger. Vi må være klar over begge deler.

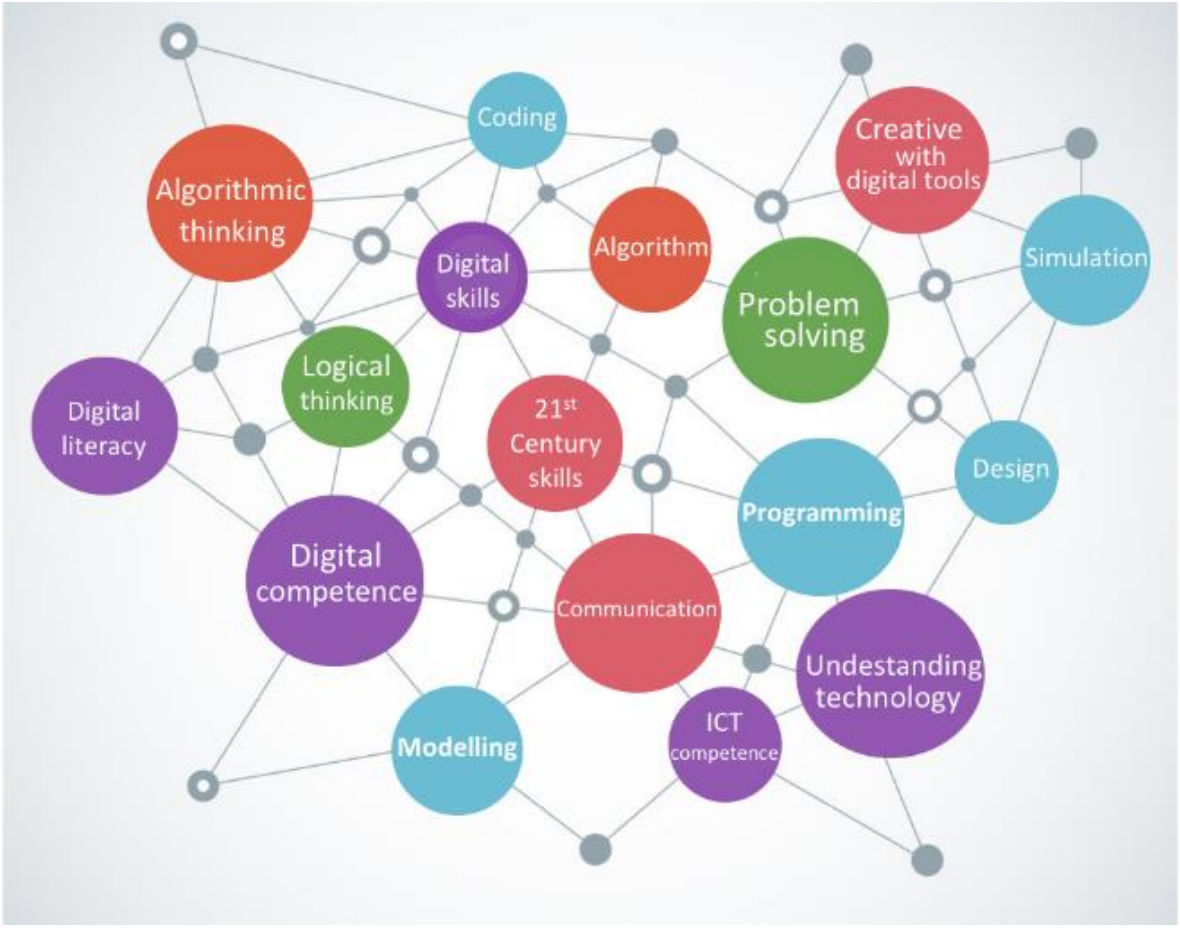
La oss bli litt konkrete med en gang! I fysikk pleier vi å se bort fra luftmotstand. Ikke fordi det er realistisk, men fordi det er lettere matematisk. Elevenes matematikkunnskaper hindrer oss i å velge mer virkelighetsnære problemstillinger. Med numeriske metoder kan vi enkelt inkludere luftmotstanden uten at elevene må lære å løse differensiallikninger først. Dermed kan vi modellere bevegelsen til fallende gjenstander, for eksempel en muffinsform. Vi kan variere parametere og sammenlikne med reelle eksperimenter. Det åpner opp for kreativitet og utforskende arbeidsmetoder. Og fordi det ikke er noe fasitsvar, åpner vi også for samarbeid og diskusjon om de ulike løsningene elevene kommer fram til. På denne måten kan vi legge til rette for dybdelæring i praksis.

Digital kompetanse i realfag

I norsk skole fikk vi med kunnskapsløftet [2] definert både lesing, skriving og regning som grunnleggende ferdigheter. I tillegg kommer muntlige og digitale ferdigheter. Når det gjelder tolkningen av grunnleggende digitale ferdigheter i dagens skole, blir det påpekt at disse i for stor grad «vektlegger verktøyaspektet ved digital kompetanse og i for liten grad hvordan digitale verktøy og medier er en integrert del av det elevene skal lære i fagene og på tvers av fag» [3]. Vi kan se på det som en slags passiv brukertilnærming. Vi skal lære å bruke digitale verktøy, men noen andre må lage dem. Med realfaglig programmering blir vi aktive brukere. Vi skaper selv de digitale løsningene og trenes i algoritmisk tenkning, logikk og problemløsing. Og vi gjør det i en faglig kontekst.

Rapporten *The nordic approach to introducing computational thinking and programming in compulsory education* [4], ser nærmere på utdanningspolitiske dokumenter om algoritmisk tenkning og programmering i de nordiske landene. Begrepet algoritmisk tenkning er oversatt fra det engelske Computational Thinking (CT). CT blir ikke brukt direkte i noen av landene (se Figur 1), men hovedforståelsen kan likevel sies å være todelt:

1) Å løse problemer/oppgaver
2) Digital kompetanse (som i å skape digitale løsninger og være en kritisk bruker)



Figur 1 CT-relaterte termer brukt i utdanningspolitisk dokumentasjon i de nordiske landene [4]. Figur hentet fra rapporten The nordic approach to introducing computational thinking and programming in compulsory education.

Problemløsing, algoritmer, logisk tenkning og argumentasjon har alltid vært en sentral del av matematikkfaget, slik det er en sentral del av realfaglig programmering. Når matematikkfaget blir hovedansvarlig for programmering i skolen er altså ikke dette tilfeldig. Det nye er å bruke datamaskinen mer aktivt i denne prosessen. Programmering skal ikke bare bli et verktøy, men har mulighet til å gi en faglig tilnærming som kan bidra til dybdeforståelse og utvikling av elevenes "computational literacy".

Computational literacy er et engelskspråklig begrep som er mer beskrivende enn vår litt utvannede bruk av digitale ferdigheter. Enkelte sidestiller literacy med grunnleggende ferdigheter innenfor et område, men i NOU om Fremtidens skole [3] kritiseres sidestillingen av "literacy" og grunnleggende ferdigheter ved å påpeke at literacy i stor grad også dreier seg om kompetanser og ikke bare ferdigheter.

Andrea diSessa er professor ved Berkeley, California. Hans forskergruppe har undersøkt hvordan elever i 6. klasse kan arbeide med matematikk knyttet til bevegelse (mekanikk). Ved bruk av diskret matematikk (numerisk kalkulus) og vektorfunksjoner, tilnærmet elevene seg mekanikken gjennom å arbeide med et sett med verdier istedenfor funksjoner. Dersom vi skal gjøre dette «for hånd» er det altfor tidkrevende, men med en datamaskin er det enkelt å studere mekanikkproblemer som et sett med samhørende verdier av tid, posisjon, fart og akselerasjon. I en artikkel publisert i tidsskriftet «Mathematical thinking and learning» i 2018 [5] skriver han

"I view computation as, potentially, providing a new, deep, and profoundly influential literacy—computational literacy—that will impact all STEM disciplines at their very core, but most especially in terms of learning."

STEM: Science, Technology, Engineering and Mathematics

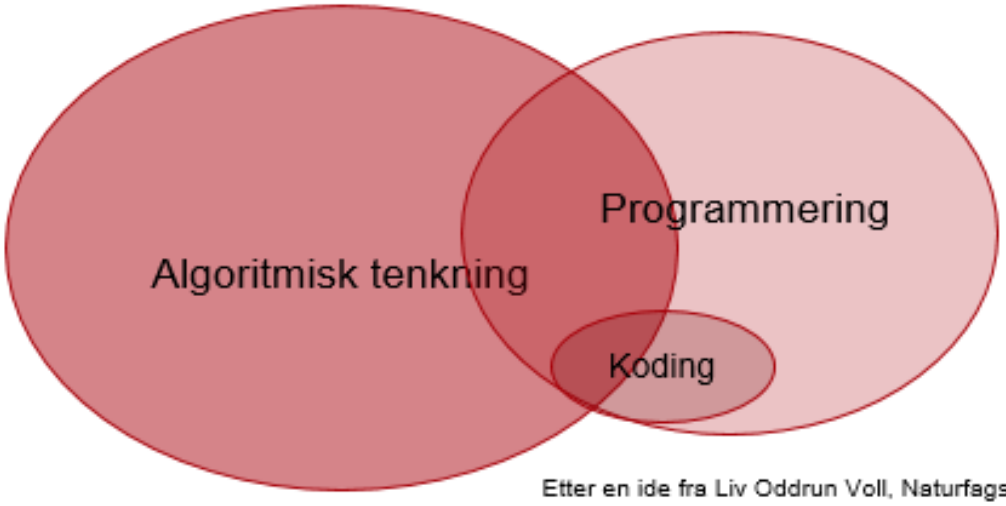
Digital kompetanse i realfag kan altså anses å være mer omfattende enn dagens forståelse av digitale ferdigheter, og innebærer blant annet en mer aktiv brukertilnærming og kompetanse i realfaglig programmering. Programmering er i stor grad knyttet til det engelske begrepet Computational Thinking (CT) som vi utdyper i neste avsnitt.

Programmering og algoritmisk tenkning

Programmering generelt og realfaglig programmering spesielt, er i stor grad knyttet til Computational Thinking (CT). Jeanette Wing er professor i informatikk ved Columbia University. Hun ser på CT som en ferdighet på linje med lesing, skriving og regning [7].

"(…) computer science offers not just useful software and hardware artifacts, but also an intellectual framework for thinking, what I call “computational thinking”. Everyone can benefit from thinking computationally. My grand vision is that computational thinking will be a fundamental skill—just like reading, writing, and arithmetic—used by everyone by the middle of the 21st Century."

CT handler om en måte å tenke på som er nyttig på mange områder, og både programmering og realfaglig programmering står sentralt i utviklingen av denne ferdigheten. Ofte oversetter vi CT til algoritmisk tenkning selv om det ikke er helt dekkende. Algoritmisk tenkning utgjør en kjerne i naturvitenskapelig praksis og i matematikk, og i skolen har vi lenge jobbet med å utvikle elevenes evne til algoritmisk tenkning. Figuren viser hvordan programmering både innebærer algoritmisk tenkning og koding, men også mye mer.



I artikkelen Defining Computational Thinking for Mathematics and Science Classrooms [8] blir vi presentert for en firedelt taksonomi for CT:

1) Dataferdigheter (samle, produsere, manipulere, analysere og visualisere data)
2) Modellerings- og simuleringsferdigheter
3) Ferdigheter i computational problem solving (herunder programmering, feilsøking og abstraksjon)

4) Systemforståelse (se sammenhenger, organisere ideer hierarkisk, beherske komplekse problemstillinger)

Hvis vi knytter dette til vårt konkrete eksempel med muffinsformer og luftmotstand, vil innsamling og presentasjon av reelle data være på nivå 1. Dette blir allerede gjort i norske klasserom i dag. Utvikling av en modell for fallet og en simulering av dette er på nivå 2. Modellen vil inneholde en differensiallikning som beskriver sammenhengen mellom fart og akselerasjon for den fallende muffinsformen. På nivå 3 kan elevene bruke Eulers metode for å løse differensiallikningen numerisk. De må abstrahere fra den konkrete situasjonen med en gjenstand som faller til hvordan dette kan representeres som et problem vi løser i en løkke der vi antar at for hver runde i løkken kan vi tilnærmet si at akselerasjonen er konstant. En fallende muffinsform er ikke et kompleks system, så vårt enkle eksempel er ikke på nivå 4 i denne taksonomien.

Skole handler om at barn og unge skal lære. De skal utvikle sin fagkunnskap, sine tenkemåter og sine evner til kritisk refleksjon og samarbeid. Programmeringen skal styrke denne fagopplæringen. Elevene skal ikke programmere for programmeringens skyld, men for fagets skyld. På denne måten kan realfaglig programmering bidra til dybdelæring, kreativitet, systemforståelse og tverrfaglighet – helt i tråd med intensjonene i Stortingsmelding 28.

Styrking av fagene

Realfaglig programmering er tett knyttet til dybdelæring og hva vi vet om hvordan ulike representasjonsformer kan hjelpe elevene i læringsprosessen. En som er ekspert innenfor sitt fagområde har mange koblinger mellom ulike fagbegreper og en dyp forståelse av faget, og det er lett å ta denne kunnskapen for gitt. Men elevene kan mangle både begrepsforståelse, ha misoppfatninger og mangle koblinger mellom begrepene og kunnskapsområdene. Gjennom nok tid og ulike representasjonsformer kan vi oppnå at elevene gradvis og over tid utvikler sin forståelse av begreper og sammenhenger innenfor et fag.

La oss konkretisere og se på likninger, nærmere bestemt løsing av andregradslikninger. Vi trenger ikke numeriske metoder for å løse en andregradslikning. I VG1 lærer elevene å løse

andregradslikninger ved å sette inn i den såkalte abc-formelen:

$$ax^2 + bx + c = 0 \text{ har løsningen } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Hvis elevene kan lage et program som løser andregradslikninger med abc-formelen, vil dette kunne gi en dypere forståelse av selve formelen. Hvorfor «kræsjer» programmet når radikanden (det som står under rottegnet) får en negativ verdi? Er det en feil i programmet eller har det en annen forklaring? Hvorfor er det to løsninger noen ganger og bare én andre ganger? Her åpnes det for diskusjon og argumentasjon på en annen måte enn å bruke formelen gjentatte ganger for å finne et fasitsvar.

Realfaglig programmering kan gi økt læringsutbytte ved at elevene utvikler en dypere forståelse gjennom eksperimentering, analyse og argumentasjon. De blir trent i å anvende faget i nye sammenhenger og med mer realistiske problemstillinger. Dette kan motivere flere elever enn de som lar seg motivere av programmering for programmeringens skyld. Elevene kan studere problemstillinger som tidligere bare var tilgjengelig for de med best evner i matematikk eller på universitetsnivå. Dette fordrer at lærerne evner å gi en ordentlig opplæring i realfaglig programmering innenfor skolefagenes rammer.

Vi avslutter denne delen med å se på naturvitenskapelige fag. I skolen dreier det seg om naturfag, kjemi, biologi, geofag og fysikk. Disse fagene bruker matematikk som et verktøy. Naturvitenskapelige fag bruker ofte modeller for å forklare fenomener i naturen. Elever trenger trening i å forstå at vi kan ha mange ulike modeller for å beskrive ett og samme fenomen. Og de trenger trening i å forstå gyldighetsområdet til modellene og begrensningene i dem. Naturvitenskapelig programmering gjør at vi lettere kan lage våre egne modeller, variere parametere, sammenlikne med eksperimenter og på denne måten få en bedre forståelse av selve modelltenkningen og av den naturvitenskapelige tenkemåte og arbeidsform.

Vi vet at mange elever blir mer motivert for å lære matematikk dersom det «kan brukes til noe» som f.eks. i en naturvitenskapelig kontekst. Vi vet også at jenter legger mer vekt på

nytten av realfag enn det guttene gjør når de skal velge fag [9]. Det er derfor ikke urimelig å anta at programmering i en realfaglig kontekst kan være motiverende for et mangfold av elever.

Utvikling av fagene

Telling og tallregning har stått sentralt i menneskehetens historie gjennom tusener av år. Automatisering av slike prosesser har derfor en lang historie som har dannet grunnlaget for utviklingen av matematikken. I tillegg har matematikk også blitt det universelle språket for naturvitenskapelige fag. Matematikken har gjennom historien benyttet ulike verktøy for å forenkle og automatisere tallregning. Noen opplagte eksempler er fingrene (som har gitt opphav til titallsystemet for representasjon av tall), kuleramme og abakus, regnestav og kalkulator. Det siste tilskuddet til skolematematikken er bruk av Computer Algebra Systems (CAS).

I dag har alle elever i videregående skole egne datamaskiner. Også i grunnskolen har mange elever tilgang til datamaskiner og nettbrett. Vi trenger ikke lenger la oss begrense av de matematiske metodene som har vært brukt de siste 300 årene. CAS tar i bruk datamaskinen, men matematikken ligger skjult i et system noen andre har utviklet. Dette stimulerer i mindre grad til algoritmisk tenkning, modellering og kreativitet.

Vi tar et eksempel fra matematikken. Elever lærer ulike fremgangsmåter for å løse likninger både på ungdomsskolen og videregående skole. Men hva betyr det å løse en likning?

La oss ta noe så enkelt som $2x + 4 = 10$. Det er mange mulige fremgangsmåter for å løse denne. Vi kan f.eks. trekke fra 10 på hver side så vi får $2x – 6 = 0$. Og så kan vi finne ut hvilken verdi av x som fører til at venstre side blir null. Numerisk løsning av likninger gjør nettopp dette ved hjelp av noe som kalles halveringsmetoden. Vi skal ikke gå gjennom metoden her, men det er litt tungvint å løse $2x + 4 = 10$ med denne metoden. Vi ser jo raskt at svaret blir $x = 3$. Poenget er at halveringsmetoden kan brukes på ALLE likninger. Ta for eksempel denne ligningen:

$$2\ln(x^4+4)-\frac{1}{2}x=0$$

Den kan vi ikke løse analytisk. Men vi kan løse den numerisk ved hjelp av halveringsmetoden. Da beregner vi en god tilnærming til løsningen heller enn å finne den eksakte løsningen. Slike numeriske løsninger kan vi så og si alltid gjennomføre bare vi regner lenge nok, mens eksakte løsninger kan vi bare finne i noen spesialtilfeller. Med bruk av numeriske metoder i realfaglig programmering blir elevenes matematiske verktøykasse betydelig utvidet.

Hvert år deles Holmboeprisen ut av Norsk matematikkråd. I 2017 gikk prisen til Hanan Abdelrahman ved Lofsrud ungdomsskole. Hun vil ha mer vekt på forståelse og dybdekunnskap i matematikkfaget og ikke bare overflatisk pugging. Hun uttaler blant annet:

«Matte er tradisjonelt kjent som «tørr gørr». Hvis elevene også lærer programmering, kan logisk tenkning i programmering støtte mattefaget og styrke det tenkende, kreative og problemløsende aspektet ved det. Da ser man sammenhenger og mønstre, og ser at det faktisk kan brukes til noe. Da blir matte noe annet enn bare et sett med regler og prosedyrer man må pugge på skolen.»

La oss gå tilbake til de fallende muffinsformene. Differensiallikninger som tradisjonelt har blitt introdusert helt på slutten av det 13-årige skoleløpet kan nå løses numerisk. Elevene må forstå sammenhengen mellom posisjon, fart og akselerasjon og de må kunne Newtons lover. Og så må de kunne litt programmering. Det er mye faglig igjen i oppgaven – både fysikk og matematikk – men matematikken er ikke lenger en like stor hindring. Med numeriske metoder og fagkunnskap i matematikk og fysikk kan vi la elevene både ta med luftmotstand og teste ut mange ulike modeller for luftmotstand.

Modellering og naturvitenskapelig metode

Utviklingen innen naturvitenskapelige og matematiske fag har de siste 50 årene dreid seg mye om hvordan datamaskinen kan brukes til å løse komplekse og virkelighetsnære problemstillinger. Denne utviklingen speiles ikke i skolen, men det trenger ikke være slik. Skolefagene står i en tradisjon der de skal både være kulturbærende og nyskapende.

Datateknologi har revolusjonert både forskning og den øvrige samfunnsutviklingen. Tunge numeriske beregninger og store datasett er ikke lenger en begrensende faktor. Biologene kartlegger det menneskelige genom og driver hjerneforskning ved å simulere milliarder av hjerneceller. Astrofysikerne simulerer supernovaeksplosjoner og produksjon av grunnstoff ved hjelp av tunge databeregninger. Klimamodeller, jordskjelvvarslinger og medisinske avbildningsteknikker er resultater av menneskers kompetanse i realfaglig programmering. Programmering åpner for at vi også i skolen kan studere mer virkelighetsnære problemstillinger og arbeide tettere opp mot forskningsfagene ved å lage modeller som vi tester mot praktiske eksperimenter. Fordi modeller ikke har fasitsvar, åpner denne arbeidsformen for kreativitet og samarbeid.

Ofte er det tungvint og tidkrevende å gjøre mange endringer i et reelt eksperiment, men en modell i form av et dataprogram kan man lett endre og «eksperimentere» med. Så kan elevene heller bruke tiden til faglig argumentasjon og resonnering, i tråd med Holmboe-prisvinner Abdelrahmans visjon for morgendagens matematikkfag.

Undervisning i realfaglig programmering

Å lære å programmere i en realfaglig kontekst

Tradisjonelt har grunnopplæring i programmering vært fristilt fra en realfaglig kontekst. På samme måte som matematikk, er programmering både knyttet til et disiplinfag (informatikk) og et verktøy for andre fag. Men koblingen mellom programmering og skolefaget/disiplinfaget er sjelden. Unntaket er Universitetet i Oslo som de siste 15 årene har vært verdensledende i å integrere programmering og numeriske metoder i realfagene helt fra

første semester [10]. Studentene har med denne satsingen fått mulighet til å løse og forstå flere ulike naturvitenskapelige problemstillinger enn tidligere, og fått en utdanning som er mer tilpasset kravende fra dagens og morgensdagens arbeidsliv. UiOs erfaringer med innføring av realfaglig programmering på begynneremner er et viktig fundament når vi nå skal i gang med etterutdanning av lærere. Samtidig må vi huske på at skolen både har yngre elever og et større mangfold enn disiplinstudiene ved UiO.

Programmering i skolen skal bidra til en styrket fagopplæring i realfagene. Innholdet i fagene vil bli endret og elevene vil få en utdanning som ruster dem til å møte fremtidens kompetansebehov. Dette fordrer lærere som ikke bare kan programmere, men også kan tilrettelegge for læring i realfaglig programmering — ikke bare for de som allerede kan programmere, men for alle elever. Dette er en betydelig utfordring fordi fagdidaktikk rundt programmering er et lite utviklet fagområde både nasjonalt og internasjonalt. Dermed blir erfaringene fra 15 år med realfaglig programmering ved UiO svært sentrale. I tillegg må vi som tilbyr etterutdanning jobbe tett med lærerne for å utvikle en bærekraftig modell for etterutdanning i realfaglig programmering.

Hvilket programmeringsspråk skal vi bruke?

Valg av programmeringsspråk komme an på hva du ønsker å oppnå. Diskusjonen om språk handler dessverre ofte om hva enkeltpersoner foretrekker – kanskje i sin egen forskning eller annen personlig bruk. Men i skolen skal dette handle om opplæring i realfaglig programmering, programmeringsspråket er bare et verktøy i denne prosessen. Basert på våre erfaringer ved UiO gir vi likevel noen argumenter for å bruke Python som programmeringsspråk i ungdomsskolen og videregående skole.

1. Python gir gode programmeringsvaner, blant annet gjennom god støtte for enhets-tester
2. Python er gratis
3. Python har god støtte for moderne statistikk (data science og maskinlæring).
4. Python har god støtte for objektorientering
5. Python er et språk som brukes av mange i «virkeligheten» (altså ikke bare i opplæring)

Referanser

1. Regjeringen.no. *Fornyer innholdet i skolen.* 2018 [cited 2018 15/10]; Om kjerneelementene i fagfornyelsen]. Hentet fra: https://www.regjeringen.no/no/aktuelt/fornyer-innholdet-i-skolen/id2606028/.
2. Utdanningsdirektoratet. *Grunnleggende ferdigheter*. 2006 [cited 2018 15/10]; Hentet fra: https://www.udir.no/laring-og-trivsel/lareplanverket/grunnleggende-ferdigheter/.
3. Kunnskapsdepartementet, *NOU Fremtidens skole*. 2015.
4. Bocconi, S., A. Chioccariello, and J. Earp, *The Nordic apporach to introducing Computational Thinking and programming in compulsory education*. , in Report prepared for the the Nordic@BETT2018 Steering Group. 2018.
5. diSessa, A.A., *Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. Mathematical Thinking and Learning*, 2018. 20(1): p. 3-31.
6. Utdanningsdirektoratet. *Forsøkslæreplan i programmering og modellering X 2017* [cited 2018 24.02]; Hentet fra: https://www.udir.no/kl06/PRM1-01.
7. Wing, J.M., *Computational thinking's influence on research and education for all*. 2017, 2017. 25(2): p. 8.
8. Weintrop, D., et al., *Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology*, 2016. 25(1): p. 127-147.
9. Bøe, M.V., *Science choices in Norwegian upper secondary school: What matters? Science Education*, 2012. 96(1): p. 1-20.
10. Malthe-Sørenssen, A., et al., *Integrasjon av beregninger i fysikkundervisningen*. Uniped, 2015. 38(04): p. 303-310.