Maugesten, Stigberg, Stigberg

Programmering på ungdomstrinnet

Hvordan skal lærere undervise programmering i matematikk slik at elevene ser sammenhengen mellom matematikk og programmering? Det er en av utfordringene når programmering er blitt en del av LK20. I denne artikkelen gir vi et eksempel på en arbeidsmåte som kan bidra til å koble programmering tettere til det matematiske innholdet. Vi har gjennomført et utviklingsprosjekt med lærere (se egen ramme), og i denne artikkelen beskriver vi ved eksempler hvordan vi har arbeidet, og hva de deltakende lærerne har rapportert om denne arbeidsmåten. Det finnes helt sikkert andre måter å nærme seg programmering i matematikk på, men vi har ikke erfart og lest om «vår» måte andre steder i norsk kontekst.

Marianne Maugesten
Høgskolen i Østfold
marianne.maugesten@hiof.no

Henrik Stigberg
Høgskolen i Østfold
henrik.stigberg@hiof.no

Susanne Koch Stigberg
Høgskolen i Østfold
susanne.k.stigberg@hiof.no

Vårt utviklingsprosjekt

Skoleåret 2019–20 planla og gjennomførte vi (to tilsatte fra lærerutdanningen og én fra IT-avdelingen) fem heldagssamlinger med ni lærere fra tre ulike ungdomsskoler.

Vårt overordnede mål var å undersøke hvordan programmering kan støtte matematikken, og det innebærer også å undersøke hvordan man kan gi et tilbud til matematikklærere som har liten eller ingen tidigere erfaring med programmering. Vi ønsket å gi de deltakende lærerne et overblikk over hvilke verktøy som kunne anvendes, men vi ønsket også å jobbe med oppgaver som vi tenkte kunne brukes på elever på ungdomstrinnet. Fordi de deltakende lærerne hadde liten eller ingen erfaring med programmering, tok vi utgangspunkt i kompetansemålene på barnetrinnet og jobbet uten digitale hjelpmidler, for så å anvende en programmerbar robot (mBot). Deretter jobbet lærerne med Excel og Scratch, og avslutningsvis jobbet vi med Python ut fra kompetansemålene på ungdomstrinnet. På hver samling fikk lærerne noe input fra oss, de fikk selv prøve ut oppgaver, og til slutt fikk de tid til å planlegge for utprøving i egne klasser. Hver påfølgende samling startet med et fokugruppeintervju der lærerne diskuterte erfaringer med egne utprøvinger i klassene sine.

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.

www.caspar.no/tangenten

Matematikkplanen i LK20 har egne kompetansemål på hvert trinn som omhandler programmering, og algoritmisk tenking er beskrevet i kjerneelementet Utforsking og problemløsing:

8. trinn: Utforske korleis algoritmar kan skapast, testast og forbetrast ved hjelp av programmering
9. trinn: Simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering
10. trinn: Utforske matematiske eigenskapar og samanhengar ved å bruke programmering

Use – Modify – Create

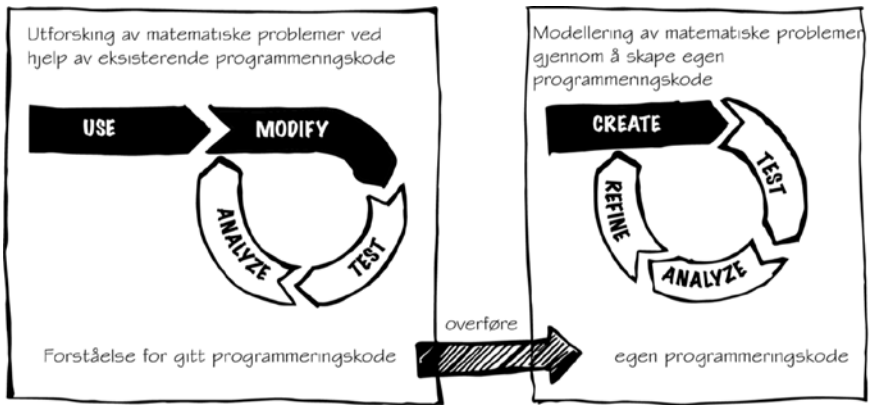
Vi har i prosjektet prøvd ut arbeidsmåten Use – Modify – Create (UMC) både i blokkprogrammering og i tekstprogrammering, og lærerne har gitt gode tilbakemeldinger på denne måten å arbeide på. Tilnærmingen er brukt i USA som en støtte til elever for å utvikle algoritmisk tenking (Lee et al., 2011; Martin et al., 2020). I kjerneelementet Utforsking og problemløsing beskrives algoritmisk tenking som viktig i prosessen med «å utvikle strategiar og framgangsmåtar for å løyse problem og inneber å bryte ned eit problem i delproblem som kan løysast systematisk» (Kunnskapsdepartementet, 2019).

Arbeidsmåten UMC består av tre faser:

- Elevene får et kodeeksempel av lærerne og undersøker hvilke funksjoner programmet har, og hvordan programmet kan være oppbygd (USE). Spørsmål de kan stille seg, er: Hvordan virker dette? Hva skjer hvis …
- Elevene går inn, leser programkoden og endrer deler av den (MODIFY) i ulik grad og utforsker matematikk samtidig som de lærer noe programmering. De kan starte med å endre lite, og etter hvert i prosessen kan mer endres. Her gis elevene gode muligheter til å prøve og feile.
- Elevene bruker kunnskapen sin og programmerer et liknende program (CREATE).

Det er ikke noen tydelig grense mellom de tre fasene. Elevene oppmuntres til å samarbeide for å finne de ønskede løsningene. Figur 1 viser arbeidet der det er tydelig hva man får utdelt, og hva man gjør til sitt (forfatternes egen figur).

Vi hadde tre argumenter for å prøve ut UMC i programmering i matematikkfaget. For det første kunne vi bruke ferdige lagde koder som tydelig viste sammenhengene mellom programmering og matematikk. Lærerne har ikke all verdens tid til å arbeide med programmering i matematikkfaget (det er mange andre kompetansemål i faget også), og sist, men ikke minst, «hjelper» metoden lærere som ikke har



Figur 1

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.

programmeringserfaring, i gang fordi det er en ferdig kode tilgjengelig.

Terningkast: ett tema, to programmeringsspråk

Vi vil vise to eksempler på UMC med programmene Scratch og Python ved en aktivitet knyttet til kompetansemålet fra 9. trinn: «eleven skal kunne simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering» (Kunnskapsdepartementet, 2019).

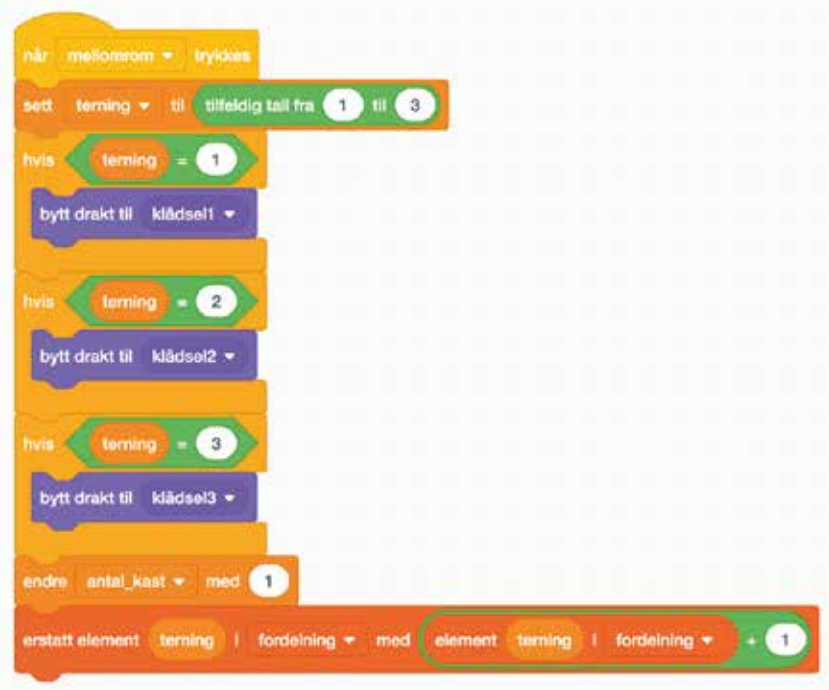
Lærerne hadde tidligere brukt regneark under arbeid med terningkast og sannsynlighetsregning. Det førte til gode diskusjoner om hvorfor det var nødvendig å bruke programmering til dette kompetansemålet, og om ikke regnearket også er en form for programmering.

Scratch

Elevene går til nettsiden https://scratch.mit.edu/studios/25998548/projects/ og velger «Start terningkast».

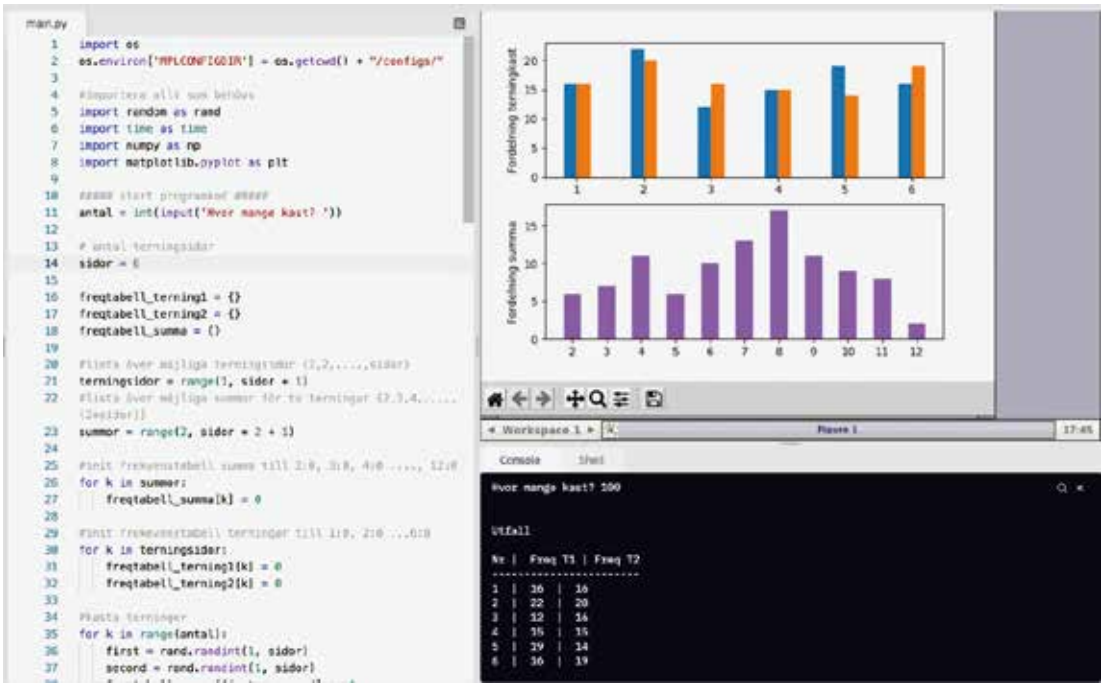
Figur 2: Eksempel på Use-fasen. Kast med terning med tre sideflater i Scratch.

USE: Elevene prøver flere ganger og observerer hva programmet gjør. De ser at her er det en terning med tre sideflater, og at programmet teller opp hvor mange enere, toere og treere de får når de velger et antall kast. Resultatene kan være et utgangspunkt for en samtale om Store talls lov og sannsynlighetsfordeling. Elevene vil muligens også diskutere om en fysisk terning kan ha tre sideflater, og bli nysgjerrige på om koden bak programmet kan endres slik at de får kast med en «tradisjonell» terning.



Figur 3: Eksempelkode i Scratch som kan forandres.

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.



Figur 4: Eksempelkode i Python etter forandring til kast med to terninger med seks sideflater.

MODIFY: Elevene prøver i denne fasen å forstå koden og endre den slik at terningen får seks sideflater. Her må de prøve og feile. Mange elever har også sett at det finnes andre terninger med for eksempel 10 sideflater, og vil lage disse. Vi oppfordrer alle til å være systematiske i utprøvingen sin, men det er alltid mulig å starte fra begynnelsen eller å angre en forandring. Det er viktig at elevene ikke føler at de ødelegger noe, men at de kjenner en utforskerglede og utforskertrang.

CREATE: Hva annet kan elevene tenke at de kan gjøre? De foreslår kanskje å utvikle eksemplet til kast med to eller tre terninger, veldig mange kast, summen av terningenes øyne osv. De kan også lage et helt nytt program.

Underveis i arbeidet vil elevene diskutere med medelever, hva skjer hvis…, og i etterkant kan klassen diskutere hvordan fordelingen endres når antall sideflater endres, og når antall kast endres, hvilke summer som er mest sannsynlige når elevene kaster to eller tre terninger, og hva som skjer når antall kast blir veldig stort, osv. Her kan elevene også trenes i skriving og argumentasjon i matematikk i egen arbeidsbok.

Lærerens rolle er viktig når programmering skal knyttes til matematikkens innhold. På samme måte som bruk av konkreter i matematikkundervisningen ikke automatisk gir forståelse, står ikke programmet for en iboende kunnskap som elevene tilegner seg. Læreren må hjelpe til med og støtte denne overgangen.

Python

Å lære seg Python fra bunnen av er tidkrevende for lærere samtidig som det vil kreve at mye av tiden i matematikkundervisningen går med til programmering. Dette er ikke et forsøk på å minimere betydningen av programmering i matematikkfaget, men våre erfaringer fra klasserommet viser at det er mange andre temaer som også krever tid.

Eksemplet med Python omhandler også terningkast. Under nettsiden repl.it har vi for-

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.

beredt en ressurs som inneholder ferdig lagde programmer:

https://repl.it/@SusanneStigberg/maprogterning#main.py

https://repl.it/@SusanneStigberg/terningkastutandiagram

Figur 4 viser hva elevene ser når de går inn på nettsiden ovenfor.

Når elevene skriver inn antall kast, vil de se kodene som trengs, i Python, diagram og tabell. Eksemplet er litt mer avansert enn det foregående eksemplet. Bruk av UMC kan da foregå slik:

USE: Elevene prøver ut programmet ved å velge forskjellige antall kast. Hva ser de i tabellen? Hvordan er fordelingen? Hvor mange terninger brukes? Hva ser de i diagrammet? Elevene ser at fordelingen av enere, toere, treere mfl. er ganske lik med mange kast. Men det gjelder ikke for summen av to terninger. Hvorfor forekommer noen summer hyppigere ved kast med to terninger? Vi håper dette kan invitere til matematiske diskusjoner der elevene forklarer og argumenterer.

MODIFY: I denne fasen lærer elevene seg syntaks. Det er sikkert ord de ikke forstår, men som de må søke opp.

Vi oppfordret deltakerne til å la elevene endre koden slik at de fikk kast med én terning eller tre terninger eller kast med terninger som har flere enn seks sideflater. Prøving og feiling står sentralt her sammen med systematisk utprøving.

CREATE: Kan elevene lage et annet program bygd på de samme ideene, for eksempel å skulle foreta valg og bruke stein, saks, papir?

Underveis og i etterkant vil læreren kunne igangsette gode matematiske diskusjoner, spesielt om resultatene av terningkast, for eksempel De store talls lov og hva vi oppnår ved å bruke Python som vi ikke oppnår ved regneark når det gjelder terningkast.

Våre erfaringer

Vi har prøvd ut metoden UMC på ni lærere fra lærerutdanningsskolene våre. Erfaringene kan ikke umiddelbart overføres til alle andre ungdomsskoler, men ungdomsskolelærere andre steder kan få noen ideer til hvordan programmering kan implementeres i matematikkfaget. Gjennom samlinger, fokusgruppeintervjuer og individuelle intervjuer med lærerne har de gitt oss beskrivelser av hvordan UMC fungerer i deres klasserom på ungdomstrinnet.

Lærerne mener elevene og de selv lettere igangsetter samtaler og diskusjoner om kodene og resultatene de observerte både i Scratch og Python. Flere av lærerne synes matematiske samtaler har vært vanskelige i andre temaer i matematikk. For eksempel rapporterte en lærer at elevene diskuterte hva i koden som måtte endres for å endre antall sideflater på terningen. De kom inn på Store talls lov, og læreren mente at programmeringsaktivitetene høynet kvaliteten på den matematiske samtalen. Igjen er lærerens rolle viktig ved at matematiske begreper kobles til kodene, og ved at de rette spørsmålene blir stilt.

Da vi startet opp kompetansehevingen, var lærerne skeptiske til hvordan de skulle få fram matematikken ved programmering i undervisningen. Flere av dem nevnte eksplisitt i intervjuene i etterkant at de kunne se hvilken nytte programmering kunne ha i enkelte matematikktemaer, for eksempel i sannsynlighet. De begrunnet dette med at vi tok utgangspunkt i kompetansemålene ved planlegging av programmeringsaktivitet. I prosessen med UMC må elevene prøve og feile, undre seg og stille spørsmålet «hva skjer hvis». Å gjøre feil og prøve å forstå hvorfor noe er feil, samt å endre feilene, hadde flere lærere tidligere slitt med å få elevene med på. De opplevde at det ble enklere i dette utviklingsprosjektet. Særlig i Modify-fasen ble prøve og feile en naturlig del av arbeidet, og lærerne håpet dette kunne ha overføringsverdi til andre matematiske temaer og oppgaver også. Flere lærere nevnte at denne arbeidsmåten også ga gode muligheter for tilpasset opplæring fordi kodene fungerte som en slags rike oppgaver: Alle kunne bruke dem, og de ga muligheter til utvi-

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.

delse av kodene ut fra elevenes forutsetninger, samt initierte gode diskusjoner.

Andre refleksjoner vi har fått fra lærerne etter utprøving i klasserommet, var at metoden UMC var spesielt godt egnet når det gjaldt tekstprogrammering. At elevene kunne lese og forandre koden, senket terskelen betydelig for å kunne anvende relativt avanserte program som var koblet til matematisk innhold, for eksempel sannsynlighetsregning. Noen nevnte at de ikke ville kunne arbeidet med tekstprogrammering hvis de ikke hadde fått programkoden på forhånd.

Det som ikke gikk som planlagt fra begynnelsen, var at lærerne bare brukte de to første stegene i modellen under utprøving med Python. Vi prioriterte da å jobbe med Use og Modify fordi lærerne hadde liten erfaring med programmering på forhånd. Vi argumenterer for Use og Modify som tilstrekkelig i starten av arbeidet fordi lærerne og elevene ikke skal bli programmerere, men demokratiske borgere som skal få utvikle forståelse for hvordan den digitale verden er oppbygd. Vi er også klar over at Create er den mest utfordrende fasen i arbeidsmåten.

Vi har presentert en måte å tilnærme seg programmering i matematikkfaget på, men det er fortsatt et stykke igjen. Vi håper våre eksempler på Use Modify, Create kan være et lite bidrag. Det finnes andre måter også, men vi synes UMC gir mange gode muligheter til arbeid med kompetansemål og kjerneelementer på ungdomstrinnet.

Referanser

Kunnskapsdepartementet (2019). *Læreplanverket for Kunnskapsløftet 2020. Grunnskolen*. Kunnskapsdepartementet.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32–37.

Martin, F., Lee, I., Lytle, N., Sentance, S. & Lao, N. (2020). Extending and evaluating the use-modify-create progression for engaging youth in computational thinking. I J. Zhang & M. Sherriff (red.), *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (s. 807–808). Association for Computing Machinery. https://doi.org/10.1145/3328778.3366971

Ressurser

Denne lenken til film forteller mer om metoden: http://www.it.hiof.no/~toremake/programmering-for-alle/vip/survey1/assets/videos/samtale_om_use_modify_create.mp4

Eksempler på undervisningsopplegg tilknyttet kompetansemålene på ungdomstrinnet kan du lese om på nettsiden vår: https://blogg.hiof.no/maprog/

Denne lenken er et eksempel på syntaks: https://www.w3schools.com/python/

Maugesten, M., Stigberg, H., Stigberg, S.K. (2021). Programmering på ungdomstrinnet. *Tangenten – tidsskrift for matematikkundervisning, 32*(3), 2–7.

www.caspar.no/tangenten