MASTEROPPGAVE

M5GLU

Mai 2022

Undervisningskunnskap i programmering

Knowledge for teaching programming

Type vitenskapelig

30 sp oppgave

Ingrid Voldmo

OsloMet – storbyuniversitetet

Fakultet for lærerutdanning og internasjonale studier
Institutt for grunnskole- og faglærerutdanning

Forord

Jeg skriver nå dette forordet med glede over at jeg har fullført lærerutdanningen, og med vemod over at disse fem fantastiske årene på OsloMet er over. Det har vært intense skippertak, fine kvelder på lesesalen, undervisning med utrolig gode lærere, praksis med gode venner og alt som hører studietiden til.

Jeg vil begynne med å takke mine studievenner for både hjelp med tekniske problemer som EndNote, moralsk støtte under skrivearbeidet, og generell hjelp med motivasjon de gangene frustrasjonen tok overhånd. Studietiden hadde ikke vært like fin uten dere. Jeg må spesielt takke Frida Sagen og Maja Ness Riediger for mye godt samarbeid i stressende eksamensperioder.

Jeg må sende en takk til min veileder, førsteamanuensis Andre Rognes. Alle innspill og konstruktive tilbakemeldinger på denne oppgaven var gull verdt. Det har også vært til stor hjelp med litt bekreftelse, når jeg til tider har mistet motet. Med god støtte fra veileder er jeg endelig i mål.

En spesiell takk til pappa som korrekturleste hele oppgaven min, og kom med gode tilbakemeldinger. Jeg vil også takke min samboer, som har holdt ut med et høyt nivå av stress den siste tiden, og samtidig kommet med motiverende og støttende ord. Sist men ikke minst en stor takk til mine informanter som stilte opp på intervju.

Arbeidet med denne masteroppgaven har vekket egen interesse for programmering, og jeg har innsett viktigheten av at elevene lærer seg dette fra tidlig alder. Interessen for programmering har vokst, og jeg har derfor valgt å søke på en bachelor i informasjonsteknologi til høsten. Nå går flere av mine medstudenter ut i jobb eller videre studier. Jeg gleder meg til veien videre, og til å se hvor vi alle ender opp.

Oslo, 2022
Ingrid Voldmo

Sammendrag

Programmering har blitt en del av læreplanen i grunnskolen, der matematikk har fått hovedansvaret. I denne oppgaven undersøkes det hvilke undervisningskunnskaper lærerne har tilegnet seg innen dette fagområdet. Problemstillingen er; **Hvilke undervisningskunnskaper har matematikklærere på ungdomskolen om programmering, og hvilke undervisningskunnskaper føler de seg mer utrygge på?**

Undervisningskunnskap i matematikk har tidligere blitt forsket på, og teori basert på forskningsresultater benyttes i denne oppgaven. Spesielt tas det utgangspunkt i en modell for undervisningskunnskap i matematikk, som bygger på det Shulmann kaller for innholdskunnskap. Kategoriene fra Balls modell for undervisningskunnskap i matematikk blir brukt for å analysere funnene i denne studien. Det er gjort kvalitative intervju av fire matematikklærere på ungdomstrinnet. Intervjudataene bygger på lærernes opplevelser av hvordan det har vært å undervise i programmering, og deres oppfatning av egen kompetanse. Lærerne viser å ha svært ulike forutsetninger for å undervise i programmering. De viser generelt sett god evne til å overføre erfaringer og undervisningskunnskaper fra «vanlig» matematikkundervisning. En av lærerne, som har gått et årsstudium i programmering, viser spesielt god kompetanse innen de fleste områder, der andre forteller at de kunne ønsket seg mer opplæring. Det kommer fram at lærerne kan ha behov for mer kunnskap innen undervisningsmetoder og kategorien «knowledge of students and teaching», men også grunnleggende fagkunnskaper i programmering, «common content knowledge», som igjen påvirker flere av de andre kategoriene. Det viktigste funnet er at noen lærere ligger på et ganske lavt faglig nivå i programmering. Det kommer fram at noen lærere lener seg på elevene som er gode i programmering, gjennom elevsamarbeid. Lærerne ville sannsynligvis dratt nytte av å ligge på et høyere faglig nivå enn elevene.

Nøkkelord:

Programmering, Undervisningskunnskap i matematikk, Innholdskunnskap, Kunnskapskvartetten, Algoritmisk tenkning, Utforskende oppgaver

Abstract

Programming has become part of the primary school curriculum, where mathematics has the main responsibility. This study investigates what kind of teaching knowledge the mathematics teachers have acquired in the programming field. The research question is therefore; **Which teaching knowledge does the mathematic teachers at the lower secondary level have, and which teaching knowledges are they more unsure of?**

Teaching knowledge in mathematics has previously been researched and is being used in this study. Ball´s "domains of Mathematical knowledge for teaching", which builds on what Shulmann defines as content knowledge, is being used for analyzing the findings in this study. Qualitative interviews have been conducted with four mathematics teachers at the lower secondary level. The data from the interviews is based on the teachers' experiences of what it has been like to teach programming, and their perception of their own competence. The teachers show that they have very different prerequisites for teaching programming. They generally show a good ability for transferring experiences and teaching knowledges from "ordinary" mathematics teaching to teaching programming. One of the teachers, who has completed a one-year program in programming, shows particularly good competence in most areas, where others say that they could have wished for more training. It appears that teachers may need more knowledge within teaching methods, and the category «knowledge of students and teaching», but also basic content knowledge in programming, «common content knowledge», which in turn affects several of the other domains of knowledge for teaching. The main finding in this study must be that some of the teachers has a relatively low level of programming competence. The study shows that some of the teachers lean on the pupils that are already good at programming, by student collaboration. The teachers would probably benefit from being at a higher academic level than the students.

Keywords:
Programming, Mathematical knowledge for teaching, Content knowledge, The knowledge quartet, Computational thinking, Inquiry based tasks

Innholdsfortegnelse

Figurliste:

Tabelliste:

Innledning

På videregående valgte jeg informasjonsteknologi som valgfag, noe som vekket min interesse for programmering. I løpet av min tid på lærerstudiet har dette kommet inn i læreplanen for grunnskolen. Programmering handler mye om det jeg som person setter pris på med matematikk. Det er logisk og noe konkret å forholde seg til. Du kan både være kreativ, samtidig som du slipper å være usikker på om du har rett svar, da du alltid kan teste og dobbeltsjekke. Fordi programmering nylig har blitt en del av grunnskolepensumet, har ikke programmering vært en tydelig del av grunnskolelærerutdanningen i løpet av de siste 5 årene.

I nåværende forskning om programmering i skolen kommer det fram hvordan programmering bidrar til algoritmisk tenkning, eller hvordan programmering er et verktøy for å lære matematikk (Kilhamn et al., 2021). Flere undersøkelser handler om lærere som har vært tidlig ute med programmering og har brukt tid på å sette seg inn i faget, eller jobbet med *lessonstudy* for å utvikle opplegg for programmering i skolen (Kilhamn et al., 2021). Programmering i skolen kan være med på å forsterke matematikkundervisningen, men dette er avhengig av at lærerne er åpne for forandringer i undervisningen og ser mulighetene. Dette vil være en stor utfordring for de lærerne som ikke har tatt til seg programmering på et tidlig stadium (Kilhamn et al., 2021).

At programmering blir en del av pensum i grunnskolen er spennende, men hva slags opplæring har de erfarne matematikklærerne der ute fått? Har lærerne fått kurs og veiledning i programmering? Er de undervisningskunnskapene lærene allerede sitter på, overførbare til å undervise i programmering? Har de gode nok fagkunnskaper til å lære elevene å kode? Dette er spørsmål jeg stiller i denne studien, og problemstillingen i denne oppgaven er derfor;

Hvilke undervisningskunnskaper har matematikklærere på ungdomskolen om programmering, og hvilke undervisningskunnskaper føler de seg mer utrygge på?

Læreplanen i utvikling

Programmering i matematikkfaget ble implementert med fagfornyelsen i 2020. I flere europeiske land har programmering blitt en del av læreplanen de siste 10 årene. I en undersøkelse fra 2015 kom det fram at 16 av 21 land har tatt programmering inn i sin læreplan (Balanskat & Engelhardt, 2014). I 2018 kom programmering inn i skolen i Sverige. Sverige, Norge og Finnland har hatt stort fokus på digital kompetanse de siste årene. Programmering har nå blitt en del av pensum i grunnskolen i Norge, der matematikkfaget har fått mest av ansvaret. Det er nå sett på som noe viktig en trenger å kunne i det 21. århundre (Erstad et al., 2021).

I Norge har programmering i utdanningen vært et fokus lenge, og er ikke noe nytt. I 2010 åpnet kunnskapsministeren Senter for IKT i utdanningen. Dette var et initiativ for å sikre barn og unges opplæring innen IKT og digitale ferdigheter (Utdanningsdirektoratet, 2010). Senter for IKT i utdanningen viste i 2016 til tre ulike måter å integrere 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*

I 2016 ble det gjort et pilotprosjekt av Senter for IKT i utdanningen i samarbeid med Utdanningsdirektoratet. Dette gikk ut på programmering som et valgfag i ungdomskolen, og skulle vare i 3 år. I forbindelse med prosjektet utviklet Senter for IKT i utdanningen et nettbasert kurs for lærere som skulle undervise i programmering, «MOOC» (Sevik, 2016).

Den nye læreplanen trådte i kraft i 2020. En av de store endringene er programmeringen som er blitt lagt til og gitt som hovedansvar til matematikkfaget. Etter endt 10. trinn skal elevene kunne «utforske matematiske egenskaper og sammenhenger ved å bruke programmering.» Dette er nytt for alle trinn og det er lagt opp til at elevene begynner med å tenke algoritmisk fra og med 2. trinn. Det som kalles problemløsning og algoritmisk tenkning er satt som et kjerneelement i matematikkfaget. Dette kan også knyttes opp til programmering, noe som bidrar til å gjøre det til et sentralt tema i faget (Kunnskapsdepartementet, 2020).
I snitt er det satt opp ett kompetansemål i matematikkfaget som handler om programmering på alle trinn bortsett fra 1. trinn. Her ser en hvordan det er tenkt at elevene skal utvikle seg til å kunne programmere etter endt grunnskole;

2. trinn	• «Lage og følge regler og trinnvise instruksjoner i lek og spill»
3. trinn	• «Lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet»
4. trinn	• «Utforske og beskrive strukturer og mønster i lek og spill» • «Lage algoritmer og uttrykke dem ved bruk av variabler, vilkår og løkker»
5. trinn	• «Lage og programmere algoritmer med bruk av variabler, vilkår og løkker»
6.trinn	• «Bruke variabler løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønster»
7. trinn	• «Bruke programmering til å utforske data i tabeller og datasett»
8. trinn	• «Utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering»
9. trinn	• «Simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe ved å bruke programmering.»
10. trinn	• «Utforske matematiske egenskaper og sammenhenger ved å bruke programmering»

Tabell 1 - Kompetansemål i matematikk fra ny læreplan (Kunnskapsdepartementet, 2020)

Teori

Undervisningskunnskap i matematikk

Undervisningskunnskap i matematikk er ikke noe som alltid har vært tydelig definert. Tidligere var det fokus på fagkunnskaper og generell pedagogisk kunnskap. Dette vil si for eksempel hvordan læreren planlegger timer og tidsbruk eller klasseledelse. Hvorfor en matematikklærer velger ut de oppgavene en gjør ut fra et tema, var det lite forskning på. Shulman kalte dette for *the missing paradigm*. Han mente det var for lite fokus på hvilke spesielle fagkunnskaper lærerne hadde, altså innholdet i timen. Hvilke spørsmål stiller læreren og hvilke forklaringer blir gitt. Shulman undersøkte hvordan nyutdannede lærere brukte sin kunnskap fra studiet til å møte elevene i praksis (Shulman, 1986). Han definerte innholdskunnskap ved å dele det inn i tre kategorier; *subject matter knowledge, pedagogical content* knowledge og *curricular knowledge*.

Innholdskunnskap

Subject matter knowledge er innholdskunnskaper læreren har i selvet faget, i dette tilfellet ren matematikkunnskap (Kleve & Hovik, 2016). En forventer at lærerens matematikkunnskap skal være god nok til å kunne forstå hvorfor vi regner som vi regner, og ikke bare vite hva man skal gjøre for å komme fram til et svar (Shulman, 1986).

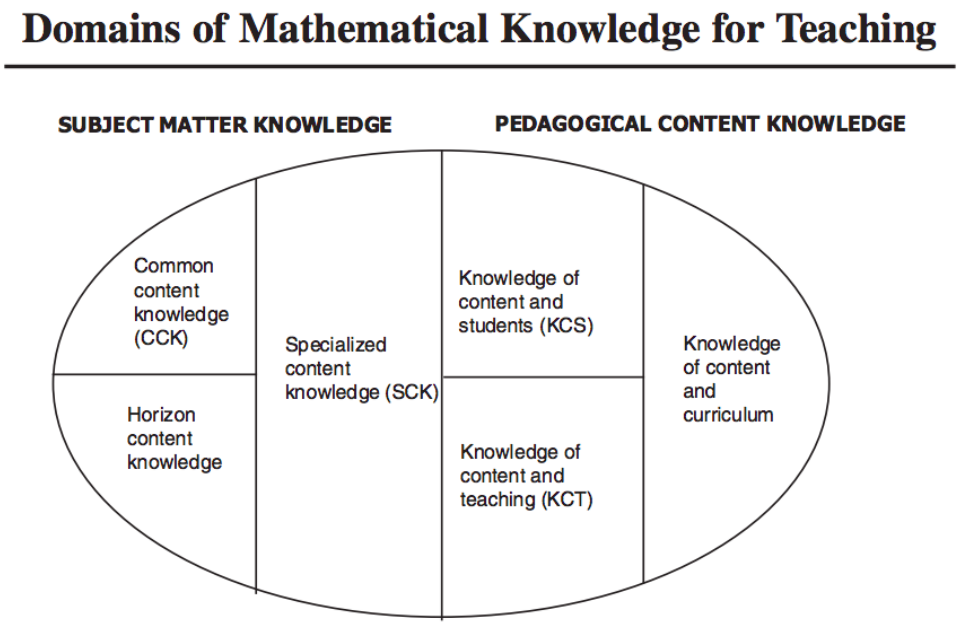
Pedagogical content knowledge beskrives som de kunnskapene i matematikkfaget som er nødvendige for å kunne undervise. Dette handler om å kunne bruke gode representasjonsformer, bruke relevante eksempler og generelt ha kunnskap om hvordan en gjør fagstoffet tilgjengelig for elevene. Da det ikke finnes en fasit på hva som fungerer for alle elever, må læreren kjenne til mange ulike måter å legge fram stoffet på. *Pedagogical content knowledge* handler også om å vite hvorfor spesifikke tema kan være vanskelig eller lett å forstå for elever. Denne kategorien til Shulmann er altså fortsatt en form for innholdskunnskap eller fagkunnskap, men henger sammen med kunnskap om læring og undervisning.

Den siste kategorien, *curricular knowledge*, kan oversettes til læreplankunnskap. Dette handler ikke bare om å kjenne til læreplanen og hva som står i den, men ulike måter å legge fram stoffet for elevene. Shulmann sammenligner denne kunnskapen med hvordan en lege har

kunnskap om ulike behandlingsmetoder, og kan vurdere disse basert på situasjon. Det forventes av en lærer å ha de samme kunnskapene om undervisningsmetoder. Læreplankunnskap kan igjen deles opp i lateral læreplankunnskap og horisontal læreplankunnskap. Med lateral læreplankunnskap menes å kjenne til læreplaninnholdet som brukes i andre fag på samme trinn. På den måten kan læreren koble innholdet til andre tema elevene holder på med. Vertikal læreplankunnskap vil si å ha kunnskap om hva elevene lærer seg tidligere i faget og hva de skal lære seg videre i skoleløpet. Dette gjelder ikke bare matematikk generelt, men hvordan spesifikke tema i matematikkfaget bygger på hverandre.

Rammeverk for undervisningskunnskap

Ball sitt rammeverk for undervisningskunnskap i matematikk kan kalles en utdypning av Shulmanns *subject matter knowledge* (SMK) og *pedagogical content knowledge* (PCK). Disse er satt som hovedinndelingen i modellen med tre underkategorier hver.



Figur 1: Modell for undervisningskunnskap i matematikk. Hentet fra (Ball et al., 2008)

Som vist i Figur 1, på siden med SMK, matematisk innholdskunnskap lærere må ha, har vi underkatekoriene *common content knowledge, specialized content knowledge* og *horizon content knowledge*. På den andre siden med PCK, pedagogisk innholdskunnkap lærere må ha, har vi underkategoriene *knowledge of content and students, knowledge of content and teaching* og *knowledge of content and curriculum*.

Den første kategorien under SMK, *common content knowledge*, er allmenn matematisk kunnskap. Dette vil si den generelle kunnskapen som kreves for å løse matematiske problemer på en korrekt måte. Dette er kunnskap som de fleste utdannede mennesker har, og brukes i ulike yrker. En må kunne kjenne igjen feilsvar, noe som blir spesielt viktig for lærere. I tillegg må lærere bruke korrekte begreper og løsningsmetoder for at undervisningen skal bli god og ikke miste dyrebar tid.
Specialized content knowledge er matematisk kunnskap som er unik for lærerprofesjonen. Det vil si for eksempel å identifisere mønstre i elevsvar, forstå misoppfatninger og undersøke tankegangen bak. Dette er matematisk kompetanse som folk i andre yrker enn lærerprofesjonen ikke har bruk for. Denne kunnskapen er altså matematisk og ikke pedagogisk, men spesiell for læreryrket.
Den siste underkategorien av SMK, *horizon content knowledge,* dreier seg om å se sammenhenger i matematikken som undervises. Dette vil si at en lærer for eksempel på 7. trinn må vite hva elevene skal lære videre på ungdomskolen for å kunne legge et grunnlag for videre læring.

Den første kategorien vi finner under PCK er *knowledge of content and students*. Denne kunnskapen er en kombinasjon av matematisk kunnskap og kunnskap om elever. Det vil si å vite noe om hva elever kan komme til å oppleve som vanskelig, lett, forvirrende eller motiverende. En lærer bør blant annet ha kunnskap om vanlige misoppfatninger. Det vil si at for å gjenkjenne en feil, handler det først om *common content knowledge*. For å forstå feilen og hvordan feilen har oppstått brukes *specialized content knowledge*, men for å forutse disse og kjenne til hva som er vanlig blant elever, behøves en annen kunnskap. Det er denne kunnskapen som her defineres som *knowledge of content and students*.

Den neste underkategorien til PCK er *knowledge of content and teaching*. Dette handler mer om hvordan læreren underviser i matematikk. Blant annet hvilke eksempler som blir brukt for å representere spesifikke matematiske tema og hvordan disse bidrar til matematisk forståelse. Læreren må kunne vurdere fordeler og ulemper ved et opplegg og velge oppgaver basert på målet med timen. Det innebærer valg av metoder og prosedyrer, men også å kunne vurdere elevinnspill. Altså vurdere om disse bidrar til undervisningen og om de er verdt å følges opp. Læreren må vurdere når man burde spinne videre på det elevene bidrar med, hvilke eksempler

som burde brukes i undervisningen, hvilke spørsmål som burde stilles og når man eventuelt må stoppe opp og utdype innholdet.

Den siste underkategorien er *knowledge of content and curriculum* er tatt ut ifra Shulmans tredje kategori, *curricular knowledge* (Shulman, 1986). Den er plassert som en egen kategori under PCK, men handler også om mye av det samme som *knowledge of content and teaching.* Kategorien må derfor sees i sammenheng med de andre (Ball et al., 2008).

Kunnskapskvartetten

Kunnskapskvartetten er basert på det Lee Shulman kaller for *subject matter knowledge* og *pedagogical content knowledge*. Ut ifra disse er det gjort observasjoner av matematikkundervisning for å undersøke hvordan disse kunnskapene kommer til syne. Det ble utviklet en modell som kategoriserer dette. Det er fire ulike kategorier som utgjør kunnskapskvartetten. Disse er *foundation, connection, transformation* og *contingency* (Rowland et al., 2003).

Foundation dreier seg om hva lærere har kjennskap til fra utdanningen sin og deres holdninger til faget. Det vil si den teorien og fagkunnskapen en har med seg som lærer, og alt hva undervisningen bygger på. Dette kan være både bevisst og ubevisst. Holdningsdelen vil si lærerens forståelse av hva matematikk er, og hva som er hensikten med å lære seg dette i skolen. Sammen blir dette grunnlaget for hvordan læreren velger å undervise elever på best mulig måte (Rowland et al., 2003).

Connection vil si lærerens evne til å koble ulike tema sammen i matematikkfaget. Læreren legger opp undervisningen i en viss rekkefølge basert på hva som er naturlig for matematikklæringen. Dette viser at læreren vet hvilke tema som bygger på hverandre i faget. Dette kan dreie seg om rekkefølgen av tema i undervisningstimer eller rekkefølge av hva man gjør i kun en undervisningstime (Rowland et al., 2003).

Transformation er kategorien som handler om representasjon. Dette handler om måten læreren velger å forklare matematiske ideer. Hvilke eksempler, analogier, illustrasjoner eller demonstrasjoner. Denne kategorien handler både om hva læreren velger når en planlegger, men også hva en velger å gjøre i undervisningssituasjonen (Rowland et al., 2003).

Den siste kategorien i kunnskapskvartetten er *contingency*. *Contingency* handler om lærerens evne til å takle uforutsette hendelser i klasserommet. Det vil si evnen til å svare og bruke elevers uforutsette innspill eller ideer i matematikkfaget. Altså hvor klar man er for å eventuelt skulle avvike fra en planlagt undervisningsøkt, og å vurdere om dette er hensiktsmessig ut i fra elevenes respons (Rowland et al., 2003).

Den algoritmiske tenkeren

Læreplanen bruker som nevnt algoritmisk tenkning som begrep i et av kjernelementene i matematikk. Det er derfor relevant å definere dette begrepet i sammenheng med programmering. Algoritmisk tenkning er en norsk oversettelse av det engelske begrepet *computational thinking*. Dette begrepet er mye brukt i sammenheng med programmering. Det kan defineres som en ferdighet som er nyttig for blant annet programmerere og dataingeniører. Likevel er algoritmisk tenkning også en ferdighet som du kan ha uten å programmere (Lye & Koh, 2014). Dette er noe elever kan lære seg fra et tidlig stadium allerede på 1. trinn gjennom analog programmering. Dette har blitt prøvd gjennom en *lesson study* der elevene jobbet med steg for steg oppgaver og dermed øvde seg opp til å senere jobbe med koding og feilsøking. En annen metode som ble brukt var arbeid med roboter, der elevene fikk enda tydeligere se om det var feil i koden deres, gjennom en praktisk tilnærming (Ahmed et al., 2020).

I 2017 ble det gjort en litteraturstudie for å komme fram til en slags fellesdefinisjon for algoritmisk tenkning (Shute et al., 2017). Tidligere studier hadde ulike tilnærminger til begrepet, noe som kan gjøre det vanskelig å forholde seg til som en del av grunnskolen. I litteraturstudien kom de fram til en definisjon som bygger på de ulike undersøkte definisjonene. Her defineres algoritmisk tenkning som «det konseptuelle grunnlaget som kreves for å løse problemer effektivt (dvs. algoritmisk, med eller uten hjelp fra datamaskiner) med løsninger som kan gjenbrukes i forskjellige sammenhenger» (Shute et al., 2017). De ulike konseptene ved algoritmisk tenkning som går igjen oftest i forskningen er abstraksjon, dekomponering, algoritmer og feilsøking.

Det ble utviklet en modell med formål å kunne vurdere grunnskoleelevers ferdigheter innen algoritmisk tenkning. Modellen kan brukes til flere formål innen utdanning og gir et slags grunnlag for hva elever faktisk skal lære når det kommer til å utvikle algoritmisk tenkning. Denne modellen definerer seks ulike sider av algoritmisk tenkning. Dekomposisjon defineres som det å dissekere et problem inn i mindre deler, som gjør det enklere å løse. Den neste er abstraksjon, som innebærer både datainnsamling og analyse, undersøke og forstå mønstre, og til slutt å kunne lage en modell som kan representere et mønster eller system. Den tredje siden ved algoritmisk tenkning er algoritmer. Dette vil si å konstruere logiske oppskrifter som kan løse et problem. Dette kan gjøres enten analogt eller ved bruk av dataprogrammer. Modellen presenterer fire underkategorier når en skal lage algoritmer. Først handler det om det å lage en steg-for-steg-oppskrift. Neste er å kunne utføre flere av disse stegene samtidig. For det tredje handler det om at oppskriften skal være mest mulig effektiv med færrest mulig steg. Den siste underkategorien til algoritmer innebærer å automatisere løsningen din slik at du kan bruke den til å løse lignende problemer. Feilsøking står oppført som en egen side ved algoritmisk tenkning. Dette vil si at når en har en løsning som ikke fungerer slik den skal, kan en gå tilbake og undersøke og finne ut spesifikt hvilket steg i løsningen som må rettes opp. Til slutt i modellen har du iterasjon og generalisering. Iterasjon defineres som det å konstruere og designe løsningen din flere ganger for å til slutt sitte igjen med det best mulige resultatet. Generalisering defineres som det å kunne overføre dine ferdigheter innen algoritmisk tenkning til flere ulike situasjoner, og på den måten bli god på å løse problemer på en effektiv måte (Shute et al., 2017).

Algoritmisk tenkning blir av Utdanningsdirektoratet definert som en metode brukt i skolen for å la elevene jobbe med problemløsning. Det vil si at en legger opp undervisningen slik at elevene får muligheten til å jobbe utforskende. Den algoritmiske tenkeren er systematisk og analyserende. Det er likevel viktig å være kreativ ved å komme opp med ulike løsninger og framgangsmåter for å løse problemer. Denne måten å jobbe på er slik informatikere jobber, og går derfor hånd i hånd med programmering. Arbeidsmetodene til den algoritmiske tenkeren kan deles opp i fem; utforske og eksperimentere, designe og lage, oppdage og rette feil, holde ut ved å fortsette og prøve igjen, og til slutt å samarbeide (Utdanningsdirektoratet, 2019).

Figur 2 - Modell for den algoritmiske tenkeren. Hentet fra UDIR sine nettsider (Utdanningsdirektoratet, 2019)

Den algoritmiske tenkeren analyserer problemer og finner ut hva som skal til for å løse dette, i tillegg må en kunne forstå hvordan en kan benytte dataprogrammer for å bidra til denne løsningen. Algoritmisk tenkning dreier seg om å løse problemer ved å bryte de ned til mindre problemer og dermed gjøre det enklere for en selv. En er nødt til å tenke logisk og kunne generalisere problemene ved å lage modeller eller algoritmer til å løse lignende problemer. Alt i alt kan de ulike konseptene en benytter seg av ved algoritmisk tenkning sammenfattes i seks nøkkelord. Disse er logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evaluering (Utdanningsdirektoratet, 2019). En ser at modellen utdanningsdirektoratet benytter, ligner den som ble utarbeidet i 2017 (Shute et al., 2017).

Programmering og undervisning

Som Linda Mannila skriver i boken «Att undervisa i programmering i skolan – Varför, vad och hur?» kreves det mye av lærerne når det innføres noe nytt i læreplanen. De skal ikke bare lære seg å programmere selv, men også sette seg inn i ulike pedagogiske strategier for å undervise i programmering. Fordi England har innført dette i skolen på et tidligere stadium,

har de også mer erfaring og forskning på emnet (Mannila, 2017). Mannila refererer til en studie fra 2015, som kategoriserer lærernes didaktiske strategier når det undervises i programmering. Disse er, etter oversetting til norsk, kontekstuell læring, samarbeid, algoritmisk tenkning, manipulere kode og analog programmering. Undersøkelsen er basert på hva lærerne selv har opplevd som vellykket, og gir en pekepinn på hvilke strategier nybegynnere kan benytte seg av i programmeringsundervisningen (Sentance & Csizmadia, 2015).

«Parsons problems»

Elevene skal etter ungdomskolen kunne bruke programmering. Veien dit kan være utfordrende. Undervisningen på veien dit bør være mest mulig effektiv og lærerik. Forskning viser at elever lærer mye mer effektivt ved bruk av det som kalles «Parsons problems», fremfor å skrive egen kode eller rette opp i feil i en kode. Dette er oppgaver som handler om å sortere bolker med ferdig kode i riktig rekkefølge. I en undersøkelse av denne oppgavetypen presterte elevene like godt som elevene som lærte det samme ved å rette feil i kode, og elevene som lærte ved å skrive kode (Ericson et al., 2017). Ved bruk av «Parsons problems» har det vist seg at læringen blir enda mer effektiv om en ikke bruker distraksjoner i oppgavene slik som er vanlig. Distraksjoner vil si at oppgaven inneholder bolkene med kode du kan velge fra, men ikke alle av disse er riktige. Distraksjonene inneholder feil kode, slik at elevene må analysere koden nøye og velge den riktige. Om elevene bare får presentert bolker som er riktig, og skal settes i riktig rekkefølge, vil altså læringen skje mest mulig effektivt (Harms et al., 2016).

Elevfeil som læring

Bray og Santagata (2014) utviklet et rammeverk for læring med fokus på feil. Det baserer seg på forskning som sier at denne måten å undervise på i matematikk har stort potensial. Rammeverket sier noe om hvordan en kan gjøre dette på en effektiv måte for læring, men også hvilke fallgruver en må prøve å unngå. For eksempel hvis du skal undervise med felles diskusjon rundt elevfeil for å fremme en felles forståelse. Da burde en blant annet bestemme en rekkefølge av hvilke feilsvar og hvilke riktige løsninger en skal diskutere. En burde også gjøre det tydelig hva som er det konseptuelle grunnlaget for feilen. Eksempel på fallgruver

ved slike diskusjoner er at det kun diskuteres feil i spesifikke prosedyrer for å løse oppgaven, eller at du lar koblingene til matematiske ideer og konsepter være underforstått i diskusjonen (Bray & Santagata, 2014).

PRIMM

PRIMM er en metode utviklet for å undervise i programmering. Denne metoden har også vist seg å være effektiv for elevenes læring. Metoden går ut på å planlegge undervisning ut ifra fem kategorier; *Predict, Run, Investigate, Modify* og *Make*. Først får elevene en ferdig skrevet kode. Elevene må derfor diskutere og forsøke å forutsi hva koden vil gjøre og hva som blir resultatet/output. De kjører deretter programmet, og tester på den måten hypotesen sin. Så må elevene undersøke. Dette kan gjøres ved for eksempel å søke etter feil, forklare, kritisere eller skrive av kode. Dette gjøres med en slags støtte i form av en mulig løsning. Deretter får elevene i oppgave å redigere koden på et vis, og på den måten vil de få et større eierskap til oppgaven. Til slutt får elevene muligheten til å bruke det de har lært til å lage et nytt program med et lignende problem å løse. PRIMM er utviklet ut ifra eksisterende forskning på programmeringsundervisning og støtter seg på sosiokulturell læringsteori. Metoden hevdes også å støtte utviklingen av læreres PCK, *pedagogical content knowledge*, der læreren skal ha en viktig rolle med høyere kunnskap enn elevene (Sentance et al., 2019).

«Unplugged programming»

Noen spesifikke ting, når det kommer til programmering, kan være ekstra vanskelig for elever å forstå. Spesielt begreper som variabler, funksjoner, vilkår og løkker er noe undersøkelser viser en må vie ekstra oppmerksomhet når det introduseres for elevene (Armoni et al., 2015; Flø, 2021; Grover et al., 2015). Dette er noe de ikke lærer av seg selv ved å bare programmere på egenhånd (Grover et al., 2015). Elevene må jobbe med dette for å få en forståelse av hva disse begrepene betyr og hvordan det henger sammen. Det handler ikke bare om å kunne programmeringsspråket, men hvordan språket fungerer. Det kan være at dette er utfordrende for elevene, fordi de ikke forstår det abstrakte (Armoni et al., 2015). Altså må elevene bli algoritmiske tenkere og få hjelp til å forstå begrepene. Dette kan øves på, for eksempel gjennom noe som kalles «un-plugged programming» eller analog programmering på norsk. Dette vil si aktiviteter som innebærer å lage oppskrifter eller forklaringer på hvordan en ting

skal gjennomføres eller se ut. For å knytte dette til programmeringsspråk, er det viktig å spesifisere for elevene hvilke begreper disse aktivitetene dreier seg om. For eksempel hvis elevene gjentar handlinger, så dreier dette seg om løkker (Flø, 2021). Analog programmering bidrar til elevenes ferdigheter i algoritmisk tenkning. Andre faktorer som også spiller inn er elevenes matematiske ferdigheter, og eventuelle tidligere erfaringer med programmering (Sun et al., 2021).

Utforskende oppgaver

Begrepet å utforske er brukt i 4 av 10 kompetansemål i matematikk på 10. trinn. Blant disse er kompetansemålet i programmering; «utforske matematiske egenskaper og sammenhenger ved å bruke programmering» (Kunnskapsdepartementet, 2020). Nettopp derfor er det relevant for oppgaven å definere hva det menes med å utforske. Dette begrepet er spesielt sentralt da det er lagt inn som et av kjerneelementene i matematikkfaget sammen med det som kalles for problemløsning. Utforskning blir i kjerneelementene definert som å «lete etter mønstre, finne sammenhenger og diskutere seg fram til en felles forståelse» (Kunnskapsdepartementet, 2020). Strategi og fremgangsmåte står nå mer sentralt enn det å finne riktig løsning fortest mulig.

Utforskende matematikkundervisning kan ha ulike definisjoner. I denne oppgaven vil jeg også trekke fram det Skovsmose (2006) kalte for undersøkelseslandskaper. Han skilte dette fra det han kalte for oppgaveparadigmet. I oppgaveparadigme er det kun ett riktig svar og kun en vei til mål. Utfordringen med oppgaveparadigmet er at det hindrer eleven i å være aktiv, og i å få eierskap til eget arbeid. I undersøkelseslandskapet er det ingen definerte oppgaver der elevene har en gitt løsningsmetode. Elevene får være mer aktive ved å gjøre valg i hva de ønsker å utforske innen et gitt tema, og de må finne fremgangsmåte på egenhånd (Skovsmose et al., 2006).

En type oppgave som er med å fremme problemløsning og resonnering kan være det Wæge og Nostrati (2018) kaller for LIST-oppgaver. LIST står for lav inngangsterskel og stor takhøyde. Disse oppgavene kan ha ulike framgangsmåter og er derfor utforskende. LIST-oppgaver kjennetegnes ved at de har mange muligheter for utforskning, både på et utfordrende nivå og på et enklere nivå. Dette er altså oppgaver som egner seg for alle elevene

i en klasse, uansett utgangspunkt. Oppgavene er enkle å sette i gang med slik at alle kan løse dem, samtidig som det åpner for et mer avansert nivå innen matematikk. Et eksempel på en slik oppgave kan være matematikkspill, der elevene skal spille spillet for deretter å reflektere over ulike strategier en kan bruke for å vinne spillet. LIST-oppgaver kan bidra til meningsfulle diskusjoner der alle elevene får bidra med det de kan, og vist seg fram. Det matematiske temaet for oppgaven trenger ikke være vanskelig, men det er tenkingen for å løse oppgaven som kan være ganske sofistikert (Wæge & Nosrati, 2018).

Metode

For å undersøke problemstillingen «Hvilke undervisningskunnskaper har matematikklærere på ungdomskolen om programmering, og hvilke undervisningskunnskaper føler de seg mer utrygge på?» har jeg valgt å gå for en kvalitativ metode, da jeg er mest interessert i nyanserte svar fra noen få informanter i motsetning til å få statistikk på lærere med utdanning innen programmering. Lærerutdanningen har til nå ikke hatt mye fokus på programmering da det ikke har vært en del av kunnskapsløftet før Fagfornyelsen 2020. Det er derfor interessant å skulle få noen utdypende svar rundt hvordan lærere i grunnskolen, som ikke har hatt dette som en del av utdanningen sin, har håndtert overgangen til ny lærerplan. Hvilke kunnskaper de har måtte tilegne seg, og eventuelt hvilke kunnskaper de allerede har som er relevante for undervisning i programmering.

Kvalitativt intervju

I denne undersøkelsen er det valgt å gå for kvalitativt intervju. Ved kvalitative intervjuer er spørsmålene laget på forhånd. Spørsmålene er åpne på den måten at det ikke er laget svaralternativer slik som i spørreskjema. Dette gir informanten mulighet til å tolke og svare på spørsmålet ulikt (Christoffersen & Johannessen, 2012). Jeg har valgt å gå for denne metoden da dette kan gi meg flere mulige svar. Det kan også åpne for at dataene blir mer fyldig og det kan dukke opp aspekter ved programmering i skolen som ikke ville kommet fram ved en mer snever spørreundersøkelse. Ved kvalitativt forskningsintervju har jeg benyttet meg av en fenomenologisk tilnærming. Dette vil si at funnene omhandler intervjuobjektenes forståelse og opplevelse av temaet (Kvale et al., 2015).

I utgangspunktet ville det vært nyttig å gjennomføre observasjon av intervjuobjektenes klasseromsundervisning for å kunne bekrefte eller avkrefte hvordan de selv beskriver sine typiske undervisningsmetoder. Her har jeg begrenset oppgaven ved å bare benytte meg av intervju, og funnene er derfor basert på hvordan lærerne selv oppfatter egen undervisning og egen undervisningskunnskap. Det vil si at svarene som kom fram, ikke nødvendigvis kan bekreftes, men kan gi innsikt i hvordan lærerne opplever å undervise i programmering uten presset det kan medføre å skulle bli observert. Det kan tenkes at det kunne oppleves som å

skulle bli vurdert, da denne oppgaven i likhet med intervjuguiden dreier seg om lærernes kunnskaper.

Videre har jeg valgt å bruke semi-strukturert intervju i dette prosjektet. Det vil si at jeg på forhånd forberedte noen spørsmål, men at rekkefølgen kan variere og oppfølgingsspørsmål vil forekomme (Christoffersen & Johannessen, 2012). Dette kan sees på som en mulighet for å få grundigere svar på visse spørsmål dersom intervjuobjektet har mye å bidra med på et område. I et semi-strukturert intervju har jeg noen tema som jeg på forhånd har valgt å introdusere, men er åpen for at intervjuobjektet kan tilføye andre tema som jeg eventuelt ikke har tenkt på. Kunnskapen skapes i intervjuet mellom meg og intervjuobjektet (Postholm et al., 2018).

Intervjuguide

For å undersøke hvilke kunnskaper lærere har, ble spørsmålene basert på kunnskapskvartetten og modellen for undervisningskunnskap i matematikk. På denne måten er spørsmålene lagt opp slik at det var mulig å gjøre en kategorisk analyse i ettertid. I intervjuguiden ble det satt opp 9 spørsmål, med noen planlagte oppfølgingsspørsmål i tilfellet svarene ikke ble utdypende nok.

1. *Hvilke tilbud har du fått til kurs eller videreutdanning innen programmering?*
   → *Hva var omfanget?*

2. *Hvilke forkunnskaper har du om programmering?*

3. *Hvordan har du tolket de nye læreplanmålene om programmering*
   → *Hva lærer elevene?*

4. *Hvordan ser en typisk mattetime med programmering ut hos deg?*
   → *Begrunnelser?*
   →*Planlegging?*
   →*Gjennomføring?*

5. *Hvilke ressurser benytter du deg av i undervisningen?*

6. *Hvilke tanker har du om hva elevene skal bruke denne kunnskapen til videre?*

7. *Hvordan jobber du med veiledning og vurdering i programmering?*

8. *Hvilke typiske utfordringer har elever med å bruke programmering?*
   →Hvordan møter du disse utfordringene?

9. *Kjenner du på noen utfordringer/fordeler ved å undervise programmering?*
   → Manglende kunnskaper?

De fleste spørsmålene er tenkt å legge opp til samtaler om ulike aspekter rundt programmering i skolen, fremfor å stille direkte spørsmål om deres egen oppfatning av egne kunnskaper. På denne måten er det mer åpne spørsmål der informantene kan fortelle om det de opplever som vanskelig eller lett ved å undervise i programmering i matematikkfaget. Det er gjort en grov kategorisering basert på kunnskapskvartetten, hvor noen av spørsmålene går under flere hovedtema. Hvordan svarene senere ble benyttet i analysen var likevel helt avhengig av hva informantene svarte på spørsmålene under intervjuene. Kategoriseringen ble gjort for å sikre samtale rundt alle de fire aspektene ved undervisningskunnskap fra kunnskapskvartetten.

Foundation	**Connection**	**Transformation**	**Contingency**
Spørsmål 1. Spørsmål 2. Spørsmål 3. Spørsmål 4. Spørsmål 5. Spørsmål 9.	Spørsmål 3. Spørsmål 4. Spørsmål 6.	Spørsmål 4. Spørsmål 5. Spørsmål 7. Spørsmål 9.	Spørsmål 4. Spørsmål 9.

Tabell 2 - Grov kategorisering av intervjuguide (egenprodusert)

Utvalg

Utvalget til intervjuene består av matematikklærere på ungdomskolen. I denne oppgaven er det benyttet eget nettverk og er derfor ikke et tilfeldig utvalg. Det er derimot forsøkt å velge ut lærere med ulikt utgangspunkt for å kunne programmering, med både godt erfarne og relativt unge lærere. Dette er gjort for å kunne få et litt bredere bilde av hvordan kunnskapen er blant lærerne. Det vil si at det er et forsøk på å få et utvalg med maksimal variasjon. Det er likevel valgt ut ifra eget nettverk, og kan derfor kalles et slags bekvemmelighetsutvalg (Christoffersen & Johannessen, 2012). Kriteriet for å ha blitt valgt ut er at de jobber som lærere i matematikkfaget på ungdomstrinnet. På den måten har de vært nødt til å sette seg inn i, og jobbe ut ifra den nye læreplanen, LK20.

Analyse og drøfting

I resultatkapittelet sammenlignes svarene fra informantene ut fra spørsmålene stilt i intervjuguiden. Her er det plukket ut det mest relevante utsagnene basert på teorien i oppgaven, og hva som kan være med på å besvare problemstillingen. I kapittelet «Drøfting» blir det gjort en teoretisk sammenligning. Det vil si at lærernes kunnskapsområder blir diskutert, og det blir benyttet teori som støtte til å sammenligne dataene i undersøkelsen. Det blir tatt utgangspunkt i hendelser og emosjoner informantene kan fortelle om, som kan gi en forklaring på hva slags undervisningskunnskap lærerne føler de sitter på (Postholm et al., 2018). Drøftingen er forsøkt kategorisert i ulike områder for undervisningskunnskap, men flyter også noe inn i hverandre, da kunnskapsområdene henger mye sammen og er avhengig av hverandre.

Validitet og reliabilitet

Transkripsjonen av intervjuene ble gjort så fort som mulig etter de ble gjennomført og tatt opp. Dette ble gjort for å sikre at jeg husket hva som ble sagt, om lydkvaliteten skulle være dårlig. Det ble også tatt notater under selve intervjuet i tilfellet det skulle skje noe med lydopptaket. Lydopptakene er hørt gjennom flere ganger for å sikre at transkripsjonen er korrekt. Dette for å sikre reliabiliteten av innsamlede data. Reliabilitet handler om intervjuenes troverdighet og etterprøvbarhet (Kvale et al., 2015). Det er også valgt å fjerne

noen fyllord som «da» og «eh» der det uansett ikke tydet på at informanten var usikker. Dette for å gjøre transkripsjonen mer lettlest. Det er også valgt å skrive om ord på dialekt for å sikre informantenes anonymitet. Utenom disse endringene, er transkripsjonene skrevet ordrett fra det informantene sa i intervjuene.

Lærernes opplevelse av hvor trygge de er i egne fagkunnskaper endrer seg etter hvert som de får mer erfaring, og vil derfor ikke være etterprøvbart. Undersøkelsen sier likevel noe om hvor lærerne er i prosessen med å tilegne seg disse fagkunnskapene og undervisningskunnskapene akkurat nå, og besvarer derfor problemstillingen.

Det er få intervjuobjekter, som gjør at funnene ikke nødvendigvis kan generaliseres for alle matematikklærere i Norge. Det kan fortsatt gi viktig innsikt i hva slags grad av kompetanse som finnes der ute. Det er også viktig å ikke stille ledende spørsmål i intervjuene, noe som er forsøkt ved å stille mer åpne spørsmål. Om man definerer validitet som om dataene reflekterer det du ønsker å undersøke, vil en kunne si at intervjuene i denne undersøkelsen er valide (Kvale et al., 2015).

Resultater

I dette kapittelet legges det fram resultater fra datainnsamlingen. For å besvare problemstillingen er det gjort en analyse av intervjuene, og det er hentet ut de mest relevante utsagnene for oppgaven. Disse utsagnene er lagt fram som en sammenligning ut ifra hva informantene svarer på de ulike spørsmålene, og hvilke likheter og ulikheter som kommer fram. Senere i oppgaven drøftes dette opp mot teori og hvilke områder for undervisningskunnskap dette viser til.

Fagkunnskap i programmering

På spørsmål om hva slags kurs informantene hadde fått kom det fram ulike svar. De fire lærerne hadde ulike kunnskaper i programmering som følge av ulik utdanning eller kurstilbud.

Lærer A var læreren med mest utdanning innen koding, men har ikke fått kurs gjennom arbeidsplassen i læreryrket.

I: «Så du har ikke fått noe kurs innen programmering da?»

A: «Nei, jeg søkte om å bli med på et kurs. Da var det to andre lærere som endte opp med å dra på det kurset.»

I: «Ikke sant, så hvilke forkunnskaper har du i programmering da?»

A: «Jeg har hatt litt IT fra universitetet i Oslo, da jeg har studert matte og fysikk. Så littegrann Java og littegrann Python, men ikke veldig mye. La oss si eh 20 studiepoeng.»

Lærer B hadde derimot fått muligheten gjennom jobb til å gå et årskurs i programmering for lærere.

B: «Jeg har gått programmering i Volda. Nettbasert. 15 studiepoeng så det tok jeg forrige skoleår da. Så det søkte jeg, eller jeg gikk til ledelsen og spurte om jeg kunne ta det, også søkte jeg og fikk godkjennelse på det.»

Lærer B forteller også at det er ingen andre lærere på skolen som har gått dette kurset per dags dato, men at dette er planlagt fremover. Utenom dette kurset har det kun vært et kveldskurs i Scratch for alle lærerne på skolen.

Lærer C kunne fortelle om et kurs tilbudt av arbeidsplassen så tidlig som i 2018.

C: «Ja, eh, jeg fikk tilbud om å være med på programmeringskurs allerede i 2018 faktisk. Da var det det Profag U som jeg nå går på som begynte opp, startet. Men de trodde nok vi lærere kunne mer enn vi kunne. Så for å ikke kunne noen ting så hadde de et høyt tempo, så vi ga opp. Rett og slett. Nei, det var null pedagogisk bak det, og det var ren programmering, vi satt bare og skrev av en powerpoint. Og det var ingen tid til å se på det, eller gjøre noe med det. Så det var.. vi ga opp. Vi sluttet, vi gikk i protest.»

I: «Så det var rent faglig da?»

C: «Det var rent faglig ja! Pluss vi var gjennom liksom grunnkurset i informatikk, så det var, ja vi ga opp. Men så har jeg vært veldig skeptisk til programmering, etterpå, fordi vi fikk litt sånn dårlig start. Men så i år så måtte jeg jo melde meg på det ProFag:U på universitetet igjen da. Og daa hadde de endret, eller vi klaget jo en del da sist, og fortalte dem hva vi mente om opplegget. Så nå har de gått noen ganger da. Jeg fikk plass nå i høst på det ProFag på universitetet, på ungdomstrinnet da. Og nå hadde de endret. Så nå er det mulig å henge med, selv for en dame i 40-åra som har jobbet i 20 år, som ikke kan noe programmering. Så nå er det faktisk ganske gøy å drive med programmering. Og de har lagt ganske mye vekt på det her med pedagogisk tilnærming da, og hvordan vi faktisk kan undervise det i klassen, og tilpasse hvordan vi kan undervise det ogsånn, i tillegg til den teoretiske og praktiske biten ved å programmere.»

Lærer D var læreren med minst forkunnskaper eller utdanning innen programmering.

D: «Eh på skolen har det vært en, eh, kall det kveld på 1,5 time hvor vi har fått innblikk i hvordan Scratch funker. Som er et kodeverktøy, og det var for alle. Det var ikke kun for mattelærerne, men det var for hele skolen. Og det var veldig vanskelig for de som ikke hadde sett det før, og det var svært lite utbytte av det kurset. For min del. Så det savner jeg når det da gjelder koding. Å få mer tilbud, mer besøk, og satt av mer tid til det. For det blir bare mer og mer koding i flere fag.»

I: «Ikke sant. Har du noen andre forkunnskaper innen programmering?»

D: «Nei, jeg veit hva poenget er. Jeg skjønner at jeg skal få noe til å skje ved å sette inn koder. Men det er alt, ved å prate med andre, men ikke hvordan det funker på PC eller hvilke verktøy man må ha for å kunne kode, jeg har null forkunnskaper.»

Læreplanmål i programmering

På spørsmål om hvordan informantene har tolket læreplanen i forhold til programmering, kommer det fram at alle fire lærerne benytter seg av programmet Scratch og programmeringsspråket Python, selv om dette ikke er noe som står spesifikt i læreplanen. Lærer B henviser til ulike læreverktøy som stort sett bruker Python som programmeringsspråk, og blokkprogrammering til å begynne med. Det blir fortalt at blokkprogrammering også er lagt inn på 8. trinn i kunst og håndverk.

Det kommer også fram viktigheten av å knytte det til mange tema. Både lærer C og lærer D nevner nettopp dette på spørsmål om hvordan de har tolket læreplanmålene i programmering.

C: «For jeg har liksom en teori om at programmering bør være en del av de ulike temaene.»

Lærer C påpeker også at dette kan være vanskelig tidsmessig, og at noen andre lærere heller legger det opp som et eget tema og heller jobber med programmering over en viss periode.

D: «Det vi prøver på er å få knytta inn litt koding i alt hva vi driver med. Om det er

geometri, om det er algebra, om det er sannsynlighet, så prøver vi å lage ett opplegg som handler om koding.»

En typisk matematikktime med programmering

Når lærerne beskriver en typisk programmeringstime kommer det noe ulike svar.

A: «Altså jeg pleier alltid å si at du lærer ikke noe programmering av å se på meg programmere. Du lærer programmering av å programmere. Så en typisk programmeringstime vil gå ut på at jeg for eksempel viser de en løkke som skriver ut tallene 1 til 10 i Scratch eller Python, hvordan det funker. Også blir det det elevene må skrive litt sammen med meg. Også ber jeg kanskje elevene etterpå å gjøre små forandringer på programmet sitt, sånn at programmet skriver ut litt andre tallrekkefølger da. Også er det litt fram og tilbake da, også går vi gjennom en ny type oppgave senere også ja, en del fram og tilbake. Også en del selvstendig jobbing på slutten av timen kanskje.»

Lærer B forteller først om et opplegg hvor elevene simulerer barnefødsler, koblet opp mot temaet sannsynlighet. Læreren legger vekt på at oppleggene skal føre til forståelse.

B: «Ja! Eller hvert fall prøver å koble det opp mot tema vi jobber med sånn som nå jobba vi med sannsynlighet og da tenkte jeg at det var best i Python, tekstprogrammering, også har jeg «Kåres kokebok i programmering» som jeg har titta litt i og der fant jeg noe som gikk på akkurat det der med å simulere barnefødsler. Så da tok jeg den som utgangspunkt også laget jeg en oppgave som passet bedre til 9. klasse for det kan være litt vanskelig det som står i den boka her for den er jo også for videregående. Men mye går på forståelse, hva er det koden gjør og hvorfor printer den ut det den printer ut. Så det handler mye om at elevene skal skrive av en kode eller på forhånd gjette hva koden gjør slik at de forstår hva det handler om. Også går det mer på da etterpå at de skal bygge ut koden eller skal manipulere koden for å få den til å vise noe annet da. Så det går mye på forståelse.»

Når lærer B planlegger undervisning tas det først og fremst utgangspunkt i læreplanmålene.

B: «Jeg ser på læreplanen, også ser jeg på kompetansemålene og der står det jo ganske tydelig når man skal bruke programmering. Men også i oppgaver hvor det ikke nødvendigvis står programmering i, det står utforskende. Også tenker jeg for eksempel; okei, hvordan kan man utforske forskjellige geometriske figurer – Jo, det kan man gjøre i Scratch. Ved å bruke penn-funksjonen og på en måte prøve å tegne og gjenskape ved å se på vinkler, løkker. Så jeg ser ikke nødvendigvis bare på kompetansemål det står programmering på, men også bruker programmering mer som et verktøy da for å bygge forståelse og å utforske.

C: «Vi har jo jobbet med skriveprogrammering, og vi bruker Python som programmeringsspråk. Sååå en typisk time er nok litt sånn at jeg starter med å gå gjennom noe, noen grunnleggende ting i fellesskap. Også har jeg sluppet dem litt løs. Jeg har gitt dem.. Eller boka som vi bruker har jo ganske fine sånne forklaringer, sånne små programmeringsbolker. Så jeg har egentlig sluppet dem litt løs og latt dem prøve på egenhånd.»

D: «Det er veldig forskjellig fra lærer til lærer, men jeg synes det er veldig greit at de får utforska litt selv. At de får en bruksanvisning, følg det. Uten at jeg viser at sånn, sånn, sånn skal det gjøres, men at de er nødt til å følge en bruksanvisning for å, hva skal jeg si, bli kjent med det selv da. At de må ta egne valg selv i stedet for at de skal følge meg som viser de på smartboard. For da får de lært litt mer av det grunnleggende, av seg selv, uten at jeg viser at du må gjøre sånn, du må gjøre sånn, du må gjøre sånn. Selv om det er det som står på et ark.»

Alle fire lærerne forteller at deres programmeringsundervisning ligner mye på vanlig matematikkundervisning. Lærer A forteller at det er noe kortere forklaringer med tavleundervisning, da det er enklere å få programmering til å fungere. Dette kan lærer D også kjenne seg igjen i, som forteller at det blir mindre tavleundervisning i programmering og at elevene får prøve seg mer på egenhånd. Lærer B forteller at hen benytter seg av den algoritimiske tenkeren mye, i både programmeringsundervisning og vanlig matematikkundervisning.

B: «Men sånn klassisk vanlig matematikk-time, så spørs det litt hva tema er da, men læreplanen har jo mye sånn utforskende så jeg prøver å i stedet for å si SÅNN er vi regner ut pytagoras så får jeg heller elevene til å utforske fenomenet. Det blir jo heller mye sånne type oppgaver da, utforskende oppgaver. Ikke så mye mengdetreningsoppgaver, men mer oppgaver som gjør at elevene må samarbeide med hverandre, diskutere med hverandre, prøve seg fram, ser de noe sammenheng, er det noe mønster her. Jeg tenker jo veldig mye på den algoritmiske tenkeren hele tiden i matematikk.»

Ressurser til undervisning

Lærer A	**Lærer B**	**Lærer C**	**Lærer D**
Scratch Python	Scratch Python «Kåres kokebok» Kidsa koder (nettside) MicroBit (nettside)	Scratch Python Campus «Matemagisk» ProFag:U (nettside)	Scratch Python Teams (delingsplattform)

Tabell 3 - Lærernes ressurser i undervisning (egenprodusert)

Elevers videre nytte av programmering

På spørsmål om hva elevenes nytte av programmering, kommer alle lærernes syn på programmering opp. Lærer A, C og D nevner at logisk tenkning er en del av programmeringsundervisningen. Lærer D beskriver koding som «planlegging, gjennomføring og evaluering». De tre lærerne mener at dette er noe alle kommer til å få bruk for i flere yrker, selv de som ikke skal kode på PC i sine fremtidige yrker. Lærer B viser også mye til den algoritmiske tenkeren ellers i intervjuet, men svarer på dette spørsmålet at i det samfunnet vi lever i dag, så blir teknologisk forståelse viktig for de fleste. Lærer A og lærer C legger også vekt på at de ser på programmering som et verktøy og en annen måte å tenke på.

A: «Nummer 2 er jo at programmering er jo en form for logikk da. På den måten er litt annerledes enn matte, men det er veldig greit at de lærer seg å tenke på denne måten da. Det tror jeg vil hjelpe de aller aller fleste, å forstå littegrann programmering. Det er så vanskelig å måle når du trenger å tenke på den måten da fordi det dukker opp når du for eksempel skal finne retningen til et sted, eller når du skal bruke eliminasjonsmetoden og så videre da. Det er sånn en tanke bak. Jeg tror det er lurt at folk lærer seg å programmere, men det er vanskelig for meg å pinpointe hva du skal bruke programmering spesifikt til. Altså utover videre programmering.»

C: «Det er jo en annen måte å tenke på da. Og jeg tenker at alle strategier er fine å ha med seg, uansett. Jeg tror ingen taper på å ha litt programmering. For jeg ser jo på det som et hjelpemiddel og ikke som et tema liksom. Og da tenker jeg at da gjør det ingenting at de har litt av det med seg alle sammen.»

Vurdering og veiledning i programmering

Lærer B har sett nye muligheter ved vurdering i programmering, og er den læreren som har kommet best i gang med dette.

B: «Ja, jeg jobber stort sett konsekvent i par at de jobber to og to. Det er også fordi at i den algoritmiske tenkeren så handler det mye om samarbeid, og som programmerere ute i arbeidslivet så jobber man også i team. Så det er sjeldent jeg lar elever jobbe alene, så jeg setter de sammen ofte etter nivå, også får de en oppgave også leverer de ett dokument med utklipp av kodene deres, særlig hvis de jobber med Python, og forklaringer og begrunnelser.»

«Og jeg har også i etterkant klippet ut forklaringene og begrunnelsene og vist det anonymt på PowerPoint også har jeg fått elevene til å rangere svarene. Hva er det på en måte det mest utfyllende svaret og det minst utfyllende svaret? Slik at de kan se hva som på en måte forventes av de da, det har vært ganske interessant. Også har vi snakket om for eksempel; ja, det svaret her var litt kort, hva kunne vi gjort for å få det

svaret her til å bli bedre? Men også hvis noen elever har gjort det bra så viser jeg fram dette også. Så jeg er egentlig opptatt av å vise fram sånn i alle fag da.»

De tre andre lærerne forteller at de ikke har hatt mange prøver eller sluttvurderinger i programmering, men jobber med underveisvurdering på samme måte som i andre temaer i matematikk. Lærer C kan også fortelle at de har lagt inn programmeringsoppgaver i noen prøver til nå, mens lærer D har fokus på muntlige tilbakemeldinger i timen framfor skriftlige prøver. Dette innebærer blant annet å gi de flinkeste elevene ekstra utfordringer.

I: «Du føler deg trygg nok i programmering til å gi utfordringer?»

D: «Ja, eller det handler ikke bare om programmering da men. Uansett om det er programmering da, så skjønner jeg. Jeg veit at du kan gjøre ekstremt mye i Scratch og Python, men jeg vet ikke selv hvordan man gjør det. Men jeg kan si at hva hvis det hadde vært sånn, får du til å gjøre det sånn at det ender opp sånn. Og da kan eleven bruke sin kunnskap til å få til, og vise meg da.»

Elevutfordringer i programmering

På spørsmål om typiske elevutfordringer nevner lærer A og lærer C problemet med å skrive grammatisk korrekt i Python.

A: «At de i Python må de skrive ganske grammatisk korrekt da, må skrive inn ting til punkt og prikke. Elevene skjønner ikke alltid at hvis du har kalt en variabel for eksempel teller med liten t. Og hvis du kaller en variabel med stor T også får du opp feilmelding.»

C: «Ja, eh, det her med at alt må være så nøyaktig, kanskje er den største utfordringen. At det må være små bokstaver eller at variabelen må hete det samme i en programmering. At du må huske å ha parenteser der du skal ha det og kolon der du skal ha det og sånt.»

Lærer A påpeker også en utfordring med det abstrakte med løkker.

A: «Og å forstå greia med løkker for eksempel. Eller at de teller variabler som blir oppdatert inni løkker da. De kan lett forstå at 1 og 1 er 2, men når de får en kode der det står $i = 1$ og $i = i + 1$, så skjønner de på en måte ikke at det legges på 1 på variabelen *i* hele veien da. Så oppdatering av variabler kan være problematisk, og syntaks.»

Lærer B nevner at utholdenhet kan være en utfordring, og lærer D påpeker at elevene kan ha utfordringer med hvordan de bruker selve programmet, eksempelvis Scratch.

For å møte utfordringen med å skrive grammatisk korrekt i Python forteller lærer A at de prøver å jobbe med varierte oppgaver.

A: «Jeg går gjennom noen eksempler ganske sakte i timene. Jeg ber de alltid gjøre de samme tingene som meg, også sier jeg at når jeg kjører mitt program så skjer dette, og sørg for at det skjer i deres program. Så hvis dere ikke får til det, så spør meg om hjelp så skal jeg komme og hjelpe til. Og det er ikke så veldig programmeringsteknisk så vanskelig det vi holder på med. Det er jo egentlig bare å putte riktig ting på riktig plass på en måte. Det er skritt 1, også er det skritt 2, det er å forandre på noen av ingrediensene sånn at da kommer forståelsen litt etter hvert da. Ved at når jeg forandrer dette, så forandrer denne tingen seg. Så når jeg underviser i programmering så er det veldig mye liksom at jeg varierer på de forskjellige ingrediensene, så skal de gjøre det samtidig.»

Lærer C møter den samme utfordringen ved å se nøyere på feilmeldinger med elevene.

C: «Nei, jeg tenker at det handler mye om at de får prøve da. Også er jo disse, jeg vet ikke hva det heter jeg, disse plattformene eller hva du kan kalle det, de gir ganske konkrete tilbakemeldinger da. «Feil i linje 5» liksom. Også lære dem å tolke tilbakemeldinger eller feilmeldinger da.»

Lærer C forteller også at denne utfordringen har mye med tålmodighet å gjøre.

C: «Jeg tror programmering er mye prøv og feil. Det er noe med den der utholdenheten vi leter litt etter i matte. Det der å stå i utfordringer som de kanskje ikke løser med en gang, og få lov å utforske. Det er litt sånn læreplanen er veldig opptatt av utforsking og utholdenhet og den biten. Jeg tenker at der har jo programmering en kjempefordel.»

Lærerne om egen undervisningskunnskap i programmering

På spørsmål om utfordringer ved å undervise i programmering føler lærer A seg ganske komfortabel, men mener også at alt som er nytt er noe vanskeligere.

A: «Nei, assa for min egen del, jeg er jo relativt nerdete. Jeg liker jo, eller når jeg fant ut av at jeg skulle begynne å undervise i programmering, så begynte jeg jo å programmere masse greier på egenhånd da. Sånn småting av ymse slag. Jeg føler ikke at jeg trenger noe ekstra av kursing eller whatever i programmering. Samtidig så tror jeg at de er ganske, jeg tror ikke at det er alle mattelærere som er like glad i programmering som det jeg er da. Jeg tipper at de fleste andre mattelærere ville ha ønsket seg et kurs da, fordi det er helt nytt og fordi plutselig så er det forventet at du skal kunne disse tingene så godt at du kan undervise i de da.»

Lærer B bruker også fritiden på å sette seg inn i programmering, og føler seg derfor trygg på det faglige i programmeringsundervisningen.

B: «Altså jeg sitter jo mye på fritiden. Jeg fikk jo litt ut av studiet, men mye går på egen interesse. Jeg synes jo programmering er veldig spennende så jeg sitter jo mye på egenhånd og lager oppgaver og tester ut, men foreløpig så kan jeg jo mye mer enn elevene så det blir jo mer på sikt da, om jeg da vil føle mer på at elevene kan mye mer enn meg når de blir mer vant til det. Men akkurat nå så føler jeg at jeg har mye mer kompetanse enn resten, jeg føler meg forholdsvis ganske sikker på det. Både med kollegaer og elever.»

I: «Så du kjenner egentlig ikke på noen særlige utfordringer?»

B: «Nei, eller jeg kjenner jo på noen utfordringer når jeg skal prøve å lage oppgaver da, eller prøve meg på noen oppgaver så møter jeg jo på noe eller får noen feilmeldinger, men da må jeg på en måte prøve å finne ut hvor den feilmeldinga er og prøve å endre på koden. Litt sånn som elevene, så jeg møter jo på de samme problemene som elevene men da har jeg lært litt av de problemene på forhånd ved å gjøre oppgavene. Så jeg er ikke der hvor jeg kan gå inn i klasserommet og be de gjøre en oppgave uten å ha gjort den på forhånd, der er jeg ikke, men det er jeg i matte ellers. Der kan jeg på en måte hoppe rett inn. Så jeg må alltid gjøre et forarbeid, det må jeg.»

Lærer C føler seg utrygg i forhold til egne programmeringsferdigheter, men påpeker at erfaring i yrket gjør at det går bra likevel.

C: «Jeg kjenner på det skikkelig fordi jeg kan ikke programmering. Jeg føler meg langt ifra trygg på det. Så jeg synes det er vanskelig. Jeg synes det er skummelt. Haha.»

I: «Ja.»

C: «Rett og slett. Samtidig så tenker jeg at i og med at elevene liksom ikke har hatt det så mye heller, så får vi liksom prøve oss fram litt sammen da. Også håper jeg at jeg har noen elever som kan mye mer enn meg også bare prøve å finne ut av det sammen liksom. Også har jeg såpass mange års erfaring at jeg kan liksom slippe kontrollen litt og la det surre og gå litt. Og la elevene få utforske. Så jeg føler meg ikke liksom kvalifisert til å stå foran en gruppe og forklare så fryktelig mye. Fordi jeg kan det samme som elevene liksom.»

I: «Og det er jo kanskje forskjellen fra «vanlig» matte, at der kan man hakket over elevene.»

C: «I forhold til vanlig matte ja! Der kan jeg jo svare på alle spørsmål. Jeg kan jo hjelpe de med alt. Nå er det litt mer sånn, jeg hadde en elev som spurte om ett eller annet også sa jeg.. Jo, de holdt på med variabler da også skulle de over i *if, else.* Også hadde jeg ikke hatt det på kurset enda, så jeg visste ikke helt hva det var, så sa jeg bare «men

okei, da får du google det da». Og han googlet, og programmerte *if, else.* Dette her er jo flinke elever da, men likevel tenker jeg at det er greit det. Det gjør ingen ting. Jeg er veldig komfortabel med det.»

Lærer D føler seg også utrygg på egne programmeringsferdigheter, og ønsker seg kurs for lærerne på skolen.

D: «Ja, eller jeg skulle jo ønske jeg var ekspert på det og kunne alt selv, men det er jeg jo ikke. Og jeg synes det er vanskelig å bruke tid på å sette seg inn i det selv. Jeg er jo ikke fan av programmering selv. Det finnes andre ting i matematikk som jeg elsker vesentlig mye mer enn programmering, og da er det vanskelig å sette seg ned en tirsdagskveld eller fredagskveld og utforske og sånt selv. Så det skulle jeg ønske at jeg var bedre på. Samtidig så skulle jeg ønske at vi hadde mer dager som gikk til koding, altså kurs. For at vi skal bli bedre, for at vi skal kunne klare de målene som står på udir. Og det koster jo ingenting. Det må bare planlegges, at den dagen her skal vi øve på koding, på skolen. Det er det jeg savner da. Så det er en utfordring på at jeg ikke kan det selv.»

Lærer D ønsker altså kurs i koding, med både Python og Scratch. I tillegg ønsker lærer D å få tips om hvordan man kan koble dette til ulike andre tema i matematikkfaget.

D: «Jeg vil ha kurs på hvilke muligheter har vi, for eksempel innen statistikk og sannsynlighet. Hvilke ord brukes i algebra, sånne ting. Det er det jeg på en måte vil ha kurs basert på selv. Ved at dem viser på smartboard eller projektor eller et eller annet. Hvor jeg på en måte blir elev da, og blir lært opp av noen som virkelig kan det. At vi bruker tid på det, for det blir jo bare mer og mer som sagt i alle fag. Og da er det jo dumt at ikke vi kan det selv. Men man trenger ikke være ekspert og kunne alt i hele verden, men at vi har et greit grunnlag selv da. Og det føler jeg at det har hvertfall ikke jeg. Det har jo blitt bedre som sagt da, for jeg har jo måtte gå inn i det. Jeg har jo ikke hatt så mye valg, men skulle ønske vi fikk liksom litt mer kurs før koding ble så stor del da, av faget.»

Generelt om programmering i skolen

Selv om noen av lærerne kan føle seg utrygge på egne kompetanser i programmering, kan det se ut til at alle har en relativt positiv innstilling til at det er kommet inn i matematikkfaget.

A: «Det gir mestringsfølelse, det er smart på tanke med hva slags verden vi lever i. Det er mange som har glede av dette, det er mange som synes at dette er gøy. Jeg er kjempepositiv til programmering i skolen.»

B: «Jeg liker jo veldig godt at programmering har kommet inn. Det var litt sånn skummelt i starten. Jeg visste ikke hva det var, men når jeg satt meg inn i det så synes jeg det er veldig bra. Hvertfall når jeg har sett mer og mer hvilken nytte det har. Hvor du kan bruke det i ulike temaer. I starten sleit jeg litt med å finne det ut. Hvordan kan jeg koble det her til kompetansemålene, men nå ser jeg mer og mer hvordan jeg kan gjøre det.»

I: «Er dette noe du har funnet ut av selv eller på studiet?»

B: «Nei, det har jeg funnet ut av selv, ikke så mye på grunn av studiet. Det var mere sånn at studiet gav deg oppgaver og du måtte gjøre mye ting selv. Så det baserte seg litt på hvor mye du selv orket å jobbe med det, hvor mye du fikk ut av det. For det var jo nettbasert, så du fikk høre en liten videosnutt også fikk du oppgaver. Også fikk du diskusjon. Så det var mer at de la til rette for at du kunne jobbe med det selv.»

C: «Det er jo det at det er framtiden da. Jeg tror det jeg. At flere og flere kan det. Men jeg tror det er viktig det at man ser på det som et hjelpemiddel og ikke som et tema da. Det er ikke en merbelastning i matte. Jeg tror man må se på det som et hjelpemiddel samtidig som elever som kanskje synes at matte er vanskelig så er det noen som synes at programmering er greit, og opplever liksom mestring i programmering.»

D: «Nei, eller jeg synes det er veldig bra at det kommer inn i skolen. Fordi vi har jo de, brutalt å si, nerdene, som elsker det her. De treffer vi på svært få områder da, bortsett fra koding. Så det er perfekt for de, og jeg ser jo at de blomstrer jo i mattetimene når vi har koding.»

Diskusjon

I dette kapittelet går jeg inn i de ulike områdene for undervisningskunnskap og ser lærernes svar i lys av dette. På denne måten analyseres dataene, for så å til slutt komme fram til en konklusjon på problemstillingen; **Hvilke undervisningskunnskaper har matematikklærere på ungdomskolen om programmering og hvilke undervisningskunnskaper føler de seg mer utrygge på?**

Grunnleggende innholdskunnskap

Her ser vi to eksempler der lærerne uttrykker at de er utrygge på sine egne programmeringsferdigheter.

C: «Jeg kjenner på det skikkelig fordi jeg kan ikke programmering. Jeg føler meg langt ifra trygg på det. Så jeg synes det er vanskelig. Jeg synes det er skummelt. Haha.»

D: «Men man trenger ikke være ekspert og kunne alt i hele verden, men at vi har et greit grunnlag selv da. Og det føler jeg at det har hvertfall ikke jeg.»

Common Content Knowledge

Som vist i resultatene har de fire informantene svært ulik fagkunnskap i programmering. Lærer D har minst kunnskap, med kun et kurs i Scratch på 1,5 time. Lærer A har mest fagkunnskap i programmering fra universitetet. Det kommer tydelig fram i intervjuet at lærer D er usikker på egne fagkunnskaper, mens lærer A er relativt trygg på dette og har også egen interesse for programmering. Dette ser vi hos lærer B og C også. Lærer A og lærer B interesserer seg for programmering og jobber mer med dette på fritiden, i tillegg til å ha søkt og gjennomført et årsstudium. Lærer C har vært skeptisk til programmering, men kommet seg i gang ved hjelp av et mindre kurs. Lærer D påpeker at programmering ikke er så gøy å holde på med hjemme. Dette viser at egen interesse kan være med å påvirke hvor mye kunnskap lærerne har tilegnet seg, da lærer C og lærer D er de lærerne som begge hevder at de ikke er helt trygge på programmering. Lærer B som søkte på et årsstudium på eget initiativ forteller også at dette la opp til mye selvstendig arbeid. Dette kan vise til at det er utfordrende for lærere å sette seg inn i den grunnleggende kompetansen i programmering om man ikke er engasjert for temaet i utgangspunktet.

Denne grunnleggende fagkunnskapen kan sees på som *common content knowledge* fra modellen til Ball (2008). Dette er allmenn kunnskap og er ikke spesiell for læreryrket, som vil si at det handler om å kunne å programmere, men ikke nødvendigvis å undervise i programmering. Derfor kan en si at selv om lærer B har et årsstudium, og lærer A har gått et fag på et semester, er det mulig at lærer A stiller høyere når det kommer til allmenn kunnskap. Dette fordi årsstudiet er designet for å videreutdanne lærere spesielt, og har derfor mer innhold som handler om å lage undervisningsopplegg. Vi ser at det er sprikende nivå av denne kunnskapen basert på hva disse fire lærerne forteller. Alle har tilegnet seg noe grunnlag gjennom erfaring i undervisning, men det kan se ut til at nivået på lærernes *common content knowledge* avhenger av egen interesse for programmering, i tillegg til tilrettelegging av kurs på skolene.

Specialized content knowledge

Når det kommer til *spescialized content knowledge*, spesialisert innholdskunnskap, er det noe mer utfordrende å drøfte lærernes nivå, basert på intervjuene. Dette handler som nevnt tidligere om kunnskap innen det å forstå hvordan en elev har tenkt for å komme fram til et eventuelt feilsvar (Ball et al., 2008). Lærerne har hatt lite prøver, og kan fortelle at de jobber med blant annet underveisvurdering, hverandrevurdering og muntlige tilbakemeldinger. I dette tilfellet kunne det vært behov for observasjon for å kunne analysere og drøfte i hvilken grad lærerne behersker å analysere hvordan elever kommer fram til en eventuell feilmelding i sin kode. Det er også diskuterbart i hvilken grad denne type undervisningskunnskap er overførbar fra matematikk til programmering. Lærerne påpeker at programmering har en annen type logikk enn vanlig matematikk. Det handler her om at lærerne trenger en dypere forståelse av løkker, variabler og vilkår for å kunne si noe om hvordan en misoppfatning kan føre til feilmeldinger. Læreren burde kunne analysere en kode og kunne si noe om hvordan eleven har tenkt, og dermed vite noe om hvordan eleven burde veiledes videre. Dette kan for eksempel handle om begrepsforståelse. Det kan også handle om selve oppbygningen av et programmeringsspråk og hvordan = tegnet for eksempel er relasjonelt i matematikk, mens det er operasjonelt når en skal kode i Python.

Lærer A, som er den læreren med mest utdanning og forkunnskaper i programmering, skiller seg ut når det kommer til å forstå elevutfordringer. Læreren forteller om mer detaljerte spesifikke faglige utfordringer elevene har.

A: «Og å forstå greia med løkker for eksempel. Eller at de teller variabler som blir oppdatert inni løkker da. De kan lett forstå at 1 og 1 er 2, men når de får en kode der det står $i = 1$ og $i = i + 1$, så skjønner de på en måte ikke at det legges på 1 på variabelen *i* hele veien da.»

Læreren her viser fagkunnskap og analyserer hva som gjør at elevene eventuelt ikke får oppdatert variabelen sin i en løkke. Dette handler altså om faglig spesialisert kunnskap som er spesifikk for læreryrket.

B: «... prøve meg på noen oppgaver så møter jeg jo på noe eller får noen feilmeldinger, men da må jeg på en måte prøve å finne ut hvor den feilmeldinga er og prøve å endre på koden. Litt sånn som elevene, så jeg møter jo på de samme problemene som elevene, men da har jeg lært litt av de problemene på forhånd ved å gjøre oppgavene.»

C: «Samtidig så tenker jeg at i og med at elevene liksom ikke har hatt det så mye heller, så får vi liksom prøve oss fram litt sammen da. Også håper jeg at jeg har noen elever som kan mye mer enn meg også bare prøve å finne ut av det sammen liksom.»

D: «Men jeg kan si at hva hvis det hadde vært sånn, får du til å gjøre det sånn at det ender opp sånn. Og da kan eleven bruke sin kunnskap til å få til, og vise meg da.»

Her ser vi noen utsagn fra lærerne hvor de forteller at de er på mer eller mindre samme nivå i programmering som elevene. Om spesialisert innholdskunnskap er overførbart til programmering, vil det si at lærerne, bortsett fra lærer A, mest sannsynlig ikke har et høyt nivå av spesialisert innholdskunnskap. Dette er noe en kan anta fordi de holder på å lære seg det mest grunnleggende i skrivende stund. Dette vil si at lærerne kan ha behov for mer spesialisert innholdskunnskap i programmering, kunnskap i programmering som er spesifikk for lærerprofesjonen. Det kan virke som at for å skape en slik spesialisert innholdskunnskap, trenger lærerne først en dypere forståelse for programmering i seg selv.

Lærernes fagkunnskaper, både spesialisert og vanlig innholdskunnskap, er en del av lærerens *foundation* og styrer derfor mange av valgene læreren gjør i løpet av undervisningen (Rowland et al., 2003). Dette kommer til syne i intervjuene på den måten at de som har et lavere nivå av fagkunnskap føler seg utrygge i undervisningen, og legger opp til mer selvstendig arbeid i matematikkundervisningen enn det de pleier.

C: «Så jeg har egentlig sluppet dem litt løs og latt dem prøve på egenhånd.»

D: «Det er veldig forskjellig fra lærer til lærer, men jeg synes det er veldig greit at de får utforska litt selv. At de får en bruksanvisning, følg det. Uten at jeg viser at sånn, sånn, sånn skal det gjøres, men at de er nødt til å følge en bruksanvisning for å, hva skal jeg si, bli kjent med det selv da.»

Lærer A som har mer grunnleggende ferdigheter i programmering forteller også at det blir noe mindre tavleundervisning. Det kommer likevel fram i intervjuet at dette er mye fram og tilbake, ved at de går gjennom oppgavene i fellesskap, heller enn at elevene jobber selvstendig hele timen. En kan derfor si at disse innholdskunnskapene i faget påvirker lærernes undervisning.

Positiv innstilling

B «Jeg liker jo veldig godt at programmering har kommet inn. Det var litt sånn skummelt i starten. Jeg visste ikke hva det var, men når jeg satt meg inn i det så synes jeg det er veldig bra.»

Læreres *foundation* handler også om hvilke holdninger læreren har til faget og hvilken nytte en tenker at dette vil ha for elevene. Her er lærerne svært enstemmig, uavhengig av forkunnskaper i programmering. Lærerne viser en positiv innstilling til at programmering har blitt en del av matematikkfaget. Dette er altså noe som kan være med å virke positivt inn i undervisningen. At lærerne har en tanke om at elevene kommer til å få bruk for dette i framtiden på ulike måter kan virke positivt ved at det blir et mer engasjerende arbeid å sette seg inn i.

Å se sammenhenger

Læreres *connection* handler som nevnt om å koble sammen ulike tema i matematikkfaget (Rowland et al., 2003). I forhold til programmering kan dette bety å ha kunnskap om hvilke tema innen matematikk det kan være relevant å bruke programmering. Dette er en type fagkunnskap, da det krever forståelse for programmering, for å kunne vurdere om det er mulig og relevant å benytte i sammenheng med for eksempel sannsynlighetsregning eller algebra. Dette er en vurdering lærerne hele tiden må gjøre når de legger opp sin undervisning. I intervjuene er det synlig at lærerne er opptatt av å koble inn programmering i ulike tema der det kan være til nytte. Det nevnes at programmering kan benyttes som et verktøy for å komme fram til gode løsninger på ulike matematiske problemer.

C: «For jeg har liksom en teori om at programmering bør være en del av de ulike temaene.»

D: «Det vi prøver på er å få knytta inn litt koding i alt hva vi driver med. Om det er geometri, om det er algebra, om det er sannsynlighet, så prøver vi å lage ett opplegg som handler om koding.»

B: «Ja! Eller hvert fall prøver å koble det opp mot tema vi jobber med sånn som nå jobba vi med sannsynlighet og da tenkte jeg at det var best i Python, tekstprogrammering, ... »

Lærer B forteller her om et opplegg som handler om å simulere store talls lov gjennom barnefødsler. Altså skal det kodes et program som viser at det er 50% sannsynlighet for å få gutt eller jente. Her har altså læreren tatt en god vurdering på hva som er mulig og relevant å gjennomføre i Python. Lærer B forteller også i intervjuet at det å koble ulike tema opp mot programmering ikke er noe hen lærte seg spesifikt på årsstudiet, men noe hen har blitt bedre på med erfaring og jobbing med emnet.

I: Er dette noe du har funnet ut av selv eller på studiet?

B: «Nei, det har jeg funnet ut av selv, ikke så mye på grunn av studiet. Det var mere sånn at studiet gav deg oppgaver og du måtte gjøre mye ting selv. Så det baserte seg litt på hvor mye du selv orket å jobbe med det, hvor mye du fikk ut av det.

Læreren har altså tilegnet seg kunnskap om hvordan programmering kan kobles opp til de ulike matematiske temaene gjennom å lage undervisningsopplegg på egenhånd. Det er tydelig at dette har lønnet seg. Når en ser på kategorien *connection*, ser en at de med mindre fagkunnskaper tenker selv at de sliter mer med å koble programmering til matematiske tema. Lærer D forteller at dette er noe hen kunne tenkt seg på en eventuell kursdag på arbeidsplassen.

D: «Jeg vil ha kurs på hvilke muligheter har vi, for eksempel innen statistikk og sannsynlighet. Hvilke ord brukes i algebra, sånne ting.»

En kan også se at Ball sin underkategori, *horizon content knowledge*, handler om mye av det samme som *connection*. *Horizon content knowledge,* horisontkunnskap, handler om å se sammenhengene i faget. I praksis vil det si hvilken kunnskap elevene har med seg, hvor de er nå, og hvor de skal videre med denne kunnskapen (Ball et al., 2008). Lærerne har hatt fokus på å komme seg på et faglig nivå der de kan undervise i programmering på ungdomstrinnet, og det ser derfor ut til at de ikke har satt seg inn i hvordan denne kunnskapen skal bygges videre på i skoleløpet når elevene skal over til videregående. På spørsmål om hva elevene skal bruke programmeringskunnskapen til videre, svarer alle lærerne mer generelt på hvordan elevene kan bruke dette videre i livet. Det kan derfor tenkes at lærerne ikke er på noe høyere faglig nivå i programmering, som gjør at de vet hvordan løkker og variabler henger sammen med hva elevene skal lære seg senere. Når det kommer til den algoritmiske tenkeren, har lærerne mer tanker om hvor nyttig dette er for elevene å lære. Lærer D nevner at programmering handler om å «planlegge, gjennomføre og evaluere». Dette kan sees på som en kort beskrivelse av hva den algoritmiske tenkeren gjør. Programmeringsundervisningen blir altså sett på som noe som gir elevene et nyttig problemløsningsverktøy, både for de som skal ut i teknologiske yrker, men også for alle som trenger å planlegge, gjennomføre og evaluere eget arbeid i fremtiden.

Lærerne viser noe mer faglig kunnskap om hva elevene har med seg fra barnetrinnet, og hvordan de derfor ønsker å bygge på dette fra og med 8. trinn. Lærer B forteller at de

begynner med Scratch på 8. trinn for så å bevege seg over til det skriftlige programmeringsspråket, Python, på 9. trinn. Alle lærerne forteller at de benytter seg av Scratch, men går ikke nærmere inn på hvilken kobling dette har til Python. Lærer C forteller mer spesifikt om hva de fremtidige elevene vil ha som utgangspunkt når de kommer til ungdomstrinnet.

C: «Når de kommer til oss så har de liksom en tanke om en del, eller sånn, variabler og vilkår og løkker og sånn. Så de vet liksom hva det er da og de har liksom jobbet med det i boksprogrammering. Så når de kommer til oss og skal begynne med skriveprogrammering så vil den overgangen gå lettere da.»

Lærer C viser her en kunnskap om at blokkprogrammering i Scratch danner grunnlaget for videre læring i skriftlige programmeringsspråk. Lærer C har altså en forståelse for hvordan kunnskaper i algoritmisk tenkning og analog programmering med fokus på begreper bidrar til bedre forståelse av Python, når elevene skal begynne med dette. Læreren vet noe om hvor elevene kommer fra faglig, men noe mindre om hvor de skal videre med programmering. At de andre lærerne ikke går inn på denne koblingen mellom Scratch og Python i intervjuet betyr ikke nødvendigvis at de ikke ser koblingen. Det er uansett et godt utgangspunkt at alle benytter dette som en inngang til programmering på ungdomstrinnet, om det er bevisst eller noe som er blitt en standard på skolen.

Undervisning

I denne oppgaven handler *transformation* om å se på hva slags type opplegg, eksempler og ressurser lærerne bruker i sin programmeringsundervisning. I hvilken grad de har kunnskaper og evner til å lage gode opplegg som lærer elevene å kode. Det må igjen nevnes at dette ikke er noen vurdering av hvor gode matematikklærere informantene er, men en undersøkelse av hvilke undervisningskunnskaper lærerne har klart å tilegne seg i det nye temaet programmering, på den korte tiden det har vært en del av læreplanen.

Lærer A	Lærer B	Lærer C	Lærer D
Scratch Python	Scratch Python «Kåres kokebok» Kidsa koder (nettside) MicroBit (nettside)	Scratch Python Campus «Matemagisk» ProFag:U (nettside)	Scratch Python Teams (delingsplattform)

Tabell 3

Ut ifra tabell 3 ser vi at alle lærerne tar utgangspunkt i å benytte Scratch og Python i programmeringsundervisningen. Lærer B og lærer C påpeker at Scratch benyttes i første omgang, for så å gå videre til Python, som er et mye brukt programmeringsspråk. Dette kan støttes av undersøkelsen gjort om Scratch i 2013. I Scratch kan elever lære seg konseptene innen programmering uten å måtte forholde seg til de detaljerte syntaksene i et programmeringsspråk. Dette gjør elevene bedre rustet for å lære seg Python i etterkant (Armoni et al., 2015). Utenom dette bruker lærerne ulike ressurser til å planlegge sin undervisning. Den største forskjellen er at lærer A lager oppleggene på egenhånd. Lærer B forteller at oppgaver innhentes, men ikke ukritisk. Lærer B tilpasser oppleggene til læreplanen og bruker derfor innholdskunnskaper i faget og læreplankunnskaper til å lage et best mulig opplegg. Lærer C og lærer D forteller at de benytter seg av oppgaver i matteboken og opplegg laget av andre lærere. Dette betyr ikke nødvendigvis at det er ukritiske valg, men basert på en tillit til at andre har den kunnskapen som kreves for å lage et tilpasset opplegg i programmering. Lærernes *transformation* handler om evnen til å legge fram faginnholdet for elevene og for noen kan dette være mer krevende, om man ikke er veldig trygg på faginnholdet selv. Som lærer C sier; «jeg føler meg ikke liksom kvalifisert til å stå foran en gruppe og forklare så fryktelig mye.»

Content knowledge and teaching

Opplegget med store talls lov er et eksempel på hvordan lærer B legger fram faginnhold i programmering for sine elever. Dette opplegget baserer seg på en oppgave som starter på et enklere nivå, og som utvikler seg til å bli noe mer avansert etter hvert. Dette er noe lærer B forteller at hen stort sett har fokus på i sin undervisning i programmering, men også generelt i

matematikktimene. Dette opplegget er derfor med å tilpasse programmeringsundervisningen for alle elevene, noe som også er svært viktig i læreplanen. Oppgaven om store talls lov er en såkalt LIST-oppgave, da den har lav inngangsterskel, og høy takhøyde når elevene får muligheten til å modifisere koden. Måten lærer B legger opp undervisningen ligner også på PRIMM-metoden.

B: «Så det handler mye om at elevene skal skrive av en kode eller på forhånd gjette hva koden gjør slik at de forstår hva det handler om. Også går det mer på da etterpå at de skal bygge ut koden eller skal manipulere koden for å få den til å vise noe annet da.»

Elevene er altså innom tre av fem faser i PRIMM i det som beskrives her, altså å forutsi, å teste koden og å modifisere den. Lærer A viser også tegn til denne metoden. Det ser ut til at dette er en metode som naturlig har kommet inn med programmering hos flere av lærerne. Ved at elevene får en kode som de skal først prøve å forstå og deretter jobbe videre med og utvikle og redigere. Det kan tyde på at det er gode metoder som har blitt del av lærernes videreutdanning, kurs og eventuelle lærebøker.

At lærer B får til å utvikle slike opplegg, viser tegn på et godt nivå når det kommer til *knowledge of content and teaching*. Lærere må være i stand til å vurdere hva slags oppgaver som er egnet for de ulike læreplanmålene. I dette tilfellet er det tatt utgangspunkt i læreplanmålet fra 9. trinn der det elevene skal kunne «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe ved å bruke programmering» (Kunnskapsdepartementet, 2020). Elevene får teste ut hva sannsynligheten er for å få en gutt eller jente ved en fødsel, og må deretter redigere koden for at dette skal simulere noe annet. For eksempel terningkast. Ifølge lærer B starter det enkelt ved at alle kan gjette på eller finne ut av hva koden gjør. Deretter blir det mer utfordrende når elevene må vise forståelse for koden ved å tilpasse den videre. Denne måten å jobbe på viser flere av lærerne at de bruker når de forteller om at elevene må forstå koden, redigere eller gjøre endringer i koden, og samarbeide mye underveis. Det er altså på denne måten lærernes *knowledge of content and teaching* kommer til syne. Hvor godt opplegget om å simulere store talls lov har fungert i klassen, er vanskelig å si uten observasjon. Disse resultatene er altså kun basert på hva lærer B forteller om et opplegg, som læreren selv mente har fungert godt. Det må også nevnes at lærer D har valgt å ha bruke samme opplegg i sin klasse, noe som kan tyde på at det har vært

et vellykket opplegg på denne skolen. I tillegg kan denne metoden å undervise på støttes opp av forskning som effektiv for læring.

Knowledge of content and students

Knowledge of content and students handler som nevnt om blant annet å kjenne til vanlige misoppfatninger, utfordringer og hva som er motiverende for elever (Ball et al., 2008). Dette er en viktig del av læreres undervisningskunnskap, for å kunne planlegge gode opplegg som er tilpasset elevene. Lærerne som er intervjuet i denne oppgaven har gjort seg opp noen ulike oppfatninger av elevenes utfordringer i programmering. Utholdenhet ved prøving og feiling kommer opp i intervjuene som en stor utfordring. Dette er en stor del av det den algoritmiske tenkeren gjør. Det må til for å kunne løse en oppgave på en mest mulig effektiv måte. Elevene må lære seg å evaluere eget arbeid og korrigere feil eller forbedre koden sin. Ut ifra det lærer B og lærer C forteller er utholdenhet en utfordring for elevene. Dette kan komme av at elevene ikke er vant med å jobbe på en slik måte, men er vant til å skulle komme fort fram til et svar. Dette er noe de jobber generelt med i matematikkfaget for å endre på.

C: «Det er noe med den der utholdenheten vi leter litt etter i matte. Det der å stå i utfordringer som de kanskje ikke løser med en gang, og få lov å utforske. Det er litt sånn læreplanen er veldig opptatt av utforsking og utholdenhet og den biten. Jeg tenker at der har jo programmering en kjempefordel.»

Å sitte lenge med en oppgave alene og ikke få til kan være frustrerende. Lærer B har gjort evaluering av oppgaver til en felles klasseaktivitet. Elevene får se løsningsmetoder på tavlen og dermed diskutere seg fram i fellesskap. Læreren setter altså her veien til mål i fokus framfor korrekt svar, i dette tilfellet at koden fungerer. Elevene får dermed diskutere effektiviteten og kvaliteten på en oppgavebesvarelse. Her møter læreren en utfordring elevene har ved å hjelpe dem med å stå i et problem, selv om de allerede har kommet fram til en løsning.

Lærer A og lærer C påpeker detaljert syntaks i Python som en utfordring for elevene. Det er ikke spesifisert i læreplanen at det skal brukes et skriftlig programmeringsspråk på ungdomstrinnet, men det ser ut til at det er dette som er tatt i bruk på skolene i denne studien,

og i norske lærebøker for ungdomstrinnet (Kunnskapsdepartementet, 2020). Utfordringen med detaljert syntaks er noe som er prøvd å overkomme ved bruk av «Parsons problems» (Ericson et al., 2017). At lærerne påpeker akkurat denne utfordringen, viser at de allerede har gjort seg noe kjent med typiske elevutfordringer i programmering gjennom erfaring. Lærerne nevner «Parsons problems» som en måte å løse utfordringen på, men lærer C løser det ved å se på feilmeldingene sammen med elevene.

Lærer A viser tydelig kjennskap til den vanlige utfordringen elever har med løkker.

A: «Og å forstå greia med løkker for eksempel. Eller at de teller variabler som blir oppdatert inni løkker da. De kan lett forstå at 1 og 1 er 2, men når de får en kode der det står $i = 1$ og $i = i + 1$, så skjønner de på en måte ikke at det legges på 1 på variabelen *i* hele veien da.»

Denne utfordringen kan støttes av forskning som spesielt peker på løkker og variabler som en utfordring for elever (Grover et al., 2015). Dette handler ikke bare om at elevene skal skrive korrekt av en oppskrift, men at de må få en forståelse for hva en variabel er og hva en løkke faktisk gjør. Læreren forteller ikke noe særlig om hvordan denne utfordringen blir møtt, men at det jobbes med å skrive korrekt i fellesskap. Dette blir en veldig instrumentell måte å jobbe på, og elevene kunne ha dratt nytte av å jobbe mer med begrepsforståelse, kanskje i form av analog programmering i forkant. Her må det tas høyde for at elevene som nå går på ungdomstrinnet ikke har jobbet med algoritmisk tenkning og koding på barnetrinnet, slik som framtidige elever vil komme til å ha gjort. Dette funnet kan tyde på at høyere fagkunnskap enn elevene i programmering, slik som lærer A har, bidrar sterkt til å oppdage og derfor forutse elevers utfordringer eller misoppfatninger og dermed bedre *knowledge of content and students*. Det viser derimot ikke at det nødvendigvis bidrar til bedre didaktiske metoder eller evner innen *knowledge of content and teaching*. En kan altså påstå at ikke alle lærerne i denne studien har den faglige tyngden til å oppdage alle elevenes utfordringer programmering. At dette derfor er noe mer utrygg grunn for dem, enn «vanlig» matematikk. De oppdager som sagt noen generelle utfordringer, som utholdenhet, men går lite inn på programmeringsspesifikke utfordringer.

Trygghet og uforutsette innspill

Knowledge of content and teaching handler også om det å ha evne til å velge ut elevinnspill, og hvordan bygge videre på disse. Dette ligner på kategorien *contingency* fra kunnskapskvartetten kommer til syne gjennom valgene lærerne tar i timen. Det handler om at lærerne må kjenne godt til innholdet i faget for å se hvilke innspill som er relevant i forhold til målet for timen. Hvordan takler lærerne uforutsette innspill i programmering? Igjen ville det vært positivt for denne oppgaven å ha hatt en bekreftende observasjon av det lærerne forteller. Likevel har lærerne i intervjuene kommet med utsagn rundt hvordan de takler innspill i timene og hvordan de opplever dette, noe som kan være vel så interessant.

Lærer C forteller blant annet at noen av elevene som er sterke i programmering trenger utfordringer. Noen ganger har elevene kommet lenger i pensum enn det læreren selv har lært på kurset. Dette gjør det vanskelig for læreren å veilede eleven videre, og å ta tak i spørsmålene som dukker opp. Likevel, på grunn av erfaring, forteller læreren at dette går greit. Læreren benytter sin kunnskap om læreplanen og algoritmisk tenkning til å heller la eleven utforske på egenhånd. Dette synliggjør hvor mye de ulike kunnskapsområdene henger sammen. Når eleven må finne ut av hvordan man bruker *if/else*, altså vilkår, blir eleven nødt til å google seg fram. Eleven må da finne en egnet fremgangsmåte, uten å bli gitt en «oppskrift» av læreren. Dette kan derfor kalles problemløsning, som er en del av algoritmisk tenkning. Eleven får prøve, feile og vurdere hva som er best og mest effektiv måte å løse problemet på. Lærer D gjør det også på samme måte når det kommer til å gi utfordringer til sterke elever.

I: «Du føler deg trygg nok i programmering til å gi utfordringer?»

D: «Ja, eller det handler ikke bare om programmering da men. Uansett om det er programmering da, så skjønner jeg. Jeg veit at du kan gjøre ekstremt mye i Scratch og Python, men jeg vet ikke selv hvordan man gjør det. Men jeg kan si at hva hvis det hadde vært sånn, får du til å gjøre det sånn at det ender opp sånn. Og da kan eleven bruke sin kunnskap til å få til, og vise meg da.»

Det kan tyde på at lærerne overfører undervisningskunnskap basert på erfaring fra matematikktimene og inn i programmeringsundervisningen. For elever som behersker

programmering og interesserer seg for det, kan dette være en god løsning. For elever som derimot ikke interesserer seg, kan det være utfordrende å sette seg inn i noe nytt på egenhånd, slik som vi har sett med lærerne som er intervjuet. Disse lærerne ønsker seg mer kurs og opplæring, og det kan derfor antas at dette også kan være gjeldene for elever som ikke interesserer seg for programmering på fritiden.

Lærerne har fortalt at det krever mer forarbeid før en time med programmering enn en time med for eksempel Pytagoras. Det er mer for lærerne å sette seg inn i da grunnkunnskapen ikke er like stor når det kommer til programmering.

C: «Også har jeg såpass mange års erfaring at jeg kan liksom slippe kontrollen litt og la det surre og gå litt. Og la elevene få utforske. Så jeg føler meg ikke liksom kvalifisert til å stå foran en gruppe og forklare så fryktelig mye. Fordi jeg kan det samme som elevene liksom.»

Selv om lærerne føler seg trygge på å ta imot uforutsette innspill, kan det antas at lærerne ikke er like forberedt på å vurdere hva slags spørsmål eller innspill som er relevant for dagens tema, hvis de ikke er trygge på egne forklaringer. Det er altså ikke sikkert at de har grunnkunnskapen til å svare på alle spørsmål som dukker opp. Dette gjør det vanskelig å spille videre på hva elevene bidrar med i timen.

Læreplankunnskap – Curricular Knowledge

Ifølge Ball må *curricular knowledge*, eller på norsk læreplankunnskap, sees i sammenheng med de andre kategoriene. I dette delkapittelet tas det utgangspunkt i Shulmanns definisjon av *curricular knowledge*. Det handler altså om i hvilken grad lærerne har kunnskap om ulike undervisningsmetoder for ulike tema. Lærerne virker å ha fokus på elevsamarbeid generelt sett i sin matematikkundervisning. Lærer A, som tidligere nevnt har mest fagkunnskaper, forteller at undervisningen er stort sett lik og virker ikke som å bruke ulike metoder for å møte elevutfordringer. Lærer B som har gått videreutdanning for lærere i programmering virker å ha tilegnet seg en høyere grad av læreplankunnskap. Lærer B forteller at når det skal lages undervisningsopplegg, blir det alltid tatt utgangspunkt i lærerplanen. Dette viser først og

fremst god kjennskap til læreplaninnholdet i forhold til undervisning. Videre viser lærer B god kjennskap til algoritmisk tenkning og kjerneelementene i læreplanen.

B: «Men sånn klassisk vanlig matematikk-time, så spørs det litt hva tema er da, men læreplanen har jo mye sånn utforskende så jeg prøver å i stedet for å si SÅNN er vi regner ut Pytagoras så får jeg heller elevene til å utforske fenomenet. Det blir jo heller mye sånne typer oppgaver da, utforskende oppgaver. Ikke så mye mengdetreningsoppgaver, men mer oppgaver som gjør at elevene må samarbeide med hverandre, diskutere med hverandre, prøve seg fram, ser de noe sammenheng, er det noe mønster her. Jeg tenker jo veldig mye på den algoritmiske tenkeren hele tiden i matematikk.»

Alle lærerne er selvfølgelig vant med å jobbe ut ifra læreplanen. Likevel skiller lærer B seg ut her ved å ha god kjennskap til ulike måter å jobbe utforskende på. Det vil si en god læreplankunnskap, da læreren kan fortelle om ulike måter de jobber med programmering på. Både ved å jobbe selvstendig, utforskende, problemløsende og ved å vurdere elevsvar i fellesskap.

Algoritmisk tenkning og utforskende oppgaver

Algoritmisk tenkning er nevnt som en viktig del av prosessen i et av kjerneelementene i læreplanen for matematikk (Kunnskapsdepartementet, 2020). Det kan beskrives som en måte å arbeide på eller som en måte å tenke på, som du kan lære deg ved for eksempel programmering. Spørsmålet her er om lærerne benytter seg av dette når elevene skal lære seg å programmere. Lærer C forteller at programmering har en fordel når det kommer til å øve på utholdenhet, da det er mye prøv og feil. Dette er en del av det den algoritmiske tenkeren gjør, og på den måten kan du si at læreren benytter seg av dette og har satt seg inn i hva det vil si å tenke og arbeide algoritmisk. Lærer C forteller også at elevene samarbeider mye i undervisningen, selv om alle må jobbe på egen PC. Det er også en del av det å jobbe algoritmisk, da en må diskutere seg fram til løsninger.

B: «Men mye går på forståelse, hva er det koden gjør og hvorfor printer den ut det den printer ut. Så det handler mye om at elevene skal skrive av en kode eller på forhånd

gjette hva koden gjør slik at de forstår hva det handler om. Også går det mer på da etterpå at de skal bygge ut koden eller skal manipulere koden for å få den til å vise noe annet da.»

Lærer B er opptatt av at elevene skal forstå og forteller også at hen benytter seg mye av teorien om algoritmisk tenkning generelt i sin matematikkundervisning. Læreren har altså et syn på algoritmisk tenkning som en metode som ikke kun er knyttet til programmering. Oppleggene bærer preg av utforsking, samarbeid og diskusjon. Elevene må se etter mønstre og sammenhenger i matematikken. Dette passer altså inn under definisjonen av utforskning i læreplanen (Kunnskapsdepartementet, 2020). Vi kan også se at det passer godt inn i modellen til Shute (2017) om algoritmisk tenkning. Ved å gjette hva en kode gjør blir elevene nødt til å dele opp problemet og analysere dataene. Når elevene skriver egen kode må de også lage steg-for-steg oppskrifter, noe lærer B også har fortalt at de jobber med i analog programmering. Ved å manipulere koden må elevene jobbe med generalisering, og de jobber også med å lage effektive løsninger når de jobber med hverandrevurdering. Denne måten å undervise på passer også inn under undersøkelseslandskapet til Skovsmose et al. (2006), da elevene får muligheten til å diskutere seg fram, og de velger selv hvilke sammenhenger de vil utforske i det matematiske temaet. Her kan vi se at læreren er åpen og ser muligheter ved algoritmisk tenkning og programmering som kan være med å forbedre matematikklæringen (Kilhamn et al., 2021).

Om elevene jobber utforskende i undervisningen, kommer også an på oppgavene som blir brukt, og hvilken definisjon du bruker av utforskende oppgaver. Om oppgavene er slik at elevene får prøve og feile, og finne sin egen løsningsmetode kan en si at det er utforskende. Om programmeringsoppgavene er slik at elevene skal gjengi en kode og følge en oppskrift for å få en spesifikk løsning, kan en argumentere for at dette ligner mer på ett oppgaveparadigme. Selv om elevene får slippe løs og jobbe på egenhånd slik lærer D beskriver sin programmeringsundervisning, må elevene likevel følge en oppskrift. Det kan derfor argumenteres for at ikke alle lærerne er enig om hva det vil si å utforske.

Lærer B og lærer A som tilpasser egne oppgaver, er mer nødt til å tenke over hvordan disse skal formuleres for at elevene skal få muligheten til å utforske. Ut ifra det som blir fortalt, kan det virke som elevene får mulighet til å utforske med begge lærerne. Lærer A forteller at i timene, gjør elevene slik som hen gjør på tavla, og dermed gjengir en spesifikk

løsningsmetode. Denne delen av undervisningen ligner oppgaveparadigmet (Skovsmose et al., 2006). Likevel får de utfordringer i hvordan de kan endre koden og tilpasse den til andre situasjoner. Her blir elevene nødt til å utforske noe, og de må være kreative for å løse et problem. Her kommer man inn på den delen ved algoritmisk tenkning som handler om å automatisere løsningen. Elevene må generalisere og gjøre nødvendige endringer som gjør at den passer med andre lignende problemer. Når læreren tar oppgaver eller opplegg som er ferdig og klar til bruk, slik som lærer C og lærer D, må en gå ut ifra at de som har laget opplegget eller skrevet boka har tatt hensyn til å lage utforskende oppgaver som innebærer algoritmisk tenkning. En kan derfor anta at alle benytter algoritmisk tenkning og utforskende oppgaver i sin undervisning, men det ser ut som at noen gjør dette mer bevisst enn andre og har mer fokus på nettopp dette.

Avslutning

Konklusjon

Jeg vil først prøve å besvare første del av problemstillingen, om hvilke undervisningskunnskaper matematikklærerne har om programmering. Lærerne i denne studien har som nevnt et variert utgangspunkt av kunnskaper innen programmering. De har også ulike erfaringer i læreryrket. For eksempel har lærer A erfaring innen realfag, mens lærer C har mange års erfaring i lærerprofesjonen og matematikkundervisning. Dette viser hvor stort spekteret av undervisningskompetanse i skolen kan være. Det burde tilrettelegges for, når lærerne skal kurses i programmering. Noen har behov for kurs innen Python, mens andre har kanskje behov for undervisningsmetoder. Lærer A og B ser ut til å ha mer kompetanse når det kommer til programmering i seg selv, og har mer kunnskaper om Python. Lærer B og lærer C har fått kurs i programmering for lærere, noe som ser ut til å ha gitt de mer kunnskaper om ulike undervisningsmetoder og derfor *knowledge of content and students*. Vi ser også at de grunnleggende programmeringskunnskapene påvirker spesielt *special content knowledge*, der lærer A skiller seg ut med fagspesifikk kunnskap.

Alt i alt kan det se ut til at lærerne har behov for noe mer kompetanse, jeg besvarer her andre del av problemstillingen; hvilke undervisningskunnskaper føler de seg mer utrygge på? Det er varierende hva slags undervisningskompetanse lærerne har behov for å utvikle, da lærer A kanskje har behov for mer innføring i undervisningsmetoder og å utvikle seg på *transformation*, har lærer C mer behov for en høyere faglig kompetanse innen programmering. Den rene fagkompetansen er kanskje det som står ut som viktigst her. Lærerne viser at de har tilgang til gode undervisningsopplegg og kan med erfaring i yrket tilpasse seg undervisningssituasjonen. Å forbedre den grunnleggende fagkunnskapen i programmering er derimot noe lærerne er nødt til å få tilbud om. Ut ifra denne studien ser det nå ut til at dette er noe kun de som interesserer seg for programmering er gode på. Om en sammenligner med «vanlig» matematikk, har lærerne et høyere nivå av fagkunnskap enn grunnskolenivå. Dette ser vi at det er et behov for også i programmering. Denne grunnleggende kunnskapen ser vi ut ifra lærer A sitt intervju også er med å påvirke de andre kunnskapsområdene positivt. Dette er spesielt viktig da undervisningskunnskap i matematikk ikke nødvendigvis er overførbart på alle områder som for eksempel *knowledge of content and students*, eller å tilpasse seg uforutsette innspill fra elever. Vi ser at lærernes gode erfaring

trygger dem i denne situasjonen. Likevel, med bedre faglig kunnskap i programmering enn å være tilnærmet elevenes nivå, vil lærerne kunne tilpasse enda bedre, lettere kunne svare på spørsmål, oppdage misoppfatninger og veilede elevene i riktig retning. Når det kommer til læreplankunnskap vil jeg trekke fram den algoritmiske tenkeren og utforskende oppgaver som noe lærerne kunne hatt behov for å bli tryggere på. Dette er en undervisningsmetode lærerne selv forteller at programmering har en fordel i, men likevel er det kun en av de intervjuede lærerne som er veldig bevisst på å benytte dette i planleggingen. Dette kunne lærerne hatt nytte av å ha kurs i for å klargjøre hva disse begrepene betyr, og for å sette seg inn i hvordan man benytter algoritmisk tenkning og utforskning i praksis. Dette kan være med å styrke lærernes *knowledge of content and students*. Dette sier også lærer D spesifikt at er et ønske å få kurs i, hvordan man underviser i programmering i de ulike matematiske temaene. En kan si at det er behov for det Shulman beskriver som *curricular knowledge*.

Veien videre

I denne studien er funnene basert på informantenes egne opplevelser av hvor trygg man føler seg i å undervise programmering. Det har kommet fram interessante funn om hvordan lærerne har tilegnet seg ulike kunnskaper med ulike tilbud om kurs og videreutdanning. Lærerne ser ut til å ha håndtert overgangen godt med tanke på de forutsetningene de hadde innen programmering. Denne studien er som nevnt ikke generaliserbar for alle matematikklærere i Norge. Det er likevel interessant å se at noen fortsatt bare har hatt 1,5 time med opplæring i Scratch som eneste kurs i programmering på arbeidsplassen.

For videre forskning vil det være interessant å observere matematikklærere på ungdomstrinnet for å se hvordan undervisningskunnskaper i programmering faktisk kommer til syne. Det kan da komme fram valg lærerne gjør i løpet av en undervisningsøkt, noe som ikke kommer fram i et intervju. Lærere gjør mange valg i løpet av en matematikktime, som de ikke nødvendigvis er klar over selv. Det ville være interessant å se om lærerne da har mer undervisningskunnskap enn de selv er klar over at de benytter seg av. Det kunne også være interessant med en kvantitativ studie for å undersøke hva slags tilbud matematikklærerne i Norge har fått, for å øke kompetansen innen programmering. Det er noe denne studien ikke kan si noe om, men en kan se bare ut ifra fire informanter at det er stor variasjon.

Litteraturliste

Ahmed, G., Nouri, J., Zhang, L. & Norén, E. (2020). Didactic Methods of Integrating Programming in Mathematics in Primary School: Findings from a Swedish National Project. Proceedings of the 51st ACM technical symposium on computer science education,

Armoni, M., Meerbaum-Salant, O. & Ben-Ari, M. (2015). From Scratch to "Real" Programming. *ACM transactions on computing education*, *14*(4), 1-15. https://doi.org/10.1145/2677087

Balanskat, A. & Engelhardt, K. (2014). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet. http://www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf/d3780a64-1081-4488-8549-6033200e3c03

Ball, D. L., Thames, M. H. & Phelps, G. (2008). Content knowledge for teaching: What makes it special?

Bray, W. & Santagata, R. (2014). Making mathematical errors springboards for learning. *Annual Perspectives in Mathematics Education (APME)*.

Christoffersen, L. & Johannessen, A. (2012). *Forskningsmetode for lærerutdanningene*. Abstrakt forl.

Ericson, B. J., Margulieux, L. E. & Rick, J. (2017). *Solving parsons problems versus fixing and writing code*. Proceedings of the 17th Koli Calling International Conference on Computing Education Research, Koli, Finland. https://doi.org/10.1145/3141880.3141895

Erstad, O., Kjällander, S. & Järvelä, S. (2021). Facing the challenges of 'digital competence' a Nordic agenda for curriculum development for the 21st century. *Nordic Journal of Digital Literacy*, *16*(2), 77-87.

Flø, E. E. (2021). Programmering i LK20. *Tangenten: tidsskrift for matematikkundervisning, 32 (1)*, 3-9.

Grover, S., Pea, R. & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer science education*, *25*(2), 199-237. https://doi.org/10.1080/08993408.2015.1033142

Harms, K. J., Chen, J. & Kelleher, C. L. (2016). *Distractors in Parsons Problems Decrease Learning Efficiency for Young Novice Programmers*. Proceedings of the 2016 ACM Conference on International Computing Education Research, Melbourne, VIC, Australia. https://doi.org/10.1145/2960310.2960314

Kilhamn, C., Bråting, K. & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. NORMA20,

Kleve, B. & Hovik, E. K. (2016). *Undervisningskunnskap i matematikk*. Cappelen Damm akademisk.

Kunnskapsdepartementet. (2020). *Læreplan i matematikk 1.–10. trinn* (MAT01-05). https:/www.udir.no/lk20/MAT01-05

Kvale, S., Brinkmann, S., Anderssen, T. M. & Rygge, J. (2015). *Det kvalitative forskningsintervju* (3. utg. utg.). Gyldendal akademisk.

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in human behavior*, *41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Mannila, L. (2017). *Att undervisa i programmering i skolan : varför, vad och hur?* (Upplaga 1. utg.). Studentlitteratur.
Postholm, M. B., Jacobsen, D. I. & Søbstad, R. (2018). *Forskningsmetode for masterstudenter i lærerutdanningen*. Cappelen Damm akademisk.
Rowland, T., Huckstep, P. & Thwaites, A. (2003). The knowledge quartet. *Proceedings of the British Society for Research into Learning Mathematics*, *23*(3), 97-102.
Sentance, S. & Csizmadia, A. (2015). Teachers' perspectives on successful strategies for teaching Computing in school. IFIP TC3 Working Conference 2015: A New Culture of Learning: Computing and Next Generations,
Sentance, S., Waite, J. & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer science education*, *29*(2-3), 136-176. https://doi.org/10.1080/08993408.2019.1608781
Sevik, K. (2016). Programmering i skolen. I. Senter for IKT i utdanningen. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf
Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational researcher*, *15*(2), 4-14.
Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational research review*, *22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003
Skovsmose, O., Blomhøj, M. & Alrø, H. (2006). *Kunne det tænkes? : om matematiklæring*. Malling Beck.
Sun, L., Hu, L. & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking skills and creativity*, *42*, 100926. https://doi.org/10.1016/j.tsc.2021.100926
Utdanningsdirektoratet. (2010, 20. januar). *Kunnskapsministeren åpnet IKT-senter*. https://www.regjeringen.no/no/dokumentarkiv/stoltenberg-ii/kd/Nyheter-og-pressemeldinger/nyheter/2010/kunnskapsminister-kristin-halvorsen-apne/id591521/
Utdanningsdirektoratet. (2019). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/
Wæge, K. & Nosrati, M. (2018). *Motivasjon i matematikk*. Universitetsforl.

Vedlegg

A: Informasjonsskriv

Vil du delta i forskningsprosjektet
" Programmering for lærere"

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å undersøke matematikklæreres kompetanse i programmering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål

Jeg vil bruke dataene fra undersøkelsen til min masteroppgave. Oppgaven dreier seg om hvilke kunnskaper lærere må tilegne seg på egenhånd og hvilke kunnskaper de får kurs/opplæring i. Jeg vil også gå dypere inn på hvilke av disse kunnskapene som er mest relevant for lærerne å ha. Min problemstilling for oppgaven er:

- Hvilke kunnskaper har matematikklærere på ungdomskolen om programmering og hvilke kunnskaper har lærerne mer behov for?

Hvem er ansvarlig for forskningsprosjektet?
OsloMet er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?

Jeg ønsker å intervjue lærere på ulike skoler og gjerne ulik erfaring med programmering. Jeg er spesielt interessert i lærere som jobber med matematikk på ungdomstrinnet..

Hva innebærer det for deg å delta?

Hvis du velger å delta i prosjektet, innebærer det å bli intervjuet. Under intervjuet vil jeg stille spørsmål rundt dine kunnskaper og erfaringer med programmering i matematikkfaget på ungdomstrinnet. Dette vil ta opp til 45 minutter. Jeg vil ta lydopptak og notater fra intervjuet. Dette blir lagret sikkert gjennom UiO´s Nettskjema-diktafon og slettes når jeg har transkribert og anonymisert intervjuet.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger

Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

- Veileder for oppgaven; Dr. Andre Rognes vil ha tilgang til lydopptaket.
- Student; Ingrid Voldmo vil ha tilgang til lydopptaket.
- For at uvedkommende ikke skal få tilgang til navn og kontaktopplysninger vil ikke dette inngå i transkripsjon eller notater.
- Lydopptaket lagres kun i Nettskjema-diktafon og ikke på privat enhet.
- I oppgaven vil skolen og deltakeren være anonymisert.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene anonymiseres senest når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er 16.05.2022. Personopplysninger/lydopptak slettes så tidlig som mulig før dette.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra OsloMet har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter

Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

- Ingrid Voldmo på epost (ingridvoldmo@gmail.com) eller telefon: 938 31 848
- OsloMet ved Dr. Andre Rognes (posjektansvarlig) på e-post (andro@oslomet.no) eller telefon: 67 23 71 39

- Vårt personvernombud: Ingrid Jacobsen ved OsloMet på epost (ingrid.jacobsen@oslomet.no) eller telefon; 67 23 55 34

Hvis du har spørsmål knyttet til NSD sin vurdering av prosjektet, kan du ta kontakt med:

- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Dr. Andre Rognes
(Forsker/veileder)

Ingrid Voldmo
(Student)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Programmering for lærere», og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i intervju

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltaker, dato)

B: Vurdering fra NSD

Meldeskjema / Programmering for lærere / Vurdering

Vurdering

Referansenummer
454089

Prosjekttittel
Programmering for lærere

Behandlingsansvarlig institusjon
OsloMet – storbyuniversitetet / Fakultet for lærerutdanning og internasjonale studier / Institutt for grunnskole- og faglærerutdanning

Prosjektperiode
17.01.2022 - 16.05.2022

Meldeskjema

Dato	Type
20.01.2022	Standard

Kommentar
Det er vår vurdering at behandlingen vil være i samsvar med personvernlovgivningen, så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet den 20.01.2022 med vedlegg. Behandlingen kan starte.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige personopplysninger frem til 16.05.2022.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 nr. 11 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse, som kan dokumenteres, og som den registrerte kan trekke tilbake.

For alminnelige personopplysninger vil lovlig grunnlag for behandlingen være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 a.

PERSONVERNPRINSIPPER
Vi vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen:

- om lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke viderebehandles til nye uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet.

DE REGISTRERTES RETTIGHETER

Vi vurderer at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18) og dataportabilitet (art. 20).

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

FØLG DIN INSTITUSJONS RETNINGSLINJER
Vi legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

Nettskjema-diktafon er databehandler i prosjektet. Vi legger til grunn at behandlingen oppfyller kravene til bruk av databehandler, jf. art 28 og 29.

For å forsikre dere om at kravene oppfylles, må prosjektansvarlig følge interne retningslinjer/rådføre dere med behandlingsansvarlig institusjon.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilken type endringer det er nødvendig å melde:
https://www.nsd.no/personverntjenester/fylle-ut-meldeskjema-for-personopplysninger/melde-endringer-i-meldeskjema
Du må vente på svar fra Personverntjenester før endringen gjennomføres.

OPPFØLGING AV PROSJEKTET
Personverntjenester vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!