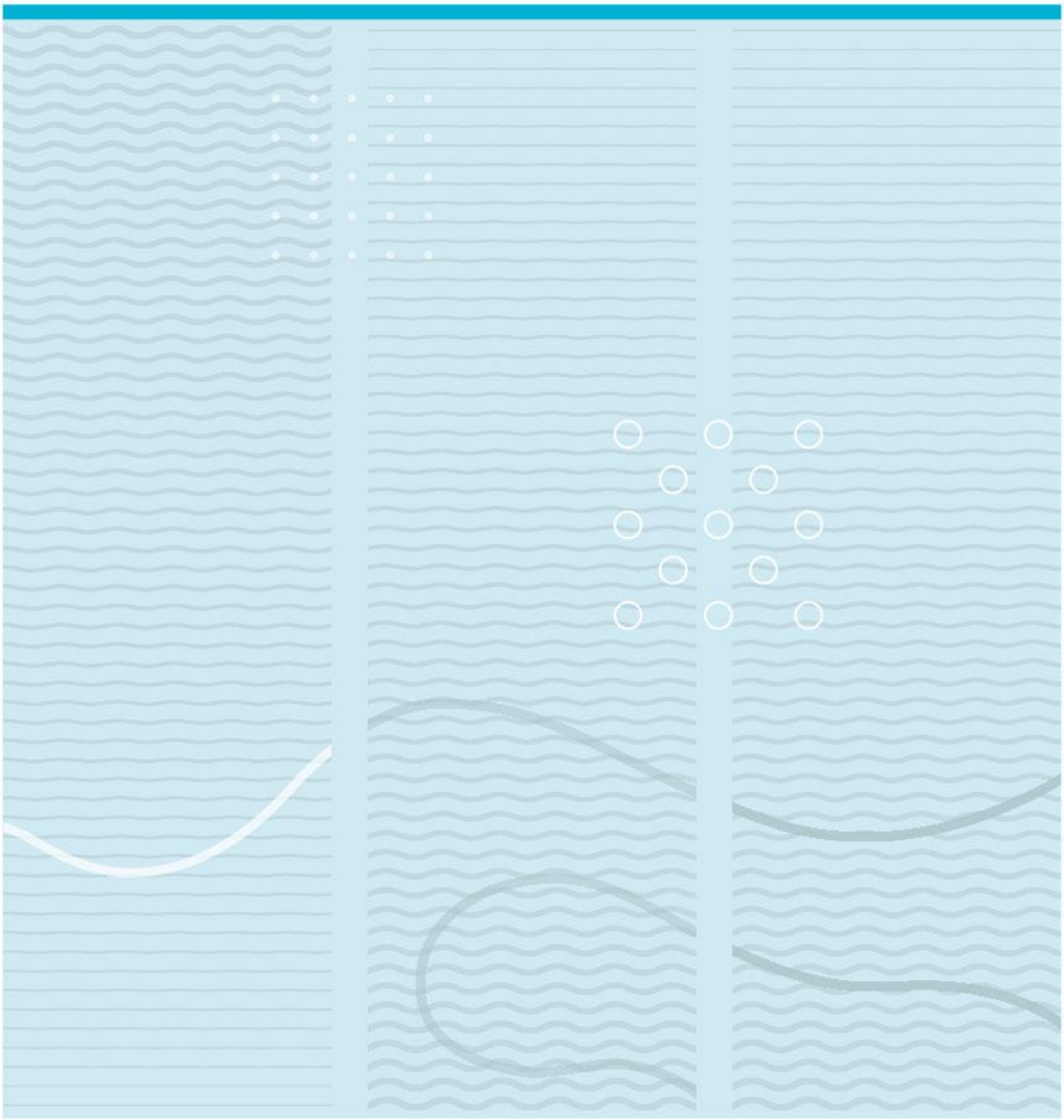
Universitetet i Sørøst-Norge
Fakultet for humaniora, idretts- og utdanningsvitenskap
–
Mastergradsavhandling i grunnskolelærerutdanning for trinn 5. – 10
Studieprogram: Profesjonspedagogikk PROGLU2
Vår 2022

Viggo Trummar Andersen

Utfordringer og erfaringer med programmering

Læreres utfordringer og erfaringer med innførelsen av programmering i matematikkfaget i grunnskolen



Universitetet i Sørøst-Norge
Fakultet for humaniora, idretts- og utdanningsvitenskap
Institutt for pedagogikk
Postboks 235
3603 Kongsberg

http://www.usn.no

Denne avhandlingen representerer 30 studiepoeng

Sammendrag

Fagfornyelsen kom høsten 2020 med innføring av trinnvise kompetansemål i matematikk fra 1. til 10. klasse. Fra og med 5. klasse og til 10. klasse har et av kompetansemålene en direkte eller indirekte tilknytning til programmering. Lærere mangler spesifikk kompetanse innen programmering, og deres utfordringer med å dekke elevens kompetansemål innen matematikk kan være utfordrende.

I denne masteroppgaven er min problemsstilling: «Læreres utfordringer og erfaringer med innføring av programmering i matematikk i forbindelse med fagfornyelsen». Jeg har sett på hvilke utfordringer lærerne opplever, og hvilke erfaringer de har erfart i løpet av snart to år med fagfornyelse (LK20).

Det er blitt brukt et kvalitativt forskningsarbeide med 5 informanter basert på et semi-strukturert forskningsintervju. Hvorav 2 av lærerne har hatt kursing eller utdannelse innen programmering, men de 3 andre ikke har hatt tidligere kompetanse innen programmering. Lærerne har hovedsakelig erfaring fra ungdomstrinnet.

Resultater fra intervjuer med lærerne avdekker varierende grad av begrepsforståelse innen programmering og algoritmisk tenkning. Lærerne trekker fra tid til egenutvikling av kompetanse, tid til samarbeid i kollegiet og systematisk kompetanseheving som sentrale utfordringer. Lærerne erfarer mangel på systematisk kursing, mangel på tiltak for utdanning innen programmering og

Konklusjon: Det bør iverksettes en helhetlig planlegging av kompetansehevingstiltak ved den enkelte skole, basert på teoretiske rammeverk for å sikre lærernes forventede kompetansenivå innen digital teknologi. Lærernes holdninger til programmering og algoritmisk tenkning bør styrkes gjennom økt fokus på hva som kreves av en «digital» lærer i grunnskolen. Det bør legges vekt på forskningsbaserte tilnærminger i undervisning i tråd med teoretiske rammeverk som PfDK og TPACK[1]. Lærerne må få tid til egenutvikling og kompetanseheving.

[1] Se kapittel 2.6 og 2.7

Abstract

The subject renewal came in the autumn of 2020 with the introduction of step-by-step competence goals in mathematics from 1st to 10th grade. From 5th grade until 10th grade, one of the competence goals has a direct or indirect connection to programming. Teachers lack specific competence in programming, and their challenges in meeting the student's competence goals in mathematics can be challenging.

In this master's thesis, my problem statement is: «Teachers' challenges and experiences with the introduction of programming in mathematics in connection with subject renewal». I have looked at the challenges the teachers experience, and what experiences they have experienced during almost two years of subject renewal (LK20).

A qualitative research work with 5 informants based on a semi-structured research interview has been used. Of which 2 of the teachers have had courses or education in programming, but the other 3 have not had previous competence in programming. The teachers mainly have experience from the lower secondary level.

Results from interviews with teachers reveal varying degrees of conceptual understanding in programming and algorithmic thinking. The teachers draw from time to self-development of competence, time for collaboration in the college and systematic competence development as key challenges. The teachers experience a lack of systematic training, a lack of measures for education in programming and

Conclusion: A comprehensive planning of competence development measures at the individual school should be implemented, based on theoretical frameworks to ensure the teachers' expected level of competence in digital technology. Teachers' attitudes towards programming and algorithmic thinking should be strengthened through increased focus on what is required of a "digital" teacher in primary school. Emphasis should be placed on research-based approaches in teaching in line with theoretical frameworks such as PfDK and TPACK. Teachers must be given time for self-development and competence development (up-skilling).

Innholdsfortegnelse

Forord

Masteroppgave er et langt lerret å bleke. Det er mye å skrive og det tar tid. Men omsider er også jeg igjennom dette. Det er et hardt arbeid, men til slutt så føler man seg fornøyd og endelig ferdig.

Det er på sin plass å takke for de informantene som har deltatt og deres velvilje til å være fleksible og tilgjengelige. De tanker og erfaringer dere har delt, har gitt meg mange gode innsikter og muligheter videre.

Oppgaven handler om programmering, men det trengs ikke inngående kjennskap til programmering. Men noe kunnskap om forskjellene på blokkprogrammering og tekstprogrammering kan være fordelaktig.

Melsomvik, 17 mai 2022

Viggo Trummar Andersen

1 Innledning

Temaet for denne mastergradsoppgaven er om programmering i matematikk. Studiet hadde oppstart høsten 2019 og fullføres nå våren 2022. Studiet har vært krevende og spennende under en pandemi og samtidig undervise i matematikk, naturfag og programmering valgfag ved en ungdomsskole.

Innføringen av LK20 høsten 2020, for 8. og 9. trinn i matematikk gav meg muligheten til å se på hvordan matematikklærere operasjonaliserer kompetansemålene i matematikk som nå inneholder programmering. Matematikkfaget er blitt gitt et særlig ansvar for programmering, sammen med naturfag, musikk og kunst og håndverk. Sanneutvalget gav i 2016 en anbefaling om at teknologi og programmering burde inn i ny læreplan som eget fag (Sanne et al., 2016, s. 76–77). Sanneutvalget peker også på stofftrengsel i skolen som et argument for et eget fag. Utvalget tar også opp forholdet mellom det teknologiske samfunnet som elevene møter, og det som elevene lærer på skolen i dag, som en utfordring og argument for et eget fag (Sanne et al., 2016, s. 77).
Balanskat og Engel (2015) har sett på 21 av Europas land arbeid med integrering av algoritmisk tenkning (AT) og programmering i læreplanene. 16 av landene hadde pr. 2015 allerede koding innpasset i læreplanene. Belgia, Nederland og Norge var ikke med da, men kom som sagt med fra 2020 i forbindelse med innføringen av LK20, fagfornyelsen (Balanskat & Engelhardt, 2015, s. 24).

Tre perspektiv blir introdusert som veien videre i (Bocconi et al., 2018, s. 3):

1. Tverrfaglig strategi.
2. Innpass i et allerede eksisterende fag.
3. Helt nytt fag.

Sverige og Finland har gått for en mellomting av 1. og 2., og Norge og Danmark, bruker «Programmering valgfag» for å videre evaluere mulighetene (Bocconi et al., 2018, s. 3–4). Forslaget videre er at det bør opprettes et eget fag for algoritmisk tenkning og programmering. Kompetanseheving av lærere bør sees i sammenheng med mulighetene for overføring av kunnskap (transfer) fra AT og programmering og til andre

Fagfornyelsen kom høsten 2020, og selv med anbefalingene fra Balanskat og Engelhardt (2015) og Bocconi et al. (2018), så forblev fagene og timeantallet likt, men innholdet ble forandret. Matematikk fikk et særlig ansvar for programmering, mens algoritmisk tenkning og dybdelæring

kommer inn som særlige overordnede tema for alle fag. Fremtidens jobber er ennå ikke «funnet» opp, og derfor er «Å lære å lære» det essensielle i de nye læreplanene (Utdanningsdirektoratet, 2019a, 2021).

Høsten 2018 benyttet jeg meg av mulighetene KFK, Kompetanse for Kvalitet, til å utdanne meg innen programmering. Kurset «Programmering for lærere» ved NTNU gikk høsten 2018 og våren 2019. Denne egne kompetanseheving gjorde at ungdomskolen jeg jobber ved kunne søke om ekstra midler via Utdanningsdirektoratet til programmeringsutstyr og skolen startet med valgfaget «Programmering valgfag». Når da høsten 2020 og fagfornyelsen kommer, så føler jeg meg ganske alene om å ha kompetanse og interesse innenfor programmering. Dette trigger min interesse for hva som hindrer andre lærere å ta i bruk mulighetene programmering gir innenfor matematikk.

Programmering og kjennskap til samhandling med datamaskiner er utvilsomt en del av fremtiden. Hvordan skal matematikklærerne inkludere programmering og algoritmisk tenkning i sin undervisning? Jeg ønsker å se på lærernes utfordringer og erfaringer med innføringen av ny læreplan i matematikk og hvordan lærerne har jobbet med programmering i forbindelse med LK20 (Fagfornyelsen).

Dette leder meg frem til problemstillingen:
«Læreres utfordringer og erfaringer med innføring av programmering i matematikk i forbindelse med fagfornyelsen»

Jeg ønsker altså å undersøke hvilke utfordringer lærere møter på når de skal under vise matematikk og bruke programmering som verktøy. Det er matematikklærere fra ungdomstrinnet som skal intervjues, og planen er å undersøke om utfordringene er i samsvar med hva aktuell forskning peker på. Seks informanter i studiet var planlagt å være med, men en informant trakk seg. Informantene jobber alle på ungdomstrinnet på 8., 9. eller 10.trinn. En av informantene har hovedsakelig jobbet på ungdomstrinnet, men har siste skoleåret undervist på videregående skole 1 år. Oversikt over kompetanse, utdannelse, kurs og fag hos informantene, kommer frem av tabell 2 i kapittel 5.1.

Jeg ønsker å få frem erfaringer lærerne har hatt så langt i fagfornyelsen.

Foruten sentrale programmeringstekniske begrep som variabler, løkker, betingelser og funksjoner, så vil også begrep som algoritmisk tenkning, dybdelæring, programmering og koding bli definert og diskutert. Programmering og koding i hverdagslig tale brukes om hverandre, men de har en forskjell som blir diskutert i kapittel 1.6.1, og figur 1 – 1 gir et godt bilde av hvordan denne sammenhengen beskrives i litteraturen.

1.1 Begrepsavklaring

1.1.1 Programmering

«Programming a computer means nothing more or less than communicating to it in a language that it and the human user can both "understand." » (S. Papert, 1980, s. 4–5).
Papert (1980), legger til grunn to sentrale ideer, hvorav den ene er behovet for en datamaskin som fungerer slik at det å kommunisere med den kan være en naturlig prosess. Papert (1980), bruker eksempelet med å lære fransk ved å bo i Frankrike, i stedet for en kunstig klasseromssituasjon hjemme i USA. Den andre ideen er hvordan programmering kan forandre hvordan vi lærer på andre måter (S. Papert, 1980, s. 6).

Balanskat og Engelhardt (2015) definerer «computer programmering» slik:

> «Computer programming is the process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems, and provide human interactivity. These instructions (source codes which are written in a programming language) are considered computer programs and help the computer to operate smoothly» (Balanskat & Engelhardt, 2015, s. 6,21).

Forsström og Kaufmann (2018) definerer «programmering» slik:

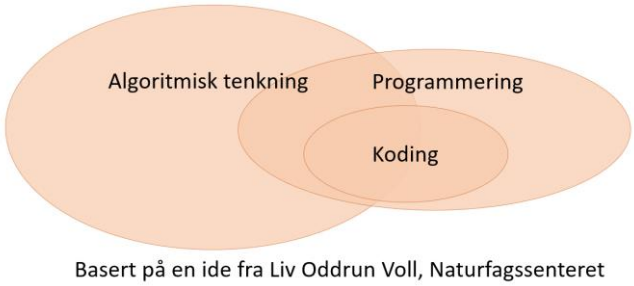
> «Programming is the process related to the development and implementation of instructions for computer programs so the computer can perform specific tasks, solve problems, and support human interactions. Therefore, programming generally requires programmers to have a knowledge of programming languages; expertise in subjects related to the development of specialized algorithms and logic; and the ability to analyze, understand, and solve problems by verifying algorithmic requirements and assessing the

> correctness and implementation (often referred to as coding) of the algorithm in a particular programming language» (Forsström & Kaufmann, 2018, s. 19).
>
> «Programmering er en ferdighet, en måte å forstå de grunnleggende mekanismene i digital teknologi og en tilegnelse av algoritmisk tekning som har likhets- punkter med matematisk logikk» (Sanne et al., 2016, s. 25).
>
> «Det å løse et problem gjennom å spesifisere en presis sekvens av kommandoer kalles algoritmisk tenkning eller algoritmisk problemløsing, og en slik presis sekvens av kommandoer kalles en algoritme. En algoritme kan sammenlignes med en matoppskrift, en strikkeoppskrift eller en regneoppskrift i matematikk. Programmering handler om å utforme algoritmer i et programmeringsspråk som en datamaskin kan tolke. Kombinasjonen algoritmisk tenkning og programmering er slagkraftig og mangfoldig» (Sanne et al., 2016, s. 18).

Disse definisjonene fra Papert (1980), Balanskat & Engelhardt (2015), Forsström og Kaufmann (2018) og Sanne (2016), bidrar til perspektiv på hva programmering er. Programmering forstås som både en prosess for å utvikle et digitalt produkt, men også selve inntastingen. Hele prosessen kan inneholde design, innhold, implementering, feilsøking og ferdig produkt, og omtales ofte som programmering, mens selve inntastingen og implementeringen kalles koding (Forsström & Kaufmann, 2018, s. 19; Stenseth et al., 2019, s. 7).

Algoritmisk tenkning (forklart senere), som er basert på hvordan en dataprogrammerer tenker, har nyttige verktøy og begreper som setter prosessene i gang, og er nært knyttet tankegangen i programmering. I Europa benyttes begrepene «Computational thinking», «Algorithmic thinking», «Programming» og «Coding» ulikt, og i Norge brukes koding og programmering om hverandre i likhet med Polen, England og Nederland (Balanskat & Engelhardt, 2015, s. 27). Det er er viktig å være klar over at begrepenes definisjoner og hverdagslige betydninger ikke nødvendigvis er de samme.

Basert på ovenstående definisjoner og definisjonen av koding som kommer under her kan Liv Oddrun Volls ved Naturfagssenteret.no gi mening til forståelsen av sammenhengen mellom AT, Programmering og koding. Modellen er referert til forfatteren etter avtale via e-post. Opphavet er ifølge forfatteren: «figuren er hentet fra en presentasjon jeg holdt en gang».



Figur 1-1 Algoritmisk tenkning, programmering og koding, en modell av Liv Oddrunn Voll, Naturfagssenteret

1.1.1.1 Koding

Koding i kontekst av programmering brukes om den fysiske operasjonen det er å fysisk skrive et program, altså selve inntastingen av programmeringsfunksjoner. Denne kodingen er en av delene i en programmeringsprosess, der også design, testing, feilsøking og dokumentasjon er andre viktige deler (Forsström & Kaufmann, 2018, s. 19; Lodi & Martini, 2021, s. 886; Stenseth et al., 2019, s. 7). Koding brukes i dag også mer populært og synonymt med programmering. Koding er også brukt som en mer avslappet og lekfylt holdning til programmering (Prottsman (2015) i (Lodi & Martini, 2021, s. 886)).

1.1.1.2 Transfer

Lodi og Martini (2021) definerer «Transfer» eller overføring, som overføring av kunnskap eller kompetanse, fra en kontekst til en annen. (Lodi & Martini, 2021, s. 886–887). Det er vanskelig å dokumentere transfer, og forskning peker både mot ingen og liten grad av transfer (Guzdial, 2016, s. 40).

1.1.1.3 Computational thinking, Computer Science, Computational Sciences

Lodi og Martini (2021, s. 887 – 888) påpeker at det er viktig å skille mellom disse. På norsk har vi algoritmisk tenkning, informatikk og beregningsvitenskap. Jeg vil påpeke at disse ligger nærere opp til eget virkeområde enn de engelske versjonene, og ikke vil være like lette å blande.

1.1.2 Algoritmer

«Algoritme er i matematikk og databehandling en fullstendig og nøyaktig beskrivelse av fremgangsmåten for løsning av en beregningsoppgave eller annen oppgave» (Hovde & Grønmo, 2020). Vi kan se på det som nøyaktige instruksjoner for hvordan datamaskinen skal utføre en oppgave.

1.1.3 Algoritmisk tenkning

Det er to hovedretninger som beskrives, den ene kommer fra Jeanette Wing (2006) og den andre fra Seymour S. Papert (1980). Begge er relevante i diskursen rundt programmeringsdidaktikk, når de blir brukt og forstått riktig (Lodi & Martini, 2021, s. 883). Wing (2006) hovedtanke er vekten på «Computer Science», informatikk, hvor det er selve utdannelsen av programmerere som er i fokus. Mens hos Papert (1980) er det den konstruktivistiske tanken som ligger til grunn der elevene deltar i en sosial setting. Men likevel skiller Papert seg fra Piaget ved at Papert ønsker et håndfast produkt, mens Piagets omskriving av skjema er en kognitiv hendelse (Resnick, 1998, s. 45). Mer om dette i kapittel 2.4 for definisjon og forklaring. Se også 2.2.1 for mer om Piagets og Paperts forskjellige perspektiv på konstruktivismen.

1.1.4 Computational thinking

«Computational thinking» er det engelske ordet for algoritmisk tenkning (S. Papert, 1980, s. 182). Algoritmisk tenkning er litt annerledes enn det engelske, men er basert på Wing (2006). En grunnleggende egenskap for alle og ikke bare forskere innenfor data og informatikk. Mer om dette i kapittel 2.4 for definisjon og forklaring.

1.1.5 Programmeringsspråk

Programmering kan sees på som en prosess der vi kommuniserer med en datamaskin. Vi forteller datamaskinen hva den skal gjøre. Vi trenger et språk som vi kan kommunisere med, og fortelle hva vi ønsker den skal gjøre. Barn benytter seg av lignende konsepter når de i lek instruerer hverandre til å gå fremover, til siden, opp ned osv. Liknende med utarbeidelse av et skattekart. Og suksessen er avhengig av hvor nøyaktige instruksjonene er. Et programmeringsspråk for en datamaskin må være nøyaktig slik at det oppfører seg likt hver gang (Mannila & Nordén, 2020, s. 33–34; S. Papert, 1980, s. 5–6). Programmeringsspråk er dermed en mediator på samme måte som et naturlig språk,

og et kulturelt redskap utviklet for å kommunisere med datamaskinen (Sentance et al., 2019, s. 145).

1.1.6 Dybdelæring

Se Kapittel 2.5 for definisjon og utredning.

1.2 Oppbygging av oppgaven

1.2.1 Kapittel 2

Teori. Ser på læringssyn i programmering, hvor Seymour Papert (1980) har et konstruktivistisk syn på undervisning av programmering, mens nyere tanker av Sentance et al. (2017) legger større vekt på sosiokulturelt læringssyn som støtter Lev Vygotskys tanker. Programmering, og sentrale deler innen programmering blir forklart. Blokk-kode, tekstprogrammering og programmeringsdidaktikk. Begrepene algoritmisk tenking og dybdelæring blir gjort rede for. Til slutt blir to aktuelle rammeverk presentert til å se på digitale forutsetninger og krav til den digitale læreren.

1.2.2 Kapittel 3 og 4

Her presenteres henholdsvis problemsstilling og metode. Metoden er semistrukturert intervju med 5 informanter. 4 underviser på ungdomstrinnet og 1 underviser VGS 1 skoleåret 2021/2022, men har undervist 8, 9, 10 trinn før det.

1.2.3 Kapittel 5

Inneholder analyse – resultater – drøfting. Hvert funn blir analysert og presentert. Deretter blir funn drøftet opp mot aktuell teori.

1.2.4 Kapittel 6 og 7

Diskusjon, konklusjon og forslag til videre forskning.

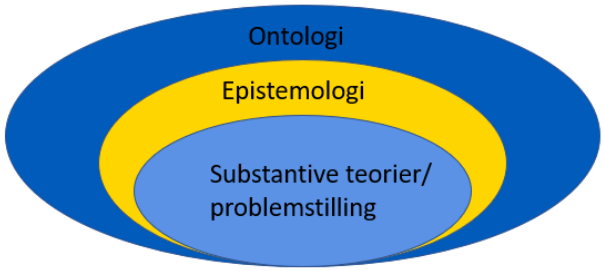
2 Teori

2.1 Læringssyn

I dette delkapitlet vil jeg presentere de aktuelle læringssyn som er aktuelle i syn på programmering i grunnskolen med tanke på programmering. Læringssynet som legges til grunn innenfor programmering er på den ene siden konstruktivistisk, og på den andre siden sosiokulturelt (S. Papert, 1980; Wenger, 2019). I delene videre skal jeg ta for meg hvordan disse to synene spiller inn mot aspekter innen programmering.

2.2 Ontologi og epistemologi[2]

Ontologi er læren om hvordan verden faktisk er. Hvordan verden er bygget opp og hvilke teorier og hvilke antakelser og tanker vi har om verden (Nyeng, 2012, s. 37). Helt overordnede filosofiske teorier slik som positivismen, konstruktivismen, pragmatismen og post-positivismen, som alle har vidt forskjellige oppfattelser av verden, er læren som det værende, det virkelige; ontologi. Hvordan vi kan tilegne oss kunnskap om disse områdene er da læren(logi) om kunnskap(episteme). Altså epistemologi (Maxwell, 2013a, s. 42; Nyeng, 2012, s. 37). Figuren til høyre viser sammenhengen mellom ontologiske, epistemologiske og substantive teorier. Der de substantive teoriene besvarer problemstillingen. Postholm (2018) argumenterer derav at en god teori beskrives ved at den har logisk forankring i epistemologiske teorier og ontologiske teorier (Postholm & Jacobsen, 2018a, s. 21). Ordet epistemisk er et adjektiv av episteme(gresk) eller avledet av epistemologi, og betyr dermed hvordan vi ser på eller oppfatter kunnskap. Det norske «epistemiske oppfatninger» brukes lite sammenlignet med det engelske «epistemic beliefs». «Epistemic beliefs are individuals' beliefs about knowledge and knowing.» (Berding et al., 2017, s. 103). Det essensielle



Figur 2-1: Oversikt over teorier på ulike nivå (Postholm, 2018, s. 21)

[2] Dette delkapitlet (2.2) s. 17 og ca 200 ord, er basert på forskningsskisse til oppgaven tidligere innlevert i studiet

her er da, slik jeg tolker det, det individuelle syn man har på kunnskap (Andersen, Viggo Trummar, 2020).

2.2.1 Konstruktivismen

Jean Piaget har preget konstruktivismen. Man sier at Piaget «[…] jobbet med hvordan tilegner seg de erfaringer de besitter, og hvordan kognitiv tilpasning skjer» (Säljø, 2016, s. 59). Säljø (2016, s. 60) forklarer videre hvor viktig denne tilpasninger er balansen mellom ytre og indre tanker og forestillinger av verden.

«Kunnskap oppstår gjennom menneskers aktiviteter, og den konstrueres aktivt av et individ, når individet tar stilling til og bearbeider omgivelsene» Piaget (1970) i (Säljö, 2016, s. 60). Piaget er i motsetning til Vygotsky, (Vygotski et al., 1978), en representant for at det er individet som er i fokus på læring. Piaget mener at assimilasjon og akkomodasjon er de sentrale tilegningsmetodene for kunnskap. Hvorpå assimilasjon innebærer at individet tar ny kunnskap og ta den inn i etablerte «strukturer og skjemaer» (S. Papert, 1980, s. 7; Säljö, 2016, s. 60). Säljø (2016, s. 60) forklarer videre at akkomodasjon innebærer at disse skjema omorganiserer når ny erfaring og kunnskaper skal tilpasses inn i eksisterende kunnskap.

Seymour S. Papert, som er medforsker på programmeringsspråket LOGO som ble introdusert i 1966, 1967 («Logo», 2022; Solomon et al., 2020, s. 4), samarbeidet med Piaget på «Piaget 's Centre for Genetic Epistemology» i Geneve fra 1959 til 1964 (S. Papert, 1980, s. 19). Paperts fasinasjon over Piagets syn på barns læring gav mening; at barn skaper sin egen kunnskaps struktur. Men Papert ser også at kunnskap ikke blir bygget ut av intet, men i samhandling med materialer og kulturen rundt (S. Papert, 1980, s. 19). Det er under oppholdet i Geneve (ca 1964) at Papert opplever bruken av datamaskiner kan være nyttefulle. Han ser på muligheten for hvordan barn kan nytte seg av dette (S. A. Papert, 2020, s. 23–24). Piaget og forskningssenteret hans kan altså sees som en viktig inspirator for Paperts del i utviklingen av LOGO, som altså er forløperen til Scratch, blockly, makecode og alle blokkbaserte, visualiserte programmeringsspråk. Paperts konstruksjonisme skiller seg også fra Piagets konstruktivisme ved at, elever har bedre læringsutbytte ved at de lager noe konkret eller har et håndfast produkt. Papert argumenterer for

at læring oppstår når folk lager noe som er personlig meningsfullt for dem selv (Resnick, 1998, s. 45).

2.2.2 Sosiokulturelle syn på læring i skolen

Et sosiokulturelt syn på læring innebærer at individet utvikler kunnskap i interaksjon med mennesker ved å delta i en kultur. Sosiokulturell kommer ved at mennesket deltar aktivt i en kultur. Lev Vygotsky (1896 – 1934) var med sitt syn på læring en av Jean Piagets (1896 – 1980) store kritikere, da Vygotsky revolusjonerte sin tids syn på læring og utvikling ved legge vekt på kultur, historie, artefakter, interaksjon og kulturelle aktiviteter som grunnleggende sosiale prosesser (Stray et al., 2014, s. 123). Verdt å nevne er også Jerome Bruner (1915 – 2016), som var bidragsyter til å bringe russiske Lev Vygotskys tanker til den vestlige verden. Bruner bygger tankene sine på Vygotskys arbeid. Bruners stillasbygging «scaffolding», der barnet får hjelp av en voksen til å løse mer utfordrende oppgaver, er inspirert av Vygotsky (Stray et al., 2014, s. 122). Bruners stillasbygging tar oss da over til Vygotskys nærmeste utviklingssone eller «Zone of Proximal Development».

2.2.3 Sosiokulturell teori

Ved å delta i en sosial praksis, får man tilgang til kunnskap. Denne kunnskapen kan man senere nyttiggjøre i egen sosial praksis. «Kunnskap og erfaringer eksisterer først mellom mennesker i kommunikasjon, og blir deretter til redskaper som individene kan benytte i fremtidige sosiale praksiser» (Säljö, 2016, s. 161). Kunnskap vokser altså frem ved at vi som individ deltar i et samfunn, en kultur, en praksis, sammen med medmennesker, der vi bruker språket som en kunnskapsformidler. Ved å delta i slike praksiser, tar vi del i kunnskapen. Vi fører kunnskapen videre til neste generasjon (Säljö, 2016, s. 161).

2.2.3.1 Vygotsky

Lev S. Vygotsky (1896 – 1934) levde samtidig som både Jean Piaget og Jerome Bruner, men det sies at Vygotsky viste om Piagets verk, men ikke motsatt mens Vygotsky var i live. Bruner er den som oversatte Vygotsky, og gjorde materialet tilgjengelig i 1962 på engelsk etter å ha først hørt om dette på en konferanse[3] i Montreal i 1954. Vygotsky var forbudt i Sovjetunionen, men Vygotsky var godt

[3] På en uformell mottakelse mot slutten av uka (J. Bruner, 1985, s. 22)

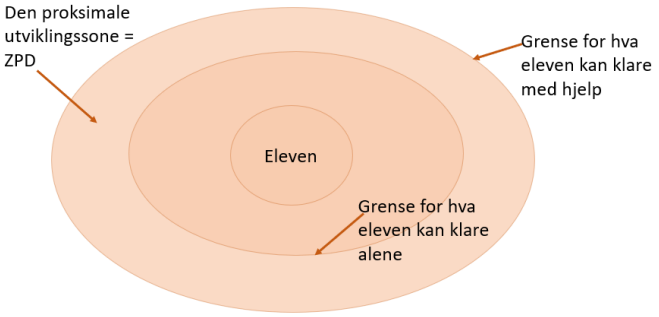
sirkulert i miljøet. Vygotsky var altså ukjent for den vestlige verden i over 20 år (J. Bruner, 1985, s. 22–23).

2.2.3.2 Den proksimale utviklingssone (ZPD)

Vygotskys egne ord:

> «[…] the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers". (J. Bruner, 1985, s. 24; Vygotski et al., 1978, s. 86)

Man kan altså forstå den proksimale utviklingssone, eller den nærmeste utviklingssone, som den avstanden mellom hva en barn/elev/student kan få til alene, og hva barn/elev/student kan få til med hjelp fra en voksen hjelp, eller veiledning av mer kompetente læringspartnere (Sentance et al., 2019, s. 144; Säljö, 2016, s. 118–119). ZPD er også brukt synonymt med stillasbygging, hvor man støtter eleven i problemløsningsarbeid ved at læreren kontrollerer elementene rundt et problem som er utenfor en elevs rekkevidde eller kunnskap (J. S. Bruner et al., 1976, s. 90). Når man ser dette i lys av programmering så kan flere av stillasene være for eksempel blokk-kode i stedet for tekstprogrammering og ferdig kode i stedet for å programmere selv (Sentance et al., 2019, s. 144).



Figur 2-1 Den proksimale utviklingssone

2.2.3.3 Sosiokulturelt syn på programmering

Sentance et al. (2019, s. 137) påpeker at hovedsakelig består forskningen innen programmeringsutdanning innen det kognitive synet på læring, og at ved å se på læring og undervisning fra et sosiokulturelt perspektiv vil vi kunne se på hvordan medierende faktorer, språk, samarbeid og artefakter vil være i henhold til Vygotskys syn.

Undervisning i programmering fra et sosiokulturelt perspektiv kan ifølge Sentance (2019, s. 146), formuleres i tre prinsipp:

1. Mediering gjennom språk – at elevene diskuterer med hverandre om ulike strategier for å løse et problem, gjennom for eksempel parprogrammering eller samarbeid rundt deler av programmeringskode.

2. Bruk av allerede programmerte eksempler som ikke ennå er forstått av elevene. Stillasbygging benyttes for å sikre at de er innenfor ZPD. Her kan man øke kompleksiteten etter hvert som selvsikkerhet og internalisering av kunnskap øker.
3. Læreren (den voksne med kunnskap) – er i dette tilfellet stillaset som bidrar til at eleven er innenfor ZPD. Eleven kan også settes sammen med en annen elev som er mer kompetent. Det er viktig at gapet ikke er for stort mellom elevene.

(Sentance et al., 2019, s. 146–147)

På bakgrunn av disse elementene så utviklet Sentance et al. (2019) PRIMM-modellen i 2017.

2.3 Programmering i fagfornyelsen

«Senter for IKT i utdanningen», som i dag er under som Utdanningsdirektoratet(udir), har kommet frem til tre tilnærminger til innføringa av 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

(Selvik, 2016, s. 7)

Disse tre tilnærmingene er i stor grad fulgt ved at programmering er i et eget valgfag: «Valgfag programmering» (Utdanningsdirektoratet, 2022b), og at fagene naturfag, matematikk, musikk og kunst og håndverk har fått en spesiell status med egne kompetanse mål som omhandler programmering(Utdanningsdirektoratet, 2020c).

Matematikk har fra 1. klasse til og med 10.klasse et kompetansemål som enten omhandler algoritmer og mønstre (1 – 4 klasse), eller begrep som går direkte på programmeringstekniske begrep: slik som variabler, løkker, funksjoner, vilkår og utforske data og tabeller i datasett (Utdanningsdirektoratet, 2020c).

Naturfag har for eksempel kompetansemålet «bruke programmering til å utforske naturfaglige fenomener», som er ganske vagt formulert, men åpner for utforsking av programmering (Utdanningsdirektoratet, 2022a).

Ludvigsenutvalget som i 2015 leverte sin rapport «Fremtidens skole», trekker frem matematikk, naturfag og teknologi som et av sentrale fagområder i norsk skole. Utvalget trekker også frem disipliner som: «medisin, naturforvaltning, ingeniørfag og teknologi», som sentrale områder for samfunnets fremtidige behov(NOU 2015:8, 2015, s. 24). Dette støttes også av Pellegrino og Hilton i (Pellegrino & Hilton, 2012, s. 31), hvis rapport omhandler dybdelæring og «21st century skills».

LK20 innfører høsten 2020 kjerneelementene som nytt tilskudd til læreplanen. Kjerneelementene elementer som overordnede fokusområder innenfor et fag, for matematikk 1. – 10. klasse ser dette slik ut (Utdanningsdirektoratet, 2020a):

1. Utforsking og problemløsing
2. Modellering og anvendelse
3. Resonnering og argumentasjon
4. Representasjon og kommunikasjon
5. Abstraksjon og generalisering
6. Matematiske kunnskapsområde

Det første elementet her er utforsking og problemløsing, og under dette elementet kommer problemløsning og algoritmisk tenkning. Det nevnes bruk av digitale verktøy kan løse delproblem (som er et av begrepene under algoritmisk tenkning). Algebra nevnes under punkt 6, og algebra er sentralt i formelregning og manipulasjon av tall inne programmering. Selv om programmering ikke nevnes eksplisitt, så er det underforstått og implisitt at programmering er et verktøy som kan brukes for å drive undervisning i henhold til kjerneelementene i matematikkfaget (Utdanningsdirektoratet, 2020a).

2.3.1 Historie

«I think there is a world market for about five computers» skal Thomas J. Watson, president for IBM, ha angivelig ha sagt I 1943, på spørsmål om behovet for datamaskiner i fremtiden. Han tok grundig feil, men likevel mener mange nå at vi allerede kanskje er på vei mot et behov for kun 5 datamaskiner, om vi tenker i termer der store firma som Google, Amazon, IBM og andre allerede har en så stor del av vår hverdag med tanke på den økende grad av skylagring og tjenester som kjører på ekstern server kontra lokale tjenester(Bonasio, 2018).

Historien om programmering starter allerede i 1883 med Ada Lovelace og Charles Babbage som utviklet en tidlig primitiv mekanisk datamaskin som ble brukt til å regne ut Bernoulli nummer(Kellner, 2022). Videre så ble kjente programmeringsspråk slik som Pascal, Smalltalk, C, SQL, MatLab, C++, Perl utviklet på 70- og 80-tallet, mens Python, Visual basic, R, Java, PHP og Javascript ble utviklet på 90-tallet(HP.com, 2018).

Dagens språk som i stor grad benyttes i grunnskolen og på VGS er forskjellige versjoner av blokkode (makecode, blockly, scratch …) og tekstbasert (Python). Disse blir ytterligere forklart i delkapittel 2.3.2 og 2.3.4.

> «Programming has been described by many authors as the new Latin of the school syllabus, a kind of mental whetstone for developing minds. As such, it was assumed that students would develop their general problem-solving skills through learning programming. However, reports from teachers of programming and results from some empirical studies now suggest that the teaching of programming has created significant difficulties for high-school and university students, and has failed to catalyze the development of higher order thinking skills» (Sleeman, 1986, s. 840).

Man kan tolke Sleeman (1986) dithen at i 1986 har man erfart problemer med innfallsvinkelen datidens programmeringsdidaktikk har. Med den nye latinen mener Sleeman at dette er det nye (og kanskje vanskelige) i læreplanene, og at dette er slipestenen som skal gjøre hjernen skarp (min tanke og oversettelse).

2.3.2 Blokk-kode

Blokk-kode eller blokkode, er et programmeringsprinsipp som bruker ferdige kode-funksjoner som kan kobles sammen slik som puslebiter passer sammen. Hver av disse blokkene er tilgjengelig med ulike funksjoner slik som variabler, løkker, betingelser, funksjoner (Aspefalten et al., 2022; Google Developers, 2018; teknologiundervisning.dk, 2022). Det finnes spesielle versjoner slik som Microsoft sin makecode.microbit.org som er for å programmere Micro:Bit, nettstedet code.org, MITs scratch.mit.edu som blir omtalt i underkapitlene som følger etter her. Blokkode brukes som en introduksjon til programmering, og en av de viktigste fordelene er at man unngår skrivefeil. I undervisning er blokkode slik som for eksempel Scratch spesielt designet for barn i alderen 8 – 16, men selvsagt kan alle aldre ha glede av Scratch(Fraser, 2022; MIT Scratch, 2022).

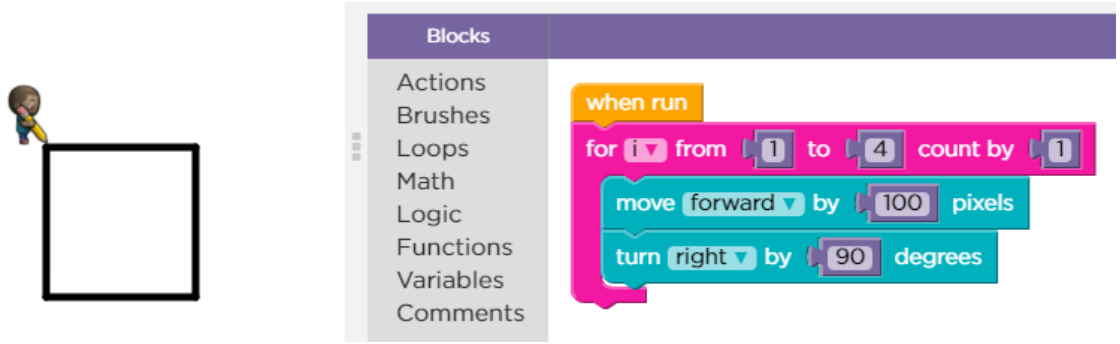
Forskningsmiljøet LUDO ved usn.no definerer blokkprogrammering:

> «Lage et program ved å sette sammen blokker som inneholder ulike kommandoer. De bygges omtrent som legobrikker og er en visuell representasjon av et bakenforliggende kodespråk»
> (Aspefalten et al., 2022)

2.3.2.1 Blockly

Blockly er et visuelt programmeringsverktøy som flere andre nettsteder og ressurser bruker. Likevel er det viktig å poengtere at blockly ikke er et eget programmeringsspråk, men et verktøy som oversetter blokkoden til et annet språk, for eksempel Python eller Javascript som kjøres på lokal maskin (Fraser, 2022).

Kodebasen fra Googles blockly er grunnlaget for både Micro:bit makecode, code.org og Scratch, og flere, men de blir ikke omtalt her.



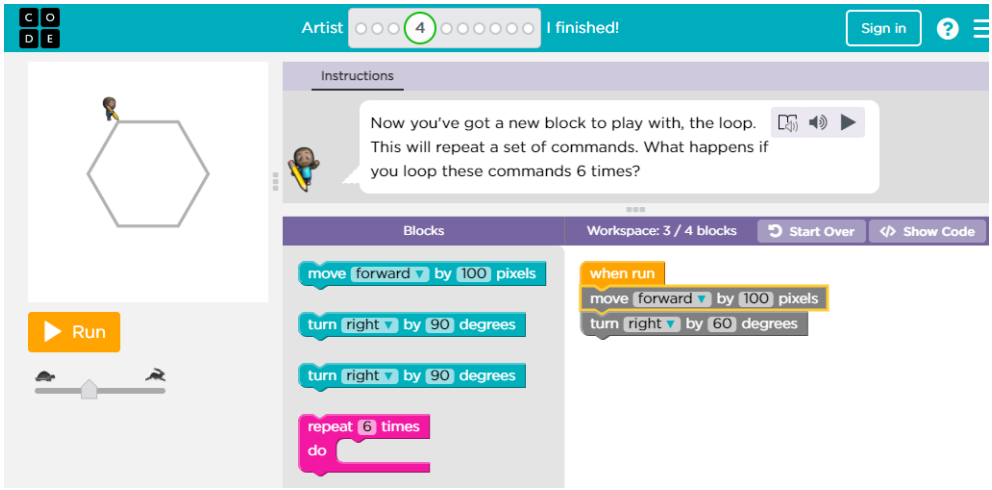
Figur 2-2 Eksempel på kode for å tegne et kvadrat i blockly

I figur 2 – 2, vises et eksempel på hvordan blokkode brukes for å få tegneren til å tegne et kvadrat. De forskjellige blokkene ligger tilgjengelige og passer i hverandre som puslebiter. En av fordelene med å bruke blokkode sammenlignet med tekstprogrammering er at skrivefeil ikke er mulig.

2.3.2.2 Code.org

Code.org er et populært nettsted som bruker teknologien fra Googles blockly som basis for sine blokker. Code.org har læringsstier som bruker både omvendt undervisning ved bruk av videoer og steg-for-steg oppgaver hvor vanskeligheten øker. Figuren under viser hvordan en slik oppgave kan se ut. Her oppfordres det til å bruke en ny løkke-funksjon(loop), til å effektivisere tidligere kode.

Løsningen er å putte inn de to grå blokkene inn i den rosa løkke-blokken, akkurat som lego-klosser. Code.org benytter seg av forklaringer, hint, videoer og halvferdige eksempler (code.org, 2022). Code.org, kan man til en viss grad er i tråd med PRIMM sitt rammeverk for undervisning i programmering, da ofte mye av koden er klar i eksemplene (Sentance et al., 2019, s. 148).



Figur 2-3 Eksempel fra code.org - sekskant

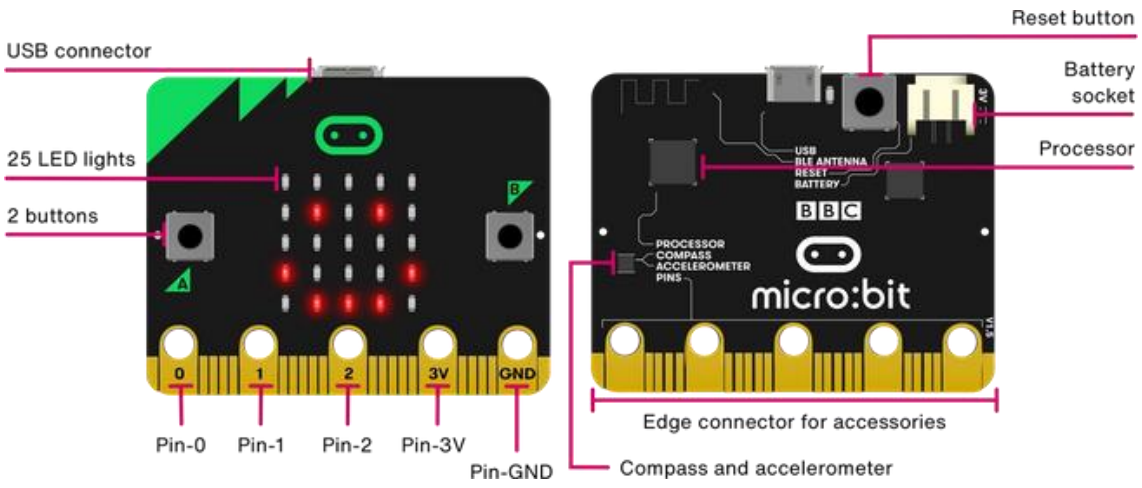
2.3.2.3 Scratch

Scratch er et blokkode-programmerings-miljø med et enkelt visuelt brukergrensesnitt som gjør at man kan lage spill, animasjoner og programmer. Scratch er i likhet med code.org er læringsmiljø med læringsstier som både motiverer og engasjerer(MIT Scratch, 2022). Scratch følger de samme prinsipper som blockly.

2.3.2.4 Micro:Bit – Makecode

Micro:Bit (aka Microbit) er en mini datamaskin med alle deler som en datamaskin har, bare at den er på størrelse med et halvt kredittkort. Den inneholder sensorer, knapper, muligheter for trådløs kommunikasjon og kan kjøre eksternt på en batteripakke (Microbit.org, 2022; Wikipedia.org, 2022).

Makecode er blokkode versjonen som kan benyttes for å programmere denne lille databrikken. Ekstra funksjoner og kommandoer kan lastes inn for å bruke denne til å styre andre roboter, led-pærer eller for eksempel servoer.



Figur 2-4 Micro:Bit - hentet fra microbit.org - https://creativecommons.org/licenses/by-sa/4.0/#

2.3.3 Tekstprogrammering

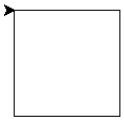
Tekstprogrammering er programmering der det forskjellige kode-elementene kun finnes i tekstlig form i motsetning til blokkode som er visuelt og grafisk. Tekstprogrammering er linje for linje programmeringskode som utføres steg for steg. Denne steg for steg operasjonaliseringen kan brukes til å definere objekter som senere kan brukes. Der hvor blokkoden allerede har ferdige definerte blokker, så må man i tekstprogrammering definere og opprette sine egne «blokker» ved hjelp av funksjoner, løkker, metoder, klasser, løkker, betingelser og variabler. Et av disse tekstspråkene er Python som ofte velges av utdanningsinstitusjoner og skoler pga enkel syntaks, enkel oppbygning og stort internasjonalt miljø (Eidsvig, 2020; Grandell et al., 2006, s. 73; Haraldsrud et al., 2020a, s. 13–14; Python.org, 2022).

2.3.3.1 Python

Python er et av det mest populære tekstbaserte programmeringsspråkene som benyttes i dag. Både grunnskole og videregående skole benytter seg av Python. PYPL som står for PopularitY of Programming Language, fører statistikk over det mest populære programmeringsspråkene, og for april 2022, så er Python på topp med ca 28%, mens de neste på listen(Java, Javascript og C/C++), har henholdsvis 18%, 9% og 7% (PYPL.github.io, 2022).

```
from turtle import *

for i in range(1,5,1):
    forward(100)
    right(90)
```



Figur 2-5 Eksempel på python-kode

Figur 3 – 4 viser i dette eksempelet det samme som figur 3 – 1 gjør. Det er biblioteket turtle som brukes, og koden til venstre i figuren, plotter resultatet til høyre. Turtle kommer fra Paperts LOGO (S. Papert, 1980; S. A. Papert, 2020, s. 11,63-64). Syntaks og oppbygning er mer rigid og nøyaktig enn det blokkoden viser. En viktig forskjell her er hvordan løkka har tilsynelatende andre parametere enn det tilsvarende i blokkoden. Her betyr løkka: «fra og med tallet 1 til tallet 5»- Det betyr at tallene 1, 2, 3, 4 er med, mens 5 er ikke med. I tillegg i steg av 1.

2.3.3.2 LOGO

LOGO er et program utviklet av Seymour A. Papert m.fl. allerede tilbake i 1966. «Den grunnleggende ideen med LOGO var at barn skulle ha et kraftig programmeringsmiljø» (Solomon et al., 2020, s. 1). Tanken med LOGO var et programmeringsmiljø basert på læring og utforskning innenfor matematikk. I LOGO kunne man styre en skilpadde som hadde form som en trekant. LOGO hadde/har mange av de samme kommandoene som Python Turtle har i dag. Noen eksempler er forward 100 og right 90, som henholdsvis flytter skilpadden 100 piksler frem og deretter snur 90 grader. Ved å kombinere disse vil man, som i Python Turtle, kunne designe mer komplekse figurer. Man kunne lage objekter som man kan gjenbruke senere, ved å kalle på en funksjon som inneholder predefinert kode. Solomon et al. (2020) har en utfyllende historie rundt LOGO.

2.3.3.3 Javascript

Er et tekstbasert programmeringsspråk som brukes mye til dynamiske websider. Har lit tyngre struktur og mindre lettlest enn Python.

2.3.3.4 C

Er et kraftig programmeringsspråk som i motsetning til Python og Javascript er nærmere datamaskinens eget interne språk. C sin kode må kompileres, dvs oversettes til datamaskinens interne språk, i motsetning til Python som tolkes linje for linje.

2.3.4 Konkreter

Det finnes mange konkreter man kan bruke til programmering. Et av de mest populære er Micro:Bit som er nevnt i 2.3.2.4. Micro:Bit kan igjen programmeres til å styre roboter slik som BitBotXL og flere elektronikksett. Ved bruk av disse robotene så kan elever flytte programmeringen vekk fra bare skjerm, til å vise noe fysisk som kan skje. Ved for eksempel å få en robot til å føle et mønster eller en linje tapet på gulvet. Slike problemløsningsoppgaver åpner en annen dimensjon innen programmering.

2.3.5 Programmeringsdidaktikk

Når man går i gang med å lete etter gode kilder til programmeringsdidaktikk, så finner man lite konkrete lærebøker som omhandler dette tema i seg selv, men min erfaring er at enkelte «programmeringsbøker» har et kapittel som omhandler dette. Det finnes mange kilder på Computational Thinking (CT) og Algoritmisk Tenkning (AT) i artikkelform når man søker i forskningsdatabaser som ERIC, oria.no og Google Scholar. Forskere som kan nevnes er J.M. Wing, som er den som tok opp igjen tanken om CT i 2006, etter at Papert introduserte det i (S. Papert, 1980, s. 182).

2.3.5.1 Noen aktuelle programmeringsbøker

Programmeringsbøker brukt i forbindelse med denne oppgaven er følgende: (Mannila & Nordén, 2020), (Mannila, 2017), (Bueie, 2019), (Haraldsrud et al., 2020b) og (Gaddis, 2019). Av disse 5 bøkene, som på ingen måte er et representativt utvalg, så er det kun de to førstnevnte som vier plass til AT («datalogisk tänkande») og didaktikk. De andre tre er instrumentelle, og viser hvordan man skal lære seg programmering med tanke på kommandoer, struktur og helhet.

2.3.5.2 5E og PRIMM

En litteraturstudie er en avhandling i seg selv, og vil ikke ytterligere bli behandlet i denne oppgaven. Det viser likevel at det er vanskelig å finne ressurser på norsk innenfor programmeringsdidaktikk, noe som kan ha sammenheng med at Sverige har innført programmering og AT i de svenske læreplanene allerede fra 2017, og Finland nevnes å ha sine planer implementert høsten 2016 (Sanne et al., 2016, s. 71–72).

Aktuell forskning innen temaet blir tatt opp Benton (2017) som tar for seg et rammeverk for programmering i matematikk. Dette rammeverket blir kalt 5E, og disse 5E står for «Explore, Envisage, Explain, Exchange og BridgE» (Benton et al., 2017, s. 122–123). 5E handler om at elevene skal først utforske og undersøke ulike ideer eller tanker rundt en oppgave eller tema. «Envisage» (se for seg), betyr at elevene skal forutse eller tenke på utfallet før de koder ferdig eller tester programmet. «Explain» (forklare), betyr at elevene og lærerne skal diskutere i klasseromsdialog eller blant læringspartnere. «Exchange» betyr at elevene kan dele kode og bygge på andres kode. «BridgE» handler om mulighetene for transfer eller overføring, og bevisstheten rundt CT og AT (Benton et al., 2017, s. 122–123).

PRIMM – «Predict, Run, Investigate, Modify, Make», er et annet rammeverk av Sentance et al. (2019, s. 148). PRIMM rammeverket har mange av elementene fra 5E. PRIMM handler om å forutse hva et program kommer til å gjøre. Premisset i PRIMM er at elever/studenter får et ferdig program som, enten skal diskuteres og prøves, eller utvides. Programmet skal så kjøres, og deretter skal utfallet diskuteres i klasserommet. Investigate handler om å utforske koden og forklare hva aktuell kode faktisk gjør/utfører. Modify og Make er tema/emner som får elevene/studentene til å modifisere eksisterende kode til å gjøre noe annet, og Make omhandler det å lage noe nytt ut av eksisterende kode. For eksempel etter Modify så kommer Make (Sentance et al., 2019, s. 148–150).

2.3.5.3 Forskningsmiljø

Forskningsmiljøene i Norge som undersøker programmeringsdidaktikk er i startgropen. Noen total oversikt er vanskelig å fremme, men det utlyses PhD innen programmeringsdidaktikk ved flere universiteter. En forsker ved NMBU skriver, i disse dager (2022), både doktorgrad og artikkel om programmeringsdidaktikk. Forskeren presenterer i en video for fagforeningen Tekna sine tanker om programmering og programmeringsdidaktikk. Noen spennende punkter som forskeren trekker frem er synet på dybdelæring og programmering, og der mener forskeren at programmering egner seg best for å øke dybdelæring i et fag, og ikke som innlæring av nytt stoff. Det pekes også til hvorvidt og når man skal introdusere elvene for ferdig kode, eller om elevenes skal programmere selv fra blanke ark (Munthe, 2021, s. tidskode 16:21). Et annet spennende aspekt som Munthe (2021) trekker frem, er at par-programmering er bedre enn å programmere alene. Par-programmering vil bli diskutert i neste avsnitt.

2.3.5.4 Parprogrammering

Parprogrammering er en teknikk kjent fra utvikling av programvare. Tanken er at den ene av to som samarbeider er den som skriver kode. Den andre er den som observerer og holder oversikt over retningen og planen (Gillis, 2021; Guzdial, 2016, s. 87). Forskningen som finnes på området kommer hovedsakelig fra CS (Computer Science), og forskning som finnes på CE (Computing Education), er ofte fra begynneropplæring på de første årene på ingeniør eller universitetsnivå. Forskning viser at parprogrammering har en viss effekt på CS og CE nivå. Studenter som er satt i parprogrammering viste at det var 90.8% sannsynlighet for at de tok eksamen, i motsetning til studenter som lærte å programmere alene: 80.4% (Hanks et al., 2011, s. 137). Karakterene på eksamen i studiet til Hanks et al. (2012), viste at de var like (75.2% vs. 74.4%). Hanks et al. (2012, s. 137 – 138) viser til flere studier med n > 300, som viser lignende resultater. Det vises også til at studenter som har parprogrammert gjør det bedre i kurs senere hvor det arbeides alene. I en undersøkelse med spørsmålet: «''I learned more in this class because I pair programmed», med en skala fra 1 – 7, hvor 1 er helt uenig og 7 er veldig enig, så scorer studentene 5.07, som indikerer at parprogrammering er til hjelp for deres forståelse for programmering Hanks (2006) i (Hanks et al., 2011, s. 147–148). En annen undersøkelse gjort om jenters forhold til programmering i et IT-kurs på nivå ca 14års alder, så kommer det frem at jentene merker økt motivasjon, fokus og oppmuntring. Andre funn er økt sosialisering, økt forståelse og at man har hjelp i sin læringspartner (Liebenberg et al., 2012, s. 232).

Effekten av programmering med hensyn på et profesjonelt samarbeid innen programvareutvikling sier at parprogrammering kan være nyttefullt i enkelte situasjoner, som for eksempel der avansert programvare skal utvikles for å opprettholde kode-sikkerhet og riktig programmering. Parprogrammering kan være tidsbesparende på enklere oppgaver (Hannay et al., 2009, s. 1120).

Williams et al. (2000) omhandler parprogrammering i et universitets eksperiment, viser til høy grad av tilfredshet med parprogrammering. Over 90% av de 41 informantene svarte at de likte parprogrammering bedre enn solo-programmering. Informantene viste også til at parprogrammeringen gav dem mer selvtillit til at eget arbeid (Williams et al., 2000, s. 24). Problemer som forskerne viser til, er at noen ganger kan det være tilfeller der partnerne ikke passer sammen. Forskerne viser i studiet at dette kan komme av personligheter med høyt ego eller

personer med sterke meninger. Personer med lite ego vil også kunne være av den typen som bidrar lite (Williams et al., 2000, s. 24).

2.4 Algoritmisk tenkning

Seymour A. Papert tilskrives å være den første som bruker termen: «Computational thinking» i sin bok: «Mindstorms – children, computers and powerful ideas» (S. Papert, 1980, s. 182; S. A. Papert, 2020, s. 205). Lodi og Martini i (Lodi & Martini, 2021), tar for seg opphav, diskusjon og hvilke perspektiv man kan ha på algoritmisk tenkning. To hovedsyn som Lodi og Martini (2021) viser til diskuteres.

> «Algoritmisk tenkning er en problemløsningsmetode. Algoritmisk tenkning innebærer å tilnærme seg problemer på en systematisk måte, både når vi formulerer hva det er vi ønsker å løse og når vi foreslår mulige løsninger. Litt forenklet kan vi si at det er 'å tenke som en informatiker' når vi skal løse problemer eller oppgaver.»
> (Utdanningsdirektoratet, 2019a)

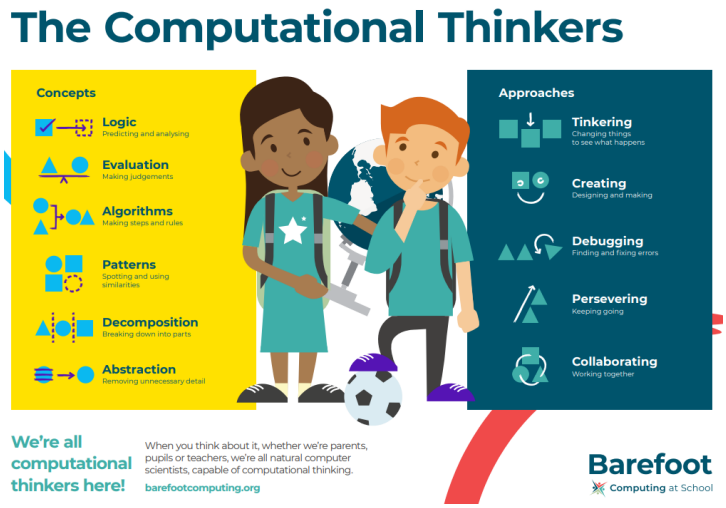
Utdanningsdirektoratets definisjon av algoritmisk tenkning nevner den systematiske måten å tilnærme seg et problem, samt dette å tenke som en informatiker. Det engelske «computational thinking (CT)» er opphavet og blir tilskrevet Jeanette M. Wing, som den som igjen satte begrepet på agendaen med bidrag til ACM.org i mars 2006 (J. M. Wing, 2006, s. 33). Wing (2006) definerer begrepet ved å bruke tankegangen fra utvikling av dataprogrammer og ved å stille spørsmål om hva datamaskiner kan gjøre bedre enn mennesker og motsatt(J. M. Wing, 2006, s. 33). CT er en grunnleggende egenskap som tar tankegang fra utvikling av dataprogram og benytter prosessen til å takle problemløsning ikke bare innenfor datateknologi og programmering, men også generell problemløsning i andre fag og områder i samfunnet. CT er en grunnleggende egenskap for alle og ikke bare forskere innen datateknologi(J. Wing, 2017; J. M. Wing, 2006, s. 33).

> « Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.”

(J. Wing, 2017, s. 8)

Sitatet over er Wing(2017) sin redefinering av begrepet i 2017. Når man ser dette i lys av (Utdanningsdirektoratet, 2019a) sin definisjon så ser vi likhetene i formuleringene.

Når vi videre ser på posteren fra Barfootcomputing.org, som er et ressurssenter for og av lærere for å støtte innføringen av nytt innhold i den britiske motsvarigheten til den norske fagfornyelsen(Barefoot Computing, 2022a), så ser vi at den norske adapsjonen av computational thinking, er basert på Barefoot Computing sin versjon. Noe det også blir vist til på (Utdanningsdirektoratet, 2019a).



Figur 2-7 Barefoot Computing - Computational thinking

Figur 2-6 Utdanningsdirektoratets Algoritmisk tenkning

Når man deretter bryter noen av disse nøkkelbegrep og arbeidsmåter i en tabell på følgende vis, så ser man at disse korrelerer med Jeanette M. Wing sine begrep (J. Wing, 2017; J. M. Wing, 2006).

Tabell 1 Utdanningsdirektoratet (2019a) nøkkelbegrep relatert til J.M.Wing (J. M. Wing, 2006, s. 33–34)

Udir - nøkkelbegrep	Wing	Udir - arbeidsmåter	Wing
Algoritmer – steg-for-steg	Retracing steps		
Dekomposisjon – abstraksjon	Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system.	Feilsøke – oppdage og rette feil	Computational thinking is thinking in terms of prevention, protection, and recovery from worst-case scenarios through redundancy, damage containment, and error correction

Algoritmisk tenkning og overførbarhet diskuteres i (Guzdial, 2016, s. 39–40, 49–50), og Guzdial viser til at elever og studenter i liten grad viser overførbarhet fra programmering til det virkelige liv. Guzdial (2016) viser til at den, så langt, eneste måten å få til overførbarhet fra programmering til det virkelige liv er om det undervises for akkurat det henseende. Et eksempel som blir brukt er hvorvidt prinsippene fra CT er overførbare til om man skal legge fliser på et gulv, hvorpå Guzdial argumenterer at ingen studie(til hans kjennskap), har siden Wing (2006) har vist transfer effekter (Benton et al., 2017, s. 118; Guzdial, 2016, s. 40). Dette står i kontrast til det fokus som finnes i (Grover & Pea, 2013, s. 41–42; Utdanningsdirektoratet, 2019a; J. Wing, 2017, s. 9; J. M. Wing, 2006, s. 41–42).

M. Guzdial og P.J Denning nevnes i (Lodi & Martini, 2021, s. 886-888 mfl) som en av kritikerene til blant annet hvorvidt transfer av algoritmisk tenkning finner sted utenfor datateknologi eller programmering. Det vises til i Lodi & Martini også at det er vist transfereffekter i (Weintrop & Wilensky, 2018, s. 18–19), der forskerne designet et eget programmeringsmiljø, der de så på transfer-effekter om finner sted, med/på overgangen mellom blokkode og tekstbasert programmering. Funnene viser at det er form for transfer som skjer, men at den ikke er entydig og at det er basis for videre arbeid (Weintrop & Wilensky, 2018, s. 90).

I artikkelen «Computational Thinking, Between Papert and Wing» av (Lodi & Martini, 2021), blir opphavet til algoritmisk tenkning tatt opp, og et av poengene Lodi og Martini er at definisjonen til algoritmisk tenkning eller «computational thinking» blir brukt forskjellig fra ulike perspektiv, for eksempel media, utdanningsinstitusjoner eller politikere (Lodi & Martini, 2021, s. 883). Et annet viktig poeng som trekkes frem er J. M. Wings sitt syn på AT/CT er mer rettet mot «computer science» eller datateknologi/informatikk, mens Paperts syn er mer konstruktivistisk, og at kun en mer sosial og mer motivert tilknytning til tverrfaglige emner, vil gjøre programmering til et nyttig verktøy for å forstå og undervise for og med algoritmisk tenkning (Lodi & Martini, 2021, s. 883). Mer om Seymour Papert og konstruktivismen i et kapittel 2.2.1. Seymour Papert stiller spørsmålet: «Why should children not also use computational ideas to improve their understanding of their own thinking, learning and playing? Well, because this is not in the national standards! I predict that it will be» (S. Papert, 2005, s. 367).

2.5 Dybdelæring

Definisjon av dybdelæring hos Utdanningsdirektoratet (Udir):

> «Vi definerer dybdelæring som det å gradvis utvikle kunnskap og varig forståelse av begreper, metoder og sammenhenger i fag og mellom fagområder. Det innebærer at vi reflekterer over egen læring og bruker det vi har lært på ulike måter i kjente og ukjente situasjoner, alene eller sammen med andre». (Kunnskapsdepartementet, 2019)

Ludvigsenutvalget lanserer dybdelæring i (NOU 2015:8, 2015), og deres definisjon basert utredningen som er gjort:

> «Dybdelæring dreier seg om elevenes gradvise utvikling av forståelse av begreper, begrepssystemer, metoder og sammenhenger innenfor et fagområde. Det handler også om å forstå temaer og problemstillinger som går på tvers av fag- eller kunnskapsområder. Dybdelæring innebærer at elevene bruker sin evne til å analysere, løse problemer og reflektere over egen læring til å konstruere en varig forståelse.» (NOU 2015:8, 2015, s. 14)

Definisjonen som Udir benytter er i stor grad basert på Ludvigsensutvalgets. Når man sammenligner setning for setning, og emne for emne så kan man si at Udir sin versjon er kanskje tydeligere i språk enn Ludvigsensutvalgets.

Disse definisjonene fra Ludvigsenutvalget og Utdanningsdirektoratet kan vi bryte ned i flere deler:

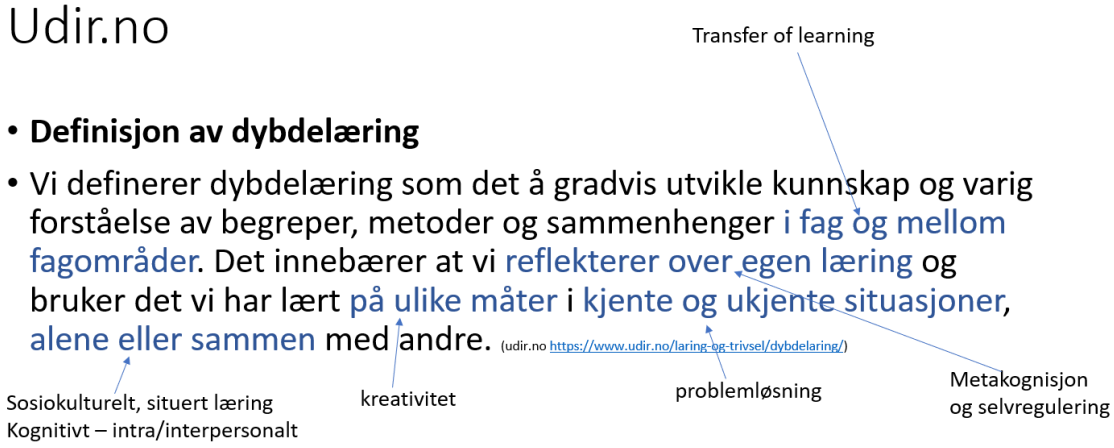
1. Utvikle kunnskap
2. Forståelse av begreper
3. Metoder
4. Sammenhenger i fag og mellom fagområder
5. reflekterer over egen læring
6. På ulike måter
7. Kjente og ukjente situasjoner
8. Alene eller sammen med andre

Vi ser på disse definisjonene, som jeg i henhold til algoritmisk tenkning har brutt opp i mindre deler. Enkelte av delene i definisjonen til dybdelæring hos Undervisningsdirektoratet er relaterbare enkelt av delene i «den algoritmiske tenkeren». Jeg finner at for eksempel punkt 8 kan relateres til «Samarbeide – dele og jobbe sammen», som er et av punktene i figur 3 – 7. Videre kan punkt 5 relateres til punkt 6, og punkt 3 (metoder) til punkt 2. Dette kan vise at dybdelæring er i tråd med algoritmisk tenkning, og begge begrepsområder er viktige for programmering og matematikk. Videre kan vi relatere punkt 4 (Sammenhenger i fag og mellom fagområder) til diskusjonen om transfer, og metoder mot algoritmer i punkt 2 på plakaten «Den algoritmiske tenkeren» (Utdanningsdirektoratet, 2019a).

For at dybdelærings skal skje legges det til grunn at en forandring må til. Bolstad (2020, s. 11) legger til grunn at dybdelæring er noe som skjer i hodet, det er noe som skjer med kroppen og det er å utvikle kompetanse. Dybdelæring forklares med at det må skjer en forandring, at noe vi ikke har hatt kunnskap om er nå blitt til kjent kunnskap. Dybdelæring innebærer også at kunnskapen vi har blir forandret basert på ny viten og kunnskap vi tar til oss (Bolstad, 2020, s. 11). Dette i samsvar med måten Piagets skjemaer som blir omskrevet (Säljö, 2016, s. 60), likevel er Vygotskys ZPD, «Den proksimale utviklingssone», mer fremtredende i definisjonen til Udir, der delen: «alene eller sammen med andre», på en måte bygger bro mellom det konstruktivistiske- og det sosiokulturelle

læringssynet. Bolstad (2020, s. 11) peker på fire forutsetninger for at dybdelæring skal finne sted: «skapes i et fellesskap og gjennom språk, krever tid, krever mening, krever sammenheng og overblikk».

Figur 2-8 er en dekomposisjon i tråd med AT for å prøve å forklare selve dybden i definisjonen på dybdelæring. Hvordan de forskjellige delene kommuniserer transfer, kreativitet, samarbeid, bruk av metoder, refleksjon, dypere forståelse og problemløsning (Kunnskapsdepartementet, 2019).



Figur 2-8 Definisjonen av dybdelæring analysert med tanke på begreper (Tatt fra egen muntlig eksamen i Pedagogikk H2019)

2.6 PfDk – Profesjonsfaglig digital kompetanse

> «Rammeverk for lærerens profesjonsfaglige digitale kompetanse er et retningsgivende dokument som politikkutviklere, instituttledere, lærerutdannere, **lærere**, **lærerstudenter** og andre kan bruke som referanse i arbeidet med å **øke kvaliteten** i lærerutdanning og **systematisk etter- og videreutdanning av lærere**. Med rammeverket ønsker vi å etablere et felles begrepsapparat og en felles referanseramme for hva lærerens profesjonsfaglige kompetanse innebærer» (Kelentrić et al., 2017, s. 5)[4].

Når man legger «Senter for IKT i utdanningen», nå under Utdanningsdirektoratet, sine tanker om PfDk-rammeverk, ser man at dette er et dokument som peke ut retningen i videreutdanningen for

[4] Min utheving

både lærerstudenter og lærere blant andre. Grunnlaget for rammeverket er basert på nasjonale styringsdokumenter, nasjonale forskrifter og krav til lærerutdanningen, aktuelle læreplaner m.m. (Kelentrić et al., 2017, s. 6). Basert på dette er det utviklet en figur 2 – 9, med 7 hovedområder, som viser hvilke kompetanser en helhetlig lærer skal besitte.

Figur 2-9 PfDk - hentet fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/rammeverk-larerens-profesjonsfaglige-digitale-komp/innledning/#om-rammeverket

Lærerens profesjonsfaglige digitale kompetanse sett i kontekst med programmering, og denne oppgavens problemsstilling, så er det flere kompetanseområder som er aktuelle:

- Skolen i samfunnet
- Fagdidaktikk
- Samhandling og kommunikasjon

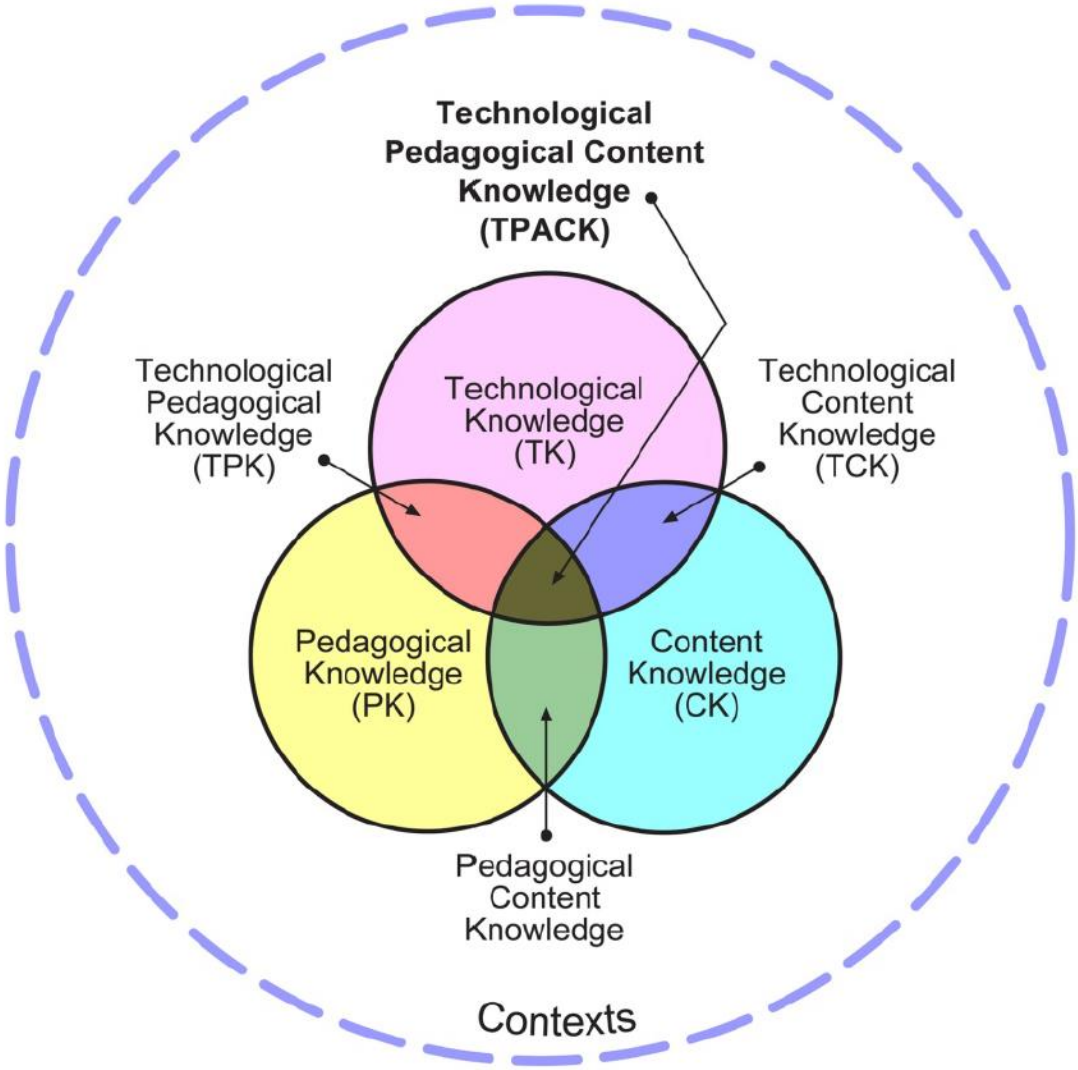
Mange av lærerens kunnskaper, ferdigheter og generell kompetanse griper over i hverandre innenfor denne PfDk-modellen, og vil tas opp i videre diskusjon.

2.7 Technological Pedagogical and Content Knowledge

«Technological Pedagogical and Content Knowledge» - TPACK, er et rammeverk for å kunne innlemme teknologi i undervisning og lærerens totale kompetanse. Rammeverket er videreutviklet

fra Lee Schulman (1986 og 1987) sitt rammeverk PCK, referert i (Mishra et al., 2009, s. 60–62). Tre sider ved sider ved TPACK er sentrale:

1. Pedagogisk kunnskap
2. Innholds kunnskap
3. Teknologisk kunnskap



Figur 2-10 TPACK - Mishra et al. (2009, s. 63)

Disse tre områdene griper over i hverandre og tolker man figuren som et Venn-diagram så vil snittet vise den samlede kunnskap en lærer må besitte for å kunne drive effektiv læring med teknologi. Dette krever at læreren har kunnskap om alle tre feltene og har forståelse for hvordan disse områdene spiller sammen (Mishra et al., 2009, s. 63–64).

Mishra et. al (2009) har identifisert utfordringer som lærere erfarer ved innføring av ny teknologi i undervisningen:

1. Teknologiens virkemåter er ukjent eller skjult for lærerne
2. Teknologien forandrer seg raskt
3. Teknologien har ikke et klart entydig virkeområde – altså kan brukes på flere forskjellige måter.
4. Hvordan bruke rett teknologi til rett område – for eksempel e-post er bra til asynkron kommunikasjon, men ikke synkron, slik som for eksempel sms og direktemeldinger (IM, nettprat, osv.) er.
5. Lærere har tatt utdannelse i en tid hvor teknologien ikke var tilgjengelig eller like aktuell.
6. Lærere erfarer at å tilegne seg kunnskap som krever at det settes av tid i en travel hverdag, er utfordrende
   (Mishra et al., 2009, s. 61–62).

2.8 Etterutdanning for lærere

Ved innføring av fagfornyelsen høsten 2020 kommer det nye kompetansemål i matematikk på hvert trinn som omhandler programmering og/eller algoritmisk tenkning. Dette fører til at lærere må tilegne seg kunnskap for å dekke kompetansemålene. I det følgende vises det til ulike ressurser fra statlig og privat/frivillig hold.

2.8.1 Kompetanse for kvalitet

Kompetanse for kvalitet er en ressurs forvaltet av Udir, som har til hensikt å styrke læreres kompetanseheving. Denne kompetansehevingen er for de fleste fag i grunnskolen, og et eksempel som er i kontekst lærere er et 15 studiepoengs fag som tilbys ved flere universiteter og høyskoler høst og vår (7.5p + 7.5p). Videre tilbyr Udir tilskudd til skoler og kommuner for innkjøp av teknologiske hjelpemidler til programmering (Utdanningsdirektoratet, 2019b).

2.8.2 Utdanningsdirektoratet (Udir) – MOOC

Udir har lansert etterutdanning som er tilgjengelig for å på individuelt nivå kan ta nettbaserte videreutviklingskurs kalt kompetansepakker. Disse kalles også for MOOC – «Massive Open Online Courses» - nettkurs, åpne for alle. Kompetansepakkene består av innhold for flere fag, men har

innhold for PfDK – Profesjonsfaglig digital kompetanse, og handler om hvordan lærere kan ta i bruk teknologi for å støtte elevenes læring. Det finnes kurs innenfor matematikk, valgfag programmering, digitale ferdigheter og PfDK. Både innenfor PfDK og programmering valgfag finnes moduler som omhandler programmering, som er aktuelle for lærere i grunnskolen. De fleste av modulene er fra 2019 og eldre (Utdanningsdirektoratet, 2022c).

2.8.3 Dekom – Dekomp

Desentralisert ordning for kompetanseutvikling – Dekom/Dekomp, er en ordning for kompetanseheving i grunnskolen. Følgende er USN sin definisjon:

> «Universitetet i Sør-Øst Norge (USN) arbeider i partnerskap med regionens grunn- og videregående skoler om å øke kollektiv kompetanse i skolene. Målgrupper for kompetansetiltakene i tilskuddsordningen er lærere, ledere, andre ansatte og andre i organisasjonen rundt som bidrar til kvalitet i grunnopplæringen. I grunnskolene samarbeider vi i nettverk med skoleeiere og i vgs samskaper vi på skolenivå». (USN, 2020b)

Denne ordningen gjør at skoleeiere kan få bidrag fra lokalt universitet gjennom ordningen for å gjennomføre kompetanseheving innenfor aktuelle emner. Ordningen støtter «kartlegging av behov, planlegging og gjennomføring av tiltak» (DEKOM, 2022). Tiltak kan være kurs innenfor programmering med blokk-kode, python tekstprogrammering, programmering innen musikkfaget, omvendt undervisning m.m (USN, 2020a).

2.8.4 Vitensenter og Lær Kidsa Koding

Vitensenter.no er en ressurs som består av 12 sentere i Norge. De legger følgende presentasjon av seg selv til grunn:

> «Et vitensenter er et populærvitenskapelig opplevelses- og læringssenter innen matematikk, naturvitenskap og teknologi hvor de besøkende lærer ved å eksperimentere selv. I et vitensenter kan barn og voksne utforske fenomener knyttet til natur, miljø, helse og teknologi gjennom egen aktivitet og i samarbeid med andre» (Vitensenter, 2021).

Vitensenteret har både besøk lokalt av klasser, holder kurs for lærere og er et opplevelsessenter for hele familien på fritiden.

Lær kidsa koder (LKK), har som formål:

> «Vi vil hjelpe de unge til å ikke bare bli brukere, men også skapere med teknologien som verktøy. I tillegg til å øke den allmenne forståelsen av informasjonsteknologi, ønsker vi å bidra til rekrutteringen til IT-yrkene og realfagene. En viktig del av vår virksomhet er å arbeide for at alle unge i skolealder får mulighet til å lære programmering og bli kjent med informatikk som fag» (LKK, 2021).

LKK er lokale initiativ for å la unge erfare og oppleve programmering. LKK er ofte lokalisert i byens bibliotek. Det finnes pr mai 2022, 203 kodeklubber i Norge. Kodeklubben er gratis og temaer kan være Scratch og Micro:Bit. LKK er primært for barn, men lærere kan hente ressurser fra LKK sine nettsider som de kan bruke i undervisning og til inspirasjon.

2.8.5 Sosiale medier

Sosiale medier kan være nyttig for læreres inspirasjon og motivasjon. Det finnes flere aktive grupper på Facebook der engasjementet til tider er stort. Aktuelle grupper (ikke uttømmende utvalg) kan være: «Programmering i skolen og programmeringsdidaktikk, 1.5k», «Lær Kidsa Koding lærernettverk 4.6k», «Programmering i fagfornyelsen 2.3k», «micro:bit Norge, 2.3k» og «Digitale lærere(tidl. Korona-dugnad), 60.2k». Disse er noen aktuelle grupper på Facebook hvor programmering diskuteres. Tallene, for eksempel 2.3k, refererer til antall medlemmer i skrivende stund.

3 Problemstilling

Problemstilling for dette forskningsarbeidet er:
«Læreres utfordringer og erfaringer med innføring av programmering i matematikk i forbindelse med fagfornyelsen».

Det kan ytterligere konkretiseres ved at jeg ønsker å se på hvilke utfordringer lærerne opplever i møte med kompetansemålene i matematikk som omhandler programmering. Kompetansekravene i matematikk på ungdomstrinnet inneholder et konkret kompetansemål som enten omhandler ordet programmering, eller tilsvarende eller ordet algoritme. Tre aktuelle kompetansemål fra ungdomstrinnet er for 8, 9, 10-trinn:

- «utforske korleis algoritmar kan skapast, testast og forbetrast ved hjelp av programmering» (8.trinn)
- «simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering» (9.trinn)
- «utforske matematiske eigenskapar og samanhengar ved å bruke programmering» (10. trinn)
  (Utdanningsdirektoratet, 2020c)

Disse kompetansemålene er på et høyere nivå, enn de kompetansemålene som ligger fra 5. til 7. trinn som omhandler mer grunnleggende ferdigheter innen programmering, slik som: Variabler, vilkår, løkker og funksjoner (Utdanningsdirektoratet, 2020b). Kompetansemålene skaper en ramme for kunnskapen som kreves av både lærere og elever.

Ovenfra-og-ned-prosjekter fra for eksempel udir.no, statlig nivå eller kommunalt nivå viser seg i stor grad å feile eller ikke ha tilsiktet effekt eller virkning dersom det ikke er forankret i de aktuelle mottakerne, som i dette tilfellet er lærer-kollegiet. En stor del av slike prosjekter oppnår ikke ønsket effekt (Bjørnsrud, 2015, s. 70; Fullan, 1994; Roald, 2012, s. 19,20,211). Og i dette premisset kan det ligge utfordringer med innføring av programmering i fagfornyelsen. Jeg ønsker da å se på lærernes utfordringer og erfaringer, og hvordan lærerne gjennomfører programmering i matematikk på ungdomstrinnet.

4 Metode

Valg av metode i dette prosjektet har vært gjenstand for diskusjon og vurdering. Mitt første ønske var å gjøre en større kvantitativ undersøkelse for å se på effekter av algoritmisk tenkning og bruk av programmering i matematikkundervisningen på dette. Men oppgavens begrensninger i form av antall studiepoeng, og det faktum at prosjektet vokser seg stort gjorde at prosjektet ble snevret inn. Med (Blikstad-Balas, 2017, s. 511,516) friskt i minne fra forelesninger om «death by data», som handler om at for mye materiale kan gjøre at man mister både oversikt og motivasjon, velger jeg å snevre inn metoden fra et ambisiøst kvantitativt prosjekt, til en metode basert på intervju av lærer på ungdomstrinnet for å besvare min problemsstilling.

Jeg vil begynne med å gjøre rede for hvilke valg som ligger til grunn for oppgavens metodiske tilnærming. Deretter vil jeg grunngi de valg jeg har tatt med hensyn til innsamling av datamateriale og hvilke valg jeg har gjort med tanke på arbeid med intervjuguide, dokumentasjon og transkribering av intervjuet.

4.1 Tilnærming og valg av metode

I denne oppgaven ønsker jeg å identifisere de utfordringer og erfaringer matematikklærere har i forbindelse med innføring av programmering i fagfornyelsen fra høsten 2020 og til dags dato. Jeg ønsker å se hvordan lærere tenker spesielt om hvordan de nye læreplanene i matematikk innlemmer kompetansemål som omhandler algoritmisk tenkning og programmering. Jeg vil søke å bruke informanter som både har programmeringsfaglig bakgrunn med kurs og/eller utdannelse, men også lærere som ikke har denne kompetansen. Jeg ønsker å se lærernes perspektiv og høre deres tanker på algoritmisk tenkning og programmering, sett i lys av faktorer som kursing og utdannelse, både i egen regi og skolens regi. Valget av metode baserer seg på ønsket om å komme i dybden på den enkelte informants syn på programmering og algoritmisk tenkning. Ved å intervjue informantene og bruke et semi-strukturert intervju vil jeg komme i dybden på lærernes tanker, refleksjoner og synspunkt. Denne type intervju vil gi meg et godt grunnlag for videre diskusjon og analyse.

Et intervju kan deles inn i 7 faser (Kvale & Brinkmann, 2015, s. 137): «Tematisering, Planlegging, Intervjuing, Transkribering, Analysering, Verifisering, Rapportering». Denne oppgaven forholder seg i store trekk til den denne oppbyggingen.

4.2 Innsamling

4.2.1 Intervju

Et kvalitativt forskningsintervju søker å forstå verden fra sett fra informantens side. Vi ønsker å finne ut noe vi ikke har kunnskap om fra før. Forskeren ønsker å finne frem til erfaringer og opplevelser sett fra informantens perspektiv og dennes verden (Kvale & Brinkmann, 2015, s. 20). Å intervjue kan alle gjøre med en lydopptaker påpeker Kvale & Brinkmann (2015, s. 20, 34), men å gjøre et godt håndverk krever tid og energi. Formålet med et forskningsintervju er å produsere kunnskap. Dette skjer mellom forskeren og informanten «i en profesjonell samtale» (Kvale & Brinkmann, 2015, s. 22). Kvale & Brinkmann (2015, s. 20) legger vekt på at det er et asymmetrisk forhold mellom den som blir intervjuet og forskeren, og at det er forskeren som styrer intervjuet og dermed styrer samtalen.

Et kvalitativt forskningsintervju setter krav til forskeren for å besvare problemsstillingen forskeren har stilt. Det er viktig å skille mellom problemstilling og forskningsspørsmål som spør om det forskeren ønsker å finne ut av. Mens de spørsmålene man spør om for å avdekke informantenes erfaringer, meninger og holdninger, er intervjuspørsmålene (Maxwell, 2013b, s. 101). Maxwell (2013, s.s) påpeker også at det burde utføres en pilot-test der man prøver ut spørsmål for å se om de fungerer som forventet eller om en revisjon og presisering av intervju-spørsmålene trengs.

I min oppgave ønsker jeg å benytte meg av et semistrukturert intervju. Et semistrukturert intervju har mer generelle tema enn et strukturert intervju, som har mer spesifiserte spørsmål. I det semistrukturerte så kan forskeren i større grad endre på spørsmål, stille oppfølgingsspørsmål eller endre rekkefølgen. Forskeren kan også i større grad stille tilleggsspørsmål basert på hvordan informanten svarer (Bryman, 2016, s. 201; Kvale & Brinkmann, 2015, s. 46).

Læreres hverdag er travel og opptatt. Min fremgangsmåte for et intervju med informant er dermed å benytte meg av samarbeidsteknologi slik som Zoom eller Microsoft Teams for å innhente data. Bryman (2016, s. 492) trekker frem fordeler og ulemper fra sin litteratur gjennomgang og påpeker at det er lovende resultater ved bruk av Skape og Face-Time (2016). Fordelene som nevnes er fleksibilitet i tidspunkt og ikke minst sted, da møtested blir irrelevant. I tillegg er kostnad for reise og tiden det totalt tar. Bryman nevner også sikkerhet som en faktor for både forsker og informant. Ulemper som kommer frem, er teknologiske utfordringer som nettilgang og WiFi-problemer. Kvaliteten på selve opptaket kan også by på problemer siden opptaket må transkriberes i etterkant (Bryman, 2016, s. 492). Mange av ulempene som Bryman (2016) nevner, slik som dårlig kvalitet og teknologiske utfordringer, ble ikke rapportert i en senere undersøkelse av Archibald et. al (2019, s. 5). Bryman (2016) nevner ikke sikkerhet eksplisitt, mens Archibald et. al (2019, s. 6) ser på Zooms personvernsinnstillinger (Zoom.us, 2022). Det er en del aspekter med verktøy slik som Zoom, men også Microsoft Teams, og et av dem er GDPR – «General Data Protection Regulation» (no.wikipedia.org, 2021) og datalagring. Det settes høye krav til databehandling og personvern, og til deltakeres muligheter til innsyn og sletting. Det er også viktig å ha fokus på forskjellene mellom USA og EU/EØS sine regler om hvem som har tilgang til lagrede data. Det er vesentlig forskjell på regler mellom USA og EU (Archibald et al., 2019, s. 6). Studien av Archibald et. al (2019) er pre-covid19 pandemi, og verden har opplevd en eksploderende bruk av Teams og Zoom i utdanningsinstitusjoner, og behovet for mer undersøkelse på dette feltet er nødvendig (UsNews, 2020).

I mitt semistrukturerte intervju har jeg delt opp i tre deler, hvor den første delen er såkalte standard spørsmål om alder, bakgrunn, kjønn og utdannelse m.m. Den andre delen stiller spørsmål om programmering, matematikk og algoritmisk tenkning. Den tredje delen tar for seg programmering og begrep. Del to er mer semistrukturert enn del tre. Selv om spørsmålene er til en viss grad avgjorte, så er jeg åpen for å kunne utdype og stille oppfølgingsspørsmål der det er nyttefullt. Et semistrukturert intervju vil ha forslag til spørsmål som er sortert under emner. Rekkefølgen er det som det legges vekt på, men det er opp til intervjuerens erfaring og teft, i hvilken grad han holder seg til intervjuguiden (Bryman, 2016, s. 468; Kvale & Brinkmann, 2015, s. 162).

4.2.2 Utvalg av informanter

Antall informanter varierer ut ifra hvilke kilder man legger til grunn. Forskning som Bryman (2016, s. 416), legger til grunn tilsier mellom 20 og 30 informanter, og alt under 60 er for lite til å «gi overbevisende konklusjoner». Men Mason (2010) i Bryman (2016) viser til i en undersøkelse at blant 560 doktorgradsavhandling så var antall informanter mellom 1 og 95, med en median på 28 og gjennomsnitt på 31 (Bryman, 2016, s. 417). Bryman (2016, s. 417) påpeker at forskjellen i median og gjennomsnitt kan tyde på enkelte høye verdier i enkelte avhandlinger. For mitt prosjekt var tanken opprinnelig tenkt på 10 informanter, men ble redusert til 6[5]. Mitt antall informanter er basert på behovet for å kunne besvare problemsstillingen og ha resultater som kan analyseres, vurderes og til å gi en diskusjon og konklusjon til slutt. Det vanlige svaret på antall informanter er: «[…] så mange det trengs for å finne ut det du trenger å vite» (Postholm & Jacobsen, 2018b, s. 148). Det er viktig å kunne forsvare antall informanter ut ifra de premisser jeg selv setter. Teoretisk metning er et begrep som benyttes og med det menes at man har nok informanter til å begrunne funn med teori (Bryman, 2016, s. 418–419).

Matematikklærere på ungdomstrinnet er fokus for utvalget av informanter. Bakgrunnen for dette ønsket er min egen interesse for programmering og matematikk, da jeg selv er lærer på ungdomstrinnet. Ut fra egen erfaring ser jeg at kolleger i varierende grad tar i bruk programmering som verktøy i undervisningen i matematikk. Lærere på ungdomstrinnet (8. – 10. klasse) var mest av interesse, da jeg som forsker (og lærer) i større grad kunne relatere meg til deres situasjon. Utvalget mitt er basert på lærere med og uten erfaring med programmering, både som kurs og utdannelse. Dette for å høre deres erfaringer og opplevelse, men også få vite noe om strategier lærerne gjør for å følge kompetansemål og kjerneelementer i fagfornyelsen. Algoritmisk tenkning og dybdelæring er også tatt med for å se hvordan disse begrepene oppleves og erfares av informantene. Etter snart to år med fagfornyelsen, har lærerne hatt tid til å erfare og oppleve fagfornyelsen sett i lys av nye kompetansemål i matematikk som inneholder programmering og algoritmisk tenkning.

Utvalg av selve informantene har skjedd i egne nettverk og i diverse grupper på Facebook. I tillegg til e-post til aktuelle skoler. Et bekvemmelighets utvalg, som Bryman (2016, s. 187) forklarer med; de som er tilgjengelig for forskeren. For eksempel en lærer som bruker egne elever, eller en

[5] Mot oppgavens slutt er dette tallet 5, da en av informantene trakk seg

professor som bruker egne studenter. Problemet med dette utvalget, er i hvilken grad informantene er et representativt utvalg for gruppen jeg ønsker å undersøke (Bryman, 2016, s. 187). Bryman (2016, s. 187) påpeker også at man ikke skal gå glipp av at en mulighet til å bruke et bekvemmelighets utvalg, men være klar over problemene som oppstår med generalisering av funn. «Snowball samling» eller snøballeffekten, beskrives som å ta kontakt med en gruppe som man deretter bruker for å oppnå kontakt med andre (Bryman, 2016, s. 188, 415). I mitt tilfelle så ble det brukt en kombinasjon av disse to tilnærmingene. 5 informanter ble med til slutt, etter at en måtte trekke seg. Informantene jobber ved fire forskjellige skoler. Alle underviser i matematikk. Informantene er gitt fiktive navn, slik at lesbarheten, slik jeg ser det, øker og at man får en relasjon til informanten gjennom narrativet som skjer i analysen. Intervjuene har skjedd i rekkefølge Anne, Mari, Arne, Lise og Ellen.

4.2.3 Gjennomføring av innsamling

Selve intervjuet som gjennomføres må planlegges. Intervjuet sees på som steg 3 i en syvstegs prosess (Kvale & Brinkmann, 2015, s. 137). Her legges det vekt på selve intervjuets kontekst og de relasjonene som finnes i intervjusituasjonen. Kvale og Brinkmann (2015, s. 114) deler forskningsintervjuet inn i 4 delers, hvorav de to første kanskje er innlysende: Intervjupersonen, intervjueren, kroppen og ikke-mennesker. I det følgende vil jeg derfor prøve å avdekke hva Kvale & Brinkmann (2015, s. 125 – 131) legger i «Kroppen og ikke-mennesker». Kort fortalt handler «Kroppen», om alt det det kroppslige rundt oss mennesker signaliserer til et annet menneske. Menneskers ytre kjennetegn slik som vekt, kjønn, klær, etnisitet og tatoveringer kan sende signaler som setter intervjuet i en kontekst. Kontekst blir i noen sammenhenger forklart som samfunn eller kultur, eller miljø og situasjon (Kvale & Brinkmann, 2015, s. 114). I mitt intervju så er det det som er innenfor intervjuet som er kontekst. Ikke-mennesker er de faktorer rundt intervjuet som ikke har kropp, altså de elementene som under intervjuet, som likevel kan påvirke situasjonen og gjennomføringen. Selve lydopptakeren, kameraet, notatblokka, en pc som bråker eller kjæledyr er eksempler på ting som kan påvirke. Selv en vond stol kan gjøre at informanten føler seg uvel (Kvale & Brinkmann, 2015, s. 129).

I min oppgave har jeg derfor valgt å bruke video-nettprat og lydopptak som datakilde. Video-delen begrunnes i at jeg vil få tak i det kroppslige som meddeles uoppfordret under intervjuet, og om det

er reaksjoner på spørsmål kroppslig. I tillegg vil det skape samme formalitet som under et ansikt-til-ansikt intervju (Archibald et al., 2019, s. 2; Bryman, 2016, s. 492). Forskning på området viser til enkelhet i bruk, kost-nytte-effekt, data-behandlingsmuligheter og sikkerhetsinnstillinger som nyttefulle og viser at for eksempel Zoom og Teams er bærekraftig innenfor kvalitativ forskning (Archibald et al., 2019, s. 1).

Hensyn til ikke-mennesker (Kvale & Brinkmann, 2015, s. 129–130) er viktig, og basert på positive erfaringer med Zoom og Teams, videokonferanse, er det riktig å benytte seg av disse med bakgrunn i fleksibilitet for informantene, muligheter til fort å lage ny avtale eller forskyve, informanten kan selv velge sted og dermed trygghet, og det er mulighet til å nå informanter uavhengig av geografi/lokasjon (Bryman, 2016, s. 492). Archibald et. al (2019, s. 4) trekker frem tilgang på informanter, tidseffektivitet og kostnadseffektivitet som hovedfordeler for innsamling via Zoom. Dette håper jeg skal gjøre at mine informanter i stor grad stiller opp til intervju.

Før selve intervjuet ønsker jeg å sende ut en kort informasjon om tema slik at informantene vet litt om hva studien handler om. Jeg velger ikke å sende ut selve intervjuguiden. Det er ikke ønskelig at informantene leser seg opp, og svarer det forskeren ønsker å høre (Kvale & Brinkmann, 2015, s. 122). I tillegg er det viktig å skape trygghet rundt intervjuet, og om hva som forventes av informanten. Et minimumskrav som settes for en slik invitasjon kan være:

1. Tilbud om å lese igjennom intervjuteksten for godkjenning.
2. Invitasjon til å diskutere tolkninger.
3. Invitasjon til å diskutere pedagogiske tiltak, dersom aktuelt.
4. Kunne lese den endelige rapporten.

   (Mellin-Olsen, 1996, s. 32)

Fleksibiliteten i et intervju via Teams har sine fordeler og ulemper. Fordelene er at man har mulighet til å avtale tid mye mer fleksibelt. Ulempene er det som kan skje med teknikken. Det første intervjuet hadde en litt trøblet oppstart, med et kamera hos intervjueren som ikke fungerte, og hodetelefoner hos informanten som ikke fungerte optimalt. Etter sjekking av innstillinger og utredning, var alt på plass. Intervjuet gikk godt, og det ble brukt ca 49 minutter i de første intervjuet. I det andre 56 minutter. Underveis ble det utdypende spørsmål fra min side, samt bekreftende spørsmål for å sikre at jeg hadde forstått riktig. I etterkant av intervjuet reflekterte jeg

over treffsikkerheten til mine spørsmål og velger å presisere noen spørsmål, samt se over guiden for redundans i spørsmålene og eventuelt flytte på noen spørsmål.

4.2.4 Intervjuguide

Intervjuguiden er dokumentet som styrer intervjuet. Graden av struktur bestemmer hvor stramt man følger rekkefølge og temaets oppbygning, samt i hvilken grad spørsmålene er nøyaktig formulerte. Det semistrukturerte forskningsintervjuet har mindre grad av rigiditet, og inneholder tema som skal dekkes, og forslag til spørsmål. Intervjuguiden er dermed en rettesnor for tema i intervjue som intervjueren/forskeren kan forholde seg til etter eget skjønn (Kvale & Brinkmann, 2015, s. 162)

Det er viktig at forskeren er fleksibel i sin tilnærming til intervjuet, og bruken av intervjuguiden. Det semistrukturerte intervjuet er nettopp det; mindre grad av struktur og legger opp til at forskeren kan komme med tilleggsspørsmål av typen:

1. Oppfølgingsspørsmål – der forskeren ønsker å vite mer dybde eller nyanser under selve intervjuet eller i et senere intervju etter at materiale er analysert. Slike spørsmål er da relatert det som er sagt i intervjuet.
2. Inngående spørsmål – er spørsmål som er til hjelp for forskeren for å få samtalen videre eller for å få informanten til å utdype, fortelle mer eller for å bekrefte at man har forstått riktig.
3. Oppklaringsspørsmål – der forskeren ønsker å bekrefte et standpunkt eller en mening. (Postholm & Jacobsen, 2018b, s. 122–126).

Intervjuguiden til denne oppgaven, er delt i flere deler, hvor den første delen er innledende spørsmål av typen: Alder, utdannelse og bakgrunn. Den andre delen handler om tema matematikk og programmering, algoritmisk tenkning og dybdelæring. Mens den siste delen handler om begreper og mer konkrete spørsmål.

Problemstillingen til oppgaven er: «Læreres utfordringer og erfaringer med innføring av programmering i matematikk i forbindelse med fagfornyelsen». Og dermed ønsker jeg å undersøke de faktorer som påvirker lærernes holdninger til programmering og algoritmisk tenkning, og

hvordan lærerne erfarer og opplever de føringer som ligger i styringsdokumenter. Jeg ønsker også å få vite mer om hvordan lærere innpasser programmering og algoritmisk tenkning i matematikkundervisning, og hvilke erfaringer og utfordringer de har i denne prosessen. Spørsmålene jeg stiller handler om å avdekke faktorer som kan vise hvordan lærerne jobber med programmering og algoritmisk tenkning. Hvordan skolen og kommunen har lagt opp kompetanseheving innen programmering og algoritmisk tenkning er et annet felt som er interessant, som blir belyst. Ønsker her er å se på faktorer som påvirker lærernes muligheter til personlig kompetanseheving, men også skolens kollektive kompetanseheving. Dybdelæring er et tema som er med for å vite hvordan lærerne ser på dybdelæring i matematikkfaget, men også i kontekst av programmering i matematikkfaget. I den siste delen har jeg lagt noen enkel korte spørsmål om begrep for å få en innsikt i lærernes begrepskompetanse innen programmering. Noen av spørsmålene traff ikke helt som planlagt og derfor forandrer jeg litt på ordlyd og oppfølgingsspørsmål for å få vite det jeg ønsker. Dette i henhold til forskningsdesign fra (Maxwell, 2013b, s. 5).

4.2.5 Dokumentasjon av innsamling

I skrivende stund er pandemien på hell, men med den utstrakte bruk av videokonferanse (Teams eller Zoom), så er mulighetene store for at informantene vil svare ja på å være med i prosjektet større enn om man avtaler intervju direkte ansikt-til-ansikt etter min erfaring. Dette støttes også av Bryman (2016, s. 492). Det blir gjort opptak av lyd og bilde i video-sesjonen. I infoskrivet opplyses det om at man kan samtykke til lyd og/eller video. Dersom man bruker diktafon via videokonferanse så får man et tap av lyd info dersom ikke pcens lyd tas opp direkte. Dette kan forringe kvaliteten av lydopptak. Teknisk databehandling av videofiler for å trekke ut lyd er mulig. Et videoopptak vil fange opp ansiktsuttrykk og kroppslige signal som er viktig for mening med utsagn og hvordan de skal tolkes og analyseres. Med tidskode på video, lyd og transkribering, kan man gå tilbake i videomaterialet for å se om utsagn man er usikker på har kroppslige signaler som kan si noe om informantens egentlige mening. Forskning viser at «ansiktsuttrykk og håndbevegelser som er synlig i en videokonferanse, bygger tillit, skaper engasjement og bidrar til en mer naturlig situasjon» (Archibald et al., 2019, s. 4). Et ensidig fokus på kun informantens del i intervjuet må unngås. Det er viktig å ta med forskerens bidrag i positiv og negativ retning. Viktig å identifisere forskerens påvirkning på intervjusituasjonen og konteksten.

4.2.6 Transkribering

For å få det muntlige intervjuet over i en tekstlig form så må det transkriberes. Transkripsjon kan være en møysommelig prosess, som tar tid. I mitt arbeide har planen vært å bruke dataprogrammet Nvivo som en hjelp i å strukturere, organisere og effektivisere arbeidet med transkripsjonen. Tanken er at Nvivo skal hjelpe til med strukturering og være til en grad tidsbesparende i transkripsjonsprosessen. Likevel så anser jeg at transkripsjonsprosessen vil være lærerik mtp videre arbeid med forskning. Transkripsjon uten hjelp fra dataprogrammer blir fort estimert til 5 timers arbeid for 1 times intervju av en erfaren transkriptør. Det munnet ut i 20 – 25 maskinskrevne sider ifølge (Kvale & Brinkmann, 2015, s. 207). Det vil være et anselig arbeid med strukturering, koding og analyse.

Selve ideen med transkripsjon er å strukturere de data jeg samler inn, slik at data blir tilrettelagt for analysen. Struktureringen er i seg selv en start på analysen (Kvale & Brinkmann, 2015, s. 206). E. G. Michler(1991) som (Kvale & Brinkmann, 2015, s. 206) viser til, at ved å transkriber selv som forsker gjør at man får et nærere forhold til selve dataene og at man lærer mer om egen intervjustil. Dette kan i mitt tilfelle være en utfordring ved bruk av dataprogrammet Nvivo.

Hvordan man bruker selve transkripsjonen og hvordan man gjengir, og koder transkripsjonen avhenger av hva slags analyse som skal utføres. Det finnes ingen standard løsninger, men det er en del valg som må tas (Kvale & Brinkmann, 2015, s. 208). Slike valg kan være om det er historien som er viktig, om det er begrepene eller om det er reaksjoner som informanten har. Detaljgraden av hvordan man koder transkripsjonen er avhengig av hvor interessert man er i de emosjonelle og kroppslige reaksjonene som informantene uttrykker. Man lager derfor en transkripsjonsnøkkel, der man har skriftlige tegn eller ord, som viser hvordan informanten uttrykker seg (Kvale & Brinkmann, 2015, s. 209). I min analyse så har det vært forståelsen av begreper, og informantenes meninger, erfaringer og opplevelser som har vært i fokus. Behovet for å beskrive tonefall, pauser, tonehøyde osv, har ikke vært tilstede i nevneverdig grad.

4.2.7 Transkripsjonsnøkkel

Et forslag:

[…] – Deler av et utsagn er utelatt, og resten er med.

[ ] – start og stopp i overlappende tale

= – pause i en ytring som fortsetter

(2) – pause i to sekunder

(.) – kort pause

. – fallende tonehøyde

? – økende tonehøyde

<tekst> – saktere tale enn vanlig

>tekst< – hurtigere tale enn vanlig

Listen er ikke uttømmende.

4.3 Bruk av analyseverktøy

Analyseverktøyene som har blitt brukt er Nvivo til tekstanalyse og behandling av data. I tillegg er Microsoft word blitt brukt som støtte til transkribering, systematisering og kategorisering av grupper med data, samt notatlogger og skrivelogger for tanker.

4.4 Etiske vurderinger og utfordringer

Informanter som deltar i et forskningsprosjekt, som denne masteroppgave i matematikkdidaktikk er, vil møte mange punkter der det er viktig at forskningsetiske vurderinger er gjort. Derfor er det viktig at vurderinger blir gjort for å ta avgjørelser som er i samsvar med etiske retningslinjer.

NESH – Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, utarbeider retningslinjer for etiske vurderinger innen forskning. I min forskning vil jeg støtte meg til NESH (2021) sine retningslinjer, samt benytte meg av prinsipper fra Bryman (2015).

To hovedprinsipper er viktige:

1. Hvordan skal vi behandle folk/informanter som vi forsker på?
2. Finnes det aktiviteter som vi bør eller ikke bør gjøre i samspill med folk/informanter? (Bryman, 2016, s. 121) *(min oversettelse/omskriving)*

Ved å ha disse vurderingene som en overordnet rettesnor, så er det enklere å gå videre med de forskningsetiske retningslinjene fra NESH. Søken etter sannhet er den basis som legges til grunn i all

vitenskapelig forskning for å sikre forskningens pålitelighet og kvalitet (NESH, 2021, s. 6). Det er forskerens ansvar å jobbe etter det retningslinjer som er gitt, og selv om retningslinjene til NESH (2021) ikke er uttømmende eller komplett, ligger det et spesielt ansvar for å identifisere etiske problemstillinger som retningslinjene ikke eksplisitt tar høyde for (NESH, 2021, s. 8). Forskningsetikken, følge Høgheim (2020), handler altså om å følge de «verdier, normer og ordninger som regulerer vitenskapelig» (Anker, 2020, s. 104; Bryman, 2016, s. 144–145; Høgheim, 2020, s. 80).

Bryman (2016) legger frem fire etiske prinsipper:

1. om det kan skade deltakere (harm to participants)
2. om det mangler informert samtykke (lack of consent)
3. om det er brudd på personvern (invasion of privacy)
4. om det er bedrag involvert (deception)

   (Bryman, 2016, s. 125)

Med bedrag mener Bryman (2016, s. 125), at den reelle forskningen eller det forskeren egentlig prøver å finne ut, ikke er meddelt informanten.

Informasjon om informanter eller deres identitet vil på ingen måte være tilgjengelig for andre enn forfatter av oppgaven eller veileder. Men skade er ikke bare fysisk skade, det kan være stress, bidra til dårlig selvbilde, utilpasshet og muligheter for at informantene blir satt i en situasjon de er ukomfortable i. Igjen er det viktig å vise til avsnitt over at listen ikke er uttømmende og det må derfor vises skjønn fra forskeren for å identifisere slike problemsstillinger (Bryman, 2016, s. 126).

Samtykkeskjema er vedlagt oppgaven og inneholder viktige elementer fra retningslinjer fra NESH og NSD – Norsk senter for forskningsdata (NSD, 2022a). Et slikt skjema vil inneholde informasjon formål, ansvar, innhold, frivillighet, personvern, rettigheter og kontakt info (NSD, 2022b). Mitt skjema er basert på malen til NSD, og er vedlagt.

Alle informanter har fått tilsendt samtykkeskjema i forkant av intervju, der de har fått informasjon om prosjektet, personvern og rettigheter. I tillegg så har vært intervju startet med en veiledning rundt hva det innebærer med opptak av lyd og video. Og hvordan opplysninger blir behandlet. Det

ble også opplyst når data vil blir slettet. Siden det er opptak av video, så ble bruken av videofilene forklart, og meddelt at det er kun lyddelen av video som var viktig. Men som nevnt i kap 4.2.3 så er det gunstig å kunne se den man intervjuer når man er via Teams eller Zoom (Archibald et al., 2019, s. 19). Identifiserende data er blitt fjernet fra transkripsjoner. Alt av datamateriale slettes etter prosjektets slutt.

4.5 Troverdighet

Gyldighet er et overordnet begrep som inne forskningen brukes for å vurdere forskningens kvalitet. Gyldigheten er dermed avhengig sammenhengen mellom mine konklusjoner og funn og virkeligheten, men ingen metoder kan med absolutt sikkerhet garantere dette (Maxwell, 2013b, s. 121). Troverdigheten er også relativ; den må vurderes i lys av den aktuelle forskningen som foretas og dens fasetter. Trusler mot troverdigheten, som Maxwell nevner, blir usannsynliggjort eller motvirket kun av bevis og ikke av metode (Maxwell, 2013b, s. 121).

Vi vurderer kvaliteten på forskning ved å benytte begrepene reliabilitet og validitet. Begrepet validitet er omdiskutert blant kvalitativ forskning, hvor det er koblingen til den absolutte definisjonen som er innen kvantitativ forskning (Anker, 2020, s. 108; Bryman, 2016, s. 383; Kleven, 2018, s. 100; Maxwell, 2013b, s. 122). Validitet innen kvantitativ forskning er delt i både intern og ekstern validitet. Og jeg bruker disse begrepene fra den kvantitative forskningen for å konkretisere og belyse forskjellene. Den interne validiteten er høy dersom vi kan trekke en slutning av at hendelse A er en årsak til B basert på resultater fra forskningen. Man har da høy intern validitet ved å kunne svare ja på dette. Høy ekstern validitet kommer av om man kan generalisere til andre situasjoner, grupper av personer eller tider (Bryman, 2016, s. 41–42; Thrane, 2018, s. 47,170). Det er nettopp i denne brytningen av begrepet validitet er omdiskutert innen den kvalitative forskningen. (Maxwell, 2013b, s. 122) argumenterer at validitetsbegrepet innen kvalitativ forskning er i stadig utvikling, og at han bruker: «[…] the correctness or credibility of a description, conclusion, explanation interpretation, or other sort of account». Altså troverdigheten til de data vi samler inn og analyserer.

På bakgrunn av dette så vil min kvalitative forskning vise til «funn og resultater», som er basert på de aktuelle informanter og den sosiale setting og samfunnstiden som var akkurat der og da. De

nøyaktige svarene til informantene vil være et produkt av den settingen, og vil ikke kunne reproduseres nøyaktig.

Som en avslutning kan det og nevnes at validitet og reliabilitet i sin kvantitative form gir mer mening når man benytter begrepene på transkripsjonens validitet og reliabilitet, da dette er tettere knyttet opp mot nøyaktighet og ord-for-ord (Kvale & Brinkmann, 2015, s. 212). Mer om dette i kapittelet om transkripsjon 5.2.6.

Bryman (2016) bruker troverdighet som en overordnet paraply for å kunne si noe om kvalitativ forskning. Hans 4 begrep blir nevnt under og videre. Og jeg vil på bakgrunn av denne diskusjonen benytte meg av Bryman (2016, s. 384) fire begrep for å si noe om reliabilitet og validitet i min forskning: Kredibilitet, overførbarhet, pålitelighet og bekreftbarhet, som jeg bruker begrepsavklaring fra (Bryman, 2016, s. 383–386; Kleven et al., 2018; Postholm & Jacobsen, 2018b).

4.5.1 Kredibilitet

Forskeren, informanten og den vanlige mann i gata vil se verden på hver sin måte. Kredibilitet er motsvarigheten til intern validitet i den kvantitative forskningen. Selv om vi i den kvalitative forskningen ikke kan si: «hvis x så y, og y så x». Kredibilitet oppnås derfor ved å ha åpenhet overfor informant på den måte at forskningen er utført etter gjeldene prinsipper og standarder. Samt at data og funn er tilgjengelig for informantene som er med i intervjuene (Bryman, 2016, s. 384). Tilgjengeliggjøring av funn fra forskning er ikke bare en god gest overfor en informant, men også vil sikre kredibilitet for forskningsprosjektet, og at undersøkeren har forstått den sosiale situasjonen eller verdenen som undersøkelsen er utført i (Bryman, 2016, s. 384). En annen spennende måte å sikre kredibilitet for forskningen på er å bruke flere metoder, som for eksempel triangulering eller «mixed methods» (Bryman, 2016, s. 386) .Kort fortalt betyr dette flere metoder for å se datafeltet for eksempel i prinsippet «mixed methods», der man bruker for eksempel en kvantitativ spørreundersøkelse først for deretter å ha kvalitative intervju for å kunne gå i dybden hos enkelte deltakere. Men et slikt prosjekt er for stort for min forskning.

Min søknad til NSD er godkjent for lydopptak, både med og uten video. Tanken her var at å sikre gode opptak. Lydopptak via diktafon via Teams/Zoom kan skape en forringelse. Men begge deler vil

bli brukt for å skape redundans og backup. Infoskriv og samtykkeskjema er utarbeidet. Gjeldene retningslinjer fra NSD og USN er fulgt.

4.5.2 Overførbarhet

Overførbarhet sammenlignes med ekstern validitet i den kvantitative forskningen. Det engelske «transferability» bruker, og ofte bruker det norske: «transfer», som i transfer av kunnskap. Generalisering og gjenskaping av de samme resultatene i kvalitativ forskning er vanskelig da vi med intervjuer går i dybden hos individet. Dette fordrer at vi beskriver den sosiale settingen og parameterne som den sosiale verden som informantene befinner seg i. Selv om settingene er tilsynelatende like, så overlates tolkningen av overførbarhet til mottakeren av forskninger til å tolke om overførbarhet er mulig (Bryman, 2016, s. 384). Dette gjøres ved bruke «tykke beskrivelser». Tykke beskrivelser, eller «thick descriptions», er fyldig beskrivelser av detaljer av den sosiale verden som undersøkelsen finner sted i. Dette gjøres ved å beskrive handlinger og meninger, og i hvilken kontekst alt finner sted. Når beskrivelsene er så tykke(fyldige) at leseren føler at han er med i eller har erfart dem selv. Mottaker vil da kunne være i stand til å relatere dette til egen setting eller kultur og dermed selv avgjøre overførbarhet (Bryman, 2016, s. 384; Postholm & Jacobsen, 2018b, s. 239).

Intervjuguide og svar fra analyse og funn vi gi en pekepinn på mulige utfall. Jeg redegjør i analysen for hvilke svar lærerne kommer med, og tar med språklige virkemidler og kroppsspråk i analysen der det er hensiktsmessig for å kunne gi utdypende meninger til et utsagn.

Oppgaven vil publiseres etter ferdigstillelse, og involverte informanter vil få tilbud om å kunne få se resultater. Forskningen kan ikke nøyaktig utføres for å få nøyaktig de samme svarene, men likevel kan forskningen etterprøves ved å følge samme metodikk og fremgangsmåter og dermed har mulighet til å verifisere mine funn og resultater.

4.5.3 Pålitelighet

Pålitelighet fra Bryman (2016) kalles «dependability», og dette er i motsvarighet til den kvantitative forskningens reliabilitet. Konseptet er å sikre fullt innsyn i prosessen fra problemformulering til ferdigstillelse. Dette vil sikre at påliteligheten ved prosjektet er overholdt (Bryman, 2016, s. 385).

I mitt prosjekt handler dette om å beholde notater fra intervjuer, transkripsjoner, utvelgelse av informanter, analysebeslutninger (stikkord tatt fra (Bryman, 2016, s. 384)).

I dette prosjektet har jeg benyttet transkripsjoner av lydfiler, notater rundt tankerekker fra intervjuene. Alt har vært systematisert og holdt utilgjengelig for andre enn meg selv. Video som har vært benyttet via Teams har vært sikret slik at de ikke er tilgjengelige for andre enn meg selv. Samtykkeskjemaer er lagret og beholdt.

4.5.4 Bekreftbarhet

Bryman (2016) sitt «confirmability», anerkjenner at absolutt objektivitet ikke kan forventes, så ligger det et premiss til grunn om at forskeren har handlet i god tro. Forskeren skal heller ikke har en egen agenda eller har latt personlige verdier farge resultater eller forskning (Bryman, 2016, s. 386). Analyser og funn må kunne bekreftes i det datamaterialet som er lagt til grunn, og at det er sammenheng mellom analyser, beskrivelser og tolkninger som er gjort (Postholm & Jacobsen, 2018b, s. 230).

I mitt arbeid med denne oppgaven har det vært viktig å ikke påvirke informantene og ikke stille ledene spørsmål, da det er deres svar og betraktninger som er de viktige. Jeg har et stort engasjement innen matematikk og programmering, og en av fallgruvene for forskningen min er å bli for ledene i spørsmålene. Utarbeidelsen av intervjuguiden vil ha dette som fokus. Likevel vil arbeidet med analyse av datamateriale vise at det er vanskelig å unngå å «vise vei» for å få frem kunnskap som finnes, men må hentes frem hos informantene. I de tilfeller hvor dette skjer så vil det komme frem av diskusjonen.

5 Analyse – drøfting

Funnene fra semi-strukturerte intervjuer med lærerne, blir analysert og diskutert opp mot aktuell teori. Funnene er delt inn i fire hoveddeler, som reflekteres i kapittel inndelingen fra 5.1 til 5.5. Hvor kapittel 5.1 tar for seg lærernes forutsetninger, slik som utdannelse, kursing innen programmering, antall år i undervisning og fag. I størst mulig grad er sitater i satt med «anførselstegn» og gjort innrykk, men også er kondensert tekst brukt for å gjengi informantenes synspunkt. Spesielt der hvor sitatene fra transkripsjonen, har høyt preg av muntlig resonnering og uklare argumenter frem og tilbake.

5.1 Utdannelse, erfaring og kompetanse

Nøkkel:

M = Matematikk, N = Naturfag, K = Kroppsøving, R = KRLE, MF = Matematikk fordypning, E = Engelsk, IT = IT-VGS, PV =Prog. valgfag

A = Adjunkt, AT = Adjunkt + tillegg, L = Lektor, LT = Lektor + tillegg, MS = Master student

Tabell 2 Informantenes bakgrunn

Informant	Utdannelse	Fag	Programmerings-kompetanse	År i skolen	Underviser på trinn:
Inf.1 «Anne»	A	M,N,K	Små kurs	4	9 – M, N
Inf.2 «Mari»	AT	M,N,K,R	IT-VGS	6	10 – M, N
Inf.3 «Arne»	A+MS	M,N,E	Kurs USN, intern kursing, fadder-ordning	4	8,10 – M, N, E
Inf.4 «Lise»	AT	M,N	Kurs, informatikk grunnfag, kollegaveiledning, Har søkt 15stp k.	23	1. VGS, M, N Har: 7,8,9,10
Inf.5 «Ellen»	AT	M, N, PV	15 stp NTNU	4	NÅ: 8 trinn 8, 9, 10

5.2 Utfordringer og erfaringer

I dette delkapitlet vil jeg se på informantenes opplevelser og tanker rundt algoritmisk tenkning og programmering. De to første underpunktene (5.2.1 og 5.2.2) vil ta for seg to store aspekt; mangel på tid og ønsket om kompetanseheving. Det andre underpunktet (5.2.3) er lærernes syn på programmering og algoritmisk tenkning (AT). Tilslutt (5.2.4) vil jeg bruke Finger og Hougets modell (Sentance & Csizmadia, 2017, s. 473) fra 2009 for indre og ytre utfordringer, for å se hvordan denne samsvarer med mine funn hos informantene.

5.2.1 Tid og kompetanseheving

> «[…] det har vel stoppet meg på grunn av mangel på kunnskap. Og ferdigheter fra min side, da? Jeg synes det bare vi burde vært obligatoriske kurs for alle matematikklærere nesten lagt inn, som alle måtte delta på» (Anne)

> «Ja det må jo si at jeg skulle ha kunnet det enda bedre, men jeg synes ikke det egentlig hindrer meg så mye. Så nei, tror jeg» (Mari)

> «Det er en holdning i på sett og vis deler at læreplanen er utformet og satt ut i live. Eller vi skal jo sette den ut i live. Ja, men så opplever vi at vi kanskje ikke har den støtten vi kanskje forventer at vi skulle fått når du kommer til det vi trenger for å kunne gjøre det på en effektiv måte da. Sånn som for eksempel kursing og sånne typer ting, at selv om vi får tilbud om vi får tilbud om kurs, men og vi har vært på kurs og vi klarer å henge med på kurs. Men allikevel så føler mange av kollegene mine seg ganske uklare til å ta den jobben som må også være programmeringslærer både i matematikk og i andre fag» (Arne 1)

> «[…] grunnen til at jeg føler meg kompetent til å sette det ut i livet nå er fordi jeg har brukt ekstremt mye egen tid fordi jeg har hatt lyst til å gjøre en god jobb med det» (Arne 2)

> «De er veldig positive til å sende folk på kurs. De (ledelsen) er også positive til å hente inn folk for å lære oss alle det vi får tid til å sette oss ned for å lære hverandre og utforske og lage opplegg». (Lise 1)

> «[…] det kunne vært enda mer systematisk. […] sette av tid og la oss på en måte samarbeide og lage opplegg […] burde alle vært kurset samtidig. Jeg har mye mer tro på å sende hele kollegiet på kurs, enn å sende 2 stykker som skal komme tilbake og fortelle hva det kurset handler om.» (Lise 2)

Informantene Anne, Mari, Arne og Lise gir uttrykk for at tid til å drive med systematisk arbeid med egen kompetanseheving, for å kunne ta i bruk programmering i matematikkfaget er en utfordring, som er i tråd med Mishra et al. (2009, s. 62). Vurdering av egen kompetanse opp imot forventninger i læreplan er også et aspekt som Anne og Mari tar opp. Arne tar opp behovet for støtte fra ledelse som en utfordring, og er på linje med Anne og Mari, når han i sitt siste sitat over her impliserer behovet for mer tid til kompetanseheving, ved at han påpeker høy bruk av egen tid til kompetanse heving. Bocconi et. al (2018) påpeker at det bør settes av tid til kursing og at det blir lagt et grunnlag for at både skoler og skoleeiere, bruker vikarer og gir tid, til at lærere skal kunne delta i kompetanseheving. (Bocconi et al., 2018, s. 24; Stigberg & Stigberg, 2020, s. 487). Sosiale medier tas også frem som et punkt for å spre kunnskap og erfaringer blant lærere i kollegiet (Bocconi et al., 2018, s. 24; Stigberg & Stigberg, 2020, s. 487), og med sosiale medier menes ikke bare Facebook, men også samarbeidsarenaer som for eksempel Teams. Erfaringene uttrykt av informantene om liten tid til å benytte programmering inn i matematikkfaget støttes av Mozelius et al. (2019), der det rapporteres om liten tid til integrering av programmering i faget, stofftrengsel i pensum og at det kan bli en avveiing mellom tid og prioritering (Mozelius et al., 2019, s. 4). Altså at programmering kan bli nedprioritert på grunn av stofftrengsel. Informanten Lise skiller seg ut fra erfaringene til Anne, Mari og Arne, der hun i (Lise 1) påpeker at ledelsen er positiv til kursing og utdannelse, men i (Lise 2) ønsker hun likevel mer tid til samarbeid på team og kurs. Dette er i tråd med funn nevnt over av både (Bocconi et al., 2018, s. 24; Stigberg & Stigberg, 2020, s. 487), men ikke uttrykt like eksplisitt som informantene Anne, Mari og Arne.

Arne påpeker tydelige utfordringer med at elevene ikke har hatt LK20 i hele sitt løp, da det er kompetansemål fra 1. – 10. klasse, som det legges opp til at man har, når man begynner i 8 klasse. Arne henviser også til at elever ikke alltid har den matematiske kompetansen som LK20 sier de skal ha hatt. Læreren må derfor jobbe med grunnleggende kompetanse før de reelle

kompetansemålene, og må starte på et lavere nivå enn planlagt. Sitatet (Lise 3) under støtter Arnes utfordring med lite motiverte elever, at elevene ikke har den kompetanse LK20 sier de skal ha.

> «[…] elevene ikke er særlig hverken motivert eller interessert. De synes det er vanskelig, og de synes det er bortkastet for de synes de ser ikke nytteverdien i det, fordi at de kan det for dårlig. Altså, jeg tror det har gått for fort fram for dem i år, fordi at læreplanen på videregående gjør noen forutsetninger om at du har noen erfaringer fra barneskolen» (Lise 3)

Dette relateres også til spørsmålene rundt kompetansemålene på 6 versus 10 trinn og problematikken der (mer senere i kap. 5.4), og støttes av erfaringene til Larke (2019, s.1147) hvor funn viser at lærere tar bevisste valg i undervisningen, der hvor elevene og/eller lærerne mangler kompetanse som står i pensum for faget (Larke, 2019, s. 1147).

Informanten (Lise) må bruke tid på elevers grunnleggende kunnskaper innen programmering, før man kan bruke programmering til utforskning. Dette var spesielt utfordrende på klasser som har kun hatt et år med LK20. Videregående skole har forventninger om kunnskap innen programmering.

> «Vi har ikke hatt noe tid som har vært satt av til det her. Jeg var jo en av de første på skolen som fikk det programmering studiet, og da fikk jeg beskjed om at jeg skulle kurset de andre, men jeg fikk aldri noe tid til å kurse de andre […] jeg etterspurte om jeg kunne liksom ha planleggingsdag for teamet der altså en time i planleggingstiden, som jeg kunne ha kurs med de andre med det jeg kunne, men det ble aldri satt av noe tid». (Ellen)

Ellens erfaringer er i samsvar med de andre informantenes erfaringer, der tid til egenutvikling. Det som er interessant i Ellens tilfelle er at det har vært kommunikasjon og forventning fra ledelse om at Ellen skulle bidra i lokal kompetanseheving, men denne muligheten ikke er fulgt opp. Dette er en ytre utfordring der det ikke er samsvar mellom skolens forventninger og reell støtte og kommunikasjon med skolens ledelse (Sentance & Csizmadia, 2017, s. 488), men i kontrast til at læreren «at læreren må kunne drive eget utviklingsarbeid og bidra til en delingskultur rundt læring i digitale omgivelser», når grunnlaget for dette ikke er tilstede (Kelentrić et al., 2017, s. 13). Det er

derfor implisitt at dersom et rammeverk med krav til lærere skal legges til grunn så må det også tilrettelegges for dette på skole- og skoleeiernivå (Bjørnsrud, 2014, s. 101).

En annen utfordring har vært i kollegiet da alle ikke har samme perspektiv, interesse eller forståelse for programmering i matematikkfaget. Dette står i kontrast til forventningene i Stortingsmelding 20, 2012 – 2013, s. 158:

> «Selv om den enkelte lærer har en selvstendig rolle i opplæringen av elevene, er opplæring også et lagarbeid. Målene for skolens virksomhet kan ikke nås dersom skolen ikke mobiliserer de ansattes samlede kompetanse. Det stiller krav til faglig samarbeid. I et kollegialt felleskap vil ulik fagkompetanse og spesialisering hos lærere være en styrke. Gode lærere er aktive bidragsytere i et profesjonelt faglig fellesskap som utvikler skolen som en lærende organisasjon» (Meld. St. 20, 2012, s. 158)

Kelentrić et al. (2017) sier følgende om læreres ferdigheter og kunnskaper i «Profesjonsfaglig digital kompetanse» (PfDk):

> «kan benytte ulike digitale arenaer til å støtte samhandling og utvikle gode relasjoner til elever, foresatte, kollegaer, ledelse og andre relevante aktører» (Kelentrić et al., 2017, s. 11)

> «kan delta på digitale arenaer og bruker profesjonelle nettverk for egen læring og utvikling, og for kunnskapsdeling mellom kollegaer»
> (Kelentrić et al., 2017, s. 12)

Rammeverket legger føringer på hvilken digital kompetanse en lærer bør besitte, og når man setter erfaringene fra Anne, Mari og Arne, om mangel på støtte fra ledelse så kan det se ut til at skolekulturen ved deres skoler bør endres. Utsagn fra Lise i (Lise 2) støtter økt fokus på kompetanse heving på systematisk nivå.

Skoleeier og ledelse ved skolene har et spesielt ansvar for å legge til rette for teamsamarbeid, og det er eksplitt nevnt i ovenstående sitat (Meld. St. 20, 2012, s. 158). Denne organiseringen i team er i særdeleshet viktig for lærer og for skolen, for å bygge på den kompetansen som allerede finnes i kollegiet. Skolens egen kultur legger grunnlag for hvordan teamarbeidet blir. En etablert skolekultur

kan være vanskelig å forandre og opprettholdes av sterke krefter i kulturen (Bjørnsrud, 2014, s. 99–101; Wittek & Bratholm, 2014, s. 20).

> «Det er ikke mulig å utvikle en skole gjennom dyktige enkeltlærere eller gode enkeltledere alene. Det må systematisk arbeid og samordnet innsats til. Det kreves kunnskap, gode profesjonsholdninger, og ledelsesverdier, og støttende strukturer til på alle nivåer» Irgens (2012, s.229) gjengitt i (Bjørnsrud, 2014, s. 101).

Her er Irgens (2012) på linje med intensjonene i PfDK, som har fokus på lærernes faglige kompetanse innen det digitale.

Bjørnsrud (2014, s. 109) trekker også frem funn fra en barneskole med gode erfaringer med teamarbeid:
«[…] vi lærer av hverandre; vi veileder hverandre, utveksler tanker, erfaringer, ideer, frustrasjoner og observasjoner med hverandre gjennom hele skoledagen […]» (Bjørnsrud, 2014, s. 109). Denne kontrasten til informanten, Arne, sine erfaringer kan gi grunnlag for å aktivt jobbe med å innføre god samarbeidskultur innen AT og programmering. Arne sine betraktninger av teamarbeid støttes også av Anne og Mari i senere diskusjoner innenfor kompetanseheving.

> «Så her hvor jeg jobber nå. Så er det ikke noe sånn veldig stor interesse for det, og. Det faktisk er sånn at den klassen jeg har nå har ikke hatt programmering, til tross for at det står det på kompetansemålene at det skal det» (Mari, tok over en klasse siste 3 måneder).

Maris opplevelse er i samsvar med Arne, og står i kontrast til overnevnte momenter fra både Stortingsmelding 20 (2012 – 2013) og Bjørnsrud (2014).

5.2.2 Kompetanseheving

«Så hvis jeg skulle liksom designet et sånt drømmekurs som jeg kunne tenkt meg vært på selv, så kunne det vært at vi rett og slett hadde vi fikk en dag og vi kunne møtes og prate sammen om ulike problemer, og så at vi kunne bruke kompetansen som jeg tenker ligger i profesjonen.» Arne gir her en beskrivelse av innhold i kurset han ønsker seg, men det er også en kritikk i dette. Lises utsagn er også i tråd med Arnes ønsker: «[…] å sette av tid og la oss på en måte samarbeide og lage opplegg om det her». Arne nevner at han opplever på kurs fra USN har vært gode, men lite fokus på

kompetansemålene i LK20: Arne beskriver her et ønske om mer søkelys på skolens kompetansemål for å gjøre det mer praksisnært. Arne ønsker også et skifte fra forelesninger om grunnleggende programmering til mer praksisnært. Disse sitatene[6] fra Arne uttrykker en utfordring med kursing, og et ønske om forandring. Systematisk kursing for hele kollegiet, tid til samarbeid om opplegg og videreutdanning er i tråd med Lise og Arnes ønsker.

Kurs som er satt opp på denne måten som Arne ønsker viser til økt forbedring i programmeringskompetanse hos lærere:

> «A particularly effective form of CPD is collaborative, which can be defined as "teachers working together on a sustained basis and/or teachers working with LEA or HEI or other professional colleagues"[3]. In all but one of 266 studies of collaborative CPD reviewed by Cordingley, Bell, Rundell & Evans [3] there was a definite teacher improvement as a result» (Sentance et al., 2012, s. 3).

Som forklaring til sitatet over så betyr CPD, «continuing professional development», noe som kan relateres til Udirs KFK[7] – Kompetanse for Kvalitet, et videreutdanningsprogram for lærere. LEA (Local Education Agency/Authority) og HEI (Higher Education Institution) er forkortelser for lokale utdannings organisasjoner og HEI er høyere utdanningsinstitusjoner(APAC, 2022; Texas Education Agency, 2020).

Arnes ønsker og tanker er derfor på linje med fra Sentance et al., (2012, s. 3). Når man deretter ser på hvordan Anne og Mari posisjonerer seg sammenlignet med Arne, så ser vi at de nevner tverrfaglighet, lengre kurs og at kurs er timeplanfestet og prioritert. Anne og Maris tanker om tverrfaglighet korrelerer med Arnes ønsker om bruk av musikklærere, naturfagslærere og kunst- og håndverklærere, og jeg anser at informantene har samsvarende syn på hva slags kurs de ønsker seg, selv om Arnes svar er mer utfyllende og detaljert.

> «Jeg føler meg liksom ikke så god i programmering som jeg helst skulle vært, selv om jeg har tatt de 15 studiepoengene, så føler jeg fortsatt ikke at jeg har nok kompetanse. Men jeg er

[6] Finnes i vedlegg 3.
[7] Se også kap 2.6 og 2.6.1

jo flinkere enn mine elever er nå, men når de elevene som begynner med programmering og koding på barneskolen nå kommer til åttende trinn, da kan det godt hende at de er langt foran meg. Det tror jeg kan være litt sånn ekkel følelse, jeg liker jo på en måte kunne mer enn elevene.» (Ellen 1)

«[…] det burde vært lettere å få ta programmerings studier. […] vært mer kursing av hele personellet, med programmering fordi det er jo ikke bare mattelærerne som trenger programmering. Men det er jo også kompetansemål i flere fag enn matte som inneholder programmering, så det burde vært satt av flere dager. […] Og så synes jeg, som jeg sa tidligere intervjuet at det burde vært jobbet med programmering blant lærerne før kompetansemålene kom. For det ble liksom slengt litt i trynet på oss, på en måte» (Ellen 2)

« […] har jeg jo prøvd liksom sitte litt sånn hjemme på kveldstid og prøve ut opplegg, men det handler jo også om at jeg har programmering valgfag, så jeg må finne opplegg» (Ellen 3)

Ellen er med disse 3 utsagnene på linje med erfaringene og ønskene fra de andre informantene, men hun legger i tillegg til at hun bruker tid hjemme til egen kompetanse heving. Ellens har en indre utfordring i egen kunnskap, hun føler seg likevel ikke fullt ut kompetent, og frykter for når elever som kommer fra barnetrinnet om noen år har bedre kompetanse enn henne selv. En ytre utfordring for Ellen, er mangelen på støtte, i det ledelsen ikke følger opp avtalt skolebasert kompetanseheving kurs som Ellen skulle holde (Bjørnsrud, 2014, s. 82, 101; Sentance & Csizmadia, 2017, s. 488). Ellens erfaringer er i tråd med erfaringer fra forskning, der ikke-ledelsesstyrt organisering av skolebasert kompetanseheving opplevelse er vanskelig for lærerne (Postholm et al., 2013, s. 102–103).

Videre er det et interessant funn hos Arne da han skisserer en lokalt initiert kompetanseheving som innebærer at en kollega har laget et opplegg for kollegiet på frivillig basis etter arbeidstidens slutt:

«Jeg har en kollega som har utdanning innenfor programmering, ikke i forbindelse med lærerjobben, men i forbindelse med tidligere utdanning. Og han har brukt sin fritid […] har jobbet gratis som har brukt sin fritid flere onsdags ettermiddager og lagt til rette for at

> andre lærere også kan bruke sin fritid på å gå og ha kurs med han, der hvor han har laget oppgaver i matematikk og naturfag. […] Vi har vært elever hos han før for å si det sånn da i på en sånn slags programmering skole». (Arne)

Her har lærerne laget et lokalt forankret opplæringssystem i tråd med funn støttet av Sentance et al., (2012, s. 3). Et slikt kompetansehevingsinitiativ som Arne beskriver er i tråd med Vygotskys ZPD, iom at man har kolleger med ulik kompetanse og man har et læringsfelleskap (Vygotski et al., 1978, s. 87). Videre påpekes det i Sentance et al., (2012, s. 3), at de skolene som er høyest ytende også rapportert at de var med i profesjonelle læringsaktiviteter. Slike aktiviteter kjennetegnes ved at de er av lengre varighet, mer aktive og at delingskulturen mellom kollegaer er hyppigere. Dette viser at kompetanseheving kjennetegnes av kvalitet, og kan måles på en skoles ytelse og effektivitet (Sentance et al., 2012, s. 3).

5.2.3 Programmering og algoritmisk tenkning

Anne peker på fremtiden og at mange elever skal sitte bak en skjerm, og elevene vokser opp i en digital verden. Mari sine betraktninger:

> «[…] det er dit verden går, og har vært på vei lenge. Er på tide at man lærer dette før man velger det på høyskolen. Forandringer skjer […] få bredere forståelse av matematikk teorien ved å gjøre AT i programmering. Lære seg AT og programmering er nyttig og fremtiden, for å forstå verden bedre. Programmering vil kunne være et verktøy på linje med Excel og Geogebra».

Dette i tråd med tankene til Pellegrino og Hilton (2012), som er et av bakgrunnsdokumentene til NOU (2015:8), Ludvigsenutvalget, om hva som kreves av fremtidens elever (NOU 2015:8, 2015; Pellegrino & Hilton, 2012, s. 32), samt vi beveger oss inn på forskjellene mellom dybdelæring og overflatelæring beskrevet av Sawyer (2006) i (Sawyer, 2006, s. 4).

Anne forklarer algoritmisk tenkning som en steg-for-steg-måte å se strategisk og systematisk på ting. Denne betraktningen av AT og programmeringens plass i matematikkfaget er forenlig med

figur 3.7 og figur 3.8, med kjernebegrep for AT (Barefoot Computing, 2022a; Utdanningsdirektoratet, 2019a). Videre følger Anne opp med følgende utsagn:

> «[…] så må du faktisk tenke algoritmisk med å bryte ned oppgaven. Hva skal jeg finne ut hva relevant informasjon? Hva er ikke relevant informasjon? Og så kan du sette sammen. Hva skal jeg faktisk bruke OK? Og så lage en plan steg for steg».

> «[…] vi kan jo bruke programmering inn i alt fra å lage mønster og mangekant. Altså da tenker jeg nå er jeg jo midt inne i geometri selv. Se mønster og gjentagende mønster, for eksempel. Bruke til store datasett. Finne typetall og median ganske fort […].»

Her trekker Anne inn dekomposisjon, abstraksjon, evaluere, mønstre og skape, som er sentral begreper fra «den algoritmiske tenkeren fra Udir (Utdanningsdirektoratet, 2019a).

Arne mener begrepene AT og programmering hører tett sammen. De begrunner hverandre. Ikke alle trenger å beherske et programmeringsspråk direkte selv, men kan ha nytte av den faglige tankegangen for å kunne løse andre problemer og oppgaver i livet. Evnen til å tenke teknisk på et problem og bryte opp et ganske omfattende og sammensatt problem i mindre deler. Nyttig strategi som man trenger i livet. Programmering er fin måte å trene algoritmisk tenkning på. Dele opp problem i mindre biter. Utsagnene til Arne er i tråd med definisjonene på AT hvor dekomposisjon blir omtalt som å dele i mindre bolker. Arne bruker også abstraksjon og evaluering som begrep uten å bruke de rette ordene, samt «prøve og feile» i sitat nr 2 under her:

> «[…] men sånn generelt for å løse problemer i livet. Og det er jo da man kommer opp at den automatiske tenkningen som er den evnen å tenke teknisk på et problem og bryte opp et stort og ganske omfattende og sammensatt problem inn i mindre bolker. Og så se på de ulike bolkene og se hva slags informasjon man trenger å eventuelt hente inn eller hva slags informasjon man har, og som man kan bruke for å komme fram til en løsning på et større problem» (Arne)

> «[…] som demonstrasjons verktøy så synes jeg også det er veldig kraftig fordi man på en veldig enkel måte kan vise fram hvordan det kan være man kan vise fram prøve og feile

> metoden på sett og vis ved å få et program til å prøve å feile for seg og at det også kan være en nyttig strategi selv med ganske komplekse problemer». (Arne)

Arne bruker eksempel fra egen undervisning der han viser til demonstrasjonsforsøk med sannsynlighet og terningkast. Antall kast kan drastisk og kjapt økes. Muligheter for kjappe utregninger ved bruk av programmering. Kraften til en datamaskin til å regne ut for eksempel 10000 kast på terning, viser nytten av programmering.

> «Og så tenker jeg også at programmering som verktøy er et veldig kraftig verktøy å bruke inn i matematikken, og at det kanskje er derfor matematikken i hovedsak har fått største delen av jobben når det kommer til det å implementere programmering» (Arne).

Denne analysen av begrep knyttet til algoritmisk tenkning viser at informantene har varierende kunnskaper om begrepene som benyttes i algoritmisk tenkning.

Arne mener det kan være enorme fordeler med å lære seg AT å tenke på når man møter problemer. Problemløsningsoppgaver eller se på sammensatte problem. For eksempel i geometri og regne ut volum ved å angripe et problem som mange delproblemer. Disse fordelene som Arne her setter opp som fordeler, blir sett på som problemområder (Gomes & Mendes, 2014, s. 3; Sentance & Csizmadia, 2017, s. 488), og at det da kan være viktig å se på strategier for å veilede ved innføring av AT. Slike strategier kan være:

1. Forandre på kode, forbedre kode, steg-for-steg-analyse av kode (evaluering og generalisering)
2. Bryte ned kode og/eller konsept/ide i mindre deler (dekomposisjon)
3. Utvikle algoritmer (selve AT, abstraksjon og evaluering
   (Sentance & Csizmadia, 2017, s. 491)

Arne forklarer at programmering kan være ganske firkantet, og kommafeil syntaksfeil kan bli utfordrende. Og påpeker toleransen for skrivefeil/syntaksfeil kan bli problematisk. Andre utfordringer kan være den grunnleggende digitale kompetansen. Kan bli utfordrende å gå fra enkel programmering til å sette sammen et mer komplekst problem. Mange kan variabler, løkker og hvordan få input, men å sette sammen til mer komplekst og sammensatt kan bli utfordrende.

> «[…] elevene ikke er særlig hverken motivert eller interessert. De synes det er vanskelig, og de synes det er bortkastet for de synes de ser ikke nytteverdien i det, fordi at de kan det for dårlig. Altså, jeg tror det har gått for fort fram for dem i år, fordi at læreplanen på videregående gjør noen forutsetninger om at du har noen erfaringer fra barneskolen». (Lise)

Som Arne og Lise over nevner, så er manglende kunnskaper innen matematikk og programmering vil være en utfordring. Selve kodingen, her inntastingen, i et program er problemløsning og kjernen i ATs område. Med det mener jeg at når man programmerer vil man benytte flesteparten av ATs begreper i kodingen og programmering. Syntaksfeil vil derfor være en del av «prøve og feile»-strategi, og elever er lært opp til å søke suksess og ikke feil, og dermed ikke komfortable med å lære av å gjøre feil (Sentance & Csizmadia, 2017, s. 491).

> «Letting them have a go and allowing them to get it wrong. They need to become problem solvers, not just the teacher telling them what to write. The students must code for themselves without too much intervention from the teacher in terms of theory. There has to be some of course, but after they have tried to figure it out first» (Sentance & Csizmadia, 2017, s. 491).

Dette sitatet kommer fra 1 av 336 intervjuede fra Sentance og Csizmadia (2017) om strategier for å unngå at elever kun vil ha et svar, og ikke er villige til å holde ut og fikle. To andre av 336 hadde lignende strategier. Holde ut, fikle og feilsøke er tre av verktøyene i Udirs algoritmiske tenker (Utdanningsdirektoratet, 2019a).

Anne peker på fordeler ved programmering at man kan komme raskere til et frem til et mål. Man kan effektivisere undervisningen. Anne bruker et eksempel på linje med Arnes eksempel på terningkast i sannsynlighet. Både Anne og Arne har sammenfallende tanker når det kommer til nytten av programmering som demonstrasjonsverktøy. Arne nevner demonstrasjonsverktøy eksplisitt, mens Anne bruker uttrykk slik som «å effektivisere undervisningen» og «komme raskere frem til et mål».

Anne og Mari deler et perspektiv på at lærere vil ha utfordringer med å huske det samme som elevene skal huske – alle begreper og symboler innenfor programmering. I tillegg nevner Mari at «mange vil komme til kort når matematikken blir så avansert at ikke de får til programmeringen». Anne, Mari og Lise nevner de foregående utfordringene, som er i tråd med Sentance og Csizmadia (2017), med elevers manglende matematiske evner (Gomes & Mendes, 2014, s. 3; Sentance & Csizmadia, 2017, s. 488).

5.2.4 Indre og ytre utfordringer

Tabell 3 Indre og ytre utfordringer - basert på Sentance et al., (2017, s. 488)

	Utfordringer	Informant		Informant
	Indre	AN, MA, AR, LI, EL	Ytre	AN, MA, AR, LI, EL
Lærere	Egen kunnskap Programmerings-didaktikk Differensiere (evne til å)	AN, MA, LI, AR[1] AN, LI, MA[2]	Mangel av støtte fra led. Mangel på tid Tekniske problemer Ressurser	AN, MA, AR, EL[3] AN, MA, (LI), EL[3] AN, MA AN, MA
Elever	Matematiske evner Problemløsningsevner Mangel på forståelse Motivasjon	AN, MA, AR, LI (AR) AR, LI	Skolens forventninger (læreplan, komp.mål) Tid	AN, MA, (AR) AN, MA, (LI), EL

Noter:

1. Nevner at han føler seg kompetent til en viss grad, men påpeker at i samarbeid med kolleger og som skole, er de i stand til å dekke kompetansemålene i LK20.
2. Både Anne og Mari nevner behov for kursing og kompetanseheving, samt lite samarbeid på team og hovedtrinn[8].
3. Ellen påpeker at hun ønsker seg kursing av hele kollegiet, og ikke bar eat noen få sendes på kurs for å lære opp de andre lærerne.

De indre og de ytre utfordringene som lærerne rapporterer, kommer fram i tabellen over, som er basert på Sentance et al., (2017, s. 488). Tabellen viser også lærernes utfordringer med hvilke

[8] Med team mens kollegiet på for eksempel 9.trinn, mens hovedtrinn er kollegiet på ungdomsskole.

utfordringer elever erfarer og opplever. Ved å studere tabellen ser man at informantenes svar samsvarer med de fleste av punktene i rammeverket, basert på en modell fra Finger og Houget (2009) i Sentance et al., (2017, s. 488). Utfordringene som står mest frem er «egen kunnskap», «mangel på støtte fra ledelse» og «elevenes manglende matematiske evner», som samsvarer med diskusjonen tidligere i kapittel 5.

5.3 Holdninger

Kap 5.3.1 viser kun informantenes erfaringer med læreplanene. Disse er kun med for å vise en dimensjon ved informantene.

5.3.1 Læreplaner

Anne er delt, men synes det er ryddig med årsvis inndeling av kompetansemålene. Synes også det er godt at den manuelle konstruksjonen er borte til fordel for andre mål. Slik at digitale ressurser kan brukes, i stedet for å bruke tid på manuelle tidkrevende operasjoner. Liker progresjonen 8,9, 10.

Mari registrerer færre kompetansemål og at programmering og personlig økonomi har fått plass. Liker tanken på årlige kompetansemål dersom man kun skal undervise et år. Konkretiserte forventninger til innhold.

Arne har mange delte tanker om at kompetansemålene er delt inn i pr år for matematikkfaget. Han mener det er for tidlig å konkludere. Synes LK20 er tettere knyttet til den generelle delen enn før. Legger vekt på at faget er åpnet opp og gir et større handlingsrom enn tidligere. Ny læreplan legger føringer på elevenes progresjon, og fører til fortetting av stoffet. Læreplanen er mer styrt på når ting skal skje og det gir trangere spillerom. Årsvis progresjon gir ryddighet, men Arne likevel uttrykker bekymring for hva som skjer dersom en elev ikke når et kompetansemål i et gitt årstrinn.

Ellen påpeker at ved LK20 så er det blitt færre kompetansemål pr år, men totalt sett flere. Hun synes dette er krevende. Etterslep fra kompetansemål fra barneskolen, som også Lise nevner, er en utfordring.
En lengre dialog mellom intervjuer og Lise ligger som vedlegg 2. Essensen i denne refleksjonen fra informanten ligger i utfordringene med at morgendagens 8, 9 og 10. klasser ikke har blitt eksponert

for kompetansemålene (Utdanningsdirektoratet, 2020b) som er på for eksempel 4, 5, 6 trinn i matematikk som omhandler programmering.

5.3.2 Holdninger til programmering

Anne og Mari er positive til programmering: «veldig positiv [..] kjempefint at det kommer in» og «Positiv [..] kunne drevet bare med det». Dette er i tråd med (Mozelius et al., 2019, s. 701).

«[…] det har en helt naturlig plass. […] programmering er viktig for å kan være et godt verktøy for å hjelpe elever med å forstå». Arnes utsagn kan sammenlignes med Paperts argument: «the Turtle's special ability to serve as a first representative of formal mathematics» (S. Papert, 1980, s. 56).

Lise er skeptisk til programmering: «ikke så lett å få det til alltid […] skulle også hatt mye lenger tid». Mozelius et al. (2019, s. 701) nevner tids og kompetanseheving som utfordringer, og det er i tråd med undertonen i Lises utsagn.

«Jeg synes jo at det er bra at det har kommet inn, men siden jeg på en måte føler meg litt usikker, så synes jeg det er litt sånn ekkelt å kunne skulle undervise det også». Dette utsagnet fra Ellen er i samsvar med: «this is with the condition that teachers will get professional training in programming and relevant information on how programming should be implemented in the syllabus» (Mozelius et al., 2019, s. 705)

I løpet av intervjuene så gir også Arne, Lise og Ellen i større grad inntrykk av å ha brukt programmering i undervisningen enn Anne og Mari. Dette kan være et uttrykk for at Lise og Ellen har erfart utfordringene, som Anne og Mari ikke enda har erfart.

5.4 Begrep

Under alle intervjuene har begrepene algoritmisk tenkning og programmering blitt brukt. Innholdet i algoritmisk tenkning har blitt dekomponert til sine betydninger med tanke på definisjonen til Udir (Utdanningsdirektoratet, 2019a). Algoritmisk tenkning som begrep og de påfølgende begrepene:

1. Koding og programmering
2. Programmering
3. To kompetansemål

4. Dybdelæring
5. Variabel, løkke, betingelse/vilkår og funksjon
6. Utforske

Blir satt i system basert på hvordan begrepet forklares av informanten, og hvor presis oppfattelsen av begrepet fremstår. En tabell som oppsummerer, vil presenteres til slutt i del-kapittelet. Sammenligningen skjer med begrepsavklaringen i kapittel 1.

5.4.1 Koding og programmering

Anne forklarer at koding er å få noe til å gjøre noe. «Hvisom atte dersom atte» sier hun. Koding oppleves som «Positivt, litt mer ufarlig, programmering er et større begrep». Mari og Arne ser også koding som positivt. Arne og Maris samsvarer videre på at: «Koding er en del av programmering», men «Programmering er et tyngre ord enn koding» mener Mari, i motsetning til Arne som mener programmering er positivt ladet. Arne forteller: «Programmering er selve handlingen med å skrive et program. Koding er mer fysisk». Se figur 1 -1 for utdyping. Lises utsagn: «Det er vel å gjennomføre og lage en kode i et av programmeringsspråkene» har også definert en forskjell på de to begrepene.

I lys av Anne, Mari, Arne og Lises forklaringer kommer det frem at Mari og Arne har de mest definerte forståelsene av koding og programmering. Alle fire er likevel på sporet av at det er en forskjell, der koding er mer fysisk inntasting, mens programmering sees på som en prosess (Lodi & Martini, 2021, s. 886).

5.4.2 Programmering

Anne forklarer programmering: «Få noe til å gjøre noe «, og er på linje med Mari: «Jeg tenker på koding. […] kode ting av noe eller programmering av en gjenstand. Kommandoer». Informantene forklarer seg enkelt om programmering. De har en enkel oppfatning, men skiller likevel på begrepene programmering og koding. Arne går litt videre: ««Hvis man har programmert så har man skrevet noe fullt ut. Et fullstendig dataprogram». Det er muligheter for kontraster her for Lodi og Martini (2021, s. 886) påpeker forskjeller mellom begrepene koding og programmering (Lodi & Martini, 2021, s. 886), men Balanskat og Engelhardt (2015, s. 27) påpeker at i flere land bruker

koding og programmering om hverandre (Balanskat & Engelhardt, 2015, s. 27). Dette kan skape forvirring for elevene og man bør være konsekvent på hvordan man bruker begrepene.

5.4.3 Dybdelæring

Definisjon av dybdelæring hos Utdanningsdirektoratet (Udir) finnes i figur 8, samt en dekomposisjon for utdyping:

> Anne forklarer dybdelæring som noe å lære mer om.
> «Lære mer om. […] metoder for å komme fram til et svar eller lære mer om et emne […] Bøker personer […] skaffe seg informasjon […]. Lære mer om et emne egentlig? Hvis du går i dybden på noe så. Ja, så vil mye av det være overførbart slik at du vil se bredden i ting» (Anne).

Anne forklarer seg her om dybdelæring og peker på overføring fra definisjonen til Udir. Anne nevner bredden, men i kontekst dybdelæring så er det nærliggende å anta at Anne mener overføring eller transfer av kunnskap.

> «[…] at programmering blir en form for dybdelæring, sånn at hvis du bruker det du har lært […] om geometri, for eksempel. Bruker algoritmer for noe du kan. Du kan for å programmere noe, så er jo det en form for dybdelæring innen emnet […] i motsetning til overflatelæring som er litt av alt.» (Mari)

> «Altså dybdelæring tenker jeg, at da må du kunne komme løse et problem på ved å kunne bruke mange eller flere ulike matematiske metoder» (Lise)

Maris forklaring på dybdelæring er vag, men gir likevel tanker som går i retning metoder når hun nevner algoritmer, og programmering er en metode. Hun setter også opp en motsetning iom at hun nevner overflatelæring. Lise har metoder med i sin forklaring av dybdelæring, men viser ikke mer kunnskap om dybdelæring. Dybdelæring og overflatelæring -se Sawyer (2006, s. 4).

> «[…] dybdelæring kommer til uttrykk i det at vi møter problemer i et mangfold av forskjellige måter […] vi ser på forskjellige problemer, men også at vi ser på samme problem med forskjellige metoder og da programmering kan jo være et eksempel, en strategi, der vi kan bruke et program for å løse et stort problem. Men vi kunne jo også brukt andre strategier, for eksempel fått fram statistikk ved å hente data på. Ja, sa jeg, og jeg tenker jo at dybdelæring også i det så ligger det for meg at elevene ser også sammenheng, ikke bare altså innad i faget at de ser det at algebra og programmering henger sammen for eksempel.» (Arne)[9]

Arnes forklaring på dybdelæring er i tråd med Udirs definisjon når vi ser på de uthevede ordene i sitatet over. Bruken av disse ordene korrelerer i stor grad med dekomposisjonen under, som igjen er tatt fra Kap. 2.5 som omhandler definisjonene til Udir og Ludvigsenutvalget (Kunnskapsdepartementet, 2019; NOU 2015:8, 2015, s. 14):

1. Metoder
2. Sammenheng
3. På ulike måter
4. Utvikle kunnskap

Som en kontrast til informantenes forståelse av dybdelæring, så nevner de ikke samarbeid, refleksjon over egen læring eller forståelse av begreper. Det kan tyde på at dybdelæring er et begrep som ikke er fullstendig definert hos den enkelte informant.

5.4.4 Variabel, løkke, betingelse/vilkår og funksjon

> «[…] variabel noe som kan endres […] variabel tall for eksempel kan være en variabel», «Får noe til å gå igjen og igjen? (informanten tegner ring i lufta). Ja ja løkke»
> «vilkår – 'dersom atte hvisom atte'» «nå gikk jo jeg rett til graf funksjon. Funksjon, noe som får noe til å gjøre noe» (Anne)

[9] Min utheving for lesbarhet

Annes forklaringer har et klart muntlig drag over seg, det mangler de rette begrepene på variabler, gjentagelser, vilkår og funksjoner. Selv om det er uklarhet i begrepene så har hun en formening om hva begrepene omtaler.

Mari forklarer begrepene kort: Noe som varierer. Gjentagelse. Noe må skjer for at noe annet skal skje.

> «hvis et eller annet, så skal dette skje, og hvis ikke, så skal dette skje for eksempel. I blokk-kode er alle en funksjon. Lage nye blokker» (Mari)

> «Variabel er en plassholder. I stedet for et tall. Kan variere. Kan også være tekst. En holder av informasjon. En informasjonsholder. Løkke gjør ting om igjen. Hvis a så b, hvis b så a. Noe gjennomføres betinget av noe annet. Funksjon er et delprogram, kan hentes fra under visse betingelser. Hentes frem når det er behov» (Arne)

Både Mari og Arne har korte klare definisjoner på begrepene. Forklaringene er ikke utdypende, men inneholder korte definisjoner som en som er kyndig med begrepene vil forstå, men utenforstående vil ha problemer med.

> «Variabel er noe som endrer seg, noe som er variabelt, men en variabel kan jo være så mangt, men oftest så er det et tall.» (Lise)

> «Kan det være tekst og?» (Intervjuer)

> «I programmering så kan det.» (Lise)

> «Det er en gjentagende hendelse. Så har du en for-løkke som går så mange ganger som du tenker at den skal gå» (Lise)

> «Ja, så hva skjer hvis?» (Lise)

I denne delen av intervjuet innehar informanten kunnskapen, men den kommer ikke uten at man ber om den. Dette kan tyde på at informanten (Lise) har begrepsforståelse, men mangler struktur på kunnskapen.

> «Bruker en variabel, der så er det lett å bruke den videre. I koden din. Men for å bruke tall videre i utregning, for eksempel i matta. Hvis man har en algebra, for eksempel, da så bruker de jo mye variabler i?» (Ellen)

> «Løkke, det tror jeg skal bli lettere. Det er jo hvis man skal gjøre en ting flere ganger da gjentagende prosess, så er det blir lettere å bruke i en løkke mindre tekst og skrive og. Ja gjentagende prosess.» (Ellen)

> «Altså man legger jo en betingelse på, at hvis jeg trykker hvis brukeren skriver ja.
> Så skal det skje og hvis brukeren sier nei, så skal noe annet skje. På en måte. Ja at man legger betingelser for hvordan koden skal kjøre videre da.» (Ellen)

Her viser Ellen at å definere variabler er vanskelig, i likhet med Lise, mens løkker er Lise og Ellen ganske klare på. Å definere betingelser er vanskeligere, men det er en viss forståelse til stede hos begge. Å definere funksjoner i kontekst av programmering viser seg å være vanskelig for alle informantene.

5.4.5 Utforske

Anne bruker ikke ordet, Mari en gang, mens Arne 9 ganger, Lise 3 ganger og Ellen 7 ganger i løpet av intervjuene. Utforske er et sentralt begrep i bruk av programmering, og burde ligget mer opp i dagen for informantene. Utforske er en sentral del av «den algoritmiske tenkeren». Fikle er arbeidsmåten, og det er en viktig del av det som sees på som «21st century skills»:

> «[…] asking schools to develop skills such as
> problem solving, critical thinking, communication, collaboration, and selfmanagement—often referred to as “21st century skills.” »

(NOU 2015:8, 2015, s. 31, 17–38; Pellegrino & Hilton, 2012, s. 1)

I Ludvigsensutvalget anbefaling av fagene legger de flere kompetanseområder som grunnlag for fornyelse, hvor av det ene lyder slik: «kompetanse i å utforske og skape» (NOU 2015:8, 2015, s. 36). I tillegg er det første kjerneelementet i matematikk: «Utforsking og problemløysing», og inneholder beskrivelser av begrep og arbeidsmåter man finner igjen i «den algoritmiske tenkeren» (NOU 2015:8, 2015, s. 22; Utdanningsdirektoratet, 2019a, 2020a). Økt fokus på å utforske som begrep og metode hos elevene er blitt tillagt stor betydning av styringsdokumenter, og dermed bør det være økt fokus på akkurat utforsking i matematikkfaget. «Algoritmisk tenkning er en problemløsningsmetode» (Utdanningsdirektoratet, 2019a).

5.4.6 Algoritmisk tenkning (AT)

Her viser jeg frem kondenserte sammenskrivinger av Anne, Mari og Arnes erfaringer, mens med Lise og Ellen ser jeg mer på sitatene, og hvilke ord de faktisk bruker, og om de kan tolkes til Udirs nøkkeord og begreper innen AT.

Anne viser til steg for steg. Hva er problemet? Relevant informasjon. Gå igjennom oppgavene. Trene på problemløsning.

Mari forklarer at algoritmisk tenkning minner mye om programmering. Problemløsning. Algoritmisk tenkning er veldig likt hvordan programmering er. Logiske algoritmer og mønstre.

Arne forklarer at han er glad i systemer og systemtenkning. Bruker prinsippene i andre fag slik som naturfag og engelsk. «at du ser på noe og så prøver du å se på alle de tingene som påvirker den tingen du ser på og så bryte ned». Ting påvirker hverandre, så han prøver å dele opp et begrep i mange underbegreper. Deler opp matematikkoppgaver i delproblemer. Programmering bruker oppdeling.

> «[…] skal lære dem både kritisk tenkning og vurdering og modellering og fikling og sånn. (Lise)

> Algoritmisk tenkning er jo egentlig veldig sammensatt og **algoritmer**. Det er jo **gjentagende handlinger** […] Mange kan den grunnleggende matematikken for dårlig, så de har for lite å bygge på rett og slett for å løse sammensatte problemer. (Lise)
>
> Noen vil komme til å sitte og bygge med lego klossene hele tiden. Andre vil kunne bygge med legoklossene og forklare hva som skjer, mens andre vil da kunne **utvikle** det enda mere til å både bygge, **forklare**, **definere**, regnet på det og utvikle da matematiske sammenhenger og de matematiske sammenhengene eller **problemløsninger**. (Lise)
>
> «[…] begynner i det små og skal utvikle og så **teste** ut funke av hypotesen, funke han ikke **videreutvikle** og se på kritisk tenkning og prøve å bygge dette her opp til en matematisk forståelse […] For å finne figurtall nummer 100 og figurtall nummer n, og se systemet. Hva har skjedd? Hvordan har dette tallet utviklet seg for eksempel? Og så lage da en regel ut av det som. (Lise)

De fire ovenstående sitatene fra Lise viser hvordan hun har mange av begrepene innenfor algoritmisk tenkning på plass i måten hun ordlegger seg på og bruker nøkkelord som er fra den algoritmiske tenkeren fra Utdanningsdirektoratet (2019a). Lises ord er vurdert opp mot definisjonen på algoritmisk tenking, vurdert mot definisjonene på CT på Barefoot Computing (Barefoot Computing, 2022b). Vi ser i tabellen under at Lises ord sammenfaller i stor grad med Udir sine begrep.

Lises ord (som jeg har uthevet) er:

Tabell 4 Lises begreper vs. Udir

Lise	Utdanningsdirektoratets AT
Kritisk tenkning	Evaluering
Vurdering	Evaluering
Problemløsning	Problemløsning – som er kjernen i AT
Modellering	Abstraksjon, Mønstre
Fikling	Fikle – utforske og eksperimentere

Forklare	Samarbeide – dele og jobbe sammen
Definere	Samarbeide – dele og jobbe sammen
Algoritmer	Algoritmer – Regler og steg-for-steg
Utvikle	Skape – designe og lage

«Det er jo en logisk tenkemåte» (Ellen)

«Matte språket er jo et eget språk. Jeg tenker at matte språket og den logiske måten å skrive koder på.» (Ellen)

«snu litt på den der gammeldagse» (Ellen)

«Det skal være sånn og sånn og sånn […] i programmering sånne regler på en måte, […]» (Ellen)

«Lærer en litt annen måte å tenke på.» (Ellen)

«Det er egentlig utforskende, jeg føler algoritmisk tenkning er.»

Tabell 5 Ellens begreper vs. Udir

Ellen	Utdanningsdirektoratets AT
Logisk	Logikk – analysere og forutse
Programmering sånne regler	Algoritmer – Regler og steg-for-steg
Annen måte å tenke på	Problemløsning – som er kjernen i AT
Utforske	Fikle – utforske og eksperimentere
sånn og sånn og sånn	Algoritmer – Regler og steg-for-steg

Ellens utsagn er vagere og mer upresise enn Lises, men det ligger en forståelse i bunnen her. Selv om det ikke er like presist som Lise. Lise har lang fartstid i yrket, og Ellen har tatt 15stp utdanning i «Programmering for lærere». Disse to lærerne har ulike perspektiv kan det se ut til. Hvor Lise har det praktiske og generelle synet på AT og Ellen relaterer AT til programmeringen mer direkte.

Det er ingen motsetninger mellom informantene, men det er en del begrepshull her, og en litt enkel tilnærming til begrepene. Ingen nevner «den algoritmiske tenkeren» (Utdanningsdirektoratet, 2019a), som har de begrepene som er sentrale i AT. Ingen nevner samarbeid som arbeidsmåte. Se mer om dette i kapittel 5.6.

5.4.7 Resultat

Tabell 6 Begrepsforståelse

Forståelse	Lav	Middel	Høy	Note
Koding		MA, AN-, LI-, EL-	AR	
Programmering		MA, AN-, LI-, EL-	AR	
To komp.mål	AN, EL	MA	AR, LI	
Dybdelæring	AN, MA	AR		
Variabel ++	AN	MA+, LI, EL	AR-	Udir def. legges til grunn
Utforske	AN, MA	LI	AR, EL-	
Alg. Tenkn.		MA-, EL, AN, MA	LI, AR	
Nøkkel: AN = Anne, MA = Mari, AR = Arne, LI = Lise, EL = Ellen (+ eller - = svakere eller sterkere) (listen er ikke uttømmende)				

Det er verdt å merke seg at dette er en subjektiv sammenfatning av informantenes svar, vurdert opp imot definisjoner av begrepene. Dersom man legger til grunn disse resultatene ser man at det er et sprik i forståelsene på sentrale begreper. I tillegg har man forståelsen av begrepet algoritmisk tenkning som er omhandlet flere steder i oppgaven. Det kan tyde på at en mangel på

begrepsforståelse kan være en utfordring for læreres perspektiv på programmering, og hvilke områder man kan legge vekt på i kursing av lærere. Utfordringene som Tabell 3 illustrerer er i tråd med utfordringer som er støttet av Sentance og Csizmadia (2017, s. 488). Begrepsforståelse er noe læreren selv kan tilegne seg, da dette er en indre utfordring som læreren selv kan løse. I motsetning til ytre utfordringer hvor ikke læreren selv har kontroll (Sentance & Csizmadia, 2017, s. 488). PfDk rammeverk viser til at læreren «forstår hvordan den digitale utviklingen utvider og forandrer fagets innhold, begrepsapparat, vurderingsformer og arbeidsmetoder» (Kelentrić et al., 2017, s. 7). God forståelse for begreper ligger også i grunntanken til rammeverket TPACK der «Content knowledge» (CK) er kritisk for lærere, slik at det ikke oppstår misoppfatninger og feil-informasjon (Mishra et al., 2009, s. 63).

5.5 Programmeringsdidaktikk

5.5.1 Organisering av klasserommet

Anne og Mari er kontraster til hverandre: «[…] organiserer elevene to og to […] at eleven skal hjelpe hverandre», mens Mari nevner at sine elever ønsker å pusle alene. Arne er konkret på når han benytter 2-og-2: «elevene jobber selvstendig når det er små biter av et program som skal kodes og når de skal feilsøke, og teste eget program når de får feilmeldinger […] i større sammenhenger jobber vi ofte i grupper og 2-og-2». Arne påpeker at generelt så liker elevene å jobbe 2-og-2 i matematikk.

Lise forteller at: «[…] ofte så synes jeg det kan være lurt å ha litt gjennomgang, og så kan elevene sitte og prøve littegranne selv på[10] […]», deretter på individuelt og så grupper. Lise er skeptisk til 2-og-2 da den ene ofte ikke gjør noe. Ellen viser til en nyanse der hennes elever ønsker å kode på egen pc, men snakker sammen når de er usikre.

Informantene legger i liten grad vekt på par-programmering som beskrevet i kap 2.3.5. Par-programmering kan vise seg nyttig og effektfullt i henhold til Hanks (2006) i (Hanks et al., 2011, s. 147–148). Økt motivasjon, fokus, oppmuntring, bedre karakterer, høyere interesse i faget og utholdenhet er noen av funnene i et forskningsprosjekt som så på par-programmering og jenter i

[10] Prøve littegranne selv på = fikle

«secondary school». Utfordringene med par-programmering var «erfaring, kunnskap, personlighet og dedikasjon» (Hanks et al., 2011, s. 153–154; Liebenberg et al., 2012, s. 230–231)

Informantenes erfaringer viser at parprogrammering er i stor grad en fysisk organisering av klasserommet og ikke en pedagogisk tilrettelegging. Her vil jeg si at både den pedagogiske delen og den begrepsmessige delen, som tilhører kunnskap om kontekst, i TPACK-modellen, mangler (Mishra et al., 2009, s. 63). Par-programmering kan være fruktbart, men det krever mer enn bare fysisk organisering av klasserommet. En ny kultur må på plass, og en endring i begrepsforståelse og kompetanse i henhold til PfDK i (Kelentrić et al., 2017, s. 4–6). Parprogrammering er i tråd med sosiokulturelle læringsteori ved Vygotsky. Det resonnerer også med dybdelæring, som sier «lære seg nye begreper og metoder alene og sammen med andre». I algoritmisk tenkning så handler det om å dele og jobbe sammen. PfDK har et eget punkt som heter «samhandling og kommunikasjon» (Kelentrić et al., 2017; Kunnskapsdepartementet, 2019; Utdanningsdirektoratet, 2019a; Vygotski et al., 1978).

5.5.2 Transfer

Arne forteller at erfaringen hans tilsier at det (transfer) er vanskelig for elevene. Men ved veiledning så klarer de det.

> «Jeg bruker algoritmisk tenkning i flere fag og når jeg eksplisitt sier at de kan bruke det (AT) som strategi så opplever jeg at elevene mestrer AT, men de får det ikke til på egen hånd» (Arne).

Har erfaringer med algebra innlæring i klasse. Hvorav en gruppe brukte mye programmering, og en annen brukte lite eller ingenting. Erfaringen, fra en papir-prøve i algebra, er at gruppa som brukte programmering sitter igjen med like mye kunnskap som de som lærte algebra med papir og blyant.

> «Og det viser seg det at den gruppa som lærer algebra via programmering sitter igjen med like mye kunnskap på en, sånn papir-prøve etterpå, som de som har lært det ved å bruke papir og blyant i» (Arne).

Arnes erfaringer med transfer samsvarer med resultater fra forskning, og redegjøringen av transfer i kapittel 2.4. Hvor Guzdial (2016, s. 39-40, 49-50), viser til ingen eller få effekter av transfer til det virkelige liv. Guzdial (2016) påpeker også at den eneste måten man kan oppnå transfer er å undervise for å oppnå transfer (Weintrop & Wilensky, 2018, s. 18-19, 90). Dette synet støttes av Benton et al. (2017, s. 118), men står i kontrast til kontrast til det fokus som finnes i (Grover & Pea, 2013, s. 41–42; Utdanningsdirektoratet, 2019a; J. Wing, 2017, s. 9; J. M. Wing, 2006, s. 35). Udirs definisjon av dybdelæring (Kunnskapsdepartementet, 2019) nevnt i kapittel 2.5 inneholder nettopp et ønske om overføring/transfer.

Oppsummert er erfaringen at det er tvetydige erfaringer med det (transfereffekter), og at det ikke har negative føringer for hva elevene sitter igjen med læringsutbytte.

5.6 Et uventet funn

Et sentralt funn er lærernes ønsker om kursing på skolenivå. Lærerne trekker frem at de ønsker å samhandle, utveksle ideer, delingskultur, samarbeid mellom lærere og samarbeid på team og skole. Når man sammenstiller dette ønsket med lærernes begrepsforståelse om dybdelæring og algoritmisk tenkning, så er samarbeid som begrep ikke nevnt i det hele tatt. Dette kan tyde på at et sosiokulturelt læringssyn ikke er klart definert hos lærerne når de ser på elevene, men de ønsker en sosiokulturell tilnærming til egen kompetanseheving. Par-programmering er også et område som er i tråd med sosiokulturelt læringssyn (Sentance et al., 2019, s. 142–147), men lite vektlagt av lærerne. Se kapittel 2.3.4.5 og 5.5.1 for mer om par-programmering, samt artikkelen til Sentance et al. (2019) om PRIMM i et sosiokulturelt perspektiv.

5.7 Oppsummering

I dette kapittelet har funnene blitt analysert og diskutert. Det er blitt sett på 5 hovedområder:

1. Lærernes utdanning og programmeringsutdanning/kursing
2. Holdninger og erfaringer til læreplanen
3. Utfordringer og erfaringer
4. Begrepsforståelse
5. Programmeringsdidaktikk

Funnene i datamaterialet som fremtrer som de mest signifikante er lærernes utfordringer med å få tid til å oppdatere seg og få nok kompetanse innen programmeringsmetodikk og programmeringsteknisk kompetanse (punkt 3). Lærerne peker på liten tid satt av til egenutvikling av kompetanse, men peker også på manglende støtte fra ledelsen/skole til kursing og kompetanseheving. (punkt 3). Lærernes begrepsforståelse er blitt gjenstand for undersøkelser, og har avdekket varierende grad av forståelse av sentrale begrep innenfor programmering og algoritmisk tenkning (punkt 4).

Det er gjort interessante funn innenfor punkt 2, som omhandler lærernes syn på ny læreplan (LK20), hvor lærerne er skeptiske til at kompetansemålene i matematikk er per år, og at det fører til en mer rigid undervisning.

I punkt 5, som omhandler organisering av klasserommet, så har jeg prøvd å avdekke erfaringer fra par-programmering, og lærernes erfaringer er delte og ikke konkluderende. Par-programmering er også et område som er i tråd med sosiokulturelt læringssyn, men lite vektlagt av lærerne. Det kan se ut til at for å oppnå positive resultater med par-programmering, så må kompetansen til den enkelte lærer opp på et høyere nivå innen både programmering, men også et økt fokus på metodikken bak par-programmering. Hvis man ser på nivå på programmeringskompetanse så ser den ut til å være høyere hos de lærerne som bruker større grad av par-programmering.

6 Konklusjon

Problemsstillingen:
«Læreres utfordringer og erfaringer med innføring av programmering i matematikk i forbindelse med fagfornyelsen».

Det er viktig å påpeke at med 5 informanter vil det være vanskelig å generalisere på funn og resultater, men funn i denne studien er støttet av forskningsprosjekter med større antall informanter, slik at i beste fall vil min studie gi en pekepinn i en retning videre forskning.

6.1 Konklusjon

Lærernes bakgrunn og utdannelse skaper en ramme for å kunne forstå forutsetningene. Av de fem informantene så har tre av de middels til høy kompetanse i programmering, to hadde lav kompetanse. Alle informantene har i varierende grad benyttet seg av egen kursing, korte kurs og kollega-samarbeid. Lærernes forskjellige tilnærming til kompetanseheving, er dermed et grunnlag på hvordan undervisningen blir i matematikkfaget med kontekst programmering.

Lærerne i studiet har en positiv holdning til at programmering er med i faget, men de viser også en skepsis til at kompetansemålene er per år, og er bekymret for at det kan bli stofftrengsel. Lærerne anerkjenner programmering og algoritmisk tenkning som viktige verktøy, men det avdekkes også at begrepsforståelsen, på sentrale begrep, innen programmering og algoritmisk tenkning er upresis og usikker. Lærerne er opptatt at kursing og workshops skal ha et sosiokulturelt preg med samarbeid mellom lærere, men nevner i liten grad samarbeid som viktige elementer i programmering, algoritmisk tenkning og dybdelæring.

Tid er en begrensende faktor for å ta i bruk programmering i matematikkfaget. Tid til egenutvikling av kompetanse, tid til å ta kurs, tid til å samhandle med kolleger og tid til å undervise andre kolleger. Dette studiet viser at tid er faktoren som begrenser mulighetene til kompetanseheving på nivåer fra egen, team, skole og utdanning. Systematisk tilretteleggelegging av tidsressurser til kompetanseheving er viktig.

Det bør iverksettes en helhetlig planlegging av kompetansehevingstiltak ved den enkelte skole, basert på teoretiske rammeverk for å sikre lærernes forventede kompetansenivå innen digital teknologi. Lærernes holdninger til programmering og algoritmisk tenkning bør styrkes gjennom økt fokus på hva som kreves av en «digital» lærer i grunnskolen. Det bør legges vekt på forskningsbaserte tilnærminger i undervisning i tråd med teoretiske rammeverk som PfDK og TPACK. Lærerne må få tid til egenutvikling og kompetanseheving.

6.2 Videre forskning

I min søken i artikler, bøker, kilder og på internett, så finner jeg at fagdidaktiske bøker innen programmering fraværende i stor grad. Lærerne i studiet etterlyser mer ressurser innen didaktikk.

En forskning basert på (Weintrop & Wilensky, 2018), og ser på transfereffekter mellom blokk-kode og tekstbasert kode, der man tar utgangspunkt i norske forhold, ville være et interessant prosjekt.

«Når en lærer forhandler med elevene om et eksisterende objekt i form av en programmerings oppgave, og endringen består i å forandre på det matematiske innholdet, så er fokus flyttet over fra selve programmeringen og til at matematikken blir fokus. Da kan vi si at det har skjedd en transformativ prosess, og at matematikken er blitt fokus for læringen» basert på forståelse fra et doktorgradsarbeide av Sanna Erika Forsström (2020) i (Forsstrøm, Sanna Erika, 2020, s. 114).

Hvordan et undervisningsopplegg, satt i en sosiokulturell kontekst, basert på rammeverk slik som PRIMM (Sentance et al., 2019, s. 147), tankene bak par-programmering (Gillis, 2021; Guzdial, 2016, s. 87; Hanks et al., 2011, s. 135–136; Liebenberg et al., 2012, s. 232), og bruk av modellene TPACK (Kelentrić et al., 2017, s. 5–6; Mishra et al., 2009, s. 60–62) og PfDk som overordnede styringsverktøy. Samt se på hvordan Sanna Erika Forsström sin bruk av aktivitetsmodellen (Forsstrøm, Sanna Erika, 2020, s. 108) kan brukes for å bruke programmering som et verktøy for at elever skal lære matematikk.

Referanser/litteraturliste

Andersen, Viggo Trummar. (2020). *Forskningskisse* [Forskningsskisse - eget arbeid levert USN H2020]. USN.

Anker, T. (2020). *Analyse i praksis: En håndbok for masterstudenter* (1. utgave, 1. opplag.). Cappelen Damm akademisk.

APAC. (2022). *Higher Education institution (HEI) | Glossary | APAC*. https://academiccareermaps.org/glossary/higher-education-institution-hei

Archibald, M. M., Ambagtsheer, R. C., Casey, M. G., & Lawless, M. (2019). Using Zoom Videoconferencing for Qualitative Data Collection: Perceptions and Experiences of Researchers and Participants. *International Journal of Qualitative Methods*, *18*, 1609406919874596. https://doi.org/10.1177/1609406919874596

Aspefalten, Bøen, Halvorsen, Myrland, Berg, & Aagard. (2022). *LUDO I Programmering*. Ludo. https://www.ludo.usn.no/programmering

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet.

Barefoot Computing. (2022a). *Barefoot Computing helping primary school teachers*. Barefoot. https://www.barefootcomputing.org/about-barefoot

Barefoot Computing. (2022b). *Concepts and Approaches Category Master Page*. Barefoot. https://www.barefootcomputing.org/concepts-and-approaches/concepts-approach-category-page

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, *3*(2), 115–138.

Berding, F., Rolf-Wittlake, K., & Buschenlange, J. (2017). Impact of Different Levels of Epistemic Beliefs on Learning Processes and Outcomes in Vocational Education and Training. *World Journal of Education*, *7*(3), 103–114.

Bjørnsrud, H. (2014). *Den inkluderende fellesskolen: Læringskraft for elever og lærere?* Gyldendal akademisk.

Bjørnsrud, H. (2015). *Skolebasert kompetanseutvikling: Organisasjonslæring for delingskultur*. Gyldendal akademisk.

Blikstad-Balas, M. (2017). Key challenges of using video when investigating social practices in education: Contextualization, magnification, and representation. I *International Journal of Research & Method in Education* (Nr. 5; Bd. 40, Nummer 5, s. 511–523). Routledge.

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education. Report prepared for the Nordic@BETT2018 Steering Group*. National Research Council of Italy, Institute for Educational Technology. https://doi.org/10.17471/54007

Bolstad, B. (2020). *Dybdelæring og tverrfaglighet* (1. utgave.). Pedlex.

Bonasio, A. (2018, juni 26). Expert View: Watson's Five Computers - Cloud and the Mainframe. *Tech Trends*. https://techtrends.tech/tech-trends/expert-view-watsons-five-computers/

Bruner, J. (1985). Vygotsky: A historical and conceptual perspective. *Culture, communication, and cognition: Vygotskian perspectives*, *21*, 34.

Bruner, J. S., Wood, D., & Ross, G. (1976). The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, *17*(2). https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Bryman, A. (2016). *Social research methods* (5. utg.). Oxford university press.

Bueie, H. (2019). *Programmering for matematikklærere*. Universitetsforlaget.

code.org. (2022). *Check out what I made*. Code.Org. https://studio.code.org/s/artist/lessons/1/levels/4

DEKOM. (2022). *Om – Dekom*. https://dekom.no/om/

Eidsvig, P.-E. (2020, desember 20). *Lærerstudentenes erfaring med tekstprogrammeringsspråket Python*. https://www.ludo.usn.no/post/l%C3%A6rerstudentenes-erfaring-med-tekstprogrammeringsspr%C3%A5ket-python

Forsström, S. E., & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18–32. https://doi.org/10.26803/ijlter.17.12.2

Forsstrøm, Sanna Erika. (2020). *Doing Mathematics with Robots—An Activity Theoretical Perspective on the Links between Mathematics and Programming in Classroom Activities)* [Western Norway University of Applied Sciences]. https://www.hvl.no/contentassets/904bb8c924ea4237ae23bf3fe6c94e87/thesis-forsstrom-280920.pdf

Fraser, N. (2022). *Code Generators*. Google Developers. https://developers.google.com/blockly/installation/code-generators

Fullan, M. G. (1994). Theories of systemic reform. I *Hentet Mai 21, 2020 fra Systemic Reform—Perspectives on Personalizing Education*. https://www2.ed.gov/pubs/EdReformStudies/SysReforms/fullan1.html

Gaddis, T. (2019). *Starting out with Python* (4. utg.). Pearson.

Gillis, A. (2021, juni). *What is Pair Programming?* SearchSoftwareQuality. https://www.techtarget.com/searchsoftwarequality/definition/Pair-programming

Gomes, A., & Mendes, A. (2014). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1–8.

Google Developers. (2018). *Introduction to Blockly | Google Developers*. https://developers.google.com/blockly/guides/overview

Grandell, L., Peltomäki, M., Back, R.-J., & Salakoski, T. (2006). Why complicate things? Introducing programming in high school using Python. *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 71–80.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, *42*(1), 38–43.

Guzdial, M. (2016). *Learner-centered design of computing education: Research on computing for everyone: Bd. Number 33*. Morgan & Claypool Publishers.

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education*, *21*(2), 135–173. https://doi.org/10.1080/08993408.2011.579808

Hannay, J. E., Dybå, T., Arisholm, E., & Sjøberg, D. I. K. (2009). The effectiveness of pair programming: A meta-analysis. *Information and Software Technology*, *51*(7), 1110–1122. https://doi.org/10.1016/j.infsof.2009.02.001

Haraldsrud, A. D., Sveinsson, H., & Løvold, H. (2020a). *Programmering i skolen*. Universitetsforlaget.

Haraldsrud, A. D., Sveinsson, H., & Løvold, H. (2020b). *Programmering i skolen*. Universitetsforlaget.

Hovde, K.-O., & Grønmo, S. (2020). Algoritme. I *Store norske leksikon*. http://snl.no/algoritme

HP.com. (2018, oktober). *Computer History: A Timeline of Computer Programming Languages | HP® Tech Takes*. Computer History: A Timeline of Computer Programming Languages. https://www.hp.com/us-en/shop/tech-takes/computer-history-programming-languages

Høgheim, S. (2020). *Masteroppgaven i GLU* (1. utgave.). Fagbokforlaget.

Kelentrić, M., Helland, K., & Arstorp, A.-T. (2017). Rammeverk for lærerens profesjonsfaglige digitale kompetanse. *Senter for IKT i utdanningen.* https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/rammeverk-larerens-profesjonsfaglige-digitale-komp/

Kellner, B. C. (2022). *The Bernoulli Number Page*. The Bernoulli Number page. https://www.bernoulli.org/

Kleven, T. A. (2018). *Innføring i pedagogisk forskningsmetode: En hjelp til kritisk tolking og vurdering* (F. Hjardemaal, Red.; 3. utg., s. 225 s.). Fagbokforl.

Kleven, T. A., Hjardemaal, F., & Tveit, K. (2018). *Innføring i pedagogisk forskningsmetode: En hjelp til kritisk tolkning og vurdering* (3 utg.). Fagbokforlaget.

Kunnskapsdepartementet. (2019, mars 13). *Dybdelæring*. https://www.udir.no/laring-og-trivsel/dybdelaring/

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

Larke, L. R. (2019). Agentic neglect: Teachers as gatekeepers of England's national computing curriculum. *British Journal of Educational Technology*, *50*(3), 1137–1150.

Liebenberg, J., Mentz, E., & Breed, B. (2012). Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education*, *22*(3), 219–236. https://doi.org/10.1080/08993408.2012.713180

LKK. (2021). Om Lær Kidsa Koding. *Lær Kidsa Koding!* https://www.kidsakoder.no/om-lkk/

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883–908.

Logo. (2022). I *Wikipedia*. Wikipedia. https://en.wikipedia.org/w/index.php?title=Logo_(programming_language)&oldid=1081602441

Mannila, L. (2017). *Att undervisa i programmering i skolan: Varför, vad och hur?* (Upplaga 1.). Studentlitteratur.

Mannila, L., & Nordén, L.-Å. (2020). *Att undervisa textbaserad programmering i skolan* (Upplaga 1.). Studentlitteratur.

Maxwell, J. A. (2013a). *Qualitative research design: An interactive approach* (3rd ed., Bd. 41, s. XI, 218 s.). Sage.

Maxwell, J. A. (2013b). *Qualitative research design: An interactive approach* (3rd ed., Bd. 41, s. XI, 218 s.). Sage.

Meld. St. 20. (2012). *Meld. St. 20 (2012–2013)*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/meld-st-20-20122013/id717308/

Mellin-Olsen, S. (1996). *Samtalen som forskningsmetode: Tekster om kvalitiv [i.e. Kvalitativ] forskningsmetode som del av pedagogisk virksomhet*. Caspar forlag.

Microbit.org. (2022). Micro Bit. I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Micro_Bit&oldid=1077802819

Mishra, P., Koehler, M. J., & Harris, J. (2009, januar). *What Is Technological Pedagogical Content Knowledge?* ResearchGate. https://www.researchgate.net/publication/241616400_What_Is_Technological_Pedagogical_Content_Knowledge

MIT Scratch. (2022). *Scratch—About*. https://scratch.mit.edu/about

Mozelius, P., Ulfenborg, M., & Persson, N. (2019). Teacher attitudes towards the integration of programming in middle school mathematics. *INTED 2019*, 701–706.

Munthe, M. (2021, mars 22). *Programmeringsdidaktikk*. https://www.tekna.no/fag-og-nettverk/realfag-og-utdanning/realfagsbloggen/programmeringsdidaktikk/www.tekna.no/fag-og-nettverk/realfag-og-utdanning/realfagsbloggen/programmeringsdidaktikk/

NESH. (2021). *Forskningsetiske retningslinjer for samfunnsvitenskap, humaniora, juss og teologi* (5. utg.). De nasjonale forskningsetiske komiteene. https://www.forskningsetikk.no/globalassets/dokumenter/4-publikasjoner-som-pdf/forskningsetiske-retningslinjer-for-samfunnsvitenskap-og-humaniora.pdf

NOU 2015:8. (2015). Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/?ch=1&q

no.wikipedia.org. (2021). Personvernforordningen. I *Wikipedia*. https://no.wikipedia.org/w/index.php?title=Personvernforordningen&oldid=22079782

NSD. (2022a). *Norsk senter for forskningsdata*. NSD. https://nsd.no/

NSD. (2022b, mai 1). *Informasjon til deltakerne*. NSD. https://nsd.no/personverntjenester/fylle-ut-meldeskjema-for-personopplysninger/sjekkliste-for-informasjon-til-deltakerne

Nyeng, F. (2012). *Nøkkelbegreper i forskningsmetode og vitenskapsteori*. Fagbokforlaget.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Papert, S. (2005). You can't think about thinking without thinking about thinking about something. *Contemporary Issues in Technology and Teacher Education*, *5*(3), 366–367.

Papert, S. A. (2020). *Mindstorms: Children, Computers, And Powerful Ideas*. Hachette UK.

Pellegrino, J. W., & Hilton, M. L. (2012). *Education for Life and Work: Developing Transferable Knowledge and Skills in the 21st Century* (s. 13398). National Academies Press. https://doi.org/10.17226/13398

Postholm, M. B., Dahl, T., Engvik, G., Fjørtoft, H., Irgens, E. J., Sandvik, L. V., & Wæge, K. (2013). *En gavepakke til ungdomstrinnet? En undersøkelse av den skolebaserte kompetanseutviklingen på ungdomstrinnet i piloten 2012/2013*. https://www.udir.no/globalassets/filer/tall-og-forskning/rapporter/2013/rapport-pilot-sku.pdf

Postholm, M. B., & Jacobsen, D. I. (2018a). *Forskningsmetode for masterstudenter i lærerutdanningen*. Cappelen Damm Akademisk.

Postholm, M. B., & Jacobsen, D. I. (2018b). *Forskningsmetode for masterstudenter i lærerutdanningen*. Cappelen Damm Akademisk.

PYPL.github.io. (2022, april). *PYPL PopularitY of Programming Language index*. https://pypl.github.io/PYPL.html

Python.org. (2022). *FrontPage—Python Wiki*. https://wiki.python.org/moin/

Resnick, M. (1998). Technologies for lifelong kindergarten. *Educational technology research and development*, *46*(4), 43–55.

Roald, K. (2012). *Kvalitetsvurdering som organisasjonslæring: Når skole og skoleeigar utviklar kunnskap*. Fagbokforlaget.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., Mørken, K. M., Svorkmo, A.-G., & Voll, L. O. (2016). *Teknologi og programmering for alle*. 91.

Sawyer, R. K. (2006). The new science of learning. *The Cambridge handbook of the learning sciences*, *1*, 18.

Selvik, K. (Red.). (2016). *Programmering i skolen*. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469–495.

Sentance, S., Dorling, M., McNicol, A., & Crick, T. (2012). Grand challenges for the UK: upskilling teachers to teach computer science within the secondary curriculum. *Proceedings of the 7th workshop in primary and secondary computing education*, 82–85.

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2–3), 136–176.

Sleeman, D. (1986). The Challenges of Teaching Computer Programming. *Commun. ACM*, *29*(9), 840–841. https://doi.org/10.1145/6592.214913

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M. L., Minsky, M., Papert, A., & Silverman, B. (2020). History of Logo. *Proc. ACM Program. Lang.*, *4*(HOPL). https://doi.org/10.1145/3386329

Stenseth, B., Kaufmann, O. T., & Forsström, S. (2019). Programmering og matematikk. *Tangenten–tidsskrift for matematikkundervisning*, *30*(2), 7–12.

Stigberg, H., & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a Swedish primary school. *Policy Futures in Education*, *18*(4), 483–496.

Stray, J., Wittek, L., & Bjørnestad, I. E. (2014). *Pedagogikk—En grunnbok*. Cappelen Damm AS.

Säljö, R. (2016). *Læring—En introduksjon til perspektiver og metaforer*. Cappelen Damm Akademisk.

Säljø, R. (2016). *Læring-en introduksjon til perspektiver og metaforer. Cappelen Damm Akademisk*. Cappelen Damm Akademisk.

teknologiundervisning.dk. (2022). *Hvad er Blokprogrammering? - Teknologiundervisning 2022*. https://www.teknologiundervisning.dk/leksikon/blokprogrammering/

Texas Education Agency, T. E. (2020, januar 15). *Glossary of Acronyms*. Texas Education Agency. https://tea.texas.gov/about-tea/glossary-of-acronyms

Thrane, C. (2018). *Kvantitativ metode: En praktisk tilnærming* (s. 202 s.). Cappelen Damm akademisk.

USN. (2020a, juni 22). *Grenlandskonferansen 2021: Elevene bak tallene – LK20*. Universitetet i Sørøst-Norge. https://www.usn.no/grenlandskonferansen/grenlandskonferansen-2021-elevene-bak-tallene-lk20

USN. (2020b, oktober 13). *Desentralisert ordning for kompetanseutvikling*. Universitetet i Sørøst-Norge. https://www.usn.no/studier/videreutdanning/desentralisert-ordning-for-kompetanseutvikling/

UsNews. (2020, november 3). *Zoom Takes Lead Over Microsoft Teams as Virus Keeps Americans at Home: Apptopia*. US News & World Report. https://money.usnews.com/investing/news/articles/2020-03-31/zoom-takes-lead-over-microsoft-teams-as-coronavirus-keeps-americans-at-home

Utdanningsdirektoratet. (2019a). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). *Den teknologiske skolesekken*. https://www.udir.no/kvalitet-og-kompetanse/nasjonale-satsinger/den-teknologiske-skolesekken/

Utdanningsdirektoratet. (2020a). *Kjerneelementer—Læreplan i matematikk 1.–10. Trinn (MAT01-05)*. https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer

Utdanningsdirektoratet. (2020b). *Kompetansemål etter 6. Trinn—Læreplan i matematikk 1.–10. Trinn (MAT01-05)*. https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv21

Utdanningsdirektoratet. (2020c). *Kompetansemål etter 10. Trinn—Læreplan i matematikk 1.–10. Trinn (MAT01-05)*. https://www.udir.no/lk20/mat01-05/kompetansemaal-og-vurdering/kv14

Utdanningsdirektoratet. (2021, juni 21). *Hvorfor har vi fått nye læreplaner?* https://www.udir.no/laring-og-trivsel/lareplanverket/stotte/hvorfor-nye-lareplaner/

Utdanningsdirektoratet. (2022a). *Kompetansemål etter 10. Trinn—Læreplan i naturfag (NAT01-04)*. https://www.udir.no/lk20/nat01-04/kompetansemaal-og-vurdering/kv78?lang=nob

Utdanningsdirektoratet. (2022b). *Læreplan i valgfaget programmering (PRG01-02)*. https://www.udir.no/lk20/prg01-02

Utdanningsdirektoratet. (2022c). *Tilgjengelige kompetansepakker*. https://bibsys.instructure.com/search/all_courses?design=udir#308

Vitensenter. (2021). *Hva er et vitensenter*. https://www.vitensenter.no/hva-er-et-vitensenter/

Vygotski, L. S., Cole, M., John-Steiner, V., Scribner, S., & Souberman, E. (1978). *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press.

Weintrop, D., & Wilensky, U. (2018). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, *17*, 83–92. https://doi.org/10.1016/j.ijcci.2018.04.005

Wenger, E. (2019). *Praksisfællesskaber: Læring, mening og identitet*. Hans Reitzel.

Wikipedia.org. (2022). *Micro Bit—Wikipedia*. https://en.wikipedia.org/wiki/Micro_Bit

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE software*, *17*(4), 19–25.

Wing, J. (2017). Computational Thinking's Influence on Research and Education for All. *Italian Journal of Educational Technology*, *1*(1). https://doi.org/10.17471/2499-4324/922

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Wittek, L., & Bratholm, B. (2014). *Læringsbaner om lærernes læring og praksis*. Cappelen Damm akademisk.

Zoom.us. (2022). *Privacy*. Zoom. https://explore.zoom.us/en/privacy/

Oversikt over tabeller og figurer

Vedlegg

Vedlegg 1: Infoskriv og samtykkeskjema

Vil du delta i forskningsprosjektet

" Utfordringer og erfaringer med programmering i forbindelse med innføringen av fagfornyelsen"

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å finne ut programmering oppleves med innføring av fagfornyelsen. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål

Etter to år med fagfornyelse og med kompetansemål i matematikk som inneholde forskjellige tema fra programmering så ønsker jeg å se på hvilke utfordringer lærere støter på i deres arbeid med matematikk og programmering. I denne masteroppgaven vil jeg gjennomføre intervjuer med spørsmål til lærere om deres erfaringer. 4 – 6 informanter vil delta i studien.

Hvem er ansvarlig for forskningsprosjektet?
Universitetet i Sørøst-Norge er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Du er lærer ved en ungdomsskole, eller jobber på ungdomstrinnet som matematikklærer.

Hva innebærer det for deg å delta?

Det er helt frivillig å delta i prosjektet, og du vil være mulig å trekke seg når som helst fra prosjektet, også under intervjuet. Det vil bli tatt opp tale, tatt notater og video fra intervjuet og data vil bli lagret uten nettilknytning. All informasjon om deg vil bli anonymisert og deretter slettet ved prosjektets slutt. Det vil ca ta 45 minutter.

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.

- Det er kun jeg som vil kunne høre, se eller behandle datamaterialet.
- Det vil være koding av navn og annen identifiserende informasjon. Denne kodenøkkelen vil være lagret separat fra datamateriale.
- Datamateriale vil lagres uten netttilgang og kryptert.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?
Prosjektet vil etter planen avsluttes 1 juni 2022. Og alle data vil slettes når oppgaven er godkjent.

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

På oppdrag fra Universitetet i Sørøst-Norge har Personverntjenester 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:
- Universitetet i Sørøst-Norge:
  - Masterstudent Viggo Trummar Andersen, Viggo.T.Andersen@usn.no, tlf: 93 23 13 83
  - Veileder: Førstelektor Lars Opdal, Lars.Opdal@usn.no, tlf: 31 00 88 53
  - Veileder: Universitetslektor Reiar Kravik, Reiar.Kravik@usn.no, tlf: 31 00 99 64
- NSD – Norsk senter for forskningsdata AS, epost: personverntjenester@nsd.no , tlf: 55 58 21 17.
- Vårt personvernombud: Paal Are Solberg, paal.a.solberg@usn.no, tlf: 35 57 50 53

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:
- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Lars Opdal
(Forsker/veileder)

Viggo Trummar Andersen
(Student)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Utfordringer og erfaringer med programmering», og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i intervju
- ☐ at intervjuet tas opp på lydopptaker
- ☐ at intervjuet tas opp som videofil via Teams eller Zoom

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet, 1 juni 2022

(Signert av prosjektdeltaker, dato)

Vedlegg 2: Utdrag fra informanten Lise

Vedlegg 3
00:35:40 Lise
De begynner nok sikkert ganske lavt.
00:35:45 Lise
Ikke nødvendigvis med python programmering, men med blokkprogrammering.
00:35:50 Lise
Så tror jeg kanskje vi må godt ned på barnetrinnet, å begynne med det.
00:35:57 Lise
Og så er det vel dryppvis oppover hele tiden, og det er vanskeligste nivået kommer du vel opp på ungdomstrinnet?
00:36:08 Intervjuer
Ja, så hva vil du se på å bruke variabler, lokker, vilkår og funksjoner? Hvilket trinn hvis du skal bare tenke litt.
00:36:17 Lise
Mellomtrinnet kanskje
00:36:19 Lise
Og den andre på hva du sa, ungdomstrinnet.
00:36:30 Intervjuer
Ja her var det ikke snakk om geometriske figurer. Det hadde jo kunnet begynne med helt nye barnetrinnet. Ja, men det er ikke sikkert de begynner å programmere på det, men det kommer jo an på hvor vidt man definerer ordet programmering da.
00:36:39 Lise
For det at hvis de har ipader, så er det jo ikke noe vanskelig også sette på plass sånne blokker som gjør at en … et program, lager, en sirkel eller en firkant.
00:36:53 Intervjuer
OK og da sier jeg sier jeg fasiten da at det første det er på sjette trinn og da siste er på 10. trinn. Hva tenker du da?
00:37:02 Lise
Og da tenker jeg, at da må man forenkle dette sånn at vi får det lenger nedover, for hvis ikke, så får vi altså og innføre det på 10. trinn. Da er det mange som allerede har falt av matematikken som aldri kommer til å få dette på plass. Du må begynne mye, mye tidligere for at de ikke, skal drukne i kunnskap
00:37:30 Intervjuer
Så når disse 2 kompetansemålene det ene er fra sjette trinn det med løkker og så videre og på utforskning og det andre er på tiendetrinn som ser på sammenhenger og egenskaper.
Og når du da i møte med dine elever. Nå som går på videregående en. Har de den … innehar den kompetansen som er satt allerede i sjette klasse og 10. Klasse.
00:37:53 Lise
Nei overhodet ikke.
00:38:07 Intervjuer
Vil du si? Dette er jo sånn et Ja/Nei spørsmål som egentlig ikke er riktig da, men hva vil du? Det er jo åpenbart en utfordring da.
00:38:16 Lise
Ja, jeg er kjempeutfordring, og jeg tror det at. Vi må begynne tidligere. Vi må begynne denne typen tenkning mye tidligere enn på slutten av barneskolen og på slutten av ungdomsskolen.

00:38:32 Lise
Ja tror det bør innføres som en lek til å begynne med, så har vi kanskje sjanse til å lære.
Det er klart, men vi begynner på med gangetabellen da i andre, tredje klasse, og de kan ikke den når de går ut av ungdomsskolen heller, og da er det jo ikke å forvente at de skal klare programmering da heller, hvis det skal være en del av matematikken.

Vedlegg 3: Arnes skissering av kursing

Arnes ønsker:

> «De kursene som vi har vært på på USN, synes jeg. Det har vært bra ting der, men det jeg synes har vært utfordrende er at de har […] på sett og vis ikke alltid tatt utgangspunkt i den faktiske læreplanen. Jeg har opplevd at når vi har kommet dit, så har det vært kursing i ting som selvfølgelig er interessant og nyttig som har vært enn del av den kompetanse som heter programmerings kompetanse, men som ikke nødvendigvis ligger knyttet til den kompetansen elevene nødvendigvis skal sitte igjen med. Jeg synes at kurs stort sett bør prøve å forholde seg til kompetansemålene slik de er beskrevet i læreplan og ikke vise fram noe som er gøy» (Arne)

> «Til å utdanne profesjonen fordi det sitter mange dyktige lærere rundt omkring som er veldig interessert i dette, og som har kompetanse i dette som jeg tror kunne gjort en like god jobb med å kurse sine kolleger da? Ja at vi rett og slett kunne møtes også at enkeltpersoner som vi vet er en styrke, for eksempel i kommunen, kunne fungert litt som lærere på sånne kurs. Da vil lærere kunne kommet sammen og drøftet kompetanse, altså læreplan» (Arne)

> «[…] så kan de melde seg på ulike workshops og kurs litt basert på.
> Hva som er deres faglige interesser, og også der hvor de føler de trenger kompetanseheving. Der har de hatt som en del av den pakka, så har de hatt programmeringskurs og for eksempel programmering i minecraft eller programmering i naturfag eller programmering i musikk eller programmering og kunst og håndverk. […] Det er ulike workshops som man kommer, og så møter man da fagpersonell gjerne fra USN, men gjerne i kombinasjon med folk i felt.» (Arne)

Vedlegg 4: Intervjuguide

Intervjuguide «utfordringer og erfaringer med innføring av programmering i fagfornyelsen»

Del 1: Bakgrunn, utdanning, klassetrinn

1. Hva slags utdannelse har du oppnådd?
2. Hvor mye matematikkutdanning har du?
    a. Studiepoeng og hvor avansert matematikk?
3. Hvor lenge har du jobbet som lærer?
4. På hvilket klassetrinn underviser du nå eller har undervist på?
5. Hvilke fag i grunnskolen underviser du eller har undervist i?
    a. Faglærer eller allmennlærer?

Del 2: Fagfornyelsen, matematikk og programmering

Læreplaner

1. Høsten 2020 ble det innført nye læreplaner i matematikk på ungdomstrinnet.
    a. Hva tenker du om de nye læreplanene i matematikk?
    b. Er det spesielle områder innenfor de nye læreplanene som du er fornøyd med?
2. Er det noen områder du kunne tenkt at kunne vært forandret?
3. Læreplanene for matematikk 1. – 10. trinn inneholder kompetansemål som omhandler programmering:
    a. Hvor godt kjenner du til innholdet i disse?
    b. Hva er dine tanker om oppbygningen av kompetansemålene i matematikk?
    c. De to følgende kompetansemålene er tatt fra matematikk 1-10:
        i. bruke variablar, lykkjer, vilkår og funksjonar i programmering til å utforske geometriske figurar og mønster
        ii. utforske matematiske eigenskapar og samanhengar ved å bruke programmering
    d. Hvilke årstrinn på 1. – 10. trinn vil du si at de tilhører?
4. Hvorfor tror du at fagfornyelsen og kompetansemålene i matematikk inneholder programmering og algoritmisk tenkning?

Erfaring

1. Hvilken erfaring har du med programmering?

    a. Har du brukt programmering i undervisningssammenheng?
        i. Hvis ja: Hvordan?
        ii. Hvis nei: Hvilke utfordringer hindrer deg i å bruke programmering i undervisningssammenheng?
    b. Har du utdannelse eller kurs innenfor programmering?
        i. Hva slags kursing evt?
        ii. Studiepoeng og kurssted?
2. Hvorfor tror du at fagfornyelsen og kompetansemålene i matematikk inneholder programmering og algoritmisk tenkning?

Utfordringer

1. Utfordringer og erfaringer med programmering og algoritmisk tenkning i matematikk?
2. Hvilken nytte kan elever ha av å bruke programmering og algoritmisk tenkning i matematikkfaget?
3. Hvilke utfordringer tror du elevene vil oppleve eller erfare i møtet med programmering og algoritmisk tenkning i matematikkfaget?
    a. Tekniske
4. Hvilken nytte kan lærere ha av å bruke programmering og algoritmisk tenkning i matematikkfaget?
5. Hvilke utfordringer tror du lærere vil oppleve eller erfare i møtet med programmering og algoritmisk tenkning i matematikkfaget?
6. Hvilken holdning har du til programmering og algoritmisk tenkning i matematikkfaget?
7. Hvilke holdninger eller synspunkt opplever du at kollegiet har til programmering og algoritmisk tenkning i matematikkfaget?

Kompetanseheving

1. Hvilke grep er blitt gjort ved din arbeidsplass for å heve kompetansen i programmering og algoritmisk tenkning i matematikkfaget?
    a. Er det noe som er spesielt bra?
    b. Er det noen som kunne vært gjort bedre eller annerledes?
2. Hvilke grep har du gjort for å heve kompetansen innenfor programmering og algoritmisk tenkning i matematikkfaget?

3. Har du nok kompetanse om programmering og algoritmisk tenkning i matematikkfaget til å kunne tilrettelegge for elevenes undervisning, for å dekke de aktuelle kompetansemålene?
    a. Utdype!

Programmering i matematikk

1. Hva legger du i begrepet algoritmisk tenkning?
2. I hvor stor grad vil du bruke programmering og algoritmisk tenkning i matematikkfaget fremover?
3. Hvorfor skal elevene lære seg programmering og algoritmisk tenkning i matematikkfaget?
4. Hvilke ressurser innen programmeringsdidaktikk bruker du?
5. Programmering og stofftrengsel i matematikk?
6. Hva tenker du om ordene Koding og programmering.
7. Digitale verktøy sammenlignes med. Excel, Geogebra, programmering. Hva er funksjonen til disse verktøyene?

Del 3: Programmering og begrep

1. Hva mener du med begrepet programmering?
2. Hvordan kommer dybdelæring til uttrykk i matematikkundervisningen?
3. 1. Hva legger du i begrepet dybdelæring?
4. Hvordan tenker du at programmering kan støtte arbeid med algoritmisk tenkning? p
5. Hvordan jobber du med å bruke utvikle algoritmisk tenkning hos elevene? p
6. Hvordan presenterer du programmering for elevene? p
7. Hvordan organiserer du klasserommet? Hvordan jobber elevene sammen?
    a. Erfaringer to og to.

Begrep

8. Begrep?
    a. Hvordan vil du definere begrepet variabel?
    b. Hvordan vil du definere begrepet løkke?
    c. Hvordan vil du definere begrepet betingelse?
    d. Hvordan vil du definere begrepet funksjon i kontekst av programmering?
9. Hvordan kan programmering føre til dybdelæring innen matematikkfaget?

Noe annet du vil tilføye?