MASTEROPPGAVE

Emnekode: MAT5006_1

Navn: Jørn Hammerø

Algoritmisk tenkning i programmeringsoppgaver

-En nytenkende analyse av en matematikklærebok

Computational thinking in programming tasks

-An innovative analysis of a mathematics textbook

Dato: 15.05.2023

Totalt antall sider: 88

NORD universitet

www.nord.no

Forord

Da går 5 interessante og lærerike år som lærerstudent mot slutten med denne masteroppgaven som krona på verket. Arbeidet med masteroppgaven har vært kjempegivende og lærerikt. Ved fagfornyelsen 2020 ble programmering og algoritmisk tenkning en del av læreplan i matematikk. Masteroppgaven har vært en flott introduksjon til et tema jeg i utgangspunktet hadde begrenset kunnskap om. Etter studiet og arbeidet med denne masteroppgaven ser jeg fram til å bli lærer og undervise programmering i matematikk.

Jeg ønsker å takke veilederne mine Alexander Schmeding og Abolfazl Rafiepour for all støtten, tilgjengeligheten, de konstruktive tilbakemeldingene og tålmodigheten de har vist gjennom denne prosessen.

Sist men ikke minst vil jeg takke mine venner, familien min og spesielt mamma. Uten dere kunne jeg ikke ha gjennomført studiet og skrevet masteroppgave.

Levanger, mai 2023
Jørn Hammerø

Sammendrag

Tema i denne masteroppgaven er algoritmisk tenkning og programmering i matematikkfaget. I fagfornyelsen 2020 ble programmering en del av læreplanen i matematikk, etter internasjonal interesse for å inkludere programmering i læreplaner som en inngang til algoritme- og teknologiforståelse.

Problemstillingen i denne masteroppgaven er:

Hvilke muligheter for algoritmisk tenkning kan programmeringsoppgaver i matematikklærebøker gi elever på 8. trinn?

For å belyse problemstillingen tar jeg i masteroppgaven utgangpunkt i oppgavens (oppgavetekstens) direktiv, altså det oppgaven ber eleven om å gjøre, og plasserer disse i kategorier og underkategorier. For analyse av kategoriene bruker jeg instrumentteori, nærmere bestemt instrumentalisering og dens grener som rammeverk. Analyse av enkeltoppgaver som er typiske for hver kategori og underkategori blir også analysert i masteroppgaven. Dette for å gi leseren innblikk i hva som ligger til grunn for argumentene og sammenhengene identifisert i analysen.

For å belyse problemstillingen har jeg formulert forskningsspørsmålet:

Hvilke grener finnes det innenfor algoritmisk tenkning i programmeringsoppgaver i matematikklærebøker på 8.trinn?

Resultatene av analysen viser at mulighetene for kjennetegnet på AT algoritmisk tankegang er store i arbeid med direktivene i programmeringsoppgavene i læreboka. Resultatene viser også at det er få muligheter for kjennetegn på AT dekomponering og feilsøking i oppgavedirektivene.

Den grundige og begrunnede analysen av programmeringsoppgaver i denne masteroppgaven viser seg å være egnet som inspirasjon for lærere som ønsker å vurdere programmeringsoppgaver for undervisning på egenhånd.

Som en stemme fra praksis for å forankre forskningen i denne masteroppgaven til den norske skolekonteksten har jeg til og med intervjuet 3 matematikklærere på ungdomstrinnet. Dette for å undersøke hvilke og hvordan de bruker læringsmateriell i undervisning av programmering i matematikk.

Abstract

This master's thesis is about computational thinking (algoritmisk tenkning) and programming in mathematics. In the Norwegian subject renewal of 2020, programming became part of the curriculum in mathematics, following international interest in including programming in curricula as an introduction to understanding algorithms and technology.

The problem to be addressed in this master's thesis is:

What opportunities for computational thinking can programming tasks in mathematics textbooks provide students in the 8th grade?

To illuminate the issue, I have done a textbook analysis starting in the textbooktask directive and place these in categories and subcategories. For the analysis of the categories, I use instrument theory, more specifically instrumentalization and its branches as a framework. Analysis of individual tasks that are typical for each category and subcategory is also analyzed in the master's thesis. This is to give the reader an insight into what underlies the arguments and connections identified in the analysis.

To shed light on the problem to be addressed in this master’s thesis I have formulated the research question:

What branches are there of computational thinking in programming tasks in mathematics textbooks for the 8th grade?

The results of the analysis show that the possibilities for the characteristic of algorithmic thinking are identified when working with most of the directives in the programming tasks in the textbook. The results also show that there are few characteristics of computational thinking decomposition and debugging in the task directives.

The thorough and reasoned analysis of programming tasks in this master's thesis proves to be suitable as inspiration for teachers who want to assess programming tasks for teaching.

In order to anchor the research in this master's thesis to the Norwegian school context, I even interviewed 3 mathematics teachers at the secondary level. This is to investigate which teaching materials teachers use and how they use them when teaching programming in mathematics.

Innholdsfortegnelse

Figurliste

1.0 Innledning

For å fungere i et samfunn i utvikling er teknologisk kompetanse helt grunnleggende (Sanne et al., 2016). I samfunnet vårt er digitalteknologi blitt en stor del av vår hverdag (Manheim & Kaplan, 2019). Dagens ungdom bruker digitalteknologi for finne en partner, holde seg underholdt og oppdatert på sosiale medier. De bruker apper på smarttelefonene sine og bruker ‘’kunstig intelligens’’ som ‘’inspirasjon’’ i skolearbeidet deres. I skolen har elevene til nå fått opplæring i digitale ferdigheter der elevene utvikler kunnskap om kritisk bruk av teknologi og til hva og hvordan teknologi brukes (Kunnskapsdepartementet, 2017; Sanne et al., 2016).

Forståelse og mestring av disse teknologiene elevene bruker daglig krever en dypere forståelse for teknologien. For at elevene som er samfunnets framtid skal kunne ha kontroll over teknologien, og ikke bare være forbrukere av den må de ha en grunnleggende forståelse for teknologiens virkemåte (Sanne et al., 2016). Grunnleggende programmering og algoritmisk tenkning er en inngang til å sikre dette (Barr et al., 2011; Wing, 2008). Algoritmisk tenkning (AT) er kjernekompetansen i digital teknologi og programmering (Sanne et al., 2016; Wing, 2008). AT er en allmenn nyttig tankemåte som kan brukes for å analysere og forstå systemer i de fleste fagfelt (Wing, 2008). Dette inkluderer matematikk og programmering og på tvers av fagfeltene (Scherer et al., 2018; Kaufmann et al., 2018).

1.1 Bakgrunn for masteroppgaven og problemstillingen

Teknologi og algoritmer er blitt en stor del av menneskers hverdag, noe vi bør være bevisst på når det kommer til personvern og demokrati (Manheim & Kaplan, 2019). I dag er det et økt behov for spesialisering og høykvalifisert arbeidskraft. Scherer et al. (2018) hevder kompetanse i dataprogrammering forbedrer kognitive ferdigheter. Inkludert kreativitet, resonnering og matematiske ferdigheter. Dette har ført til en verdensomspennende interesse for å inkludere programmering i læreplaner som en inngang til algoritme- og teknologiforståelse (Tamborg et al., 2022). Fagfornyelsen 2020 innførte programmering og algoritmisk tenkning (AT) som en del av norske læreplaner inkludert i matematikkfaget i grunnskolen (Kunnskapsdepartementet, 2019). Hvilket igjen har ført til utgivelsen av nye

lærebøker i matematikk da det blant annet har blitt tilført nye kjerneelementer og kompetansemål knyttet til programmering.

Mange læreverksforlag har gitt ut nye lærebøker som er ment å gi elevene trening i det som fremheves i fagfornyelsen, hvilket inkluderer AT og programmering. For flesteparten av matematikklærerne i Norge har ikke programmering vært en del av lærerutdanningen deres og omfanget av kursing knyttet til programmering varierer. Det er også ifølge Mæland og Myklebust (2022) uklart hva læreplan i matematikk egentlig ønsker av undervisning og elevers læring i når det gjelder programmering. Bråting og Kilhamn (2022) er en svensk studie som blant annet tar for seg innføringen av programmering i deres læreplan i 2018. Studien deres viser at matematikklærere i Sverige ønsker programmeringsoppgaver som elevene kan følge uten ytterligere instruksjoner og velger ofte oppgaver i lærebøker som utgangspunkt for undervisning. Dette gjør at lærere blir påvirket av lærebøkene i matematikk om hva og hvordan de underviser i programmering, ifølge studien. Noe som også kan gjelde i en norsk kontekst, da lærebøker er en av de mest brukte undervisningsmaterialene i norsk skole (Mullis et al., 2012). Van Den Ham og Heinze (2018) fremhever matematikktekstbøkers betydning for elevers læring i faget. Jablonka og Johansson (2010) fremmer også undersøkelse av innholdet i matematikklærebøker som noe som kan hjelpe med å avdekke hvilke muligheter for læring elever blir tilbudt.

En viktig del av grunnlaget for innføringen av programmering i skole, både i Norge og internasjonalt, er også inngangen temaet gir elevene til AT som noe som er allmenn nyttig og overførbart til matematikk (Bocconi et al., 2016; Kunnskapsdepartementet, 2019). Wing (2008) skriver at AT er allmenn nyttig da egenskapen styrker intellektuelle egenskaper som kommunikasjon. AT inneholder flere kjennetegn. Bocconi et al. (2016) sine definisjoner av 6 kjennetegn på AT er:

-Abstraksjon

-Generalisering

-Algoritmisk tankegang

-Dekomponering

-Automatisering

-Feilsøking

Kjennetegnene er oversatt av Gjøvik og Torkildsen (2019). Disse kjennetegnene er også velkjente konsept i matematikken og er overførbare mellom programmering og matematikk, ifølge Kaufmann et al. (2018). Tamborg et al. (2022) sier også at programmeringsoppgaver bør være designet med tanke på AT som formål.

Problemstillingen for denne masteroppgaven blir dermed:

Hvilke muligheter for algoritmisk tenkning kan programmeringsoppgaver i matematikklærebøker gi elever på 8. trinn?

For å få bakgrunn til å kunne belyse problemstillingen har jeg også formulert forskningsspørsmålene:

Hvilke læringsmateriell benytter lærere på ungdomsskolen for å finne programmeringsoppgaver for deres undervisning?

Hvilke grener finnes det innenfor algoritmisk tenkning i programmeringsoppgaver i matematikklærebøker på 8.trinn?

1.2 Begrepsavklaringer

Så langt i masteroppgaven har jeg introdusert noen begreper som bør avklares. Derfor skal jeg i denne delen av oppgaven definere betydningen av sentrale begreper i oppgaven for å kunne belyse problemstillingen og forskningsspørsmålene.

1.2.1 Algoritmisk tenkning

Denne masteroppgaven handler om muligheter for algoritmisk tenkning (AT) i programmeringsoppgaver i matematikklærebøker. Det er derfor viktig med en begrepsavklaring av AT, og velge en definisjon av begrepet som jeg skal bruke i masteroppgaven.

Det finnes mange lignende begreper med språklige likheter, men som ikke har samme innhold som AT, ifølge Gjøvik og Torkildsen (2019). AT er ifølge Gjøvik og Torkildsen (2019) innført på norsk som et begrep med samme betydning som det engelske begrepet

‘’Computational thinking’’. AT er et forholdsvis nytt begrep og det er ikke enighet i forskningsmiljøet om hva begrepet inneholder (Shute et al., 2017).

Gjøvik og Torkildsen (2019) tar i sin artikkel opp begrepsforvirringen rundt AT. De skriver at det finnes flere forskjellige lignende begreper i litteraturen. Det vil si algoritmisk tenkning, algoritmisk tenking, algoritmisk tenkemåte og algoritmisk tankegang. De skriver at disse blir tilnærmet brukt synonymt med hverandre og poengterer at språkrådet viser til forskjeller i både ordbokoppføringene og betydningene mellom disse. ‘’Tenkning har vært oppfattet som mer generelt enn tenking (som betyr «det å tenke») og tankegang (som betyr «en bestemt tankerekke»)’’ (Gjøvik & Torkildsen, 2019, s. 31). Videre i artikkelen viser de også til algoritmisk resonnering som et begrep som blir brukt. Hvilket viser til å følge en algoritme for å finne svar. Det er altså det mer generelle AT jeg velger å bruke i denne masteroppgaven.

Definisjonen jeg velger å bruke på AT i denne masteroppgaven er Bocconi et al. (2016) sin definisjon på seks kjennetegn på AT. Til venstre i oversikten nedenfor:

Originale kjennetegn av Bocconi et al. (2016)	Oversatte kjennetegn av Gjøvik og Torkildsen (2019)
Abstraction	Abstraksjon
Generalization	Generalisering
Algorithmic thinking	Algoritmisk tankegang
Decomposition	Dekomponering
Automation	Automatisering
Debugging	Feilsøking

Gjøvik og Torkildsen (2019) har oversatt kjennetegnene til Bocconi et al. (2016), til høyre i oversikten. Disse kjennetegnene på AT mener jeg egner seg som utgangspunkt for analyse av oppgavedirektiv i matematikklærebøker og til denne masteroppgaven, da den har tydelig definerte oppdelinger innenfor begrepet som er likt for flere andre definisjoner. Likevel har jeg valgt Bocconi et al. (2016) sin definisjon som skiller seg fra andre definisjoner. Dette da den fremmer AT som en tankemåte og fordi den har en nesten unik dybde og overførbarhet til direktiv som kan brukes til kategorisering av oppgaver i matematikklærebøker.

1.2.2 Hva er en lærebok?

I denne masteroppgaven tar jeg utgangspunkt i direktiv i oppgaver i en lærebok og det er derfor nødvendig å definere hva jeg legger i begrepet lærebok. Jeg har valgt å anvende definisjon i Opplæringsloven § 9-4 på lærebøker/læremidler: ‘’Med lærebøker er her meint alle trykte læremiddel som elevane regelmessig bruker for å nå vesentlege delar av kompetansemåla i eit fag’’ (Opplæringslova, 2022).

1.2.3 Hva er en lærebokoppgave?

For å besvare problemstillingen i masteroppgaven er det også nødvendig med en definisjon på begrepet lærebokoppgave, altså oppgavene i en lærebok. Bakken og Andersson-Bakken (2021, s. 733) definerer lærebokoppgave slik: ‘’ vi definerer en lærebokoppgave som en paratekst som utfører et direktiv rettet til eleven. En paratekst er et eget tekstelement som omgir hovedteksten, og et direktiv er en talehandling som oppmuntrer eller krever at mottakeren utfører en bestemt handling’’.

1.3 Programmering og AT i læreplanens kjerneelementer

Læreplan er sentral i utformingen av oppgavene i lærebøker (Skjelbred et al., 2017). I denne masteroppgaven undersøker jeg hvordan direktivene i programmeringsoppgaver i en matematikklærebok kan gi mulighet for AT. Det er derfor sentralt å undersøke læreplanens kjerneelementers tilknytning til AT og programmering. På utdanningsdirektoratet sine nettsider under ‘’støtte til læreplan’’ finner vi hvordan de har valgt å knytte læreplanmålene i programmering, og dermed AT, opp mot kjerneelementene i læreplan (Utdanningsdirektoratet, 2019b). Der kobles programmering til to kjerneelementer i læreplan for matematikk på 1.-10. trinn, gjennom støtte til læreplan. Disse er utforskning og problemløsning og abstraksjon og generalisering.

1.3.1 Utforskning og problemløsning

Problemløsning har vært en del av norske læreplaner i betraktelig lengre tid enn AT og programmering. Problemløsning kom først med i mønsterplan (m87), der det fikk plass som et eget hovedområde (Kirke- og undervisningsdepartementet, 1987). I læreplan L97 var ikke problemløsning nevnt, men begrepet ble flettet inn i ulike hovedområder (Det kongelige kirke-, utdannings- og forskningsdepartementet, 1998) . I kunnskapsløftet Lk06 ble problemløsningskompetanse en sentral del av det å inneha en helhetlig kompetanse i matematikk (Kunnskapsdepartementet, 2006).

Kjerneelementet, utforskning og problemløsning i fagfornyelsen 2020 er koblet til flest læreplanmål der programmering nevnes. Dette kjerneelementet handler ifølge utdanningsdirektoratet om at elevene:

> «leter etter mønstre, finner sammenhenger og diskuterer seg fram til en felles forståelse. Elevene skal legge mer vekt på strategiene og framgangsmåtene enn på løsningene. Problemløsing i matematikk handler om at elevene utvikler en metode for å løse et problem de ikke kjenner fra før. Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer og innebærer å bryte ned et problem i delproblemer som kan løses systematisk».

(Kunnskapsdepartementet, 2019).

1.3.2 Abstraksjon og generalisering

Det andre kjerneelementet som utdanningsdirektoratet knytter til læreplanmål om programmering er abstraksjon og generalisering. Kjerneelementet handler om at elever gradvis går fra å bruke et konkret matematisk språk til å utvikle et mer formelt symbolspråk og resonnement som er formelle (Kunnskapsdepartementet, 2019). Generalisering er også et kjerneelement i fagfornyelsen 2020. I den står det at generalisering gjør at elever oppdager sammenhenger, strukturer og mønstre i matematikk (Kunnskapsdepartementet, 2019).

1.3.3 Kjerneelementene og AT

Dette kan ses i sammenheng med Bocconi et al. (2016) sine 6 kjennetegn på AT. Utdanningsdirektoratet definerer AT som ‘’ … en problemløsningsmetode. AT 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’’ (Utdanningsdirektoratet, 2019a). Wing (2008) og Bocconi et al. (2016) har en utvidet definisjon som ser på begrepet som en måte å tenke på istedenfor en problemløsningsmetode. Likevel er de tre kildene samstemte om at AT også omhandler noe som kan brukes i utforskning og løsning av matematiske problem slik det er beskrevet i kjerneelementet. Generalisering og abstraksjon er også felles for både AT og kjerneelementene (Bocconi et al., 2016; Kunnskapsdepartementet, 2019). Likevel er det noen forskjeller mellom kjerneelementenes beskrivelse av begrepene og litteraturen knyttet til AT sin beskrivelse. Dette ettersom AT kan sies å være en mer generell tankemåte og kjerneelementene er knyttet til matematisk tankemåte.

1.4 Struktur for oppgaven

I kapittel 2 presenteres relevant teori for å kunne belyse problemstillingen. Teorien som undersøkes i kapitlet handler om relevante aspekter ved algoritmisk tekning, programmering, lærebøker, oppgaver i lærebøker, instrumentteori og tidligere forskning.

Kapittel 3 handler om metodiske valg som er tatt i masteroppgaven og redegjør for analyseverktøyene i lærebokanalysen og i de semistrukturerte intervjuene. I tillegg presenteres det vitenskapsteoretiske ståstedet for masteroppgaven og kvaliteten i oppgaven drøftes. Til slutt i kapitlet gjennomgås forskningsetiske betraktninger og sammenhengen mellom semistrukturerte intervju av matematikklærere og lærebokanalyse i masteroppgaven.

I starten av kapittel 4 presenteres resultatene av intervjuene. Deretter analyseres direktivene i programmeringsoppgavene i den aktuelle læreboka. Analysen inneholder også analyse av enkeltoppgaver tilhørende hver kategori. I slutten av kapitlet presenteres de kvantitative resultatene av lærebokanalysen.

I begynnelsen av kapittel 5 presenteres resultatene av lærebokanalysen. Deretter drøftes aspekter ved masteroppgaven som er relevante for å kunne belyse problemstillingen opp mot

tidligere forskning. Videre i kapitlet drøftes avgrensningene for analysen. Til slutt i kapitlet drøftes programmeringsoppgaver og deres direktiv som analysegrunnlag.

I kapittel 6 oppsummeres masteroppgaven og jeg anvender funnene i oppgaven som grunnlag for noen anbefalinger til lærebokforfattere og videre forskning.

2.0 Teori

I dette kapitlet i masteroppgaven presenteres relevant teori. Jeg tar for meg begrepet algoritmisk tenkning (AT), undersøker begrepets historie, ulike definisjoner og Bocconi et al. (2016) sin bakgrunn for definisjonen og inndeling av begrepet. Samt en gjennomgang av denne inndelingen. En annen sentral del av masteroppgaven er programmering og jeg undersøker hva programmering er, historien til programmering i en skolekontekst, AT som begrunnelse for programmering, programmeringstyper, tekstbasert programmering og programmeringsspråket Python. Videre i teorikapitlet ser jeg på lærebøkers posisjon i skolen, hvordan lærebøker kan brukes, oppgavene i matematikklærebøker og presenterer den analyserte læreboka Matemagisk 8. Deretter undersøkes instrumentteori og konseptene bak instrumentalisering av artefakter som rammeverk for lærebokanalysen. Til slutt i dette kapitlet presenteres relevant tidligere forskning.

2.1 Algoritmisk tenkning (Computational thinking)

I masteroppgaven bruker jeg AT som forkortelse for det oversatte begrepet algoritmisk tenkning når litteraturen viser til det engelske begrepet ‘’Computational thinking’’ ettersom at jeg skriver på norsk. Papert (1980) anvendte begrepet ‘’computational thinking’’ først i sin bok Mindstorms—Children, Computers and Powerful Ideas. Likevel ble ikke begrepet definert før Wing (2006). Hun skriver at begrepet innebærer evnene problemløsning, systemdesign, og det å forstå menneskers oppførsel ved bygge videre på konsepter fundamental innenfor informatikk. Shute et al. (2017) hevder også at Wing’s argumentasjon ga et nytt perspektiv på forholdet mellom mennesker og datamaskiner og ble dermed opphavet til en bølge av forskning på begrepet ‘’Computational thinking’’.

Den mest siterte definisjonen av begrepet kommer fra Cuny et al. (2010). De mener at AT er en tankeprosess der løsninger er representert slik at det kan bli effektivt utført ved å følge et sett av instruksjoner for å utføre en oppgave, gjerne av en datamaskin. Andre forskere har definert AT knyttet til deres fagfelt. Barr et al. (2011) beskriver en definisjon som passer til grunnskoleutdanning. I deres artikkel skriver de at AT involverer problemløsningsferdigheter og spesielle disposisjoner som selvtillit og utholdenhet når en skal løse et problem. Berland og Wilensky (2015) definerer AT som evnen til å tenke med ''computer-as-tool'' (datamaskin som verktøy). En annen definisjon av AT er ''Elever bruker datamaskiner til å modellere deres ideer og utvikle program'' (Israel et al., 2015), der de knytter AT eksplisitt til programmering. Det kommer tydelig fram at forskningen på AT er i startfasen, og at det ikke er en entydig definisjon på begrepet. Likevel er det noen likheter blant de fleste definisjonene på begrepet. Det er et konsept som er fundamentalt for å kunne løse problemer effektivt og enklest mulig, altså algoritmisk med eller uten hjelp av en datamaskin (Shute et al., 2017).

Bocconi et al. (2016) har vurdert et vidt utvalg av definisjoner på AT og har funnet at det er noen kjernekonsepter og ferdigheter som er gjentagende i litteraturen. De bruker flere studiers definisjoner på ulike konsepter og ferdigheter knyttet til AT som utgangspunkt for en egen liste av konsepter og ferdigheter som begrepet AT omfavner. Disse studiene vurderer de selv som prominente innenfor fagfeltet (Angeli et al., 2016; Barr & Stephenson, 2011; Cuny et al., 2010; Grover & Pea, 2013; Lee et al., 2011; Wing, 2006, 2008; Selby & Woollard, 2013). Valget av disse studiene baserer de på antall siteringer, deres brede omfang i å rapportere andre studier og at de gir en rekke perspektiver og synspunkter når det gjelder forskningsstrenger og internasjonale forskningsgrupper (CSTA task force on CT, Computing At School, IFIP's Task Force Curriculum and the EDUsummIT TWG9 Curriculum group). I tillegg til at ferdighetene foreslått i disse artiklene er i tråd med ferdigheter fra resten av litteraturen som er gjennomgått i Bocconi et al. (2016).

Bocconi et al. (2016) definerer AT som: AT beskriver tankeprosesser involvert i å formulere et problem for å kunne anvende en ''computational'' løsning som involverer abstraksjon, algoritmisk tankegang, automatisering, dekomponering, feilsøking og generalisering. Definisjonen de velger å bruke på disse punktene henter de fra ulike kilder. Jeg meddeler i den skjematiske oversikten nedenfor definisjonene Bocconi et al. (2016) har valgt for disse konseptene og ferdighetene.

AT ferdighet	Definisjon
Abstraksjon	I følge Csizmadia et al. (2015, s. 7) er abstraksjon prosessen å gjøre et artefakt mer forståelig gjennom å redusere unødvendige detaljer. Ferdigheten knyttet til abstraksjon handler om å velge de rette detaljene og gjemme slik at problemet blir lettere, uten at noe viktig går tapt. En nøkkeldel av det er å velge en god representasjon av et system. Forskjellige representasjoner gjør at forskjellige ting blir lettere å gjøre.
Algoritmisk tankegang (Algorithmic thinking)	Algoritmisk tankegang er en måte å komme til et svar gjennom tydelig definerte instruksjoner. Altså trinn-for-trinn instruksjon (Csizmadia et al., 2015, s. 7).
Automatisering	Automatisering er ifølge Lee et al. (2011) en arbeidsbesparende prosess der en datamaskin blir instruert til å utføre et sett med repetive oppgaver raskt og effektivt i sammenligning til prosesseringskraften til et menneske. Når det ses på denne måten er dataprogram ‘’automatiseringer av abstraksjoner’’.
Dekomponering	Dekomponering er en måte å tenke på artefakter når det gjelder deres komponentdeler. Delene kan deretter bli forstått, løst, utviklet og evaluert separat. Dette gjør komplekse problemer lettere å løse, nye problemer bedre forstått og store systemer lettere å designe (Csizmadia et al., 2015, s. 8).
Feilsøking (Debugging)	Feilsøking (Debugging) er den systematiske anvendelsen av analyse og evaluering ved bruk av ferdigheter som testing, sporing og logisk tenking til å forutse og verifisere utfall (Csizmadia et al., 2015, s. 9)
Generalisering	Generalisering er assosiert med å indentifisere mønstre, likheter og forbindelser, og utnytte disse egenskapene. Det er en måte å raskt løse nye problemer basert på løsninger til tidligere problemer og bygge på erfaringer. Spørsmål som: ‘’er dette likt et problem jeg allerede har løst?’’ og ‘’Hvordan er det annerledes?’’ er viktige å stille her, det er også prosessen å

	gjenkjenne mønstre i både dataen som blir bruk og prosessene/strategiene som blir brukt. Algoritmer som løser noen spesifikke problem kan bli utviklet til å løse en hel klasse av like problemer (Csizmadia et al., 2015, s. 8)

2.2 Programmering

Gjøvik og Torkildsen (2019) beskriver programmering som et naturlig miljø å implementere AT via. Ifølge dem er programmering noe som dreier seg om å løse et problem ved hjelp av en algoritme eller et program:

> «Det vil si forarbeid, planlegge, skrive, tegne, bearbeide, undersøke om problemet er løst, eksperimentere, revurdere, argumentere for at algoritmen alltid virker, generalisere algoritmen til å løse en hel klasse av problemer og sjekke effektivitet. Programmering kan sies å være å kommunisere med logikk. Det kan vi også si er en del av matematikken.»
> (Gjøvik & Torkildsen, 2019, s. 34).

De legger også vekt på at programmering er mer enn bare koding. Koding er aktiviteten med å reformulere dette inn i et programmeringsspråk.

2.2.1 Programmering i skolen

Papert (1980) var den første til å inkludere programmering i undervisning i grunnskole. Han var med på å skape et programmeringsspråk (Logo) som hadde som formål å lære yngre elever matematikk. I 1967 ble Logo laget av Papert og kollegaene hans på Massachusetts Institute of Technology (MIT) gjennom ‘’the Logo Project’’(Feurzeig et al., 1970). I Logo kunne elevene programmere en virtuell skilpadde som kunne utføre ulike kommandoer. Papert (1980) beskriver skilpadden som ‘’objects to think with’’ og ideen var at elevene skulle arbeide med programmering og matematikk på samme tid.

Med hjelp av Logo oppnådde unge elever imponerende resultater. Papert (1980) brukte erfaringene fra disse forsøkene til å argumentere for at det var mulig for unge elever å

programmere relativt avanserte algoritmer. Samt at læringsprosessen en elev gjennomgikk ved å programmere kunne gi nye muligheter for læring, tenkning og emosjonell vekst, og burde være idealet for kognitiv utvikling hos elever.

Papert tok til orde for en storstilt skolereform på 1980-tallet, men mange forskere var kritiske til Papert's Logo (Mayer et al., 1986; Pea & Kurland, 1984). Utviklingen mistet kraft og ideen om at programmering kunne anses som en idealvei til kognitiv utvikling ble forlatt.

På 1970-tallet ble EDB (elektronisk databehandling) introdusert som valgfag i grunnskolen, som yrkeslinjeretning- og som yrkesfag i videregående skole (Kirke og undervisningsdepartementet, 1984, s. 15). EDB begrepet ble på 1990-tallet i stor grad byttet ut med begrepet IT (informasjonsteknologi). Den teknologiske utviklingen gikk mye raskere enn mange hadde forventet. Denne utviklingen fant også sted i den norske grunnskole, noe senere enn i resten av samfunnet. Hovedfokuset innenfor teknologi ble lagt på hvordan en skulle bruke disse nye teknologiene, ikke hvordan og hva som fikk de til å fungere (Meld.St. 42 (1989-1990)). I 2013 kom en utredning fra Digitutvalget fram til at det har vært for stort fokus på å lære hvordan datamaskiner kan brukes til kommunikasjon, tekstskriving og andre humanistiske aspekter, mens algoritmer, matematikk og teknologi har blitt nedprioritert. Hvilket er til hinder for et mål om å sikre digital-verdiskapning, samt utdanne fremtidige generasjoner til å kunne skape og ikke bare konsumere (NOU 2013: 2).

Dette endret seg da Wing (2006) fremmet ''Computational thinking'' (AT) som en universell tenkemåte og ferdighet for alle. Det ble starten på en bølge med etterlysning av utdanning i AT med programmering som det mest sentrale verktøyet for å få det til. Mange land i verden har nå innført programmering og AT i skolen. I fagfornyelsen 2020 er programmering blitt en del av mange fag i grunnskolen, deriblant matematikk der det er innført i størst grad knyttet til begrepet AT (Kunnskapsdepartementet, 2019).

2.2.2 Programmeringstyper og Python

Det finnes flere typer programmering, analog-, blokk- og tekstbasert programmering (Kölling, 2015). Hovedforskjellen på disse er at analog programmering er når elever arbeider med grunnleggende prinsipper innenfor programmering uten datamaskiner eller roboter (Statped, 2021a). I blokkprogrammering anvendes ferdige brikker eller blokker til å lage kommandoer. I tekstbasert programmering anvendes tekst som kilde til det å lage kommandoer. Begge disse

typene programmering gjøres i et programmeringsspråk (Statped, 2021b). Ofte går elevene over til tekstbasert programmering på ungdomsskolen, Ifølge Kölling (2015). Det er tekstbasert programmering i programmeringsspråket Python det legges opp til å bruke i arbeid med programmeringsoppgavene i læreboka Matemagisk 8 (Kongsnes & Wallace, 2020).

Tekstbaserte programmeringsspråk er programmeringsspråk der en skriver inn tekst i programmet i form av kommandoer, istedenfor å bruke blokker til å utføre kommandoer slik det gjøres i blokkbaserte programmeringsspråk (Kölling, 2015; Rossen, 2019). Python ble utgitt i 1991 og er et programmeringsspråk som kan brukes til mange ulike formål. Det kan brukes til profesjonelle formål som inkluderer datavitenskap, programvareutvikling og automatisering, i tillegg til utdanningsformål (Dvergsdal, 2019; Kölling, 2015; Python Software Foundation, u.å).

Python er et høynivå programmeringsspråk (Dvergsdal, 2019; Rossen, 2019). Det er flere elementer som går igjen i de fleste høyspråk-programmeringsspråkene (direkte sitat av liste):

- ulike typer variabler, som tekst, tegn, heltall, reelle tall, logiske variabler (med verdiene sann og usann)
- operasjoner som utføres på disse variablene, blant annet aritmetiske operasjoner på tall og logiske operasjoner
- kommandoer relatert til inn- og utdataenheter
- adgangen til å gruppere rekker av kommandoer og operasjoner i enheter som funksjoner, prosedyrer og objekter, slik at man slipper å måtte gjenta dem hver gang de skal utføres
- adgangen til å strukturere operasjoner som skal gjentas et visst eller vekslende antall ganger

  (Rossen, 2019)

2.4 Lærebøker og lærebokoppgaver

Lærere i Norge i dag bestemmer hvilket undervisningsmateriell og innholdet i deres undervisning så lenge det er i tråd med styringsdokumentene til skolen (Dahl et al., 2015). Som bakgrunn for valget av analyse av en lærebok i matematikk og oppgavene i den har jeg derfor valgt å undersøke hva litteraturen sier om lærebøkers posisjon i skolen og hvordan

lærebøker kan brukes i undervisning. Jeg har også tatt med et avsnitt om hva litteraturen sier om lærebokoppgaver, da det er dette jeg spesifikt skal analysere senere i oppgaven.

2.4.1 Lærebøkers posisjon

Læreverkene som er produsert spesifikt for skolen er en vesentlig faktor som skiller skole fra andre arenaer (Skjelbred et al., 2017). Ifølge Mullis et al. (2012) ble lærebøker i matematikk i 2011 brukt av flere enn 75% av elever i grunnskoleutdanningen på verdensbasis og av flere enn 95% av elever i norsk grunnskole (Mullis et al., 2012). Lærebøker er sentrale i undervisningen i skolen i Norge, ifølge Skjelbred et al. (2017, s. 23). Lenge var læreboka enerådende som læremiddel i skolen. Først etter andre verdenskrig fikk skolen inn andre læringsressurser. Innføringen av felles 9-årig grunnskole for alle har, ifølge Skjelbred et al. (2017, s. 24), hatt stor innvirkning på lærebøkers rolle da det ble stilt høyere krav til læremidlene. Dette gjennom at fokus på selvstendig arbeid, elevaktivitetsprinsippet og differensiering kom mer fram i skolen enn tidligere. Det gjorde altså at bøkene måtte by på oppgaver som aktiviserte elevene og kunne gi elevene overkommelige utfordringer ved at vanskelighetsgraden på oppgavene skulle variere. Den teknologiske utviklingen har ført til mange endringer i samfunnet i nyere tid, men selv etter denne utviklingen har lærebøker beholdt en sterk posisjon i skolen, ifølge Skrunes (2010).

2.4.2 Hvordan brukes lærebøker?

Bakken og Andersson-Bakken (2021) sin forskning viser at oppgavene i lærebøker er ganske like fra 1990-tallet og i dag. Men hvordan lærebøker brukes har endret seg gjennom de nesten 300 år av lærebøkers eksistens. I den siste halvdelen av 1900-tallet fikk lærere lite spillerom da bøkene skulle være selvinstruerende. I senere tid har dette endret seg og det er nå lærerne som selv velger hvordan og i hvilken grad læreboka skal brukes i undervisningen deres (Skjelbred et al., 2017). Likevel kan undervisningen til de fleste lærere sies å være lærebokstyrt, ifølge Koritzinsky (2014, s. 231). Koritzinsky mener at dette kan komme av at selv om det er læreplanen og styringsdokumentene som bestemmer mye av det elevene skal lære, kan læreboka sees på som en form for fasit eller ramme for dette. Remillard (2005) skriver om forskning innenfor læreres bruk av undervisningsmateriell i matematikk inkludert

lærebøker, og beskriver i sin studie forskjellige måter å vurdere læreres bruk av disse. En av disse er å ta utgangspunkt i undervisningsmateriellet og vurdere i hvilken grad læreren følger det materiellet legger opp til. Studien til Bråting og Kilhamn (2022) viser at lærerne følger det læreboka legger opp til i stor grad når det kommer til undervisning i programmering i matematikk, da lærerne ikke selv er trygge på fagstoffet.

2.4.3 Oppgaver i matematikklærebøker

På grunn av den pågående debatten om utdanningsstandarder og den medfølgende nyinnspillingen av læreplaner i tilknytning med den nye orienteringen mot forventet kompetanse, har oppgaveorientering og kvaliteten på oppgavene i økende grad blitt flyttet til sentrum for oppmerksomheten innenfor utdanningsforskning (Andersen, 2020). Ifølge Thonhauser (2008) er oppgaver katalysatorer for læringsprosesser og det er i lærebøker dette potensialet kan utfolde seg på en meningsfull måte. Storskala vurderinger av resultatene av PISA og TIMSS, oppsummert av Andersen (2020), viser at dagens undervisning kommer til kort når det gjelder å aktivere kognisjon og nå opp til didaktiske standarder. En kritikk har vært at lærere bruker lite krevende oppgaver som ikke legger til rette for utvikling av kompetanser.

2.4.4 Matemagisk 8

Matemagisk 8 er en matematikk lærebok for 8. trinn utgitt av forlaget Aschehoug. Det er 165 markerte programmeringsoppgaver i læreboka. Det finnes ikke et krav om hvilket programmeringsspråk elevene skal bruke i arbeidet med oppgavene, men alle disse er rettet mot programmeringsspråket Python. Programmeringsoppgavene finnes fordelt utover i kapitlene og temaene i læreboka. Det er også ulike nivåer på programmeringsoppgavene i boka. I de fleste av oppgavene i boka er det ingen tilknyttede føringer som bestemmer hvordan de skal arbeides med. Noen oppgaver er det som omtales som ‘’snakke matte’’ oppgaver i læreboka der elevene skal samarbeide om og diskutere i arbeidet med dem (Kongsnes & Wallace, 2020).

Asbjørn Lerø Kongsnes og Anne Karin Wallace er forfatterne av Matemagisk 8. Kongsnes har PPU-utdanning med fagdidaktikk i matematikk fra høgskolen i innlandet. Han har en

bachelor i matematikk og økonomi, og jobber som lærer på en ungdomsskole i Kongsvinger. Anne Karin Wallace er lektor ved Molde videregående skole. Hun har undervist biologi, matematikk og informasjonsteknologi på videregående skole. I tillegg har hun undervist informatikk ved høyskolen i Molde (Aschehoug, u.å).

2.5 Instrumentteori

For å kunne analysere oppgaver i en matematikklærebok i tilknytning til problemstillingen i denne masteroppgaven er det sentralt med et teoretisk rammeverk for å kunne undersøke læreboka, og dermed oppgavene i den mot elevers AT. Dette har jeg tenkt å undersøke med hjelp av Instrumentteori og instrumentaliseringsdelen av teorien.

Instrumentalisering handler om bruken av artefakter som instrumenter som er skapt og blir brukt av mennesker for å hjelpe, assistere, støtte, forstørre og/eller styrke aktiviteten deres (Trouche, 2020; Artigue, 2002). Instrumentalisering er handlingen å gi noen et instrument eller handling der noen anskaffer seg et instrument, for å utføre en gitt aktivitet. Den vanlig oppfatning at matematikk er en ren mental aktivitet, står i motsetning til betydningen av verktøy i matematisk aktivitet. Det å gjøre matematiske handlinger handler altså også om utvelgelsen, bruken og utviklingen av verktøy. Disse verktøyene åpner for muligheter, men fastsetter også hvordan vi gjør matematiske handlinger. Tanken om instrumentalisering er en del av et nettverk av konsepter som er knyttet sammen, ifølge Trouche (2020). Disse konseptene kan forklares ut fra 4 ulike grener, som vist i oversikten nedenfor:

Gren av instrumentaliseringskonsept	Forklaring
Begrensninger (constraints)	Grenen begrensninger handler om artefaktets forpliktende egenskap, der instrumentet forplikter brukeren å anvende instrumentet på en gitt måte. Det kan gjøre det vanskelig å utføre noen operasjoner (Trouche, 2020).
Aktivisering (enablement)	Aktivisering er de mulighetene instrumentet gir å gjøre noe på en effektiv måte (Trouche, 2020).
Potensialer (potentialities)	Mulighetene til å gjøre noe som ikke var mulig å gjøre for brukeren før, er det potensialer handler om (Trouche, 2020).
Anbud (affordances)	Anbud handler om instrumentet i lys av hvilken tilnærming til tema instrumentet støtter, hvilket gjør det lettere for brukeren å gjøre noe på den måten (Trouche, 2020).

2.6 Tidligere forskning

Det har tidligere blitt forsket noe på programmeringsinnholdet i oppgaver knyttet til AT. Elicer og Tamborg (2022) og Tamborg et al. (2022) undersøker forholdet mellom PCT (programming and computational thinking) og matematikk i ulike undervisningsoppgaver. De gjør dette i lys av det daværende danske pilotprosjektet teknologifag. Metoden for analyse i disse studiene er egendefinerte og henter inspirasjon fra Bråting og Kilhamn (2022) og Misfeldt et al. (2020).

I Elicer og Tamborg (2022) bruker de Shute et al. (2017, s. 142) sin definisjon på AT: ‘’det konseptuelle fundamentet nødvendig for å løse problemer effektivt med løsninger som er mulig å gjenbruke i forskjellige kontekster’’. De viser også til diskusjonen om AT karakteriseres best med en lukket definisjon eller av dens kjennetegn. De bruker likevel hovedsakelig begrepet ‘’programming and computational thinking’’ hvilket kan oversettes til programmering og algoritmisk tenkning, i studien. De referer til at AT er et bredt begrep der programmering ikke er en av kompetanseområdene i mange definisjoner av begrepet. De bruker derfor PCT i sin studie for klarhetens skyld.

Det finnes også noe tidligere forskning som undersøker programmeringsinnholdet i matematikklærebøker knyttet til AT. Bråting og Kilhamn (2022) ser på hva som karakteriserer programmeringsinnholdet i svenske matematikklærebøker fra 1. til 6. klasse, for elever 7 til 12 år gamle, etter at programmering ble satt inn i svensk læreplan i 2018. Studien har et spesielt søkelys på forholdet mellom matematikk og programmering i lærebøkene. Der de tar utgangspunkt i tekst, bilder, aktiviteter og illustrasjoner, men ser hovedsakelig på oppgavenes direktiv.

Analyseverktøyet deres kategoriserer oppgavene utfra hvilken type handling elevene blir bedt om å gjennomføre i oppgaven. Oversatt fra svensk er kategoriene:

- Følge
- Finn regel
- Feilsøke
- Forme og skape
- Forklare
- Forestille seg

Analyseverktøyet er basert på Brennan og Resnick (2012) sitt rammeverk *computational concepts* og Benton et al. (2016) sitt rammeverk 5’e. Analysen er en summativ innholdsanalyse der søkelyset er på hvor ofte AT og matematiske begreper forekommer i oppgaver.

Bråting og Kilhamn (2022) definerer AT (computational thinkning) slik: AT kan forklares som en prosess der vi beskriver, analyserer og løser problemer på en måte slik at datamaskiner kan assistere arbeidet. De deler også Aho (2012, s. 832) sin definisjon på AT:

‘’tankeprosessen involvert i å formulere problemer slik at løsningen deres kan representeres som ‘’computational’’ steg og algoritmer’’.

Det bør også nevnes at Stokkenes (2022) brukte det samme analyseverktøyet som Bråting og Kilhamn (2022) i sin masteroppgave, der han analyserte norske matematikklærebøker for både mellom- og ungdomstrinnet.

3.0 Metode

Etter å ha undersøkt relevant teori i forrige kapittel skal jeg i dette kapitlet ta for meg metoden i masteroppgaven samt undersøker alternative metoder opp mot tema for forskningsprosjektet. Jeg begynner med bakgrunn for valg knyttet til problemstillingen. Videre i kapitlet tar jeg for meg masteroppgavens vitenskapsteoretiske ståsted. Deretter presenterer jeg bakgrunn for valg knyttet til problemstillingen, vitenskapsteorien, kvalitetsvurderingen med reliabilitet og validitet i forskningen. Jeg tar også for meg etiske betraktninger og min forforståelse for forskningsprosjektet. Til slutt i kapitlet blir det semistrukturerte intervjuet knyttet opp mot lærebokanalysen.

3.1 Bakgrunn for valg knyttet til problemstillingen

Den valgte problemstillingen for masteroppgaven er:

Hvilke muligheter for algoritmisk tenkning kan programmeringsoppgaver i matematikklærebøker gi elever på 8. trinn?

Bakgrunnen for denne problemstillingen:

Programmering har blitt en stor del av læreplan i matematikk på grunnskolen (Kunnskapsdepartementet, 2019). Denne vektleggingen av programmering i læreplan for matematikk har etter min mening ikke blitt reflektert i studiet jeg tar, grunnskolelærer 5-10.trinn. I utgangspunktet føler jeg meg ikke godt nok forberedt for å undervise programmering i matematikk. Derfor ser jeg på masteroppgaven som en strålende mulighet til å opparbeide meg kompetanse innenfor programmering og undervisning av det i

matematikkfaget. Det er også uklart hva elevene skal lære når det kommer til programmering i matematikk (Mæland & Myklebust, 2022).

Bakgrunnen for at jeg har valgt nettopp **algoritmisk tenkning** som en tenkemåte for undersøkelse av programmeringsoppgaver er todelt. AT kan forklares som noe som brukes når en programmerer (Csizmadia et al., 2015). AT er også en meget nyttig tankemåte både i arbeid med matematikk og i veldig mange andre disipliner og hverdagslige situasjoner (Wing, 2008; Elicer & Tamborg, 2022).

Når jeg skulle velge noe å undersøke med AT som utgangspunkt knyttet til programmering i matematikk falt valget på oppgaver i en **matematikklærebok**. Valget av lærebok som analyseobjekt baserer seg på at oppgavene som finnes i programmering er forholdsvis lik på tvers av læringsplattformer (Bråting & Kilhamn, 2022). I tillegg viser tidligere forskning at lærere velger å bruke læreboka som ‘’fasit’’ for undervisning av programmering i matematikk (Bråting & Kilhamn, 2022).

Bakgrunnen for valget av **oppgavedirektiv** i matematikklærebok er at det er oppgavene som gir et direktiv om handling til elevene og er derfor best egnet til analyse, i motsetning til for eksempel forklaringene i boka eller boka som helhet. Søkelys på direktiv gir også en mulighet til avgrensning for å kunne undersøke enkeltoppgaver i dybden.

Grunnen til at jeg bruker ordet mulighet i problemstillingen er at en kan kun si noe om **muligheten for læring** gjennom å undersøke oppgavene i en lærebok (Høgheim, 2020). En kan altså ikke si noe om hva elever faktisk gjør. Likevel vil det være mulig å si noe om tenkningen til alle elever som klarer å gjennomføre direktivet.

Når det kommer til valget av **8.trinn** som målgruppe er det i den aktuelle lærebokserien hovedsakelig lærebøkene for 8.- og 10. trinn som inneholder programmeringsoppgaver. Jeg har valgt læreboka for 8.trinn da oppgavene i denne boka er enklere å sette seg inn i og forstå enn oppgavene i læreboka for 10.trinn. Hvilket gjorde analyse av oppgavene i læreboka for 8.trinn mer gjennomførbar i denne masteroppgaven.

3.2 Vitenskapsteori

I delkapitlet over har jeg gjennomgått bakgrunnen for valg knyttet til problemstillingen og fortsetter i dette delkapitlet med gjennomgang av vitenskapsteorien som ligger til grunn for denne masteroppgaven.

Et helt sentralt spørsmål innenfor vitenskapsteorien er knyttet til framgangsmåten for å anskaffe *sikker* kunnskap om verden. Framgangsmåten vi anvender for å skape kunnskap avhenger av vårt syn på kunnskap og henger sammen med hvilken informasjon vi ser på som nødvendig for å skape denne kunnskapen. Målet med vitenskapsteori er å bygge en bro mellom metode og sikker kunnskap. For å få til dette må vi stille spørsmål til ordene ‘’sikker’’ og ‘’kunnskap’’. Definisjonen av kunnskap er: *en begrunnet sann oppfatning* (Høgheim, 2020). Ontologi og epistemologi er to filosofier knyttet henholdsvis til ordene ‘’begrunnet’’ og ‘’sann’’. Det er viktig å være bevisst disse to aspektene i forskning, noe jeg er i arbeidet med dette forskningsprosjektet, i søken etter å skape kunnskap.

I kvalitativ forskning er den ontologiske forutsetningen at det finnes mange virkeligheter (Nyeng, 2021). Resultatet av forskningen i denne masteroppgaven vil dermed bli påvirket av min virkelighet, som jeg skal utvide og forklare i løpet av forskningen. I denne forskningen er det min fortolkning av det som observeres som jeg prøver å skape kunnskap gjennom. Det er helt klart et ønske at mine fortolkninger av oppgavene i læreboka også skal gi mening i andres virkelighet. Noe jeg forsøker å oppnå gjennom forskningsgrunnlaget til forskningsprosjektet, metoden jeg velger for å belyse problemstillingen og gjennom beskrivelsen av denne metoden.

3.2.2 Hermeneutikk

Hermeneutikk er fortolkningslære. Ordet hermeneutikk har opprinnelse i det greske ordet hermeneus, hvilket betyr tolk eller fortolker. Gadamer er en filosof som står sentralt i hermeneutikken. Han mente at hermeneutikk er noe mer grunnleggende og omfattende enn metodene i vitenskapen. Hermeneutikken er kritisk til positivismens tro på objektiv og sikker viten basert på rene erfaringsdata. Hermeneutikken ser i stedet for på forskning og vitenskap

som systematisk arbeid med fortolkninger (Johannessen et al., 2016). I mitt forskningsprosjekt er det oppgavene i den aktuelle læreboka som skal fortolkes.

3.3 Metode for å belyse problemstillingen

Målet med denne masteroppgaven er å finne ut noe om AT i programmeringsoppgaver i matematikkfaget. Metoden for å undersøke dette skal jeg vurdere i denne delen av kapitlet. Videre i dette kapitlet skal jeg ta for meg metoden jeg har valgt for dette forskningsprosjektet.

3.3.1 Valg av metode

AT er utgangspunktet for analyse av programmeringsoppgaver i matematikklæreboka. Det er flere grunner til å undersøke programmeringsoppgaver i en matematikklærebok opp mot AT. Tidligere forskning knyttet til AT har hovedsakelig undersøkt observerbare tegn på AT hos elever (Csizmadia et al., 2015). Observasjon som forskningsmetode er når forskeren bruker blant annet synet og hørselen sin i innsamlingen av data som har utspring fra menneskelig adferd (Dalland, 2020). I dette forskningsprosjektet kunne jeg ha brukt Csizmadia et al. (2015) sin beskrivelse av gjenkjennbar adferd som kan bli observert i klasserommet. Ved eksempelvis observasjon av en undervisningsøkt eller lignende kunne denne metoden gitt data på hva noen elever uttrykker av AT der og da. Eller jeg kunne gjennom et undervisningsopplegg undersøkt utviklingen av AT elevene ga uttrykk for i begynnelsen og slutten av undervisningsopplegget. Likevel har jeg valgt lærebokanalyse da disse tilnærmingene til tema gir begrenset potensial for generalisering av resultater, da det kun er noen enkelte elevers læring i en unik læringsituasjon som undersøkes.

Jeg har valgt å anvende en annen inngangsvinkel til tema som er lite brukt i forskning tidligere, nemlig å undersøke mulighet for AT gjennom lærebokanalyse (Bråting & Kilhamn, 2022; Elicer & Tamborg, 2022). Som jeg ved hjelp av relevant teori for masteroppgaven argumenterer frem som en gunstig måte å finne ut hvilke muligheter for AT som finnes i programmeringsoppgaver i den aktuelle læreboka.

Ettersom at jeg velger lærebokanalyse er praksisnærhet viktig å sikre da jeg i masteroppgaven ikke direkte undersøker noe som skjer i praksis i skolen. Dermed blir et sentralt spørsmål knyttet til metodevalget: hvordan sikrer jeg praksisnærhet i forskningsprosjektet? Jeg har valgt å gjøre dette ved å anvende semistrukturerte intervju av matematikklærere på ungdomsskolen som en ‘’stemme fra praksis’’ som skal sikre at forskningsgrunnlaget jeg baserer masteroppgaven på stemmer også i en norsk kontekst. Jeg har valgt å undersøke dette gjennom forskningsspørsmålet:

Hvilke læringsmateriell benytter lærere på ungdomsskolen for å finne programmeringsoppgaver for deres undervisning?

Dette er allerede undersøkt i andre lands kontekster og viser at lærere støtter seg på læreboka i undervisning av programmering i matematikkfaget (Bråting & Kilhamn, 2022; Tamborg et al., 2022). Likevel må det undersøkes om dette også stemmer i en norsk kontekst. Derfor har jeg valgt å sikre praksisnærheten i masteroppgaven gjennom semistrukturerte intervju med 3 matematikklærere i ungdomsskolen.

Jeg bruker også en operasjonell definisjon av instrumentteoriens instrumentaliseringsgrener som rammeverk for analyse av oppgaver i masteroppgaven. I undervisning er det mye som påvirker læringssituasjonen (Van Den Ham & Heinze, 2018; Dahl et al., 2015). Dette kan være alt fra lærerens veiledning til bildene i læreboka. Det vil også gjelde elevers bruk av AT. Derfor er det nødvendig med en tydelig avgrensning av hva som skal analyseres. Instrumentteoriens instrumentaliseringsgrener setter søkelyset på det valgte artefaktet for analyse. Det viser seg likevel at instrumentaliseringsgrenene med den formen de tar i Trouche (2020) trenger en tilpasning for å best mulig kunne undersøke muligheter for AT i programmeringsoppgaver i den aktuelle læreboka. Derfor har jeg i denne masteroppgaven operasjonalisert rammeverket.

3.3.2 Lærebokanalyse som metode

Rezat og Sträßer (2015) beskriver forskningen på lærebøker i matematikk som delt. Disse delene er forskning som fokuserer på: hva som påvirker innholdet i lærebøker, selve matematikklæreboka og på bruken av lærebøker og effekten av dem. Dette forskningsprosjektet er hovedsakelig knyttet til forskning om selve matematikklæreboka, men

har bakgrunn i forskning om bruken av lærebøker i faget som en sikring av praksisnærheten gjennom semistrukturerte intervju. En av metodene som vanligvis anvendes i de to førstnevnte delene av forskningsfeltet, inkludert i denne masteroppgaven, er innholdsanalyse av lærebøker. Hvilket er en veletablert metode innenfor samfunnsvitenskapen og beskriver prosessen der en forsker analyserer en tekst knyttet til en bestemt kontekst. De aller fleste studiene innenfor feltet anvender også innholdsanalyse som hovedmetode (Rezat & Sträßer, 2015). Krippendorff (2018) beskriver innholdsanalyse som en forskningsmetode for å gjøre replikerbare og gyldige slutninger fra tekster.

Innholdsanalyser av lærebøker er hovedsakelig kvantitative eller kvalitativ med kvantitative elementer. Kvantitativ innholdsanalyse kan være basert på numerisk telling av forhånsdefinerte forekomster, etterfulgt av en statistisk analyse av dataen. Kvalitative innholdsanalyser undersøker i større grad betydningen av innholdet i teksten. Kvalitative innholdsanalyser kan deles opp i to underkategorier, induktive- og deduktive innholdsanalyser. Hovedforskjellen er at deduktive innholdsanalyser tar utgangspunkt i forhåndsdefinerte kategorier utfra eksisterende forskning, mens i induktive innholdsanalyser skapes kategoriene underveis i innsamlingen av dataen (Rezat & Sträßer, 2015). I dette forskningsprosjektet er jeg ute etter å finne ut av den kvantitative mengden av oppgaver som faller inn under de ulike kategoriene, hvilken betydning disse kategoriene har og betydningen mengdefordelingen av dem har for elevers mulighet for AT. For å gi leseren innblikk i hva som ligger til grunn for vurderingene og sammenhengene som identifiseres i analysen har jeg også gjennomført analyse av enkeltoppgaver. Dermed kan jeg si at innholdsanalysen i mitt forskningsprosjekt er kvalitativt med kvantitative elementer og er induktiv da jeg skal lage kategorier basert på oppgavenes direktiver underveis i analysen.

En av utfordringene med kvalitativ dataanalyse er reduksjon av store mengder skriftlige data til håndterbare og forståelige proporsjoner. Reduksjon av data på en slik måte at kvaliteten på de kvalitative dataene blir opprettholdt er et nøkkelelement og målet med kvalitativ innholdsanalyse (Cohen et al., 2007). Kategoriene i analysen i forskningsprosjektet skal ha bakgrunn i den eksisterende forskningen til Bocconi et al. (2016) og deres 6 kjennetegn på AT. Likevel skal kategoriene utvikles underveis i analysen. Fauskanger og Mosvold (2015) sin definisjon på summativ innholdsanalyse der fokuset er på hvordan ord eller innhold forekommer i en bestemt kontekst, passer bra til dette forskningsprosjektet. Dette da jeg i masteroppgaven undersøker hvor ofte og i hvilken kontekst de forskjellige kjennetegnene på

AT framtrer i oppgavene i læreboka. Cohen et al. (2007) beskriver en kvalitativ innholds-analytisk prosess som jeg henter inspirasjon fra i analyseprosessen i denne masteroppgaven.

Cohen et al. (2007) anbefaler å begynne prosessen med definereringen av tydelige forskningsspørsmål som skal møtes gjennom innholdsanalysen. Dette vil også inkludere hva man ønsker fra tekstene som skal analyseres. Forskningsspørsmålet kan også være avledet fra teorien som skal testes (Cohen et al., 2007). I mitt tilfelle er et av del-målene og ønskene med oppgaven om og hvilke av de 6 kjennetegnene på AT av Bocconi et al. (2016) som er til stede i oppgavene i den aktuelle matematikklæreboka. Forskningsspørsmålet jeg har valgt for å undersøke dette blir:

Hvilke grener finnes det innenfor algoritmisk tenkning i programmeringsoppgaver i matematikklærebøker på 8.trinn?

Videre i prosessen skal en definere populasjonen datamaterialet skal hentes fra. Når det kommer til innholdsanalyse handler ikke bare begrepet populasjon om mennesker, men også og i all hovedsak om tekster, inkludert lærebøker (Cohen et al., 2007). I mitt forskningsprosjekts innholdsanalyse er oppgavene knyttet til programmering i norske matematikklærebøker for 8.trinn utgitt med bakgrunn i fagfornyelsen, da programmering og AT ikke var en del av tidligere læreplaner (Det kongelige kirke-, utdannings- og forskningsdepartementet, 1998; kunnskapsdepartementet, 2006). Dette gjør at det er få eller ingen programmeringsoppgaver i matematikklærebøker utgitt før utviklingen av fagfornyelsen.

Deretter skal utvalget defineres (sampling). En må bestemme seg for hvilken strategi, av mange, en skal bruke for å velge analyseobjekt. De sentrale spørsmålene ved sampling av tekster handler om representativitet, tilgang, utvalgets størrelse og hvor generaliserbare resultatene er (Cohen et al., 2007). Jeg har tilgang til de fleste matematikklærebøker som selges i Norge. Oppgavene i norske lærebøker er forholdsvis like (Bakken & Andersson-Bakken, 2021). Dette ser jeg også når det kommer til programmeringsoppgaver i lærebøker for 8.trinn. Derfor blir behovet for et bredt utvalg minimal. Etter å ha intervjuet 3 lærere ved en ungdomsskole, har jeg funnet at læreboka de bruker i undervisning av programmering i matematikk er Matemagisk 8, og jeg har derfor valgt denne læreboka for gjennomføring av analyse.

Definering av konteksten som frembringer datamaterialet er også sentralt i analyseprosessen, ifølge Cohen et al. (2007). Når det kommer til programmeringsoppgavene i Matemagisk 8 ser

jeg derfor på bakgrunnen for hvorfor læreboka ble skrevet. Mye kan spores tilbake til fagfornyelsen 2020 som innførte programmering i matematikkfaget i norsk grunnskole (Kunnskapsdepartementet, 2019). Forfatternes yrkeskarriere og det som fremmes som målet og læremetoden i boka er også sentralt her.

Når konteksten som frembringer datamaterialet er definert skal analyseenheten defineres. Dette kan skje på mange nivåer. Eksempelvis ord, fraser, setninger, avsnitt, hele teksten og temaer (Cohen et al., 2007). Når det kommer til lærebøker er det mange deler av den som kan analyseres, eksempelvis bildene, forklaringene eller hvordan læreboka brukes i undervisning (Bakken & Andersson-Bakken, 2021). Jeg har som nevnt valgt direktivene i programmeringsoppgavene i den aktuelle matematikklæreboka.

Videre i prosessen skal det fastsettes eller utvikles koder og kategorier til bruk i analysen (Cohen et al., 2007). I analysen i denne masteroppgaven går dette ut på å lage ett sett med regler som bestemmer i hvilke(n) kategori(er) oppgaven tilhører. Kategoriseringen baseres på den spesifikke oppgavens direktiv til eleven, altså hva oppgaven ber eleven om å gjøre for å kunne løse den.

Når kodene og kategoriene er bestemt fremmer Cohen et al. (2007) selve utføringen av kodingen og kategoriseringen. Deretter utføres selve dataanalysen (Cohen et al., 2007). Jeg bruker dataprogrammet Microsoft Excel for å føre de kvantitative resultatene, se figur 9 og 10. Tilslutt skal analysen oppsummeres og antagelser knyttet til problemstillingen skal gjøres (Cohen et al., 2007). Dette vil jeg kunne utdype gjennomføringen av senere i analyseprosessen, i tråd med induktiv innholdsanalysemetode (Rezat & Sträßer, 2015).

3.4 Kvalitetsvurdering

Kvalitetsvurderingen i en masteroppgave gjøres ved å vise hvordan arbeidet kan karakteriseres i lys av de kvalitetskravene som stilles til gjennomføringen av den. En må mene noe om arbeidets grad av kravene til validitet, reliabilitet og hvorvidt resultatene lar seg generalisere (Dalland, 2020).

3.4.1 Validitet og Reliabilitet

Reliabilitet omhandler studiens resultaters pålitelighet, kontinuitet og replikerbarhet (Cohen et al., 2007). Når det gjelder innholdsanalyser handler reliabilitet om graden av likhet i kategorisering av tilfeller på tvers av observatører (Høgheim, 2020; Silverman, 2015). I mitt forskningsprosjekt er dette et relevant spørsmål da jeg skal kombinere en kvantitativ opptelling med en kvalitativ analyse. For å sikre dette skal jeg anvende en versjon av det Høgheim (2020, s. 216) beskriver som ‘’inter-koder relabilitet: to eller flere koder det samme datamaterialet, og utfallet av de uavhengige arbeidene sammenlignes i ettertid’’. Koderne, altså personene som koder datamaterialet, arbeider etter de samme grunnleggende prinsippene. I mitt forskningsprosjekt vil hovedprinsippet eller regelen være at kategoriene skal knyttes til kjennetegnene på AT. Når koderne er ferdig med kodingen av datamaterialet skal kategoriene og kodene som vi har kommet fram til sammenlignes. Det er veilederen til denne masteroppgaven som er den andre observatøren som sikrer tilstrekkelig grad av enighet i kategoriseringen. Målet er å se hvorvidt det er samsvar mellom kodingene, som altså utgjør graden av relabilitet (Høgheim, 2020).

Validitet eller gyldighet i forskning innenfor samfunnsvitenskapene og lærebokanalyser handler om hvorvidt metoden som anvendes i forskningen er egnet til å undersøke det som er meningen at den skal undersøke (Høgheim, 2020). I en slik kvalitativ forskning der forskeren anvender seg selv i forskningen vil forskerens forventninger, fordommer og forutinntatte oppfatninger ha en innvirkning på forskningens validitet (Cohen et al., 2007). Validitet handler også om at framgangsmåten og funnene skal reflektere studiens formål og gi en representasjon av virkeligheten, og til hvilken grad dette oppnås i studien (Cohen et al., 2007). Avsnittene under tar for seg mine forkunnskaper og forforståelse. Overførbarhet, bekreftbarhet og generaliserbarhet er også viktige aspekter for validitet i kvalitativ forskning (Cohen et al., 2007).

Mine forkunnskaper og forforståelse av innholdet i de sentrale begrepene programmering og AT vil til en grad innvirke analysen i mitt forskningsprosjekt. Jeg har lest meg opp på litteratur knyttet til disse to begrepene på forhånd for å minimere denne innvirkningen. Det at jeg anvender et allerede etablert utgangspunkt i oppgavenes direktiver som bakgrunn for kategoriseringen i analysen distanserer min innvirkning noe fra resultatene i analysen. Likevel er det jeg som til slutt definerer kategoriene og kodene som skal anvendes i analysen.

Resultatene av analysen vil også kunne være bekreftbar da det som analyseres, læreboka er permanent trykt. Noe som gjør det mulig for andre forskere å følge metoden i forskningsprosjektet og få det samme resultatet dersom metoden er godt beskrevet (Høgheim, 2020). Dersom en forsker skulle ha gjort den samme analysen av en annen lærebok ville hen likevel kunne ha fått andre resultater.

Resultatene fra analysen i denne masteroppgaven vil kun gjelde for den læreboka jeg skal analysere. Derfor er ikke utvalget av lærebok representativ. At funnene i forskningsprosjektet mitt kun gjelder den aktuelle læreboka gjør at disse funnene ikke er generaliserbare. Om funnene gjelder for flere tilfeller stemmer altså ikke og kan ikke si noe om eksempelvis matematikklærebøker som en helhet. Dette ser ikke jeg på som et problem da Kvale og Brinkmann (2015) skriver at det ikke er et krav om at forskningen må være generaliserbar i kvalitativ forskning.

3.5 Forskningsetiske betraktninger

Forskningsetikken har som mål å fremme fri, god og forsvarlig forskning. Som forsker har en forskningsetiske forpliktelser og hensyn som må tas. Forskningsetiske retningslinjer bør ligge til grunn i hele forskningsprosessen og bidrar til å utvikle forskningsetisk skjønn og refleksjon, avklare etiske dilemma, fremme ansvarlig forskning og forebygge uredeligheter. Dette gjelder i planlegging og gjennomføring så vel som formidling og eventuell publisering av forskningen (Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, 2021).

En av disse retningslinjene, og sentral i mitt forskningsprosjekt knyttet til de semistrukturerte intervjuene, er hensyn til personer. Hvilket handler om forskerens ansvar overfor alle deltakerne i forskningen. Deres menneskeverd må respekteres og hensyn til deltakernes personlige, integritet, sikkerhet og velferd må tas. Denne forskningsetiske retningslinjen handler også om informert samtykke til deltakelse. I mitt forskningsprosjekt er denne retningslinjen hovedsakelig relevant knyttet til semistrukturert intervju som metode da det er her jeg hovedsakelig har med mennesker å gjøre. Alle intervjupersoner har skrevet under på et standard samtykkeskjema utviklet av Nord Universitet som beskriver hva intervjuet handler om, hva intervjuet skal brukes til og deres rettigheter. Alle personopplysninger i intervjuet blir anonymisert. Intervjuene ble også tatt opp og lagret på en forsvarlig måte da det ble brukt

diktafon-app fra Universitet i Oslo som sikrer dette (Universitet i Oslo, 2022). Andre personer som nevnes i forskningsprosjektet og som jeg forskeren har et ansvar overfor er lærebokforfatterne av den aktuelle læreboka. Privatlivets fred fremmes som en viktig del av forskningsetikken (Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, 2021). Da den informasjonen jeg skal dele om forfatterne er offentlig tilgjengelig og delt på lærebokforlaget sin nettside anser jeg det som greit å vise til i dette forskningsprosjektet. I læreboka kan det også fremkomme verdier og holdninger. Disse har jeg som forsker plikt til å vise respekt overfor (Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, 2021). Dette skal jeg overholde da jeg i mitt forskningsprosjekt kun er interessert i beskrivende data om oppgavene i læreboka. Jeg skal altså ikke analysere noen verdier eller holdninger som eventuelt framkommer i læreboka.

Forskningsfelleskapet er en også en viktig del av de forskningsetiske retningslinjene Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora (2021) beskriver. Innenfor dette er det viktig med åpenhet og redelighet i forskningen, som en forutsetning for faglig utvikling, etterrettelighet og kritikk (Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, 2021). Dette forsøker jeg å etterkomme i forskningsprosjektet ved en tydelig og åpen framgangsmåte, metode, beskrivelse av slutninger og resultat. God henvisningsskikk er også en del av denne åpenheten og redeligheten i forskningen, der det handler om å anerkjenne andres arbeid. Forskning skal bygge videre på andres arbeid og må gjøres med respekt, grundighet og etterrettelighet (Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora, 2021). I mitt forskningsprosjekt henviser jeg til andres forskning og kilder i tråd med dette, etter beste evne. Jeg bruker American Psychological Association (APA) sin referansestil APA 7th i masteroppgaven, i tråd med standarden for referering til kilder innenfor utdanningsforskning og Nord Universitets retningslinjer (American Psychological Association, 2020; Nord Universitetet, 2022). Det bør også her nevnes at jeg oversetter sitatene som tatt med i denne masteroppgaven der det er hensiktsmessig til norsk.

Jeg har også fått godkjent søknad om gjennomføring av forskningsprosjektet av Norsk senter for forskningsdata som skal sørge for trygg og lovlig innsamling, bearbeiding, deling og lagring av data om mennesker og samfunn (Norsk senter for forskningsdata, 2022). se vedlegg 2 og 3.

3.6 Hvorfor semistrukturerte intervju sammen med lærebokanalyse?

Hittil i metodekapitlet har jeg sett på valg av metode, lærebokanalyse som metode og kvalitetsvurdering. Grunnen til at jeg har valgt semistrukturerte intervju med 3 lærere er også som nevnt en form for kvalitetssikring. I denne delen av kapitlet skal jeg se nærmere på dette valget.

I tilknytning til intervjudelen av forskningsprosjektet har jeg formulert forskningsspørsmålet**:**

Hvilke læringsmateriell benytter lærere på ungdomsskolen for å finne programmeringsoppgaver for deres undervisning?

Målet med de semistrukturerte intervjuene er å belyse lærernes bruk og opplevelse av undervisningsmateriell knyttet til undervisning av programmering i matematikk. Dette brukes som en innledende del av masteroppgaven for å sikre praksisnærhet i forskningen og for å få et virkelighetsnært bilde av hvordan programmering i matematikk undervises. Praksisnærhet og et virkelighetsnært bilde av virkeligheten rundt undervisning av programmering i matematikk er spesielt viktig i denne masteroppgaven, da oppgaven handler om hvilke muligheter for AT elever får i arbeid med programmeringsoppgaver. Dette vil være avhengig av undervingen elevene mottar, læringsmateriellet læreren bruker i undervisningen og hvordan dette læringsmateriellet brukes i undervisningen (Elicer & Tamborg, 2022).

I denne masteroppgaven vil semistrukturerte intervju gi et nært forhold til praksis. Dette da det stadig kommer nye læreplaner og undervisningsmateriell som lover å gi elever læring innenfor det som framheves i læreplan. Det er blant annet fordi samfunnet er dynamisk og samfunnet og menneskene i det endrer seg konstant. Samfunnsvitenskap er en vitenskap om mennesker, samfunnet og hvordan de påvirker hverandre (Dalland, 2020; Høgheim, 2020). I fagfeltet matematikkdidaktikk vil det som er ønskelig å oppnå med undervisning og dermed det som forskes på innenfor disiplinen, variere utfra en samfunnskontekst, ettersom at matematikkdidaktikk er et samfunnsvitenskapelig felt. Dermed vil for eksempel spørsmålet om den beste eller mest optimale måten å undervise programmering i matematikk på, være avhengig av hvem du spør, hvor undervisningen skal foregå, når undervisningen skal ta plass og utfra hvilke mål undervisningen skal ha. Virkeligheten knyttet til undervisning av tema kan endre seg utfra skapelsen av ny kunnskap som for eksempel belyser sider av den gjeldende normen for undervisningen. Det er det som er visjonen med denne masteroppgaven, altså sette

lys på nåværende praksis slik at den kan utvikles. For å få til dette må jeg først og fremst undersøke virkeligheten rundt undervisning av programmering i matematikk, og nærmere bestemt hvordan emnet undervises. Det blir derfor logisk å søke innsikt hos lærerne som driver med undervisning.

I aspekter knyttet til forskningsintervjuet som utformingen av intervjuguide og transkribering av intervjuene har jeg i denne masteroppgaven tatt utgangspunkt i Kallio et al. (2016) sitt rammeverk for semistrukturerte intervjuer. Da denne delen av masteroppgaven er ment som en ''stemme fra praksis'' har jeg ikke beskrevet dette nærmere i oppgaven. Sammendrag av intervjuene er lagt til som vedlegg (se vedlegg 1).

4.0 Analyse

I dette kapitlet skal jeg presentere hva som ligger til grunn for kategoriseringen og lærebokanalysen i masteroppgaven. I kategoriseringen tar jeg utgangpunkt i oppgavens (oppgavetekstens) direktiv, altså det oppgaven ber eleven om å gjøre, og plasserer disse i kategorier og eventuelle underkategorier. For analyse av kategoriene bruker jeg instrumentteori, nærmere bestemt instrumentalisering og dens grener. Disse grenene er aktiviseringer, begrensinger, potensialer og anbud (Trouche, 2020). Jeg bruker operasjonelle definisjoner på instrumentaliseringsgrenene i analysen som et rammeverk for analyse opp mot kjennetegnene på AT (Csizmadia et al., 2015). Til slutt i kapitlet presenteres en kvantifisering av resultatene til lærebokanalysen.

Oppgavene i læreboka er delt inn i kategorier og underkategorier utfra hva direktivet i oppgaveteksten er. Oppgaver der direktivet i oppgaveteksten har samme betydning er også plassert i samme kategori. Underkategorier er til i kategorier der direktivene har ulike potensielle og anbudsmessige kjennetegn på AT. Analysen av direktivene inneholder en analyse som er generell for alle oppgavetekstene i kategorien. Deretter er eventuelle likheter og ulikheter i underkategoriene adressert. For å unngå gjentagelser nevnes kjennetegnene på AT som er felles for kategorien under hovedkategorien og det som er unikt for underkategoriene under hver enkelt underkategori. Det vil også være med en analyse av oppgaveteksten i enkeltoppgaver i de mest forekommende kategorier og underkategorier for å vise hva som ligger til grunn for tilknytninger mellom direktiv og kjennetegn på AT. Både de generelle analysene og analysene av enkeltoppgaver vil følge samme struktur med:

- Innledning som skal gi generell informasjon om direktivet og oppgaveteksten som er nyttig for leseren å vite. Samt i analysene av enkeltoppgaver inneholde et løsningsforslag av analyseoppgaven.

- Aktivisering skal navngi hvilke tema oppgavens direktiv har og kan ha.

- Begrensninger. Her undersøkes hvilke forpliktelser og føringer oppgavens direktiv gir utover oppgavens tema.

- Potensialer skal indentifisere hva som er mulig å oppnå av kjennetegn på AT innenfor utføringen av direktivet som analyseres, utfra rammekonteksten identifisert i innledningen, aktiviserings- og begrensninger delen av analysen.

- Anbud skal identifisere det som er nødvendig av kjennetegn på AT for å besvare oppgavens direktiv.

- Konklusjon skal oppsummere analysen og inneholder eventuelle kommentarer og merknader fra analyseprosessen.

Deretter er bi-direktivene adressert. Disse er knyttet til andre direktiv og finnes færre av i oppgavene i læreboka. Bi-direktivene er analysert generelt imot slutten av analysekapitlet. Helt i slutten av kapitlet presenterer jeg diagrammer over kjennetegnene funnet i direktivenes potensialer og anbud.

4.1 Oversikt kategorier

Jeg begynner med en oversikt over kategoriene og underkategorienes kjennetegn på AT potensialer og anbud, plassert i rekkefølge utfra antall i parentes:

Kategori/-underkategori:	Potensialer:	Anbud:
Lag (83) + (18)	Abstraksjon Algoritmisk tankegang	Abstraksjon Algoritmisk tankegang
-Kalkulator (76) +(12)	Automatisering Dekomponering Generalisering	Automatisering Generalisering
-Kommando (7) +(6)	-	-

Forklar (32)	Abstraksjon Algoritmisk tankegang Dekomponering Generalisering	Abstraksjon Algoritmisk tankegang
-Løkker (15)	Automatisering	Automatisering
-Feilsøk (2)	Feilsøk	Feilsøk

Endre (28) – (18)	Abstraksjon Algoritmisk tankegang Dekomponering Generalisering	Algoritmisk tankegang Dekomponering

Beskriv (25)	Algoritmisk tankegang Feilsøking	Feilsøking

4.1.1 Oversikt bi-direktiver

Direktiver det finnes forholdsvis få av i programmeringsoppgavene i læreboka og som gjerne har lite tilknytning til kjennetegn på AT er kategorisert som bi-direktiver. Se oversikten nedenfor:

Kjør (25)	-	-

Test (9)	Generalisering	-

Skriv inn (5)	Algoritmisk tankegang	-

Gjett (2)	Abstraksjon Algoritmisk tankegang Dekomponering Feilsøking	-

4.2 Analyse av direktivene

I denne delen av kapitlet skal direktivene og tilhørende enkeltoppgave(r) analyseres. For å unngå 4 nivåer med overskrifter er analysene av hovedkategoriene gitt et høyere overskriftsnivå for å gjøre det mer oversiktlig. Dermed får hovedkategoriene overskriftsnivå 2 og tilhørende analyse av enkeltoppgave(r) får overskriftsnivå 3. I analyse av enkeltoppgaver presenterer jeg også løsningsforslag jeg har laget selv. For å lage disse har jeg brukt Jupyter Notebook som blant annet inneholder en Python-leser (Project Jupyter, u.å).

4.3 Lag

Lag er et direktiv som finnes i 91 av programmeringsoppgavene i den analyserte læreboka. I 80 av disse er det også det eneste hoved-direktivet. Direktivet **Lag** handler i alle tilfellene i programmeringsoppgavene i læreboka om aktiviteten å programmere noe i Python-

programmeringsspråket. Oppgavene med direktivet **Lag** i boka er delt opp i to underkategorier. Disse er *Kalkulator* og *kommando*. I underkategorien *kalkulator* er oppgaver der direktivet handler om å programmere et program som regner ut noe plassert, samt oppgaver der en eller flere løkker forpliktes eller er nærliggende å anvende. Mens i underkategorien *kommando* handler det om å programmere et program som ikke skal regne ut noe, men skal inneholde noen spesifikke kommandoer, gjerne for å skape en forståelse for kommandoens egenskaper til videre bruk. Her er det hovedsakelig snakk om print-kommandoen og input-kommandoer.

Når det kommer til aktivisering, har begge disse underkategoriene tema programmering og kommandoer til felles. Programmering da direktivet i alle disse oppgavene er en form for ''Lag et program''. Hvilket tilsier at eleven skal programmere i arbeid med disse oppgavene. Kommandoer er også et tema som er til stede i alle disse oppgavene i ulik grad da alle programmene som programmeres i arbeid med direktivet inneholder en eller flere kommandoer.

Den eneste felles forpliktelsen knyttet til dette direktivet er at alle programmene som programmeres skal skrive noe til skjermen. Det er ofte også mange føringer og forpliktelser i tilknytning direktivet, noe som gjør at det i mange av oppgavene i praksis kun finnes en måte å løse oppgaven på.

Som nevnt tidligere må en programmere for å følge direktivet **Lag**. Felles for all programmering er abstraksjon og algoritmisk tankegang. Dette da direktivet krever at en abstraherer direktivet og omgjør det til tydelig definerte instrukser, i form av variabler, kommandoer og strenger, som er skrevet rett og plassert i riktig rekkefølge for at programmet skal fungere på ønsket måte.

Felles for *Lag*-kategoriens anbud er algoritmisk tankegang og abstraksjon da disse må i ulik grad gjennomføres for å programmere, hvilket er premissen bak direktivet ''Lag et program'' i programmeringskontekst.

Temaet matematisk innhold er forskjellen når det kommer til aktiviseringsdelen av analysene i kategorien. I *kommando*-underkategorien er kun temaet programmering og kommandoer, samt i oppgavene som omhandler input-kommandoen finnes tema variabler. I *kalkulator*-kategorien finnes også tema matematikk. Hva dette matematiske innholdet er varierer fra mønster knyttet til løkker til aritmetikk, geometri og andre matematiske emner som brukes i kalkulatoren som skal programmeres i arbeid med det enkelte direktivet.

Forpliktelsene og føringene i *kommando*-underkategorien er knyttet til innholdet i strenger og navn på variabler. I *kalkulator*-underkategorien handler forpliktelsene i større grad om hvordan kalkulatoren skal fungere, da gjerne i form av hva brukeren av programmet kan skrive inn. Betingelser for løkken som skal brukes i programmet, antallet gjennomkjøringer knyttet til for-løkker eller start og slutt betingelser for while-løkker, er også en forpliktelse en finner i oppgavene i denne underkategorien.

Forskjellene i potensialene mellom underkategoriene er automatisering og generalisering. Der disse kjennetegnene på AT kun finnes i *kalkulator*-underkategorien.

I underkategorien *kalkulator* skal elevene i mange av oppgavene utøve matematikk ved å programmere kalkulatorprogram. Arbeidet med programmering av et løkke- og kalkulatorprogram er programmering med generaliserende innhold. Elevene blir utsatt for en framvisning av hva som endrer seg og hva som forblir likt mellom utregningene. Disse kalkulatorene er også mer spesifikk enn en fysisk kalkulator. Dette da kalkulatoren de programmerer i arbeidet med disse oppgavene regner ut noe spesifikt på samme måte hver gang. Dette ved at en eller flere av variablene er forhåndsbestemt gjennom koden i kalkulatorprogrammet. Fysiske kalkulatorer har en helt annen bredde i hva de kan regne ut, da det er brukeren som bestemmer alle variablene. Noe som gjør at eleven potensielt kan generalisere spesifikke konstanter i utregninger knyttet til tema for kalkulatoren som programmeres.

Eleven arbeider også med automatisering i disse oppgavene da brukeren av kalkulatoren ikke behøver å regne noe ut selv, kun skrive inn enkelte variabler. På lignende vis som med generalisering blir elevene utsatt for arbeid med noe som automatiserer noe, altså at noe skjer automatisk for brukeren.

Forskjellen i anbudet mellom underkategoriene er automatisering og generalisering. I underkategorien *kalkulator* utsettes elevene for arbeid med generaliserings- og automatiseringsprosesser, der løkken og eller kalkulatoren de programmerer skal kunne gjøre en automatisk handling med grunnlag i en generell formell utarbeidet av eleven. Hvilket er et matematisk innhold som ikke finnes i *kommando*-underkategorien.

Lag er det direktivet som finnes flest av i programmeringsoppgavene i læreboka. Jeg har delt opp denne kategorien inn i to underkategorier. *Kalkulator* og *kommando*. Der underkategorien *kalkulator* inneholder løkker og **Lag** direktiv som omhandler å programmere en kalkulator, gjerne knyttet til et spesifikt tema. Underkategorien *kommando* inneholder oppgaver der

direktivet er å programmere program med print-kommandoer og input-kommandoer. Disse oppgavene krever dermed gjerne et lavere nivå for å løse. Disse underkategoriene har også mye tilfelles.

Når det kommer til aktivisering og begrensinger har begge underkategoriene tema programmering og kommandoer til felles og programmene knyttet til direktivet skal skrive noe til skjermen.

Potensialene og anbudene til felles for alle oppgavene i kategorien er abstraksjon og algoritmisk tankegang. For at programmene i disse oppgavene skal fungere på ønsket måte må direktivet omgjøres slik at det kan bli lest og kjørt av en Python-leser. Hvilket krever tydelig definerte instrukser, knyttet til kjennetegnene abstraksjon og algoritmisk tankegang.

Det er også noen ulikheter mellom disse underkategoriene. Når det kommer til aktivisering, finnes et matematisk innhold kun til stede i *kalkulator*-underkategorien. Føringene og forpliktelsene som finnes i oppgavene er også forskjellige da forpliktelsene i *kommando*-underkategorien handler om innholdet i strenger og navn på variabler. I *kalkulator*-underkategorien handler dette om betingelser for løkker og egenskaper til programmering av kalkulatorer, knyttet til generalisering og automatisering som potensial og anbud.

4.2.1 Analyse Kalkulator oppgave 7.24b

Oppgaveteksten for denne oppgaven er: ‘’Lag et program som lar brukeren skrive inn diameteren til fem ulike sirkler. Programmet skal hver gang brukeren har skrevet inn en diameter, regne ut og skrive omkretsen til denne sirkelen. Prøv å få med forklarende tekster i svaret’’. Oppgaven nevner også at eleven kan bruke 3,14 for (PI).
Et løsningsforslag til denne oppgaven er:

```
In [3]: print('Gitt π = 3.14')
        for i in range(1,6,1):
            d = float(input('skriv diameteren til sirkel'))
            print('sirkel nr:',i)
            o = d*3.14
            print('omkretsen til sirkel',i,'kan regnes ut slik',d,'* 3.14')
            print('og svaret er:',o)

Gitt π = 3.14
skriv diameteren til sirkel3
sirkel nr: 1
omkretsen til sirkel 1 kan regnes ut slik 3.0 * 3.14
og svaret er: 9.42
skriv diameteren til sirkel3
sirkel nr: 2
omkretsen til sirkel 2 kan regnes ut slik 3.0 * 3.14
og svaret er: 9.42
skriv diameteren til sirkel1
sirkel nr: 3
omkretsen til sirkel 3 kan regnes ut slik 1.0 * 3.14
og svaret er: 3.14
skriv diameteren til sirkel2
sirkel nr: 4
omkretsen til sirkel 4 kan regnes ut slik 2.0 * 3.14
og svaret er: 6.28
skriv diameteren til sirkel3
sirkel nr: 5
omkretsen til sirkel 5 kan regnes ut slik 3.0 * 3.14
og svaret er: 9.42
```

Figur 1- Løsningsforslag oppgave 7.24b

Linje 1 i dette programmet skriver til skjermen at PI tolkes som 3,14 av programmet.

Linje 2 i programmet er en ‘’for-løkke’’ som kjører koden med innrykk nedenfor 5 ganger. Løkken er koblet til variabelen ‘’i’’.

Linje 3 gir brukeren mulighet for å skrive til programmet med teksten ‘’skriv diameteren til sirkelen’’. Leseren vil koble det brukeren skriver inn, altså kommandoen ‘’input’’ til variabelen ‘’d’’ for diameter. Python-leseren vil også tolke det brukeren skriver inn som desimaltall derav ‘’float’’.

Linje 4 i programkoden skriver ‘’sirkel nr’’ og antallet av hvor mange ganger ‘’for-løkken’’ har kjørt knyttet til ‘’i’’ variabelen.

Linje 5 i programmet fastsetter verdien for variabelen ‘’o’’ som variabelen ‘’d’’ multiplisert med 3,14.

Linje 6 i programkoden skriver til skjermen en forklaring på hva programmet regner ut. I den siste linjen av programmet skrives svaret og verdien av variabelen ‘’o’’ til skjermen.

Tema i direktivet er programmering med kommandoer og variabler, matematikk og geometri med sirkler og sirklers egenskaper. Når det kommer til programmering, kommandoer og variabler skal eleven i arbeid med direktivet programmere et program som skal inneholde variabler, løkke- og ‘’print’’ kommandoer, samt strenger knyttet til det valgfrie direktive om å inkludere forklarende tekster til programmet. Det geometriske og matematiske i direktivet handler om at programmet skal ha funksjonen å regne ut sirklers omkrets.

Den første forpliktelsen i oppgaveteksten er: ‘’ … som lar brukeren skrive inn’’. Programmet som eleven skal programmere må altså kreve at brukeren skriver inn verdien til en diameter for at programmet skal fungere slik det er ment. Direktivet forplikter også at det er spesifikt diameteren tilhørende sirkelen som skal brukes til å regne ut sirkelens omkrets. Den neste forpliktelsen direktivet gir er ‘’fem ulike sirkler’’. Her begrenses hvor mange sirkler som kan undersøkes. Videre i oppgaveteksten forpliktes det også en bestemt rekkefølge på operasjonene programmet skal utføre ved ‘’Programmet skal hver gang brukeren har skrevet inn en diameter, regne ut og skrive omkretsen til denne sirkelen’’. Altså når brukeren skriver inn verdien av en diameter skal programmet regne ut og skrive omkretsen til denne sirkelen til skjermen.

Eleven må abstrahere formålet med programmet. Altså omgjøre formålet med programmet til en algoritme som kan leses av Python-leseren. Direktivet omhandler som sagt å programmere et program som regner ut omkretsen av 5 sirkler ved at brukeren skriver inn diameteren til sirklene. Eleven må altså klare å omgjøre og minimere direktivet til variabler og kommandoer. Variabler for diameteren, omkretsen og løkken. Kommandoer for å la brukeren skrive inn verdier for diametere, kommandoer for at programmet skal skrive forklaring og svar til skjermen, samt kommandoer for løkken.

Disse må også framkomme som tydelig definerte instruksjoner trinn-for-trinn til leseren ut fra algoritmen, knyttet til algoritmisk tankegang.

For å lage programkoden må eleven formulere en formel, en generalisering, for utregningen av omkretsen på en sirkel ved hjelp av en fast konstant, altså 3,14 og en variabel som brukeren skriver inn.

Ved å programmere dette programmet automatiseres regneoperasjonen knyttet til å finne omkretsen av en sirkel når diameteren er kjent, i det ferdige programmet. Programmet multipliserer automatisk det brukeren skriver inn som diameteren til en sirkel med (PI) og regner ut omkretsen av sirkelen som skrives til skjermen, i henhold til direktivet. Dersom eleven velger å anvende en løkke i programmet sitt kan være med å forsterke potensiale knyttet til generalisering da løkker også er en form for automatisering av kode. I dette tilfelle fungerer den slik at eleven slipper å skrive inn den samme koden 5 ganger, men bare skrive koden inn en gang.

Abstraksjon og algoritmisk tankegang er nødvendig for å gjøre om oppgavens direktiv til et faktisk program som Python-leseren kan kjøre. Det generaliserende aspektet av utformingen av en formel som gir mulighet til å programmere et program som kan automatisk regne ut sirklers omkrets er også med i oppgavens anbud.

Dette er en oppgave der direktivet går ut på å programmere et program som kan regne ut sirklers omkrets. Temaene for dette direktivet blir derfor programmering med kommandoer og variabler og matematikk, nærmere bestemt geometri. Direktivet inneholder også flere forpliktelser utover tema. Disse omfatter både programmets virkemåte, variablene som skal brukes i programkoden og begrensninger knyttet til antall utregninger. Det viser seg at potensiale og anbudet for dette direktivet er det samme. Hvilket er algoritmisk tankegang og Abstraksjon, knyttet til programmeringen av et fungerende Python program utfra oppgaveteksten. Samt generalisering og abstraksjon ved å formulere en formel i form av programkode som kan gjennomføre en automatisert utregning.

Potensialer og anbudet knyttet til direktivet kan være de samme kjennetegnene på AT fordi potensialene i arbeid med direktivet ikke omfavner flere kjennetegn på AT utover de som må til for å gjennomføre direktivet, slik som i andre direktiv. Hva analysedelene anbud og potensialer rommer er noe som kommer til å bli undersøkt og drøftet nærmere i drøftingsdelen av masteroppgaven.

4.3.2 Analyse Kommando oppgave 2.83

Dette er en oppgave med direktivet ‘’Lag et program som skriver ‘’Hei’’ til skjermen’’. Et løsningsforslag til denne oppgaven kan være:

```
In [1]: print('Hei')
        Hei
```

Figur 2- Løsningsforslag oppgave 2.83

Dette programmet består av 1 linje med kode. I denne linjen finnes en print-kommando som, når programmet kjøres, skriver strengen ‘’Hei’’ til skjermen.

Direktivet har programmering og print-kommando som tema. Dette kommer av at for å programmere et program som skriver ‘’Hei’’ til skjermen trengs det i utgangspunktet kun en print-kommando.

Den eneste forpliktelsen i direktivet, utover oppgavens tema er at det som skal skrives til skjermen er ‘’Hei’’.

For å følge oppgavens direktiv må eleven til en viss grad abstrahere direktivet for å gjøre det om til nøyaktige instruksjoner som kan leses av Python-leseren. I denne oppgaven må altså eleven vite at det er en print-kommando som må til for å skrive noe til skjermen, samt hvordan den presist skrives inn som kode med ‘’Hei’’ som valgt streng for å følge direktivet.

For å gjennomføre direktivet i oppgaven må eleven abstrahere direktivet om til en instruks som kan leses av Python-leseren etter ønsket sluttprodukt som er å skrive ‘’Hei’’ til skjermen.

Dette er et program der direktivet er å lage et program som skriver ‘’Hei’’ til skjermen. Det krever i utgangspunktet kun en print-kommando. Dermed er programmering og print-kommandoen tema til direktivet. Potensialer og anbud i til dette direktivet er i likhet med den andre analyserte **Lag-**direktiv det samme. Hvilket er abstraksjon og algoritmisk tankegang. Dette da eleven må til en viss grad abstrahere direktivet om til en nøyaktig instruks som python-leseren kan kjøre.

4.4 Forklar

32 av oppgavene i læreboka inneholder direktivet **forklar**. Direktivet finnes ofte i samme oppgavetekst som direktivet **kjør**. **Forklar**-direktivet kan sies å være mer gjennomgående enn direktivet **beskriv**. Der direktiv **beskriv** handler om å gjengi overfladiske egenskaper eller

forskjeller, krever direktivet **forklar** en forklaring av egenskaper. For å kunne forklare egenskaper kreves en forståelse for det som skal forklares. I denne typen oppgaver i læreboka er det gjerne program og komponentdeler i program som skal forklares, samt forskjeller mellom program. Jeg har laget 2 underkategorier til denne kategorien, **løkker** og **feilsøk**. Der hovedforskjellen er at kjennetegnet på AT feilsøk er til stede i oppgavene i **feilsøk**-underkategorien.

Det som er særegent for oppgaver i **feilsøk** under-kategorien er at direktivet handler om å gjenkjenne en feil i programkode, forklare hvorfor programmet ikke fungerer eller hvorfor programmet ikke fungerer på ønsket måte. I noen **feilsøk-**direktiv skal eleven forklare hvorfor noe er feil slik at det spesifikke programmet som behandles i oppgaven ikke fungerer eller ikke fungerer slik det er ment å fungere. I de resterende oppgavene, i denne underkategorien, skal eleven finne og rette opp feilen(e) slik at programmet fungerer på ønsket måte. Dermed skal elevene aktivt programmere i den sistnevnte delen av underkategorien, men dette er ikke et krav i den første. Denne forskjellen blir tatt hensyn til ved at de oppgavene der direktivet inkluderer å aktivt programmere telles også i **lag** kategorien. I **forklar** hovedkategorien finnes oppgavene som handler om å forklare hvordan programkode eller et helt program fungerer. For å få til dette må eleven gjerne forstå sammenhengen mellom komponentdelene i programmet. Oppgavene deles også opp i underkategorien **løkker** som inneholder oppgaver med matematisk innhold i form av løkker eller kalkulatorprogram. Et direktiv i denne kategorien kan være i begge underkategoriene.

Begge underkategoriene har programmering med kommandoer og variabler som tema. I disse oppgavene er det enten et helt program, deler av et program og sammenhenger i programkode som skal forklares eller feilsøkes.

Tilfelles har direktivene i **forklar** kategorien forpliktelser der eleven må ta utgangspunkt i å forklare noe ved den spesifikke programkoden som blir gitt av læreboka.

Algoritmisk tankegang er et potensial i disse oppgavene. Dette da det i arbeid med disse direktivene kreves en viss forståelse for programmeringsspråket Python, og dens avhengighet av tydelig definerte instrukser i form av koder, for å kunne gjennomføre direktivet om å forklare.

Eleven må også videreføre denne forståelsen for å kunne abstrahere programmets virkemåte om til en forklaring som en uten forståelse for koden kan forstå.

Dekomponering er et potensial i forklar-oppgavene på grunn av muligheten for å anvende dekomponering av programkode for å lettere kunne forklare egenskaper, hvilket er direktivet i disse oppgavene. I noen oppgaver skal også eleven aktivt dekomponere for å forklare utfra direktiv som tilsier at eleven skal forklare et program ‘’linje for linje’’.

Når eleven skal, utfra direktivet, forklare en egenskap krever det en viss forståelse for den egenskapen. Egenskapen har dermed lettere for å bli gjenkjent i videre forklaring og oppgaveløsning av samme art, utover generell mengdetrening.

Direktiv-kategorien har kjennetegnet abstraksjon og algoritmisk tankegang tilfelles. Dette på grunn av at for å kunne forklare en virkemåte eller finne en feil i programkode må en bruke eller utvikle forståelse for Pythons egenskaper. Videre fra denne forståelsen må også eleven abstrahere programmets virkemåte om til en forklaring som en uten forståelse for koden kan forstå.

Begge under-kategoriene har som nevnt programmering og kommandoer som tema. Noen oppgaver har også et matematisk innhold, der direktivet er å forklare et program og matematikken i det eller en mulighet til å forklare programmets virkemåte utfra matematiske sammenhenger som blir skrevet til skjermen, knyttet til løkker-underkategorien.

I **løkker**-underkategorien finnes oppgaver der direktivet handler om å forklare løkker og eller kalkulatorprogram. Løkker er en form for automatisering der løkken gjentar koden knyttet til den et bestemt antall ganger eller til vilkårene satt for løkka er nådd. Kalkulatorprogram er også en form for automatisering der brukeren skriver inn for eksempel en variabel og programmet manipulerer variabelen på en bestemt måte automatisk, utfra programkoden. Som nevnt tidligere må en ha en viss forståelse for programkoden for å kunne forklare den. I dette tilfelle gjelder det automatiseringen i løkker og kalkulatorprogram.

Å systematisk lete etter og eventuelt rette opp feil er som nevnt i teoridelen en del av aktiviteten feilsøking. Direktivene i underkategorien **feilsøk** handler som nevnt over om å gjenkjenne feil i programkode, forklare hvorfor programmet ikke fungerer eller ikke fungerer på ønsket måte. Dermed kan en si at direktivet kan ses på som en omformulering av ordet ‘’’feilsøk’’.

Karakteristisk for underkategorien **feilsøk** er at anbudet i alle oppgavene i kategorien også er feilsøk.

4.4.1 Analyse Forklar oppgave 4b s.135

Oppgaveteksten for denne oppgaven er: ‘’Forklar hva programmet gjør, linje for linje’’. Programmet oppgaven refererer til ser slik ut når det kjøres inkludert programkode:

```
In [1]: sum = 0
        for tall in range(5):
            sum = sum + tall
            print(sum)

        0
        1
        3
        6
        10
```

Figur 3- Ferdig skrevet programkode som utgangspunkt for oppgave 4b s. 135

En måte å løse oppgaven på kan være:

Linje 1: definerer variabelen ‘’sum’’ som 0

Linje 2: Er en ‘’for-løkke’’ tilknyttet variabelen ‘’tall’’ som kjører 5 ganger fra 0-4.

Linje 3: er satt med innrykk i løkken og redefinerer variabelen ‘’sum’’ som ‘’sum’’ addert med løkkevariabelen ‘’tall’’ for hver del av løkken som får verdiene 0,1,2,3 eller 4 ettersom hvor mange ledd løkken har kjørt, noe som øker med 1 for hvert ledd.

Linje 4: Er også satt med innrykk til løkken og skriver variabelen ‘’sum’’ til skjermen for hver del av løkken. Dermed skrives det forrige leddet, som er et resultat av verdien knyttet til ‘’sum’’ variabelen og verdien av det forrige løkkeleddet, addert med verdien av det gjeldende løkkeleddet til skjermen.

Dette direktivet inneholder teoretisk programmering og mønstre og addisjon i matematikk. Teoretisk programmering er tema i direktivet da det handler om å forklare programkoden i og virkemåten til et ferdig program. Mønstre og addisjon er et tema i arbeidet med dette direktivet da det er et addisjonsmønster som skal forklares i denne oppgaven. Programmets virkemåte kan også forklares utfra mønstret som skrives til skjermen.

Oppgavens direktiv forplikter eleven å forklare programmets virkemåte utfra kodelinjene til programmet ‘’ … linje for linje’’. Direktivene forplikter også eleven å ta utgangspunkt i det spesifikke programmet som blir gitt i læreboka.

Jeg bruker gjennomføringen av direktivet knyttet til første kodelinje som eksempel her: sum = 0 må gjøres om til en forklaring som er enklere å forstå for noen som ikke forstår hva kodene betyr som for eksempel: Linje 1: definerer variabelen ‘’sum’’ som 0. For å følge direktivet må eleven altså abstrahere kodene.

Eleven vil kunne bruke kodelinjene som separate komponentdeler med tilknytning til hverandre for å forklare sammenhengene i programkoden som vist i løsningsforslaget over, knyttet til kjennetegnet på AT dekomponering.

Er også en del av oppgavens potensiale da oppgavens direktiv ‘’…,linje for linje’’ tilsier at eleven skal forklare programmets instruksjon trinn-for-trinn. I dette tilfelle er som sagt de enkelte kodelinjene trinnene. Altså kodelinje 1 er det første trinnet, kodelinje 2 er det andre trinnet og så videre. For å kunne gjøre dette må eleven ha en viss forståelse for programmeringsspråkets avhengighet av riktig rekkefølge på programkode, samt at programkoden er skrevet korrekt.

Den forhåndsbestemte løkken i programmet gjør at eleven må forklare løkken i arbeidet med direktivet i oppgaven. Løkker er som tidligere nevnt en automatisering.

Arbeidet med direktivet er generaliserende. Dette er på grunn av at eleven her arbeider med å forklare egenskapene til programmet, noe de kan generalisere. I dette direktivet vil det lette arbeide med å forklare programmets virkemåte dersom eleven ser og forstår triangelmønstret som skrives til skjermen. Triangelmønster er et mønster der regelen er å legge til 1 mer for hvert ledd. Det vil si: 1 + 2 = 3, 3 + 3 = 6, 6 + 4 = 10. Eleven trenger ikke nødvendigvis å vite hva triangelmønstret heter, men vil ha nytte av å kunne knytte mønstret opp mot programkoden i forklaringen sin, i arbeid med direktivet.

For å følge direktivet må eleven anvende abstraksjon og algoritmisk tankegang. Eleven må abstrahere kodelinjene om til en forklaring som kan forstås av en som ikke forstår koden. Algoritmisk tankegang er knyttet til forståelse til programmet. Der eleven selv skal forstå programmet, i tilknytning til å kunne forklare det. En kan ikke forklare noe en selv ikke forstår. Det er også en løkke i programmet, noe som gjør at eleven må forklare egenskapene dens, knyttet til automatisering.

Dette er en oppgave der eleven skal forklare kodelinjene til et program. Tema i direktivet er teoretisk programmering og matematikk med addisjon og mønstre der eleven skal forklare et spesifikt ferdig kodet Python-program og mønstret i det. Videre forplikter oppgaven at eleven forklarer hva kodelinjene i det spesifikke programmet gjør. Potensialet utover anbudet i arbeid med dette direktivet er dekomponering og generalisering. Der eleven vil ha stor nytte av å se programmet utfra komponentdelene i programmet og kunne se sammenhengene i det. De vil også ha stor nytte av å forstå sammenhengen mellom mønstret programmet skriver til skjermen og programkoden, for å kunne forklare programmet og dermed følge direktivet. Anbudet i direktivet er abstraksjon og algoritmisk tankegang. Dette da for å kunne følge direktivet om å forklare programmets virkemåte må eleven forstå hva programkoden gjør gjennom bruk av algoritmisk tankegang. For deretter å kunne abstrahere virkemåten slik at noen som ikke forstår koden kan gjøre det gjennom forklaringen.

4.4.2 Analyse Feilsøk oppgave 2.90b

Dette er en av programmeringsoppgavene i Matemagisk 8 der direktivet er å finne feil i et ferdig skrevet program som gjør at programmet ikke fungerer. Hele oppgaven består av 3 program med små ulikheter i kodene. Disse programmene skal doble et tall som brukeren skriver inn. 2.90b er knyttet til det første programmet der oppgaveteksten er: ‘’Forklar hvorfor det første programmet ikke fungerer’’. Dersom en skriver inn programmet i en Python-leser og kjører det er dette resultatet:

```
In [45]: #input fra brukeren lagres som tekst
         tall = input('skriv inn et tall')
         svar = 2 * tall
         print('Det dobbelte av tallet er', svar)

         skriv inn et tall2
         Det dobbelte av tallet er 22
```

Figur 4- Ferdig skrevet programkode som utgangspunkt for oppgave 2.90b

Her blir det brukeren skriver inn tolket av leseren som en streng. Ved datatype streng brukes dobling som oppfordring for å lage en streng som har to ganger opprinnelig tekst. Python forstår det som at svaret skal skrives til skjerm to ganger. Dersom brukeren skriver inn 2

skriver Python 22 til skjermen da 22 er det samme som tallet 2, 2 ganger etter hverandre. I analysen vil jeg se bort fra det matematiske i dette kalkulatorprogrammet. Dette da det er feilsøking som er fokuset og det matematiske innholdet dekkes av **løkke**-underkategorien.

Tema i denne oppgaven er teoretisk programmering og nærmere bestemt input-kommandoen i Python. Oppgavens direktiv er som nevnt å forklare hvorfor programmet over ikke fungerer og i innledningen fant jeg at grunnen til at programmet ikke fungerer slik det skal er input-kommandoen.

Forpliktelsen som er mest sentral i oppgavens direktiv er **forklar**. I dette tilfelle når eleven skal forklare må de spesifisere hva i programmets kode og dens egenskaper som gjør at programmet ikke fungerer slik det er ment for å utføre direktivet. Det er også dette spesifikke programmet som skal feilsøkes utfra: ‘’… det første programmet..’’ i oppgaveteksten. Hvilket forplikter eleven å feilsøke det første programmet i oppgaven.

Eleven kan spore hva som gjør at feilen oppstår. Knyttet til dette direktivet kommer feilen fra input-kommandoen som gir brukeren mulighet til å skrive inn, se linje to i skjermbildet av programmet over. Eleven kan teste ulike egenkomponerte løsningsforslag for å løse feilen. De kan for eksempel endre det betydningen av det brukeren skriver inn til heltall. Eleven kan også forutse hva utfallene av disse rettelsene blir og verifisere de ved å først tenke seg til om og hvorfor rettelsen deres blir rett og deretter kjøre programmet for å verifisere om det fungerer eller ikke. Alt dette kan i tillegg gjøres ved hjelp av logisk tenkning og systematisk feilsøking. Dette ved at eleven for eksempel anvender et system for feilsøkingen der de undersøker linje for linje i koden for seg selv for å lettere gjennomføre direktivet.

Knyttet til direktivet om å forklare må eleven abstrahere programkode om til en forståelig forklaring i arbeidet med direktivet **forklar** funnet i oppgaveteksten.

For å suksessfullt feilsøke programmet i oppgaven må eleven vise en forståelse for Pythons egenskaper og avhengighet av nøyaktig beskrevet instrukser for at programmet skal fungere på ønsket måte. I dette tilfelle er det input-kommandoen eleven må vise forståelse for. De må altså forstå at koden input() i dette programmet skriver strengen i input-kommandoen to ganger til skjermen når programmet kjøres.

Ved å undersøke og feilsøke ulike bestanddeler av programmet kan også kjennetegnet på AT, Dekomponering brukes i arbeidet med direktivet. I det aktuelle programmet kan koden deles opp i ulike bestanddeler. For eksempel kan eleven feilsøke alle de tre linjene med kode. Eller

de kan dele opp programmet utfra dens variabler og funksjoner for å gjøre arbeidet med direktivet lettere.

I arbeid med dette direktivet kan eleven bruke og utvikle forståelse for egenskapene til kommandoen ‘’input(‘Tekst’)’’. Dette ved at de får erfart flere av input-kommando versjoner på et nivå slik at de er i stand til å videreføre det til senere programmering og oppgaveløsning.

Dersom en ser på minimumskravet for å følge oppgavens direktiv trenger eleven kun å forklare hva som gjør at programmet ikke fungerer. De trenger dermed ikke å løse feilen i programmet bare identifisere det gjennom kjennetegnet feilsøking. Eleven må også forstå hva Pythons input-kommando gjør og hvorfor det blir feil å skrive kommandoen på måten den er skrevet i programmet for å få ønsket resultat, for å kunne finne og formulere feilen, knyttet til abstraksjon og algoritmisk tankegang.

Direktivet inneholder tema teoretisk programmering og input-kommandoen. Oppgaven forplikter også eleven å spesifisere hva som gjør at et spesifikt program ikke fungerer gjennom en forklaring av programmets kode. En ser videre at oppgaven har potensiale til å fremme flere kjennetegn på AT. Hele definisjonen av kjennetegnet ‘’feilsøk’’ kan fremmes da oppgavens direktiv er en omformulering av ‘’feilsøk’’. Algoritmisk tankegang kommer fram når eleven undersøker programmets instruksjoner til leseren. Dekomponering der eleven feilsøker programmet i form av dens komponenter kan også skje ved arbeid med oppgaven. Undersøking av og erfaringer med programkodens egenskaper kan kobles opp mot kjennetegnet generalisering. Oppgavens anbud er en grunnleggende form for feilsøking, samt abstraksjon og algoritmisk tankegang for å kunne forklare feilen i programkoden.

4.5 Endre

28 av programmeringsoppgavene i den analyserte læreboka inneholder en form for **Endre** direktiv. Direktivet handler i disse oppgavene om å gjennomføre eller foreslå endringer. Om endringene gjennomføres eller ikke er uvesentlig for analysen av dette direktivet. Dette på grunn av at endringer som gjennomføres, altså aktiv programmering, og de kjennetegnene dette medfører er inkludert i **Lag** direktivets kategori. Endringene dette direktivet omhandler er endringer i programs egenskaper, og/eller endringer til variablene i program. Endringene kan være med eller uten videre føringer for hvordan disse endringene skal utføres.

Alle oppgavene med **Endre** som direktiv har en form for programmering som tema. Dette da det er programmerte program endringene skal gjelde.

Direktiver knyttet til denne kategorien setter også føringer og forpliktelser om at endringer skal gjelde et forhåndsbestemt program. Noe som hindrer eleven i å programmere eller foreslå et helt nytt program for å følge direktivet.

Eleven kan potensielt foreslå endringer i programegenskaper, kommandoer og variabler i arbeid med direktivet. I slike tilfeller må eleven bruke kunnskap om disse, knyttet til algoritmisk tankegang.

Dermed kan eleven også abstrahere direktivet om til programkode for å følge direktivet om å foreslå endringer for eksempel et programs egenskaper.

Dekomponering kjennetegnet brukes i arbeid med disse oppgavene fordi når et program skal endres må programmereren vite hva programmets bestanddeler gjør for å korrekt endre det. Programmereren videreutvikler også en spesifikk bestanddel av programmet, separat fra andre deler av programmet.

I arbeidet med dette direktivet kan eleven generalisere sammenhengene mellom den opprinnelige programkoden og endringene som gjøres til den. Dette da forskjellene mellom programkodens virkemåte fra det opprinnelige programmet til det nye programmet kommer tydelig fram i arbeid med oppgaver med dette direktivet.

I arbeid med disse direktivene i oppgavene i læreboka må eleven vite hva programmet og dens bestanddeler gjør for å kunne foreslå å gjennomføre endringer til det. Knyttet til kjennetegnene på AT algoritmisk tankegang og dekomponering.

Direktivet **Endre** finnes i ulike former i 28 av programmeringsoppgavene i læreboka. Direktivet handler om å foreslå og/eller gjennomføre endringer til programkode. I alle oppgavene i direktivet må elevene ta utgangspunkt i et forhåndsbestemt program og dermed gjøre eller forslå endringer til det. Potensielt kan elevene generalisere sammenhengene mellom det opprinnelige programmet og det nye programmet. Anvende algoritmisk tankegang for å foreslå spesifikke endringer til programkoden, utover egenskapene til bestanddelene i programmet. Noe som krever abstraksjon av direktivet til programkode. Det siste potensiale og anbudet til direktivet er dekomponering da eleven må vite hva programmets bestanddeler gjør for å korrekt endre og utvikle deler av programmet.

4.5.1 Analyse Endre oppgave 9.13c

I oppgavene 9.13 a og b skal eleven først lage en algoritme for en valutakalkulator som regner fra euro til norske kroner. Hvordan en valutakalkulator fungerer blir også innledningsvis forklart. Analyse objektet som er direktivet i oppgave 9.13c ser slik ut: ‘’Hvilke endringer må du gjøre om du skal lage en kalkulator som regner andre veien, fra NOK til EUR? Lag kalkulatoren’’. I denne analysen begynner jeg først med et løsningsforslag etter direktivet til den innledende programmeringsoppgaven 9.13b:

```
In [4]: a = float(input('Skriv inn antall EUR som skal regnes om til NOK'))
        b = a * 9.51
        print(b)

        Skriv inn antall EUR som skal regnes om til NOK10
        95.1
```

Figur 5- Egenskrevet programkode som utgangspunkt for oppgave 9.13c

Programmet over viser et program jeg har programmert selv som skal fungere som en kalkulator som gjør om EUR til NOK.

En måte å løse selve analyseoppgaven kan være:

For å endre programmet slik at det regner andre veien, fra NOK til EUR bør teksten knyttet til input-kommandoen i linje 1 endres til for eksempel: ‘’Skriv inn antall NOK som skal regnes om til EUR’’. I linje to kan variabel ‘’a’’ divideres med det gitte forholdet mellom NOK og EUR istedenfor å multipliseres, for å gjøre det motsatte av operasjonen i den originale kalkulatoren.

Programmet kan dermed se slik ut:

```
In [6]: a = float(input('Skriv inn antall NOK som skal regnes om til EUR'))
        b = a / 9.51
        print(b)

        Skriv inn antall NOK som skal regnes om til EUR100
        10.515247108307046
```

Figur 6- Egenskrevet programkode som viser endringer som kan gjøres i oppgave 9.13c

Med utgangspunkt i oppgavens direktiv og mitt eget forslag til oppgaveløsning inneholder oppgaven i hovedsak teoretisk programmering og matematikk med aritmetikk. Innenfor programmering er det input- og print-kommandoer og variabler som endringene knyttet til direktivet skal gjelde for. Når det kommer til aritmetikk er det multiplikasjon og divisjon av variabler knyttet til valutakalkulatorens formler som er grunnlaget for aritmetikk som tema.

Utfra oppgavens direktiv kommer det fram tre forpliktelser utover tema i denne oppgaven. Den første forpliktelsen er at eleven skal behandle et ferdiglaget program som eleven selv har programmert i forrige deloppgave. Dette setter føringer for hvilke endringer eleven kan gjøre utfra et program hen selv har programmert. Den andre forpliktelsen er at det er fast bestemt hvilke valutaer som skal behandles. Tilknyttet til dette er også valutakursen forhåndsbestemt. Der eleven kan velge mellom å bruke 9,51 som kurs mellom NOK til EUR eller finne den faktiske kursen på nett som utgangspunkt.

Eleven kan i arbeidet med oppgaven klare å se valutaene og kursen som variabler ved framprovosering av direktivet ‘’endre’’ og programmeringsspråkets egenskaper. Der komponentdelene som kan endres og vurderes er hovedsakelig variablene, men også kommandoene i programmet. Antallet NOK som brukeren skriver inn er variabel a som blir dividert med valutakurs, som jeg angir variabel v, er lik omregnet sum av euro, gitt av variabel e. Se løsningsforslaget.

I forslaget til endring skal hensiktsmessige kommandoer som input være med slik at brukeren kan skrive inn ønsket antall NOK som skal gjøres om til EUR og variablene nevnt over, altså en valutakalkulator. Noe som krever forståelse for programmeringsspråkets egenskaper knyttet til algoritmisk tankegang.

Dette vil også potensielt kunne generaliseres til sammenhenger utover denne oppgaven dersom eleven forstår at den abstraherte sammenhengen dermed blir:

$$\mathrm{e} = \frac{a}{v} \text{ og } \mathrm{a} = \mathrm{e} * \mathrm{v}$$

Noe som er nyttig utover denne oppgaven.

Det som er nødvendig for å kunne følge oppgavens direktiv er algoritmisk tankegang og dekomponering for å foreslå endringene utfra det eksisterende programmet. Eleven må helt grunnleggende vite eller komme fram til hva programmet og bestanddelene i den gjør. For å foreslå endringen som skal til for at programmet skal kunne dividere x antall NOK med valutakursen, altså forholdet mellom NOK og EUR, for å finne det tilsvarende beløpet i EUR.

Det analyserte direktivet i oppgaven handler om å forklare hvilke endringer som må til for å endre en valutakalkulator slik at den regner motsatt fra utgangspunktet som er fra EUR til NOK. Oppgaven omhandler en valutakalkulator og temaene som er knyttet til direktivet i oppgaven er hovedsakelig aritmetikk og teoretisk programmering. Føringene og forpliktelsene utover dette er at programmet eleven skal ta utgangspunkt i, har de programmert selv. I tillegg er det forhåndsbestemt hvilke valuta som skal brukes i oppgaven og til hvilke kurs. Potensialet innenfor denne rammen knyttet til den teoretiske endre delen av direktivet er abstraksjon, algoritmisk tankegang, dekomponering og generalisering. Dette utfra at eleven kan anvende eller utvikle forståelse for sammenhenger mellom multiplikasjon og divisjon:

$$\mathrm{e} = \frac{a}{v} \text{ og } \mathrm{a} = \mathrm{e} * \mathrm{v}$$

som en dekomponert inngang til videre forståelse og bruk av denne sammenhengen.

Anbudet i arbeid med dette direktivet er abstraksjon og algoritmisk tankegang. Eleven må helt grunnleggende vite eller komme fram til hva programmet og bestanddelene i den gjør for å kunne foreslå egnede endringer.

4.6 Beskriv

Beskriv er et direktiv som er til stede i 25 av programmeringsoppgavene i læreboka. Direktivet **kjør** er ofte i oppgavetekster med direktivet **beskriv**. Det som skal beskrives i disse oppgavene er variasjoner av: hva skrives til skjermen når programmet kjøres og hva er likt/ulikt eller forskjellig mellom kodene, eller det som skrives til skjermen, i to eller flere program. Direktivet **beskriv** handler om å gjengi skriftlig og noen ganger muntlig noe overfladisk og skiller seg dermed fra direktivet **forklar**, som skal fortelle om dypere sammenhenger.

I arbeid med oppgavene med dette direktivet er det hovedsakelig programmeringsspråket Python sine egenskaper som er hovedtema. I noen få oppgaver er også matematiske tema en del av konteksten.

Oppgaver med dette direktivet forplikter gjerne eleven å beskrive noe forhåndsbestemt som for eksempel noe som blir skrevet til skjermen.

Algoritmisk tankegang blir er et potensial da viktige egenskaper i programkode kan bli tydeliggjort i arbeidet med direktivet.

For å gjennomføre direktivet om å beskrive noe må en først identifisere det som skal beskrives. Eleven kan altså spore det som ønskes beskrevet gjennom kjennetegnet feilsøking.

Eleven må i arbeidet med direktivet spore hva som skal beskrives, knyttet til kjennetegnet på AT feilsøking.

4.6.1 Analyse Beskriv snakke matte oppgave a) s.112

Oppgaveteksten til snakke matte oppgave a) side 112 i læreboka er: ‘’Hva er likt og hva er ulikt i koden til de to programmene?’’ de to kodene det er snakk om, inkludert det som skrives til skjermen når programmet er ferdig kjørt, ser slik ut:

```
In [1]: for tall in range(3):
            print(tall)
            print('Nå er løkka slutt')

        0
        Nå er løkka slutt
        1
        Nå er løkka slutt
        2
        Nå er løkka slutt
```

Figur 7- Del 1 av ferdig skrevet programkode som utgangspunkt for oppgave a) s.112

Og:

```
In [2]: for tall in range(3):
            print(tall)
        print('Nå er løkka slutt')

        0
        1
        2
        Nå er løkka slutt
```

Figur 8- Del 2 av ferdig skrevet programkode som utgangspunkt for oppgave a) s.112

I det første programmet er det i linje 1 en for-løkke med 3 som verdi der både linje 2 og 3 er satt med inntrykk under løkka og begge kodelinjene vil derfor kjøre for hvert steg i løkka. Linje 2 er en print-kommando som skriver til skjermen hvilke ledd løkka er i. Linje 3 er også en print-kommando. Den skriver strengen ‘’Nå er løkka slutt’’ til skjermen. Dermed blir både linje 2 og 3 for hvert ledd i løkka skrevet til skjermen, i det første programmet.

Det andre programmet har samme innhold, men forskjellen fra det første programmet er at det kun er linje 2 som står med inntrykk til løkken i linje 1. Dermed er det kun linje 2 som blir skrevet til skjermen for hvert ledd av løkka og print-kommandoen med tilhørende streng i linje 3 blir skrevet til skjermen etter at løkka er ferdig kjørt.

Et mulig løsningsforslag til dette direktivet er å beskrive forskjellen mellom programmene. Hvilket kan være: Linje 3 står med innrykk i løkka i linje 1 i det første programmet og uten innrykk i det andre programmet.

Teoretisk programmering og innrykk i løkker er tema i direktivet. Dette utfra at eleven(e) i denne oppgaven skal beskrive hva som er annerledes mellom et program med og et program uten innrykk. Hvilket viser betydningen av inntrykk.

Den eneste forpliktelsen utover hoved-konteksten i oppgavens direktiv er at det er en forskjell mellom to spesifikke program eleven skal beskrive.

Viktigheten av tydelig definerte instruksjoner kommer fram gjennom at det er overfladisk forholdsvis lite forskjell mellom programmene i form av et innrykk, men forskjellen i programmets virkemåte er forholdsvis stor.

Sporingsdelen av feilsøking er noe som også kan bli trent i arbeidet med denne oppgaves direktiv. Eleven(e) kan da spore forskjellen som skal beskrives.

Eleven(e) må først finne forskjellen som utgjør differansen i programmene gjennom sporing, knyttet til kjennetegnet på AT feilsøking, for å deretter kunne beskrive den.

Dette er en oppgave der betydningen av innrykk i løkker i programmeringsspråket Python kommer fram. Oppgaven viser til to nok så like programmer der den eneste forskjellen mellom de er at linje 3 er satt med inntrykk under løkken i linje 1 av programmene. Elevene kan potensielt bruke to kjennetegnene på AT i arbeid med direktivet. Algoritmisk tankegang, der betydningen av tydelig definerte instruksjoner er sentral i arbeidet med direktivet. Det siste potensialet og anbudet i direktivet er feilsøking der eleven(e) kan spore forskjellene i programmene.

4.7 Bi-direktiver

I denne delen av kapitlet skal bi-direktivene kort presenteres.

4.7.1 Kjør

Kjør direktivet finnes i 25 av oppgavene i læreboka. Som nevnt er direktivet **kjør** ofte sammen med direktivet **beskriv** i oppgavetekster. Dette direktivet handler om å trykke på ‘’Kjør knappen’’ i Python-leseren. Dette direktivet har tilsynelatende ingen potensialer eller anbud. Dette da direktivet krever minimalt for å gjennomføres.

4.7.2 Test

Direktivet test forekommer 9 ganger i oppgavetekster i læreboka. 6 ganger sammen med direktivet **endre,** 3 ganger sammen med direktivet **beskriv,** 2 ganger er direktivet sammen med direktivet **skriv inn** og 1 gang med direktivet **lag**. Direktivet kan sies å være synonymt med ‘’prøv ut’’. I oppgavetekstene der direktivet er sammen med direktivet **endre** skal eleven teste flere endringer. Direktivet **test** sammen med direktivet **endre** kan gjøre at innholdet i arbeidet med direktivet er generaliserbart ved at de opplever endringer og konstanter i programkode og det som skrives til skjermen. Anbudsmessig er det ikke nødvendig at arbeidet med direktivet er generaliserende.

4.7.3 Skriv inn

Skriv inn direktivet finnes i 5 av oppgavetekstene i læreboka og alle disse oppgavetekstene inneholder også et annet direktiv. Direktivet er hovedsakelig brukt i sammenhenger der eleven skal skrive et program eller deler av et program inn i en Python-leser. Direktivet **skriv inn** har tema programmering og løkker i alle oppgavetekstene. Direktivet er også begrenset til en spesifikk løkke for hver oppgave. Potensielt kan da eleven bruke algoritmisk tankegang ved at viktigheten av tydelig definerte instruksjoner kommer fram i programmet for at det skal

fungere. Når det gjelder direktivets anbud vil arbeid med dette direktivet i seg selv ikke kreve at elevene bruker noen av kjennetegnene på AT. Eleven trenger altså nødvendigvis ikke å se viktigheten av tydelig definerte instruksjoner.

4.7.4 Gjett

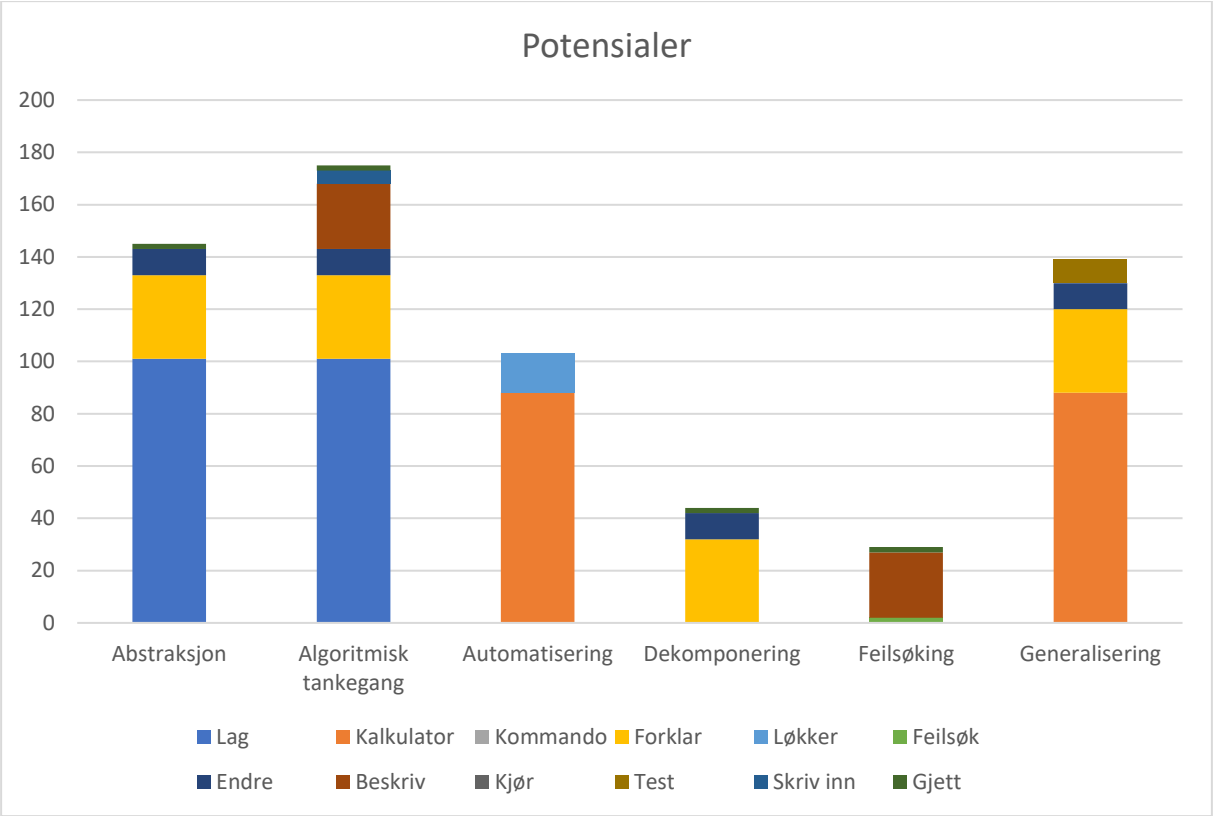
Gjett er et direktiv som finnes i 2 av programmeringsoppgavene i læreboka. I begge tilfellene er direktivet **gjett** sammen med direktivene **endre** og **kjør** i oppgaveteksten. I disse oppgavetekstene handler direktivet om å gjette hva som kommer til å skje når en spesifikk endring gjøres til programkoden og deretter skal eleven kjøre programmet.

Potensialene for bruk av kjennetegn på AT i arbeid med **gjett** direktivet er: algoritmisk tankegang, abstraksjon, dekomponering og feilsøking. Disse kan brukes av eleven for å forsøke å bedre forstå hva som kommer til å skrives til skjermen, knyttet til deres gjetning.

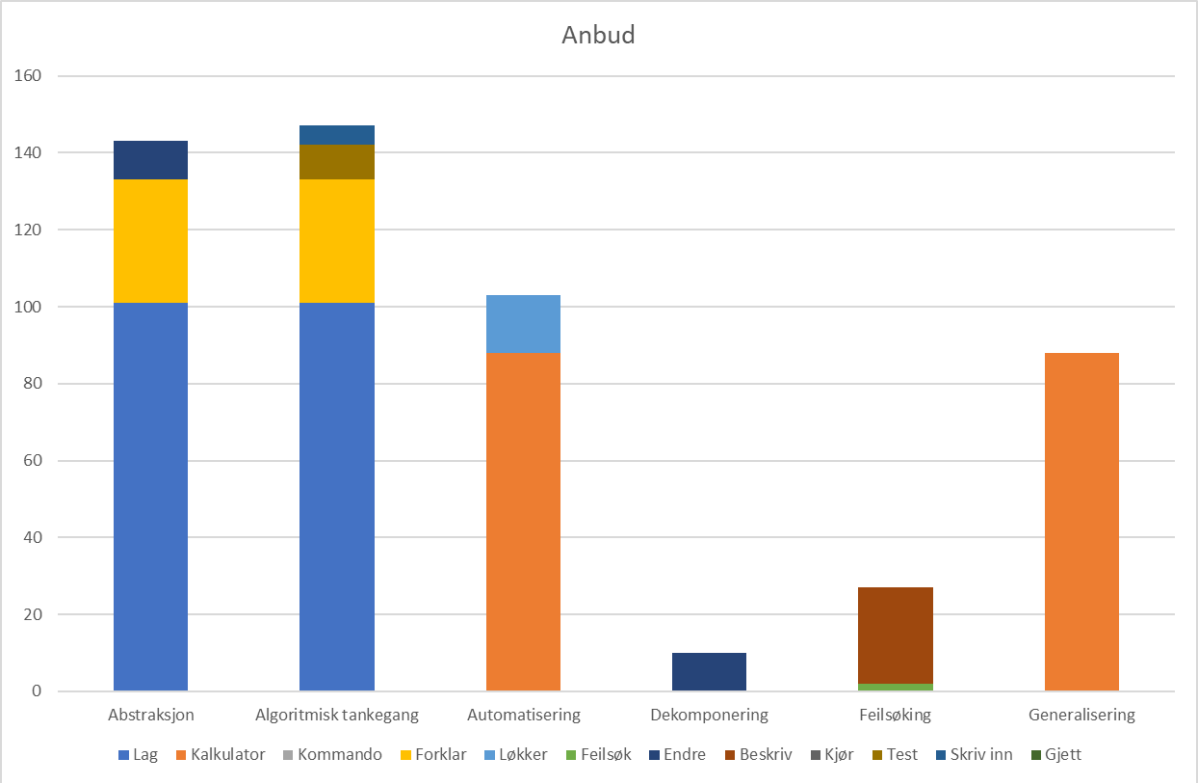
Når det kommer til anbudet til direktivet, finner jeg ingen kjennetegn på AT. Dette er fordi eleven nødvendigvis ikke trenger å undersøke noe som helst for å gjøre en gjetning.

4.8 Kvantitative resultater av lærebokanalysen

I denne delen av kapitlet presenteres de kvantitative resultatene av lærebokanalysen. Det første stolpediagrammet nedenfor viser fordelingen av kjennetegn på AT som kan potensielt finnes i arbeid med de ulike kategoriene av direktiv i programmeringsoppgavene i læreboka (se figur 9). Det andre stolpediagrammet viser fordelingen av kjennetegn på AT som er anbudet i arbeidet med de ulike kategoriene av direktiv i programmeringsoppgavene i læreboka (se figur 10).



Figur 9- Andelsfordeling kjennetegn på AT i direktivenes potensial



Figur 10- Andelsfordeling kjennetegn på AT i direktivenes anbud

5.0 Drøfting

For å kunne belyse problemstillingen skal jeg i dette kapitlet drøfte funnene fra analysen opp mot relevant teori og tidligere forskning. Jeg begynner kapitlet med å repetere problemstillingen og grunnlaget for den. Videre presenteres funnene fra analysen. Deretter presenterer jeg forskningen i denne masteroppgaven opp mot tidligere forskning og ser på ulike definisjoner på AT som grunnlag for analyse. Videre undersøkes analysemetoder i tidligere forskning opp mot analysemetoden i denne masteroppgaven. Valgene som er gjort i denne masteroppgaven rundt avgrensning av analysen skal deretter drøftes. Her er det AT og oppgavers direktiv med instrumentteori som analysegrunnlag som tas opp.

Problemstillingen i denne masteroppgaven er:
Hvilke muligheter for algoritmisk tenkning kan programmeringsoppgaver i matematikklærebøker gi elever på 8. trinn?

Jeg har valgt å belyse problemstillingen ved å undersøke mulighetene elevene får for bruk av kjennetegn på AT gjennom et operasjonelt instrumentaliserings rammeverk, Bocconi et al. (2016) sine kjennetegn på AT og direktivene i programmeringsoppgavene som analyseobjekt. Potensialer og anbud vil også fungere som en avgrensning som tar for seg hva elever kan og må bruke av kjennetegn på AT i arbeidet med direktivet. Dette gjøres ved at i potensialer- og anbudsdelene av analysen undersøkes hva eleven respektivt kan og må bruke av kjennetegn på AT i arbeid med direktivet. På denne måten vil alle elevene som klarer å gjennomføre direktivet bruke kjennetegn på AT til en grad innenfor potensialet og anbudet i direktivet.

5.1 Resultater

I analysen fant jeg, både når det gjelder potensialer og anbud, ut at algoritmisk tankegang er til stede i arbeidet i størst grad knyttet til de fleste direktivene. I matematikklæreboka Matemagisk 8 legges det opp til å bruke Python og dens programkode som bakgrunn for arbeid med programmeringsoppgaver. Der eleven gjerne skal lage, forklare eller endre noe i tilknytning til en bestemt Pythonkode. Python er et høynivå og tekstbasert programmeringsspråk der en har tilgang til mange ulike kommando- og variabeltyper (Dvergsdal, 2019; Rossen, 2019). Python, i likhet med de fleste andre tekstbaserte

programmeringsspråk, er likevel avhengig av tydelige og korrekt definerte variabler og kommandoer plassert i rett rekkefølge for at programmet å fungere på ønsket måte. Noe som gjør at algoritmisk tankegang kan og må brukes i arbeidet med de fleste direktiv i programmeringsoppgaver. Grunnen til at algoritmisk tankegang finnes så mye av i oppgavene kan også skyldes at det er mange oppgaver som har et trinn-for-trinn instruksjons direktiv.

Kjennetegnet på AT feilsøking finnes i et fåtall av direktiv med dette kjennetegnet på AT, både potensielt og anbudsmessig. Bocconi et al. (2016) sin definisjon på kjennetegnet feilsøk: ‘’Feilsøking (Debugging) er den systematiske anvendelsen av analyse og evaluering ved bruk av ferdigheter som testing, sporing og logisk tenking til å forutse og verifisere utfall’’ (Csizmadia et al., 2015, s. 9). Hvilket er en bred definisjon og deler av definisjonen må brukes i mange reelle programmeringssituasjoner. For eksempel dersom en programmerer et program som ikke fungerer og ønsker å rette det opp. Feilsøking finnes i Matemagisk 8 i all hovedsak i direktivet **beskriv**. Kompetansemålet knyttet til programmering i matematikk for 8. trinn: ‘’utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering’’ (Kunnskapsdepartementet, 2019). Der begrepet ‘’testes’’ er sentral i kjennetegnet på AT feilsøking. Dette er dermed en oppdagelse det er verdt å nevne da Matemagisk 8 er ment for 8.trinn og det er nærliggende å tro at feilsøking vil være en stor del av programmeringsinnholdet i læreboka. Dette på grunn av feilsøkings tilknytning til vellykket programmering og det nevnte kompetansemålet.

Når det kommer til kjennetegnet på AT dekomponering finnes også få direktiv med dette kjennetegnet på AT. I direktivets anbud er det kun i *endre*-kategorien dekomponering må brukes for å gjennomføre direktivet i oppgavene. Årsaken til at det er så få av disse tankemåtene forpliktet i oppgavenes direktiv kan skyldes flere ting. En mulig forklaring kan være at lærebokforfatterne ser på feilsøking og dekomponering som noe som er indirekte eller innebygd i innholdet i programmeringsoppgavene. Dette da elevene antageligvis vil gjøre noen feil i arbeidet med oppgavene og blir nødt til å feilsøke og dekomponere programkode for å få den til å fungere på ønsket måte. Hvilket er knyttet til aktiv programmering i *Lag*-kategorien, som er forholdsvis mange av i læreboka.

Kjennetegnet på AT generalisering inneholder potensielt flere typer direktiv enn i dens anbud. I anbudet er det kun den forholdvis store under-kategorien **Kalkulator** som er det eneste direktiv hvor arbeidet må være generaliserende. Denne mangelen på generaliserende innhold funnet i analysen kan skyldes at forfatterne har lagt det generaliserende innholdet i oppgavene

utenfor direktivet. Slik at det ikke blir regnet med i analysen, som er avgrenset til direktivet i oppgavene.

5.2 Tidligere forskning sett opp mot masteroppgaven

I denne delen av kapitlet skal jeg drøfte tidligere forskning opp mot denne masteroppgaven. Jeg skal først drøfte AT som bakgrunn for analyse og metode da det er mange ulike definisjoner på AT. Hvilket påvirker grunntankene i studien. Deretter skal jeg drøfte analysen i denne masteroppgaven opp mot analysemetodene i tidligere forskning.

5.2.1 AT som bakgrunn for analyse og metode

Det er som sagt mange ulike definisjoner på AT. Dermed er vurdering av definisjoner på AT viktig da det er med å avgjøre hvilke grunntanker studien bygger på. Noen ser på AT som en problemløsningsmetode, mens andre ser på AT som en form for tenkning knyttet til datamaskiners virkemåte og programmering (Mæland & Myklebust, 2022; Wing, 2008).

Elicer og Tamborg (2022) og Tamborg et al. (2022) knytter AT opp mot programmering og bruker programmering og algoritmisk tenkning som hovedbegrep i studiene (PCT). I Elicer og Tamborg (2022) bruker de Shute et al. (2017, s. 142) sin definisjon på AT: ‘’Det konseptuelle fundamentet nødvendig for å løse problemer effektivt med løsninger som er mulig å gjenbruke i forskjellige kontekster’’. De viser også til diskusjonen om AT bør karakteriseres med en lukket definisjon eller av dens kjennetegn. Utfra at de kobler programmering som en handling til deres operasjonelle analyse kan en si at de ser på AT som en problemløsningsmetode.

Måten Bråting og Kilhamn (2022) definerer AT på er både som et arbeid der noe utføres og som en tankeprosess som kan representeres. De skriver at AT kan forklares som en prosess der vi beskriver, analyserer og løser problemer på en måte slik at datamaskiner kan assistere arbeidet. Samtidig som at de bruker Aho (2012, s. 832) sin definisjon: ‘’tankeprosessen involvert i å formulere problemer slik at løsningen deres kan representeres som ‘’computational’’ steg og som algoritmer’’.

Når AT blir sett på som en problemløsningsmetode og en kan identifisere innhold med et krysskjema, som er en mekanisk eller automatisert vurderingsmetode der kategoriene brytes slik at de er selvevidente og oppgavene er kognitivt enkle å plassere. Hvor en videre vurderer oppgaver utfra hvor mange kryss det har fått i ulike kolonner, vil en ha mulighet til å kategorisere mange oppgaver på tvers av lærebøker, slik som Bråting og Kilhamn (2022) og Stokkenes (2022) har gjort. Dermed får en et bredt datamateriale som sier noe om et helt felt av programmeringsoppgaver. Leseren vil dermed få en god oversikt over programmeringsoppgaver i lærebøker på generell basis. Analysen kan også inneholde en bredere avgrensning av oppgaver, slik som Elicer og Tamborg (2022) og Tamborg et al. (2022) gjør i sine studier der de analyserer hele undervisningsopplegg. Fordelen med denne metoden er at de undersøker en større del av konteksten en oppgave befinner seg i.

I denne masteroppgaven ser jeg på AT som en tankemåte. Denne tankemåten kan brukes i arbeid med oppgaver både i programmering og matematikk og er overførbar mellom dem (Wing, 2008). Bocconi et al. (2016) definerer AT som: tankeprosesser involvert i å formulere et problem for å kunne anvende en ‘’computational’’ løsning som involverer abstraksjon, algoritmisk tankegang, automatisering, dekomponering, feilsøking og generalisering.

5.2.2 Analyse i denne masteroppgaven og tidligere forskning

I Bråting og Kilhamn (2022) bruker de en summativ innholdsanalyse der søkelyset er på hvor ofte AT og matematiske begreper forekommer i programmeringsoppgaver. De har undersøkt dette i svenske matematikklærebøker fra 1. til 6. klasse. På denne måten fikk de til å kategorisere programmeringsoppgaver i flere matematikklærebøker. Når de kategoriserer oppgavene utfra ord og begreper og egenvurderinger i stor skala kan det tenkes at den operasjonelle definisjonen på AT i deres arbeid er knyttet sterkest til AT som en problemløsningsmetode. Der begrepene identifisert i kategoriseringen ses i sammenheng med hvordan disse typene av programmeringsoppgaver kan løses.

Elicer og Tamborg (2022) og Tamborg et al. (2022) har en annen tilnærming til analyse. I disse studiene undersøker de et fåtall oppgaver på et bredt vis. De undersøker noen få oppgaver utfra flere analysesteg med ulikt søkelys og får med dette god oversikt over potensialet i oppgavene. I tillegg undersøker de undervisningsopplegg der det de kan

identifisere som betydningsfullt av AT- og matematisk innhold innenfor oppleggets rammer blir analysert.

Når AT blir behandlet som en tankemåte er det å identifisere det i oppgaver annerledes, enn dersom AT behandles som en problemløsningsmetode. Dette er fordi hva problemet er i oppgaven, i form av hva den inneholder av tema og begreper kan identifiseres utfra et krysskjema slik som Bråting og Kilhamn (2022) gjør i deres kategorisering av oppgaver. Dette med forbehold om at andre mulig vil kategorisere oppgavene på en annen måte. (Tamborg et al., 2022; Elicer & Tamborg, 2022) sine studier er også basert på innholdet og problemløsningsmetoder i undervisningsopplegg. Når AT behandles som en måte å tenke på er det derimot tankemåter oppgaven inneholder som skal analyseres. Hvordan tankemåter arbeidet med en oppgave inneholder krever en dypere og mer avgrenset analyse.

I en analyse der AT blir sett på som en tankemåte er det dybde og nyanser som er styrken. Hva oppgavens direktiv til eleven er analyseres på en mer nyansert måte når målet er å finne ut hva eleven kan bruke av AT som tankemåte. I analysen av denne masteroppgaven, som bruker AT som en tankemåte, undersøkes direktivet ut fra konteksten den er plassert i, tema i den og begrensninger og føringer i direktivet. Dermed kan kategorier med samme type direktiver, utfra hvordan kjennetegn på AT eleven kan og må bruke i arbeidet med direktivet, dannes. Leseren vil på denne måten få et dypt innblikk i tankemåter knyttet til direktiv. Sammenhengene mellom AT og matematikk kommer tydeligere fram i dypere analyse med AT som en tankemåte. Der Bocconi et al. (2016) sine kjennetegn på AT: abstraksjon, algoritmisk tankegang, automatisering, dekomponering, feilsøking og generalisering er kjente tankemåter også innenfor matematikk.

5.3 AT, direktiv og instrumentteori som avgrensninger for analyse

Hittil i dette kapitlet har jeg drøftet denne masteroppgaven opp mot tidligere forskning. Analysemetoden knyttet til AT som en tankemåte krever tydelige avgrensninger på hva som undersøkes. Derfor har jeg i arbeidet med masteroppgaven valgt å avgrense arbeidet til AT gjennom instrumentteori og oppgavedirektiv som bakgrunn for analyse. Disse tre avgrensningene skal drøftes nærmere i denne delen av kapitlet.

5.3.1 AT som utgangspunkt for analyse

Da dybde og nyanser er styrken i denne masteroppgaven har jeg valgt å avgrense analysen til og undersøke AT. Masteroppgaven bygger også på kompetansemålet for programmering i matematikk for 8.trinn: ‘’utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering’’ (Kunnskapsdepartementet, 2019). Kompetansemålet har i all hovedsak et AT-innhold og mangler matematisk innhold, dersom en ser bort fra sammenhengen mellom temaene. Dette utfra at kompetansemålet har likheter med kjennetegnene en finner i AT. Der kjennetegnet algoritmisk tankegang handler om å skape algoritmer og kjennetegnet feilsøking handler blant annet om å teste og forbedre algoritmer (Bocconi et al., 2016).

5.3.2 Instrumentteori som rammeverk

For å kunne analysere AT-innhold og finne kjernen i oppgavedirektivene har jeg brukt en operasjonell definisjon på instrumentteori og instrumentalisering som rammeverk for analyse. Instrumentalisering som rammeverk er laget for å isolere artefaktet og dens egenskaper til analyse, hvilket gjør at en får en grundig analyse av det en ønsker å analysere (Trouche, 2020). Dette gjør denne masteroppgaven egnet for lesere som ønsker å lese om AT i programmeringsoppgaver og knytte det opp mot for eksempel undervisning. Dette da analysen i denne masteroppgaven er grundig og inneholder forklaringer og sammenhenger leseren kan ta stilling til. De kan selv vurdere forklaringene og sammenhengene identifisert opp mot funnene i analysen. Disse forklaringene kan igjen brukes som et rammeverk for leseren til å vurdere og analysere andre oppgaver. Det er i norsk skole opp til læreren å vurdere hvilke læremidler de skal bruke i undervisning (Skjelbred et al., 2017). Det er derfor nyttig å analysere noen oppgaver grundig slik at en raskere kan vurdere og analysere oppgaver senere. En lærer vil ikke kunne ta seg tiden til å analysere alle oppgaver like grundig som i analysen i denne masteroppgaven, men de vil kunne få en bedre forståelse for hvordan de kan vurdere oppgaver basert på å ha lest analysen i denne masteroppgaven.

Grunnen til at jeg har brukt en operasjonell definisjon på instrumentalisering som rammeverk er at oppgaver i en lærebok ikke er et typisk artefakt å analysere med rammeverket. Hvilket jeg kan si etter å ha søkt etter instrumentteori og instrumentalisering i tilknytning til

lærebokanalyse i relevante databaser uten funn. Da rammeverket ikke er tilpasset analyse av oppgaver i lærebøker trengte den noen tilpasninger slik at den kan brukes som et rammeverk for analyse av programmeringsoppgaver. Denne videreutviklingen av rammeverket er dermed et bidrag til instrumentteorien slik at den kan passe til analyse av flere typer artefakter.

5.3.3 Analyse av direktiv

Analysen av programmeringsoppgavene tar utgangspunkt i direktivene i oppgaveteksten i oppgaven som en inngang til dybde i analysen. Studien til Bråting og Kilhamn (2022) legger opp til direktiv som det som har størst betydning for hva innholdet i oppgaven er. Avgrensning av analyse av direktiv er også et bevisst valg i denne masteroppgaven ettersom at eleven kan potensielt bruke alle kjennetegnene på AT i de fleste oppgaver dersom de arbeider utover direktivet. Feilsøking og dekomponering er eksempler på dette. Feilsøking (Debugging) er den systematiske anvendelsen av analyse og evaluering ved bruk av ferdigheter som testing, sporing og logisk tenking til å forutse og verifisere utfall (Csizmadia et al., 2015, s. 9). Dersom en tar utgangspunkt i analysen av *kalkulator* oppgave 7.24b under potensialer kan feilsøking være et potensial dersom analysen strekker seg utover direktivet i oppgaven:

I likhet med alle program kan eleven analysere programmet, spore feil, forutse og verifisere utfall fra programmet på en logisk og systematisk måte i tråd med kjennetegnet på AT, feilsøking. Dette stemmer for programmet i den aktuelle oppgaven også. Eleven kan analysere programmet de har laget ved å sikre at alle komponentdelene som skal være i programmet er der og har den ønskede funksjonen, både forutse resultater teoretisk og verifisere utfall i praksis. I denne oppgaven kan det gå ut på å sjekke: variabler for sirkelomkrets, multipliser og utregning. At programmet lar brukeren skrive inn verdier for variabelen knyttet til diametere til sirkler ved hjelp av en ‘’input-float’’ kommando. Oppgavens direktiv tilsier også at programmet skal inkludere kommandoer som kan skrive til skjermen sirkelomkretser og sirkel nummer med utregning, med strenger som forklarer disse. Disse variablene og kommandoene må også være satt sammen/fungere sammen på en hensiktsmessig måte slik at programmet fungerer og resultatet er oversiktlig, noe eleven kan undersøke ved feilsøking. Jeg går ikke nærmere inn på hvordan de bør være satt sammen da dette kan gjøres på ulike måter og er opp til eleven som arbeider med oppgaven å avgjøre.

Kjennetegnet på AT dekomponering er en måte å tenke på artefakter når det gjelder deres komponentdeler. Delene kan deretter bli forstått, løst, utviklet og evaluert separat. Dette gjør komplekse problemer lettere å løse, nye problemer bedre forstått og store systemer lettere å designe steg for steg (Csizmadia et al., 2015, s. 8). I tilknytning til samme analyseoppgave som over kan også dekomponering være et potensial dersom en undersøker oppgaven utover dens direktiv:

For å kunne lage dette programmet bør eleven ha en viss forståelse for de ulike komponentene i programmet. De bør forstå hva variablene er og står for. De bør også forstå hva kommandoene gjør med variablene. Eksempelvis bør de vite at ''for-løkken'' i programmet gjentar den innrykkede koden under. Da må de samtidig bør de ha en forståelse for hva som skjer med variablene som blir påvirket av løkken. De bør forstå at variabelen ''i'' øker med 1 for hver gang programmet kjører som indikasjon på sirkelnummeret. Eleven bør også vite at variabelen ''d'' er avhengig av hva brukeren skriver inn. I tillegg til at de bør forstå at denne prosessen gjentar seg opptil 5 ganger.

Det å undersøke utover oppgavens direktiv har nytteverdier da læreren kan legge til rette for det gjennom egne føringer til elevene. Likevel er det vanskelig å vurdere hva en elev kan teoretisk legge til i arbeidet med en oppgave. Det viser seg også at mange lærere vil følge det lærebøkene og det oppgavene i den legger til rette for.

5.4 Programmeringsoppgaver og deres direktiv som analysegrunnlag

Bruk av direktivene i programmeringsoppgaver i læreboka som utgangspunkt for analyse kan også drøftes. Wing (2006) fremmer som nevnt i teoridelen ''Computational thinking'' (AT) som en universell tenkemåte og ferdighet for alle. Det ble starten på en bølge med etterlysning av utdanning i AT med programmering som det mest sentrale verktøyet for å få det til. Mange land i verden har nå innført programmering og AT i skolen.

I fagfornyelsen 2020 er programmering blitt en del av mange fag i grunnskolen, deriblant matematikk der det er innført i størst grad knyttet til begrepet AT (Kunnskapsdepartementet, 2019). Dette gjorde at lærere i matematikkfaget skulle begynne å undervise programmering i tilknytning til faget.

Med utgangspunkt i tidligere forskning har jeg som sagt til og med valgt å intervjue 3 matematikklærere på ungdomsskolen med programmering og AT som tema i masteroppgaven. Kort oppsummert viser det seg at funnene i tidligere forskning gjennomført i andre lands kontekster stemmer godt overens med resultatene av intervjuene (Bråting & Kilhamn, 2022). I intervjuene kommer det fram at lærerne altså er utrygge på å undervise i det nye programmeringsinnholdet i matematikkfaget. De mangler formell utdanning i programmering og kan lite om det. De har også lite kunnskap om AT. De støtter seg på læreboka og oppgavene i den dersom de skal undervise programmering i matematikk. Skjelbred et al. (2017, s. 23) mener også at lærebøker er sentrale i undervisningen i norsk skole. Ifølge (Thonhauser, 2008) er oppgaver katalysatorer for læringsprosesser og det er i lærebøker dette potensialet kan utfolde seg på en meningsfull måte.

6.0 Konklusjon

I denne masteroppgaven har jeg brukt tidligere forskning, relevant teori og innsamlet data for å belyse problemstillingen:

Hvilke muligheter for algoritmisk tenkning kan programmeringsoppgaver i matematikklærebøker gi elever på 8. trinn?

For å belyse problemstillingen har jeg besvart to forskningsspørsmål som er kommentert og oppsummert nedenfor. Til slutt i dette kapitlet skal jeg også gi noen anbefalinger til lærebokforfattere og videre forskning basert på arbeidet i denne masteroppgaven.

Forskningsspørsmål 1:

Hvilke læringsmateriell benytter lærere på ungdomsskolen for å finne programmeringsoppgaver for deres undervisning?

Målet med denne masteroppgaven er å belyse mulighetene for AT i programmeringsoppgavene i en matematikklærebok for 8. trinn. Tidligere forskning fra andre lands skolekontekster viser at matematikklærerne i disse landene er usikre på det nye programmeringsinnholdet knyttet til matematikk i skolen. Dette gjør at de støtter seg på lærebøkene som en form for fasit til undervisning i tema (Bråting & Kilhamn, 2022; Elicer & Tamborg, 2022). Det finnes etter min beste evne ingen tidligere forskning på om dette også gjelder i den norske skolekonteksten. Derfor gjennomførte jeg de semistrukturerte intervjuene

i denne masteroppgaven tilknyttet forskningsspørsmål 1. Den tidligere forskningen om dette viste seg å stemme godt overens med resultatene av intervjuene. Dermed vurderer jeg analyse av læreboka de intervjuede lærerne bruker på deres skole som et gyldig analyseobjekt.

Forskningsspørsmål 2:

Hvilke grener finnes det innenfor algoritmisk tenkning i programmeringsoppgaver i matematikklærebøker på 8.trinn?

I arbeidet med denne masteroppgaven viste det seg at for å kunne belyse dette forskningsspørsmålet var det behov for noen tydelige avgrensninger. Direktivene i oppgavene ble tatt utgangspunkt i for kategorisering og analysering. Direktivene ble kategorisert utfra likheter og ulikheter mellom dem.

For å se på mulighetene til AT i disse direktivene i læreboka ble en operasjonalisering av instrumentaliseringsgrenene presentert i Trouche (2020) brukt. Direktivkategorienes aktivisering i form av tema og begrensinger i form av føringer og forpliktelser ble identifisert, som bakgrunn for hvilke potensialer og anbud for Bocconi et al. (2016) sine 6 kjennetegn på AT direktivene inneholder. Dette gjorde at mulighetene for AT, altså det som kan og må brukes av AT i gjennomføringen av direktiv ble identifisert. I analysen ble også typiske oppgaver fra hver kategori og underkategori analysert på samme vis. Dette for å gi leseren innsikt i hva som ligger til grunn i tilknyttingene og sammenhengene som ble identifisert i analysen av tilhørende kategori. Slik kan leseren vurdere analysen opp mot funnene i masteroppgaven på egenhånd, samt gir de mulighet til å selv analysere programmeringsoppgaver ut fra samme logikk.

Resultatene av analysen viste at når det gjelder potensialer og anbud er algoritmisk tankegang til stede i arbeidet i størst grad knyttet til de fleste direktivene i programmeringsoppgavene i læreboka. Årsaken til dette kan være programmeringsspråkets egenskaper og at mange programmeringsoppgaver har et trinn-for-trinn instruksjons direktiv. Kjennetegnet på AT feilsøking fantes det lite av i læreboka. Dette var merkverdig da kompetansemålet for programmering i matematikkfaget for 8. trinn er knyttet til feilsøking. Kjennetegnet på AT dekomponering finnes det også få direktiv med. Årsaken til denne mangelen på direktiv med kjennetegnene på AT dekomponering og feilsøking kan være at de er ment å være indirekte eller innebygd i innholdet i programmeringsoppgavene. Anbudsmessig er det også få direktiv med kjennetegnet på AT generalisering. Forklaringen på dette kan være at forfatterne har lagt det generaliserende innholdet i oppgavene utenfor direktivet i oppgaven.

6.1 Anbefalinger til lærebokforfattere

Denne masteroppgaven er vinklet mot AT og anbefalingene som gis her vil naturligvis være med AT som utgangspunkt. Begrepet AT er som sagt sentralt innenfor programmering, programmerings- og matematikkdidaktikk. Kompetansemålet i matematikkfaget knyttet til programmering for 8.trinn har også hovedsakelig et AT-innhold. Det bør nevnes at jeg anerkjenner selvfølgelig at lærebokforfattere også har andre hensyn å ta enn AT og kan ikke basere alle programmeringsoppgaver på AT. Jeg tror heller ikke dette ville ha gitt gode, balanserte og gjennomførte oppgaver. Disse anbefalingene er heller ikke ment å gi retningslinjer for alle programmeringsoppgaver da jeg mener det er ønskelig med varierte oppgaver med en hensiktsmessig variasjon av direktiv.

Jeg synes det er positivt at det finnes så mange **lag** direktiv i oppgavene i læreboka. **Lag**-direktivet gjør som sagt at elevene må og kan bruke mange kjennetegn på AT. Likevel kan det være flere oppgaver der elevene får **abstrahere** innholdet i oppgavens direktiv (oppgavens bestilling) til programkode uten en trinn-for-trinn instruksjon som gis i mange oppgaver.

Når det kommer til kjennetegnet på AT feilsøking fant jeg lite av dette kjennetegnet på AT knyttet direkte til arbeidet med programmeringsoppgavene. Gjerne se drøftingen knyttet til kjennetegnet tidligere i drøftingskapitlet. Der konkluderer jeg med at feilsøking er en viktig del av programmering og er en sentral del av kompetansemålet om programmering i matematikk for 8. trinn. Derfor mener jeg at det kan være hensiktsmessig å bruke dette direktivet i større grad i programmeringsoppgaver.

I tilknytning til avsnittet over, men også mer generelt, vil jeg anbefale i større grad enn nå å inkludere noen innledende sider generelt om programmering i læreboka. Disse sidene bør forklare hva programmering brukes til, hvorfor det er nyttig å kunne grunnleggende programmering, hvordan Python og programmeringsspråk fungerer og hvordan en kan feilsøke programkode i Python. Slik vil både elevene og lærere vite hva som ligger til grunn for arbeid med programmering, hvordan programmering kan arbeides med og hva en kan gjøre dersom en får opp feilkoder eller programmet ikke fungerer slik som ønsket.

6.2 Videre forskning

I denne masteroppgaven har jeg sett på muligheter for AT i direktivene i programmeringsoppgaver. Det hadde vært spennende og gjennomført videre forskning på tema knyttet til hvilken grad de ulike kjennetegnene på AT er til stede i direktivene. Direktivet **beskriv** er et eksempel på et direktiv der eleven må bruke kjennetegnet på AT feilsøk. Likevel er graden av tilknytning til kjennetegnet i arbeid med direktivet minimalt da eleven kun bruker feilsøking i sporingen av det de skal beskrive. En studie med instrumentteori og instrumentalisering som rammeverk der søkelyset er dypt og handler om kvaliteten til programmeringsoppgavers AT-innhold kan være en måte å forske videre på dette. Dette vil da kreve en enda tydeligere avgrensning enn i denne masteroppgaven.

Referanseliste

Aho, A. V. (2012). Computation and computational thinking. *The computer journal*, *55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

American Psychological Association. (2020). Concise guide to APA Style (7th ed.) https://doi.org/10.1037/0000173-000

Andersen, K. N. (2020). Assessing task-orientation potential in primary science textbooks: Toward a new approach. *Journal of Research in Science Teaching*, *57*(4), 481-509. https://doi.org/10.1002/tea.21599

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J. & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, *19*(3), 47-57. https://pure.uva.nl/ws/files/8964271/A_K_6_Computational_Thinking_Curriculum_Framework.pdf

Artigue, M. (2002). Learning mathematics in a CAS environment: The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International journal of computers for mathematical learning*, *7*, 245-274. https://doi.org/10.1023/A:1022103903080

Aschehoug. (u.å). *Våre forfattere* Hentet 13.12.2022 fra https://aschehoug.no/forfattere/vaare-forfattere

Bakken, J. & Andersson-Bakken, E. (2021). The textbook task as a genre. *Journal of Curriculum Studies*, *53*(6), 729-748. https://doi.org/10.1080/00220272.2021.1929499

Barr, D., Harrison, J. & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23. https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads*, *2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2016). Building mathematical knowledge with programming: insights from the ScratchMaths project. https://www.researchgate.net/publication/295912532_Building_mathematical_knowledge_with_programming_insights_from_the_ScratchMaths_project

Berland, M. & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, *24*(5), 628-647. https://doi.org/10.1007/s10956-015-9552-x

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P. & Punie, Y. (2016). Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report*, *68*. https://doi.org/10.2791/792158

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Bråting, K. & Kilhamn, C. (2022). The integration of programming in Swedish school mathematics: investigating elementary mathematics textbooks. *Scandinavian Journal of Educational Research*, *66*(4), 594-609. https://doi.org/10.1080/00313831.2021.1897879

Cohen, L., Manion, L. & Morrison, K. (2007). Research methods in education *Oxon: Routledge*, *6*. https://gtu.ge/Agro-Lib/RESEARCH%20METHOD%20COHEN%20ok.pdf

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. & Woollard, J. (2015). Computational thinking-A guide for teachers. https://www.computingatschool.org.uk/media/kscbloob/computationalthinking.pdf

Cuny, J., Snyder, L. & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress.* Hentet 30.11.22 fra

Dahl, T., Askling, B., Heggen, K., Kulbrandstad, L., I., Lauvdal, T., Qvortrup, L., . . . F.W., T. (2015). Om lærerrollen – Et kunnskapsgrunnlag. . https://www.regjeringen.no/contentassets/17f6ce332c47437c8935d7ccc0a72769/rapport-om-laererrollen.pdf

Dalland, O. (2020). *METODE OG OPPGAVESKRIVING* (Bd. 7). Gyldendal Akademisk.

Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora. (2021). *Forskningsetiske retningslinjer for samfunnsvitenskap og humaniora*.

. Hentet 18.12.22 fra https://www.forskningsetikk.no/retningslinjer/hum-sam/forskningsetiske-retningslinjer-for-samfunnsvitenskap-og-humaniora/

Det kongelige kirke-, u.-o. f. (1998). *Læreplanverket for den 10-årige grunnskolen*. https://www.nb.no/items/f4ce6bf9eadeb389172d939275c038bb?page=0

Dvergsdal, H. (2019). *Python (programmeringsspråk).* Hentet 30.11.22 fra https://snl.no/Python_-_programmeringsspr%C3%A5k

Elicer, R. & Tamborg, A. L. (2022). Nature of the relations between programming and computational thinking and mathematics in Danish teaching resources. Proceedings of the 15th international conference on technology in mathematics teaching (ICTMT 15), https://www.researchgate.net/publication/360642397_Nature_of_the_relations_between_programming_and_computational_thinking_and_mathematics_in_Danish_teaching_resources

Fauskanger, J. & Mosvold, R. (2015). En metodisk studie av innholdsanalyse–med analyser av matematikklæreres undervisningskunnskap som eksempel. *Nordic Studies in Mathematics Education*, *20*(2), 79-96. https://www.researchgate.net/publication/328139215_En_metodisk_studie_av_innholdsanalyse_-_med_analyser_av_matematikklaereres_undervisningskunnskap_som_eksempel

Feurzeig, W., Papert, S., Bloom, M., Grant, R. & Solomon, C. (1970). Programming-languages as a conceptual framework for teaching mathematics. *ACM SIGCUE Outlook*, *4*(2), 13-17. https://doi.org/10.1145/965754.965757

Gjøvik, Ø. & Torkildsen, H. A. (2019). Algoritmisk tenkning. *Tangenten–tidsskrift for matematikkundervisning*, *30*(3), 31-37. http://tangenten.no/wp-content/uploads/2021/12/Tangenten-3-2019-Gjovik-Torkildsen.pdf

Grover, S. & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Høgheim, S. (2020). *MASTEROPPGAVEN I GLU.* Fagbokforlaget

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M. & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, *82*, 263-279. https://doi.org/10.1016/j.compedu.2014.11.022

Jablonka, E. & Johansson, M. (2010). Using texts and tasks. *The first sourcebook on Nordic research in mathematics education*, 363-372. https://www.researchgate.net/publication/326231382_Using_texts_and_tasks_Swedish_studies_on_mathematics_textbooks

Jensen, M. S., Julien, A., Rafiepour, A. & Schmeding, A. An analytical framework for programming tasks in mathematics textbooks. https://www.researchgate.net/publication/369334923_An_analytical_framework_for_programming_tasks_in_mathematics_textbooks

Johannessen, A., Lufte, P. A. & Christoffersen, L. (2016). *INTRODUKSJON TIL SAMFUNNSVITENSKAPELIG METODE* (Bd. 5). Abstrakt forlag

Kallio, H., Pietilä, A. M., Johnson, M. & Kangasniemi, M. (2016). Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. *Journal of advanced nursing*, *72*(12), 2954-2965. https://doi.org/10.1111/jan.13031

Kaufmann, O. T., Stenseth, B. & Holone, H. (2018). Programmering i

matematikkundervisningen. I (s. 73-96). Cappelen Damm Akademisk.
Kirke- og undervisningsdepartementet. (1987). *Mønsterplan for grunnskolen: M87*. https://www.nb.no/items/2aef891325a059851965d5b8ac193de5
Kirke og undervisningsdepartementet. (1984). *Datateknologi i skolen / Kirke- og undervisningsdepartementet*. https://www.nb.no/items/41d61a08747bda0e3bc819fbfe7a9605?page=0&searchText=oaiid:%22oai:nb.bibsys.no:999412771004702202%22
Kongsnes, A. L. & Wallace, A. K. (2020). *Matemagisk 8*. Aschehoug
Koritzinsky, T. (2014). *Samfunnskunnskap – Fagdidaktisk Innføring*. Universitetsforlaget.
Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.
Kunnskapsdepartementet. (2006). *Læreplan i matematikk fellesfag (MAT1-04). Fastsatt som forskrift*. Læreplanverket for Kunnskapsløftet 2006. https://www.udir.no/kl06/MAT1-04/
Kunnskapsdepartementet. (2017). *Rammeverk for grunnleggende ferdigheter*. https://www.udir.no/laring-og-trivsel/rammeverk/rammeverk-for-grunnleggende-ferdigheter/2.1-digitale-ferdigheter/
Kunnskapsdepartementet. (2019). *Læreplan i matematikk 1.–10. trinn (MAT01-05). Fastsatt som forskrift.* Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nob
Kvale, S. & Brinkmann, S. (2015). *Det kvalitative forskningsintervju* (Bd. 3). Gyldendal akademisk
Kölling, M. (2015). Lessons from the design of three educational programming environments: Blue, BlueJ and Greenfoot. *International Journal of People-Oriented Programming (IJPOP)*, *4*(1), 5-32. https://doi.org/10.4018/IJPOP.2015010102
Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32-37. https://doi.org/10.1145/1929887.1929902
Manheim, K. & Kaplan, L. (2019). Artificial intelligence: Risks to privacy and democracy. *Yale JL & Tech.*, *21*, 106. https://heinonline.org/HOL/LandingPage?handle=hein.journals/yjolt21&div=4&id=&page=
Mayer, R. E., Dyck, J. L. & Vilberg, W. (1986). Learning to program and learning to think: what's the connection? *Communications of the ACM*, *29*(7), 605-610. https://doi.org/10.1145/6138.6142
Meld.St. 42 (1989-1990). *Organisering av informasjonsteknologi i skole og opplæring*. Kirke- og undervisningsdepartement. https://www.stortinget.no/no/Saker-og-publikasjoner/Stortingsforhandlinger/Lesevisning/?p=1989-90&paid=3&wid=c&psid=DIVL833&pgid=c_0575
Misfeldt, M., Jankvist, U. T., Geraniou, E. & Bråting, K. (2020). Relations between mathematics and programming in school: Juxtaposing three different cases. 10th ERME topic conference on mathematics education in the digital era, MEDA 2020, 16-18 September. Linz, Austria, http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1472841&dswid=-277
Mullis, I. V., Martin, M. O., Foy, P. & Arora, A. (2012). *TIMSS 2011 international results in mathematics*. TIMSS & PIRLS International Study Center & International Association for the Evaluation of Educational Achievement. https://timssandpirls.bc.edu/timss2011/downloads/t11_ir_mathematics_fullbook.pdf
Mæland, D., Magnus & Myklebust, M. (2022). Forventning, forvirring og forundring *Tangenten–tidsskrift for matematikkundervisning*, *1*, 20-26.
Nord Universitetet. (2022). *Referansestilen APA 7th*. Hentet 18.12.22 fra https://www.nord.no/no/bibliotek/soke-og-skrivestotte/apa
Norsk senter for forskningsdata. (2022). *Datahåndtering*. Hentet 20.12.22 fra https://www.nsd.no/
NOU 2013: 2. (2013). Hindre for digital verdiskaping. *Kommunal- og distriktsdepartementet*. https://www.regjeringen.no/no/dokumenter/nou-2013-2/id711002/
Nyeng, F. (2021). *NØKKELBEGREPER I FORSKNINGSMETODE OG VITENSKAPSTEORI*. Fagbokforlaget

Opplæringslova. (2022). *§ 9-4.Lærebøker og andre læremiddel (LOV-2022-06-17-68)* Lovdata https://lovdata.no/dokument/NL/lov/1998-07-17-61/KAPITTEL_10#%C2%A79-4

Papert, S. (1980). *" Mindstorms" Children*.

Pea, R. D. & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New ideas in psychology*, *2*(2), 137-168. https://doi.org/10.1016/0732-118X(84)90018-7

Project Jupyter. (u.å). *Jupyter Notebook* Hentet 14.05.2023 fra https://jupyter.org/about

Python Software Foundation. (u.å). *Python.* Hentet 30.11.22 fra https://www.python.org/about/apps/

Remillard, J. T. (2005). Examining key concepts in research on teachers' use of mathematics curricula. *Review of educational research*, *75*(2), 211-246. https://doi.org/10.3102/00346543075002211

Rezat, S. & Sträßer, R. (2015). Methodological issues and challenges in research on mathematics textbooks. *Nordic Studies in Mathematics Education*, *20*(3-4), 247-266. https://ncm.gu.se/wp-content/uploads/2020/06/20_34_247266_rezat.pdf

Rossen, E. (2019). *Store norske leksikon- Programmeringsspråk.* Hentet 29.11.22 fra https://snl.no/programmeringsspr%C3%A5k

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., . . . Voll, L. O. (2016). *Teknologi og programmering for alle - En faggjennomgang med forslag til endringer i grunnopplæringen*. https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/teknologi-og-programmering-for-alle

Scherer, R., Siddiq, F. & Viveros, B. S. (2018). *Technology and the Mind: Does Learning to Code Improve Cognitive Skills?* Proceedings of the technology, mind, and society, https://doi.org/10.1037/edu0000314

Selby, C. & Woollard, J. (2013, 05.03-08.03 2014). *Computational thinking: the developing definition*. Special Interest Group on Computer Science Education, Atlanta https://www.researchgate.net/publication/299450690_Computational_thinking_the_developing_definition

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Silverman, D. (2015). *Interpreting qualitative data* (Bd. 5). SAGE Publications.

Skjelbred, D. A., Norunn, Maagerø, E. & Aamotsbakken, B. (2017). *Norsk lærebokhistorie - Fra allmueskole til grunnskole 1739-2013*. Universitetsforlaget.

Skrunes, N. (2010). *Lærebokforskning - en eksplorerende presentasjon med særlig fokus på Kristendomskunnskap, KRL og Religion og etikk*.

Statped. (2021a). *Programmering for elever med nedsatt syn- Analog programmering (uten datamaskin).* Hentet 30.11.22 fra https://www.statped.no/laringsressurser/syn/temaside-programmering-for-elever-med-nedsatt-syn-temaside/programmering-for-elever-med-nedsatt-syn/analog-programmering-uten-datamaskin/

Statped. (2021b). *Programmering for elever med nedsatt syn - Blokkprogrammering og tekstprogrammering.* Hentet 30.11.22 fra https://www.statped.no/laringsressurser/syn/temaside-programmering-for-elever-med-nedsatt-syn-temaside/programmering-for-elever-med-nedsatt-syn/blokkprogrammering-og-tekstprogrammering/

Stokkenes, J. H. (2022). *Programmering i norske lærebøker i matematikk* [Oslomet]. https://oda.oslomet.no/oda-xmlui/bitstream/handle/11250/3031623/Stokkenes_m5glu2022.pdf?sequence=1&isAllowed=y

Tamborg, A. L., Elicer, R. & Spikol, D. (2022). Programming and Computational Thinking in Mathematics Education. *KI - Künstliche Intelligenz*, *36*(1), 73-81. https://doi.org/10.1007/s13218-021-00753-3

Thonhauser, J. (2008). *Aufgaben als Katalysatoren von Lernprozessen: eine zentrale Komponente organisierten Lehrens und Lernens aus der Sicht von Lernforschung, allgemeiner Didaktik und Fachdidaktik*. Waxmann Verlag.

Trouche, L. (2020). Instrumentation in mathematics education. I *Encyclopedia of mathematics education* (s. 404-412). Springer. https://doi.org/10.1007/978-3-030-15789-0_80

Universitet i Oslo. (2022). *Nettskjema-diktafon mobilapp*. Hentet 18.12.22 fra https://www.uio.no/tjenester/it/adm-app/nettskjema/hjelp/diktafon.html

Utdanningsdirektoratet. (2019a). *Algoritmisk tenkning.* Hentet 30.11.22 fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). *Støtte til læreplan - Tilknyttede læreplanmål*. Hentet 24.12.202 fra https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer?lang=nob&TilknyttedeKompetansemaal=true&curriculum-resources=true

Van Den Ham, A.-K. & Heinze, A. (2018). Does the textbook matter? Longitudinal effects of textbook choice on primary school students' achievement in mathematics. *Studies in Educational Evaluation*, *59*, 133-140. https://doi.org/10.1016/j.stueduc.2018.07.005

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Vedlegg 1 – Sammendrag intervju

Intervju av ‘’Ole’’

Ole er en lærer mellom 45 og 50 år. Han begynte sin lærergjerning i 2001 og jobbet da i videregående skole. Deretter gjennomførte han den 4-årige lærerutdanningen og har jobbet som lærer i både mellom- og ungdomstrinnet siden da. Han har undervist i matematikk i stort sett hele sin yrkeskarriere, sett bort fra noen få år der han var lærer i spesialundervisningsgrupper.

Ole forteller at han har litt erfaring med å undervise programmering i matematikkfaget. Han har brukt programmeringsspråket Scratch for det meste, men har også undervist noe Python. Han forteller også at han har undervist endel analog programmering. Han beskriver det som aktiviteter der elevene skal følge en oppskrift som andre elever gir, for så å se hvorvidt de klarte å følge oppskriften.

Ved spørsmål om utdanning innenfor programmering forteller han at han har vært på et etterutdanningskurs på noen få timer. Han kritiserer det han beskriver som en mangel på utdanning hos lærere i et emne som har fått stor plass i skolen og han etterlyser mer etterutdanning i programmering.

På spørsmål om AT sier han at han hørt om det gjennom læreplanen, men har ikke tatt det videre til undervisning.

På spørsmål om hvor han finner ideer og informasjon for undervisning av programmering i faget svarer han at han stort sett bruker læreboka. Han forteller at han bruker å samle opp programmeringsinnholdet i kapitlene og gjennomføre hele økter med programmering.

I tillegg bruker han også lærebokforlagets nettressurs som inneholder instruksjonsvideoer og mye av de samme oppgavene som i læreboka med en tilhørende Python-leser til hver oppgave der det i noen tilfeller er ferdig skrevet programkode. Han forteller at gjennom arbeid i nettressursen får elevene prøve og feile.

På spørsmål om forholdet mellom matematikk og programmering sier han at det definitivt er en sammenheng mellom emnene. Han sier at programmering kan være en mulighet for læring om mønstre, variabeltenkning, geometri og systemer. Han stiller seg også skeptisk til om undervisning i programmering er nyttig og drøfter om andre læringsmidler som regneark kan

være like egnet til å undervise i disse emnene. Han legger til at programmeringsinnholdet i matematikkfaget er svært nytt og mener at nytten av programmering i faget vil vise seg om noen år når flere elever har fått kompetanse i programmering.

Intervju av ''Anna''

Anna er en kvinne imellom 30 og 35 år. Hun har 4-årig lærerutdanning. Hun har arbeidet som lærer i 6 år og rundt halvparten av tiden som fast matematikklærer på ungdomstrinnet. Før hun ble lærer har hun jobbet i diverse service- og omsorgsyrker.

På spørsmål rundt erfaring innenfor undervisning av programmering i faget sier hun at hun ikke har undervist i det. Hun har heller ingen formell utdanning eller kursing innenfor tema. Hun mener det er problematisk at lærere blir satt til å lære seg dette nye innholdet i faget på egenhånd. Hun legger til at for å sikre bra undervisning må lærerne ha kompetanse i det selv.

På spørsmål om forholdet mellom matematikk og programmering sier hun at det er mange instrumentelle oppgaver i læreboka og at programmering kan være et godt hjelpemiddel dersom en kan det.

Intervju av ''Berit''

Berit er en kvinne i alderen 25 til 30 år. Hun har jobbet som matematikklærer på ungdomstrinnet i 3 år etter at hun ble ferdig med en 4-årig lærerutdanning med 60 studiepoeng i matematikk.

Ifølge seg selv har hun har ingen utdanning eller kursing i programmering. Hun sier at hun så vidt har undervist programmering. I denne undervisningen har hun fulgt programmeringsoppgavene i matematikklæreboka.

På spørsmål om forholdet mellom programmering og matematikk sier hun at det oppleves som en separat del av faget. Hun sier det vanskelig å se sammenhenger mellom emnene da programmering for henne er ukjent. Hun legger til at hun synes det er en utfordring da programmering har blitt innført i læreplanen uten at lærere har fått innføring i emnet verken gjennom lærerstudiet eller i jobbsammenheng.

Vedlegg 2 – Meldeskjema

Sikt

Meldeskjema / Programmering og algoritmisk tenkning i matem... / Vurdering

Vurdering av behandling av personopplysninger

Referansenummer	Vurderingstype	Dato
539696	Standard	07.11.2022

Prosjekttittel
Programmering og algoritmisk tenkning i matematikk

Behandlingsansvarlig institusjon
Nord Universitet / Fakultet for lærerutdanning og kunst- og kulturfag / Grunnskole

Prosjektansvarlig
Alexander Schmeding

Student
Jørn Hammerø

Prosjektperiode
07.11.2022 - 15.05.2023

Kategorier personopplysninger
Alminnelige

Lovlig grunnlag
Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Behandlingen av personopplysningene er lovlig så fremt den gjennomføres som oppgitt i meldeskjemaet. Det lovlige grunnlaget gjelder til 15.05.2023.

Meldeskjema

Kommentar
OM VURDERINGEN
Personverntjenester har en avtale med institusjonen du forsker eller studerer ved. Denne avtalen innebærer at vi skal gi deg råd slik at behandlingen av personopplysninger i prosjektet ditt er lovlig etter personvernregelverket.

Personverntjenester har nå vurdert den planlagte behandlingen av personopplysninger. Vår vurdering er at behandlingen er lovlig, hvis den gjennomføres slik den er beskrevet i meldeskjemaet med dialog og vedlegg.

VIKTIG INFORMASJON TIL DEG
Du må lagre, sende og sikre dataene i tråd med retningslinjene til din institusjon. Dette betyr at du må bruke leverandører for spørreskjema, skylagring, videosamtale o.l. som institusjonen din har avtale med. Vi gir generelle råd rundt dette, men det er institusjonens egne retningslinjer for informasjonssikkerhet som gjelder.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til den datoen som er oppgitt i meldeskjemaet.

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

Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

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

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

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

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

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

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

Ved bruk av databehandler (spørreskjemaleverandør, skylagring eller videosamtale) må behandlingen oppfylle kravene til bruk av databehandler, jf. art 28 og 29. Bruk leverandører som din institusjon har avtale med.

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

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde: https://www.nsd.no/personverntjenester/fylle-ut-

meldeskjema-for-personopplysninger/melde-endringer-i-meldeskjema

Du må vente på svar fra oss før endringen gjennomføres.

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

Lykke til med prosjektet!

Vedlegg 3 - Informasjonsskriv

Vil du delta i forskningsprosjektet
Programmering og algoritmisk tenkning i matematikk?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å finne ut av hvordan undervisningsmateriell i programmering i matematikkfaget gir mulighet for læring i algoritmisk tenkning. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Prosjektet er i forbindelse med at jeg er en masterstudent som skal skrive masteroppgave i matematikkdidaktikk innenfor tema programmering. Formålet med prosjektet er å finne ut av hvordan undervisningsmateriell i programmering i matematikkfaget gir mulighet for læring i algoritmisk tenkning. Det kan bli aktuelt å bruke det som kommer fram i intervjuet til videre forskning innenfor fagfeltene matematikkdidaktikk eller pedagogikk i ettertid av prosjektet. Intervjuet vil da være transkribert og alt av personopplysninger fjernet.

Hvem er ansvarlig for forskningsprosjektet?
Nord Universitet er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?

Du har fått spørsmål om å delta ettersom at du er en lærer som underviser i matematikk. Hvilke lærere som får spørsmål om å delta er på bakgrunn av tilgjengelighet. Planen er samle inn 2-3 intervju.

Hva innebærer det for deg å delta?

Hvis du velger å delta i prosjektet, innebærer det at du svarer på spørsmål gjennom et intervju. Det vil ta deg ca. 30minutter. Intervjuet inneholder spørsmål om alder, yrkeskarriere, forberedelse og gjennomføring av undervisning, matematikk- og programmeringsdidaktikk. Lydopptak på en app for sikker lagring vil bli benyttet under intervjuet.

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

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

Det er kun prosjektleder Jørn Hammerø som vil ha tilgang til det innsamlede datamaterialet før det er transkribert.

For å sikre at uvedkommende får tilgang til personopplysninger kommer intervjuet til å bli tatt opp på en mobilapp som lagrer intervjuet på forskningsserveren til universitet i Oslo. Ingenting vil bli lagret lokalt på mobilen som utfører lydopptaket.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?
Prosjektet vil etter planen avsluttes når oppgaven blir godkjent 15.05.2023. Etter prosjektslutt vil datamaterialet med dine personopplysninger anonymiseres. Innsamlet data anonymiseres ved at alt som kan brukes til å gjenkjenne intervjuobjektet blir fjernet. Det vil si for eksempel personopplysninger og navn på arbeidsplass/skole. Anonymiserte opplysninger vil ikke slettes, men kunne gjenbrukes til for eksempel forskning.

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

På oppdrag fra Nord Universitetet 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:

Nord Universitetet ved Jørn Hammerø mob. +4795482060 Epost: 334532@student.nord.no og veileder Alexander Schmeding mob. +4774022816 Epost: alexander.schmeding@nord.no.

- Vårt personvernombud: Toril Irene Kringen, tlf. 74022750. toril.i.kringen@nord.no

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

Forsker og student Jørn Hammerø
Veileder Alexander Schmeding

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet Programmering og algoritmisk tenkning i matematikk, og har fått anledning til å stille spørsmål. Jeg samtykker til:

☐ å delta i intervju

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

---
(Signert av prosjektdeltaker, dato)