Høgskulen på Vestlandet

MASTEROPPGÅVE

Programmering sitt bidrag til elevars operasjonelle og strukturelle forståing av geometri

The contribution of programming to pupils' operational and structural conceptions of geometry

Madelen Kjøde Aarsheim

Master i undervisningsvitskap med fordjuping i matematikk

Fakultet for lærarutdanning, kultur og idrett (FLKI)

1. juni 2021

Eg stadfestar at arbeidet er sjølvstendig utarbeida, og at referansar/kjeldetilvisingar til alle kjelder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Forord

Mi tid som student på Høgskulen på Vestlandet har vore veldig fin og innhaldsrik. Eg har lært mykje, møtt fantastiske folk og bygd fleire vennskap som forhåpentlegvis vil vare livet ut. Det å avslutte studietida med å skrive ein masteroppgåve har vore utfordrande, men samstundes lærerikt. I arbeidet med masteoppgåva har eg lært mykje som eg gleder meg til å ta i bruk når eg byrjar å undervise til hausten.

Det er fleire personar eg gjerne vil takke for støtte og hjelp i arbeidet med masteroppgåva. Fyrst vil eg takke Inge Olav som har vore ein trygg og god rettleiar gjennom heile masterprosjektet. Eg set pris på at eg har fått rom til å gjennomføre masterprosjektet slik eg sjølv ynskte, samstundes som eg har fått gode råd og konstruktive tilbakemeldingar undervegs.

Eg vil takke dei positive og lærevillige elevane som deltok i studien. Eg vil òg takke elevanes matematikklærar for eit fint samarbeid i forkant av datainnsamling, og rektor som gav meg tillating til å gjennomføre datainnsamlinga.

Tusen takk til mamma, pappa og systrene mine for støtte og kjærleik gjennom masterskrivinga, og i livet generelt. Ein ekstra takk til storesystera mi Susanne som har korrekturlest halve masteroppgåva. Mi gode venninne Emilie fortener òg ein stor takk for å ha korrekturlest den andre halvparten.

Takk til studievenninnene mine for mange, og nokre litt for lange, lunsjpausar der det har vore rom for både latter og frustrasjon. Eg vil òg takke jentene eg bur i kollektiv med for fine avbrekk i masterskrivinga.

Madelen Kjøde Aarsheim
Bergen, 1. juni 2021

Samandrag

Vi lever i eit teknologirikt samfunn som er i stadig utvikling. Dette har stilt krav til utdanningssektoren som skal utruste elevane med viktige kompetansar for framtida. Frå og med hausten 2020 har eit nytt læreplanverk blitt gradvis innført i norske skular. I det nye læreplanverket blir mellom anna programmering inkludert som ein del av matematikkfaget.

Det finst fleire argument for å inkludere programmering i matematikkfaget. Til dømes kan bruken av programmering til å utforske og løyse problem, vere eit godt verktøy for å utvikle matematisk forståing (Utdanningsdirektoratet, 2020). Dette er noko som blir undersøkt ytterlegare i denne oppgåva, gjennom fylgjande problemstilling: *Korleis kan elevar utvikle strukturell og operasjonell forståing av geometri gjennom utforsking i programmeringsspråket Python?*

For å svare på problemstillinga har det blitt forska på seks elevar på 10. trinn som har arbeida i par for å løyse opne geometrioppgåver i Python. Vidare har det blitt valt ut ein arbeidsprosess frå kvart elevpar, som blir presentert i form av samtaleutdrag og skjermbilete frå Python. For å kunne seie noko om elevane sin *strukturelle* og *operasjonelle forståing* for geometri har det blitt utarbeida eit nytt samansett rammeverk bestående av fire omgrep: *problembehandlingskompetanse*, *tankegangskompetanse*, *algoritmisk tenking* og *utforsking.* Dei fire omgrepa er valt ut fordi dei inneheld element som har noko til felles med Sfard (1991) sin beskriving av *strukturell* og/eller *operasjonell forståing.* Dessutan kan dei fire omgrepa vere med på å beskrive elevane sin tilnærming til programmeringsoppgåvene, sidan alle omgrepa på ein eller anna måte handlar om utøvinga av matematikk.

Analysen, som baserer seg på dei fire omgrepa, viser at ein kan utvikle *operasjonell* og/eller *strukturell forståing* av geometri gjennom utforsking i Python. Diskusjonen viser kva rolle Python kan ha spelt i denne utviklinga. Det viser seg at moglegheita til prøving og feiling, samt det at ein må løyse oppgåvene steg-for-steg kan bidra til utvikling av dei to forståingane. I tillegg er det mykje som tyder på at elevar kan ha positivt læringsutbytte av å programmere saman i par.

Abstract

We live in a technology-rich society that is constantly developing. This has resulted in certain demands for the educational sector which is supposed to provide pupils with necessary skills for the future. From the autumn of 2020, a new curriculum has been gradually introduced in Norwegian schools, in which programming is included as part of the mathematics subject.

There are several arguments for including programming in mathematics. For example, the use of programming in inquiry and problem solving can be a useful tool for developing mathematical understanding (Utdanningsdirektoratet, 2020). This is further investigated in this assignment, through the following research question: *How can inquiry through the programming language Python help pupils develope their structural and operational conceptions of geometry?*

To answer the research question, this study focuses on six pupils in 10th grade who have worked in pairs to solve open geometry tasks in Python. One work process has been selected from each pair of pupils and presented as dialog extracts and screenshots from Python. To examine the pupils' structural and operational conceptions, a new framework has been designed. The framework consists of four concepts: *thinking competency*, *problem handling competency*, *algorithmic thinking*, and *inquiry*. The four concepts have been selected because they contain elements that have something in common with Sfard's (1991) description of *structural* and/or *operational conceptions*. In addition, the four concepts can be used to describe the pupils' approach to the programming tasks, since all the concepts in one way or another are regarding the practice of mathematics.

The analysis, which is based on the four concepts, indicates that *operational* and/or *structural conceptions* of geometry can be developed through inquiry in Python. The discussion examines Python's contribution in this development. It appears that the trial-and-error method, along with the fact that one must solve the tasks step-by-step, can contribute to the development of the two mathematical conceptions. The analysis and discussion also indicate that pupils can benefit from programming in pairs.

Innhaldsliste

Figurliste

Tabelliste

1. Innleiing

Dette masterprosjektet tek føre seg programmering og forståing av matematikk på 10. trinn. I dei neste delkapitla blir det presentert bakgrunn for val av tema, problemstilling og oppgåva sin struktur.

1.1 Bakgrunn for val av tema

Vi lev i eit teknologirikt samfunn der dei fleste føreheld seg til fleire digitale einingar dagleg. Utviklinga av det teknologirike samfunnet utfordrar både måten vi lev på i dag og stiller krav til kva kompetansar som blir viktige i framtida. Ei grunnleggjande forståing av teknologiens oppbygging og verkemåtar vil difor vere ein naudsynleg del av framtidas kompetansar, uavhengig av yrkesval (Nygård, 2018, s. 16). Programmering er ein naudsynleg kompetanse for å greie seg i ei ny teknologidriven verd, og bør difor introduserast tidleg i utdanningsløpet (Husain, Kamal, Ibrahim, Huddin & Alim, 2017). Dei siste åra har programmering fått stadig meir politisk fokus, spesielt med tanke på korleis skular og undervisingssektoren handterer naudsynte krav til å bruke og utvikle ny teknologi (Kaufmann & Stenseth, 2020). Ei aukande mengd av land delar det politiske synet om at elevar må få undervising som gjer dei i stand til å lære om og forstå grunnleggjande prinsipp i programmering (Kaufmann & Stenseth, 2020).

Med den norske fagfornyinga i 2020 har programmering blitt ein obligatorisk del av fleire fag i skulen. Programmering har mellom anna blitt ein del av dei grunnleggjande digitale ferdigheitene i matematikk (Utdanningsdirektoratet, 2019b), noko som betyr at skulen må leggje til rette for utvikling av programmering gjennom heile opplæringsløpet. Den nye læreplanen i matematikk blir gradvis innført frå 2020, der den er gyldig på 1.– 9. trinn frå hausten 2020, og på 10. trinn frå hausten 2021 (Utdanningsdirektoratet, 2019b). Som fylgje av fagfornyinga har lærarar og elevar på 1. – 9. trinn rundt om i heile landet sannsynlegvis kome i gong med programmering i matematikkundervisinga, medan elevar og lærarar på 10.trinn framleis held seg til den gamle versjonen av læreplanen i matematikk.

I Udirbloggen (2017) seier dei at ein av hovudgrunnane til at programmering leggast inn i matematikkfaget er at opplæringa av programmering egner seg godt til å kombinere med opplæringa av algoritmisk tenking. Kjernen i algoritmisk tenking er, i fylgje Udirbloggen (2017), å kunne løyse eit problem ved å dele det opp i presise steg ved hjelp av tilgjengelege

grunnoperasjonar. Algoritmar finst både i den digitale og analoge kvardagen vår. Det er difor viktig at elevar får eit bevisst forhold både til korleis ulike algoritmar fungerer, og korleis ein kan lage og vurdere enkle algoritmar (Haraldsrud, Sveinsson & Løvold, 2020, s. 184). Datamaskiner kan utføre eit stort spekter av algoritmar mykje raskare og meir presist enn menneske, utan å oppfatte det som keisam terping. For å gje liv til algoritmisk tenking er det difor viktig at det kombinerast med læring av programmering (Udirbloggen, 2017).

I ein artikkel på Utdanningsdirektoratet (2020) sine nettsider blir dei viktigaste endringane i matematikkfaget forklart. Der seier dei at matematikkfaget skal legge vekt på at elevane skal bli gode problemløysarar og oppdage samanhengar i og mellom faget sine kunnskapsområder og andre fag sine kunnskapsområder. Det er desse samanhengane som legg til rette for blant anna forståing i faget. Vidare peikar dei på at algoritmisk tenking er synleggjort fordi dette er ein viktig problemløysingsstrategi. Matematikkfaget skal førebu elevane på eit samfunn og arbeidsliv i utvikling ved å gje dei kompetanse i utforsking og problemløysing (Utdanningsdirektoratet, 2019b). Når elevane brukar programmering til å utforske og løyse problem, kan det vere eit godt verktøy for å utvikle matematisk forståing (Utdanningsdirektoratet, 2020).

Det finst utallege forklaringar på kva matematisk forståing er. Matematisk forståing vart i siste halvdel av 1900-talet i fleire tilfelle beskrive som dikotomiar. Ein dikotomi er noko som er delt i to, slik som instrumentell og relasjonell forståing av Skemp (1976) eller konseptuell kunnskap og prosedyrekunnskap av Hiebert og Lefevre (1986). Ein annan dikotomi er Sfard (1991) si beskriving av termane *operasjonell* og *strukturell forståing*. Til tross for at Sfard sin dikotomi har fleire fellestrekk med dei to andre dikotomiane som er nemnt, hevdar ho sjølv at det finst to fundamentale eigenskapar som skil hennar dikotomi frå resten. Det er kombinasjonen av ontologi og psykologi, og at den er komplementær (Sfard, 1991, s. 8). Ho fokuserer på kjenneteikna til matematiske einingar (ontologi) og korleis desse blir oppfatta av tenkaren (psykologi). Dessutan peikar ho på at andre dikotomiar deler matematisk kunnskap i to ulike komponentar, medan hennar dikotomi er komplementær og utfyllande. Ho hevdar at ho presenterer ein dualitet framfor ein dikotomi. Grunnen til det er at termane *operasjonell* og *strukturell* er to uskiljande, men samstundes ulike sider av same sak. Sfard (1991) si beskriving av dei to forståingane vil bli ytterlegare beskrive i teorikapittelet, då det er desse forståingane som vil bli vektlagt i teksten.

1.2 Studien sitt formål og problemstilling

Mi interesse for programmering oppstod våren 2020 etter å ha hatt ei introduksjonsøkt i det blokkbaserte programmeringsspråket Scratch på høgskulen. Eg kunne ingenting om programmering frå før, og det var på dette tidspunktet eg innsåg at eg ein dag måtte vere i stand til å integrere programmering i undervisinga mi. Eg brukte difor litt av fritida mi på å bli betre kjent med Scratch. Etter kvart oppdaga eg at programmeringsverda i stor grad handlar om tekstbasert programmering. Eg vart såleis merksam på Python som viste seg å vere eit programmeringsspråk som er veleigna til å bruke i skulen, spesielt på ungdomstrinnet. Det vart difor naturleg å bruke Python i dette masterprosjektet.

Som nemnt innleiingsvis kan bruken av programmering til å utforske og løyse problem, vere eit godt verktøy for å utvikle matematisk forståing (Utdanningsdirektoratet, 2020). Dette er noko eg skal undersøke ytterlegare i denne oppgåva, gjennom å spisse meg inn på programmeringsspråket Python og utvikling av *operasjonell* og *strukturell forståing*. Det har dermed blitt utforma fylgjande problemstilling:

Korleis kan elevar utvikle strukturell og operasjonell forståing av geometri gjennom utforsking i programmeringsspråket Python?

For å svare på problemstillinga har det blitt forska på seks elevar på 10. trinn som har arbeida i par for å løyse opne geometrioppgåver i Python. Målet er å identifisere om elevane viser *operasjonell* og *strukturell forståing* for geometri, og vidare undersøke korleis programmering kan bidra til at denne forståinga kan utvikle seg. For å kunne seie noko om elevane si strukturelle og operasjonelle forståing av geometri har det blitt utarbeida eit nytt samansett rammeverk bestående av fire omgrep som på ulike måtar handlar om utøvinga av matematikk: *problembehandlingskompetanse*, *tankegangskompetanse*, *algoritmisk tenking* og *utforsking.* Desse fire omgrepa er valt ut fordi dei inneheld element som har noko til felles med strukturell og/eller operasjonell forståing. Ved å bruke desse omgrepa til å studere datamaterialet har eg hatt eit grunnlag for å kunne seie noko om forståinga elevane viser. Kvart enkelt omgrep vil bli ytterlegare forklart i teorikapittelet, medan det samansette rammeverket vil bli gjort greie for i metodekapittelet.

1.3 Oppgåva sin struktur

I dette kapittelet har bakgrunn for val av tema, samt studien sitt formål og problemstilling blitt beskrive. I kapittel 2 presenterast teorien som studien baserer seg på. Her går eg fyrst inn på tidlegare forsking om programmering. Deretter presenterast dei fire omgrepa som er med på å utgjer rammeverket i studien. Til slutt blir det gjort greie for strukturell og operasjonell forståing, slik som beskrive av Sfard (1991).

I kapittel 3 blir oppgåva sin forskingsmetode og datainnsamlingsprosessen forklart. Det vil bli gjort greie for førebuinga til datainnsamlinga, korleis sjølve datainnsamlingsprosessen gjekk føre seg og korleis materialet vart behandla og analysert i etterkant. Her presenterast òg det nye samansette rammeverket som brukast i analysen. Til slutt vil kapittelet ta føre seg oppgåva sin validitet, reliabilitet og etiske omsyn.

I kapittel 4 blir samtaleutdraga frå tre elevpar analysert og tolka i tre delkapittel (4.1-4.3). Kvart delkapittel fokuserer på samtala til eit elevpar, samt utklipp frå aktiviteten deira i Python medan dei løyser ei geometrioppgåve. Dei fire omgrepa *problembehandlingskompetanse*, *tankegangskompetanse*, *algoritmisk tenking* og *utforsking* har blitt brukt til å analysere samtaleutdraga til elevpara. Ved hjelp av det nye samansette rammeverket blir desse omgrepa vidare brukt til å beskrive elevane si *operasjonelle* og *strukturelle forståing* for geometri. Her blir det vurdert om elevane har operasjonell og strukturell forståing frå før, og om denne forståinga utviklar seg.

I kapittel 5 blir sentrale funn frå analysen løfta fram for å kunne drøfte rolla til Python i elevane si utvikling av operasjonell og strukturell forståing av geometri. Vidare vil det bli sett nærmare på kva betyding det kan ha å vere to elevar som programmerer saman. Til slutt blir det retta fokus mot kva konsekvensar programmering i Python kan ha for matematikkundervising.

Avslutningsvis blir det gjort ei oppsummering av oppgåva i kapittel 6. I tillegg presenterer eg nokre avsluttande refleksjonar knytt til forskingsprosjektet og kva som kan vere interessant å forske vidare på.

2 Teori

2.1 Tidlegare forsking

I ein litteraturstudie av Forsström og Kaufmann (2018), går dei gjennom 15 relevante artiklar for og kartlegge grunnane for å implementere programmering i matematikkfaget. Dei fann ut at i nokre tilfelle, kan det å integrere programmering i matematikkundervisinga vere med på å motivere elevane og heve prestasjonane deira i matematikk. Fem av artiklane i litteraturstudien rettar fokus mot elevane sitt læringsutbytte ved å sjå på endringar i karakterar og testresultat. Alle studiane hadde noko positiv å seie om elevane si matematiske utvikling etter ei testperiode med programmering.

I ein studie av Popat og Starkey (2019) undersøker dei ti forskingsartiklar for å finne ut kva læringsutbytte elevar kan få av å lære seg å kode på skulen. Bakgrunnen for at dei vil undersøke dette er at oppblomstringa av programmering i skulen kjem med eit løfte om å førebu elevar på ei framtid som går utover det å kode. Resultata viser at sjølv om elevane lærer å kode, kan dei i tillegg ha anna læringsutbytte av sjølve kodinga. Koding kan mellom anna kan bidra til at elevane styrkar sine sosiale ferdigheiter og blir betre på matematisk problemløysing.

Problemløysing er eit openbart tema i alle dei ti studiane Popat og Starkey (2019) undersøker. I ni av studiane testa dei elevane sine evner til problemløysing ved å gje dei kodingsoppgåver der dei måtte iverksetje slike evner for å handtere dei matematiske konsepta i oppgåva. Den tiande studien peikar på at i prosessen der elevane lærer korleis dei skal programmere, lærer dei òg matematiske ferdigheiter som gjer dei betre på problemløysing. I to av studiane peikar dei på at sosiale ferdigheiter, noko som inneber samarbeid, kan bli utvikla når ein lærer å kode eller programmere. Sosiale ferdigheiter handlar om å kunne ha effektiv kommunikasjon med andre, noko som skjer gjennom eit positivt samarbeid (Popat & Starkey, 2019, s. 268). Ein studie peikar på at koding legg til rette for interaksjon mellom elevar, noko som kan føre til utvikling av sosiale ferdigheiter. Den andre studien føreslår at for å fremje samarbeid, så bør elevar arbeide i par for å dele kunnskap og forståing, og ta stilling til kvarandre sine synspunkt.

Husain et al. (2017) sin studie viser at det å lære seg programmering kan blant anna bidra til at elevar utviklar matematisk forståing. I studien tek dei føre seg undervising av grunnleggjande matematiske konsept med bruk av programmering. Dei oppdagar at nokre elevar tar aktivt i bruk dei nye konsepta i programmeringa og viser tydelege evner til problemløysing og logisk

tenking. Før- og etter-prøver i studien viser ei tydeleg auking i elevane si forståing for gjevne konsept etter undervising i programmering.

2.1.1 Programmering

Forsström og Kaufmann (2018, s. 19) forklarar at programmering er prosessen som er knytt til utviklinga og implementeringa av instruksjonar til eit dataprogram som får ei datamaskin til å utføre oppgåver, løyse problem og støtte mennesket sine interaksjonar. Vidare peikar dei på at programmering krev programmerarar som har kunnskap om programmeringsspråk og ekspertise om tema som er knytt til utviklinga av spesialiserte algoritmar og logikk. Dessutan krevjast det programmerarar med evne til å analysere, forstå og løyse problem ved å verifisere algoritmiske krav og vurdere korrektheita og implementeringa (ofte omtalt som koding) av algoritmen i eit bestemt programmeringsspråk. I fylgje Nygård (2018, s. 13) finst det truleg opp mot 2000 programmeringsspråk. Nokon har gått ut av bruk, og mange er variantar av større språk, omtrent som dialektar. Ho seier at ulike programmeringsspråk er gode til forskjellige ting, og at Python er eit av dei vanlegaste språka å byrje med (Nygård, 2018, s. 14).

Programmeringsspråket Python vart lansert på 90-talet av den nederlandske informatikaren Guido van Rossum (Bueie, 2019, s. 30). Bueie (2019, s. 30) seier at programmeringsspråket har mange styrkar som gjer at det ser ut som at dette blir det naturlege valet i matematikkundervisingssamanheng i åra som kjem. Ein styrke er at Python er fritt tilgjengeleg og krev liten maskinkraft for å kunne køyrast. I tillegg fungerer Python godt på både windows-, linux- og MAC OSX-baserte system og er slik sett tilgjengeleg på dei fleste plattformar. Dessutan er Python eit programmeringsspråk med låg inngangsterskel og høg takhøgd. Han forklarer dette med at det er lett å kome i gong med å programmere i språket, samstundes som språket i liten grad er til hinder for kva som er mogleg å få til av program. Måten eit programmeringsspråk skrivast på blir ofte referert til med ordet syntaks, og i fylgje Bueie (2019, s. 31) har Python ein lettlest og god syntaks. Han peikar òg på fordelen med det universelle med språket. Når ein har lært seg Python, vil overgangen til andre programmeringsspråk vere enklare.

Haraldsrud et al. (2020, s. 14) seier at den reine syntaksen i Python gjer det enklare å oversetje matematikk til Python-kode. Han peikar på at *skilpadde* (eit bibliotek i Python) egner seg godt til å kombinere med geometri og trigonometri. Elevane kan bruke Pytagoras' setning, formlikhet, kunnskap om ulike typar trekantar eller trigonometri for å løyse geometrioppgåver

ved hjelp av *skilpadde*. Fordelen med å programmere dette, er at ein må gå gjennom kvart teiknetrinn eksplisitt. Dessutan er det enkelt å teste ut og sjekke om ein har gjort riktig (Haraldsrud et al., 2020, s. 120). Dersom ein skal teikne regulære mangekantar er det nyttig å bruke løkker, slik at ein prosedyre kan gjentakast fleire gongar (Haraldsrud et al., 2020, s. 117). I ein studie av Jackson, Stenger, Jerkins & Terwilliger (2019) fann dei ut at programmering ved bruk av *skilpadde* kan bidra til at ein dannar seg mentale bilete og dermed utviklar evna til abstraksjon. Informantane som hadde delteke i aktivitetar knytt til skilpadde-programmering viste ei høgare abstraksjonsevne etter introduksjon av gjevne geometriske konsept, enn dei informantane som fekk tradisjonell undervising.

2.1.2 Parprogrammering

Når elevar skal programmere kan det vere nyttig å programmere i par, då det kan bidra til færre feil og auka effektivitet (Nygård, 2018, s. 46). I fylgje Haraldsrud et al. (2020, s. 171) går parprogrammering ut på at to og to arbeidar med éi kode i fellesskap. Då brukar dei berre éin datamaskin til å skrive koden, og dei kan bytte på kven som har tastaturet. Det er viktig at dei diskuterer gjennom heile prosessen. Den som ikkje skriv kodar, skal kome med innspel til korleis kodelinjene kan bli betre og meir effektive. Det er òg viktig at den som ikkje programmerer stiller spørsmål dersom noko som er uklart.

2.2 Opne oppgåver

Når elevar skal parprogrammere i matematikk, så kan oppgåvene dei får tildelt ha innverknad på kva utbytte dei får av programmeringa. I fylgje Hana (2013, s. 238) finst det mange måtar å klassifisere oppgåver på. Han seier at eit skilje kan vere mellom problem og øvingar. Eit problem er ei oppgåve som eleven ikkje veit korleis ho skal løyse når ho set i gong, medan ei øving er ein oppgåvetype som eleven kjenner til frå før. Kva som er eit problem eller ei øving varierer frå person til person, og avhenger av den matematiske bakgrunnen til personen.

Hana (2013, s. 238) peikar på at det er vanleg å skilje mellom opne og lukka oppgåver innan matematikkdidaktikk. I ei lukka oppgåve er målet eintydig formulert i oppgåveteksten, og det finst berre eit riktig svar. Han hevdar at oppgåver kan opnast på fleire måtar, der ein av måtane kan vere å la elevane bestemme kva metode dei skal bruke for å løyse oppgåva. I staden for at læraren eller læreboka gjev retningslinjer for korleis ei oppgåve skal løysast, kan ein opne dei opp slik at ulike metodar blir gyldige og naturlege å ta i bruk. Hana (2013, s. 239) argumenterer

for at opne oppgåver gjev høve til å diskutere ulike tolkingar og alternative løysingar, noko som kan gjere det lettare å vere kritisk og reflekterande over matematikkfagets innhald. Dessutan seier han at opne oppgåver vil krevje meir av eleven, då ho må forhalde seg til fleire alternativ og gjere matematiske val. Det kan difor tenkjast at opne oppgåver kan stimulere til utforsking i matematikk.

2.3 Utforsking

I dei to siste tiåra har omgrepet utforskande læring (inquiry-based learning) i matematikk fått stadig meir merksemd innan utdanningspolitikk og læreplandokumenter. I den nye læreplanen blir *Utforsking og problemløysing* rekna som eit av kjerneelementa i matematikk, og utforsking blir definert på fylgjande måte:

> «Utforsking i matematikk handlar om at elevane leiter etter mønster, finn samanhengar og diskuterer seg fram til ei felles forståing. Elevane skal leggje meir vekt på strategiane og framgangsmåtane enn på løysingane» (Utdanningsdirektoratet, 2019b).

Andersen, Fiskum og Rosenlund (2018) seier at utforskande læringsaktivitetar blant anna kan forståast som undersøkande arbeidsmetodar der elevane skal finne ut av ei problemstilling. Det er læraren som set mål og rammer for det som skal undersøkast, medan elevane vel strategi. Dei peikar på at utforskande arbeidsmetodar er oppdagingsorienterte, samstundes som dei dannar eit godt grunnlag for samarbeidslæring. I fylgje Artigue og Blomhøj (2013, s. 797) kan utforskande pedagogikk bli sett på som ein undervisningsmetode der elevane får arbeide på liknande måtar som det matematikarar gjer. Med utgangspunkt i ulike teoretiske rammeverk har dei kome fram til fleire sentrale element i utviklinga og implementeringa av utforskande verksemd i klasserommet. Det må blant anna bli lagt opp til ei eksperimentell tilnærming til matematikk der elevane får utvikle undersøkande vaner og evna til å løyse problem. I tillegg må elevane bli tildelt autonomitet og ansvar i alle steg frå å formulere spørsmål til å validere svar. Samarbeid blir òg fremja som ein viktig del av utforskingsprosessen (Artigue & Blomhøj, 2013, s. 809).

Sikko og Grimeland (2020) beskriv utforskande og undersøkande arbeidsmåtar som moglege inngangar til læring der ein utviklar ei spørjande haldning og auka motivasjon. Dei meiner at eit utforskande læringsmiljø bør vere prega av relevante oppgåver som er opne og/eller kan løysast på fleire måtar. Elevane skal få søkje etter forklaringar i staden for øve på gjevne reknemåtar. Læraren skal oppmuntre elevane til å resonnere, i staden for å berre fortelje dei i

detalj kva dei skal gjere (Sikko & Grimeland, 2020, s. 106). Utbyttet av undervisinga bør vere at elevane utviklar seg til å bli tenkjande, kreative og kritiske. Dette er viktig med tanke på at dei skal bli utrusta for ei usikker framtid. Sikke og Grimeland (2020, s. 106) fremjar i tillegg at det er viktig at elevane utviklar forståing for at matematikk er eit fag i utvikling der det er essensielt å stille spørsmål, undersøke og prøve seg fram.

Trends in International Mathematics and Science Study (TIMSS) er ein internasjonal studie som blir gjennomført kvart fjerde år, der eit av dei viktigaste føremåla er å kunne presentere data av høg kvalitet om elevar sine prestasjonar i matematikk (Bergem, Kaarstein & Nilsen, 2016, s. 11). Viktige funn frå TIMSS 2015 viser at undervisningskvalitet har ein signifikant og positiv samanheng med elevar sine prestasjonar i matematikk (Bergem, 2016, s. 175). Funna viser blant anna at elevar som har lærarar som gjev faglege/kognitive utfordringar, tenderer til å ha betre læringsresultat enn elevar som ikkje har slike lærarar. Dersom ein ser dette i samanheng beskrivingar av utforskande læringsmiljø (Artigue & Blomhøj, 2013; Sikko & Grimeland, 2020), er det fleire faktorar som tyder på at utforsking kan bidra til faglege/kognitive utfordringar og dermed betre læringsresultat i matematikk.

2.3.1 Utforsking med digitale verktøy

Utforsking med digitale verktøy kan handle om å stille kritiske spørsmål og reflektere rundt matematiske problem. Det kan òg handle om korleis programvarer og digitale verktøy kan brukast i undervising og læring av matematikk (Fuglestad, 2010, s. 95). For å kunne utforske med digitale verktøy må ein ha tilgang på ei open programvare som gjev høve til å undersøke og eksperimentere. Fuglestad (2010, s. 105) hevder at slike programvarer er meir utfordrande enn oppgåveretta programvarer som brukast til terping og øving.

Arbeidet med å introdusere programvarer i matematikk er utfordrande, og det krev innsats for å utvikle kompetanse. Det å kunne bruke ein del av ei programvare i undervising krev meir enn å berre jobbe med det på eiga hand (Fuglestad, 2010, s. 105). Lærarar bør utvikle ulike delar av programvara til å bli instrument for matematikk, ikkje berre for deira eige arbeid, men for å kunne forme og styre elevane sine allereie eksisterande erfaringar med programvara (Fuglestad, 2010, s. 106). Eit «instrument» viser til ein tankemodell hos brukaren som utviklar sitt mentale skjema etter kvart som ho/han blir kjent med, forstår og brukar ein artefakt (Fuglestad, 2010, s. 93). Når ein digital artefakt utviklar seg til å bli eit personleg instrument som ein har djup kunnskap om, dannar dette grunnlag for undersøkande bruk og den digitale artefakten blir

dermed eit instrument for utforsking (Fuglestad, 2010, s. 105). Utforsking med digitale verktøy som instrument for undervisning vil tilby nye arbeidsarenaer der læraren ikkje alltid treng å vete svaret (eller svara). Ei utforskande tilnærming kan innebere at læraren og elevane undersøker og utforskar matematikk saman. Digitale verktøy kan tilby interessante situasjonar for dette (Fuglestad, 2010, s. 106).

2.4 Algoritmisk tenking

Arbeidet med programvarer i matematikk har ei tett tilkopling til algoritmisk tenking. Å tenkje algoritmisk er å vurdere kva for nokre steg som skal til for å løyse eit problem, og å kunne bruke sin teknologiske kompetanse for å få ei datamaskin til å løyse heile eller delar av problemet (Utdanningsdirektoratet, 2019a). Dette handlar i tillegg om å forstå kva slags problem/oppgåver som kan løysast med teknologi og kva som bør overlatast til menneske.

Viktige kjenneteikn hos den algoritmiske tenkaren er å vere skapande, eksperimenterande og open for alternative løysingar. Tenkaren brukar sin nysgjerrighet og utforskande tilnærming for å formulere og løyse problem. Å gjere feil er ein viktig del av prosessen, og den algoritmiske tenkaren må ha strategiar for å oppdage feila og rette på dei (Utdanningsdirektoratet, 2019a). Det finst mange måtar å definere algoritmisk tenking på. Figuren under viser dei mest samanfallande hovudtrekka:

Figur 1: Den algoritmiske tenkaren (Utdanningsdirektoratet, 2019a)

Typiske arbeidsmåtar og sentrale nøkkelomgrep blir presenterte i figuren. Eit av nøkkelomgrepa er å kunne kjenne igjen mønster. I fylgje Haraldsrud et al. (2020, s. 189) handlar mønstergjenkjenning i algoritmisk tenking om å kjenne att strukturar og likskapar mellom ulike prosessar og rutinar. Når mønstera i eit problem er analysert, kan ein nytte likskapane i mønstera til å konstruere algoritmar.

I fylgje Nygård (2018, s. 8) er algoritmisk tankegang (computational thinking) strategiar for problemløysing som blant anna brukast i programmering. Ho peikar på at programmering handlar om å lage sett av reglar og uttrykk for å styre digitale einingar. Her inngår prosessen frå å identifisere problem og utforme moglege løysingar, til å lage kodar som kan forståast av ei datamaskin. Vidare skal ein systematisk feilsøke og forbetre denne koda, og dokumentere løysinga på ein forståeleg måte. Dette omfattar alle stega frå å føresjå og analysere kva eit program skal gjere, til å kjenne att mønster, eksperimentere og evaluere moglege løysingar. Dette bør ein òg kunne gjere i samarbeid med andre. Nygård (2018, s. 7) kallar summen av desse ferdigheitene for algoritmisk tankegang.

2.5 Prøving og feiling

I algoritmisk tenking blir det å prøve ut og gjere feil sett på som ein viktig del av prosessen (Utdanningsdirektoratet, 2019a). I fylgje Hana (2014, s. 218) står prøving og feiling sentralt når ein skal løyse problem både i matematikk og elles. Han hevdar at ved rein prøving og feiling er det tilfeldig om ein finn svaret. Det er heller meir fornuftig å gjennomføre ein systematisk form for prøving og feiling der ein gjennom å studere tidlegare forsøk kjem nærmare ei løysing for kvar gong ein prøver på nytt. Han peikar òg på at det stort sett vil vere mest effektivt å prøve nokre få tilfelle og tenkje grundig gjennom kva som skjer i desse, enn å ukritisk prøve mange tilfelle.

Både lærarar og elevar vil støyte på mange feil i koda når dei byrjar å programmere (Haraldsrud et al., 2020, s. 35). Dette kan vere småting som eit kolon, eit innrykk eller ein feilskriven kommando. Det kan òg vere meir omfattande ting som går på misforståingar av større konsept og prinsipp. Haraldsrud et al. (2020, s. 35) seier at det er både lov og lurt å feile, sidan ein kan lære mykje av det. Vidare peikar dei på at med datamaskiner så er det lett å prøve å feile utan at det gjev store konsekvensar, noko som er ein stor fordel både pedagogisk og fagleg.

I ein studie av Kaufmann og Stenseth (2020) tek dei føre seg ungdomsskuleelevar sine argument når dei skal løyse eit matematisk problem ved bruk av programmering. I studien viste det seg at prøving og feiling kan ha hatt ei negativ verknad på dei matematiske argumenta til elevane. Moglegheita til å prøve og feile kan ha gjort terskelen for å prøve nye ting lågare, noko som kan ha ført til at elevane ikkje kjenner eit behov for å argumentere.

2.6 Kommunikasjon ved bruk av PC

Samarbeid blir sett på som viktig både innan utforsking og algoritmisk tenking (Artigue & Blomhøj, 2013; Utdanningsdirektoratet, 2019a). For å få eit innblikk i om elevar faktisk samarbeider, kan ein sjå på kommunikasjonsmønstera til elevane. Herheim og Krumsvik (2011) tek føre seg kommunikasjonsmønstera til to elevpar som jobbar med matematikk på PC. Det viser seg at elevpar 1 samarbeider og kommuniserer meir enn elevpar 2. Eit kommunikasjonsmønster hos par 1 er at dei brukar pronomen som «vi» og «oss» når dei snakkar om det dei gjer. Herheim og Krumsvik (2011) hevdar at bruken av slike pronomen tydeleggjer det sosiale aspektet i samtala og styrkar den felles forståinga. Dei peikar òg på at par 1 stiller kvarandre spørsmål, og at utsegnene deira er retta mot den andre eller mot begge. Dette gjer at dei fungerer som ei eining og ikkje som to individuelle elevar.

Eit anna sentralt kommunikasjonsmønster hos elevpar 1 er at dei er synkroniserte ved at dei snakkar i kor og byggjer vidare på kvarandre sine setningar. Herheim og Krumsvik (2011) brukar Ole, Dole og Doffen som ein metafor til å skildre denne måten å kommunisere på, då dei er kjende for å fullføre kvarandre sine setningar, og å lage setningar saman der alle kjem med korte bidrag. Herheim (2011, s. 65) seier at når ein ser slike kommunikasjonsmønster hos elevar, kan ein vere rimeleg sikker på at elevane har eit felles fokus og at dei verkeleg samtalar. Han hevdar at ein elev ikkje klarar å fullføre den andre si setning på ein fornuftig måte utan at ho både har lytta til den andre, og i tillegg har forstått mykje av korleis den andre tenkjer.

2.7 Matematiske kompetansar

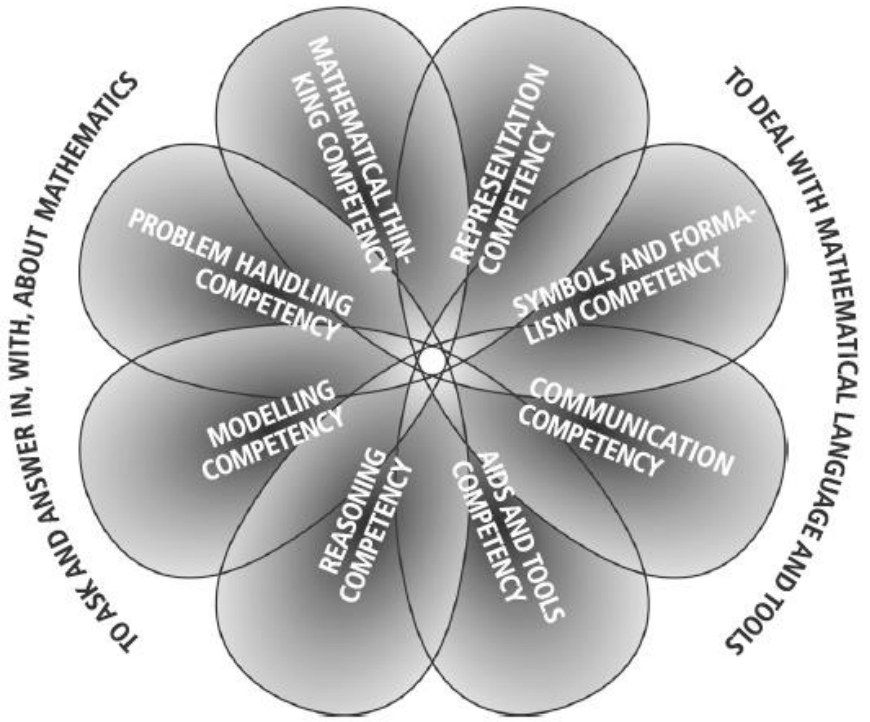
Dei to siste tiåra har ideen om matematisk kompetanse blitt stadig meir anerkjent innan matematisk utdanning, forsking, utvikling og praksis (Niss & Højgaard, 2019). Det danske KOM-prosjektet (KOM: Competencies and the Learning of Mathematics) si publisering av ein rapport frå 2002 (Niss, 2003), der Niss og Højgaard var forfattarar, har spelt ei betydeleg rolle i denne utviklinga (Niss & Højgaard, 2019). Sidan utviklinga har gått mykje framover dei to

siste tiåra, har Niss og Højgaard (2019) valt å lage ein oppdatert versjon av det originale rammeverket og terminologien. Samstundes som den oppdaterte versjonen bevarer sentrale element frå den originale rapporten, legg den òg til fleire forklaringar som er tydelegare og meir opplysande.

Når det er snakk om matematisk kompetanse legg Niss og Højgaard (2019, s. 11) vekt på utøvinga av matematiske aktivitetar og prosessar. Dei definerer matematisk kompetanse som «someone's insightful readiness to act appropriately in response to all kinds of *mathematical* challenges pertaining to given situations» (Niss & Højgaard, 2019, s. 12). Kva som blir rekna som ei utfordring (challenge) avhenger både av situasjonen og personen. Det same gjeld når det er snakk om kva det betyr å møte denne utfordringa. I kva grad ein møter ei utfordring er avhengig av kven «dommarane» er, då det er dei som avgjer kva som skal til.

Niss og Højgaard (2019) deler åtte matematiske kompetansar inn i to kategoriar, der begge kategoriane består av fire kompetansar. Fyste kategori handlar om å stille og svare på spørsmål om og ved hjelp av matematikk, medan den andre kategorien går ut på å handtere språket, omgrep og verktøy i matematikk. Desse er illustrert i figuren under:

Figur 2: Ein visuell representasjon av dei åtte kompetansane (Niss & Højgaard, 2019, s. 19)



Denne oppgåva legg vekt på to kompetansar frå fyrste kategori: problem handling competency (*problembehandlingskompetanse*) og mathematical thinking competency (*tankegangskompetanse)*. Grunnen til at desse kompetansane havnar i fokus er at dei er mest relevante for å belyse problemstillinga i oppgåva, då dei tilsynelatande vil kunne seie mest om datamaterialet.

Tankegangskompetanse går ut på å ta del i undersøkande matematikk, noko som blant anna handlar om å forhalde seg til matematiske spørsmål ved å kunne stille spørsmål sjølv og vete kva svar ein kan forvente å få (Niss & Højgaard, 2019, s. 15). Vidare handlar det om å ta stilling til det varierande omfanget til matematiske konsept i ulike kontekstar og å skilje mellom ulike typar matematiske uttrykk slik som definisjonar, påstandar og hypotesar. Til slutt involverer det å ta stilling til og utøve abstraksjonar av konsept, og generalisere påstandar som prosessar i matematisk aktivitet.

Problembehandlingskompetanse omhandlar å handtere matematiske problem, altså problem som har blitt gjeve innan eit matematisk univers. I omgrepet «problem» ligg det meir enn direkte tilnærmingar og prosedyrar som er rutinemessige for problemløysaren. Problembehandlingskompetansen går ut på å kunne identifisere, stille opp og løyse ulike problem innan ulike matematiske områder (Niss & Højgaard, 2019, s. 15). Ein må i tillegg kunne vurdere gyldigheita av eigne og andre sine løysingar på slike matematiske problem. Eit viktig aspekt med denne kompetansen er at ein må kunne tenkje ut og iverksetje strategiar for å løyse matematiske problem. Det er hensiktsmessig å påpeike at omgrepet «problem» er relativt, då det som kan oppfattast som eit problem for ein person kan vere ei standard oppgåve for nokon andre.

Niss og Højgaard (2019, s. 25) seier at hovudgrunnen til at dei tek for seg kva det inneber å vere matematisk kompetent, er at dei ynskjer å danne eit grunnlag for undervisning og læring av matematikk som reflekterer utøvinga av matematikk på ein forståeleg måte. Dei meiner at kompetansane kan brukast som eit analytisk verktøy til å beskrive og karakterisere tilstanden til kompetansen som ein prøver å oppnå. Dessutan kan kompetansane brukast til å finne framgangsmåtar for å avdekke og karakterisere nøkkelelement i matematikklæringa til den enkelte elev.

Det er viktig at dei ulike kompetansane ikkje blir sett på som uavhengige seier Niss og Højgaard (2019, s. 19). Sjølv om den enkelte kompetansen er veldefinert og skil seg ut frå resten, vil den vere overlappande overfor dei sju andre. Når ein kompetanse er i fokus vil nokre eller alle dei andre kompetansane, avhengig av situasjon og kontekst, kome til syne og fungere som stønad. Dette er synleg i illustrasjonen av dei åtte kompetansane (Figur 2), der ein kan sjå ein blomster med åtte kronblad der kvart kronblad er unikt og samstundes overlappande med resten.

2.7.1 Tre dimensjonar av ein kompetanse

Ein kan aldri fullstendig inneha ein matematisk kompetanse, då matematiske kompetansar blir utøvd i ulike kontekstar (Niss & Højgaard, 2019, s. 21). Det finst ingen ende på situasjonar der kompetansane kan oppstå, og det finst fleire ulike måtar å aktivere kompetansane på. Arbeidet mot å få ein fullstendig kompetanse kan difor sjåast som eit prosjekt i stadig utvikling. På bakgrunn av dette har Niss og Højgaard (2019, s. 21) identifisert tre dimensjonar av personar sin kompetanse. Den fyrste er *degree of coverage*, altså i kor stor grad ein kompetanse er dekka. Dette inneber i kva grad ein person har dekka alle aspekta som utgjer og karakteriserer ein kompetanse. Den andre dimensjonen er *radius of action*. Denne dimensjonen representerer det vide spekteret og variasjonen av kontekstar der ein kan aktivere ein kompetanse. Den tredje og siste dimensjonen er *the technical level.* Den angjev vanskegrada av matematiske konsept, resultat, teoriar og metodar som ein person tar stilling til under utøvinga av ein kompetanse (Niss & Højgaard, 2019, s. 21).

Dersom ein person manglar ein eller fleire av dei tre dimensjonane av ein kompetanse, så har ikkje personen kompetansen i det heile tatt (Niss & Højgaard, 2019, s. 22). Dei tre dimensjonane kan bli brukt til å definere og beskrive progresjon i personar sin kompetanse. Ein oppnår progresjon dersom ein opplever og kan vise utvikling innan minst ein av dimensjonane, og samstundes unngår at dette går utover ein annan dimensjon.

2.8 Strukturell og operasjonell forståing av matematikk

Sfard (1991) presenterer eit teoretisk rammeverk som kan brukast til å undersøke rolla til algoritmar i matematisk tenking. I rammeverket tek ho føre seg to ulike forståingar av matematiske konsept – *operasjonell* og *strukturell*. Ordet «konsept» blir brukt om ein matematisk idé i si «offisielle form» – som eit teoretisk omgrep innan «det formelle universet

av ideell kunnskap», med alle dei interne representasjonane og assosiasjonane som konseptet framkallar (Sfard, 1991, s. 3).

I fylgje Sfard (1991) handlar *operasjonell forståing* i matematikk om prosessar, algoritmar og handlingar, i staden for objekt. Ho peikar på at grundig innsikt i prosessane som ligg til grunn for matematiske konsept kan vere viktig for å forstå konseptet. Nokre gongar kan det til og med vere avgjerande for forståinga at ein fokuserer på prosessane i seg sjølv framfor å legge vekt på resultatet.

Strukturell forståing handlar om å behandle matematiske konsept som om dei refererer til eit abstrakt objekt, og dermed sjå at matematikken refererer til ekte ting (Sfard, 1991). Mentale bilete kan bidra til strukturell forståing av noko. For at ein skal kunne snakke om matematiske objekt, må ein kunne ta for seg produkta av visse prosessar utan å snakke om prosessane i seg sjølv. Visualisering gjer at abstrakte idear blir meir handfaste og lettare å behandle som om dei var materielle einingar (Sfard, 1991). Det å kunne sjå for seg dei usynlege objekta i matematikk viser seg å vere ein essensiell komponent i matematiske evner. Sfard (1991) hevder at mangel på denne ferdigheita kan vere ein av dei største grunnane til at matematikk blir sett på som uforståeleg for så mange «kloke hovud».

Strukturell tilnærming til matematikk blir gjerne sett på som eit avansert steg for å tileigne seg konsept. Likevel hevder Sfard (1991, s. 9) at ei strukturell tilnærming er heilt naudsynt, då det nesten er umogleg å få skikkelig innsikt i matematikken dersom ein ikkje har evna til å sjå abstrakte objekt. Sidan den strukturelle forståinga av matematikk kan vere vanskeleg å oppnå, formoder Sfard (1991) at operasjonell forståing er, for folk flest, det fyrste steget mot å tileigne seg ny matematisk kunnskap.

2.8.1 Modell for tileigning av matematiske konsept

I fylgje Sfard (1991) treng ein begge dei matematiske forståingane for å kunne få ei djup forståing av matematikk. Ho peikar på at stort sett alle matematiske konsept kan bli forstått både strukturelt og operasjonelt. Likevel hevder ho at vegen frå rekneoperasjoner til abstrakte objekt er lang og vanskeleg, og skjer i tre steg. Sfard (1991) presenterer desse tre stega i ein hierarkisk modell, noko som betyr at eit steg ikkje kan bli nådd før ein har fullført stega før.

1. Indregjering (interiorization)

I det fyrste steget, som kallast *indregjering,* blir eleven kjend med prosessar som byggjer opp eit nytt konsept. Dette er prosessar som kan gjennomførast på enkle matematiske objekt, og elevane blir gradvis betre på å utøve desse. Til slutt vil elevane kunne danne seg eit mentalt bilete av objektet, utan at dei treng å utføre desse prosessane.

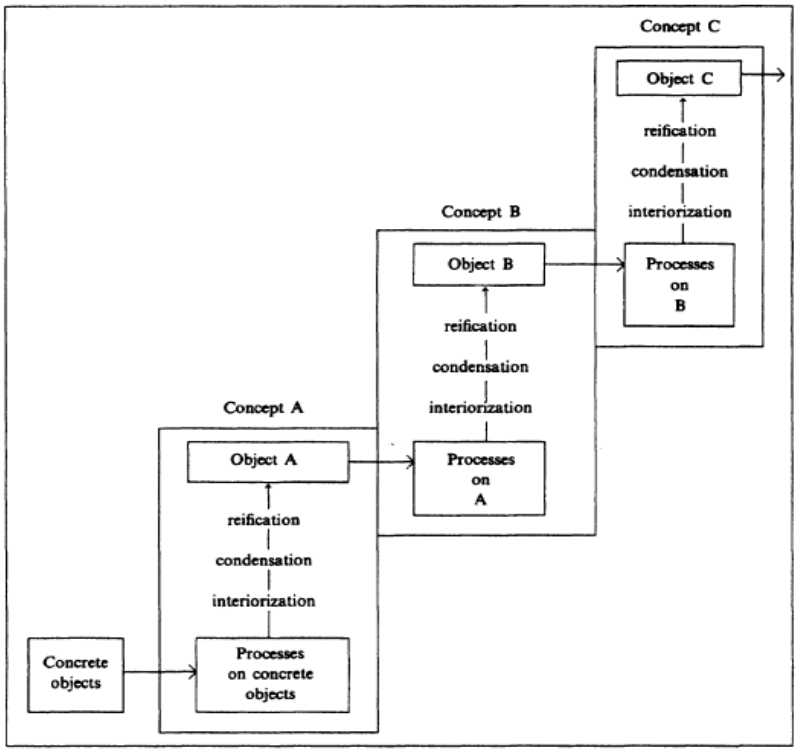
2. Samantrekning (condensation)

Vidare kjem ein til steget *samantrekning*, der ein jobbar med å trekkje saman lange operasjonar til mindre og meir handterlige einingar. På dette steget blir eleven stadig betre til å tenkje på prosessar som heilskapar, utan å kjenne behovet for å gå inn på detaljar. Samantrekningssteget fører til at ein kan kombinere prosessar, gjere samanlikningar og lettare generalisere. Dessutan blir det lettare å veksle mellom ulike representasjonar av eit konsept. Ein oppheld seg på dette steget så lenge ein har oppfatninga av at eit konsept er knytt til ein viss prosess.

3. Tingleggjering (reification)

Når ein person er i stand til oppfatte eit konsept som eit fullverdig objekt, kan ein seie at konseptet har blitt *tingleggjort*. Tingleggjering blir difor definert som eit ontologisk skifte, der ein plutseleg kan sjå noko kjent i eit heilt nytt lys. Ein kan seie at *indregjering* og *samantrekning* er gradvise kvantitative endringar, medan *tingleggjering* er eit momentant kvantesprang der ein prosess går over til å bli eit objekt. Sfard (1991) hevdar at det er på dette steget ein oppnår relasjonell forståing slik som beskrive av Skemp (1976); ein kjenner både til reglane og grunngjevingane. Sfard (1991) seier at når ein har oppnådd steg tre og eit konsept har blitt tingleggjort, kan konseptet ta del i nye prosessar. Nye matematiske objekt kan no bli konstruert ut frå det allereie tileigna konseptet. Dei tre stega startar dermed på nytt på eit høgare nivå, der ein byrjar på steg 1 - indregjering. Dette er illustrert i fylgjande modell:

Figur 3: Modell for tileigning av matematiske konsept (Sfard, 1991, s. 22)



Modellen har oppstått frå førestillinga om det operasjonelle opphavet til matematiske objekt. Teoretisk sett så er det mogleg å gjere nesten all matematikk reint operasjonelt. Ein kan gå frå grunnleggjande til meir avanserte prosessar utan å referere til abstrakte objekt, og i fylgje Sfard (1991, s. 23) er det nettopp dette som har blitt gjort i store delar av matematikken dersom ein ser tilbake i historia. Ein kan dermed spørje seg kvifor ein bør ha begge forståingane av matematikk. Sfard (1991) hevder at det kan vere vanskeleg å prosessere informasjon som ein har tileigna seg operasjonelt, då denne informasjonen blir lagra i ustrukturerte kognitive skjema. Dette er ein tungvindt måte å tileigne seg kunnskap på, sidan det fører til stor kognitiv belastning og ei kjensle av manglande forståing. Ei slik belastning vil naturligvis vere til hinder dersom ein skal løyse eit komplekst problem. Sfard (1991) meiner difor at det ofte kan vere best å starte med dei strukturelle sidene av eit konsept for å få oversikt. Til samanlikning vil det vere lurt av ein person som vil gå til ei spesiell gate å slå opp i eit kart før personen byrjar å gå. Ein kan seie at i problemløysingsprosessar vil abstrakte einingar fungere som vegvisarar til meir detaljert informasjon. Ein kvar matematisk problemløysingsprosess kan difor bli sett på som eit samspel mellom operasjonelle og strukturelle tilnærmingar, der begge tilnærmingane blir utnytta på best mogleg måte (Sfard, 1991, s. 28).

2.9 Matematisk kompetanse samanlikna med operasjonell og strukturell forståing

I artikkelen av Niss og Højgaard (2019) samanliknar dei kompetansar med prosedyrekunnskap, faktakunnskap og forståing. Vi skal gå nærmare inn på *kompetansar* sett opp mot *prosedyrekunnskap*, då prosedyrekunnskap ser ut til å ha fleire fellestrekk med Sfard (1991) si beskriving av *operasjonell forståing* av matematikk. I fylgje Niss og Højgaard (2019) er kompetanse eit meir komplekst omgrep enn prosedyrekunnskap. Medan prosedyrekunnskap handlar om å gjennomføre spesifikke, ofte algoritmiske og svarorienterte oppgåver, krev utøvinga av ein kompetanse at ein aktiverer mengdevis av ulike og spesifikke prosedyrekunnskapar. Niss og Højgaard (2019, s. 20) forklarar dette metaforisk ved å beskrive prosedyrekunnskap som «atomar», og kompetansar som svære «molekyl» bestående av ein stor mengde atomar i ulike posisjonar. Eigenskapane til eit molekyl vil ikkje vere utelukkande bestemt av verknader frå atoma, på same måte som at ein matematisk kompetanse ikkje berre er definert av prosedyrekunnskapane som den tar utgangspunkt i. Ein gjeven mengde av prosedyrekunnskap kan nemleg vere del av fleire ulike kompetansar. Oppsummeringsvis kan ein seie at prosedyrekunnskap er naudsynt, men ikkje tilstrekkeleg i utøvinga av ein matematisk kompetanse.

Beskrivinga av prosedyrekunnskap (Niss og Højgaard, 2019) inneheld fleire fellestrekk med Sfard (1991) si beskriving av *operasjonell forståing* av matematikk. Begge beskrivingane vektlegg prosessar, handlingar og gjennomføringar av oppgåver, der ein gjerne fylgjer ein algoritme. Sfard (1991) hevdar at ei grundig innsikt i prosessane kan vere avgjerande for å forstå gjevne konsept, medan Niss og Højgaard (2019) meiner at ein må aktivere prosedyrekunnskap for å utøve ein kompetanse. Likevel hevdar både Sfard (1991) og Niss og Højgaard (2019) at slike tilnærmingar er naudsynte, men ikkje tilstrekkelege for å utøve og forstå matematikk.

Niss og Højgaard (2019) samanliknar òg *matematiske kompetansar* med forståing. Dei hevdar at ein kan ha djup og vid forståing for til dømes eit matematisk konsept, resultat eller bevis, utan å ha matematisk kompetanse. Forståing kan nemleg bli lagra i minnet på ein passiv måte, medan kompetanse handlar om å utøve matematikk. Niss og Højgaard (2019, s. 21) brukar ein metafor til å beskrive dette òg; ein kan ha god og vid forståing for kjemiske stoffer, fenomen og teoriar, utan å vere i stand til å bruke utstyr og gjennomføre kjemiske eksperiment. På ein

annan side, krev solid kompetanse i matematikk at ein har tilstrekkeleg matematisk forståing (Niss & Højgaard, 2019, s. 21).

3 Metode

Denne studien har som hensikt å undersøke korleis elevar kan utvikle *strukturell* og *operasjonell forståing* av geometri gjennom utforsking i programmeringsspråket Python. I dette kapittelet blir det gjort greie for forskingsprosessen som har blitt gjennomført for å undersøke dette. Fyrst vil val av metode og førebuing til datainnsamling bli presentert. Deretter blir det forklart korleis datainnsamlinga har gått føre seg, samt korleis informantane har blitt valt ut. Vidare vil eg gjere greie for behandlinga av datamaterialet og korleis det har blitt analysert gjennom eit nytt samansett rammeverk. Til slutt vil eg ta føre meg studien sin reliabilitet og validitet, samt etiske omsyn.

3.1 Val av metode

Oppgåva baserar seg på kvalitativ forsking der seks elevar på 10. trinn er i fokus. I kvalitativ forsking arbeidar ein som regel i djupna med relativt få strategisk utvalde einingar (Tjora, 2021, s. 47). Eit fellestrekk for dei fleste kvalitative tilnærmingar er at datamaterialet forskaren analyserar, uttrykkast i form av tekst, der teksten kan beskrive personar sine handlingar, utsegner, intensjonar eller perspektiv (Thagaard, 2013, s. 14). Datamaterialet i studien er innhenta gjennom skjerm- og lydopptak, observasjonar og intervju. Postholm og Jacobsen (2018, s. 236) meiner at ein kombinasjon av fleire datainnsamlingsmetodar og datakjelder er ein måte å styrke pålitelegheit og gyldigheit på. Dei kallar ein slik kombinasjon for triangulering. Vidare beskriv dei triangulering som ei prosedyre som har som intensjon å skape eit meir utfyljande bilete av ein kompleks og samansett røynd gjennom beskrivingar frå ulike vinklar. Med informasjon frå ulike kjelder vil forskaren verte mindre sårbar for skeivheter som kan oppstå når ein baserar forskinga på berre éi kjelde (Postholm & Jacobsen, 2018, s. 237).

I denne studien blir triangulering i form av ulike metodar brukt for å skape eit godt utgangspunkt for å kunne svare på problemstillinga i oppgåva. Prosjektet starta med to introduksjonsøkter i Python for heile 10. trinn, der læraren styrte undervisinga og eg observerte. I desse øktene danna eg meg eit bilete av kva elevane fekk til og kva dei synast var utfordrande i Python. Dette vart tatt utgangspunkt for å tilpasse opplegget til hovudøkta i veka etterpå. Eg observerte òg i fyrste del av hovudøkta medan elevane løyste geometrioppgåver i Python. Observasjon i denne økta gav meg moglegheit til å tilpasse oppgåvene til elevane undervegs. I tillegg til observasjon vart det gjort skjerm- og lydopptak medan elevane løyste oppgåver. Etter at elevane hadde jobba med oppgåver i Python, deltok dei i eit semi-strukturert intervju som gav meg ein idé om kva

tankar dei hadde om programmering. Gjennomføringa av datainnsamlinga vil bli ytterlegare forklart i kapittel 3.2.

3.1.1 Førebuing til datainnsamling

I starten av prosjektet brukte eg mykje tid på å setje meg inn i programmeringsspråket Python gjennom nettsida Lær Kidsa Koding (https://www.kidsakoder.no/). Det å styre skilpadder ved hjelp av kodar fanga interessa mi og eg tenkte dette kunne vere interessant å sjå nærmare på. Undervegs i prosessen då eg skulle bli kjent med programmeringsspråket Python undra eg meg over korleis det er mogleg bruke programmet på eiga hand, utan at ein får presentert bestemte framgangsmåtar. Dette var med på å forme problemstillinga for oppgåva.

3.1.2 Samarbeid med elevane sin matematikklærar

Då det var bestemt kva oppgåva skulle ta føre seg, tipsa rettleiaren min meg om ein ungdomsskulelærar, som i denne oppgåva har fått tildelt det fiktive namnet Runar. Runar er ein lærar som var ideell å samarbeide med då han underviser både matematikk og valfaget programmering. I og med at eg ikkje har noko tidlegare erfaring med undervising av programmering, såg eg på dette som ein tryggheit. Eg kom tidleg i kontakt med han, og han var positiv til å delta i prosjektet. Vi hadde to møter i forkant av datainnsamling.

I det fyrste møtet snakka vi om kva elevane kunne om programmering frå før. Det var viktig for meg at vi gjennomførte prosjektet på ein måte som gjorde at både læraren og elevane skulle føle at det var nyttig og relevant for dei, og ikkje berre noko som vart gjennomført til fordel for mitt masterprosjekt. Dermed fann vi ut at vi kunne gjennomføre prosjektet i november når elevane skulle lære om geometri. Styring av skilpadde i programmeringsspråket Python vart dermed naturleg å flette inn i som ein del av undervisninga. I det andre møtet fastsette vi tid for prosjektet og blei einige om korleis det skulle gjennomførast.

3.2 Gjennomføring av datainnsamling

Datainnsamlinga er delt i to delar. Den fyrste delen består av to introduksjonsøkter som vil bli omtalt som *introduksjonsøkt 1* og *introduksjonsøkt 2*. Den andre delen inneber oppgåveløysing og intervju av seks elevar, og vil bli referert til som *hovudøkta*.

3.2.1 Introduksjonsøkter

Haraldsrud et al. (2020, s. 102) seier at mykje av hjernekapasiteten vil gå med på å skrive riktig syntaks når ein byrjar å programmere med tekst, noko som betyr at matematikken kan oppta ein mindre del av hjernekapasiteten til elevane. Ein bør difor bruke nokre timar på å få elevane opp å stå på syntaksen, seier dei. Formålet med introduksjonsøktene i denne studien var at elevane skulle bli kjend med det tekstbaserte programmeringsspråket Python. Ved å bruke eit par timar til introduksjon kunne elevane forhåpentlegvis opparbeide seg eit grunnlag til å kunne anvende Python til å løyse matematikkoppgåver i hovudøkta, utan at for mykje tid skulle gå med til å skrive riktig syntaks. Eg og Runar hadde på førehånd blitt einige om at han skulle styre introduksjonsøktene, medan eg var observatør. Vi fann ut at dette kom til å vere både meir tidsparande og effektivt sidan Runar kjenner klassa og har erfaring med programmering i klasserommet.

I utgangspunktet var planen at alle elevane skulle få éi introduksjonsøkt. Klassa er vanlegvis delt i to på tysdagar og tanken var å gjennomføre introduksjon til Python denne dagen. Då kunne elevane få best mogleg oppfølging med berre halvparten til stades i klasserommet, men på grunn av covid-19 vart det bestemt at 10. klasse skulle ha heimeskule denne veka. Ein konsekvens av dette var at introduksjonsøkta måtte gå føre seg digitalt, noko som førte til diverse utfordringar. Til dømes tok det lang til å sørgje for at alle elevane kom seg inn på nettsida og inn i programmet som skulle brukast, sidan vi ikkje kunne sjå skjermane til elevane. I tillegg vart det av ulike grunnar bestemt at introduksjonsøkta skulle gjennomførast i full klasse, noko som gjorde det enda vanskelegare å fylgje med på om alle elevane kom seg inn i programmet og om dei greidde å fylgje med då lærar Runar gjekk gjennom introduksjonen.

Nokre elevar hadde openbart meir erfaring med programmering enn andre, då dei vart tidleg ferdig med alle oppgåvene som var planlagde for timen. Likevel var det tydeleg at fleire synast programmering var vanskeleg, sidan det vart sagt ting som «eg skjønar ingenting» og «eg anar ikkje kva eg skal gjere». I slutten av timen måtte alle elevane levere inn eit dokument der dei hadde limt inn kodane dei brukte for å løyse oppgåvene. Då eg og Runar såg gjennom desse dokumenta etter timen var det klart at fleire av elevane hadde gjort svært lite, sannsynlegvis fordi dei synast det var vanskeleg. Runar foreslo difor at vi skulle ha enda ei introduksjonsøkt, slik at alle elevane kunne få litt «dreisen» på programmering. Det vart difor gjennomført enda ei introduksjonsøkt same veka.

Hovudfokuset i introduksjonsøktene var at elevane skulle få kunnskap om styring av skilpadder i Python. Eg og Runar var einige om at elevane skulle lære nokre få grunnleggjande, enkle koder og løkker som dei kunne ta utgangspunkt i for å løyse oppgåver som var retta mot pensum i geometri. Runar tok deretter utgangspunkt i ein Power Point framvising han hadde brukt tidlegare og gjorde nokre små justeringar i denne. Fyrste økt starta med at eg introduserte meg sjølv og prosjektet mitt. Eg forklarte at i veka etterpå ville eg forske nærmare på nokre elevpar. Det kom tydeleg fram at det var frivillig om ein ville delta i vidare forsking. Etter introduksjonen tok Runar over undervisninga. Han viste elevane korleis dei skulle kome seg inn i programmet og korleis dei fekk fram skilpadda som skulle styrast. Deretter viste han nokre enkle kodar og korleis desse kunne køyrast. Elevane kunne kopiere kodane frå eit dokument som han hadde lagt ut på Google Classroom. Deretter kunne elevane endre på kodane for å få eit nytt resultat (output). Det finst mange andre funksjonar i Python som elevane kunne ha fått introduksjon til, men dette vart ikkje prioritert då det ville ha vore veldig tidkrevjande. Prosjektet kunne ikkje ta for lang tid sidan programmering ikkje er ein del av læreplanmåla på 10. trinn dette skuleåret (2020-2021). I introduksjonsøkt 2 gjekk Runar gjennom noko av det same som i introduksjonsøkt 1, men i denne timen la han vekt på at elevane skulle lære det aller enklaste. Han forklarte at dei ikkje trengte å fokusere på løkkene enda, då kodane *forward* og *left* var det viktigaste til å starte med.

Under introduksjonsøktene var eg ikkje-deltakande observatør, noko som Postholm (2005, s. 154) beskriv som ein handling der observatøren er til stades utan å delta direkte i handlingsprosessane. Eg skreiv notatar til introduksjonsøktene, som vart tatt utgangspunkt i for å planleggje hovudøkta.

3.2.2 Hovudøkta

Veka etter introduksjonsøktene var elevane tilbake på skulen og hovudøkta kunne gjennomførast med fysisk oppmøte. Runar underviser klassa i matematikk og naturfag i til saman tre timar på torsdagar, og eg fekk ta ut tre elevpar desse timane. Eg hadde dermed éin time disponibelt til kvart par, som vart brukt til omtrent 40 minutt oppgåveløysing, etterfylgt av 20 minutt intervju. Eg vart tildelt eit eige klasserom som eg kunne bruke gjennom heile dagen. Då elevpara kom inn i klasserommet vart dei plasserte ved eit eige gruppebord, og eg oppheldt meg ved ein pult litt lenger borte. Eg plasserte oss på denne måten fyrt og fremst for å halde avstand med tanke på covid-19, men òg for at elevane skulle få arbeide utan å føle seg konstant overvaka.

I hovudøkta var eg ikkje-deltakande observatør medan elevane jobba med oppgåver i Python. Eg lytta til diskusjonen til elevpara under arbeidsprosessen. I tillegg gjekk eg til og frå gruppebordet deira slik at eg hadde oversikt over kva koder dei skreiv og kva resultat dei fekk i output. Eg skreiv notatar undervegs der eg blant anna skreiv kor lang tid elevpara brukte på dei ulike oppgåvene. Dette vart tatt utgangspunkt i for å bestemme om neste oppgåve skulle vere lettare eller vanskelegare enn den føregåande, med tanke på den resterande tida av dei planlagde 40 minutta. Løpet vart lagt opp slik at alle para fekk fullført siste oppgåva før dei gjekk vidare til intervju. Dette såg eg på som viktig, fordi det var ynskjeleg at alle para skulle kjenne på mestring og dermed få ei god oppleving med programmering.

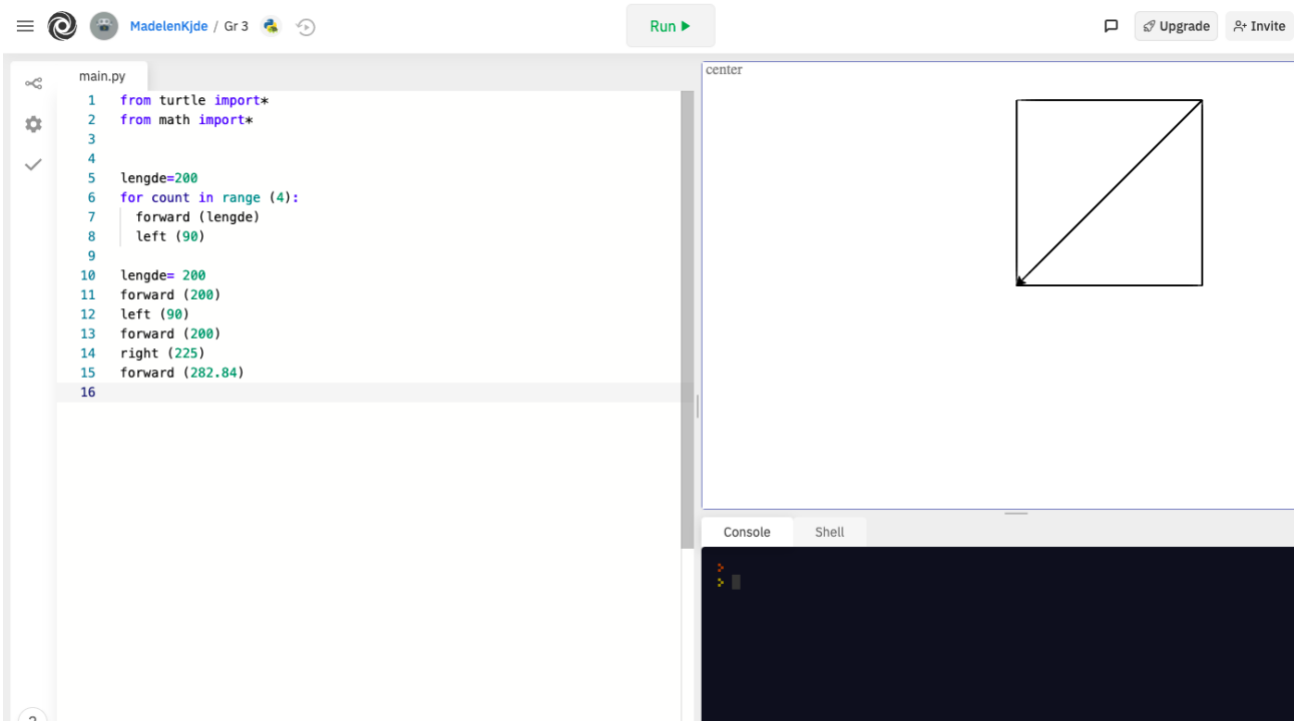
3.2.2.1 Skjerm- og lydopptak

For å få innsikt i elevane sitt arbeid med programmering i Python har det blitt gjort skjerm- og lydopptak medan elevane jobba i par. Det kunne ikkje bli lasta ned program for skjermopptak på elevane sine eigne datamaskiner, og elevane brukte difor min MacBook. Programmet QuickTime-Player er ein app på MacBook som vart brukt for å gjere opptak av all aktivitet på PC-skjermen. I dette programmet kan ein enkelt leggje inn ein «klikk-funksjon» som gjer at det kjem ring rundt datamusa kvar gong ein trykker på noko. Dette gjer det lettare å fylgje med på kva elevane gjer på PC-skjermen når ein skal sjå på skjermopptaka i ettertid.

3.2.2.2 Python med skilpadde

Når nybegynnarar skal lære seg å programmere i Python, kan skilpadde-biblioteket bidra til at ein får ei kjensle av korleis programmering i Pyton er, på ein morosam og interaktiv måte (Silaparasetty, u.å.). Skilpadde-biblioteket er ein funksjonar i Python som gjer det mogleg å lage bilete og figurar ved hjelp av ein form for robotar, som blir omtalt som skilpadder. Skilpadder er i hovudsak brukt til å introdusere Pyton til barn, men det kan òg vere nyttig for vaksne som prøver å bli kjent med programmeringsspråket (Silaparasetty, u.å.). I dette prosjektet har ein gratis nettstad (www.replit.com) blitt brukt når elevane skulle programmere med skilpadder i Python.

Figur 4: Utklipp av programmering med skilpadde i nettstaden www.replit.com



Figuren er eit utklipp frå programmeringsaktiviteten til eit av elevpara som deltok i studien. Python-skallet til venstre viser kodane elevane har skrive inn. Øvst kan ein sjå at det har blitt importert to Python-bibliotek: from turtle import* og from math import*. Slike bibliotek må importerast for at ein skal kunne bruke funksjonane som inngår i biblioteket. Oppe til høgre ser vi skjermen med resultatet (output) av kodinga. Den svarte pila (roboten) kallast skilpadda. Det svarte feltet nede til høgre viser eventuelle feilmeldingar i kodane som er skrive.

3.2.2.3 Oppgåvene til elevane

Med utgangspunkt i introduksjonsøktene og samtalar med Runar var det tydeleg at det var stor forskjell mellom elevane sine programmeringskunnskapar. Til hovudøkta førebudde eg difor ti opne programmeringsoppgåver (sjå Vedlegg I) av ulik vanskegrad, slik at alle elevpara kunne bli utfordra og samstundes oppleve meistring. Sikko og Grimeland (2020) seier at eit utforskande læringsmiljø bør vere prega av opne oppgåver som kan løysast på fleire måtar. Runar gjekk gjennom oppgåvene for å sjekke om vanskegrada var passeleg og om alle oppgåvene var forståelege. Både eg og Runar lagde kvar våre løysingsforslag til oppgåvene, der det kom tydeleg fram at oppgåvene kunne løysast på fleire måtar. Medan eg brukte mange enkle kodar, brukte Runar færre og meir effektive kodar. Dette fungerte dermed som ei stadfesting på at oppgåvene var opne nok til at elevane kunne utforske og løyse dei på ulike

måtar. Nokre oppgåver var òg opna opp slik at der var fleire moglege svar. Samstundes hadde oppgåvene eit klart mål, slik at elevane skulle ha noko konkret å jobbe mot. Programmering er heilt nytt for nokon av dei, og altfor opne oppgåver kunne ha ført til at dei ikkje visste kvar dei skulle starte.

Elevane fekk utdelt éi oppgåve i gongen, slik at det ikkje vart for mykje å fokusere på. Oppgåvene var nummererte, og kvar gong elevane fekk utdelt ei ny oppgåve sa eg oppgåvenummeret til dei for å skape klarhet i lyd-opptaka. Elevane fekk beskjed om at samarbeid var viktig og at dei skulle diskutere før dei sat i gong med å skrive kodar. Dei fekk beskjed om at dersom dei ikkje fekk ynskja resultat, kunne dei prøve på nytt eller endre litt på kodane. Dei fekk tilgang på papir, blyant og kalkulator, slik at dei hadde høve til å gjere utrekningar for hand. Dette var valfritt, då det òg er mogleg å gjere utrekningar i Python. Dei fekk i tillegg utdelt eit hjelpeark (sjå Vedlegg II) med nokre enkle kodar og løkker, slik at dei hadde noko å ta utgangspunkt i dersom dei ikkje visste korleis dei skulle setje i gong. Eit liknande ark lagde eg til meg sjølv då eg lærte å programmere i Python. Eg erfarte at det kunne vere vanskeleg hugse på alle kodane i hovudet og difor nyttig å ha noko å ta utgangspunkt i.

3.2.2.4 Semi-strukturert intervju

Etter at elevpara hadde arbeida med programmeringsoppgåver i omtrent 40 minutt, stilte dei til eit semi-strukturert intervju. Eg plasserte meg på den andre sida av gruppebordet dei satt ved, slik at vi såg mot kvarandre samstundes som vi heldt trygg avstand. Før eg starta intervjuet forklarte eg elevpara at eg ynskte at dei sa akkurat det dei tenkte på og at dei ikkje skulle bry seg om kva dei trudde eg ville høyre. Eg la vekt på at det ikkje fans eit riktig svar på spørsmåla.

Postholm og Jacobsen (2018, s. 121) beskriv eit semi-strukturert intervju som ein situasjon der det skapast kunnskap i møte mellom synspunkta til forskaren og forskingsdeltakarane. Forskaren har tema og forslag til spørsmål klare på forhand, men det er ikkje naudsynt at desse blir presenterte i ei gjeven rekkjefølgje. Spørsmåla stillast der det er naturleg å bringe dei inn i samtala, og forskaren er open for at forskingsdeltakarane kan introdusere tema som han/ho ikkje har tenkt på tidlegare. I dei semi-strukturerte intervjua i denne studien vart det stilt spørsmål som skulle gje innblikk i elevane sine forkunnskapar og tankar om programmering (sjå Vedlegg III). Det var det ynskjeleg å skape eit bilete av korleis elevane opplever programmering og nytta av programmering i skulen. I tillegg ville eg vete kva tankar dei hadde om matematikk og kva type matematikkundervisning dei trivast best med. Denne bakgrunnsinformasjonen har blitt

brukt til å beskrive elevutvalet. Dessutan har informasjonen fungert som eit støttande supplement under analyseprosessen, sidan den har vore med på å danne eit bilete av elevane si tilnærming til oppgåvene.

3.3 Utval

Elevpara som deltok i hovudøkta vart valt ut i samarbeid med lærar Runar. For å kunne få eit så rikt datamateriale som mogleg, fortalde eg Runar at eg ynska elevpar bestående av både jenter og gutar som samarbeider godt. I tillegg ville eg at elevane som skulle samarbeide var omtrent på same nivå både i matematikk og programmering, slik at dei hadde nokså likt grunnlag for å kunne bidra i programmeringsaktivitetane. Eg sa òg at det ville vere interessant å inkludere eit par som har hatt programmering som valfag. Runar tenkte seg ut tre elevpar, og etter introduksjonsøktene fortalde han meg kven han hadde sett seg ut. Dei tre elevpara var svært ulike både med tanke på prestasjonar i matematikk og erfaring med programmering. Dette såg eg på som ein fordel for å kunne få større innsikt, og dermed skape eit breiare grunnlag for å kunne svare på problemstillinga. Forskinga ville ha vore smalare dersom eg berre såg på ei gruppe presterande elevar. Dessutan ville eg ikkje ha visst om oppgåva sitt forskingsdesign hadde fungert for ulike elevgrupper. Utvalet i studien kan fungere som ei stadfesting på at oppgåva sitt forskingsdesign kan fungere for elevar som presterer på ulike nivå. I dei neste avsnitta vil utvalet bli presentert, og for å ivareta anonymiteten har elevane fått tildelt fiktive namn.

Elevpar 1 består av Oliver og Alex som er blitt omtalt som høgt presterande elevar i matematikk. Dei to gutane har programmering som valfag på ungdomskulen. I intervjuet kjem det fram at dei har vore innom programmeringsspråket Scratch på barneskulen og at dei har programmert litt utanfor skulen. Oliver og Alex forklarar at dei har samarbeida mykje tidlegare og at dei jobbar på ein slik måte at begge forstår kva dei gjer. Dei synast matematikk er gøy, spesielt dersom dei greier å løyse oppgåver dei har brukt lang tid på. Då dei skulle svare på spørsmålet om dei likar best å utforske og løyse problem eller å rekne reine oppgåver i matematikk, svarte begge to at dei likar best problemløysing. Likevel påpeikar Alex at sjølv om han synast det er keisamt å berre rekne og skrive, så er dette noko ein må kunne.

Elevpar 2 består av Maria og Silje som er opplyst som lavt/middels presterande elevar i matematikk. Jentene har samarbeida mykje tidlegare og sjølv om dei synast matematikk er

vanskeleg, ynskjer dei gjerne å få det til. I intervjuet fortel dei at den einaste gongen dei har prøvd programmering var i ein skuletime for fleire år sidan, og at dei ikkje hugsar så mykje av det. Silje fortel at under introduksjonsøktene, kom ho seg ikkje inn i programmet og ho var difor heilt «blank» då dei skulle løyse oppgåver i hovudøkta. Likevel peikar dei på at matematikken var meir utfordrande enn sjølve programmeringa, sidan det var Pytagoras' setning dei hadde mest problem med. Dei forklarar at etter å ha løyst den fyrste oppgåva, følte dei at dei hadde skjønt det som hadde med koding å gjere. Då jentene skulle svare på om dei likar best å utforske og løyse problem eller å rekne reine oppgåver i matematikk, svarte begge to at dei likar best å sitte å rekne fordi dei følar dei lærer meir av det.

Elevpar 3 består av Anette og Else som er opplyst som middels presterande i matematikk. I intervjuet kjem det fram at dei ikkje har noko erfaring med programmering og at dei difor var redde for at dei ikkje skulle greie nokon av oppgåvene dei vart tildelt. Til tross for dette greidde jentene likevel å løyse tre oppgåver. Dei sa dette var veldig gøy og at dei fekk skikkelig meistringskjensle. Dei forklarte at dei kunne gjette seg litt fram og at dei berre måtte ta eit steg av gongen. Dei la i tillegg vekt på at dei ikkje hadde greidd å løyse oppgåvene aleine, sidan dei kom med ulike forslag og idear til korleis dei skulle gå fram. Då dei skulle svare på om dei likar best å utforske og løyse problem eller å rekne reine oppgåver i matematikk, svarte begge to at dei likar best å sitte å rekne. Begge påpeikte at dei ikkje likar å sitte aleine å gruble, men at det er greitt dersom dei får samarbeide.

3.4 Behandling av datamaterialet

Av alt datamaterialet i studien er det skjerm- og lydopptaka av oppgåveløysingsprosessane til elevane som har blitt mest vektlagt. Intervjua og introduksjonsøktene har fungert som støttande og supplerande element. Introduksjonsøktene hadde ein støttande funksjon undervegs i forskingsprosessen ved at dei danna grunnlag for vidare forsking. Med grunnlag er det meint at eg fekk eit overblikk som gjorde det lettare å planlegge hovudøkta, og elevane fekk litt innsikt i Python som gav dei eit betre grunnlag til å løyse oppgåvene i hovudøkta. Dessutan kan introduksjonsøktene ha bidrege til at elevane gjorde mindre syntaktiske feil i hovudøkta. Dei semi-strukturerte intervjua har gjeve eit tydelegare bilete av elevane og synet deira på matematikk og programmering.

Etter å ha gjennomført hovudøkta med dei seks elevane tok eg til å transkribere det empiriske materialet. Befring (2015, s. 39) beskriv transkripsjon av lydopptak som ein prosess der ein overfører data til tekst før innhaldet blir analysert. Transkripsjonen vart gjort så tidleg som mogleg etter datainnsamling for at mesteparten av det som var blitt sett og høyrt skulle sitte friskt i minnet. Eg tok for meg ei og ei gruppe, der eg fyrst transkriberte oppgåveløysingprosessen og deretter intervjuet. Transkripsjonen vart lagra i to ulike word-dokument per gruppe for best mogleg oversikt. Under transkripsjonen av oppgåveløysingprosessen tok eg skjermbilete av det elevane gjorde i Python. Det vart tatt skjermbilete kvar gong elevane køyrte programmet og når dei hadde skrive noko i kodefeltet som var interessant. Skjermbileta vart lagt ved teksten fortløpande, slik at det i analyseprosessen skulle vere lett og oversiktleg å fylgje framgangsmåtane til elevane. Undervegs i transkripsjonen vart det skrive ned tidspunkt for kvar eg hadde kome i skjermopptaket slik at det i ettertid skulle bli lettare å navigere seg i skjermopptaket dersom det var noko som skulle dobbeltsjekkast.

I fylgje Kvale og Brinkmann (2015, s. 207) er transkripsjon frå lydopptak til tekst forbundet med ei rekkje tekniske og fortolkingsmessige problemstillingar, spesielt angåande ordrett talespråkstil versus skriftspråkstil. Dei hevdar at det ikkje finst standardreglar for desse problemstillingane, men heller ei rekkje val som må takast. Ein må til dømes velje om ein skal inkludere pausar, intonasjonsmessige understrekingar og kjensleutrykk som latter og sukk (Kvale & Brinkmann, 2015, s. 208). Alt dette, i tillegg til kommentarar, har eg valt å inkludere i mine transkripsjonar. Fylgjande utdrag frå ein av transkripsjonane viser korleis teikn har blitt brukt for å inkludere dette:

Maria: Nei [ler]. Åja, vent litt, vi er jo nøydde til å tenkje på kva for ein vinkel...
Silje: Her står ein annan ein då [på hjelpearket]...
Maria: Neimen kan ikkje vi berre sjå her, vent litt, vi må tenkje... Fordi at den har, eeh... okei det er 90 sant, i hjørna [peikar på figuren med datamusa]. Så det der er vi nøydde til å bytte, så *left*, det er nøydd til å vere 90, visst ikkje så får vi ikkje 90.

Eg har skrive tre punktum for å angje pausar eller for å vise at eleven dreg på orda. Det å inkludere pausar i transkripsjonen såg eg på som hensiktsmessig då det kan tyde på at elevane tenkjer eller er usikre på noko. Når elevane snakkar høgt eller brukar ivrig stemme har eg skrive ropeteikn bak utsegnene. Komma blir brukt både der det vanlegvis er naturleg, og stader der

elevane snakkar samanhengande i ufullstendige setningar. Intonasjonsmessige understrekingar har eg tydeleggjort ved å bruke store bokstavar. Det er brukt hakeparentesar for å beskrive kva elevane gjer eller for å forklare situasjonane nærmare. Til slutt har eg brukt *kursiv* når det er snakk om kodar. Alle samtalane har blitt omsett frå dialekt til nynorsk i transkripsjonen.

3.5 Analyseverktøy

Etter å ha transkribert og studert datamaterialet tok eg til å analysere. Det fyrste som vart gjort var å ta utgangspunkt i det transkriberte datamaterialet for å få oversikt over kva elevane gjorde på PC-skjermen og kva dei sa undervegs i arbeidsprosessen. Det vart valt ut ein arbeidsprosess frå kvar av dei tre elevpara, der elevane såg ut til å anstrenge seg for å løyse ei oppgåve. Arbeidsprosessane der oppgåvene vart rutineoppgåver for elevane, altså at dei vart løyst utan problem, har blitt valt vekk. Det er truleg situasjonane der elevane opplev oppgåvene som eit problem som kan bidra til utvikling av *operasjonell* og *strukturell forståing* hos elevane, sidan dei måtte gå utover sin allereie eksisterande kunnskap.

3.5.1 Samansett rammeverk

For å kunne belyse elevane sin *strukturelle* og *operasjonelle forståing* av geometri har det blitt laga eit nytt samansett rammeverk som består av fire omgrep som er beskrive i teorikapittelet: *problembehandlingskompetanse*, *tankegangskompetanse*, *algoritmisk tenking* og *utforsking*. Desse teoretiske omgrepa vil heretter vil bli omtalt som *komponentar*. Dei fire komponentane er valt ut fordi dei inneheld element som har noko til felles med Sfard (1991) si beskriving av strukturell og/eller operasjonell forståing. Det samansette rammeverket kan difor brukast til å belyse dei to forståingane. Dessutan kan rammeverket vere med på å beskrive elevane si tilnærming til programmeringsoppgåvene, sidan alle komponentane på ein eller anna måte handlar om utøvinga av matematikk. I denne oppgåva blir strukturell og operasjonell forståing tolka som tilstandar som kan utviklast, medan dei fire komponentane i rammeverket blir sett på som handlingar som potensielt kan vere med på å utvikle forståingane. Rammeverket gjer det dermed mogleg å belyse oppgåva si problemstilling frå ulike vinklar.

I arbeidet med å skaffe oversikt over fellestrekka mellom dei fire komponentane og dei to forståingane har dei blitt beskrive med stikkord og plassert i tabellen som er vist under (Tabell 1). Vidare har stikkorda blitt samanlikna og fått tildelt fargar. Fargane representerer mi tolking av fellestrekka mellom dei fire komponentane og dei to forståingane. Til dømes har stikkorda

som kan knytast til *operasjonell forståing* fått tildelt fargen raud. Av tabellen kan ein sjå at raudfargen er mest dominerande under komponentane *problembehandlingskompetanse* og *algoritmisk tenking*, noko som indikerer at desse komponentane har fleire element som kan knytast til operasjonell forståing. Stikkorda som har fått tildelt fargen grøn kan knytast til *strukturell forståing*. Av tabellen kan ein sjå at komponent 1, 3 og 4 inneheld stikkord som har fått tildelt fargen grøn, noko som indikerer at desse kan knytast til strukturell forståing. Fargane raud og grøn viser i tillegg at dei fire komponentane har fellestrekk med kvarandre.

Stikkorda som har fått tildelt andre fargar enn raud og grøn knytast ikkje direkte til *strukturell* og *operasjonell forståing*, men viser at dei fire komponentane har fleire fellestrekk med kvarandre. Sjølv om stikkorda med andre fargar ikkje kan koplast direkte til forståingane, kan dei likevel vere med på å gje ei grundigare beskriving av elevane si tilnærming til programmeringsoppgåvene.

Tabell 1: Fargekodar

Komponent 1: Tankegangskompetanse	**Komponent 2 - Problembehandlingskompetanse**
- Undersøkande matematikk - Matematiske spørsmål - Ta stilling til varierande omfang av matematiske konsept - Abstraksjonar - Generalisering	- Handtere matematiske problem - Identifisere, stille opp og løyse problem - Tenke ut strategiar - Meir enn prosedyrar og rutinar - Vurdere gyldighet

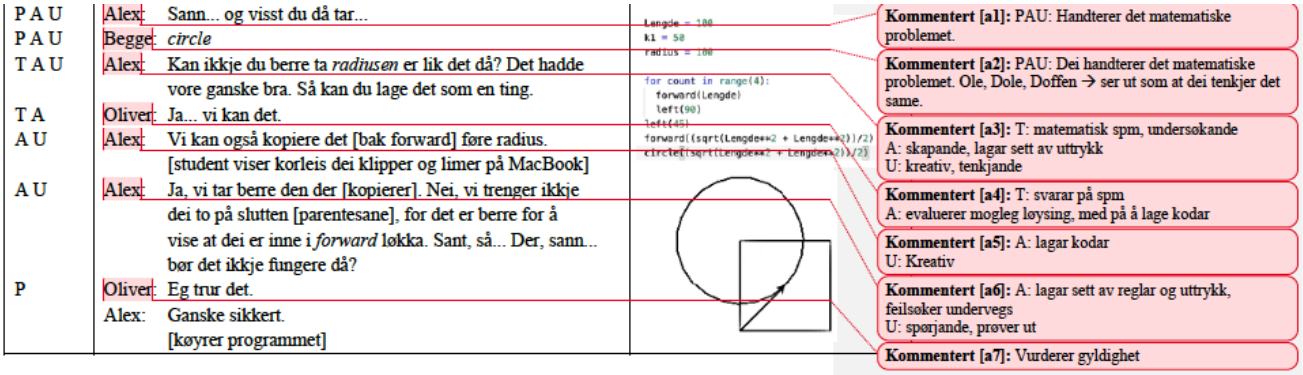
Komponent 3: Algoritmisk tenking	**Komponent 4 - Utforsking**
- Vurdere kva steg som trengst for å løyse eit problem. - Algoritmar (reglar og steg-for-steg) - Skapande, eksperimenterande og open for alternative løysingar. - Å gjere feil er ein viktig del av prosessen. - Fikle, samarbeide, feilsøke, halde ut. - Abstraksjon (fjerne unødige detaljar) - Mønstre (finne og bruke likskapar) - Evaluering - Logikk (analysere og føresjå) I programmering: - Lage sett av reglar og uttrykk - Identifisere problem og utforme løysingar - Lage kodar - Feilsøke og forbetre koda - Kjenne at mønster, eksperimentere og evaluere moglege løysingar. - samarbeid	- Leite etter mønster/finne samanhengar. - Diskutere seg fram til ei felles forståing - Undersøkande arbeidsmetode, oppdagingsorientert. - Spørjande haldning - Elevane skal utvikle seg til å bli tenkjande og kreative. Prøve seg fram. - Viktig med faglege/kognitive utfordringar for å få betre læringsresultat - Spørje, prøve ut, undersøke, undre, identifisere problem og søke løysingar, og ha eit kritisk blikk på det ein utforskar. - Samarbeid er ein viktig del av utforskingsprosessen.

Operasjonell forståing	**Modell i tre steg**	**Strukturell forståing:**
- Prosess, algoritmar, handlingar	1. Indregjering (interiorization) 2. samantrekning (condensation) 3. tingliggjering (reification)	- Behandle konsept som om dei refererer til eit abstrakt objekt - Visualisering, sjå «usynlege» objekt i matematikk - Ta for seg produkt av visse prosessar utan å snakke om prosessane i seg sjølv.

3.5.2 Koding

For å kunne studere arbeidsprosessane til elevane med utgangpunkt i det samansette rammeverket har transkripsjonen av samtalane blitt koda. I fylgje Thagaard (2013, s. 158) handlar koding om å beskrive utsnitt av eit datamateriale med omgrep som gjev uttrykk for meiningsinnhaldet i teksten. Dei utvalde elevsamtalane i denne studien har blitt koda ut frå dei fire komponentane i rammeverket, som har fått tildelt kode etter fyrste bokstav: *Tankegangskompetanse* [T], *Problembehandlingskompetanse* [P], *Algoritmisk tenking* [A] og *Utforsking* [U]. Sidan komponentane er ganske omfattande, har det undervegs i kodinga blitt lagt inn stikkord i margen som utdjupar kvifor ei utsegn har fått tildelt ei kode. Transkripsjonane har blitt plassert inn ein tabell med tre kolonnar, der kodane står i kolonnen til venstre, dialogen er i midtarste kolonne, og utklipp frå Python havnar i kolonnen til høgre. Dette er illustrert i fylgjande utklipp frå datamaterialet:

Figur 5: Utklipp frå koding av datamaterialet



Som vist i Tabell 1, har komponentane fleire fellestrekk med kvarandre. Til dømes høyrer det å «identifisere eit problem» til under tre komponentar: *problembehandlingskompetans*e [P], *algoritmisk tenking* [A] og *utforsking* [U]. Dette betyr at utsegner i dialogane kan få tildelt fleire kodar på same grunnlag. For at analyseteksten ikkje skal bli for repetitiv, vil slike element som det å «identifisere eit problem» bli omtalt i samband med berre ein av komponentane. Målet med det samansette rammeverket og kodinga er ikkje fyrst og fremst å klassifisere elevane sine utsegner. Målet er at rammeverket i sin heilskap kan vere med på å gje ei rik beskriving av arbeidsprosessane til elevane. Vidare kan denne beskrivinga blir tatt utgangspunkt i for å belyse deira *operasjonelle* og *strukturelle forståing* for geometri.

3.5.3 Ei hermeneutisk tilnærming

Analysen i denne studien vil bere preg av ei hermeneutisk tilnærming til datamaterialet. Hermeneutikken framhevar betydinga av å fortolke folk sine handlingar gjennom å utforske eit djupare meiningsinnhald enn det som er direkte innlysande (Thagaard, 2013, s. 41). I fylgje Kvale og Brinkmann (2009, s. 214) går fortolkaren utover det som direkte blir sagt ved å finne fram til meiningsstrukturar og betydingsrelasjonar som ikkje er direkte synleg i ein tekst.

Kvale og Brinkmann (2009, s. 216) beskriv eit av fortolkingsprinsippa innan hermeneutikk som ein kontinuerleg fram- og tilbake prosess mellom delar og heilskap, som ein fylgje av *den hermeneutiske sirkel*. Med utgangspunkt i ei ofte uklar og intuitiv forståing av teksten som heilskap fortolkast dei ulike delane, og ut i frå desse fortolkingane settast delane inn i ny relasjon til heilskapen. I denne studien vil eg som forskar ha ei hermeneutisk tilnærming til datamaterialet, der eg tek i bruk det nye samansette rammeverket for å finne fram til meiningsstrukturar i transkripsjonen. Ein kan sjå på fortolkinga av datamaterialet som ein hermeneutisk sirkel der elevane sine utsegner er delane, og deira utvikling av forståing er heilskapen. Dette vil vere ein fram- og tilbakeprosess der det vil kontinuerlig bli vurdert om elevane faktisk utviklar forståing, eller om dei viser forståing dei har frå før.

3.6 Validitet

Kvale og Brinkmann (2015, s. 279) seier at validitet handlar om i kva grad ein studie undersøkar det den er meint til å undersøke. Dette inneber å kvalitetssjekke dei ulike stega av ein forskingsprosess og undersøke feilkjeldene. Forskaren har eit kritisk syn på eigne fortolkingar og uttrykkjer eksplisitt sitt perspektiv på emnet som studerast. Ein mogleg feilkjelde i denne studien er at Silje og Maria i gruppe 2 hadde ganske like stemmer, noko som gjorde transkripsjonen utfordrande. Det er dermed mogleg at nokre av utsegnene i transkripsjonen har blitt tildelt feil person. Sidan studien tar for seg elevane si utvikling av forståing, kan denne potensielle feilkjelda ha ført til at det har blitt tillagt forståing til feil elev. Til tross for dette vart det brukt lang tid på å høyre gjennom lydklippa til jentene fleire gongar. Dessutan vart transkripsjonen gjort tidleg etter datainnsamling, noko som stort sett gjorde det mogleg å kjenne att dei små forskjellane som var i stemmene deira.

Studien tar for seg elevar si utvikling av forståing for geometri når dei programmerer i Python, og det er oppgåveløysinga i hovudøkta som blir mest vektlagt. Hovudøkta skal basere seg på at

elevane har hatt introduksjonsøkter om Python. I intervjuet viste det seg at Silje ikkje hadde kome seg inn i programmeringsprogrammet under introduksjonsøktene og ho hadde difor ikkje særleg grunnlag for å setje i gong med oppgåvene i hovudøkta. Dette kan sjåast på som ei feilkjelde i studien, då eit av stega i forskingsprosessen ikkje har gått som planlagt.

For at studien skal kunne undersøke det den er meint til å undersøke må elevane få utdelt oppgåver som stiller kognitive krav til dei slik at dei kan utvikle forståing. I utviklinga av oppgåvene var det forsøkt å lage oppgåver av ulik vaskegrad, slik at elevane kunne bli utfordra på det nivået dei ligg på. Oppgåvene (Vedlegg I) er opne slik at elevane sjølve må finne ut kva framgangsmåte dei vil bruke. Samstundes har oppgåvene eit klart mål, slik at elevane veit kva dei skal jobbe mot. I nokre tilfelle verkar det likevel som at oppgåvene var rutineoppgåver for elevane (spesielt elevpar 1), då elevane tilsynelatande løyste oppgåvene utan problem. Det kunne dermed sjå ut som at elevane brukte forståinga dei hadde frå før og at dei ikkje utvikla ei ny forståing. Arbeidsprosessane knytt til desse oppgåvene har difor ikkje blitt vektlagt i studien, då dei ikkje egnar seg til å svare på problemstillinga.

3.7 Reliabilitet

I fylgje Postholm og Jacobsen (2018, s. 222) handlar reliabilitet om korleis gjennomføringa av forskinga kan ha påverka dei endelege resultata, og kor mykje ein kan stole på funna som forskingsprosjektet har produsert. For å styrke reliabiliteten i oppgåva har det blitt gjort grundige beskrivingar av forarbeidet til datainnsamlinga, kva som vart gjort undervegs og korleis datamaterialet vart behandla etterpå. Dette har blitt gjort for å oppretthalde ein transparens i oppgåva, der lesaren heile vegen blir forklart korleis studien har gått føre seg.

Reliabilitet behandlast ofte i samband med spørsmålet om i kva grad eit resultat kan reproduserast på andre tidspunkt av andre forskarar (Kvale & Brinkmann, 2009, s. 250). Sidan studien baserar seg på ei kvalitativ og hermeneutisk tilnærming, vil forskaren spele ei betydeleg rolle både i datainnsamlinga og når datamaterialet skal transkriberast og fortolkast. For å styrke reliabiliteten er det forsøkt å tydeleggjere korleis datainnsamlinga har føregått og korleis det samansette rammeverket i oppgåva er utarbeida og tatt i bruk for å beskrive datamaterialet. Vidare er det forklart korleis desse beskrivingane skal brukast for å svare på oppgåva si problemstilling. Eg vil dermed påstå at gjennomføringa av forskinga er nøye beskrive i metodekapittelet. Dette betyr i teorien at det skal vere mogleg for ein annan forskar å

gjennomføre den same studien og reprodusere resultata på eit anna tidspunkt. Likevel er det ikkje så lett i praksis. Fyrst og fremst vil datamaterialet som blir samla inn variere med utgangspunkt i kva for nokre elevar som deltek i undersøkinga. Dette kjem tydeleg fram i denne studien, der dei tre elevpara har ulike utgangspunkt og tilnærmingar når dei skal løyse oppgåvene dei får tildelt. Sjølv om ein hypotetisk sett hadde enda opp med det same datamaterialet og brukt det same rammeverket, vil to ulike forskarar truleg tolke materialet på ulike måtar og dermed få ulike resultat.

3.8 Etiske omsyn

I forsking har informantane krav på at absolutt alle opplysningar om personlege forhold blir behandla konfidensielt og at innsamla forskingsdata til vanlig skal vere anonymisert. Likevel vil det i kvalitative studiar ofte vere problemtisk å ivareta anonymiteten til informantane, då enkeltpersonar ofte vil stå fram på ein direkte og synleg måte (Befring, 2015, s. 32-33). I forkant av datainnsamling vart prosjektet godkjent av Norsk Senter for Forskningsdata (NSD) og rektor ved forskingsskulen. For å ivareta anonymiteten til informantane i studien, har dei blitt tildelt fiktive namn frå starten av. Med få deltakarar har det ikkje vore naudsynt å lage namnelister, då namna har blitt hugsa på. Det førekjem heller ikkje informasjon i eigne notatar, lydopptak eller skjermopptak som gjer det mogleg å identifisere elevane eller læraren.

Eit grunnleggjande prinsipp i forskingsetikken er at all deltaking skal bygge på eit samtykke som er gjeve på eit fritt, informert og forstått grunnlag (Befring, 2015, s. 31). Etter at NSD og rektor ved prosjektskulen hadde gjeve meg tillating til å gjennomføre prosjektet, informerte Runar om prosjektet til både elevar og føresette. I fyrste innføringstime fortalde eg om prosjektet og gav informasjon om kva ei eventuell deltaking i hovudøkta ville innebere. Alle elevar og føresette fekk deretter utdelt eit samtykkeskjema med utfyljande informasjon om prosjektet. Samtykkeskjema innehaldt kontaktinformasjon og oppfordring om å stille spørsmål dersom dei lurte på noko. Då verken eg eller Runar fekk spørsmål angåande prosjektet, kan ein anta at informasjonen var tydeleg og forstått. Ettersom alle dei seks elevane som deltok i hovudøkta hadde fylt 15 år, kunne dei samtykke på eiga hand. Dei samtykka skriftleg i form av underskrift på samtykkeskjema før dei møtte meg.

Rektor var positiv til prosjektet, samstundes som det vart stilt krav til at vi fylgde retningslinjene for smittevern med tanke på covid-19, sidan datainnsamlinga vart gjennomført på eit tidspunkt

med aukande smittetal. Det vart dermed gjort fleire tiltak for å hindre smitte. I tida før datainnsamling hadde eg minimal sosial kontakt og tok lite kollektiv transport. På sjølve dagen for datainnsamling fekk eg tildelt eit eige klasserom der eg skulle opphalde meg gjennom heile dagen. Med det same elevpara kom inn i klasserommet sprita dei hendene. Eg hadde sprita PC-en og anna utstyr dei skulle bruke, og utstyret vart sprita på nytt kvar gong eit elevpar var ferdig. For å halde trygg avstand sat elevane ved eit gruppebord og jobba, medan eg satt ved eit anna bord som var minst to meter frå dei. Eg haldt avstand då eg gjekk til og frå elevane for å sjå kva dei jobba med på PC-skjermen. Dersom elevane trengte hjelp til noko, måtte eg nærme meg for å sjå tydeleg på PC-skjermen kva dei trengte hjelp til, men deretter trakk eg meg vekk. Under intervjuet satt eg på andre enden av gruppebordet elevpara satt ved, og hadde minst to meter avstand. På slutten av dagen vart alle overflater sprita før eg forlét klasserommet.

4 Analyse

I dette kapittelet blir samtaleutdrag frå tre elevpar analysert og tolka for å danne grunnlag for å kunne svare på problemstillinga:

Korleis kan elevar utvikle strukturell og operasjonell forståing av geometri gjennom utforsking i programmeringsspråket Python?

Kapittelet er delt inn i tre delkapittel (4.1-4.3). Kvart delkapittel fokuserer på samtala til eit elevpar, samt utklipp frå aktiviteten i Python medan dei arbeidar med ei oppgåve frå oppgåvearket (sjå Vedlegg I). Samtalane til elevpara har blitt koda, slik som beskrive i metodekapittelet, med utgangspunkt i det samansette rammeverket som består av fylgjande komponentar: [T] *Tankegangskompetanse*, [P] *Problembehandlingskompetanse*, [A] *Algoritmisk tenking* og [U] *Utforsking*. Kodinga har bidrege til å skape oversikt i elevsamtalene og dermed gjort det lettare å velje innhaldsrike samtaleutdrag til vidare analyse.

I kvart delkapittel blir nokre samtaleutdrag presentert og teke utgangspunkt i for å beskrive elevparet sitt arbeid medan dei programmerer. Beskrivinga vil bestå av eigne refleksjonar av det som skjer, samt tolkingar som baserer seg på komponentane som har blitt identifiserte gjennom koding. Dei fire komponentane vil bli presentert i den rekkefylgja som dei er nemnt i avsnittet over: [T], [P], [A], [U]. I dei ulike samtaleutdraga er nokre komponentar meir synlege og dominerande enn andre, noko som vil kome tydeleg fram av teksten.

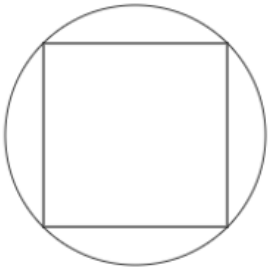
På slutten av kvart delkapittel blir beskrivinga som baserar seg på dei fire komponentane brukt til å seie noko om elevane sin *strukturelle* og *operasjonelle forståing* av dei geometriske konsepta dei arbeidar med. Det blir undersøkt om elevane viser teikn til operasjonell og strukturell forståing i samtaleutdraga, og om det er mogleg å identifisere utvikling i desse forståingane. Her vil fargane som er beskrive i samband med det samansette rammeverket (3.5.1) fungere som eit hjelpemiddel for å tydelegare kunne sjå samanhengane mellom dei fire komponentane og dei to forståingane. Fargane vil ikkje bli uttrykt eksplisitt i analysen, men fungere som ein stønad.

4.1 Alex og Oliver programmerer eit kvadrat innskrive i ein sirkel

I dei fylgjande utdraga skal elevparet Alex og Oliver programmere eit kvadrat innskrive i ein sirkel. Dette er den femte oppgåva dei løyser i denne økta. Oppgåva dei får tildelt kan løysast på mange måtar og gutane finn fort sin eigen metode. Dei bygg vidare på noko av det dei har gjort i tidlegare oppgåver, der dei blant anna brukte Pytagoras' setning i ei oppgåve og programmerte ein sirkel i ei anna oppgåve. Oppgåva gutane får utdelt ser slik ut:

Figur 6: Oppgåve 7 frå oppgåveark

7. a) Diskuter korleis ein kan gå fram for å programmere figuren under.
b) Utforsk framgangsmåten de kom fram til.



I fylgjande utdrag startar gutane med oppgåve 7 a), der dei diskuterer korleis dei kan gå fram for å programmere figuren. Dei har mange småpausar i starten av samtala, noko som gjev rom for at begge kan kome med forslag:

4.1.1 Utdrag 1

1) Alex: Tingen er, eg vil anta at man kan lage eit kvadrat fyrst. Nei nei vent, korleis...
2) Oliver: Jo jo, vi kan sikkert det, ehm, vent då...
3) Alex: Så gå i midten...
   [Stillhet]
4) Oliver: Æh, den var vanskelig...
5) Alex: Nei, men vist vi tenkjer på diagonalen sant, du kan jo sjå at, her har du... diameteren er jo...
6) Begge: Diagonalen.
7) Alex: Sant, så difor, visst vi berre... vi byrjar med eit kvadrat, så visst vi tar berre *forward*... Kva skal vi byrje med, *100*?

Alle dei fire komponentane har gjennom kodinga blitt identifiserte i dialogutsnittet. *Tankegangskompetansen* kjem til syne hos begge elevane i sitat 5 og 6 då dei påpeikar at diagonalen i kvadratet vil vere det same som diameteren i sirkelen. Ved å påpeike denne samanhengen viser gutane teikn til at dei abstraherer, sidan dei trekkjer fram viktige eigenskapar i figurane som ikkje er direkte synlege. Det er tilsynelatande Alex som ser samanhengen fyrst, då han byrjar å forklare korleis han tenkjer i sitat 5. Likevel ser det ut som at Oliver blir tidleg merksam på tankegongen til Alex, sidan dei i sitat 6 fullfører Alex si setning saman.

Komponenten *problembehandlingskompetanse* har blitt identifisert fleire stader i dialogutsnittet. Gutane er tidleg ute med å identifisere samanhengen mellom diagonal og diameter, og dei har dermed identifisert sentrale element for å løyse det matematiske problemet. I utsegn 7, legg Alex fram eit forslag for korleis dei kan setje i gong med å løyse oppgåva. Han byggjer på informasjonen om at diagonalen vil vere det same som diameteren, og føreslår at dei startar med eit kvadrat. Han legg ikkje fram ein fullstendig strategi for korleis oppgåva skal løysast, men eit forslag til korleis dei kan byrje. Han handterer det matematiske problemet ved å setje i gong med oppgåva, med utgangspunkt i samanhengane dei har identifisert.

Komponenten *algoritmisk tenking* viser seg òg fleire stader i utdraget. Til dømes viser Alex og Oliver algoritmisk tankegang når dei brukar likskapane mellom kvadratet og sirkelen til å abstrahere og finne eit mønster i figuren. Vidare er elevane skapande, ved at dei tek denne informasjonen i bruk når dei vurderer kva steg som skal til for å løyse oppgåva. Til slutt kjem den algoritmiske tangegongen til uttrykk når elevane set i gong med å skrive kodar i Python.

Måten gutane uttrykkjer seg på tyder på at dei er undrande og har ei utforskande tilnærming til oppgåva. Komponenten *utforsking* syner seg når gutane prøver seg fram ved at dei byrjar å lage eit kvadrat utan å vere sikre på kva neste steg vil vere. Dei viser i tillegg at dei er utforskande ved at dei er spørjande og samarbeidande, då dei gjev rom for at begge kan kome med innspel.

I neste utdrag er gutane ferdige med å programmere eit kvadrat. Dei har oppretta variabelen *Lengde* for å lage kvadratet, og tar no utgangspunkt i denne for å lage ein diagonal:

4.1.2 Utdrag 2

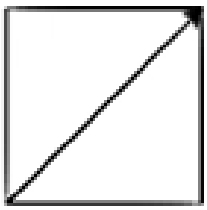
8) Oliver: *Forward*, eh, kva blir det då... *Lengde* i andre... [skriv]
9) Begge: Pluss.
10) Alex: Så må du ta *square root* [*sqrt*] av den.
11) Oliver: Ja, *Lengde*... to... ja *Lengde.*
12) Alex: [ler] du skriv feil, du skreiv «Lenge»
13) Oliver: Åja [ler]. Lenge... Ehm...
14) Alex: Så må du ta det inn i ein annan parentes.
[Oliver skriv]
15) Alex: Sann, så køyrer du.
[køyrer programmet]

Figur 7: Utklipp av kodar

```
Lengde = 100
k1 = 50
radius = 100

for count in range(4):
  forward(Lengde)
  left(90)
left(45)
forward(sqrt(Lengde**2 + Lengde**2))
```

Figur 8: Utklipp av resultat (output)



I dette utdraget er det berre komponenten *algoritmisk tenking* som er identifisert, noko som er grunnen til at utdraget blir løfta fram. Ein mogleg grunn til at dei andre komponentane ikkje er identifiserte, er at gutane tilsynelatande har rutinemessige tilnærmingar til oppgåva og at dei ikkje ser på den som eit «problem» i dette utdraget. Likevel gjev gutane utrykk for at dei synast oppgåva er vanskeleg i utdrag 1 (kapittel 4.1.1), og det kan difor tenkjast at oppgåva i sin heilskap ikkje er rein rutine for gutane.

Komponenten *algoritmisk tankegong* er identifisert gjennom heile utdraget. Med tanke på transkripsjonane av gutane sine samtalar, er dette utdraget eitt av fleire døme som viser teikn på at gutane har programmert tidlegare. Kodinga skjer omtrent automatisk, der Oliver skriv og Alex bidrar med innspel. Utdraget viser at gutane kjenner godt til algoritmen bak Pytagoras' setning og at dei ser fort korleis dei skal skrive dette ut med kodar. Dei nemner ikkje Pytagoras' setning eksplisitt og dei diskuterer heller ikkje kva denne setninga inneber før dei skriv kodane. Dette kan vere eit teikn på at dei har kontroll på algoritmen, og at dette difor går omtrent på

automatikk. Det kan òg vere eit resultat av at gutane har brukt Pytagoras' setning i oppgåvene før denne, og at dei difor ikkje nemner setninga i denne samanhengen.

Ein viktig eigenskap hos *den algoritmiske tenkaren* er å kunne samarbeide med andre. I dette utdraget viser gutane evne til å samarbeide sidan begge tar aktivt del i å skrive kodane. Medan Oliver skriv, les Alex gjennom kodelinja og rettar på syntaksen. Dette gjer truleg at dei sparar tid då dei unngår å få feilmelding når dei køyrer koda. Alex kjem i tillegg med forslag til kodar, som til dømes i utsegn 10 når han seier: «Så må du ta *square root* av den». Begge gutane viser at dei meistrar å lage sett av uttrykk, ved at dei bidreg med forslag til sirka annakvar kode i kodelinja.

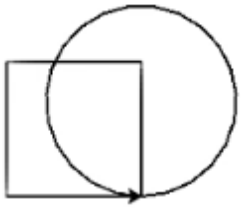
I samtala som fortset etter utdraget over, blir Alex og Oliver einige om å halvere diagonalen slik at pila havnar i sentrum av kvadratet og bli startpunktet for sirkelen. Vidare brukar dei same formel for radius som dei brukte for å lage halve diagonalen. Dei køyrer programmet og oppdagar at resultatet blir feil, då sirkelen blir plassert feil i forhold til kvadratet. Dei vel å behalde storleiken på sirkelen, men blir einige om at den heller må starte nede i høgre hjørne av kvadratet. Dei endrar noko på kodane, køyrer programmet på nytt og får fylgjande resultat:

Figur 9: Utklipp av kodar

```
Lengde = 100
k1 = 50
radius = 100

for count in range(4):
  forward(Lengde)
  left(90)
forward(Lengde)
circle((sqrt(Lengde**2 + Lengde**2))/2)
```

Figur 10: Utklipp av resultat (output)



I neste utdrag fortset gutane med å prøve å få posisjonert sirkelen riktig med utgangspunkt i kvadratet:

4.1.3 Utdrag 3

16) Alex: Den må, den må halvegs... så lengde / 2. Bør ikkje det fungere då?

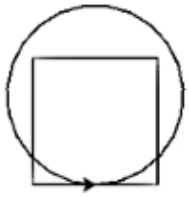
17) Oliver: Nei... eg trur eigentleg ikkje det, men det kan hende... [Alex skriv og køyrer programmet]

Figur 11: Utklipp av kodar

```
Lengde = 100
k1 = 50
radius = 100

for count in range(4):
  forward(Lengde)
  left(90)
forward(Lengde/2)
circle((sqrt(Lengde**2 + Lengde**2))/2)
```

Figur 12: Utklipp av resultat (output)



18) Alex: Nei. Det gjer ikkje det.

19) Oliver: Nei... [stillhet]

20) Alex: Eg...

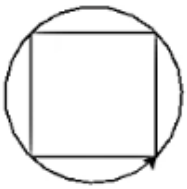
21) Oliver: Neimen... okei, eg prøver noko her...
Forward (Lengde), sant? Så tar vi...
left (45)... her... [skriv, og køyrer programmet]

Figur 13: Utklipp av kodar

```
Lengde = 100
k1 = 50
radius = 100

for count in range(4):
  forward(Lengde)
  left(90)
forward(Lengde)
left(45)
circle((sqrt(Lengde**2 + Lengde**2))/2)
```

Figur 14: Utklipp av resultat (output)



22) Oliver: DER JA!

23) Alex: ÅH, STEIKE!

24) Oliver: Der jaaaa.

25) Alex: Sann ja, den måtte jo ut!

Alle dei fire komponentane er registrerte i utdraget. Komponenten *tankegangskompetanse* handlar mellom anna om å kunne stille spørsmål og å vete kva svar ein kan forvente å få. Alex føreslår i utsegn 16 at pila bør starte på midten av linja i staden for nede i høgre hjørne når dei skal lage sirkelen, og at dei difor bør halvere lengda. Vidare spør han: «Bør ikkje det fungere då?». Ved å stille dette spørsmålet viser Alex at han kan stille eit matematisk spørsmål, og truleg veit han kva svar han kan forvente å få. Sidan Alex i sitat 16 hevdar at pila «må halvegs» og deretter spør om det ikkje bør fungere, verkar det som at han forventar å få svaret «ja». Oliver gjev uttrykk for at han ikkje er heilt einig med forslaget, men gjev likevel rom for at Alex kan prøve det ut.

Elevane sin *problembehandlingskompetanse* kjem til syne i utdraget ved at dei held fram med å handtere det matematiske problemet som no viser seg å vere plasseringa av sirkelen i forhold til kvadratet. I utsegn 16 presenterer Alex eit forslag til kva som bør vere utgangspunktet til sirkelen, og i utsegn 21 kjem Oliver med eit anna forslag. Gutane viser med dette evne til å vurdere gyldigheita av det dei har gjort, identifisere problemet og tenkje ut ein ny strategi for å løyse problemet.

Komponenten *algoritmisk tenking* er identifisert fleire stader i utdraget. Begge gutane er skapande, uthaldande og viser at dei kan evaluere moglege løysingar. Dei kjem med nye forslag til korleis dei skal løyse oppgåva, og viser at dei kan forbetre og endre kodane dei har skrive. Det å gjere feil er ein viktig del av prosessen til *den algoritmiske tenkaren*, og han må ha strategiar for å oppdage feila og rette på dei. I utdraget og i teksten før utdraget kjem det fram at gutane gjer fleire feil på vegen mot riktig løysing. Det er litt uklart kva for nokre strategiar gutane brukar til å oppdage feila, men det ser ut som at dei tek utgangspunkt i den figuren dei allereie har laga til å vurdere kva endringar dei skal gjere for å få pila til å bevege seg dit dei ynskjer. Når dei rettar på feila dei har gjort verkar det som at dei har ein logisk tanke bak det dei gjer og at dei prøver å føresjå korleis pila vil bevege seg med dei nye endringane. Dette kjem til syne ved at gutane heile vegen forklarar kva dei gjer og snakkar medan dei skriv.

Komponenten *utforsking* har òg vist seg fleire stader i utdraget. I sitat 17 når Oliver svarar at han eigentleg ikkje trur at forslaget til Alex vil stemme, viser han at han eit kritisk blikk på det dei utforskar. Likevel gjev Oliver rom for at Alex kan prøve ut forslaget sitt når han seier «men det kan hende». Oliver viser teikn på utforskande kvalitetar ved at han er kritisk, men samstundes samarbeidsvillig og lar Alex få prøve ut forslaget sitt. I utsegn 21 kjem det fram at Oliver vil prøve ut noko nytt. Han viser ei undersøkande tilnærming til oppgåva og kjem med forslag til ei løysing. Samstundes ser det ut som at Oliver ynskjer at dei skal oppnå ei felles forståing då han seier «sant?» undervegs. Etter å ha køyrt programmet jublar gutane når dei ser at dei har fått riktig svar. Alex avsluttar med å seie «den måtte jo ut!», noko som kan bety at han har skjønt kva Oliver tenkte og at dei har oppnådd ei felles forståing.

4.1.4 Operasjonell og/eller strukturell forståing

På bakgrunn av dei fire komponentane kan ein sjå at Alex og Oliver viser teikn til både *strukturell* og *operasjonell forståing* for dei geometriske konsepta dei tek føre seg i samtaleutdraga over. Då dei i utdrag 1 påpeikar samanhengen mellom diagonalen i kvadratet

og diameteren i sirkelen viser dei *tankegangskompetanse* gjennom abstraksjon. Tankegangskompetansen kan i dette tilfellet knytast til elevane sin strukturelle forståing, sidan dei behandlar konsepta som om dei refererer til abstrakte objekt. Ved å visualisere ser elevane eigenskapar hos figuren som gjer dei abstrakte ideane om diagonal og diameter meir handfaste og lettare å behandle som om dei var materielle einingar. Etter å ha oppdaga denne samanhengen har gutane noko å ta utgangspunkt i for å gå vidare med oppgåva. Dette kan sjåast i samanheng med metaforen om å orientere seg i eit kart før ein går til ei spesiell gate. Likskapen mellom sirkelen og kvadratet brukar elevane som eit «kart» når dei skal tenkje ut ein strategi for å løyse oppgåva. Deretter bevegar gutane seg mot den «spesielle gata» dei skal til, som i dette tilfellet vil vere å lage figuren som oppgåva ber om.

Alex set i gong arbeidsprosessen når han seier at dei kan starte med å lage eit kvadrat. Han viser *problembehandlingskompetanse* ved at han set i gong med oppgåva og handterer det matematiske problemet. Dette kan sjåast som eit teikn på at Alex har *operasjonell forståing* for konseptet kvadrat. Operasjonell forståing handlar i stor grad om prosessar, og her viser Alex evne til å iverksetje prosessar. Gutane programmerer kvadratet utan problem, før dei i utdrag 2 skriv inn kodane som skal til for å lage diagonalen. I dette utdraget vart komponenten *algoritmisk tenking* identifisert i alle utsegnene, sidan det ser ut som at gutane har god innsikt i algoritmane og prosessane som ligg til grunn for Pytagoras' setning. Dette kan vere ein sterk indikator på at gutane har operasjonell forståing for setninga.

I utdrag 3 kan *operasjonell forståing* identifiserast på bakgrunn av komponentane *problembehandlingskompetanse* og *algoritmisk tenking* som kjem til syne hos gutane. Problembehandlingskompetanse kjem til uttrykk når gutane vurderer gyldigheita av det dei har gjort i Python, og brukar dette til å tenkje ut ein ny strategi. Algoritmisk tenking er identifisert i det gutane kjem med nye forslag til korleis dei skal løyse oppgåva gjennom å forbetre kodane dei har skrive. Gjennom desse komponentane kan ein sjå teikn til operasjonell forståing hos gutane, sidan det ser ut som at dei har god innsikt i algoritmane og prosessane dei går gjennom. Dei greier å ta utgangspunkt i kodane dei allereie har skrive og gjer små endringar i desse for at resultatet (output) skal verte annleis. Til slutt fører endringane til riktig resultat. I dette tilfellet kan det sjå ut som at gutane utviklar sin operasjonelle forståing for konseptet kvadrat innskrive i sirkel, sidan dei blir merksame på eit ekstra steg som må til for å løyse oppgåva.

Som nemnt viser gutane teikn til *strukturell forståing* i starten av arbeidsprosessen. Vidare i arbeidsprosessen kan det vere vanskeleg å identifisere denne forståinga hos gutane, sidan det «abstrakte objektet» (kvadrat innskrive i sirkel) allereie er kjent ved at det er illustrert i oppgåveteksten. Likevel er det verdt å nemne at gutane ikkje diskuterer eigenskapane til sirkelen eller kvadratet, noko som kan vere eit teikn på at eigenskapane til desse konsepta er openbare og treng difor ikkje å bli nemnt eksplisitt. Det at dei kan snakke om sirkel og kvadrat utan å måtte snakke om prosessane som ligg til grunn, antyder at dei har strukturell forståing for desse konsepta.

Til tross for at den *strukturelle forståinga* kan vere vanskeleg å identifisere når elevane set i gong med programmeringa, kan det tenkjast at den likevel er der gjennom heile arbeidsprosessen. Gutane har nemleg danna seg eit mentalt «kart» som dei tek utgangspunkt i for å løyse oppgåva, og arbeidsprosessen handlar om å finne fram i dette kartet. Dei veit tilsynelatande heile tida kva dei vil fram til, og kva som må til for å lage ein sirkel og eit kvadrat i riktig storleik. Det som viser seg å vere hovudproblemet i oppgåva er å finne ut korleis sirkelen og kvadratet skal plasserast riktig i forhold til kvarandre. Det kan sjå ut som at gutane løyser problemet gjennom ei *utforskande* tilnærming der dei prøver seg fram. Samstundes er dei kritiske ved at dei tilsynelatande tenkjer gjennom dei ulike forslaga og prøver å sjå for seg korleis kodane vil fungere før dei skriv dei inn. Dette er truleg grunnen til at Oliver i utdrag 3 plutseleg ser ein samanheng som er essensiell for å løyse oppgåva. Ein kvar matematisk problemløysingsprosess kan bli sett på som eit samspel mellom operasjonelle og strukturelle tilnærmingar, der begge tilnærmingane blir utnytta på best mogleg måte. Med utgangspunkt i refleksjonane over kan det hevdast at det er ein slikt samspel Alex og Oliver har oppnådd, då dei har utnytta både *operasjonell* og *strukturell forståing* for å løyse oppgåva.

Hittil har det blitt synleggjort at gutane har vist teikn til *operasjonell* og *strukturell forståing* i arbeidsprosessen. Eg har òg løfta fram at gutane truleg har utvikla sin operasjonelle forståing for konseptet kvadrat innskrive i sirkel. Det som ikkje har blitt belyst er om gutane har utvikla sin strukturelle forståing gjennom å løyse denne oppgåva. Samtaleutdraga er sterke indikatorar på at Alex og Oliver er evnerike elevar i matematikk, og at dei har mykje kunnskap frå før. Dette kan vere ein grunn til at det er utfordrande å avgjere om dei har utvikla strukturell forståing ved å løyse denne oppgåva. For å kunne sjå nærmare på dette skal eg bruke *modellen for tileigning av matematiske konsept* (sjå kapittel 2.8.1).

4.1.4.1 Modell for tileigning av matematiske konsept

Ved fyrste augekast kan det sjå ut som at gutane er på *steg 3 - tingleggjering* i modellen, sidan dei tilsynelatande ser på kvadrat og sirkel som fullverdige objekt. Likevel har dei utfordringar med å plassere sirkelen og kvadratet riktig i forhold til kvarandre, noko som indikerer at dei er på *steg 2 - samantrekning* når det kjem til konseptet kvadrat innskrive i sirkel. Steg 2 - samantrekning handlar om å kombinere prosessar og gjere samanlikningar. I dei to fyrste utdraga samanliknar gutane kvadratet og sirkelen, og kombinerer prosessane som ligg til grunn for diagonalen og diameteren. I utdrag 3 kan det sjå ut som at det skjer eit «momentant kvantesprang» hos Oliver, då han oppdaga at pila må rotere 45 grader før dei kan lage sirkelen. Han ser plutseleg noko kjent (kvadrat, sirkel og vinklar) i eit heilt nytt lys, og det kan dermed tenkjast at han har gått vidare til steg 3 i modellen då konseptet «kvadrat innskrive i ein sirkel» tilsynelatande har blitt tingleggjort. Vidare betyr dette at nye matematiske objekt kan bli konstruert ut i frå dette tileigna konseptet, og at Oliver kan starte dei tre stega på nytt på eit høgare nivå.

I Alex sitt tilfelle er det litt meir uklart om ein kan seie at det «momentane kvantespranget» skjer hos han, sidan det er Oliver som oppdagar den siste naudsynlege endringa. Men, med tanke på reaksjonen Alex har når han ser at dei får riktig svar på oppgåva, kan det sjå ut som at han ser det same som Oliver. Sjølv om det ikkje er Alex sin idé å rotere pila, ser det likevel ut som at det går opp eit lys for han då han i sitat 25 seier at «den måtte jo ut!». Det kan difor tenkjast at Alex òg har nådd *steg 3* og at konseptet «kvadrat innskrive i sirkel» har blitt *tingleggjort*. Vidare inneber dette at han er klar for å starte dei tre stega på nytt, på eit høgare nivå. Det at konseptet tilsynelatande har blitt *tingleggjort* for begge gutane, tyder sterkt på at dei har utvikla sin *strukturelle forståing*. Oppsummeringsvis kan ein difor seie at gutane truleg har utvikla både *operasjonell* og *strukturell forståing* for konseptet «kvadrat innskrive i sirkel» gjennom å løyse oppgåva i Python.

4.2 Silje og Maria programmerer to formlike kvadrat

I dei neste samtaleutdraga møter vi Silje og Maria. I ei kort samtale med lærar Runar får eg informasjon om at dette er to elevar som kjenner seg svært usikre når det gjeld programmering. Eg og Runar blir difor einige om at elevane kan få litt hjelp undervegs. Dette blir forklart for elevane og dei får beskjed om at det berre er ynskjeleg at dei samarbeider og gjer sitt beste. Deretter får dei utdelt si fyrste oppgåve:

Figur 15: Oppgåve 1 frå oppgåvearket

1. Lag to formlike kvadrat.

Jentene har litt utfordringar med å kome i gong, og student forklarar difor i korte trekk kva dei ulike kodane gjer, med utgangspunkt i hjelpearket (sjå Vedlegg II). Deretter set jentene i gong med oppgåva. Maria byrjar med å skrive koda *forward (100)* med ein gong, utan å diskutere med Silje kva framgangsmåte dei skal bruke for å løyse oppgåva. Maria køyrer programmet med denne koda og lagar dermed ein strek. Deretter finn jentene ut saman at koda *left (90)* kan skrivast for at pila skal rotere og peike oppover. Vidare skriv dei *forward(100)* enda ein gong.

I fylgjande dialogutdrag har jentene programmert to sider i eit kvadrat, og Maria stiller spørsmål til korleis dei kan lage to kvadrat:

4.2.1 Utdrag 1

1) Maria: Okei, då er det tilbake til...
2) Silje: Venstre
3) Maria: Venstre igjen, men korleis skal vi få lage to då? Vi må sikkert berre prøve å få dei til å... vent då... «to formlike kvadrat» [les oppgåveteksten]... nei, vi får finne ut av det etterpå. Visst vi berre prøver å få dette ferdig fyrst, så har vi eitt. [skriv] Det var *90*, oi! Vent... sann. [køyrer programmet]

Figur 16: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
```

Figur 17: Utklipp av resultat (output)

I dialogutsnittet har jentene tilsynelatande forstått kva som må til for å fullføre kvadratet dei har byrja på, sidan det i sitat 1 og 2 ser ut som at dei har kome inn i ei rutine. I sitat 3 verkar det som at Maria skjønar at dei må lage to kvadrat, men at ho foreløpig ikkje veit korleis dei skal gjere det, eller kva det inneber at dei skal vere «formlike». Ho bestemmer seg dermed for at dei skal fullføre eitt kvadrat fyrst.

Alle dei fire komponentane er identifiserte i samtaleutdraget. Maria viser noko teikn til *tankegangskompetanse* i sitat 3 då ho stiller eit matematisk spørsmål til korleis dei kan lage to kvadrat. Til tross for at ho stiller eit spørsmål, verkar det ikkje som at ho forventar eit svar på dette spørsmålet. Ho svarar på spørsmålet sjølv, noko som kan indikere at dette er hennar måte å vise korleis ho tenkjer. Det ser ut som at ho i utsegn 3 prøver abstrahere og sjå for seg korleis den endelige figuren bør sjå ut, men at ho ikkje får det heilt til. Dette er det fyrste tilfellet i samtala mellom jentene at dei gjev uttrykk for at dei vurderer korleis dei skal løyse oppgåva i sin heilskap.

Komponenten *problembehandlingskompetanse* er identifisert i utsegn 3 av Maria, sidan det ser ut som at ho er inne på ein midlertidig strategi for å løyse oppgåva. Når det viser seg at ho ikkje heilt greier å sjå for seg figuren oppgåva ber om, så seier ho: «Nei, vi får finne ut av det etterpå. Visst vi berre prøver å få ferdig dette fyrst...». Med denne kommentaren viser ho noko evne til å tenkje ut kva dei kan gjere, med eitt steg av gongen. Dette kan tyde på at ho handterer problemet etter beste evne, på sin eigen måte.

Komponenten *algoritmisk tankegang* er identifisert i alle tre utsegnene i dialogen. I utsegn 1 og 2 er komponenten identifisert ved at jentene skriver kodar og stegvis lagar det fyrste kvadratet. I utsegn 3 vurderer Maria vidare kva for nokre steg som trengst for å løyse oppgåva. Sjølv om ho ikkje veit alle stega som må til, så er ho skapande og eksperimenterande ved å seie at dei kan lage eit kvadrat fyrst, og finne ut av resten etterpå.

Komponenten *utforsking* kjem til uttrykk ved at Maria tilsynelatande har ei undrande og spørjande tilnærming når ho prøver å finne ut kva som må til for å lage to kvadrat. Det ser ut som at ho har ein undersøkande arbeidsmetode der ho prøver seg fram for å halde framgang i arbeidsprosessen.

I neste dialogutsnitt er jentene ferdige med å programmere tre sider av det fyrste kvadratet, og dei vender seg nok ein gong tilbake til oppgåveteksten. Det fylgjande dialogutsnittet viser at jentene er usikre på kva som ligg i omgrepet «formlike»:

4.2.2 Utdrag 2

4) Maria: Men er dette formlike då? [ler]
5) Silje: Eg veit ikkje kva formlike er ein gong eg.
6) Maria: Ja. Vi berre ser.
[Jentene programmerer ferdig kvadratet].

Komponenten *tankegangskompetanse* har kome til uttrykk i utsegn 4, der Maria føreheld seg til matematiske spørsmål, ved å sjølv stille eit spørsmål og truleg vite kva svar ho kan forvente å få. I motsetnad til spørsmålet hennar i utdrag 1 (4.2.1) , ser det ut som at ho i dette tilfellet faktisk søker eit svar, og at ho ikkje berre tenkjer høgt. Når Silje svarar at ho ikkje veit kva formlike er, seier Maria: «Ja. Vi berre ser». Den potensielle diskusjonen rundt konseptet formlikhet blir dermed sett på vent.

Det har òg blitt identifisert element av komponenten *problembehandlingskompetanse* i utsegn 4 i utdraget. Denne utsegna kan sjåast på som ei slags vurdering av måten dei handterer oppgåva på. Maria stoppar opp og stiller spørsmål om dette er «formlike», noko som kan tyde på at ho tenkjer på oppgåveteksten og vurderer om dei gjer det som oppgåva ber om. Utdraget kan i tillegg vere ein indikator på at det ligg meir i oppgåva enn rutinemessige prosedyrar, då jentene tilsynelatande oppfattar den som eit problem som må identifiserast og løysast.

I dialogutdraget er komponenten *algoritmisk tenking* identifisert i utsegn 6, då det ser ut som at Maria ikkje er redd for å gjere feil. Det kjem fram av utdraget at jentene ikkje veit kva som ligg i omgrepet «formlike», som er eit sentralt omgrep i oppgåveteksten. Likevel vel Maria å halde fram med oppgåva, sjølv om ho truleg skjønar at dei kan gjere feil sidan dei ikkje veit kva dette omgrepet inneber. Ved å ta avgjersla om å likevel gå vidare med oppgåva gjev ho utrykk for å vere skapande og eksperimenterande, noko som òg er sentrale kjenneteikn hos den algoritmiske tenkaren.

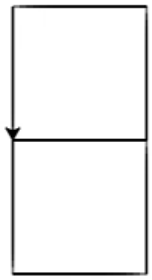
Komponenten *utforsking* har blitt identifisert i utsegn 6, sidan det ser ut som at Maria vil gå vidare i oppgåva ved å prøve seg fram og undersøke. I tillegg kjem utforskande eigenskapar til syne hos Maria når ho stiller spørsmål i utsegn 4 og gjev uttrykk for å vere undrande.

I arbeidsprosessen etter utdraget over blir elevane ferdige med å programmere det fyrste kvadratet. Deretter finn dei ut at dei bør gå til høgre for å lage eit nytt kvadrat som kan henge saman med det fyrste. Dei prøver koda *right (45)* som står på hjelpearket, men ser då at pila ikkje går i ynsket retning. Dei fjernar denne koda og samtala stoppar opp. Student viser dei dermed ei løkke på hjelpearket som dei kan bruke for å teikne eit nytt kvadrat meir effektivt. Jentene prøver å nytte seg av løkka, og køyrer programmet fleire gongar der dei får opp feilmelding på grunn av feil syntaks. Til slutt får jentene fylgjande resultat:

Figur 18: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
forward (100)
left (90)
forward (100)
for count in range (4):
  forward(100)
  left(90)
```

Figur 19: Utklipp av resultat (output)



Dialogen fortset:

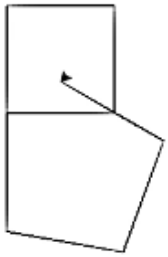
4.2.3 Utdrag 3

7) Silje: Er det riktig? [ser på student]
8) Student: Altså, de har laga to heilt like kvadrat no. De skal lage to formlike, så dei må jo eigentleg vere i to ulike storleikar.
9) Silje: Kan ikkje du berre ta *left,* så *100*? [ler]
10) Maria: [ler] Åja, jo det gløymte eg, vi kan jo skru opp denne greia trur eg, så blir den større. Vent då, vi kan berre ha den der [*forward*].
11) Silje: Eller mindre, så kan du berre ta den på sann *90*... nei, *80*. Vi tar den sant...
12) Maria: Eg veit ikkje kva som går eller ikkje.
13) Silje: Vi tar den berre på sann *110*, eg veit ikkje om det går [skriv]. Så tar vi ned til, det var *left* sant. Den kan vi ta til, ehm, *80*. Eg veit ikkje heilt om det er sann, vi prøver [køyrer programmet].

Figur 20: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
forward (100)
left (90)
forward (100)
for count in range (4):
  forward(110)
  left(80)
```

Figur 21: Utklipp av resultat (output)



Det at Silje spør om svaret er riktig, indikerer at ho er usikker på om dette er formlike kvadrat eller ikkje. Maria kommenterer ikkje figuren og ein kan dermed anta at begge elevane framleis er usikre angåande konseptet formlikhet, før student forklarar at kvadrata må ha ulik storleik.

Komponenten *tankegangskompetanse* har i dette utdraget blitt identifisert i Silje si utsegn 9. Etter at studenten har sagt at kvadrata må ha ulik storleik, spør Silje om dei kan endre talet bak *left* i løkka. I utsegna stiller ho fyrst og fremst eit spørsmål, men dersom ein ser litt nærmare kan det sjå ut som at spørsmålet inneheld eit forsøk på generalisering. Sannsynlegvis tenkjer Silje at endringa ho føreslår vil gjere kvadratet større, då ho vil endre talet bak *left* frå *90* til *100*. Denne endringa vil derimot endre på vinklane, og ikkje på sidene i figuren. Når Maria i sitat 10 svarar «Åja, jo det gløymte eg», verkar det som at ho tenkjer tilbake på introduksjonsøktene der dei lærte om løkker. Vidare legg Maria til at dei kan «berre ha den der [forward]». Truleg tenkjer ho òg at dei kan endre talet bak *left* for å få eit større kvadrat, og la *forward* stå urørt. Dersom Silje heller hadde føreslått ei endring i talet bak *forward,* ville forslaget vore riktig. Ved å berre justere talet bak *forward* kan løkka fungere generelt for alle kvadrat, då vinklane alltid vil vere 90 grader og dei fire sidene kan bli lenger eller kortare.

Komponenten *problembehandlingskompetanse* har ikkje blitt identifisert i utdraget. Noko av grunnen til dette kan vere at det verkar som at jentene ikkje heilt greier å identifisere det matematiske problemet.

I likskap med forgje utdrag er komponenten *algoritmisk tenking* identifisert i dette utdraget ved at jentene tilsynelatande ikkje er redde for å gjere feil. I sitat 11 kjem Silje med eit forslag om å justere ned talet bak *left*. Maria svarar at ho ikkje veit kva som går eller ikkje. Silje vel til slutt å skrive kodane *forward(110)* og *left(80)* og seier: «Eg veit ikkje heilt om det er sann, vi

prøver». Det kan sjå ut som at jentene nemner tilfeldige tal dei kan skrive inn i løkka, sidan begge gjev uttrykk for at dei ikkje veit kva som vil fungere.

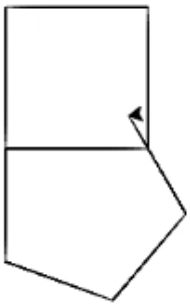
Komponenten *utforsking* har blitt identifisert mot slutten av utdraget. Det at jentene gjev uttrykk for at dei ikkje veit kva som vil fungere eller ikkje, kan vere eit teikn på at dei prøver seg fram. I dei siste sitata går jentene vekk i frå idéen om at dei skulle endre på berre éin av kodane i løkka. Dei endar opp med å endre begge kodane, noko som dei ikkje grunngjev. Dette kan bety at dei prøver ut tilfeldige tal.

I den fortsetjande samtala etter utdraget over held jentene fram med å prøve ut fleire ulike tal i løkka. Det ser framleis ut som at tala dei skriv inn er nokså tilfeldige. Det verkar ikkje som at dei tenkjer på at vinklane må vere 90 grader i eit kvadrat, og dei diskuterer heller ikkje kva funksjon kodane *forward* og *left* har i løkka. Tala jentene prøver ut gjev dei to fylgjande resultata:

Figur 22: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
forward (100)
left (90)
forward (100)
for count in range (4):
  forward(80)
  left(70)
```

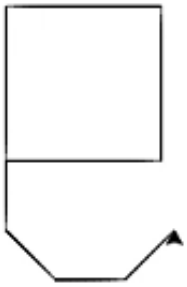
Figur 23: Utklipp av resultat (output



Figur 24: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
forward (100)
left (90)
forward (100)
for count in range (4):
  forward(45)
  left(45)
```

Figur 25: Utklipp av resultat (output)



Neste utdrag startar med at Maria påpeikar at kodane dei har skrive ikkje har fungert fordi *left* må vere *90* for at det skal verte eit kvadrat. Denne observasjonen gjer at jentene greier å løyse oppgåva:

4.2.4 Utdrag 4

14) Maria: Neimen kan ikkje vi berre sjå her, vent litt, vi må tenkje. Fordi at den har, eh, okei det er 90 sant, i hjørna [peikar på Figur 25 med datamusa]. Så det der er vi nøydt til å bytte, så *left*, det er nøydt til å vere *90*, visst ikkje så får vi ikkje 90.

15) Silje: Ja...

16) Maria: Så kan vi endre på den fyrste [koda] sjølvsagt.

17) Silje: Ja, den kan vere sann...

18) Maria: Fordi den [*left*] er 90 grader.

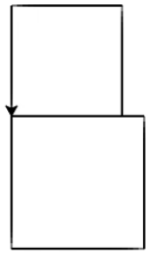
19) Silje: Ja.

20) Maria: Vi kan berre ta den [*forward*] på sann *120* og sjå om det går [køyrer programmet].

Figur 26: Utklipp av kodar

```
forward (100)
left (90)
forward (100)
left(90)
forward (100)
left (90)
forward (100)
for count in range (4):
  forward(120)
  left(90)
```

Figur 27: Utklipp av resultat (output)



21) Silje: Der ja!

22) Maria: Ja! Er det...[ser på studenten (som nikka)] Vi hadde jo 45! [ler]

I dette siste utdraget til elevparet Silje og Maria er alle dei fire komponentane identifisert. *Tankegangskompetanse* er identifisert hos Maria då ho oppdagar at *left* må vere *90* for at vinklane i det siste kvadratet skal bli 90 grader. Denne oppdaginga kan sjåast på som ein abstraksjon, der Maria stoppar opp for å sjå for seg eit kvadrat og dermed trekkjer fram viktige eigenskapar til kvadratet.

Komponenten *problembehandlingskompetanse* er synleg i utdraget ved at elevane ser ut til å oppdage problemet (vinklane). Vidare handterer dei problemet ved å gjere naudsynte endringar i løkka, noko som endar med riktig løysing på oppgåva. Elevane jublar etter dei at ser løysinga i Figur 27, noko som tyder sterkt på at begge meiner dei har løyst oppgåva riktig og fått eit gyldig svar. Maria vender seg likevel mot student for å få ei stadfesting, noko som er naturleg med tanke usikkerheita dei hadde rundt programmering og konseptet formlikhet

Elevane viser teikn til *algoritmisk tenking* ved at dei feilsøker og forbetrar koda/løkka dei har skrive. Dei viser dessutan at dei har greidd å halde ut med oppgåva, til tross for at dei har gjort mange feil undervegs.

Til slutt kan ein sjå *utforskande* eigenskapar hos Maria ved at ho har eit kritisk blikk på det dei utforskar. Tilsynelatande innser ho at dei må stoppe opp og tenkje over kva dei har gjort, i staden for å fortsetje å prøve seg fram. Dette medfører at ho identifiserer problemet og dermed greier å søke ei løysing.

4.2.5 Operasjonell og/eller strukturell forståing

Med utgangspunkt i dei fire komponentane som er identifisert i samtaleutdraga over, er det fleire element som peikar på at Silje og Maria har ein *operasjonell tilnærming* til oppgåva. I utdrag 1 (4.3.1) set jentene i gong med å skrive kodar utan å ha planlagt korleis dei skal løyse oppgåva. Komponenten *algoritmisk tenking* er identifisert gjennom heile dette utdraget, då jentene stegvis skriv kodar for å lage det fyrste kvadratet og vurderer kva som må til for å lage to kvadrat. Dette indikerer at jentene har operasjonell forståing for konseptet kvadrat, sidan dei gjennomfører prosessane som skal til for å lage kvadratet. I utdrag 2 og 3 (4.2.2 og 4.2.3) er komponenten algoritmisk tenking identifisert på bakgrunn av at jentene tilsynelatande ikkje er redde for å prøve og feile. Begge gjev uttrykk for at dei ikkje veit kva som ligg i omgrepet formlike, men likevel vel dei å prøve seg fram. Dette kan sjåast på som ei operasjonell tilnærming til oppgåva, sidan dei handlar utan å legge for mykje vekt på resultatet. I arbeidet med oppgåva blir Maria og Silje kjent med korleis dei skal teikne eit kvadrat i Python, på to ulike måtar. Då dei programmerte det fyrste kvadratet, lagde dei ei side av gongen. For å lage det andre kvadratet, brukte dei ei løkke som studenten viste dei på hjelpearket. Ved å gå inn og endre vinklane og sidene i dette siste kvadratet får jentene truleg ein betre innsikt i eigenskapane til formlike kvadrat, sidan alle stega (kodane) må vere heilt riktige for å oppnå ynsket resultat. Det kan dermed tenkjast at dei har utvikla noko operasjonell forståing for formlike kvadrat gjennom å løyse denne oppgåva.

I utdrag 1 (4.2.1) viser Maria noko teikn til *tankegangskompetanse* då ho gjev uttrykk for at ho abstraherer og prøver å sjå for seg korleis den endelege figuren bør sjå ut. I utgangspunktet kunne dette ha vore eit teikn på *strukturell forståing*, då det er snakk om abstrakte objekt. Likevel verkar det som at ho ikkje greier å visualisere to formlike kvadrat, noko som kan vere

eit teikn på manglande strukturell forståing. I lyset av dei fire komponentane kan det sjå ut som at elevane har manglande strukturell forståing i utdrag 2 og 3 òg (4.2.2 og 4.2.3). Dette kjem mellom anna til utrykk gjennom komponenten tankegangskompetanse, då elevane stiller spørsmål til omgrepet «formlike». Dette kan vere ein indikator på at dei ikkje greier å sjå for seg det abstrakte objektet oppgåva ber om. Manglande strukturell forståing kjem òg til uttrykk gjennom komponenten *utforsking* i utdrag 2 og 3, då det verkar som at jentene prøver seg fram utan å vete kva dei skal fram til.

Sjølv om det ikkje er identifisert teikn til *strukturell forståing* hos elevane i dei tre fyrste utdraga, verkar det som at det går opp eit lys for dei i utdrag 4 (4.2.4). Komponenten *tankegangskompetanse* vart i dette utdraget identifisert hos Maria då ho tilsynelatande abstraherer ved å stoppe opp for å sjå for seg eit kvadrat. Dermed oppdagar ho at *left* må vere *90* for at vinklane i kvadratet skal bli 90 grader. Denne oppdaginga kan knytast til ei strukturell forståing sidan ho viser evne til å abstrahere ved å trekkje fram viktige eigenskapar i eit kvadrat.

Gjennom *algoritmisk tenking* og *utforsking* har jentene prøvd ut mange forskjellige kodar før dei kom fram til riktig resultat. Dei gongane jentene fekk feil resultat viste dei teikn til at dei var missnøgde med resultatet, enten ved å kommentere at det var feil eller ved å le. Då dei derimot fekk riktig svar jubla dei, noko som kan tyde på at resultatet samsvarte med deira forventning til kva som ville vere riktig svar. Det bør nemnast at forventningane til elevane sannsynlegvis var forma av studenten sin kommentar om at kvadrata måtte vere av ulik storleik. Det kan likevel sjå ut som at elevane sin forventing av korleis to formlike kvadrat burde sjå ut, har utvikla seg undervegs. Dette kan vere eit teikn på at elevane i løpet av arbeidsprosessen har danna seg eit mentalt bilete av to formlike kvadrat, og dermed utvikla noko *strukturell forståing*.

Oppsummeringsvis kan ein seie at elevane har ei operasjonell tilnærming til oppgåva i starten, då dei gjev utrykk for at dei fokuserer mest på algoritmar og prosessar. Mot slutten av arbeidsprosessen ser det ut som at jentene går over til ei meir strukturell tilnærming til oppgåva, då dei blir meir og meir opptatt av å få eit resultat som vil stemme overeins med deira mentale bilete. I denne overgangen blir jentene tilsynelatande for ivrige, og fokuset på prosessane vart overdøyvd av iveren til å kome fram til riktig svar. Det er truleg dette som gjorde at elevane brukte mykje tid på prøving og feiling, då dei ikkje såg ut til å tenkje gjennom kva dei ulike kodane faktisk ville gjer. Det var fyrst når Maria stoppa opp og hadde eit kritisk blikk på kodane

dei hadde brukt, at ho greidde å løyse oppgåva. Ho brukte i dette tilfellet både sin *operasjonelle* og *strukturelle forståing* for å løyse oppgåva, då ho fokuserte både på algoritmar og det abstrakte objektet (to formlike kvadrat). Arbeidsprosessen til jentene viser eit eksempel på at strukturell og operasjonell forståing er to ulike sider av same sak. Ein kjem seg ikkje langt ved å berre sjå for seg eit objektet, utan å kjenne prosessane som ligg til grunn for dette objektet. Ein kjem seg heller ikkje langt ved å handle og gjennomføre prosessar utan å ha ein plan for kva desse skal føre til.

4.3 Anette og Else programmerer eit kvadrat med ein diagonal

I dette delkapittelet møter vi Anette og Else som skal programmere eit kvadrat med ein diagonal. Jentene har i forkant løyst ei oppgåve der dei har programmert ein rettvinkla trekant (som dei valte å løyse som ein 30°, 60°, 90°-trekant) og ei oppgåve der dei programmerte to formlike kvadrat. Fylgjande oppgåve er den tredje og siste oppgåva elevparet skal jobbe med:

2. Lag eit kvadrat med ein diagonal.

I fylgjande utdrag har jentene nettopp fått tildelt oppgåva. Dei startar med å diskutere omgrepet diagonal, og korleis dei skal gå fram for å løyse oppgåva:

4.3.1 Utdrag 1

1) Anette: Diagonal, det er sann skrå det.
2) Else: Skrå det, ja.
3) Anette: Då må vi få den til å gå...
4) Else: Vi må lage ein trekant... med ein firkant.
5) Anette: Den er jo 90, så då må den gå 45 grader ned.
6) Else: Ja, vi må lage ein trekant med ein firkant. Nei, ein firkant med ein trekant inni.
7) Anette: Ein firkant med ein strek inni.
8) Else: Eit kvadrat med ein diagonal... er ikkje ein diagonal sann då? [peikar med datamusa på eit kvadrat frå forgje oppgåve]
9) Anette: Det er berre skrått gjennom.
10) Else: Ja
11) Anette: Ja, då blir det to trekantar
12) Else: Det blir trekantar.

I utdraget er komponenten *tankegangskompetanse* identifisert hos begge elevane sidan det verkar som at dei abstraherer for å prøve å sjå for seg figuren, samstundes som dei berre snakkar om relevante detaljar. Det kan sjå ut som at jentene ser for seg den same figuren på ulike måtar. Medan Else seier at dei må lage ein «firkant med ein trekant inni», seier Anette at dei må lage «ein firkant med ein strek inni». Undervegs ser det ut som at dei blir innforstått med at dei ser

for seg den same figuren, men at dei forklarar den på ulike måtar. Til slutt konkluderer dei med at figuren vil bestå av to trekantar.

Jentene viser teikn til *problembehandlingskompetanse* ved at dei brukar tid på å diskutere kva oppgåva ber om, og dermed identifiserer kva noko av problemet er. Foreløpig har dei ikkje gått i detalj når dei snakkar om kva strategi dei skal bruke for å lage figuren, men dei set i gong med å handtere det matematiske problemet.

Komponenten *algoritmisk tenking* kjem til syne i utdraget ved at jentene evaluerer moglege løysingar for oppgåva. Begge jentene forklarar korleis dei oppfattar oppgåva, før dei set i gong med å vurdere kva steg som trengst for å løyse den.

Ei *utforskande tilnærming* til oppgåva er identifisert ved at jentene tilsynelatande skapar ei felles forståing for kva oppgåva dreier seg om. Dei dannar seg eit felles språk der dei brukar ord som «skrå» når dei diskuterer korleis dei vil gå fram i oppgåva. Deretter identifiserer dei eigeskapar i den tenkte figuren ved å samanlikne den med trekantar og firkantar. Else føreslår at dei må lage ein firkant med ein trekant inni, medan Anette seier at dei må lage ein firkant med ein strek inni. Dei blir einige om at figuren må sjå ut som ein firkant med ein strek som går «skrått gjennom», som dermed blir to trekantar.

I dialogen som fylgjer etter utdraget over blir jentene einige om at dei skal løyse oppgåva ved å fyrst programmere ein firkant og deretter ein trekant. Dei beheld eit kvadrat frå forgje oppgåve, og vel deretter å lage ein trekant. Framgangsmåten dei går for er noko tungvind, då dei berre manglar ein strek for å lage diagonalen og dermed fullføre figuren som oppgåva ber om. Dei vel i staden å gå halvegs rundt kvadratet igjen, slik at dei byrjar å lage ein trekant over kvadratet. Pila havnar dermed oppe i høgre hjørne av kvadratet, peikande oppover. Jentene brukar ein del tid på å finne ut kor mykje pila skal rotere derfrå, slik at dei kan lage diagonalen. Dei finn til slutt ut at den kan rotere 225 grader mot høgre - eit tal som dei finn ved å rekne 180 + 45. Dei blir deretter einige om at diagonalen må vere 400, og grunngjev dette med at den kortaste kateten er 200 og at hypotenusen (diagonalen) må vere det dobbelte av denne.

I neste utdrag startar jentene med å skrive *forward (400)* for å lage diagonalen. Det viser seg at *400* er for mykje, og jentene prøver dermed lågare tal. Undervegs i dialogen bryt student inn, då ho ser at elevane truleg vil klare å løyse oppgåva dersom dei prøver og feilar nok gongar.

Dette er ikkje ynskjeleg sidan det kan føre til at jentene overser moglegheita til å bruke Pytagoras' setning. I tillegg er elevane litt på overtid, med tanke på at dei skal bli intervjua før neste skuletime. For å kunne synleggjer utprøvingane til elevane og avbrytinga til studenten er fylgjande utdrag noko lenger enn tidlegare utdrag:

4.3.2 Utdrag 2

13) Else: Den må vere 400, no skal den vere 400.

14) Anette: No skal den vere det, okei.

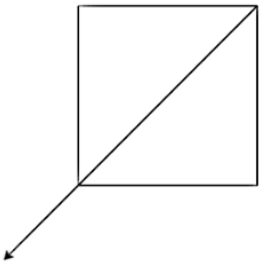
15) Else: Det må vere riktig med 400. [skriv og køyrer programmet]

Figur 28: Utklipp av kodar

```
lengde=200
for count in range (4):
  forward (lengde)
  left (90)

lengde= 200
forward (200)
left (90)
forward (200)
right (225)
forward (400)
```

Figur 29: Utklipp av resultat (output)



16) Anette: [ler]. Den må vere, den må vere...

17) Else: 300.

18) Anette: Eg veit ikkje [skriv 300 og køyrer programmet].

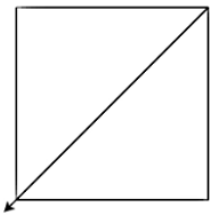
19) Else: Eg veit heller ikkje.

Figur 30: Utklipp av kodar

```
lengde=200
for count in range (4):
  forward (lengde)
  left (90)

lengde= 200
forward (200)
left (90)
forward (200)
right (225)
forward (300)
```

Figur 31: Utklipp av resultat (output)



20) Anette: Er det 200?

21) Else: Er det 200 då? Det skal jo ikkje vere like langt derfrå til der.

22) Anette: Kanskje det skal det?

23) Student: Men, korleis veit de kor lang den skal vere?

24) Anette: Eg veit ikkje.

25) Else: Vi veit ikkje. Det er hypotenusen, det er den der...

26) Student: Korleis finn de hypotenusen då?

27) Else: Den minste kateten...

28) Anette: Då må vi rekne ut ein...

29) Student: Det der med den kortaste kateten gjeld berre når det er 30°, 60°, 90°-trekantar.

30) Else: Ja, for no er det jo ikkje ein 30°, 60°, 90°-trekant.

31) Anette: Nei, det er sant. Då sliter vi litt.

32) Else: Hypotenus er hypotekalsk lære.

[Anette ler]

33) Else: Pytagoras' lære

34) Anette: Pytagoras' lære

[begge ler]

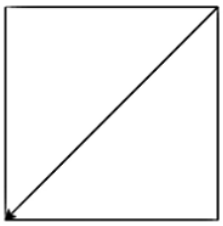
35) Else: Okei, vi prøver Pytagoras' lære. K i andre, pluss k i andre, er lik h i andre [skriv på papir]. Vi har katet pluss katet, så skal vi finne hypotenus.
200 i andre pluss 200 i andre er lik x i andre.
[Jentene reknar ut hypotenusen og finn ut at den er 282.84. Dei skriv dette inn i koda, køyrer programmet]

Figur 32: Utklipp av kodar

```
lengde=200
for count in range (4):
  forward (lengde)
  left (90)

lengde= 200
forward (200)
left (90)
forward (200)
right (225)
forward (282.84)
```

Figur 33: Utklipp av resultat (output)



I dette litt lengre utdraget er det berre komponentane *algoritmisk tenking* og *utforsking* som er identifiserte. Ein mogleg grunn til at dei andre komponentane ikkje er identifiserte kan vere at utdraget berer noko preg av at jentene prøver seg fram. Når dei oppdagar at den lengda som dei trudde var riktig (400) viser seg å vere feil, ser det ut som at dei ikkje prøver å finne ut kvifor det vart slik.

Komponenten *algoritmisk tenking* kjem til uttrykk når elevane snakkar om reglar og algoritmar i utdraget. I sitat 13 og 14 ser begge jentene ut til å vere einige i at hypotenusen må vere 400. Dei har tilsynelatande blanda inn regelen som gjeld for 30°, 60°, 90°-trekantar, sidan dei seier at hypotenusen må vere dobbelt så lang som kortaste katet. Sjølv om dette er feil, er det å gjere

feil ein viktig del av prosessen i *algoritmisk tenking*. Etter at student påpeikar at hypotenusen berre er det dobbelte av kortaste katet i 30°, 60°, 90°-trekantar, finn jentene fort ut at dei må bruke Pytagoras' setning for å rekne ut lengda på diagonalen. Det viser seg at dei har kjennskap til Pytagoras' setning, med tilhøyrande algoritmar. I utdraget er det Else sin kjennskap til Pytagoras' setning som kjem mest til syne, då ho i sitat 35 ramsar opp setninga steg-for-steg. Likevel bør det nemnast at jentene gjorde utrekninga i fellesskap i samtala som fylgjer etter dialogutdraget.

Komponenten *utforsking* er identifisert fleire stader i utdraget. I utsegn 16-19 vel jentene å skrive inn talet 300, sjølv om begge gjev uttrykk for at dei ikkje veit om det vil gå. Dette kan vere eit teikn på at talet 300 er noko tilfeldig, og at dei prøver seg fram. Det kan òg tenkjast at dei prøver ut talet 300, med utgangspunkt i augemål. I Figur 29 kan ein sjå at diagonalen blir litt for lang med koda *forward (400),* og det er dermed ikkje urimeleg å anslå at *forward (300)* kan vere riktig. Vidare kan ein sjå teikn til utforskande eigenskapar hos Else når ho i sitat 21 stiller seg kritisk til forslaget til Anette om å prøve ut talet 200. Ved å stille seg kritisk til dette forslaget gjev ho uttrykk for at ho tenkjer gjennom tala dei skriv inn, og at dei dermed ikkje er heilt tilfeldige. Ein kan i tillegg sjå ei utforskande tilnærming hos Anette då ho i sitat 20 spør om det kan vere talet 200. Ved å presentere forslaget som eit spørsmål, viser ho teikn til at ho er undrande og at ho prøver seg fram. Til slutt har komponenten utforsking blitt identifisert i Else si utsegn 35, der ho seier «Okei, vi prøver Pytagoras' setning». Det at ho brukar ordet «prøver» kan vere ein indikator på at ho er noko usikker på om forslaget er riktig og at ho har ei undersøkande tilnærming til prosessen.

4.3.3 Operasjonell og/eller strukturell forståing

Med bakgrunn i dei fire komponentane som er identifiserte i samtaleutdraga over, kan ein sjå fleire teikn til *operasjonell* og *strukturell forståing* hos Else og Anette. I utdrag 1 (4.3.1) kan ein i lyset av komponentane *tankegangskompetanse* og *utforsking* identifisere teikn til strukturell forståing hos dei to jentene. Tankegangskompetanse kjem til syne ved at jentene tilsynelatande abstraherer for å prøve å sjå for seg eigenskapane til eit kvadrat med ein diagonal. Til å byrje med verkar det som at jentene ser for seg den same figuren på ulike måtar. I løpet av dialogutdraget kan ein sjå teikn på at dei dannar seg ei felles forståing for figuren, noko som er sentralt i utforsking. Else seier at dei må lage ein «firkant med ein trekant inni», medan Anette føreslår ein «firkant med ein strek inni». Til slutt blir dei einige om at figuren vil sjå ut som to trekantar. Komponentane tankegangskompetanse og utforsking indikerer i dette tilfelle

at jentene har ei strukturell forståing for konseptet kvadrat med ein diagonal, sidan dei tek føre seg eit produkt utan å snakke om prosessane som ligg til grunn for dette produktet. Dessutan viser jentene noko evne til å visualisere ved at dei tilsynelatande ser «usynlege» objekt i matematikk.

I utdrag 2 (4.3.2) når jentene skal lage diagonalen, kan det i starten sjå ut som at dei har noko manglande *operasjonell forståing* for hypotenus. Dette kjem til uttrykk gjennom komponenten *algoritmisk tenking* ved at jentene ser ut til å blande inn regelen som gjeld for 30°, 60°, 90°-trekantar. Dette viser teikn til manglande operasjonell forståing for dei ulike reglane som ligg til grunn for rettvinkla trekantar og Pytagoras' setning. Dessutan ser det ikkje ut som jentene prøver å finne ut kva som gjer at talet 400 er feil, noko som òg kan vere eit teikn på at dei ikkje har kontroll på reglane og algoritmane. I staden for å prøve og finne ut kvifor talet 400 er feil, ser dei ut til å bruke ei *utforskande tilnærming* der dei prøver ut andre tal som kan kanskje kan vere riktige. Student bryt dermed inn, og gjev dei nokre hint på vegen. Deretter viser komponenten algoritmisk tenking seg ved at jentene tilsynelatande kjenner godt til algoritmen bak Pytagoras' setning, sidan dei ramsar opp alle stega og reknar ut hypotenusen utan hjelp frå student. I lyset av jentene sin algoritmiske tankegong kjem ei operasjonell forståing for Pytagoras' setning til uttrykk, ved at dei har kontroll på prosessar og algoritmar. Det kan sjå ut som at jentene har kjennskap til dei ulike reglane som gjeld for rettvinkla trekantar, men at dei har utfordringar med å identifisere i kva for nokre tilfelle dei ulike reglane kan brukast og i kva tilfelle dei ikkje kan brukast. Gjennom å løyse denne oppgåva har jentene truleg blitt meir merksame på dei ulike reglane som gjeld for rettvinkla trekantar, og dermed utvikla sin operasjonelle forståing.

Som nemnt viser elevane teikn til *strukturell forståing* for konseptet «kvadrat med diagonal» i utdrag 1 (4.3.1). Dei tek føre seg eit produkt av visse prosessar, utan å snakke om prosessane i seg sjølv. Det kan difor sjå ut som at dei har starta med dei strukturelle sidene av konseptet for å skape ei oversikt. Vidare i arbeidsprosessen er den strukturelle forståinga til jentene vanskeleg å identifisere. Grunnen til dette er truleg at dei set i gong prosessane som skal til for å lage produktet. Likevel kan det sjå ut som at jentene brukar sin strukturelle forståing som ein vegvisar til meir detaljert informasjon. Det kan dermed tenkjast at den strukturelle forståinga er til stades i heile arbeidsprosessen, men at den ikkje utviklar seg.

Alt i alt kan det sjå ut som at Anette og Else har utvikla noko *operasjonell forståing* for konseptet «kvadrat med diagonal» ved å løyse denne oppgåva, sidan dei har blitt betre kjent med algoritmane og prosessane som byggjer opp konseptet. Den *strukturelle forståinga* har tilsynelatande vore til stades heile vegen og ikkje utvikla seg. Det kan tenkjast at ved å ha ei strukturell forståing for «kvadrat med diagonal», så har jentene hatt eit betre utgangspunkt for å utvikle den operasjonelle forståinga for konseptet. Med ei strukturell forståing hadde dei ei viss oversikt frå starten av, der dei viste kva dei skulle arbeide mot.

5 Diskusjon

I kvart delkapittel i analysen vart samtala til eit elevpar analysert med utgangspunkt i det nye samansette rammeverket som er beskrive i kapittel 3.5.1. Analysen av samtalane vart teke utgangspunkt i for å undersøke om elevane viste teikn til *operasjonell* og *strukturell forståing* av dei geometriske konsepta dei jobba med, og om denne forståinga utvikla seg eller ikkje. I dette kapittelet vil sentrale funn frå analysen bli presentert og diskutert for å svare på problemstillinga:

Korleis kan elevar utvikle strukturell og operasjonell forståing av geometri gjennom utforsking i programmeringsspråket Python?

For å kunne svare på problemstillinga vil det bli retta fokus mot kva programmering i Python kan ha tilbydt den *strukturelle* og *operasjonelle forståinga* til elevane. I kapittel 5.1 vil det bli løfta fram sentrale funn frå analysen som kan seie noko om kva rolle Python kan ha hatt for utviklinga av forståingane. Vidare skal vi i kapittel 5.2 sjå nærmare på kva betyding det kan ha å vere to stykk som programmerer saman. Til slutt vil det i kapittel 5.3 bli retta fokus mot kva konsekvensar programmering kan ha for matematikkundervising. Refleksjonane som blir gjort i dei tre delkapitla vil bli sett opp mot teori og tidlegare forsking.

5.1 Python si rolle i elevane si utvikling av *operasjonell* og *strukturell forståing* for geometri

5.1.1 Prøving og feiling

I analysen av samtalane til Oliver og Alex (par 1) kom det fram at dei viste teikn til *operasjonell* og *strukturell forståing* for nokre av konsepta dei arbeidde med. Elevparet hadde tilsynelatande god kontroll på algoritmane og prosessane som låg til grunn for kvadratet og sirkelen dei tok føre seg, noko som er ein sterk indikator på at gutane har operasjonell forståing for desse konsepta. Likevel viste det seg at det var plasseringa kvadratet og sirkelen i forhold til kvarandre som var utfordrande for elevparet. Mot slutten av arbeidsprosessen såg det likevel ut til at det gjekk opp eit lys for gutane, då dei greidde å programmere figuren riktig. Ein kan ikkje med sikkerheit påpeike kva som er grunnen til at det gjekk opp eit lys for gutane, men det kan sjå ut som at moglegheita til å prøve og feile har hjelpt dei på vegen.

Framgangsmåten til gutane ser ut til å samsvare godt med Hana (2014) sin beskriving av kva som stort sett er den mest effektive forma for prøving og feiling. Han peikar på at det som regel er mest effektivt å prøve ut nokre få tilfelle og tenkje grundig gjennom kva som skjer i desse. I analysen kjem det fram at Alex og Oliver viser fleire teikn til at dei tenkte nøye gjennom dei ulike kodane dei prøvde ut, og dei kjem med fleire fornuftige forslag til kva kodar som kan fungere. Det verkar som at prøving og feiling førte til at Oliver såg ein liten samanheng (rotere 45 grader mot venstre) som gjorde at dei greidde å løyse oppgåva. Han gjekk nemleg tilbake til ei kode dei allereie hadde prøvd ut og gjorde ei lita endring før han køyrte programmet på nytt. Dette kan bety at moglegheita til å prøve og feile i Python har bidrege til utvikling av gutane si forståing for konseptet «kvadrat innskrive i sirkel». Den *operasjonelle forståinga* ser ut til å utvikle seg ved gutane blir merksame på eit ekstra steg som må til for å programmere figuren. Det kan sjå ut som at den *strukturelle forståinga* blir utvikla på same grunnlag. Ved å sjå dette ekstra steget verkar det som at det skjer eit «momentant kvantesprang» hos gutane, der dei ser noko kjent i eit nytt lys. Det kan dermed sjå ut som at konseptet «kvadrat innskrive i sirkel» har blitt *tingleggjort*, noko som betyr at gutane er på steg 3 i modellen til Sfard (1991). Vidare kan dette bety at gutane er i stand til behandle det matematiske objektet (kvadrat innskrive i sirkel) på eit høgare nivå enn det dei var før dei løyste oppgåva.

I Silje og Maria (par 2) sin prosess med å lage to formlike kvadrat ser det ut som at dei hadde både positiv og negativ effekt av prøving og feiling. Då dei skulle programmere det andre kvadratet verka det som at moglegheita til prøving og feiling gjorde at dei slutta å tenkje, sidan dei sa ting som: «Vi tar den berre på sann *110*, eg veit ikkje om det går». Arbeidsprosessen deira ser i dette tilfellet ut til å vere prega av rein prøving og feiling slik som Hana (2014, s. 218) beskriv det. Han seier at det er tilfeldig om ein finn svaret med ei slik tilnærming. Sjølv om jentene tilsynelatande har ei rein prøving og feiling tilnærming når dei skal lage det andre kvadratet, kan det tenkjast at det har kome noko positivt ut av dette. Det kan sjå ut som at det er moglegheita til å prøve og feile som gjer at jentene held ut med oppgåva. Det å halde ut og prøve igjen er ein arbeidsmåte som står sentralt i algoritmisk tenking (Utdanningsdirektoratet, 2019a). Dei held liv i problemløysingsprosessen, sjølv om dei seier at dei «ikkje veit kva som går eller ikkje». I Python kan dei prøve ut så mykje dei vil og heile tida få ein ny respons på det dei gjer. Til slutt ser jentene at prøving og feiling som metode, ikkje fungerer. Det er fyrst når jentene stoppar opp, og tenkjer gjennom det dei har skrive i løkka, at dei ser kva som må endrast. Det er uklart om jentene ser kva som må endrast i løkka fordi dei har prøva og feila, eller om dei kunne ha oppdaga dette utan prøving og feiling. Likevel kan det tenkjast at moglegheita til

prøving og feiling har vore med på å holde liv i problemløysingsprosessen, og dermed spelt ei rolle i jentene si utvikling av *operasjonell* og *strukturell forståing*.

I arbeidsprosessen til Anette og Else kan det sjå ut som at prøving og feiling hadde ein meir negativ effekt. Dette kom til syne då jentene skulle prøve å finne lengda for diagonalen i kvadratet. Etter at dei såg at 400 var feil byrja dei tilsynelatande med systematisk prøving og feiling, slik som beskrive av Hana (2014, s. 218). Han seier at gjennom ein slik tilnærming studerer ein tidlegare forsøk og kjem nærmare ei løysing for kvar gong ein prøver noko nytt. I Anette og Else sitt tilfelle ville dei sannsynlegvis fått eit tilnærma riktig svar dersom dei fortsetta å prøve og feile, men dei ville truleg oversett Pytagoras' setning. Dei vart difor avbrotne av student, sidan eit av måla med oppgåva var at elevane skulle ta i bruk Pytagoras' setning. Else og Anette sin arbeidsprosess med prøving og feiling kan sjåast i samanheng med Kaufmann og Stenseth (2020) sin studie der det viste seg at prøving og feiling kan ha ei negativ verknad på elevar sine matematiske argument. I studien oppdaga dei at moglegheita til å prøve og feile kan ha gjort terskelen for å prøve nye ting lågare, noko som kan ha ført til at elevane ikkje kjenner eit behov for å argumentere. Det kan sjå ut som at det skjer noko liknande hos Anette og Else, sidan det verkar som at dei ikkje kjenner eit behov for å diskutere og tenkje gjennom dei ulike kodane dei skriv inn. Dei ser truleg at dei kan kome fram til eit riktig svar gjennom prøving og feiling, noko som kan føre til at lærerike diskusjonar og tankar rundt dei matematiske konsepta kan gå tapt.

5.1.2 Steg-for-steg

Komponenten *algoritmisk tenking* har vist seg i store delar av analysen. Utdanningsdirektoratet (2019a) seier at det å tenkje algoritmisk handlar om å vurdere kva for nokre steg som skal til for å løyse eit problem, og å kunne bruke sin teknologiske kompetanse for å få ei datamaskin til å løyse heile eller delar av problemet. Haraldsrud et al (2020, s. 120) peikar på at ein av fordelane med å programmere med skilpadde i Python er at ein må gjennomgå kvart teiknetrinn eksplisitt. For at elevpara i studien skal kunne programmere figurane som oppgåvene ber om, må dei gå gjennom kvart enkelt steg som skal til for å lage figuren.

Det at ein må gå gjennom kvart steg kan sjåast i samanheng med utviklinga av *operasjonell forståing*. Sfard (1991, s.10) peikar på at grundig innsikt i prosessane som ligg til grunn for matematiske konsept kan vere viktig for å forstå konseptet. I elevane sitt arbeid med å programmere dei geometriske figurane kan dei ikkje unngå å gå gjennom prosessane og

algoritmane som ligg til grunn for figurane. Det er difor rimeleg å påstå at programmering kan vere med på å utvikle elevane si *operasjonelle forståing* for konsepta dei tek føre seg. For å underbyggje denne påstanden, skal vi sjå på eit døme frå analysen. I arbeidsprosessen til Maria og Silje (par 2) blir dei kjent med korleis dei skal teikne eit kvadrat i Python, på to ulike måtar. Jentene har tilsynelatande *operasjonell forståing* for kvadrat, då dei lagar det fyrste kvadratet utan store problem. Det ser likevel ut som at dei ikkje greier å anvende denne forståinga når dei skal lage eit nytt kvadrat ved hjelp av løkker. Etter gjentatt prøving og feiling greier jentene å løyse oppgåva, og dei påpeikar sjølv kva som har blitt gjort feil. Det kan dermed sjå ut som dei har blitt enda betre kjent med prosessane som ligg til grunn for ein kvadrat, noko som kan bety at dei har utvikla sin *operasjonelle forståing* for konseptet.

I analysen vart det identifisert teikn til at elevpar 1 og 2 utvikla noko *strukturell forståing* for konsepta dei tok føre seg, og det kan sjå ut som at det å løyse oppgåvene steg-for-steg kan ha bidrege til denne utviklinga. I fylgje Haraldsrud et al. (2020, s. 120) er det enkelt å teste ut og sjekke om ein har gjort riktig når ein programmerer med skilpadder i Python. Gjennom analysen viste det seg at alle elevpara i fleire tilfelle utnytta moglegheita til å køyre programmet undervegs, og dermed kontrollere om kodane førte til det utfallet dei hadde sett føre seg. Ved å arbeide på denne måten får elevane respons på arbeidet sitt gjennom heile prosessen, og har dermed moglegheit til å gjere justeringar i kvart enkelt steg. Det skal igjen bli trekt fram eit døme frå elevpar 2 (Silje og Maria). I analysen kjem det tidleg fram at jentene er usikre på kva ordet «formlike» inneber. Likevel set jentene i verk prosessar som skal til for å lage to kvadrat. Arbeidsprosessen til jentene kan sjåast i samanheng med modellen til Sfard (1991), som skal vise vegen frå rekneoperasjonar til abstrakte objekt. Jentene blir gradvis betre kjende med operasjonane som ligg til grunn for dei ulike kvadrata. Samstundes ser jentene ut til å opparbeide seg eit bilete av korleis det endelege produktet skal sjå ut. Det at jentene gradvis opparbeider seg eit mentalt bilete kan bety at dei utviklar sin *strukturelle forståing* for formlike kvadrat.

Oppsummeringsvis kan ein seie at det er fleire faktorar i analysen som tyder sterkt på at å arbeide med oppgåver steg-for-steg i Python kan bidra til utvikling av både *operasjonell* og *strukturell forståing* for geometriske konsept.

5.2 Parprogrammering

I analysen har samtalane til dei tre elevpara vore prega av samarbeid der begge elevane har delteke. I ein studie av Herheim og Krumsvik (2011) tek dei for seg kommunikasjonsmønster som kan identifiserast hos elevpar som samarbeider godt føre ein PC. Eit kommunikasjonsmønster som dei løftar fram er bruken av pronomen som «vi» og «oss». Dei hevdar at bruken av slike pronomen gjer at elevane fungera som ei eining og ikkje som to individuelle elevar. I samtaleutdraga i analysen kan ein sjå at alle tre elevpara brukar pronomenet «vi» fleire gongar, noko som kan vere ein indikator på at dei samarbeider ved å fungere som ei eining.

Eit anna kommunikasjonsmønster som Herheim og Krumsvik (2011) har identifisert hos samarbeidande elevpar, er at elevane snakkar i kor og byggjer vidare på kvarandre sine setningar. Dette kommunikasjonsmønsteret kan ein òg sjå i samtaleutdraga til dei tre elevpara i analysen. Vi skal sjå på eit døme frå det fyrste samtaleutdraget til Alex og Oliver (par 1):

Alex:	Nei, men vist vi tenkjer på diagonalen sant, du kan jo sjå at, her har du... diameteren er jo...
Begge:	Diagonalen.

I dette utdraget fullfører gutane saman Alex si setning. I fylgje Herheim (2011, s. 65) så klarar ein ikkje å fullføre den andre si setning på ein fornuftig måte utan å ha både lytta til den andre, og i tillegg forstått mykje av korleis den andre tenkjer. Ein kan difor anta at Oliver har forstått korleis Alex tenkjer i dette tilfellet. Grunnen til at samtalekvalitetane blir løfta fram er for å belyse samarbeidet til elevane. I analysen av samtaleutdraga til dei tre elevpara kan det nemleg sjå ut som at samarbeid kan vere avgjerande når elevane skal utvikle *strukturell* og *operasjonell forståing* for geometriske konsept ved hjelp av Python. Dette gjeld både når elevane skal finne ut korleis dei skal gå fram for å lage dei geometriske figurane, og når dei skal skrive sjølve kodane som trengst for å lage figuren.

Når ein programmerer må kodelinjene ha riktig syntaks for at programmet skal fungere på riktig måte. I analysen kom det fram at dei tre elevpara hadde større eller mindre problem med å skrive kodane riktig. I fylgje Haraldsrud et al. (2020, s. 102) så kan mykje av hjernekapasiteten gå med på å skrive riktig syntaks, noko som kan føre til at det faglege (matematikken) kan oppta ein mindre del av hjernekapasiteten til elevane. Ei mogleg løysing på dette problemet kan vere

å la elevane programmere i par. Nygård (2018, s. 46) hevdar at å programmere i par kan føre til færre feil og økt effektivitet. Dette er noko som òg har blitt oppdaga i analysen i denne studien. Dersom ein ser på utdrag 2 (4.1.2) til Alex og Oliver så kan ein sjå at Alex les gjennom kodelinja og rettar på syntaksen, medan Oliver skriv. Dette gjer at dei kan oppretthalde fokuset på matematikken, og ikkje la syntaksen ta all fokus. Dermed ligg det betre til rette for at elevane kan utvikle matematisk forståing for dei geometriske konsepta dei jobbar med. Elevpar 2 og 3 greidde òg å rette eigne feil i nokre tilfelle. I andre tilfelle der elevane brukte for mykje tid på syntaksen, fekk dei hjelp med student slik at dei kunne ha hovudfokuset på matematikk.

Ein annan fordel med å programmere i par kan vere at ein er to hovud som jobbar saman om å løyse ulike problem. Av analysen kan ein sjå at ulike elevar oppdagar ulike ting. Til dømes, var det Oliver i gruppe 1 som oppdaga at dei måtte rotere pila ved hjelp av koda *left (45)* for å plassere sirkelen riktig i forhold til kvadratet. Sjølv om det ikkje er Alex som oppdagar dette, går det tilsynelatande opp eit lys for han òg. Dette kan bety at Alex har utvikla sin forståing med utgangspunkt i ein samanheng som Oliver oppdaga. Det er mogleg at Alex hadde kome til å oppdaga denne samanhengen på eigenhand, men det kan òg tenkjast at han ikkje hadde gjort det. Sett frå ein annan side, er det mogleg at Oliver ikkje hadde sett denne samanhengen dersom han ikkje hadde samarbeida med Alex. Alex har mellom anna vore ein viktig bidragsytar i arbeidet med å skrive riktig syntaks, noko som kan ha bidrege til at Oliver har halde fokuset på matematikken og dermed hatt eit betre utgangspunkt for å oppdage endringa som måtte til i den geometriske figuren. Det kan difor sjå ut som at gutane sitt samarbeid kan ha vore ein avgjerande faktor for deira utvikling av *strukturell* og *operasjonell forståing* for geometri.

5.3 Konsekvensar for matematikkundervisning

Den pågåande utviklinga av vårt teknologirike samfunn stiller krav til kva kompetansar som blir viktige i framtida. Eit av hovudargumenta for å integrere programmering i matematikkfaget er at det kan kombinerast med opplæringa av algoritmisk tenking som blir sett på som ein problemløysingsstrategi som er ein viktig kompetanse for framtida (Udirbloggen, 2017; Utdanningsdirektoratet, 2020). I fylgje Nygård (2018, s. 7) kan algoritmisk tenking omfatte alle stega frå å føresjå og analysere kva eit program skal gjere, til å kjenne att mønster, eksperimentere og evaluere moglege løysingar. Ein kan med dette sjå at matematikk og algoritmisk tenking har ein sentral verdi i opplæringa av programmering. Forsking viser at programmering kan òg ha ein sentral verdi i opplæringa av matematikk: Forsström og

Kaufmann (2018) hevdar at det å integrere programmering i matematikkundervisinga i visse tilfelle kan vere med på å motivere elevane og heve prestasjonane deira i matematikk. Popat og Starkey (2019) seier at det å lære seg og kode på skulen kan bidra til at elevane styrkar sine sosiale ferdigheiter og blir betre på matematisk problemløysing. I ein studie av Jackson et al. (2019) fann dei ut at programmering ved bruk av skilpadde kan bidra til at ein dannar seg mentale bilete og dermed utviklar evne til abstraksjon. Husain et al. (2017) sin studie viser at det å lære seg programmering kan blant anna bidra til at elevar utviklar forståing av matematiske konsept.

I dette masterprosjektet har det blitt oppdaga at utforsking av geometri i programmeringsspråket kan ha ein positiv innverknad på elevar sin utvikling av *strukturell* og *operasjonell forståing* av geometriske konsept. Alle dei tre elevpara som har delteke i studien har vist teikn til utvikling av ein eller to av forståingane. Elevpar 3 viste teikn til at dei utvikla operasjonell forståing, medan elevpar 1 og 2 såg ut til å utvikle både operasjonell og strukturell forståing for dei geometriske konsepta dei tok føre seg. I fylgje Sfard (1991) treng ein begge dei matematiske forståingane for å kunne få ein djup forståing av matematikk. Ho seier òg at forståingane er to uskiljande, men samstundes ulike, sider av same sak. Ein mogleg grunn til at elevpar 3 ikkje såg ut til å utvikle sin strukturelle forståing, kan vere at dei tilsynelatande hadde tilstrekkeleg av denne forståinga frå før. Tidleg i dialogen gav begge jentene i elevpar 3 uttrykk for at dei greidde å sjå føre seg objektet oppgåva bad om, og ein kan dermed anta at jentene allereie hadde ei strukturell forståing for objektet. Det såg ut som at dei hadde eit godt utgangspunkt for å utvikle sin operasjonelle forståing ved å ta i bruk den strukturelle forståinga. Dette kan vere eit døme på at dei to forståingane er uskiljande og samstundes ulike sider av same sak. I elevpar 1 og 2 sit tilfelle, kan det sjå ut som at dei har fått ei litt djupare forståing for dei geometriske konsepta dei tok føre seg, sidan begge para viste teikn til utvikling av både operasjonell og strukturell forståing. Det kan òg argumenterast for at elevpar 3 kan ha oppnådd ei djupare forståing for konsepta dei tok føre seg, sidan dei tok i bruk både sin operasjonelle og strukturelle forståing for å løyse oppgåva.

Ein kan stille spørsmål til om elevane i denne studien ville ha oppnådd akkurat same utvikling dersom dei hadde mottatt tradisjonell undervisning av dei same geometriske konsepta. Er det eit poeng i at elevane skal lære dette gjennom programmering i Python? Har Python ein verdi for at elevane skal forstå matematikk betre? I denne studien viser dei to funna som er beskrive i kapittel 5.1.1 og 5.1.2 at nokre av moglegheitene som finst i Python kan bidra til at elevane

får ein annan tilnærming til matematikken som kan gjere at dei ser samanhengar som dei kanskje ikkje hadde sett elles. I kapittel 5.1.1 vart det løfta fram korleis moglegheita til å prøve og feile i Python kan bidra til at elevane ser nye samanhengar mellom eigenskapane i geometriske figurar. I kapittel 5.1.2 vart det diskutert korleis det å gjere oppgåver steg-for-steg kan gjere at ein får grundigare innsikt i prosessane som ligg til grunn for eit geometrisk konsept. Poenget her er ikkje å peike på programmering i Python som eit betre undervisingsverktøy enn andre verktøy. Poenget er at ulike verktøy byr på ulike moglegheiter, og at Python byr på nokre unike moglegheiter som kan bidra til at elevar utviklar *strukturell* og *operasjonell forståing* for geometriske konsept.

6 Avslutning

I dette masterprosjektet har det blitt forsøkt å svare på fylgjande problemstilling:

Korleis kan elevar utvikle strukturell og operasjonell forståing av geometri gjennom utforsking i programmeringsspråket Python?

For å svare på problemstillinga har det blitt gjennomført kvalitativ forsking der tre elevpar på 10.trinn har vore i fokus. Elevpara har fått tildelt geometrioppgåver av ulik vanskegrad som dei har arbeida med i programmeringsspråket Python (sjå Vedlegg I). For at elevane skulle ha ei utforskande tilnærming til geometrioppgåvene, vart oppgåvene opna opp slik at elevane sjølve måtte finne ut kva for ein strategi dei ville bruke. Nokre av oppgåvene opna i tillegg opp for at fleire svar kunne vere riktig. Samstundes hadde oppgåvene eit klart mål, slik at elevane skulle ha noko konkret å jobbe mot. Medan elevane arbeidde med oppgåver på PC vart det gjort skjerm- og lydopptak som seinare vart transkribert. Transkripsjonen danna grunnlaget for å kunne analysere arbeidet til elevane.

For å analysere arbeidsprosessen til elevane vart det laga eit nytt samansett rammeverk som består av fylgjande komponentar: *Tankegangskompetanse*, *Problembehandlingskompetanse*, *Algoritmisk tenking* og *Utforsking*. Analysen som baserte seg på dei fire komponentane var med på å danne eit grunnlag for å kunne analysere vidare om elevane viste teikn til *operasjonell* og *strukturell forståing* for dei geometriske konsepta dei tok føre seg. Det vart òg analysert om dei hadde forståing frå før, eller om dei utvikla forståing gjennom arbeidet med oppgåvene dei fekk tildelt.

Ved hjelp av det samansette rammeverket og analysen har det blitt trekt fram nokre funn som viser kva verdi programmering i Python kan ha for utviklinga av elevane sin *strukturelle* og *operasjonelle forståing* for geometri. Prøving og feiling har blitt løfta fram som ein moglegheit i Python som kan bidra til at elevane utviklar dei to forståingane ved at dei oppdagar nye samanhengar i geometrien. Likevel bør ein vere merksam på at tilfeldig prøving og feiling kan vere lite lærerikt, sidan det kan vere tilfeldig om elevane kjem fram til riktig svar. Ein annan verdi som har blitt identifisert i Python med skilpadde er at ein må løyse oppgåvene steg-for-steg. Det å måtte gjere kvart teiknetrinn eksplisitt med skilpadda kan føre til at elevane får ei grundigare innsikt i prosessane som ligg til grunn for eit geometrisk konsept. I fylgje Sfard

(1991) kan grundig innsikt i prosessane som ligg til grunn for matematiske konsept vere viktig for å forstå konseptet i sin heilskap.

Fordelande med å vere to elevar som programmerer saman har òg blitt vektlagt i oppgåva. Å vere to elevar som samarbeider om kodane kan i fylgje Nygård (2018) bidra til at ein gjer færre feil. Dette har òg blitt oppdaga i denne studien. Elevane har hjelpt kvarandre med å skrive riktig syntaks, noko som har gjeve meir rom for å fokusere på matematikken. Når det kjem til sjølve matematikken har elevane òg vist teikn til at det er ein fordel å vere to. Dei ulike elevane oppdaga ulike samanhengar, noko som tilsynelatande har vore ein viktig faktor i elevane sin utvikling av *operasjonell* og *strukturell forståing* for dei geometriske konsepta dei tok føre seg.

6.1 Vegen vidare

I denne studien var eit og eit elevpar til stades då dei skulle programmere i Python, noko som inneber at eg som student var disponibel til ein kvar tid dersom elevane stod fast på noko. Dei tre elevpara er svært ulike, både med tanke på tidlegare erfaring med programmering og med tanke på presteringar i matematikkfaget. Dette kan ha bidrege til at eg har fått eit djupare innblikk i korleis utforsking og oppgåveløysing i Python kan bidra til at elevar utviklar *operasjonell* og *strukturell forståing* for geometri. Likevel er dette ein nokså kunstig setting med tanke på at ein vanlegvis vil vere fleire elevar til stades i eit klasserom. Det kunne difor ha vore interessant å gjennomføre eit liknande forskingsprosjekt i full klasse, og undersøke om elevane hadde fått same læringsutbytte med tanke på utvikling av *strukturell* og *operasjonell forståing*. I ein slik klasseromsituasjon kan det truleg oppstå fleire uventa situasjonar. Det kunne difor òg ha vore interessant å undersøke læraren si rolle i ein slik setting. Ein kunne dermed ha undersøkt om læraren opplever at ho er tilstrekkeleg i klasserommet både med tanke på eigne programmeringskunnskapar, og med tanke på om ho får hjelpt elevane like mykje som ho sjølv tenkjer er naudsynt.

Denne studien og tidlegare forsking viser at programmering kan potensielt bidra til mykje læring i matematikk. I åra framover blir det interessant å sjå om lærarar greier å realisere dette potensialet når programmering skal implementerast i matematikkundervisinga.

Litteraturliste

Andersen, H. P., Fiskum, T. A. & Rosenlund, M. R. (2018). Hva menes med undrende, utforskende og aktiviserende undervisning? I T. A. Fiskum, D. Gulaker & H. P. Andersen (Red.), *Den engasjerte eleven. Undrende, utforskende og aktiviserende undervisning i skolen* (s. 17-29). Oslo: Cappelen Damm Akademisk.

Artigue, M. & Blomhøj, M. (2013). Conceptualizing inquiry-based education in mathematics. *ZDM : The International Journal on Mathematics Education, 45*(6), 797-810. https://doi.org/10.1007/s11858-013-0506-6

Befring, E. (2015). *Forskningsmetoder i utdanningsvitenskap*. Oslo: Cappelen Damm akademisk.

Bergem, O. K. (2016). VI KAN LYKKES I REALFAG – VIKTIGE FUNN FRA TIMSS 2015. I O. K. Bergem, H. Kaarstein & T. Nilsen (Red.), *VI KAN LYKKES I REALFAG. Resultater og analyser fra TIMSS 2015* (s. 173-177). Scandinavian University Press (Universitetsforlaget).

Bergem, O. K., Kaarstein, H. & Nilsen, T. (2016). TIMSS 2015. I O. K. Bergem, H. Kaarstein & T. Nilsen (Red.), *VI KAN LYKKES I REALFAG. Resultater og analyser fra TIMSS 2015* (s. 11-20). Scandinavian University Press (Universitetsforlaget).

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

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research, 17*(12), 18-32. https://doi.org/https://doi.org/10.26803/ijlter.17.12.2

Fuglestad, A. B. (2010). Inquiry into mathematics teaching with ICT. I B. B. Sriraman, Christer, S. Goodchild, G. Pálsdóttir, B. Dahl & L. Haapasalo (Red.), *The first sourcebook on nordic research in mathematics education: Norway, Sweden, Iceland, Denmark, and contributions from Finland.* (s. 91-108). Charlotte, N.C.: Information Age Publishing.

Hana, G. M. (2013). *Matematiske byggesteiner*. Bergen: Caspar.

Hana, G. M. (2014). *Matematiske tenkemåter*. Bergen: Caspar forl.

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

Herheim, R. (2011). Kommunikasjonsmønster. *Tangenten - tidsskrift for matematikkundervisning, 22*(2), 63-66.

Herheim, R. & Krumsvik, R. J. (2011). Verbal communication at a stand-alone computer. *Journal for educational research online, 3*(1), 29.

Hiebert, J. & Lefevre, P. (1986). Conceptual and Procedural Knowledge in Mathematics: an Introductory Analysis. I J. Hiebert (Red.), *Conceptual and procedural knowledge: The case of mathematics* (s. 1-27). New York: Routledge.

Husain, H., Kamal, N., Ibrahim, M. F., Huddin, A. B. & Alim, A. A. (2017). Engendering problem solving skills and mathematical knowledge via programming. *Journal of Engineering Science and Technology, 12*(12), 1-11.

Jackson, J. L., Stenger, C. L., Jerkins, J. A. & Terwilliger, M. G. (2019). Improving abstraction through Python programming in undergraduate computer science and math classes. *Journal of Computing Sciences in Colleges, 35*(2), 39-47.

Kaufmann, O. T. & Stenseth, B. (2020). Programming in mathematics education. *International journal of mathematical education in science and technology*, 1-20. https://doi.org/10.1080/0020739X.2020.1736349

Kvale, S. & Brinkmann, S. (2009). *Det kvalitative forskningsintervju* (2. utg., T. M. Anderssen & J. Rygge, Oms.). Oslo: Gyldendal akademisk.

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

Niss, M. (2003). Mathematical competencies and the learning of mathematics: The Danish KOM project. *3rd Mediterranean conference on mathematical education* (s. 115-124).

Niss, M. & Højgaard, T. (2019). Mathematical competencies revisited. *Educational studies in mathematics, 102*(1), 9-28. https://doi.org/10.1007/s10649-019-09903-9

Nygård, K. (2018). *Programmering i skolen : hvordan komme i gang?* Oslo: Pedlex.

Popat, S. & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education, 128*, 365-376.

Postholm, M. B. (2005). Observasjon som redskap i kvalitativ forskning på praksis. *Norsk pedagogisk tidsskrift, 89*(2), 146-158.

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

Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin. *Educational studies in mathematics, 22*(1), 1-36. https://doi.org/10.1007/bf00302715

Sikko, S. A. & Grimeland, B. (2020). Kritisk matematisk literacy i ein inquiry-basert kontekst på småskulesteget. *Nordisk tidsskrift for utdanning og praksis, 14*(1), 104-117. https://doi.org/10.23865/up.v14.2065

Silaparasetty, N. (u.å.). The Beginner's Guide to Python Turtle. Henta frå https://realpython.com/beginners-guide-python-turtle/#reader-comments

Skemp, R. R. (1976). Relational understanding and instrumental understanding. *Mathematics teaching, 77*(1), 20-26.

Thagaard, T. (2013). *Systematikk og innlevelse : en innføring i kvalitativ metode* (4. utg.). Bergen: Fagbokforl.

Tjora, A. H. (2021). *Kvalitative forskningsmetoder i praksis* (4. utg.). Oslo: Gyldendal.

Udirbloggen. (2017). Kjerneelementer i matematikk, men hvorfor programmering? Henta 21. september frå https://udirbloggen.no/kjerneelementer-i-matematikk-men-hvorfor-%20programmering/

Utdanningsdirektoratet. (2019a, 23. mars). Algoritmisk tenkning. Henta frå https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). *Læreplan i matematikk 1.–10. trinn (MAT01-05)*. Henta frå https://www.udir.no/lk20/mat01-05?lang=nob

Utdanningsdirektoratet. (2020, 3. september). Hva er nytt i matematikk? Henta frå https://www.udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-matematikk/

Vedlegg

Vedlegg I: Oppgåveark

Oppgåver i Python

1. Lag to formlike kvadrat.

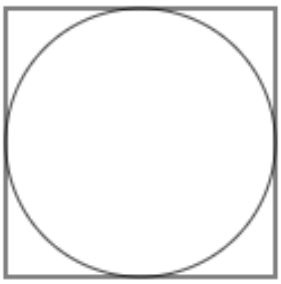
2. Lag eit kvadrat med ein diagonal.

3. Lag ein rettvinkla trekant.

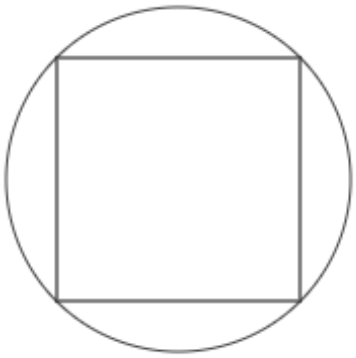
4. Lag to rettvinkla trekantar som er formlike.

5. Lag ein trekant med vinklar på 30°, 60° og 90°.

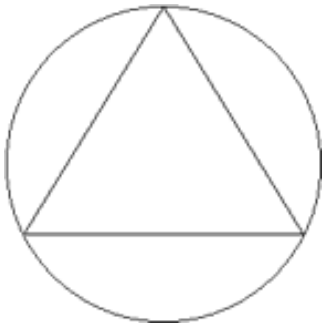
6. a) Diskuter korleis ein kan gå fram for å programmere figuren under.
b) Utforsk framgangsmåten de kom fram til.



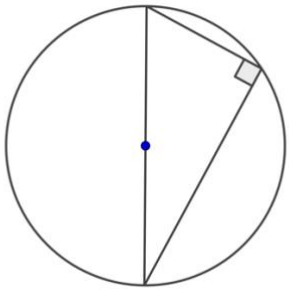
7. a) Diskuter korleis ein kan gå fram for å programmere figuren under.
b) Utforsk framgangsmåten de kom fram til.



8. a) Diskuter korleis ein kan gå fram for å programmere figuren under.
b) Utforsk framgangsmåten de kom fram til.



9. a) Diskuter korleis ein kan gå fram for å programmere figuren under.
b) Utforsk framgangsmåten de kom fram til.



10. Programmer akkurat det de vil.

Vedlegg II: Hjelpeark til elevar (i hovudøkta)

Geometri med Python - hjelpeark

Start med å skrive inn:

```
from turtle import *
from math import *
```

Koder som kan vere til hjelp:

forward ()	eksempel: forward (100)
left ()	eksempel: left (90)
right ()	eksempel: right (45)

```
lengde = 150

for count in range (4):
    forward (lengde)
    left (90)
```

```
for count in range (4):
    forward (100)
    left (90)
```

```
radius = 100
circle (radius)
```

Bruk kommandoane under for å la programmet rekne:

- Pluss: +
- Minus: -
- Gange: *
- Dele: /
- I andre: **2
- Kvadratrot: sqrt() eksempel: sqrt(81)

Vedlegg III: Intervjuguide

Intervjuguide

1. Kva kan de om programmering frå før? Har de vore bort i programmering på skulen før?

2. Korleis tenkte de då de løyste oppgåvene saman i Python?
   a) Korleis er det å samarbeide med slike oppgåver?
   b) Kva tenkjer de om vanskelegheitsgrada på oppgåvene? (Kva for nokre utfordringar dukka opp, og korleis løyste de dei?)

3. Kva tenkjer de at programmering er? Er programmering gøy?
   Frå og med neste år vil programmering vere ein del av læringsmåla til 10.klassingar. Kvifor trur dykk programmering har kome inn i skulen?

4. Kva er matematikk, er matematikk gøy?

5. Kva lærte de om matematikk no? Kva lærte de om programmering no?

6. Kva type undervisning trivast de best med? Likar de best å utforske og løyse problem, eller rekne reine oppgåver?

7. Korleis ville de ha gått fram for å lære meir om Python på eiga hand?

8. Vil de lære meir om programmering?

Vedlegg IV: Samtykkeskjema til elevar og føresette

Førespurnad om deltaking i forskingsprosjektet *"Programmering og utforsking på 10.trinn".*

Hei! Dette er eit spørsmål om elevdeltaking i eit forskingsprosjekt der formålet er å *utforske og jobbe med matematikk(geometri) ved hjelp av programmering (Python).* I dette skrivet gjev eg informasjon om måla for prosjektet og kva deltaking vil innebere.

Formål
Mitt navn er Madelen Kjøde Aarsheim. I dette forskingsprosjektet skal eg samle inn informasjon som skal brukast i mi masteroppgåve. Eg skal sjå på korleis elevar jobbar med programmering (Python) i matematikk. Alle elevane skal få ei introduksjonsøkt der dei lærer grunnleggjande om korleis dei brukar programmeringsspråket Python. Deretter vil nokre elevar bli tatt ut av klasserommet for å jobbe saman i par og løyse ulike oppgåver i geometri ved hjelp av programmering. Målet er at dei skal samarbeide og ha ei utforskande tilnærming til oppgåvene. Forskingsprosjektet vil ta til saman to til tre timer

Kven er ansvarleg for forskingsprosjektet?
HVL (Høgskulen på Vestlandet) er ansvarleg for prosjektet.

Kvifor får ditt barn spørsmål om å delta?
Sidan formålet er å forske på programmering på 10.trinn, vil alle elevane i klassa få spørsmål om å delta. Nokre av elevane som har gitt samtykke til å delta i forskingsprosjektet vil bli tatt ut av klasserommet i ein time for å løyse nokre programmeringsoppgåver saman i par.

Kva inneber det å delta?

- Deltaking inneber å løyse oppgåver i par på ein PC. Det vil bli gjort skjermopptak av alt som blir gjort på skjermen. I tillegg vil eg ta lydopptak av det elevane seier for å få eit innblikk i korleis dei samarbeider med oppgåvene. Eg kjem ikkje til å filme elevane.

- Eg vil også intervjue desse elevane etter at de har jobba med programmeringsoppgåvene. Då vil eg stille spørsmål om korleis dei tenkte då dei jobba med oppgåvene, korleis dei synast det gjekk å samarbeide om oppgåvene og om dei synast det var utfordrande, eller liknande. Det vil ikkje bli stilt spørsmål der elevane må oppgje personleg informasjon. Eg tar lydopptak og notatar frå intervjuet.

Føresette/elevar kan få sjå intervjuguide på forhand ved å ta kontakt.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Dersom du vil delta, kan du når som helst trekke samtykket tilbake utan å oppgje nokon grunn. Alle personopplysningar vil då bli sletta. Det vil ikkje føre til nokon negative konsekvensar.

Personvern – korleis vi oppbevarer og bruker opplysningar om deg
Vi behandlar opplysningane konfidensielt og i samsvar med personvernregelverket.

- Namn og kontaktopplysingar vil eg erstatte med fiktive namn som lagrast på eiga namneliste adskilt frå øvrige data. Opplysningane vil bli lagra på forskingsserveren til HVL.

Du vil ikkje kunne bli kjent att i publikasjon av masteroppgåva. Lydklipp vil berre bli brukt til intern forsking i HVL (Høgskulen på Vestlandet), og vil ikkje bli publisert. Oppgåvene elevane har løyst på PC, og skjermbilete av desse vil ikkje kunne sporast tilbake til elevane.

Kva skjer med opplysningane når vi avsluttar forskingsprosjektet?
Forskingsmaterialet blir oppbevart i HVL sin forskingsserver og sletta innan 31.07.2021.

Dine rettigheiter
Så lenge du kan bli identifiserast i datamaterialet, har du rett til:

- innsyn i kva for nokre personopplysningar som er registrerte, og å få utlevert kopi av opplysningane
- å få sletta personopplysningar, og
- å sende klage til Datatilsynet om behandlinga av personopplysningane.

Kva gjev oss rett til å behandle dine personopplysningar?
Vi behandlar dine opplysningar basert på ditt samtykke. For elevar som ikkje har fylt 15 år, må også ein føresatt gje samtykke.
På oppdrag frå HVL(Høgskulen på Vestlandet) har NSD – Norsk senter for forskningsdata AS vurdert at behandlinga av personopplysningar i dette prosjektet er i samsvar med personvernregelverket.

Kvar kan eg finne ut meir?
Dersom du har spørsmål til studien, eller ynskjer å nytte deg av dine rettigheiter, ta kontakt med:

- Rettleiar/prosjektansvarlig Inge Olav Hauge på tlf: 55585596
  eller på epost: Inge.Olav.Hauge@hvl.no
- Masterstudent Madelen Kjøde Aarsheim på tlf: 47899099 eller på epost: madelen1996@hotmail.com
- HVL sitt personvernombod: Trine Anikken Larsen (Trine.Anikken.Larsen@hvl.no)

Dersom du har spørsmål knytt til NSD si vurdering av prosjektet, kan du ta kontakt med:

- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennleg helsing

Prosjektansvarlig	*Student*
Inge Olav Hauge	Madelen Kjøde Aarsheim

Samtykkeerklæring

Eg har mottatt og forstått informasjon om prosjektet *"Programmering og utforsking på 10.trinn"* og har fått anledning til å stille spørsmål. Eg samtykker til at (kryss av):

- ☐ eg kan delta i intervju med lydopptak
- ☐ eg kan delta i oppgåveløysing på PC der det vil bli tatt skjermopptak og lydopptak.

Eg samtykker til at mine opplysningar blir lagra i HVL sin forskingsserver og sletta innan 31.07.2021.

_____ (Signatur av elev, dato)

_____ (Signatur av føresatt, dato)