Norges miljø- og biovitenskapelige universitet

**Masteroppgave 2020** **30 stp**
Fakultet for realfag og teknologi

# Kreativt resonnement i matematikk ved løsning av programmeringsoppgaver i videregående skole

Creative Mathematical Founded Reasoning in solving programming tasks in upper secondary school

Ina Ek Tveiten
Lektorutdanning i realfag

# FORORD

Denne oppgaven er det siste og avsluttende leddet i min mastergrad i Lektorutdanningen i realfag ved Norges Miljø- og Biovitenskaplige Universitet. Det har vært svært fine og lærerike år på NMBU, og veien fra Industriell økonomi til Lektorutdanningen i realfag har gitt meg en mangfoldighet jeg ikke ville vært foruten. Jeg har fordypet meg i fagene matematikk og økonomi, og valgte senere didaktiske masterfag og en didaktisk masteroppgave fordi didaktikk er spennende og tross alt det en lærer bruker mest.

Det er mange å takke etter en fin studietid. Takk til studievenner, og da vil jeg særlig takke Martine og Tone som har gjort studietiden ekstra fin. Jeg vil også takke familien og kjæresten min som har vært like støttende som de er på alle andre områder. Den som fortjener den største takken av alle er faren min som jeg har diskutert med, som har kommet med innspill og hørt på meg til ørene hans omtrent har falt av. At vi har hatt så mange og lange telefonsamtaler om masteroppgaven, og at du faktisk har lest det jeg har skrevet, har vært uvurderlig. Det er det nok ikke alle fedre som hadde tatt seg tid til i en krevende hverdag.

Takk til læreren og elevene som ville bli med på prosjektet. Jeg har opplevd samarbeidet med læreren som godt fra første stund, du har hjulpet meg med å tilpasse oppgaver og vært engasjert i prosjektet. Det er ingen sak å gjennomføre en undersøkelse med et så godt samarbeid.

Jeg vil til slutt rette en stor takk til mine tre veiledere, Margrethe, Morten og Marte, som har hjulpet meg med å få denne masteroppgaven i havn. Vi har hatt møter på Zoom og dere har vært raske til å svare når jeg har hatt spørsmål, og ikke minst har dere kommet med gode innspill!

Ina Ek Tveiten

Oslo, desember 2020

## SAMMENDRAG

Vi er nå inne i den 4. industrielle revolusjon og det er etterspurt forskning på programmering i matematikk i skolen fra forskningsmiljøet i matematikkdidaktikk. Det er også et fokus i litteratur på at det er viktig å utdanne unge mennesker til å kunne programmere. Innføringen av ny læreplan i norsk skole begynte høsten 2020, med programmering som et fokus særlig i matematikk og andre realfag. I tillegg til å kunne programmere skal elevene lære i dybden i nye læreplaner, og da er det både interessant og nyttig å se på hvordan programmering kan bidra til dybdelæring og forståelse. Kreative resonnement er et ledd i forståelsen og dybdelæringen til elevene, og det er ifølge Lithner (2015) for lite kreative resonnement i dagens skole. Det er med denne bakgrunnen både interessant og nyttig å undersøke programmeringsoppgaver og hvordan disse kan bidra med tanke på kreative resonnement, og forskningsspørsmålet er derfor: **Hvordan kan programmeringsoppgaver bidra til kreative resonnement i matematikk?**

For å finne svar på forskningsspørsmålet benyttet jeg en kvalitativ casestudie, der jeg observerte fire elever med videokameraer i Matematikk 1T (VG1) i arbeid med matematiske programmeringsoppgaver. Videoopptakene ble transkribert og Lithner (2015) og hans kjennetegn på kreative resonnement var rammeverket for å kode og analysere transkripsjonen.

Resultatene viste at programmeringsoppgaver kan bidra til kreative resonnement på mange måter: Det oppstod kreative resonnement både ved programmering av egne kodesekvenser og ved åpne diskusjonsoppgaver. Felles for oppgavene som førte til kreative resonnement var at de dreide seg om matematikk, hadde flere løsninger og var naturlig nok tilpasset elevenes nivå i både matematikk og programmering. Oppgaver i programmering bør være laget med tanke på en helhet og helst med problemløsning i alle ledd. Funnene er relevante da kreative resonnement i matematikk fører til dybdelæring og forståelse.

# ABSTRACT

We are now in the 4th industrial revolution, and there is requested more research concerning programming in mathematics in school from the research environment in mathematics didactics. There is also a focus in the litterature on that there is important to educate students' programming ability. The implementation of the new curriculum in Norway started the during fall 2020, with programming as a focus in mathematics and other sciences. In the new curriculum students will also learn in depth, and it is not only interesting, but also useful to look at how programming can contribute to deep learning and understanding. Creative reasoning is a link in understanding and deep learning, and it is, according to Lithner (2015), too little reasoning in today's school. With this bakground it is useful and interesting to research how programming tasks can contribute to creative resoning and the research question is therefore: **How can programming tasks contribute to creative reasoninig in mathematics?**

To find an answer to the research question, I used a qualitative case study where I observed four students with video cameras in Matematikk 1T (a theoretical mathematics course in Norwegian High School), working with mathematical programming tasks. The recordings where transcribed and Lithner (2015) and his characteristics of creative reasoning was used as a framework for the analysis.

The results showed that programming tasks in mathematics can contribute to creative reasoning in a multiple way: Creative reasoning sequences was present during programming sessions and in open discussion tasks. The common features of the tasks that lead to creative resoning was that they focused on mathematics, had multiple solutions and that they where skill level adjusted with respect to the students skill level in mathematics and programming. Tasks in programming should be made with totality and problem solving in mind. The findings are relevant because creative reasoning leads to deep learning and understanding.

INNHOLD

# 1 INTRODUKSJON

## 1.1 Bakgrunn

Dagens samfunn er svært annerledes enn det var for bare noen tiår siden. I dag har de fleste mobiltelefon i lomma og PC i sekken, bruker teknologi stort sett hele tiden og informasjon om alt mulig er bare et tastetrykk unna. Likevel er en stor del av den unge befolkningen ikke kjent med det som ligger bak alle programmene på PC, mobil og nettbrett: I 2019 var det kun 21 % av de mellom 16 og 24 år som hadde skrevet datakode i et programmeringsspråk (SSB, 2019). Det er et økende behov for programmeringskompetanse når vi nå er inne i den fjerde industrielle revolusjon (Forsström & Kaufmann, 2018). Ved starten av mitt eget utdanningsløp ved NMBU lærte jeg å kode matematiske problemer, noe som ga meg ny innsikt i hvordan utregning med programmering foregår: Det er et poeng at de fleste tunge matematiske problemer vi har i dag, ikke kan løses bare med vanlige analytiske metoder, men må løses med programmering eller matematiske verktøy på datamaskiner.

Det har vært ulike overbevisninger om hva som er viktig for læring av matematikk i fagmiljøet gjennom tiden: Pendelen har svingt fra at det som er viktig i matematisk kompetanse er ren regnekompetanse, til forståelse, tilbake til det å regne nøyaktig og effektivt og tilbake til resonnement og problemløsning igjen på 90-tallet (Kilpatrick et al., 2001). På 2000-tallet kom det teoretisk litteratur som viste oss at kompetanse i matematikken ikke består av enten det ene eller andre, men er kompleks og består av flere deler som flettes sammen til matematikkompetansen (Kilpatrick et al., 2001). De siste årene har det vært en dreining i fagmiljøet mot at det som heter en rikere matematisk kompetanse blir verdsatt, som består av blant annet dypere forståelse (Lithner, 2017).

Vi kan også se trenden om forståelse i nye krav i skolen. Krav er blant annet at opplæringen skal sørge for fordypning og forståelse for elevene: Hele grunnskolen og

videregående opplæring får fornyede læreplaner fra høsten 2020 (UDIR, 2020), og stortingsmeldingen om den nye læreplanen het «Fag – Fordypning – Forståelse – En fornyelse av Kunnskapsløftet» (Regjeringen.no, 2015-2016). På det skrivende tidspunkt (andre halvdel av 2020 har, for eksempel i videregående skole, VG1 fått nye læreplaner og de blir gradvis innført i VG2 og VG3 (UDIR, 2020). Med nye læreplaner skal det være mer vekt på dybdelæring i undervisning, og da er sentrale begreper forståelse, anvendelse og involvering av elevene (NOU, 2015:8).

I fagfornyelsen er det flere nye elementer enn dybdelæring og forståelse som har kommet inn, og programmering blir en del av matematikken. Læreplaner med programmering som en del av kompetansemålene er på skrivende tidspunkt gjeldende i for eksempel Matematikk 1T. Der er dette ett av disse kompetansemålene:

> *(eleven skal kunne) formulere og løyse problem ved hjelp av algoritmisk tenking, ulike problemløysingsstrategiar, digitale verktøy og programmering (UDIR, n.d.)*

Det er altså et fokus på programmering i ny læreplan i Matematikk i 1T. Forskning på området er viktig fordi det er et tydelig økende behov for didaktiske betraktninger på hvordan programmering kan implementeres i matematikk i skolen: Flere forskere i akademia påpeker at det er et behov for videre forskning på dette (Forsström & Kaufmann, 2018; Fox & Farmer, 2011; Weintrop et al., 2015).

Matematikkfaget er i endring, og vi har i de siste årene utviklet effektive og kraftfulle maskiner som sammen med analytiske metoder gir oss viktige verktøy for å forstå verden rundt oss (Weintrop et al., 2015). Weintrop og kollegaer (2015) skriver at det med en dreining mot at algoritmisk tenkning i læreplaner ble viktigere, var det å ha en klar definisjon på hva begrepet algoritmisk tenkning innebærer i matematikk og de

andre realfagene relevant. Algoritmisk tenkning i denne oppgaven er basert på definisjonen til Weintrop og kollegaene (Weintrop et al., 2015), og den kan minne om UDIR sin definisjon av algoritmisk tenkning (UDIR, 2019). Algoritmisk tenkning handler om utøvelse av datamateriale, modellering og simulering, problemløsning og systemtenkning (Weintrop et al., 2015). Weintrop og kollegaer (2015) definerer programmering som en del av kategorien problemløsning som går innunder algoritmisk tenkning. Forskerne mener en av grunnene til at skolen bør utdanne elever og studenter i algoritmisk tenkning er at det man gjør i skolen blir mer lignende det forskerne i fagfeltene jobber med, noe jeg mener er viktig fordi læringen blir mer virkelighetsnær og rettet mot videre utdanning. Vi kan se en utvikling på at programmering implementeres i undervisningen rundt om i verden: Flere land har begynt å innføre programmering i større eller mindre grad i skolen og læreplaner (Csernoch, BirÓ, Máth, & Abari, 2015; Forsström & Kaufmann, 2018; Misfeldt & Ejsing-Duun, 2015; Stephens & Kadijevich, 2020). Siden det virker fornuftig å implementere algoritmisk tenkning i skolen, og programmering i større grad er en del av utdanningen verden over, er det nyttig og interessant å se på hvordan programmering kan bidra til matematikkundervisningen.

Weintrop og kollegaer (2015) skriver at programmering kan bidra til dybdelæring i matematikken, men at matematikken også kan bidra inn i programmeringen fordi det kan føles mer nyttig å programmere fenomener som er knyttet til matematiske problemer. Ifølge Dahl, Ranestad og Hole (2017) kan programmering føre til lavere motivasjon og at det sentrale i programmering er hvordan det skal gjøres, ikke hvorfor, noe de mener er et problem. De skriver: *Hvis elevene kun lærer seg regning etter fastlagte mønstre, føles det for mange som et meningsløst og virkelighetsfjernt spill med symboler. I det lange løp ødelegger dette matematikken for disse elevene. Det ødelegger deres motivasjon og deres faglige selvbilde. I dette perspektivet er ideen om å legge programmering inn i matematikkfaget særlig problematisk. Programmering dreier seg om koding av metoder, altså algoritmer*. Forfatterne mener også at programmering ikke er tilstrekkelig tilstede i dokumentene forut for innføringen, for eksempel i

Ludvigsenutvalgets rapport (NOU, 2015:8) der programmering ikke nevnes og i rapporten fra Sanne-utvalget (Sanne et al., 2016) der programmering er anbefalt som et eget fag. Slike skeptiske betraktninger viser enda tydeligere at det er viktig å finne ut hvordan programmeringsoppgaver kan føre til dybdelæring i matematikk, og da er resonnementene en måte å få et innblikk i dette på.

Matematisk resonnering spiller en stor rolle for læring i matematikk (Jeannotte & Kieran, 2017). Lithner (2015) deler resonnement inn i imitativ og kreativ resonnering. Imitativ resonnering er typisk å pugge en måte å løse en oppgave på. Dette betyr at elevene kanskje bare ser én løsningsmetode, eller ingen ved oppgaveløsning, ved for mye imitativ resonnering. Denne typen læring kalles *rote learning,* og er ikke målet med matematikkundervisning, da elevene ikke blir selvstendige og ikke får forståelse som gir dem innsikt i at konsepter i matematikken har en sammenheng. Kreativ resonnering er noe annet enn imitativ resonnering og bidrar til problemløsningskompetanse, ifølge Lithner (2015). Kreativ resonnering består av tre elementer: kreativitet, plausibilitet og matematisk forankring. Med kreativitet, det første elementet, menes nyskapning: Resonnementet skal være nytt eller gjenskapt for den som resonnerer. Plausibilitet går ut på at resonnementet skal ha plausible, rimelige, argumenter for at det er på den måten den som resonnerer mener. Sist, men ikke minst, har Lithner (2015) matematisk forankring som det siste elementet, og dette går ut på at argumentasjonen skal være forankret i de matematiske egenskapene til komponentene som er med i resonneringen. Kreativ resonnering øver elevene i problemløsning (Lithner, 2015).

## 1.2 Forskningsspørsmål

Litteraturen til Lithner (2015) om imitativt og kreativt resonnement i matematikk fungerer som et rammeverk for oppgaven, og sammen med den nye innføringen av programmering, definisjonen av algoritmisk tenkning og dybdeforståelse i norsk skole, som en motivasjon for forskningsspørsmålet. Formålet for denne masteroppgaven er å finne ut: **Hvordan kan programmeringsoppgaver bidra til kreative resonnement i**

**matematikk?** Matematikk, resonnering, programmeringsoppgaver, og samspillet mellom dem et interessant tema å studere som bidrar til et forskningsområde der praksis er i utvikling og mer forskning er etterspurt.

# 2 TEORI

I teorikapittelet blir litteratur som er relevant for forskningsspørsmålet presentert. Det er mange ting som er relevant for forskningsspørsmålet, og det første jeg går inn på er algoritmisk tenkning, i 2.1, som er sterkt knyttet til programmering og matematikk. Deretter blir problemløsning presentert (2.2), og der får leseren et innblikk i matematisk problemløsning og problemløsning i programmering. 2.3 handler om matematisk kompetanse, noe som leder til 2.4, der matematisk resonnering (imitativ og kreativ) blir presentert. Hva resonnementene har å si for forståelsen i matematikk har jeg skrevet om i 2.5, og til slutt, i 2.6, ser jeg på hva litteratur forteller meg om design av oppgaver.

## 2.1 Algoritmisk tenkning

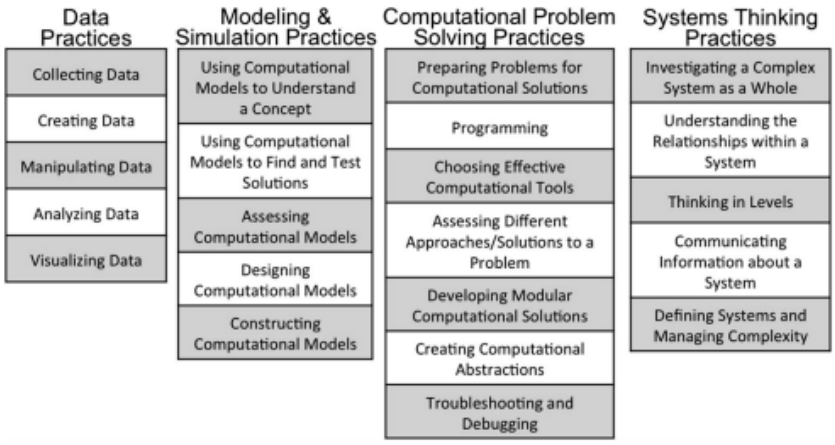
Algoritmisk tenkning blir brukt i ulike sammenhenger og er en vesentlig del av matematikken, men er også viktig i forbindelse med programmering: Alle programmer er basert på matematiske algoritmer (Lambic, 2010). Det er likevel viktig å huske på at algoritmisk tenkning er mer enn bare programmeringskompetanse (Saritepeci, 2020). En algoritme er en slags oppskrift på hvordan noe skal gjøres (Misfeldt & Ejsing-Duun, 2015). I denne oppgaven er uttrykket algoritmisk tenkning det samme som det engelske uttrykket *computational thinking*, som ikke må forveksles med det engelske *algorithmic thinking*, som er noe snevrere (Stephens & Kadijevich, 2020). Når jeg skriver algoritmisk tenkning er det altså *computational thinking* som menes. Definisjonen av algoritmisk tenkning brukt i denne oppgaven, som vi kommer tilbake til hva innebærer senere, har en del likhetstrekk ved UDIR sin definisjon av algoritmisk tenkning (2019). Dette gir mening, da bildet UDIR bruker som en illustrasjon av nøkkelbegrep og arbeidsmåter i algoritmisk tenkning (UDIR, 2019) er hentet fra Barefoot Computing UK sin illustrasjon som skal illustrere det samme i *computational thinking* (BarefootComuputingUK, 2016). Stephens & Kadijevich (2020) skriver at det finnes mange definisjoner av algoritmisk tenkning, men fellestrekkene er: Bryte opp noe i komponenter, abstraksjon, å gjenkjenne mønstre eller algoritmer og automatisering. Ifølge Kaufmann og Stenseth (2020) kan algoritmisk tenkning bli sett på som et sett med problemløsningsmetoder

som innebærer å uttrykke oppgaver og løsninger på en måte som en datamaskin kan skjønne.

Wing (2008) skriver at algoritmisk tekning har en essens i abstraksjon, og abstraksjon betyr å gjøre om noe fra konkret til abstrakt (Monaghan & Ozmantar, 2006). Et eksempel på dette er å generalisere en løsningsstrategi: Vi kan si at 2 ganger 3 er tretall plusset med hverandre to ganger, men hvis vi generaliserer det kan vi si at x ganger y rett og slett er y plusset sammen x ganger. Med denne generaliseringen kan vi bruke stykket på hvilke to tall vi vil. Det algoritmisk tenkning deler med matematikken er måten vi tilnærmer oss et problem på (Wing, 2008), og da kan vi se at det kan bidra inn i matematikkundervisning.

Weintrop og kollegaer (2015) har laget et rammeverk for algoritmisk tenkning som passer godt i denne oppgaven fordi det er en nyansert forståelse av algoritmisk tenkning og ser det i en matematisk kontekst. Den nyanserte forståelsen bidrar til å se nytteverdien av programmering i matematikk og det at det er i en matematisk kontekst er sentralt da det er programmering i matematikk denne oppgaven handler om. De deler inn algoritmisk tenkning i fire hovedkategorier: utøvelse av data, modellering og simulering, problemløsning og systemtenkning. Se figur under:

**Fig. 2** Computational thinking in mathematics and science taxonomy




**Figur 1 (Weintrop et al., 2015, p. 135)**

**Utøvelse av datamateriale** handler om hvordan data samles inn, skapes, analyseres, håndteres og visualiseres. Ifølge Weintrop og kollegaer (2015) er utøvelse av data viktig i matematikk og realfag generelt. Forfatterne mener det at data har blitt så viktig i det matematiske feltet understreker at det er nødvendig med algoritmisk tenkning, og mer spesifikt utøvelse av data, i matematikklasserom.

**Modellering og simulering** handler om å bruke algoritmiske modeller og forstå konsepter, bruke algoritmiske modeller til å finne og teste løsninger, og vurdere, designe og konstruere algoritmiske modeller. Modellering er stort sett en forenkling av virkeligheten, men selv om de er forenklinger kan de være svært nyttige. Matematiske modeller og simuleringer kan hjelpe elever med å forstå fenomener og gjør fenomenene mer tilgjengelige for elevene (Weintrop et al., 2015).

**Problemløsning** er kanskje spesielt interessant for denne oppgaven. Det går i denne sammenhengen ut på å forberede for en algoritmisk løsning, programmere, velge riktige algoritmiske verktøy, vurdere ulike tilnærminger og løsninger på problemer, utvikle algoritmiske løsninger, lage abstraksjoner og feilsøke og debugge. Problemløsning som en del av algoritmisk tenkning kan bidra til en dypere forståelse for matematiske konsepter hos elever (Weintrop et al., 2015).

Den siste kategorien av algoritmisk tenkning, **systemtenkning**, går ut på å utforske et system som en helhet, forstå forholdene innenfor et system, tenke nivåbasert, uttrykke informasjon om et system og definere et system og forholde seg til komplekse forhold. Mange matematiske problemer i dag, er komplekse, har mange variabler, utallige direkte og indirekte effekter og består av mange deler. Hvis en person jobber med slike problemer gjelder det å ha en systemtenkning, en oversikt. Det er viktig at befolkningen

utvikler systemtenkning fordi det er en økende kompleksitet i problemene som oppdages i verden (Weintrop et al., 2015).

Weintrop og kollegaenes definisjon av algoritmisk tenkning (2015) forteller oss at det er mange likheter mellom matematikk og algoritmisk tenkning, som for eksempel problemløsning, modellering og systemtenkning. Definisjonen av algoritmisk tekning som denne oppgaven baserer seg på er svært relevant inn mot dybdelæring, problemløsning og forståelse. Programmering er en type problemløsning i algoritmisk tenkning (Weintrop et al., 2015), og funn tyder på at programmering bidrar til den algoritmiske tenkningen generelt (Saritepeci, 2020).

## 2.2 Problemløsning

Matematikk og programmering blir ofte koblet gjennom problemløsning, fordi algoritmisk tenkning som tidligere nevnt er et slags sett med problemløsningsmetoder (Kaufmann & Stenseth, 2020). Ifølge Psycharis og Kallia (2017) er problemløsningskompetanse i matematikk det å kunne gi mening til ny informasjon riktig og effektivt. Kompetansen bygger altså på å kunne forstå problemet og deretter kunne løse oppgaven. Det som karakteriserer en problemløsningsoppgave i matematikk er at den ikke har en åpenbar løsning (Psycharis & Kallia, 2017). Det kan altså være flere måter å løse en problemløsningsoppgave på, og framgangsmåtene skal ikke være selvfølgelige for elevene.

George Polýa (1945) delte opp matematisk problemløsning i fire faser, der den første fasen er å forstå problemet. En sentral del ved problemløsning er altså forståelse. Når eleven har forstått oppgaven er neste fase i oppgaveløsningen å lage en plan. Her må man for eksempel eliminere muligheter, dele opp i mindre delmål, sette opp regnestykket eller tegne problemet. Deretter skal planen gjennomføres, og den skal holdes helt til den ikke fungerer. Til slutt er det viktig at eleven ser tilbake på hva han

eller hun har gjort for læring hvis det er gjort rett, eller for å finne ut hva som kan være feil hvis dette er tilfellet. Trinnene gjentas til oppgaven er løst (Polýa, 1945).

Det er likheter mellom problemløsning i programmering og matematikk, særlig er det likheter mellom programmering av problemer og Polýas problemløsningsfaser i matematikk (Govender, 2007). Det å forstå problemet, dele det opp i mindre biter, se tilbake på det som er gjort og prøving og feiling er en stor del av problemløsning i programmering (Govender, 2007), og det er som vi har sett også en stor del av problemløsning i matematikk. Ifølge Govender (2007) er problemløsning den største utfordringen ved å lære studenter å programmere, selv om studenter er klar over at det er noe av det viktigste i programmering. Forskeren har et blikk på problemløsning fordi problemer studenter opplever med programmering, som blir sett på som at de har for lite kunnskap om syntaks, kanskje heller handler om å kunne sette programmet sammen på en logisk måte (Govender, 2007).

Slik jeg forstår problemløsning er det slik at vi finner fram ulike strategier og algoritmer vi kan bruke på det matematiske problemet, lager en abstrakt løsningsstrategi og gjør det konkret til den gitte oppgaven. For å bli en god problemløser i matematikk er det viktig med kreative resonnement, som vi så i innledningen.

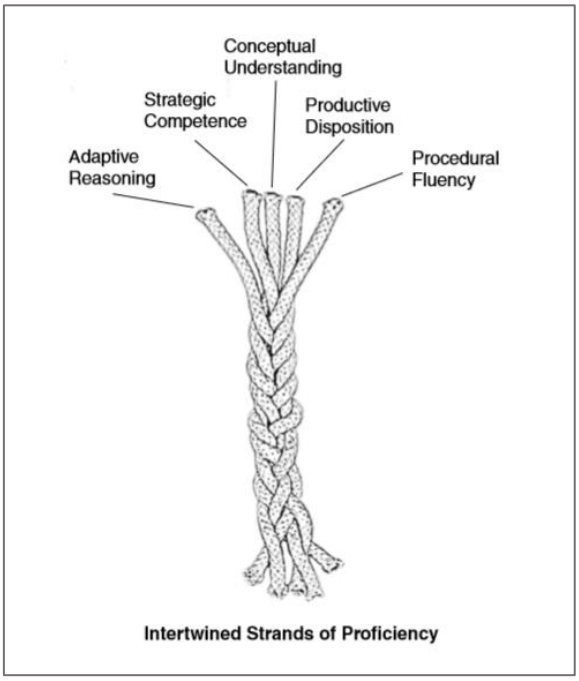
## 2.3 Matematisk kompetanse

Målet for enhver matematikktime må være at elevene lærer, og målet er derfor at de bygger matematisk kompetanse. Kilpatrick, Swafford og Findell (2001) skriver i sitt kapittel om matematisk kompetanse:

> *In this chapter, we describe the kinds of cognitive changes that we want to promote in children so that they can be successful in learning mathematics. (Kilpatrick et al., 2001, p. 116).*

Kilpatrick og kollegaer (2001) skriver at en analyse av hva slags matematikk som skal læres, litteratur i kognitiv psykologi og matematikkdidaktikk, egne erfaringer som matematikklærere og –studenter og tolkningen av hva som hva slags matematisk kunnskap, forståelse og ferdigheter som trengs har ledet dem til en definisjon av matematisk kompetanse. Den består av følgende elementer:

- **Resonnering** (adaptive reasoning) – kapasitet til logisk tenkning, refleksjon, forklaring og argumentasjon
- **Problemløsning** (strategic competence) – evnen til å formulere, representere og løse matematiske problemer
- **Forståelse** (conceptual understanding) - forståelse av matematiske konsepter, operasjoner og relasjoner
- **Engasjement** (productive disposition) – iboende tilbøyelighet til å se matematikk som fornuftig, nyttig og verdifullt, kombinert med troen på egen matematisk kompetanse og at jobbing lønner seg
- **Regneferdigheter** (procedural fluency) – ferdigheter som går ut på å kunne utføre prosedyrer fleksibelt, presist, effektivt og riktig




**Figur 2 (Kilpatrick, Swafford, & Findell, 2001, p. 117)**

Som figur 1 viser, er ikke disse elementene adskilte, men vevd sammen til den matematiske kompetansen: For å øke ens matematiske kompetanse nytter det ikke å fokusere på kun én av dem (Kilpatrick et al., 2001). Det er særlig resonnering, problemløsning og forståelse vi skal se nærmere på i denne oppgaven, da det er disse

definisjonene som er mest relatert til uttrykket kreativt resonnement, som er en del av forskningsspørsmålet.

## 2.4 Matematisk resonnement

Matematisk resonnement handler om å kunne analysere matematiske situasjoner og bygge logiske resonnement (Kaur & Toh, 2012). Resonnering kan være en tankeprosess, resultatet av tankeprosesser eller begge deler (Lithner, 2015). Resonnering, som er en av de fem delene til den matematiske kompetansen, refererer til evnen til å tenke logisk om forholdet mellom konsepter og situasjoner, er limet som holder alt i matematikken sammen og ledestjernen i matematisk læring (Kilpatrick et al., 2001).

> *Adaptive reasoning interacts with the other strands of proficiency, particularly during problem solving. Learners draw on their strategic competence to formulate and represent a problem, using heuristic approaches that may provide a solution strategy, but adaptive reasoning must take over when they are determining the legitimacy of a proposed strategy (Kilpatrick et al., 2001, pp. 130-131) .*

Utfra det forskerne skriver er matematiske resonnement et svært viktig bindeledd i de matematiske kompetanseområdene. Matematiske resonnementet til en elev er naturlig nok avhengig av den matematiske kompetansen til eleven, og matematisk resonnering er ifølge Kilpatrick, Swafford og Findell (2001) viktig for de andre delene av den matematiske kompetansen. Matematisk resonnering er altså noe som bygger matematisk kompetanse, samtidig som resonneringen avhenger av kompetansen.

Matematisk resonnering kan ifølge Lithner (2015) deles inn i to kategorier: imitativ resonnering og kreativ resonnering. Ofte bruker vi begge typer i arbeidet med

matematikkoppgaver, men hva som fokuseres på har mye å si for hva slags læringsprosesser vi opparbeider oss (Lithner, 2015).

### 2.4.1 Imitativt resonnement

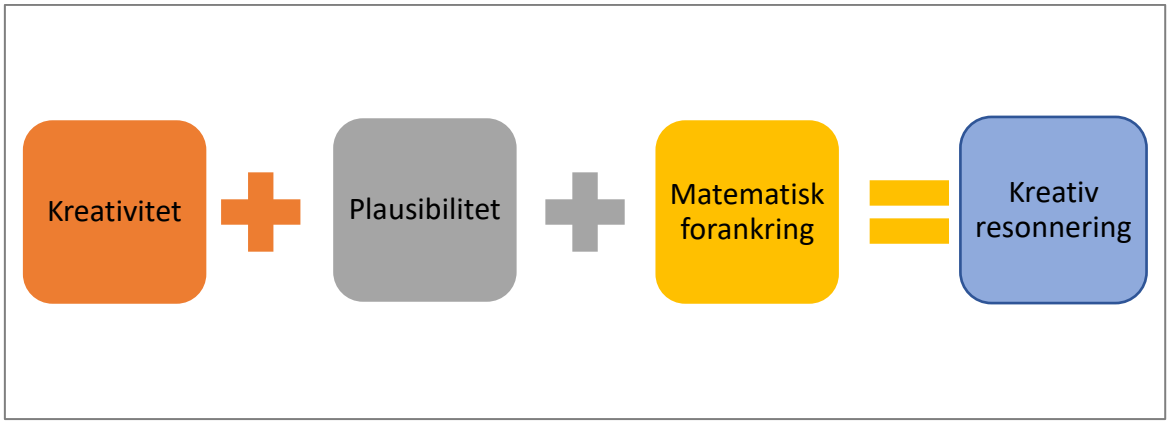
Imitativ resonnering (IR) går ut på å huske typer algoritmer eller memorere oppgaver slik at man kan løse oppgaver som er like de man har sett før. Imitativ resonnering deles inn i memorerende resonnement og algoritmisk resonnement. Memorerende resonnement er å huske en løsningsmetode for én type oppgave, mens algoritmisk resonnement er å huske en algoritme som er innlært. Algoritmisk resonnement må ikke forveksles med algoritmisk tenkning, som ble beskrevet i et tidligere kapittel. Repetitiv matematisk opplæring med fokus på å bruke algoritmer og memorering, som fører til imitativ resonnering, kan gjøre at elevene lærer på en bestemt måte. Dette kaller Lithner (2015) *rote learning*. *Rote learning* skjer når elever bare pugger og gjør like type oppgaver, og det blir da vanskelig når oppgavene er problemløsningsoppgaver, som da er noe annet enn det elevene vanligvis regner på (Lithner, 2015). Altså: Imitativ resonnering kan føre til *rote learning*. Problemløsningsoppgaver i undervisning er med på å forhindre at elever lærer på denne måten (Lithner, 2015).

### 2.4.2 Kreativt resonnement

Grunnen til at det er kreativt resonnement forskningsspørsmålet setter søkelyset på er at det, ifølge Lithner (2015), er for mye imitativ resonnering i skolen. Kreativ resonnering er den andre kategorien forskeren deler resonnement inn i, han kaller den *Creative Mathematically Founded Reasoning* (CMR), og det er det motsatte av imitativt resonnement (Lithner, 2015). Ifølge Adams og Hamm (2013) baserer kreativitet i matematikk seg på flyt, originalitet, fleksibilitet og omhyggelig utforming. Kreativitet krever en viss form for memorering og «pugging» først, men er essensielt for å forstå matematikk og forsterke glede i læringsprosessen (Szymanski, 2018). Det er altså et samspill mellom kreativ og imitativ resonnering ved løsning av matematikkoppgaver.

Lithner (2015) har tre kriterier for at et resonnement skal være kreativt: *kreativitet, plausibilitet* og *matematisk forankring*. Kreativitet går ut på at resonneringssekvensen skal være en ny eller gjenskapt, og kjennetegn er flyt og fleksibilitet. Den trenger ikke være genial eller eksepsjonell – det som er viktig er at den er ny eller gjenskapt *for den som resonnerer*. Resonneringen må også være plausibel for at den skal være kreativ, noe som betyr at den har argumenter for strategivalg og/ eller –implementering som forklarer hvorfor konklusjonen er sann eller plausibel. Matematisk forankring er det tredje og siste kravet for kreativ resonnering, og går ut på at argumentene må være forankret i matematiske egenskaper til komponentene som er med i resonneringen (Lithner, 2015).

Figuren under viser elementene som må være med i et resonnement for at det skal kalles kreativt:




**Figur 3 De 3 elementene som må være med for at et resonnement i matematikk skal være kreativt. Basert på rammeverket til (Lithner, 2015).**

Programmering kan bli sett på som å øve seg på rene algoritmer, og da er det kanskje ikke så lett å se koblingen til kreative resonnement. Problemløsningskompetanse, er som vi har sett, viktig i programmering og algoritmisk tenkning, og kreative resonnement må til i problemløsning (Lithner, 2015), og er en stor del av algoritmisk tenkning, som

tidligere sett. Forståelse krever kreative resonnement (Lithner, 2017), og mye av programmering handler om logisk oppbygning og forståelse (Govender, 2007).

## 2.5 Matematisk forståelse

Forståelse er viktig for å konstruere kunnskap (Lavy, 2006), og er en av fem elementer som skal til for å bygge den matematiske kompetansen Kilpatrick, Swafford og Findell (2001) skriver om. I denne oppgaven blir forståelse først og fremst sett på som noe som er viktig i samhandling med kreative resonnement i programmering.

Det finnes forskjellige definisjoner av forståelse i matematikk, og det er et svært komplekst konsept (Lithner, 2017). Ifølge Skemp (1978) er relasjonell forståelse i matematikk noe som er vanskeligere å oppnå enn det han kaller instrumentell forståelse. Relasjonell forståelse er å se sammenhenger, mens instrumentell forståelse er å lære seg metoder for å løse oppgaver. Det kan være at læreren underviser med tanke på en instrumentell forståelse, noe som kan være ødeleggende for elevers motivasjon, spesielt hvis elevenes mål er relasjonell forståelse (Skemp, 1978). Kilpatrick og kollegaer (2001) definerer forståelse som et integrert og funksjonelt grep om matematiske idéer som blant annet dreier seg om at tidligere lærte konsepter kobles til lærende konsepter. Både Skemp (1978) og Kilpatrick med kollegaer (2001) skriver om at det er lettere å huske riktig metode for løsning hvis man har henholdsvis relasjonell og konseptuell forståelse, og disse kan sammenlignes som en type forståelse som vi ønsker å se i dybdelæringen i skolen, som nevnt i introduksjonen.

Ifølge NCTM (2014) viser elevene forståelse i matematikk når de kan representere, diskutere og se sammenhenger i matematikken på ulike måter. Det som bidrar til argumentasjon som igjen fører til forståelse er ifølge Lavy (2006) at det er utfordring i resonnementene i oppgavene elevene får:

> *The significance of argumentation to conceptual understanding in mathematics is related to the development of students' thinking that occurs during the engagement in challenge and justification* (Lavy, 2006, p. 168).

Denne teorien underbygger argumenter til Lithner (2015) om at problemløsning er viktig for å øve kreative resonnement.

## 2.6 Design av oppgaver

Ved design av oppgaver er det viktig å tenke på at de skal legge til rette for at elevene resonnerer kreativt fordi det er målet i studien å finne ut hvordan programmering kan bidra til det. Saritepeci (2020) skriver:

> *Closely linked to creativity, CT* (algoritmisk tenkning*) support individuals' creativity, by enabling them not only to achieve a productive role affecting the society, but also avoiding being just a technology consumer. In this regard, CT* (algoritmisk tenkning) *can be considered as one of the core skills encompassing creativity as well as being a skill enabling individuals to form new expressions, by expanding the traditional ones (Saritepeci, 2020, p. 38)*

Dette betyr at algoritmisk tenkning bidrar til kreativitet. Studien til Saritepeci (2020) viser at oppgaver som fører til kreativitet også fører til bedre algoritmisk tenkning. Algoritmisk tenkning er, som vi ser i sitatet, en kjernekompetanse i kreativitet. Programmering er et verktøy til å bedre ens algoritmisk tenkning (Saritepeci, 2020). Algoritmisk tenkning blir altså et viktig aspekt ved laging av programmeringsoppgaver i matematikk.

Oppgaver i matematikk som fører til kreativ resonnering er ofte på et høyt nivå for eleven. Det er tre viktige funn i forskning angående oppgaver i matematikk de siste to tiårene ifølge NCTM (2014):

1. Ikke alle oppgaver gir den samme muligheten for elevenes tenking og læring
2. Læringsutbyttet er størst i klasserom der oppgavene oppmuntrer elevene til å tenke og resonnere på et høyt nivå og minst der det løses oppgaver av rutinemessig art
3. Oppgaver som krever mye er ofte vanskelig å implementere i undervisningen og blir omgjort til mindre krevende oppgaver

For å legge til rette for resonnering og problemløsning er det viktig å legge til rette med oppgaver som oppmuntrer til aktiv resonnering, å gjøre mening ut av det som regnes på og problemløsning for å utvikle dybdelæring i matematikk (NCTM, 2014). Ifølge NCTM (2014) er et kjennetegn ved effektiv matematikkopplæring at den oppfordrer elevene, og forsterker deres evne, til å koble matematiske representasjoner for å få dypere forståelse og som et verktøy til problemløsning.

Det kan være en utfordring å få til problemløsning, algoritmisk tenkning og kreative resonnement i matematikk ved programmering med begrensede forkunnskaper innen programmering og programmeringens logikk og syntaks (programmeringsspråkets regler). Da er det viktig å tenke på hvordan oppgaver bygges opp og hva de skal inneholde – hvordan skal elevene resonnere kreativt hvis de ikke er sikre på hvordan de skal bygge opp programmer eller sliter med syntaks? Ifølge Govender (2007) er det viktig å ha med deloppgaver slik at elevene lærer hvordan elementer i programmet skal skrives, men disse bør kobles til problemløsning.

Det er en utfordring med programmering i matematikk å holde fokus på matematikken og det matematikkvitenskapelige (Misfeldt & Ejsing-Duun, 2015). Når det kommer til programmering i matematikk finnes det flere eksempler på at det i klasserom og forelesninger kan være mer fokus på syntaks enn logikken og problemløsningen (Govender, 2007). For å få fram kreative resonnement og algoritmisk tenkning hos elever som ikke er så stødige på programmeringens syntaks og oppbygning kan det være lurt å ha innslag av andre algoritmiske oppgaver som bygger opp mot en mer kreativ problemløsningsoppgave med flere svar senere.

# 3 METODE

Forskningsmetoden har mye å si for hva slags resultater man får i forskning (Christoffersen & Johannesen, 2012). Først skal jeg presentere forskningstilnærmingen i masteroppgaven, som er kapittel 3.1, og deretter ser jeg på observasjon som metode i 3.2. For å få et innblikk i hvordan utforskingen foregikk kommer deltakere og kontekst som 3.3 og beskrivelse av og begrunnelser i oppgaveheftet som 3.4. 3.5 handler om transkripsjon og koding og analyse av denne, 3.6 om kvalitet i studien og 3.7 har med etiske betraktninger for gjennomføringen av studien.

## 3.1 Forskningstilnærming

Kvalitativ og kvantitativ metode er noe vi ofte hører i sammenheng med samfunnsfaglig forskning. Ifølge Christoffersen og Johannesen (2015) er hovedforskjellen på kvantitativ og kvalitativ samfunnsforskning hvor fleksible metodene som brukes i denne typen forskning er. Har forskeren en helling mot kvalitativ metode vil forskningen være mer fleksibel enn ved en mer kvantitativ metode. Vi kan se på et eksempel med kvantitative studier med spørreskjema: Det er begrenset hvor fleksible spørsmål i et spørreskjema kan være, derav en mindre fleksibel studie. I samfunnsforskning er det grader av kvalitativ eller kvantitativ metode, eller blanding av begge metoder, noe som vil si at det ikke trenger å være enten kvalitativt eller kvantitativt. (Christoffersen & Johannesen, 2012). Når jeg nå skal se på sammenhengen mellom forståelse og resonnementene til elevene vil det ikke passe med en lite fleksibel studie, da resonnering kan skje på mange forskjellige måter basert på deres forståelse.

I kvalitative studier er det viktigere med ord enn tall (Bryman, 2012), og det er ordene og resonnementene jeg skal se på – ordene gir meg et innblikk i hvordan de tenker. Bryman (2012) mener det er tre kjennetegn til ved å være forsker i kvalitative studier som er verdt å merke seg:

1. Et induktivt syn på forholdet mellom teori og forskning, *veien blir til mens man går.*
2. En vitenskapelig rolle som en slags oversetter, som er en annen tilnærming enn kvantitativ forskning som heller bruker eksisterende sannheter som utgangspunkt. I kvalitativ forskning ser vi den sosiale verdenen gjennom analysen av *oversettingen* av deltakernes verden i studien.
3. En ontologisk rolle som slags *teoriarkitekt*, som betyr at kjennetegn i samfunnsforskning er et resultat av interaksjoner mellom deltakerne i studien i stedet for fenomenene *der ute.* Ved kvantitativ forskning er ikke deltakerne en del av den teoretiske konstruksjonen.

Siden jeg ønsker å ha et åpent blikk, se på hva som blir sagt av elever og på denne måten oppdage nye sammenhenger mellom resonnement i matematikk og oppgaver i programmering, har jeg valgt kvalitativ tilnærming som metode i studien. Jeg tolker Bryman (2012) som at kjennetegn ved forskere i kvalitativ forskning er at de har et åpent sinn, ser verden gjennom deltakerne i studien, bygger opp generell teori gjennom det som blir oppdaget og at det er praksisnær forskning som drives.

Innenfor kvalitativ forskning finnes noe som heter kvalitative casestudier. Jeg har valgt kvalitativ casestudie fordi når jeg ønsker å se på de matematiske resonnementene til elever når de løser ulike programmeringsoppgaver, vil det være naturlig å se på noen få elever i et bestemt tilfelle da det er ordene, som en representasjon av tanker, som er i fokus. På denne måten kan jeg finne ny informasjon om hva sammenhengen mellom programmering, matematikk og resonnering er. Kvalitative casestudier studerer spesifikke tilfeller, noe som navnet kan antyde: Ordet case kommer fra det latinske casus som betyr tilfelle (Christoffersen & Johannesen, 2012). Ifølge Christoffersen og Johannesen (2012) brukes casestudier mye innenfor utdanningsforskning og er studier av noe spesifikt innenfor noen få enheter - det kan være et individ, en gruppe eller en klasse.

Man kan bruke mange typer datainnsamlingsmetoder, men felles er at kildene er tids- og stedsavhengige (Christoffersen & Johannesen, 2012).

## 3.2 Observasjon som metode

Det er ikke mulig å observere alt som skjer i en hendelse: Wadel (1991) eksemplifiserer dette med at hvis vi skulle observere alt som kunne være relevant i et pasningsmottak i en fotballkamp, måtte vi hatt med observasjon av hver eneste muskel som er med på samspillet av mottaket. I tillegg måtte vi hatt et blikk for de andre spillerne på banen (Wadel, 1991). Det er tydelig at det er mye å observere i et klasserom når *bare* et pasningsmottak kan gi så mange iakttagelser.

Wadel (1991) fortsetter eksempelet med en fotballkamp når det kommer til tolkning av en hendelse – der vil ikke tilskuerne tolke det samme utfra det som skjer på banen. Synet deres er altså farget av hvilke følelser de har for de to lagene, men tolkningene av det som ses vil også være forskjellig. Dette illustrerer han med denne figuren:


**Figur 4 (Wadel, 1991, p. 76)**

Det er altså både ved observasjon og ved tolkningene av data at vi blir farget av våre egne oppfatninger, og det har derfor vært viktig med en viss induktiv tilnærming.

Siden vi i denne studien skal se på matematisk resonnering ved programmering, var det viktig å komme tettest mulig på resonneringen når de programmerer, og siden det er usikkert hvor mye innsikt elevene hadde i egen resonnering, var det bedre for meg som

forsker å følge resonneringen selv enn å for eksempel gjennomføre et intervju. Dette er en kvalitativ studie, og det er derfor en styrke ved studien at jeg får innblikk i hva som blir sagt, noe som er vesentlig for å få et innblikk i resonnementene ved programmeringen. Ifølge Christoffersen og Johannesen (2012) egner observasjon seg som metode når forskeren ønsker direkte adgang til det som skal forskes på. I denne studien var det viktig med direkte tilgang på resonneringen.

Jeg hadde muligheten til å filme elevene og valgte å benytte den muligheten. Med observasjon med filmkamera får forskeren med et komplekst bilde av situasjonen han eller hun studerer. Videoopptakene kunne få frem flere detaljer som var vanskelige å observere i øyeblikket. Det er blitt vanlig å ta opp med videokameraer da det er det som er minst inntrengende, men som likevel gir et komplekst innblikk (Powell, Francisco, & Maher, 2003). Det fine med videoopptak er at forskeren har enda mer direkte adgang på det som skal observeres enn uten, men det er samtidig en fare for at man tar med bare de resultatene som presenterer det en ønsker å finne, ikke noe som representerer det typiske (Powell et al., 2003).

## 3.3 Deltakere og kontekst

Timen var bygget opp med en innføring av læreren ved starten av timen, to skoletimer der elevene skulle løse et oppgavesett tilpasset utforskingen så godt de fikk til, og en avslutning i siste time med læreren. I studien er det delen der deltakerne gjorde oppgavesettet som ble observert vi skal se på. Jeg lånte et rom av skolen der elevene som ble valgt ut skulle observeres av meg og filmes med to filmkameraer.

Det var viktig for meg å forske i en klasse der læreren var en god samarbeidspartner i innsamlingen. Læreren som var med i prosjektet hadde jeg et tett samarbeid med og hun er framoverlent når det gjelder programmering. Hun syntes prosjektet var spennende og hadde kompetanse i programmering og matematikk som var verdifull for

prosjektet. Klassen læreren underviser i og som er forsket på i dette prosjektet var en Matematikk 1T-klasse med cirka 30 elever. Fire deltakere ble utvalgt og spurt av denne læreren fordi det var viktig å komme tett på resonneringen, og da valgte jeg å kun ha med fire stykker. Læreren tok en vurdering av nivået på oppgavesettet og fant ut at disse elevene antakelig ville kunne resonnere matematisk og få til å være kreative kodere. På denne måten hadde jeg et godt utgangspunkt for et innblikk i de matematiske resonnementene, da læreren kjente elevene og deres faglige styrker godt. Når det gjaldt programmeringsspråk er det tydelig at forlagene bruker Python i sine bøker eller nettkurs, og da elevene allerede hadde jobbet med Python og denne syntaksen, var det naturlig at det var dette programmeringsspråket som ble brukt i utforskningen.

De fire elevene ble delt i to par, guttene ble satt sammen og jentene ble det andre paret. I denne oppgaven heter de naturlig nok noe annet enn i virkeligheten. Jentene har jeg kalt Sonja og Oda, guttene heter her Olav og Jesper. Grunnen til at jeg delte de inn i parene Sonja og Oda, og Olav og Jesper var fordi både de to jentene og de to guttene var trygge på hverandre, derfor var tanken at dette skulle få elevene til å snakke sammen, og at resonnementene på den måten kunne synliggjøres for meg som forsker. I tillegg kan samarbeidet hjelpe dem med å resonnere. Det å samarbeide om løsninger på problemløsningsoppgaver kan hjelpe elevene med å løse problemer selv senere (Adams & Hamm, 2013), men samarbeid er først og fremst et verktøy i studien – det er ikke mulig å vite hva elevene tenker uten noen form for data. I Lithners (2015) rammeverk for resonnement er den skriftlige eller muntlige dataen fra elever en representasjon av resonnementene som skjer. Det er viktig å ha med seg at med samarbeid (i par) blir resonnementene mer komplekse og interaksjonene mellom studentene kan føre til mer resonnering for hver av dem (Granberg & Olsson, 2015). Ifølge Hershkowitz og kollegaer (2017) kan det oppstå endringer i argumentene i den uttalte resonneringen når elever samarbeider. Selv om en elev ikke starter et matematisk resonnement, kan andres matematiske resonnement føre til at eleven bidrar med dette (Hershkowitz et al., 2017). Det er derfor viktig at elevene er verbale i samarbeidet.

Ifølge Granberg og Olsson (2015) bør det ikke legges vekt på å ikke dele oppgaver mellom seg for å få til et vellykket samarbeid, men at det er viktig å ha en delt forståelse på spørsmål som: Hvor skal vi, hvor er vi og hvordan kommer vi oss dit vi vil? Her er det en tydelig kobling til Polýa (1945) og hans problemløsningsteknikker, men at i et samarbeid er det viktig å dele tanker og finne ut planen, hva som har gått galt og løsninger *sammen*. Granberg og Olsson (2015) har funnet ut at hvis elevene deler kreative resonnement fører dette til en bedre læringsarena for dem begge.

## 3.4 Oppgavehefte

For at elevene skulle lære matematikk og resonnere kreativt var det viktig å lage oppgaver som først lar dem beskrive og begrunne kode og resultater av ferdige programmer som ble presentert for dem, men at de også skulle programmere egne småprogram. I oppgavene skulle elevene generalisere og utforske egenskaper ved potenser mot slutten, slik at de skulle få reflektere over sammenhenger mellom potensene, sum av tall, og se nytten av å kunne disse sammenhengene. Det var altså en oppbygning i oppgavene som baserte seg på at elevene skulle forstå problemene, resonnere matematisk og bygge matematisk kompetanse.

Ved laging av programmeringsoppgavene (vedlegg 1) var det viktig å sørge for at elevene ville resonnere matematisk, og derfor at gjennomføringen av dem krevde en sterk evne til kognitive prosesser:

> *Mathematical tasks with high cognitive demand often require students to make explicit their thinking. These tasks are necessary for the advancement of reasoning, communication and connections during lessons.*(Kaur & Toh, 2012, p. 3)

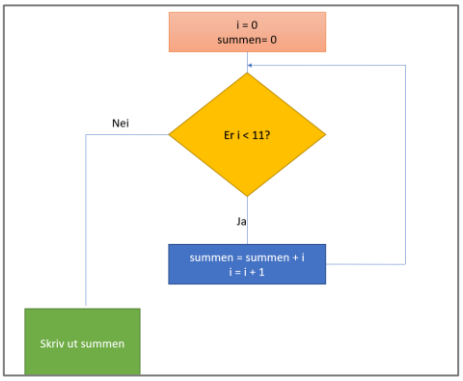
Det var altså viktig med oppgaver som ikke er rett frem å løse, men som lignet mer på problemløsningsoppgaver. Fokuset mot problemløsning og ikke syntaks er en utfordring – hvordan lage oppgaver som ikke krever utfordrende syntaks for elevene, men legger

til rette for problemløsning? For å forhindre for mye fokus på syntaks, satte jeg inn kode med et flytskjema i starten av oppgave 1 som sammen med teksten forklarte hvordan koden fungerte. Grunnen til at jeg starter med while-løkker og deretter, som vi kommer til å se, går over til for-løkker er at elevene i klassen var kjent med while-løkker, men ikke for-løkker på samme måte. Da kunne første oppgave få elevene inn i tankegangen, og at de deretter kunne bruke en for-løkke for å programmere potenser. Valget om å først bruke while-løkke også bruke for-løkke gjorde jeg altså på bakgrunn av elevenes forkunnskaper.

Dette var den første oppgaven:

### While-løkker

Under ser du en while-løkke i Python. While-løkker går igjennom runder for å finne ut ulike ting, og har en test i starten der noe (en variabel) må oppfylle visse krav for at løkka skal fortsette å kjøre. Her vil vi finne summen av tallene fra og med 1 til og med 10 (dvs.: 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10). Her har vi skrevet: «`while i < 11`», som betyr at programmet kjøres så lenge «i» er under 11. Altså: så lenge «i» er under 11, så går løkka videre til neste runde.




```
#Sum av tallene fra 1 til 10, while

summen = 0
i = 1

while i < 11:
    summen = summen + i
    i = i + 1

print(summen)
```

### Oppgave 1, while-løkke

a) Hva skjer i første runde av while-løkka?
b) Hva skjer andre runde i while-løkka? Hvordan blir «`summen`» forskjellig fra første runde?
c) Hvorfor har vi med «`i = i + 1`»? Hva hadde skjedd hvis denne ikke hadde vært med i koden, tror dere?
d) Lag en egen while-løkke som skriver ut tallene i 3 gangeren **opp til 30.** Hvor bør `print()` stå i koden? Bruk gjerne tips og triks fra koden med while-løkka som dere har diskutert nå, eller tenk nytt hvis dere ønsker!

Her skal elevene først forklare koden som er presentert og hvorfor elementer er med i denne, og i 1d skal de programmere en egen liten kodesekvens som skriver ut tallene i 3-gangen opp til 30. Denne legger opp til flere løsninger: Elevene kan plusse på tre for hver gang, eller bruke «i» (som øker med én for hver runde) og gange denne med tre.

Oppgavene legger opp til at elevene først skal diskutere seg frem til en forståelse av koden, og at de, i denne første oppgaven, etter diskusjon av de tre første deloppgavene skal programmere et lite program selv. Som presentert i teoridelen, er det et samspill mellom imitativt og kreativt resonnement (Szymanski, 2018), selv om kreativt resonnement bør være målet og fokuset i undervisning. Jeg tenker at dette samspillet er særlig viktig i programmering, fordi det programmeringstekniske naturlig nok også er en del av programmeringen, selv om det ikke bør være fokuset. Likevel har jeg strukket meg etter å legge til rette for at alle deloppgavene skal være koblet til problemløsning på en eller annen måte, et grep som ble trukket fram i teoridelen (Govender, 2007). Derfor har jeg tenkt på forståelse og algoritmisk tenkning gjennom alle oppgavene, men det er ofte de siste deloppgavene innunder hver oppgave som legger best til rette for kreativ resonnering.

I oppgave 2 forklarer jeg også en løkke i starten. Grunnen til at jeg forklarer løkkene og syntaks i starten av de to første oppgavene er elevenes begrensede forkunnskaper i programmering, og dette skal være en oppfriskning på det de har jobbet med forut for timen. Oppgave 2d er kanskje den mest utforskende deloppgaven i oppgave 2, der de skal snakke om hva slags typer utregninger man kan bruke slike løkker til, og med slike løkker er det referert til for-løkka som regner sum som blir presentert her. Oppgaven som spør hva slags matematiske utregninger man kan bruke slike løkker til har en tydelig kobling til matematikk og veldig mange løsninger fordi det er så mange måter å tolke oppgaven på, og desto flere svar på tolkningene. Dette var oppgave 2:

### For-løkker

For-løkker er en annen type løkke, men kan brukes på samme måte som while-løkker. Forskjellen på de to er at for-løkker går automatisk videre til neste element mens while-løkker kjører helt til noe er oppnådd. Vi trenger dermed ikke skrive «`i = i + 1`» for at den skal gå videre til neste element. Her vil vi vise et program som gjør det samme som while-løkka som ble vist tidligere. Vi skriver «range(11)» som forteller for-løkka at den skal starte på 0 (det gjør den som standard) og gå gjennom heltallene til og med 10. Skriver vi «range(51)» vil dette programmet regne summen av alle tall fra 0 og til og med 50. Vi kunne også skrevet «range(1,11)» som forteller oss at vi skal starte på 1 og slutte på 10. Programmet slutter altså på tallet før når vi bruker den innebygde range-funksjonen.

```
7  #sum av tallene fra 1 til 10
8
9  summen = 0
10
11 for i in range(11):
12     summen = summen +  i
13
14 print(summen)
15
```

### Oppgave 2, for-løkke

a) Forklar hva som skjer første runde i for-løkka vist over.
b) Hvis vi hadde skrevet range(1,11) hadde vi fått samme resultat, men med én runde mindre i for-løkka, hvordan er det mulig?
c) Hva skjer 2. runde i for-løkka? Hvordan blir «summen» på høyre side av likhetstegnet forskjellig til 3. runde?
d) Hvilke typer matematiske utregninger kan man bruke sånne løkker til?
e) Skriv av programmet i Python. Hva slags resultat forventer dere å få? Kjør programmet og se om det blir det dere tenkte på forhånd. Hvorfor ble det som forventet eller ikke som forventet?
f) Prøv å forklare hva som skjer hvis vi bytter ut «+ i» med «-i».
g) Gjør endring i programmet slik at det står «-i» i stedet for «+i». Skjedde det dere trodde ville skje?

I oppgave 3 skulle elevene igjen, som de også gjorde i slutten av oppgave 1, lage egen kode, og denne gangen skulle de programmere potenser:

Oppgave 3, programmere potenser med for-løkke

a) Hva skjer når eksponenten til en potens øker med 1? Dere skal lage et program som bruker for-løkker og «*» (som betyr multiplisert i Python) til å regne ut potenser. Hvor mange runder bør løkken gå?

b) Lag et program som regner ut 2^7 ved å bruke en for-løkke.

c) Generaliser programmet ved å kalle 2 for «grunntall» og 7 for «eksponent». Da kan vi bestemme øverst i programmet hva grunntallet skal være («`grunntall = 4`» for eksempel) og hva eksponenten skal være («`eksponent = 3`» for eksempel), som gir oss $4^3$. Både «`grunntall`» og «`eksponent`» kalles variabler. Da er det viktig å erstatte alle steder der grunntallet 2 står til «`grunntall`» og alle steder der 7 står til «`eksponent`», sånn at vi kan velge øverst i programmer hvilken verdi variablene skal ha.
   - Hva skjer hvis `grunntall = 10` og `eksponent = 4`. Diskuter først og sjekk deretter ved å kjøre programmet med disse tallene om dere har tenkt riktig.
   - Kan vi kjøre programmet med en eksponent som er negativ? Diskuter, prøv og forklar hva du tror skjer!
   - Hva skjer hvis «grunntall» er en brøk, for eksempel $\frac{1}{2}$ og eksponenten er 5? Hvordan kan dette uttrykkes med 2 som grunntallet i en potens?
   - Få programmet til å gi dere tallet 0,001. Hvilke grunntall kan dere bruke? Hva blir eksponenten i tilfellene?

d) Hva er likheten mellom potenser og sum når det kommer til programmering av de to matematiske konseptene i løkker?

Tanken med oppgave 3c som handler om å generalisere, var at elevene skulle utforske potenser og egenskaper til potenser ved hjelp av programmering og se nytten av å kunne ulike representasjoner av potenser. I tillegg var det et poeng i at en datamaskin kan effektivisere utregninger. Siste deloppgave i oppgave 3 skal få elevene til å reflektere over det de har jobbet med ved å se på likheten mellom sum og potenser, og det var tenkt at de skulle få oppsummere det de hadde jobbet med.

I oppgavene blir algoritmisk tenkning øvet. Weintrop og kollegaer (2015) lagde en definisjon av algoritmisk tenkning som gjenspeiler seg i oppgavene. Selv om det er tydelig at programmering, som er en del av algoritmisk tenkning som går innunder kategorien problemløsning, er noe som blir øvet i oppgaveheftet, er det flere andre deler av algoritmisk tenkning som også blir øvet. Modellering og simulering blir øvet gjennom å lage programmene, fordi elevene blant annet skal konstruere algoritmiske

modeller. Elevene bør også tolke resultater som programmene gir, noe som bidrar til å øve det å evaluere data (som er en del av algoritmisk tenkning). De må også tenke nivåmessig (systemtenkning) fordi de for eksempel i oppgave 1d, der de skal programmere 3-gangen, må ha print-funksjonen på et annet «nivå» (med et innrykk) enn i koden som blir presentert forut for oppgaven. I en programmeringsløkke må elevene uansett kunne tenke i nivåer. I flere deler av oppgaveheftet må altså en stor del av det algoritmisk tenkning innebærer brukes.

## 3.5 Transkripsjon og bearbeiding av data

Transkriberingen ble gjort av meg, og det ga mye tekst: Det ble totalt over 15 000 ord med transkripsjon fra to videofilmer (siden det var to par), og hver videofilm var cirka 1 time og 45 minutter lang. Det var også planlagt å ta opptak av skjermene til elevene, men disse videoene ble ikke spilt inn på grunn av tekniske problemer. Da jeg transkriberte fikk jeg et forhold til dataene og gjorde meg tanker underveis. Det er lurt å streke under eller skrive «OBS» når man transkriberer hvis det kommer noe som er interessant for forskningen (Nilssen, 2012) og dette gjorde jeg: Jeg satte en annen farge på det som virket interessant inn mot resonnement, programmering og matematikk, noe som lettet arbeidet med gjennomlesing og koding senere.

Jeg tok til slutt utgangspunkt i Lithners rammeverk (2015) til koding og kategorisering av transkripsjonen og markerte resonnement som var imitative og kreative med hver sin farge, for det er en viktig del å kode og kategorisere dataene før forskeren begynner analyseprosessen (Nilssen, 2012). Ved gjennomlesing var det helt klart mest kreative resonnementer i paret med Sonja og Oda. Ved en nærmere titt på transkripsjonen til den andre gruppa (Olav og Jesper), så jeg at det var det flest resonnement mot programmeringstekniske forhold og mye av det de sa gikk igjen, siden de stod noe fast. Jeg har derfor heller gjort meg generelle betraktninger om denne transkripsjonen og hva som gjorde at det var få kreative resonnement. Selv om jeg ikke gikk dypt ned i transkripsjonen med tanke på kreative resonnement, har jeg hentet ut en

samtalesekvens fra denne gruppa som kanskje illustrerer noen av utfordringene de stod overfor.

Alle resonnementene jeg mente var kreative hos jentegruppa ble satt inn i et skjema, der jeg markerte hvor jeg kunne finne de tre tegnene på at resonnementene var kreative med en begrunnelse for hvert kriterium i neste kolonne. Deretter gjorde jeg en utvelgelse av samtalesekvenser basert på hvor det tydelig var kreative resonnement, og samtalesekvenser som var typiske for transkripsjonen.

## 3.6 Kvalitet i studien

Reliabilitet og validitet er noe vi hører om i forskning, og er viktige faktorer for å si noe om kvaliteten i en studie. Reliabilitet handler om pålitelighet av data, og avhenger av måten datainnsamlingen blir gjort på (Christoffersen & Johannesen, 2012). Jeg tolker det slik at hvis metoden er godt egnet og får pålitelige data så har dataen høy reliabilitet, men hvis metoden derimot gir et feilaktig inntrykk av fenomenet i studien er det lavere reliabilitet. Bruk av filmkamera, som ble gjort i denne studien, kan styrke reliabiliteten fordi når jeg ser på resonnement får jeg det de gjør og sier gjenvist på video og kan få frem detaljer som var vanskelig å oppdage der og da. Hvis vi tenker på temaet for studien, programmering som bidrar til kreative resonnement i matematikk, er det en styrke for kvaliteten i studien å kunne gå tilbake og se situasjonen igjen, da det er viktig å observere hva elevene faktisk sier i analysen slik at jeg som forsker ikke går rett til over til tolkning, men her er selvfølgelig kode- og kategoriseringsprosessen også viktig (Nilssen, 2012). Noe som kan være en svakhet med å observere med videokamera er at elevene kan ha blitt påvirket av at videokameraene var der. Det er nemlig slik at opptak med videokamera kan føre til endret oppførsel hos de som blir filmet (Powell et al., 2003). Det at jeg filmet i to timer kan likevel gjøre at deltakerne ikke tenkte på kameraene hele tiden.

Ved denne utforskingen er det min tolkning som kommer frem. Enhver forsker har et visst fokus eller et interesseområde på forhånd, og utfra teorien man har lest, en forhåndsoppfatning om hva slags konklusjoner som studien kan resulteres i (Christoffersen & Johannesen, 2012). Christoffersen og Johannesen (2012) skriver at hvis to forskere skulle observere en klasse var kanskje den ene opptatt av hvordan læreren fungerte, mens den andre ville være opptatt av hvordan elevene oppførte seg. Det avhenger altså av forskeren hva slags datamateriale som blir hentet ut. I denne studien ville kanskje andre sett på lærerens rolle, men et forskerbestemt fokus er en naturlig del av kvalitativ forskning (Christoffersen & Johannesen, 2012). Det er en svakhet at man ikke får frem andre vinklinger på temaet og i analysen, og at kanskje resultater som kunne vært interessante «går tapt». Jeg har ikke hatt forhåndsoppfatninger om hva elevene kom til å si, og på den måten heller ikke lagt opp oppgavene mot et spesielt resultat - jeg har dermed hatt en delvis induktiv tilnærming, som jeg tidligere nevnte at er viktig i kvalitativ observasjon. Det at jeg har hatt et åpent sinn styrker validiteten

En styrke i studien er at oppgavene er laget sammen med veileder Morten Munthe som har kompetanse på dette området etter egen doktorgradsforskning, noe som øker validiteten. Validitet handler om at dataene fra studien kan representere det generelle fenomenet som skal undersøkes, men må ikke oppfattes som noe absolutt – det sier noe om kvaliteten i studien (Christoffersen & Johannesen, 2012). Det må være et generelt fenomen som ikke bare er til stede i studien. At oppgavene er laget med en som har studert matematisk programmering i skolen på doktorgradsnivå øker integriteten og kritikaliteten i studien, som ifølge Whittemore, Chase og Mandle (2016) er en del av validiteten i kvalitative studier. Integritet og kritikalitet handler om *bakgrunnskunnskap*, åpen utforskning og kritisk analyse (Whittemore, Chase, & Mandle, 2016). Validiteten kan bli mindre med tanke på at jeg bare har gått ordentlig inn i transkripsjonen til én av de to gruppene, noe som er begrunnet i at det var lite nivåtilpasset til den gruppa jeg ikke gikk dypere ned i transkripsjonen til.

## 3.7 Etiske betraktninger

I denne studien ble det benyttet et informasjonsskriv (vedlegg 2) og en samtykkeerklæring (vedlegg 3), og det ble sendt en søknad til Norsk senter for forskningsdata (NSD) om å gjennomføre studien som ble godkjent. Alle som ble filmet fikk informasjonsskrivet, en muntlig gjennomgang med det foran seg og skrev deretter under på samtykkeerklæringen. Ifølge Nilssen er det viktig at forskeren er klar over det etiske ansvaret overfor deltakerne i studien. Informert samtykke har sin hensikt å sikre at deltakerne er frivillig med og at de på en så god måte som mulig blir informert om forskningen de skal være med på (Nilssen, 2012). Elevene fikk beholde, som de naturlig nok skal, informasjonsskrivet og har all rett til å kontakte noen i prosjektet for å få slettet data. Ved transkribering har navnene til elevene *ikke* blitt skrevet ned og alle som var med er blitt anonymisert i både transkripsjoner og selvfølgelig oppgaven. Ved fullført masteroppgave vil, som beskrevet i informasjonsskrivet, all innsamlet data fra prosjektet bli slettet.

# 4 RESULTATER

Jeg vil i dette kapittelet fremstille fire sekvenser der to av dem inneholdt kreativt resonnement. Det er en gjenganger hos de to jentene at de resonnerer en del kreativt og hele tiden prøver å forstå hvordan de skal kode problemene. På veien mot forståelse og ferdig kode er det 14 kreative resonnement hos denne gruppa som inneholder de tre kriteriene for kreative resonnement.

Jeg valgte bevisst å dykke og analysere dypest hos gruppa med de to jentene, da den andre gruppa ikke fikk til like mange kreative resonnementer. Forskjell på gruppene er likevel også resultater som vi skal se nærmere på, og som viser at det kan være krevende å implementere oppgaver i matematikken som fører til kreativ resonnering med tanke på at de skal være riktig nivå. Den gruppa jeg analyserte dypest med tanke på kreative resonnement, viste at det er mulig å få til kreativ resonnering med riktig nivå på programmeringen og matematikken.

Samtalesekvensene har jeg gitt nummer fra 1 til 4. Samtalesekvens 1 og 3 viser at jentene fikk til kreative resonnement og disse var det som nevnt mange av underveis i transkripsjonen. Der de skulle beskrive algoritmene i programmene som ble presentert brukte ikke jentene mye tid, og de forstod fort hvordan kodene var bygget opp og hva som skjedde. Samtalesekvens 2 illustrerer en logisk brist hos jentene, som de etter hvert løser opp i senere i programmeringen. Samtalesekvens 4 viser guttenes arbeid med oppgave 3 som handler om å lage et program som regner ut potenser. Her svarer de på oppgaven, men programmet kan ikke generaliseres.

Det er ofte slik i transkripsjonen at delene av det kreative resonnementet ikke dukker opp på samme sted, og jeg har derfor laget en figur etter to av samtalesekvensene (de som inneholder kreative resonnement) som viser hva som illustrerer at resonnementet i samtalesekvensen er kreativt og nyskapende, plausibelt og matematisk forankret.

## 4.1 Samtalesekvens 1

Den første samtalesekvensen er hentet fra oppgave 2d, der elevene skulle svare på hva slags matematiske utregninger man kan bruke «slike typer» løkker til. Med slike typer løkker var det ment henvist til løkken som ble presentert i starten av oppgave 2 som var en for-løkke som regnet ut summen av tallene fra 1 til 10. Sonja foreslår at de kan prøve gange i stedet for pluss i koden presentert, hun foreslår altså at de skal skrive `summen = summen * i`, men da får hun 0. Grunnen til dette er at «`summen`» starter på 0, og det er hvorfor de får 0 til slutt jentene snakker om her:

---

*Sonja: Vi kan jo prøve gange?*

*Oda: ja. Gange er stjerne tror jeg*

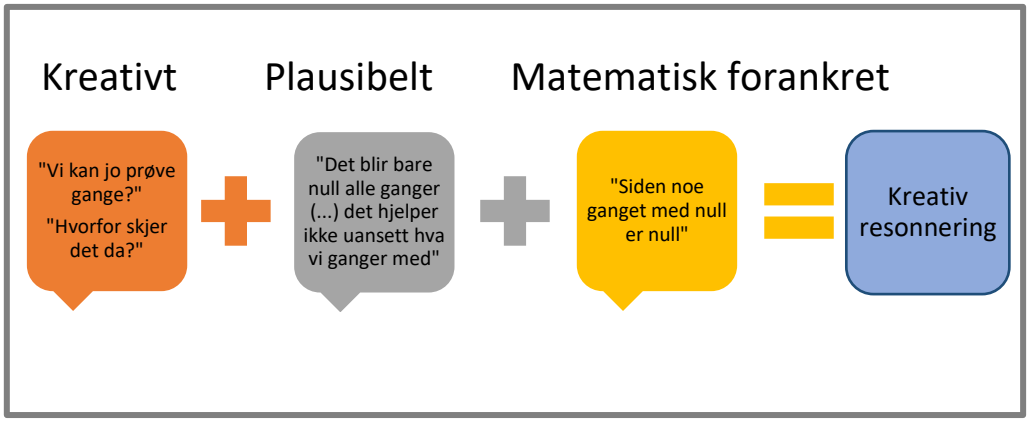
*Sonja: (får 0 når hun setter inn gange i stedet for pluss) Ja, fordi at første gang så..*

*Oda: Hvorfor skjer det da? For summen er null…*

*Sonja: det blir bare ganger null alle ganger. Siden noe ganget med null er null, så da hjelper det ikke uansett hva vi ganger med*

---

Vi finner de tre kriteriene for kreativt resonnement i denne samtalesekvensen. Kreativitet ledes opp med spørsmålet *vi kan jo prøve gange?*. Det er tydelig at jentene må resonnere på en ny måte enn de har gjort før, særlig med spørsmål som *hvorfor skjer det da?* Plausibilitet i denne resonneringssekvensen dukker opp i forklaringen av hvorfor de får 0 som svar til slutt når de bytter ut pluss med gange, da sier Sonja: *det blir bare ganger null alle ganger. Siden noe ganget med null er null, så da hjelper det ikke uansett hva vi ganger med.* Her ligger også den matematiske forankringen, fordi hun sier *siden noe ganget med null er null.*




**Figur 5 Kreativt, plausibelt og matematisk forankrede utsagn gir kreativt resonnement.**

En slik åpen diskusjonsoppgave om programmering førte til et kreativt resonnement hos disse to elevene. Dette resonnementet lignet på det de skulle gjøre senere, nemlig lage et program som regnet ut potenser. Her ble det forhåpentligvis satt i gang tankeprosesser som kanskje hjalp elevene senere i oppgaveløsningen.

## 4.2 Samtalesekvens 2

De to jentene har naturlig nok også resonnementer som ikke er kreative resonnement i matematikk, og her skal jeg presentere en samtalesekvens med en logisk brist i et resonnement som jeg mener mangler plausibilitet for at det skal kunne kalles kreativt. Her skal elevene programmere sitt eget program som skriver ut alle tallene i tregangen opp til 30. Jentene står fast på hvordan de skal bruke `i` som er presentert i løkka på oppgavearket, som de egentlig ikke trenger:

---

*Oda: Vi kan skrive while tallet er mindre enn 31, print tallet +3, også innskrevet.. sånn ja, også print tallet +3. tror du det går?*

*Sonja: Det var veldig lett da.. Oi! Åja, okei. Ja, men da fikk vi noe, det var litt digg å få noe*

*Oda: Men da må vi kanskje ha i også i +3*

*Sonja: Men hva er i-en? I-en er 3?*

*Oda: Ja. Og da tar vi bare.. vi kan skrive, okay*

*Sonja: åja den bare fortsetter den,*

*Oda: ja, bare x den ut, det skjedde med meg i stad for dette her står fortsatt her, så går vi bare inn igjen på nytt*

*Sonja: Okay, hehehe*

*Oda: Okay, while tallet er mindre enn 31 print (tallet+i), men da må vi skrive tallet =..*

*Sonja: Men skal vi prøve å gjøre noe av det samme her?*

*Oda: Ja, vi kan skrive tallet = tallet +i under while*

*Sonja: tallet = tallet +i ..*

*Oda: ja, også p..*

*Sonja: ..skal jeg skrive noe mer?*

*Oda: i = i*

*Sonja: i = i +*

*Oda: + 3?*

*Sonja: ja*

*Oda: +3, bare endre den… da trenger vi ikke ta den + 3 greia*

*Sonja: Jeg skriver den på ikke innrykk jeg*

*Oda: Ja*

*Sonja: Sånn, skal vi prøve det? Vi får egentlig bare prøve litt*

*Oda: 33… Da blir det liksom sånn sum*

---

Jentene prøver å få med `i`, som de egentlig ikke trenger å ha med. Det å plusse `i` på tallet gir ikke helt mening, da `i` øker med 3 for hver runde. Det kreative og nyskapende her er det som etter hvert løser oppgaven, de er på sporet om at «`i`» er noe de ikke trenger å ha med, og at de ikke kan plusse den på. «Hva er `i`-en?» er et godt spørsmål fordi i øker i koden deres hele tiden, noe som gjør det meningsløst å plusse i på tallet (da blir det sum). De får hjelp av lærer etter hvert, men er på sporet av at det er galt å plusse det på tallet. Matematisk forankring i denne sekvensen handler om at de skjønner at i tregangeren opp til 30 er en løsning at 3 plusses på tallet og at det skal fortsette hele tiden mens tallet er under 31**.** Plausible argumenter for hvorfor de skal ha med `i` er noe som mangler i resonnementet, noe som ikke er rart da det er en logisk brist å ha med

en `i` som hele tiden øker. Her kan oppgaven ha forvirret jentene, da i var med i eksemplene før, og da økte den og økningen hadde en funksjon.

## 4.3 Samtalesekvens 3

Den tredje samtalesekvensen er et ledd i oppgaveløsningen til jentene som gikk ut på å lage et program som skulle regne ut tallet til potensen $2^7$, i oppgave 3b og de fikk ikke lov til å bruke den innebygde funksjonen opphøyd (som skrives **) i Python. De to jentene har funnet ut at antall ganger løkka skal kjøres avhenger av eksponenten, men diskuterer hvordan de kan regne ut $2^7$ ved å ta med seg det forrige tallet videre. Altså, for å kunne programmere en potens uten å bruke den innebygde funksjonen opphøyd (**) i Python. Det at ikke «i-en» ikke skal brukes til noe er forvirrende for elevene, fordi de er vant til at økningen av den har noe å si for programmet. Her kunne de altså bare latt i-en øke gjennom løkka og gjort iterasjonene så mange ganger som de skulle, uten at i-en ble brukt til noe i koden. Samtalesekvensen har til tross for litt forvirring om i-en (igjen) mye kreativt og er representativt for hvordan jentene jobber. Den så slik ut:

---

*Sonja: for i in range(8) da*

*Oda: Okay?*

*Sonja: Noe sånt. Print tallet?*

*Oda: tallet *i*

*Sonja: Ja… nei ikke gange? Fordi tallet*

*Oda: gange tallet…*

*Sonja: Fordi tallet er jo alltid lik to*

*Oda: tallet *i, nei tallet*tallet.. **det** er jo ikke noe vits*

*Sonja: tallet er jo alltid to*

*Oda: tallet *i nei talle t*tallet også må vi lage en ny under som sier tallet = tallet *2, nei gange i*

*Sonja: Vi må ta ett eller annet som er sånn tallet *tallet*

*Oda: se skriv nå: print( tallet *tallet) også lager du en ny under som er i=tallet *i. Også prøv det, for da blir det…*

*Sonja: okay, fordi at tallet*tallet*

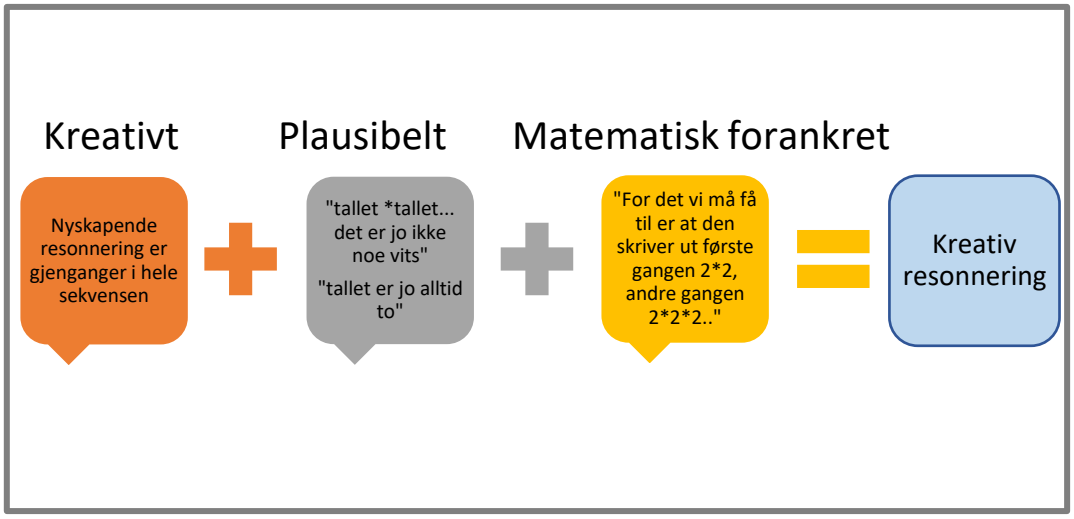
*Oda: For da starter tallet på 0 også blir det 0*0, som er 0, også pluss… nei det gir jo ikke mening*

*Sonja: for det vi må få til er at den skriver ut første gangen: 2*2, andre gangen 2*2*2. tredje gangen… Hvis vi tar print tallet*… hmmmmm….*

---

I denne samtalesekvensen prøver jentene å finne ut hvordan de skal gange med et nytt totall for hver runde, og denne sekvensen leder til at de på et senere tidspunkt finner ut hvordan dette skal gjøres. Grunnen til at denne sekvensen blir trukket fram er at måten de snakker sammen på er typisk for hele transkripsjonen av gruppa, og denne tankeprosessen med at *tallet er jo alltid to* noe som de skjønner betyr at de ikke kan skrive tallet ganger tallet betyr en del for videre framgang senere i transkripsjonen.

Vi finner alle kjennetegnene på kreativt resonnement her. Kreativitet viser jentene gjennom hele sekvensen, det er tydelig at de er inne på noe nytt fordi utsagn som *vi må ha ett eller annet sånn tallet *tallet* viser at ikke løsningen er åpenbar eller innøvd. Plausibilitet kan vi særlig se i argumentene om hvorfor det ikke kan være tallet ganger tallet som er riktig å skrive, for som Sonja sier: *Tallet er jo alltid to.* Det siste, matematisk forankring, ser vi særlig mot slutten av sekvensen, da Sonja sier det de begge antakelig har tenkt på hele tiden, begge to: *For det vi må få til er at den skriver ut den første gangen 2*2, andre gangen 2*2*2.*




**Figur 6 Tegnene på kreative resonnement er oppfylt**

## 4.4 Samtalesekvens 4

Den fjerde samtalesekvensen er fra den andre gruppa, som er hentet fra da de var på oppgave 3b, der de skulle regne ut $2^7$. Guttene kommer til slutt fram til en løsning, men den hjelper dem ikke i 3c der de skal generalisere, fordi de ganger x med seg selv 7 ganger, også bruker de en for-løkke til å teste om tallet er to. Her er samtalesekvensen:

---

*Olav: Altså 7-potensgangen hvis det er noe som heter det? Skjønner du hva jeg mener med det?*

*Jesper: Da tenker du på at..?*

*Olav: Hvis vi fjerner denne da så blir det liksom tall = x*x*x*x*x*x*x, 7 ganger da, også if x==2, så blir det da print tall. Da har vi sånn at for den vil da gå, det her er da x^7 også hvis x = 2*

*Jesper: så vil den printe tallet*

*Olav: ja, og det er da 2^7. Det blir da 128 og hvis ikke min matematikk er helt feil*

*Jesper: Det er riktig.*

---

Her er det tydelig at resonnementet er matematisk forankret og er nyskapende, men de plausible argumentene mangler for at resonnementet skal være kreativt. Den matematiske forankringen ligger i konseptet til potensen: 2 skal ganges med seg selv 7 ganger. Kreativiteten/ nyskapningen ligger i at de lager et nyskapende program som svarer på oppgaven og at det er nye tanker for guttene. Plausible argumenter for hvorfor de gjør det på denne måten mangler for at resonnementet skal være et kreativt resonnement i matematikk. Hvorfor loope igjennom en for-løkke og teste om tallet er to, når vi vet at tallet er to? Dette gir ikke helt mening algoritmisk eller med tanke på programmeringslogikk. De svarer på oppgaven fordi de bruker en for-løkke til å regne ut $2^7$, men får senere utfordringer når de skal generalisere programmet. De kan ikke skrive grunntall = (et tall) og eksponenten = (et tall) øverst i programmet fordi løkken kun fungerer på $2^7$. Derfor måtte de prøve å tenke nytt da neste oppgave skulle løses.

# 5 DISKUSJON

Forskningsspørsmålet i denne oppgaven er: Hvordan kan programmeringsoppgaver bidra til kreative resonnement i matematikk? Jeg har med utgangspunkt i analyseprosessen beskrevet i 3.5 utarbeidet resultatene som beskrevet i forrige kapittel (kapittel 4) og skal i dette kapittelet gå nærmere inn på disse med en diskusjon rundt hvordan programmeringsoppgavene har bidratt til kreative resonnement.

## 5.1 Flere løsningsmetoder viktig for å få fram kreative resonnement

Begge de to eksemplene fra jentene der det var kreative resonnement (samtalesekvens 1 og 3) viste det som var typisk for de oppgavene der de kreative resonnementene som oftest dukket opp i transkripsjonen: Det var flere mulige løsningsmetoder. Dette er et av kjennetegnene til problemløsningsoppgaver (Psycharis & Kallia, 2017). Det betyr altså at ved mer åpne oppgaver med flere løsningsmetoder, kan det dukke opp kreative resonnement. Disse resultatene er i tråd med litteraturen til Lithner (2015).

En utfordring, og noe å tenke på, er likevel at ved laging av oppgaver som gjør at elevene resonnerer kreativt kan elevene bli stående fast på utfordringer med syntaks eller programmeringslogikk, og det er derfor ikke nødvendigvis riktig å gi elevene programmeringsoppgaver som kun går innunder kategorien problemløsning eller som bare tilrettelegger for kreative resonnement. Som nevnt i teoridelen (kapittel 2) er det et samspill mellom imitativ og kreativ resonnering (Lithner, 2015; Szymanski, 2018), og en slags innlæring som gir innblikk i syntaks kan derfor være lurt, og er i starten antakelig nødvendig, hvis elevene ikke er stødige i programmering.

## 5.2 Kompetanse i algoritmisk tenkning som et verktøy

Samtalesekvens 4 (av Jesper og Olav) viser at det ikke alltid er like enkelt, selv med en oppbygning mot det som skal programmeres, for elever på videregående skole med begrenset programmeringsbakgrunn, å lage en generell algoritme og å resonnere

kreativt. Det var antakelig nivået på programmeringen og den algoritmiske tenkningen som var utfordringene hos Jesper og Olav. De skulle lage et program som regnet ut potensen $2^7$, og programmet skulle generaliseres etterpå. Det var tydelig at de skjønte det matematiske som skulle utføres siden de ganget x med seg selv syv ganger, men å bygge opp en generell løkke der verdien til potensen ble tatt med videre for hver runde i løkka virket å være vanskelig for guttene. Plausibiliteten manglet for at resonnementet til guttene her skulle kunne kalles kreativt, noe som sikkert likevel kan diskuteres. Hadde den algoritmiske tenkningen vært mer øvet, kunne de kanskje funnet en algoritme til programmeringen som gjorde at det plausible kunne vært mer synlig.

Oppgavene må være på riktig nivå, og vi har sett at det kan være utfordrende hvis oppgavene ikke er det, og at det hos Jesper og Olav ikke var på langt nær like mange matematiske resonnement, antakelig fordi det ikke var riktig nivå for dem og de stod fast. Oppgavene bør naturlig nok være nivåtilpasset både med tanke på programmerings- og matematikkunnskaper. Ifølge Wing (2008) har algoritmisk tenkning en essens i abstraksjon, og en generalisering er en måte å gjøre noe abstrakt på. Elevene er antakelig for lite øvet i algoritmisk tenkning og generalisering i programmering. En kan tenke seg at hvis guttene hadde vokst opp med programmering i skolen så hadde antakelig det logiske i programmeringen vært lettere tilgjengelig for dem, og den algoritmiske tenkningen vært mer øvet.

Weintrop og kollegaer (Weintrop et al., 2015) skrev om at elevene kan, gjennom algoritmisk tenkning, se en gjensidig nytteverdi mellom programmering og matematikk, og her var det antakelig et forbedringspotensial i oppgaveheftet. En kan diskutere om nytteverdien av å lage et program som gjør akkurat det samme som en innebygget funksjon i programmeringsspråket er tydelig for elevene.

## 5.3 Helhetlig tenking i laging av oppgaver

Ifølge NCTM (2014) er det at oppgavene er kognitivt utfordrende viktig for forståelse, og dermed blir det også viktig for kreativ resonnering. Det er derfor en fin balanse det å få til riktig vanskelighetsgrad i matematikk og programmering, og å gjøre matematikken utfordrende nok. Dette kunne oppgavene vært bedre på: Det kan tenkes elevene fikk en dypere forståelse av potenser, men hvor vanskelig er egentlig konseptet potenser for 1T-elever?

Det har vist seg at grundig gjennomtenkning av alle ledd i oppgavene er vesentlig for at kreative resonnement skal være gjennomgående. I samtalesekvens 2 ble jentene forvirret av `i` og dens funksjon. Her kan jeg også ha forvirret mer enn hjulpet da jeg fikk spørsmål fra jentene hvordan de skulle løse oppgaven, og jeg spurte om de hadde med en teller. Dette trengte man ikke i oppgaven, fordi man kunne begrense det til «`while tallet < 31`», da tallet økte hele tiden. Det kunne også tenkes at hvis jeg hadde brukt en annen kodesekvens som eksempel, som ikke trengte en teller, at jentene hadde blitt mindre forvirret. Det at jentene kan ha blitt forvirret med eksempelet før oppgaven betyr at en helhetlig tenkning rundt sammenhengene mellom de ulike oppgavene som skal løses er vesentlig.

Oppgavene skal henge sammen, og vi har sett at oppgaver som for eksempel går ut på å beskrive en algoritme ikke er tilstrekkelig for å få fram kreative resonnement – det kreves åpne oppgaver som ikke har én løsning for å få frem dette. Det er, som vi har sett, lurt med en oppbygning der problemløsning er fokus hele veien.

# 6 KONKLUSJON OG IMPLIKASJON

## 6.1 Konklusjon

Fra et matematikkdidaktisk perspektiv, er det muligheter for å få en dypere forståelse ved å jobbe med programmeringsoppgaver som bidrar til kreativ resonnering hos elevene. Algoritmisk tenkning som beskrevet i denne oppgaven er bredt, og handler blant annet om å løse algoritmiske problemer og rett og slett forstå verden rundt seg gjennom løsning av matematikkoppgaver med datamaskiner (Weintrop et al., 2015). Når det kommer til programmeringsoppgaver i matematikk er algoritmisk tenkning en stor del av det som trenes og brukes under løsning av slike oppgaver, og vi har også sett at for eksempel problemløsning og modellering er deler av den algoritmiske tenkningen (Weintrop et al., 2015). Programmering i matematikk kan føre til dybdelæring, og matematikk kan bidra inn i programmeringens verden fordi å programmere matematiske problemer kan føles mer nyttig (Weintrop et al., 2015). Hvis en elev resonnerer kreativt og får oppgaver rettet mot problemløsning vil dette føre til dybdelæring hos eleven (Lithner, 2017), som de nye læreplanene har som et krav til undervisning.

Fokuset i denne oppgaven er hvordan programmering kan bidra til kreative resonnement, og hva forteller resultatene og diskusjonen oss i lys av teorien? Vi så på at kreativ resonnering består av tre faktorer: kreativitet, plausibilitet og matematisk forankring. Det kan være nyttig å se hvordan programmering bidrar i hver av de tre delene.

### 6.1.1 Kreativitet

Kreativitet gikk ut på at resonneringen skulle være ny eller gjenskapt for den som resonnerer (Lithner, 2015), og det vi har sett er at i samtalesekvensene fremstilt i denne oppgaven, og noe jeg la merke til ved analyse av dataene, er at nyskapning er en gjenganger. Så lenge elevene får oppgaver som ikke er helt like det de har gjort før, bruker de på dette nivået kreativitet og nyskapning svært ofte.

Det å ha en kobling til problemløsning ved hver deloppgave var nevnt i teoridelen som et grep for å få fram matematikkforståelse (Govender, 2007), og jeg så at mange av deloppgavene som ikke handlet om å lage egne program likevel førte til kreative resonnement med nyskapning og kreativitet fordi de hadde flere svar.

### 6.1.2 Plausibilitet

Plausibilitet er et annet krav for at et resonnement i matematikk skal være kreativt, da må det å ha plausible argumenter til løsningsstrategi på en oppgave (Lithner, 2015), noe som betyr at argumentene må være rimelige. Da er det vesentlig at elevene forstår oppgaven og har en idé om hvordan de skal programmere dem, oppgavene må altså være på riktig nivå. Målet bør være at elevene skal resonnere kreativt, og det er ifølge Govender (2007) ofte logikken og problemløsningen i programmene som er det utfordrende i programmering i matematikk, noe som også har kommet fram i denne studien. Derfor er det å koble opplæring i syntaks til problemløsning og logisk tenkning i programmering et grep for å hindre for mye imitativ resonnering, men samtidig gjøre slik at elevene ikke står fast ved programmeringstekniske utfordringer.

### 6.1.3 Matematisk forankring

Matematisk forankring er det siste kravet for at et resonnement skal være kreativt, og for å oppfylle dette kravet må resonneringen ha noe med matematiske egenskaper å gjøre (Lithner, 2015). Derfor er det viktig at det er en kobling til matematikk, og resonnementene må inneholde matematiske egenskaper om komponentene som er med. Det bør derfor være enten direkte matematiske oppgaver, eller naturlig å koble programmeringen til matematikk.

## 6.2 Implikasjon

For at programmeringsoppgaver i matematikk skal bidra til kreative resonnement må de være nye for elevene, hver oppgave må ha en kobling til problemløsning og være tilpasset nivået til elevene og det må være mulig, og aller helst naturlig, å koble det som programmeres til matematiske egenskaper. Vi har sett at dette kan være krevende. Det krever en viss kjennskap til elevenes kompetanse i matematikk og programmering. Litteraturen forteller oss at for at oppgaver i matematikk skal føre til kreativ resonnering må oppgavene ikke være repetitive, men åpne opp for nye løsningsmetoder og forståelse, noe vi også har sett i denne studien.

Når det gjelder videre forskning er det å se på en lignende oppgave eller andre oppgaver i videregående skole ved høyere programmeringskunnskap om løkker noe som trengs. En kunne også tenke seg at å forske på hvordan de som designer programmeringsoppgaver skal heve utfordringen på matematikken uten å heve utfordringen i programmering hadde gitt forskningen mye, da nivået på programmeringen i forhold til nivået på matematikken kanskje var en av de største utfordringene i denne studien, da elevenes logikk ble aller mest brukt på programmeringen.

# REFERANSER

Adams, D., & Hamm, M. (2013). *Demystify math, science, and technology: creativity, innovation, and problem solving* (2nd ed.). Lanham, USA: Rowman & Littlefield Education.

BarefootComuputingUK. (2016). The Computational Thinker: Concepts and Approaches. Retrieved from https://www.barefootcomputing.org/docs/default-source/resource-downloads/15234_bt-barefoot-computational-thinker-poster_a1_eng.pdf?sfvrsn=aeb390ea_2

Bryman, A. (2012). *Social research methods* (4th ed.). Oxford: Oxford University Press.

Christoffersen, L., & Johannesen, A. (2012). *Forskningsmetode for lærerutdanningene*. Oslo: Abstrakt forlag.

Csernoch, M., BirÓ, P., Máth, J., & Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. *Informatics in Education An International Journal, 14*(2), 175-197. doi:10.15388/infedu.2015.11

Dahl, G., Ranestad, K., & Hole, A. (2017). Programmering rammer dybdelæring i matematikk. *Aftenposten*. Retrieved from https://www.aftenposten.no/meninger/kronikk/i/E0zga/programmering-rammer-dybdelaering-i-matematikk-geir-dahl-kristian-ra

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.

Fox, R., & Farmer, M. E. (2011). *The effect of computer programming education on the reasoning skills of high school students.* Paper presented at the Proceedings of the international conference on frontiers in education: Computer science and computer engineering (FECS'11), Las Vegas, NV, USA.

Govender, I. (2007). Experiences of learning and teaching: Problem Solving in computer programming. *African journal of research in mathematics, science and technology education, 11*(2), 39-50. doi:10.1080/10288457.2007.10740620

Granberg, C., & Olsson, J. (2015). ICT-supported problem solving and collaborative creative reasoning: Exploring linear functions using dynamic mathematics software. *The Journal of mathematical behavior, 37*, 48-62. doi:10.1016/j.jmathb.2014.11.001

Hershkowitz, R., Hershkowitz, R., Tabach, M., Tabach, M., Dreyfus, T., & Dreyfus, T. (2017). Creative reasoning and shifts of knowledge in the mathematics classroom. *ZDM, 49*(1), 25-36. doi:10.1007/s11858-016-0816-6

Jeannotte, D., & Kieran, C. (2017). A conceptual model of mathematical reasoning for school mathematics. *Educational studies in mathematics, 96*(1), 1-16. doi:10.1007/s10649-017-9761-8

Kaufmann, O. T., & Stenseth, B. (2020). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, 1-20. doi:10.1080/0020739X.2020.1736349

Kaur, B., & Toh, T. L. (2012). *Reasoning, communication and connections in mathematics*. Singapore: World Scientific Pub. Co.

Kilpatrick, J., Swafford, J., & Findell, B. (2001). *Adding It Up: Helping Children Learn Mathematics*. Washington, DC: The National Academies Press.

Lambic, D. (2010). Presenting practical application of Mathematics by the use of programming software with easily available visual components. *Teaching mathematics and its applications, 30*(1), 10-18. doi:10.1093/teamat/hrq014

Lavy, I. (2006). A Case Study of Different Types of Arguments Emerging From Explorations in an Interactive Computerized Environment. *The Journal of mathematical behavior, 25*(2), 153-169. doi:10.1016/j.jmathb.2006.02.006

Lithner, J. (2015). Learning Mathematics by Creative or Imitative Reasoning. In S. J. Cho (Ed.), *Selected Regular Lectures from the 12th International Congress on Mathematical Education* (pp. 487-506). Cham: Springer International Publishing.

Lithner, J. (2017). Principles for designing mathematical tasks that enhance imitative and creative reasoning. *ZDM : The International Journal on Mathematics Education, 49*(6), 937-949. doi:10.1007/s11858-017-0867-3

Misfeldt, M., & Ejsing-Duun, S. (2015, 2015-02-04). *Learning mathematics through programming: An instrumental approach to potentials and pitfalls.* Paper presented at the CERME 9 - Ninth Congress of the European Society for Research in Mathematics Education, Prague, Czech Republic.

Monaghan, J., & Ozmantar, M. F. (2006). Abstraction and Consolidation. *Educational studies in mathematics, 62*(3), 233-258. doi:10.1007/s10649-006-8753-x

NCTM. (2014). *Principles to Action*. Reston, VA: The National Council of Teachers of Mathematics, Inc.

Nilssen, V. (2012). *Analyse i kvalitative studier. Den skrivende forskeren.* Oslo: Universitetsforlaget.

NOU. (2015:8). *Fremtidens skole - Fornyelse av fag og kompetanser*. Oslo: Departementets sikkerhets- og serviceorganisasjon Retrieved from https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/?ch=2#KAP1-1-2

Polýa, G. (1945). How To Solve It A Dialogue. In *How To Solve It* (pp. 33-36). Princeton: Princeton University Press.

Powell, A. B., Francisco, J. M., & Maher, C. A. (2003). An Analytical Model for Studying the Development of Learners' Mathematical Ideas and Reasoning Using Videotape Data. *The Journal of mathematical behavior, 22*(4), 405-435. doi:10.1016/j.jmathb.2003.09.002

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional science, 45*(5), 583-602. doi:10.1007/s11251-017-9421-5

Regjeringen.no. (2015-2016). Stortingsmelding 28: Fag - Fordypning - Forståelse - En fornyelse av Kunnskapsløftet. Retrieved from https://www.regjeringen.no/no/dokumenter/meld.-st.-28-20152016/id2483955/

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., . . . Voll, L. O. (2016). Teknologi og programmering for alle. Retrieved from https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/teknologi-og-programmering-for-alle

Saritepeci, M. (2020). Developing Computational Thinking Skills of High School Students: Design-Based Learning Activities and Programming Tasks. *The Asia-Pacific education researcher, 29*(1), 35-54. doi:10.1007/s40299-019-00480-2

Skemp, R. (1978). Relational Understanding and Instrumental Understanding. *The Arithmetic Teacher, 26*(3), 9-15.

SSB. (2019). Skrevet datakode i programmeringsspråk. *Erfaring/ferdigheter med PC- og Internett-bruk (prosent), etter kjønn, alder, statistikkvariabel og år.* Retrieved from https://www.ssb.no/statbank/table/11438/tableViewLayout1/

Stephens, M., & Kadijevich, D. M. (2020). Computational/Algorithmic Thinking. In S. Lerman (Ed.), *Encyclopedia of Mathematics Education* (pp. 117-123). London: Springer International Publishing.

Szymanski, A. (2018). Prototype Problem Solving Activities Increasing Creative Learning Opportunities Using Computer Modeling and 3D Printing. In V. Freiman & J. L. Tassell

(Eds.), *Creativity and Technology in Mathematics Education* (pp. 323-344). Cham: Springer International Publishing.

UDIR. (2019). Algoritmisk tenkning. Retrieved from https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

UDIR. (2020). Innføring av nye læreplaner. Retrieved from https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/innforing-av-nye-lareplaner/

UDIR. (n.d.). Matematikk 1T - Kompetansemål og vurdering. Retrieved from https://www.udir.no/lk20/mat09-01/kompetansemaal-og-vurdering/kv42

Wadel, C. (1991). *Feltarbeid i egen kultur : en innføring i kvalitativt orientert samfunnsforskning*. Flekkefjord: SEEK.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of science education and technology, 25*(1), 127-147. doi:10.1007/s10956-015-9581-5

Whittemore, R., Chase, S. K., & Mandle, C. L. (2016). Validity in Qualitative Research. *Qual Health Res, 11*(4), 522-537. doi:10.1177/104973201129119299

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the Royal Society of London, 366*(1881), 3717-3725. doi:10.1098/rsta.2008.0118

# VEDLEGG

# Programmere potenser

Programmeringsoppgaver i Python. Forfatter av tekst og oppgaver: Ina Ek Tveiten

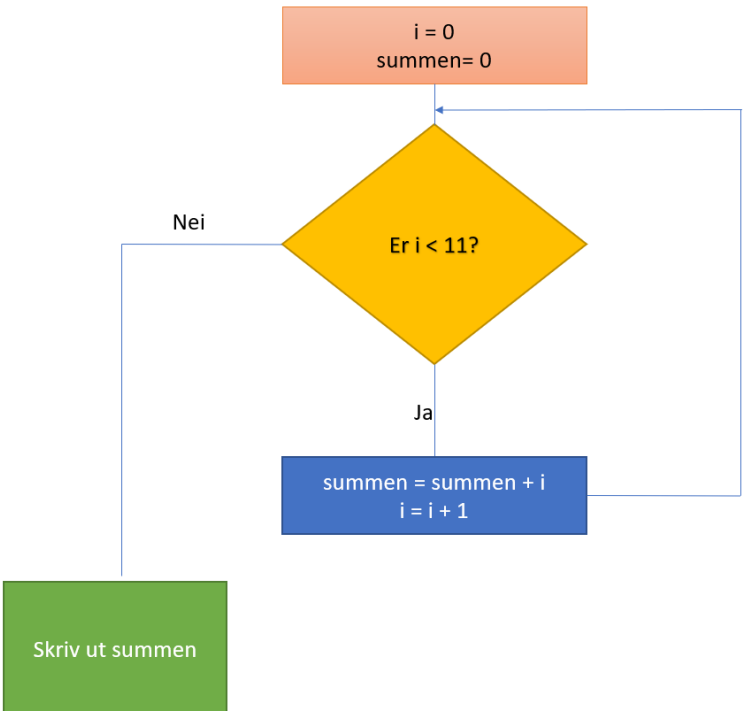
## While-løkker

Under ser du en while-løkke i Python. While-løkker går igjennom runder for å finne ut ulike ting, og har en test i starten der noe (en variabel) må oppfylle visse krav for at løkka skal fortsette å kjøre. Her vil vi finne summen av tallene fra og med 1 til og med 10 (dvs.: 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10). Her har vi skrevet: «`while i < 11`», som betyr at programmet kjøres så lenge «i» er under 11. Altså: så lenge «i» er under 11, så går løkka videre til neste runde.

```
#Sum av tallene fra 1 til 10, while

summen = 0
i = 1

while i < 11:
    summen = summen + i
    i = i + 1

print(summen)

```

Vi må huske å fortelle datamaskinen at «`summen = 0`» og at «`i = 1`» i starten, da har den startverdier for disse og kan regne ut summen i første runde i løkka.

Vi kan fremstille det i et flytskjema:



Hvordan kan «`summen = summen + i`» være riktig å skrive? Det er ikke det samme, ifølge hva matematikk har lært oss. Likhetstegnet brukes annerledes i programmering enn i matematikk, vi *oppdaterer* summen til det som står til høyre for likhetstegnet. Det betyr at «`summen`» (venstre side) blir til «`summen + i`» (høyre side) til den neste gang løkka gjentas. Dette betyr at «`summen`» øker med «`i`» for hver runde.

## Oppgave 1, while-løkke

a) Hva skjer i første runde av while-løkka?
b) Hva skjer andre runde i while-løkka? Hvordan blir «`summen`» forskjellig fra første runde?
c) Hvorfor har vi med «`i = i + 1`»? Hva hadde skjedd hvis denne ikke hadde vært med i koden, tror dere?
d) Lag en egen while-løkke som skriver ut tallene i 3 gangeren **opp til 30.** Hvor bør `print()` stå i koden? Bruk gjerne tips og triks fra koden med while-løkka som dere har diskutert nå, eller tenk nytt hvis dere ønsker!

## For-løkker

For-løkker er en annen type løkke, men kan brukes på samme måte som while-løkker. Forskjellen på de to er at for-løkker går automatisk videre til neste element mens while-løkker kjører helt til noe er oppnådd. Vi trenger dermed ikke skrive «`i = i + 1`» for at den skal gå videre til neste element. Her vil vi vise et program som gjør det samme som while-løkka som ble vist tidligere. Vi skriver «range(11)» som forteller for-løkka at den skal starte på 0 (det gjør den som standard) og gå gjennom heltallene til og med 10. Skriver vi «range(51)» vil dette programmet regne summen av alle tall fra 0 og til og med 50. Vi kunne også skrevet «range(1,11)» som forteller oss at vi skal starte på 1 og slutte på 10. Programmet slutter altså på tallet før når vi bruker den innebygde range-funksjonen.

```
7  #sum av tallene fra 1 til 10
8
9  summen = 0
10
11 for i in range(11):
12     summen = summen +  i
13
14 print(summen)
15
```

### Oppgave 2, for-løkke

a) Forklar hva som skjer første runde i for-løkka vist over.
b) Hvis vi hadde skrevet range(1,11) hadde vi fått samme resultat, men med én runde mindre i for-løkka, hvordan er det mulig?
c) Hva skjer 2. runde i for-løkka? Hvordan blir «summen» på høyre side av likhetstegnet forskjellig til 3. runde?
d) Hvilke typer matematiske utregninger kan man bruke sånne løkker til?
e) Skriv av programmet i Python. Hva slags resultat forventer dere å få? Kjør programmet og se om det blir det dere tenkte på forhånd. Hvorfor ble det som forventet eller ikke som forventet?
f) Prøv å forklare hva som skjer hvis vi bytter ut «+ i» med «-i».
g) Gjør endring i programmet slik at det står «-i» i stedet for «+i». Skjedde det dere trodde ville skje?

## Oppgave 3, programmere potenser med for-løkke

a) Hva skjer når eksponenten til en potens øker med 1? Dere skal lage et program som bruker for-løkker og «*» (som betyr multiplisert i Python) til å regne ut potenser. Hvor mange runder bør løkken gå?

b) Lag et program som regner ut 2^7 ved å bruke en for-løkke.

c) Generaliser programmet ved å kalle 2 for «grunntall» og 7 for «eksponent». Da kan vi bestemme øverst i programmet hva grunntallet skal være («`grunntall = 4`» for eksempel) og hva eksponenten skal være («`eksponent = 3`» for eksempel), som gir oss $4^3$. Både «`grunntall`» og «`eksponent`» kalles variabler. Da er det viktig å erstatte alle steder der grunntallet 2 står til «`grunntall`» og alle steder der 7 står til «`eksponent`», sånn at vi kan velge øverst i programmer hvilken verdi variablene skal ha.
   - Hva skjer hvis `grunntall = 10` og `eksponent = 4`. Diskuter først og sjekk deretter ved å kjøre programmet med disse tallene om dere har tenkt riktig.
   - Kan vi kjøre programmet med en eksponent som er negativ? Diskuter, prøv og forklar hva du tror skjer!
   - Hva skjer hvis «grunntall» er en brøk, for eksempel $\frac{1}{2}$ og eksponenten er 5? Hvordan kan dette uttrykkes med 2 som grunntallet i en potens?
   - Få programmet til å gi dere tallet 0,001. Hvilke grunntall kan dere bruke? Hva blir eksponenten i tilfellene?

d) Hva er likheten mellom potenser og sum når det kommer til programmering av de to matematiske konseptene i løkker?

# Vil du delta i forskningsprosjektet

## *Elevers resonnement ved bruk av programmering i matematikk?*

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å finne ut hvordan programmering kan bidra til resonnement i matematikk. I dette skrivet gir jeg deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

**Formål**
I dette prosjektet skal jeg undersøke hvordan programmering kan bidra til ulike resonnementer i matematikk. Programmering og dybdelæring har blitt en del av matematikk i de nye læreplanene og derfor mener jeg det er viktig at det kommer studier som belyser sammenhenger mellom programmering og dere elevers matematiske resonnement. Resonneringen kan gi innblikk i hvordan dere har forstått matematikken dere jobber med. Dette er et masterprosjekt som skal ende i en masteroppgave. Hva slags typer programmeringsoppgaver som bidrar til matematiske resonnement blir viktig i masteroppgaven, og det blir viktig for meg at matematikken ikke glemmes i syntaks (hvordan programmeringsspråket er bygget opp). Dette håper jeg å få til med de oppgavene jeg lager til dere.

**Hvem er ansvarlig for forskningsprosjektet?**
Norges Miljø- og Biovitenskapelige Universitet (NMBU) er ansvarlig for prosjektet.

- Masterstudent som skal gjennomføre prosjektet: Ina Ek Tveiten
- Veiledere: Margrethe Naalsund, Morten Munthe og Marte Bråtalien.

**Hvorfor får du spørsmål om å delta?**
Jeg skal forske på et utvalg elever som har Matematikk 1T dette året. Jeg spurte læreren din om hun ville være med, og det ville hun. Du får spørsmål om å delta fordi læreren din tror dere som har blitt spurt vil prate godt sammen og gi meg god innsikt i deres matematiske resonnement når dere jobber med programmeringen.

**Hva innebærer det for deg å delta?**

- Hvis du velger å delta vil det innebære at du løser oppgavene som blir gitt så godt **du** kan, med den partneren du blir satt sammen med. Denne personen er også med i studien. Det er ikke viktig å undersøke hvor god prestasjonen deres er, men i studien skal jeg se på hvordan dere tenker når dere løser oppgavene.

- Resten av 1T-gruppa får de samme oppgavene, men dere sitter i et annet rom sammen med meg som skal forske. I rommet der dere sitter vil det være et filmkamera, og både dere og skjermen blir filmet slik at det er lettere å tolke resonnementene deres i ettertid.

- Dataene blir slettet etter studien og vil ikke bli delt med andre enn veilederne. Mer om dette kommer lenger ned.

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

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

- De som kommer til å ha tilgang til opplysningene om dere er kun jeg som masterstudent og veilederne mine (navngitt tidligere).
- For å strukturere datainnsamlingen skal alt som sies skrives ned (transkriberes), og det blir gjort av meg som masterstudent. Jeg kommer til å bruke andre navn slik at transkripsjonen blir anonym.
- Transkripsjonen og videoopptak lagres på NMBU's skytjeneste (OneDrive) som har innlogging med Authentificator-appen til Microsoft (tofaktorautentifisering).

**Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?**
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er 31. mars 2021. I oppgaven nevnes ingen med navn eller beskrives – på den måten kan ikke leseren skjønne hvem dere ved å lese masteroppgaven. Videoopptak og skjermopptak blir slettet når masterprosjektet er avsluttet.

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

- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

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

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

**Hvor kan jeg finne ut mer?**
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:

- NMBU ved Margrethe Naalsund (margrethe.naalsund@nmbu.no) eller masterstudent Ina Ek Tveiten (ina.ek.tveiten@nmbu.no)
- Vårt personvernombud: Hanne Pernille Gulbrandsen (personvernombud@nmbu.no )

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

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

Med vennlig hilsen

*Prosjektansvarlig* *Eventuelt student*
(Forsker/veileder)

Vedlegg 2

---

# Samtykkeerklæring

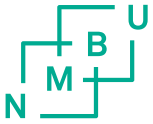
Jeg har mottatt og forstått informasjon om prosjektet *Elevers resonnement ved bruk av programmering i matematikk,* og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i videoopptak som blir transkribert og analysert
- ☐ å delta i skjermopptak som blir transkribert og analysert

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

---

(Signert av prosjektdeltaker, dato)




**Norges miljø- og biovitenskapelige universitet**
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway