Høgskulen på Vestlandet

MASTEROPPGAVE

Utforsking av delelighet og rest med tekstbasert programmering

Exploration of division and remainders with text-based programming

Fredrik Eidsvåg

Master i undervisningsvitenskap med fordypning i matematikk
Fakultet for lærerutdanning, kultur og idrett (FLKI)
Veileder: Rune Herheim
29.05.2020

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Forord

Arbeidet med masteroppgaven har vært spennende, givende og utfordrende. Bakgrunnen for å begynne et masterstudie i undervisningsvitenskap kommer fra et ønske om å utforske muligheter og forbedre egen undervisning i matematikkfaget etter å ha praktisert som matematikklærer i et par år. Å gjennomføre de ulike delene av prosjektet på egenhånd har vært både spennende og krevende. Læringskurven har blitt bratt, med mye å sette seg inn i både teoretisk og metodisk. Datainnsamlingen har også vært tidkrevende men jeg har lært mye underveis, noe jeg ikke ville vært foruten.

Jeg vil takke veileder Rune Herheim som fra starten av har vist stor interesse for prosjektet og som har bidrat med gode råd og grundig veiledning underveis. Tilbakemeldingene jeg har fått har vært til god hjelp både i det konkrete arbeidet med oppgaven, samt de har hjulpet meg til å utvikle meg som skribent. Tamsin Jillian Meaney fortjener også en stor takk for hjelp med prosjektsøknad til NSD gjennom LATACME.

Jeg vil også takke rektor, kollegaer og elever ved skolen der prosjektet er utført for å ha latt meg gjennomføre studien, for å ha stilt opp ved behov og for å ha deltatt i studien.

Tusen takk til min kjære kone Sunniva som har støttet meg gjennom arbeidet på flere områder. Faglig har du bidratt med gjennomlesning, korrektur og gode råd. I tillegg har du støttet i form av å ta vare på meg, familien og hjemmet i perioder hvor arbeidet med masteroppgaven har vært altoppslukende.

Som fulltidsstudent, lærer og pappa til tre små jenter (hvorav den ene ble født i mars) har hverdagen vært langt fra normal, spesielt da samfunnet ble snudd på hodet under koronautbruddet våren 2020. Å gjøre ferdig masteroppgaven, spesielt under disse omstendighetene, ville aldri vært mulig uten familien da uttallige timer har gått med til barnepass, både i helger og ukedager. Jeg vil derfor også gi en stor takk til familien for øvrig, som har gitt uforbeholden støtte gjennom hele perioden.

Mai, 2020

Fredrik

Sammendrag

Fra høsten 2020 blir algoritmisk tenkning og programmering en integrert del av norske læreplaner i matematikk. Forskning på feltet peker på koding og programmering som potensielt nyttig i matematikkundervisningen, men det vises også til utfordringer; spesielt gjelder dette å knytte ulike programmeringsaktiviteter til matematikkfaget.

Fokuset for denne oppgaven er å undersøke *muligheter og utfordringer ved tekstbasert programmering i matematikkundervisning på mellomtrinnet.* Det søkes *særlig innsikt i samspillet mellom elever, oppgave og programvare* (programmeringsspråket Python). For å best mulig kunne belyse temaet er det stilt følgende tre forskningsspørsmål; (1) Hva kjennetegner elevenes handlinger i interaksjon med oppgave og programvare? (2) Hvordan påvirker oppgavedesignet, sammen med verktøyet (programvaren), elevenes argumentasjon og resonnering? (3) Hvilken didaktisk funksjon spiller programvaren i de valgte situasjonene?

For å besvare forskningsspørsmålene er det gjennomført et aksjonsforskningsinspirert prosjekt hvor det er undervist i tekstbasert programmering gjennom en periode med til sammen 10 undervisningsøkter. Mot slutten er det forsøkt å rette fokus mot matematikkfaglig innhold og datamaterialet er samlet inn ved å filme og ta skjermopptak av elevpar i arbeid. Utdrag fra tre elevpar er analysert som kasus-studier med bakgrunn i Brousseaus didaktiske teori (TDS) for å kunne si noe om elevenes handlinger og matematiske aktivitet (argumentasjon og resonnering) i møte med oppgavene og programvaren. SAMR-modellen brukes i analysen som et tilleggsperspektiv for reflektere over muligheter og utfordringer ved å eksperimentere med digitale verktøy og nye tilnærminger.

Analysene viser at det er utfordrende å heve elevenes argumentasjon og resonnering til et nivå hvor elevene gjør vurderinger av påstanders gyldighet, forklarer sentrale elementer ved programvaren og kodene, og forsøker å bevise egne påstander. Det er likevel et tydelig potensiale for å utforske matematiske sammenhenger i et programmeringsmiljø (her: Python IDLE og trinket.io). Deler av resultatene viser en konstruktiv dialektikk mellom elev og programvare preget av matematiske diskusjoner. Ved å i større grad inkludere lærer og legge til rette for diskusjoner, antas det at disse sammenhengene vil kunne forsterkes til et nivå som også preges av validering (Jamfør Brousseau), men dette krever mer forskning.

Abstract

From 2020, algorithmic thinking and programming will become an integral part of Norwegian mathematics curricula. Research points to coding and programming as potentially useful in mathematics education, but it also points to challenges; in particular, this applies to linking various programming activities to specific subjects within mathematics.

The focus of this thesis is *to explore the affordances and constraints that text-based programming can provide in middle school mathematics education*. The insight that is particularly sought, is the *interaction between students, the task and the software* (the programming language Python). In order to best highlight the topic, the following three research questions have been asked; (1) What characterizes students' actions in interaction with the task and the software? (2) How does the task design, together with the tool (software), influence the students' argumentation and reasoning? (3) What didactic function does the software play in the selected situations?

To answer the research questions, an action research inspired project has been carried out, where text-based programming has been taught over a period of 10 lessons. Towards the end, an attempt was made to focus on mathematical content and the data material was collected by filming pupil pairs while working and by screen capturing software. The excerpts from three of the pupil pairs were analyzed as case studies, based on Brousseau's Theory of Didactical Situations (TDS), in order to be able to say something about the students' actions and mathematical activity (argumentation and reasoning) in relation to the tasks and the software. The SAMR model is used in the analysis as an additional perspective for reflecting on affordances and constraints when experimenting with new approaches using digital tools.

The analyses show that it is challenging to raise students' argumentation and reasoning to a level where students make assessments of the validity of the statements, explain key elements of the software and the codes, and try to prove their own statements within their work. The results nevertheless show a clear potential for exploring mathematical concepts in a programming environment (here: Python IDLE and trinket.io). Some of the results show a constructive dialectic between student and software characterized by mathematical discussions. By including the teacher to a greater extent and facilitating discussions, it is

assumed that these relationships can be strengthened to a level characterized by validation (i.e. Broussau), but this requires more research.

Innhold

Figurliste

1 Innledning

1.1 Bakgrunn

Debatten rundt programmering i skolen har i løpet av de siste to tiår vokst fra sporadiske kronikker i lokalaviser til en sentral komponent i en nasjonal debatt om skolens fremtid. Meningene rundt bruk av digitale verktøy generelt, og programmering spesielt, er mange og varierte. Noen hevder at «*Koding må inn i læreplanen for grunnskolen*» og begrunner dette med at elever må «*opparbeide seg en overordnet forståelse for hvordan teknologien henger sammen og kan brukes i samfunnet vårt i dag*» (Nordseth, 2017). Andre hevder at man bør være varsomme med å innføre programmering og koding som del av skolefagene, spesielt med tanke på lærerne som skal undervise i fagene og som trenger kompetanse (Aasen, Waaler & Vinje, 2018). I utdanningsforsknings blogg legger Melby-Lervåg (2018) en større undersøkelse til grunn (Scherer, Siddiq & Sánchez Viveros, 2019) når hun konkluderer med *«mer et tja enn ja»* på spørsmålet om elever skal lære programmering og koding i skolen.

Debatten viser uten tvil mange gode intensjoner og muligheter, men også utfordringer, spesielt for lærere i skolen. Noen av disse utfordringene kan være å definere hvilken rolle programmering skal ha i skolen generelt, og hvilken kompetanse det er behov for hos lærere som skal undervise. Andre handler om hvordan påvirkes læreres didaktiske valg påvirkes av å implementere et nytt fag – en viktig, men kompleks teknologi – inn i et allerede eksisterende matematikkfag og hvordan dette påvirker samspillet i klasserommet. Dette er sentrale utfordringer som det er viktige å diskutere og studere nærmere for å generere best mulig evidens- og kunnskapsbase for fremtiden. I tillegg til utfordringene vises det også til mange potensielt positive effekter ved innføring av programmering i skolen, noe som utdypes nærmere i kapittel 1.2.

Formålet med denne masteroppgaven er å se på hvordan spesifikke emner i matematikkfaget (her: utforsking av delelighet og rest) kan undervises ved bruk av et programmeringsverktøy (her: Python). Med bakrunn i innsamlede data ønsker jeg å belyse både muligheter og utfordringer knyttet til dette. Før jeg går inn på selve oppgaven, ønsker jeg å kort gi en historisk kontekst for valg av tema, samt gi en innføring i ulike programmeringsverktøy, med hovedfokus på Python.

1.1.1 Historisk kontekst

Programmering i skolen og implementering gjennom læreplaner kan sies å ha foregått i bølger (Tedre & Denning, 2016). På 80-tallet ble programmering proklamert som et viktig redskap for å lære matematikk gjennom praktisk anvendelse (Feurzeig, Papert & Lawler, 2011; Papert, 1980). På tross av stor optimisme ble aldri programmering implementert i stor skala, verken i matematikkundervisningen eller i skolen for øvrig. Popat og Starkey (2019) viser til fokus på bruk av tekstredigeringsprogramvare og søkemotorer som noe av forklaringene til hvorfor programmering ikke ble mer populært i skolesammenheng. Det hevdes også at en av årsakene til hvorfor LOGO-programmering, et av programmeringsspråkene som ble forsøkt, ikke ble allment brukt var vansker med å gjøre matematikken eksplisitt hos elevene (Misfeldt & Ejsing-Duun, 2015). Benton, Hoyles, Kalas og Noss (2016) viser også til utfordringer med et allerede stort pensum, manglende lærerkompetanse og vanskelig syntaks som avgjørende for hvorfor LOGO ikke ble allment i skolesammenheng.

I løpet av det siste tiåret har det igjen blitt et stadig større fokus på bruk av digitale verktøy i skolen og det er på nytt argumentert for å integrere programmering i skolen, enten som eget fag, eller integrert i andre fag (Bocconi, Chioccariello & Earp, 2018). Ulike Argumenter brukes for å få økt fokus på programmering i dagens skole. Blant annet nevnes behov for arbeidskraft i en verden med et stadig økende krav om digital kompetanse, endret syn på elever som produsenter av egen kunnskap og påstander om læring av generelle ferdigheter som en «bi-effekt» av selve kodingen (Popat & Starkey, 2019). Dagens situasjon hevdes å være helt annerledes enn med tidligere, og karakteriseres av lettere tilgang til verktøy, totalt forandret samfunnssituasjon og økt fokus på digital kompetanse i læreplanene (Sevik, 2016).

1.1.2 Nye læreplaner fra 2020

Etter flere høringer ble nye læreplaner publisert høsten 2019 (Utdanningsdirektoratet, 2019b). Det tidligere bestemt at algoritmisk tenkning (AT) og programmering skulle være del av kjerneelementene i matematikk (Udirbloggen, 2017) og matematikkfaget pekt på som det mest egnede faget til å ha hovedansvar for programmeringsinnhold. Det er imidlertid påpekt at en slik innføring må skje på matematikkfagets premisser (Udirbloggen, 2017). I tillegg til å nevnes ekplisitt i matematikkfagets kjerneelementer, er programmering en del av kompetansemålene i det nye matematikkfaget (Utdanningsdirektoratet, 2019b). I

kompetansemålene etter 7. trinn står det blant annet at mål for opplæring er at eleven skal kunne *«bruke programmering til å utforske data i tabellar og datasett»* (Utdanningsdirektoratet, 2019b). Å vite hvordan man skal undervise for å nå et slikt kompetansemål er på ingen måte gitt, og det er heller ikke sikkert hvilken kompetanse som kreves for å gjøre dette på en god måte.

1.1.3 Algoritmisk tenkning - 21. århundrets kompetanse

Et av argumentene for å innføre programmering i skolen viser til kompetansebehov for fremtiden, både innenfor arbeidslivet og akademia. Mer spesifikt blir AT løftet frem som et begrep som dekker disse behovene. I læreplanen beskrives AT som det «*å bryte ned eit problem i delproblem som kan løysast systematisk. Vidare inneber det å vurdere om delproblema best kan løysast med eller utan digitale verktøy*» (Utdanningsdirektoratet, 2019b). En utvidet definisjon på begrepet AT diskuteres av Gjøvik og Torkildsen (2019). De hevder at AT går ut over bare å bryte ned et problem (dekomponering), til også å omfatte generalisering, abstraksjon, automatisering og algoritmebehandling. Dette gjør at AT kan sees som et videre og mer innholdsrikt begrep, og sammenfaller da mer med det engelske begrepet *computational thinking* (CT) slik det brukes av Bocconi et al. (2018).

En slik vid tilnærming støttes også av Wing (2006), som beskriver CT som en universell holdning og et sett ferdigheter. Dette inkluderer å tenke abstrakt, kunne løse komplekse problemer, bruke matematisk tenkning i praksis og kommunisere med andre. I tillegg presiseres det at CT er en menneskelig måte å tenke på, som innebærer å *bruke* datamaskiner fremfor å *tenke* som en datamaskin, og at CT er en helt nødvendig ferdighet i et moderne samfunn (Wing, 2006). Det hevdes at arbeidsliv og akademia vil ha behov for fundamendalt annerledes kompetanse i fremtiden, og at denne kompetansen vil være preget av nettopp *CT* (Wing, 2006). Også i et notat om programmering i skolen (Sevik, 2016) blir AT beskrevet som en 21. århundres kompetanse, og programmering fremheves som en måte å oppnå dette på. Slik kompetanse blir sett på både som relevane både for elevenes fremtid og som noe som kan bidra til dypere fagspesifikk kompetanse i matematikk. Det blir hevdet at programmering også kan bidra til å styrke de sentrale fagområdene i matematikk ved at programmering gir matematisk innhold en kontekst, som igjen kan bidra til økt forståelse (Sevik, 2016).

1.1.4 Generelle ferdigheter – Higher Order Thinking Skills (HOTS)

Foruten AT som en nødvendig ferdighet i fremtidens samfunn, hevdes det at programmering som aktivitet er med på å utvikle en del generelle ferdigheter som blant annet problemløsning (Wing, 2006). Slike påstander krever nyansering. Popat og Starkey (2019) viser til mulighet for læring av generelle ferdigheter gjennom koding, som matematisk problemløsing, kritisk tenkning, sosiale ferdigheter, selv-regulering og akademiske ferdigheter, men kvaliteten på instruksjonen og oppgavedesign identifiseres som kritiske for at disse effektene skal realiseres. Det hevdes videre at ferdigheter knyttet til matematisk problemløsning er et mulig utfall når man lærer seg å kode (Popat & Starkey, 2019). Dette synet støttes av Scherer et al. (2018) som viser til at det å lære programmering er assosiert med økt kreativitet, matematiske ferdigheter og logisk resonnering. Johan, Inge Olav og Tamsin Jillian (2017) viser til en kasus-studie hvor programmering blir sett på som potensielt fruktbart for matematisk problemløsning, spesielt knyttet til matematiske kompetanser som resonnering, samhandling og modellering. Til tross for flere rapporter som setter programmering i et lovende lys, påpeker Popat og Starkey (2019) at dersom matematisk problemløsning er målet, finnes det mer effektive måter for elever å oppnå disse ferdighetene. Et viktig spørsmål i denne sammenhengen er hvorvidt matematisk problemløsning og koding som aktivitet kan kombineres på en konstruktiv måte i en undervisningssammenheng?

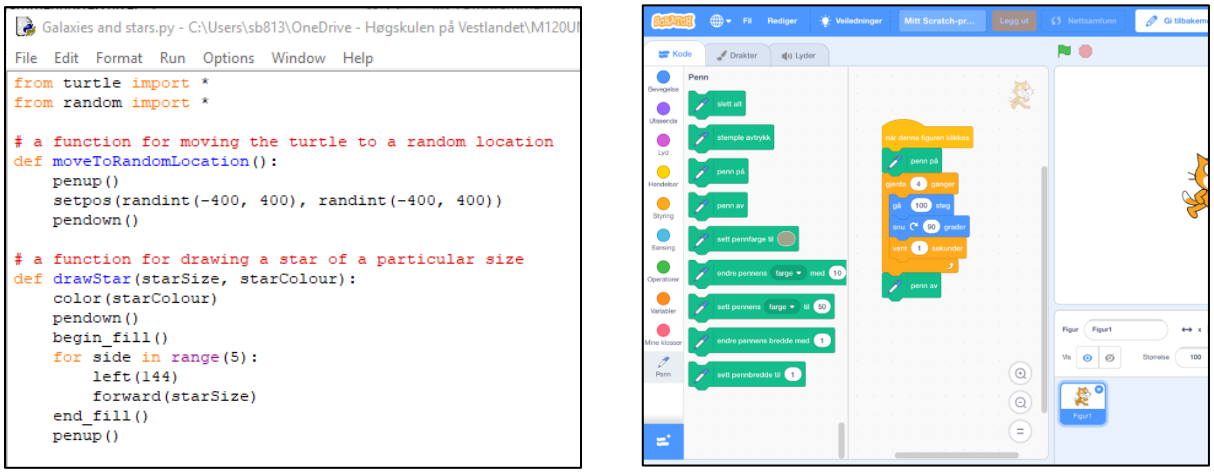
1.2 Programmering

1.2.1 Hva er programmering?

Programmering handler om å lage instruksjoner som løser et problem på et språk datamaskiner forstår (Bueie, 2019). Instruksjonene, eller oppskriften, kalles en *algoritme*. Er oppskriftene skrevet ned i et spesifikt programmeringsspråk, brukes gjerne betegnelsen *kode*. Det finnes flere forskjellige programmeringsspråk med ulike egenskaper. For eksempel er apper til android telefoner skrevet i Java, mens nettsider skrevet i en kombinasjon av HyperText Markup Language (HTML)-kode og JacaScript. Innen skole- og utdanning er det flere programmeringsspråk som er aktuelle, blant annet Scratch, Blockly, Python, Processing og Javascript. Noen av disse er blokkbaserte programmeringsverktøy som Scratch og Blockly.

Her lages programmer ved å sette sammen blokker med ulike funksjoner på et lerret. Andre programmeringsspråk er tekstbasert hvor man skriver inn kode selv.

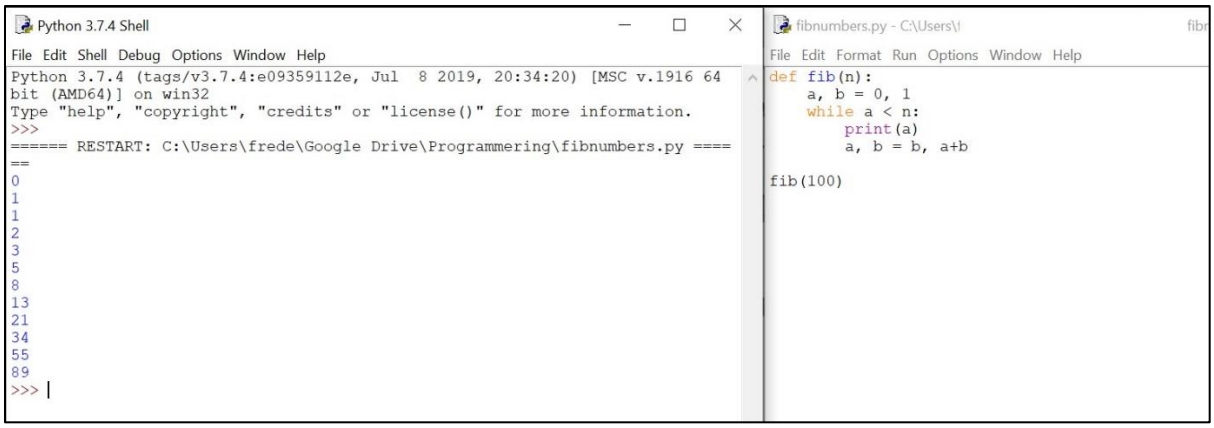
Fordelene med å bruke blokkbasert programmering kan virke åpenbare, blant annet ettersom terskelen for å lage et fungerende program senkes betraktelig ved at man simpelthen drar blokker inn i skriptet (se figur 1). Dersom målgruppen er yngre barn og aktiviteten handler om å lage program fra bunnen, er for eksempel Scratch et naturlig valg. Ved gjennomgang av aktuell litteratur (presentert nedenfor) ser man at de fleste prosjekter knyttet til programmering i matematikkfaget på lavere årstrinn er knyttet til blokkprogrammering (primært Scratch). Tekstbaserte programmeringsspråk hevdes derimot å gi gode muligheter i matematikkfaget, spesielt til temaer som omhandler tallbehandling (Bueie, 2019; Saha, 2015). Et slikt aktuelt tekstbasert språk er Python som er valgt som utgangspunkt for arbeid i denne masteroppgaven.



Figur 1: Eksempler på ulike programmeringsspråk. Python (t.v.) og Scratch (t.h.).

1.2.2 Python – et tekstbasert programmeringsspråk

Python er et programmeringspråk som beskrives ved at det har lav inngangsterskel, stor takhøyde, og er godt egnet til å arbeide med tall (Bueie, 2019). Python er også åpent tilgjengelig, gratis og et allsidig tekstbasert programmeringsspråk som er mye brukt over hele verden, også innen utdanning. Spesielt i videregående skole og ungdomsskole pekes det på Python som et naturlig valg når det i nye læreplaner skal jobbes spesifikt med programmering (Bueie, 2019), men ser man til andre ressurser som kidsakoder.no er yngre elever tydelig i målgruppen.

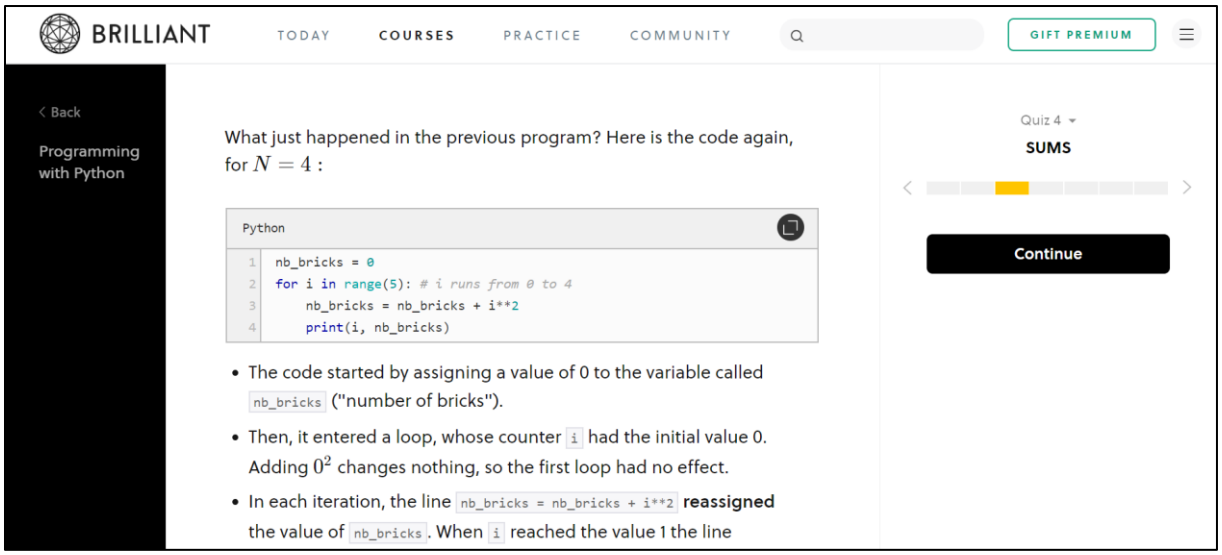


Figur 2: Python editor og python skall.

Editoren (til høyre) inneholder instruksjoner (kode). Når programmet kjøres, vises output i skallet (til venstre). Programmet som vises, skriver ut fibonacci-rekken opp til 100.

Innhold i ulike læringsressurser knyttet til Python viser tydelig at delelighet, faktorisering, primtall og tallrekker er emner som kan være passende å utforske ved hjelp av nettopp dette verktøyet. Typiske oppgaver kan innebære å finne primtall, skille ut partall/oddetall fra lister, sortere tallrekker, skrive ut fibonaccitall (Figur 2), regne ut summer av tallrekker eller lignende. Flere nettsteder presenterer oppgavene i et spesifikt format, ofte som en blanding av tekstbaserte oppgaver, medier som filmer/bilder og kodeeditor (Figur 3). Det er også vanlig at oppgavene blir presentert i sekvenser som følger en spesifikk progresjon. Flere forlag ser også ut til å bruke dette formatet i forbindelsen med fagfornyelsen[1]. Ser man til tidligere nevnte kompetansemål etter 7. trinn (Utdanningsdirektoratet, 2019b) om å utforske data i tabeller og datasett, tenker jeg at Python er godt egnet til nettopp dette.

[1] For eksempel Aschehoug som har laget et kurs i Python-programmering beregnet på ungdomstrinnet, https://aunivers.lokus.no/fagpakker/realfag/programmering/python



Figur 3: Eksempel fra nettkurs i Python-programmering. https://www.brilliant.org/Python

1.2.3 Digitalt miljø og bruk av teknologi i matematikkundervisning

Sfard og Leron (1996) beskriver egenskaper ved digitale miljø (programmering) som fremmende for et mer «utforskende tankesett». Sammenlignet med blyant og papir, er et digitalt miljø responsivt, og ser i større grad ut til å ufarliggjøre prøving og feiling. I det tradisjonelle klasserommet kan det å gjøre feil ofte oppfattes som skummelt og negativt. I et digitalt miljø er imidlertid det å gjøre feil ofte en helt naturlig del av prosessen. Prøving og feiling, gjetning og antakelser hevdes å være konstruktive prossesser som hjelper elevene med å utvikle problemløsningsferdigheter. Datamaskinen (et programmeringsmiljø) kan derfor sees på som et godt miljø for slik utforsking (Sfard & Leron, 1996).

Olive et al. (2009) argumenterer for at fremvekst av teknologi i klasserommet medfører endringer i hva matematisk aktivitet innebærer. Det vises til matematisk aktivitet som muliggjøres gjennom bruk av teknologi. Dette innebærer blant annet en styrket relasjon mellom kunnskap og handling, et skifte fra lærersentrert til elevsentrert undervisning. Disessa (2018) hevder at endringer i bruk av teknologi, blant annet gjennom programmering, fører til et skifte i hva matematisk aktivitet innebærer, og at blant annet rekkefølgen på det man underviser må revurderes. Det vises til et eksempel hvor sjetteklassinger ved hjelp av et programmeringsverktøy på en konstruktiv måte utforsket matematikk forbeholdt elever på videregående og oppover (gravitasjon og akselerasjon) (Disessa, 2018). Dette betyr ikke at målet er at yngre elever skal lære avansert matematikk på et tidlig stadie, men det illustrerer

hvordan teknologi/redskaper kan gjøre konsepter *tilgjengelig* på en annen måte enn hva man ville klart uten verktøyet. I dette eksempelet er for eksempel en rekke beregninger og notasjon overlatt til datamaskinen, mens elevene i større grad manipulerer variablene og observerer hva som skjer. Disessa (2018), i tillegg til blant annet Bray og Tangney (2017), etterspør mer forskning på hvordan teknologi brukes og hvorvidt teknologi kan være et svar på relevante utfordringer innen matematikkdidaktikk.

1.2.4 Programmering og matematikkundervisning

Ettersom likhetene mellom AT og ferdigheter som behøves i matematikk er store, argumenteres det for at innføring av programmering i skolen også kan utvikle elevenes matematiske kompetanse (Feurzeig et al., 2011). Som nevnt har digitale redskaper i flere tiår blitt sett på som potensielt gode utgangspunkt for matematikkundervisning, og særlig er dette knyttet til en mer konstruktivistisk tilnærming (Feurzeig et al., 2011; McCoy, 1996). Programmering ble for eksempel av Papert (1980) sett på som et rammeverk for å undervise algebra og problemløsing i matematikk, med spesifikt fokus på tekstoppgaver i algebra. Forsström og Kaufmann (2018) viser til økt motivasjon, økte matematikkferdigheter, økt samarbeid og en forandret lærerrolle som gjennomgående temaer og begrunnelser i litteraturen. I tråd med Papert (1980) sin formodning fra 80-tallet blir programmering på nytt fremhevet som en løsning for å gjøre matematikkfaget mer relevant for elever, men det blir samtidig påpekt at det mangler forskning på effektene avprogrammering som knyttes spesifikt til matematikkfaget i skolen.

ScratchMaths prosjektet er et større forskningsprosjekt som tar utgangspunkt i konkret pensum og verktøy (Benton et al. (2016). I prosjektet er det utviklet undervisningsopplegg i matematikk og samlet inn data i en toårsperiode for å studere hvordan man kan bruke konsepter fra datavitenskap (AT/CT) i matematikkundervisning for å lære geometri og algerbra. I prosjektet fokuseres det blant annet på spenningen mellom redskap og læring, retning og utforsking. Et konstruktivistisk rammeverk skisseres for å imøtekomme de ulike spenningene og utfordringene mellom matematikkundervisning og datavitenskap på en god måte. Basert på ideene til Papert (1980) beskriver rammeverket fem E'er som beskriver god praksis når man kombinerer matematikk og programmering i skolen; *Explore*, *Explain*, *Envisage*, *Exhange* og *bridgeE*.

I møte med et ukjente verktøy beskriver Benton et al. (2016) det som viktig å utforske (*Explore*) begrensninger og muligheter ulike funksjoner i programmeringsverktøyet gir, og oppgavene bør legge til rette for dette gjennom aktiviter. For å få frem den underliggende matematikken i elevens arbeid bør man legge til rette for elevers forklaringer (*Explain*), noe som skjer gjennom å gi passende navn til funksjoner og variabler, samt forklare kodene steg for steg og prøve å forutse hva som skjer når kodene kjøres. Å se for seg hva som kommer til å skje (*Envisage*) beskrives også som en viktig del av programmering, da dette krever inngående forståelse av ideene og konseptene som ligger til grunn i programmet. Gjennom samarbeid og deling (*Exchange*) av ideer får elevene mulighet til å utfordre egne ideer og forståelser. Dette skjer gjennom samarbeid mellom elever, spørsmål som stilles og fordrer refleksjon, og diskusjoner av underliggende konsepter i plenum (klasse). Videre påpekes det også at undervisning som tar utgangspunkt i to særegne (men noe overlappende) fagdisipliner er nødt til å adressere sammenhenger mellom de to fagene (*bridgE*). Lærerens valg av spørsmål og oppgaver må på ulikt vis være med å eksplitt fremheve disse sammenhengene. Oppsummert kan et slikt rammeverk være en nyttig indikator og mal for undervisning som kombinerer matematikk og programmering.

Emmy-Charlotte Förster, Klaus-Tycho Förster og Löwe (2018) har gjennom en pilotstudie forsøkt å integrere programmering i Scratch (computer science) i ordinær matematikkundervisning. Det beskrives som et mål å utvikle elevenes programmeringsferdigheter samtidig som man underviser matematikk, og fokuserer på det matematiske innholdet – i denne sammenheng knyttet til geometri. Forfatterne hevder at piloten har gitt positive resultater og at elevene både har nådd de matematiske målene for undervisningen, samt lært grunnleggende programmering i Scratch. De hevder også at det er mulig å inkludere aktiviteter som tar i bruk programmering i matematikkundervisning, samtidig som fokus fremdeles er på matematikkfaglig innhold (Emmy-Charlotte Förster et al., 2018).

I lys av programmering som et digitalt verktøy i matematikkundervisning er det også relevant å påpeke at det skilles mellom å se på teknologi som et pragmatisk verktøy for å øke ferdigheter (som gangetabellen) eller som redskap for å øke konseptuell forståelse (Bray & Tangney, 2017). Sistnevnte kalles et *epistemiske fokus* og er sentralt blant annet for Misfeldt og Ejsing-Duun (2015) rammeverk for å forstå elevers læring av matematikk gjennom

programmering. Forfatterne viser til nødvendighet av å holde et epistemisk fokus på matematikken fra lærers side, for at programmering i matematikken skal kunne sies å bli en suksess. Dette innebærer blant annet et kontinuerlig fokus/arbeid fra lærer hvor det rettes fokus mot de matematiske aspekter av arbeidet/objektene – spesielt i kognitivt krevende problemløsning. Lærers didaktiske tilnærming blir fremhevet som avgjørende for hvilket potensiale programmering har i matematikkundervisning. Dette fremheves også av McCoy (1996) som viser til en tilstedeværende lærer for å overvåke og lede prosessene mot det matematiske innholdet som kritiske for suksess.

1.2.5 Matematiske temaer og programmeringsspråk

At programmering er et godt utgangspunkt for arbeid med geometri, virker å være nokså etablert (Forsström & Kaufmann, 2018). Dette gjenspeiles også i mengden av studier relatert til matematikk og programmering som tar for seg emner relatert til geometri (For eksempel Emmy-Charlotte Förster et al., 2018; Khasawneh, 2009). Det finnes eksempler på prosjekter som også tar for seg andre emner, for eksempel Scratch-Maths prosjekt beskrevet ovenfor, hvor Benton, Saunders, Kalas, Hoyles og Noss (2018) dokumenterer elevers arbeid med plassverdisystemet. Basert på forskningen presentert ovenfor er det likevel tydelig at det trengs mer forskning på bruk av programmering innen andre emner i matematikken, spesielt relatert til lavere årstrinn i skolen. Flere forskere, blant annet Forsström og Kaufmann (2018) anbefaler derfor å utforske hvilken rolle programmering kan ha i andre emner i matematikkfaget, spesielt med tanke på å utvikle AT gjennom problemløsning. Kan for eksempel yngre elever jobbe med faktorisering, delelighetsregler eller parentesregning gjennom aktiviteter som inneholder programmering?

1.3 Fokus, formål og problemstilling

Med programmering som ny komponent i norsk skole generelt, og i matematikkfaget spesielt, er det en rekke nye spørsmål og problemstillinger som dukker opp. Ut fra litteraturen ser det ut til å være en utfordring å beholde fokus på nettopp matematikken i det man inkluderer aktiviterer som innebærer programmering. I tillegg er det et åpenbart fokus på emner knyttet til geometri når programmering tas i bruk i matematikkfaget og en stor del av forskningen retter seg mot blokkbasert programmering. Evidensbasen for bruk av

tekstbasert programmering og fokus på andre emner er derfor mangelfull. Problemstillingen for studien min er formulert som et åpent fokus:

Fokuset i denne oppgaven rettes mot hvilke muligheter og utfordringer tekstbasert programmering kan gi i matematikkundervisning på mellomtrinnet. Det søkes særlig innsikt i samspillet mellom elever, oppgave og programvare.

Da formålet med denne oppgaven er å bidra til økt forståelse knyttet til bruk av programmering i matematikkfaget er et slikt åpent fokus sett på som hensiktsmessig. Målet er å utforske og finne muligheter for bruk av programmering som redskap i matematikkundervisning, særlig med fokus på temaer som delelighet og rest. Ved å fokusere på samspillet mellom elever, oppgavene som stilles og programvaren (programmeringsspråk) som benyttes, kan oppgaven bidra til å belyse aspekter knyttet til hvordan man skal forstå programmering som element i skolen. Dette fokuset er videre spesifisert gjennom tre forskningsspørsmål:

1. Hva kjennetegner elevens handlinger i interaksjon med oppgave og programvare?
2. Hvordan påvirker oppgavedesignet, sammen med verktøyet, elevenes argumentasjon og resonnering?
3. Hvilken didaktisk funksjon spiller programvaren i de valgte situasjonene?

Videre presiseres personlig utgangspunkt, egen forståelse av noen sentrale begreper og det gjøres rede for valg som er tatt i forbindelse med fokus og problemstilling.

1.3.1 Avgrensninger og presiseringer

Arbeidet med masteroppgaven er et videreutdanningsprosjekt med utgangspunkt i egen hverdag som lærer og et ønske om faglig utvikling. Egne erfaringer og interesser har vært med å prege både fokus, vinkling og ikke minst valg av problemstilling. Spesielt på grunn av tidspress og «fagtrengsel» har det fra starten av vært interessant å studere hvordan man klarer å beholde et matematikkfaglig fokus *samtidig* som man tar i bruk programmering. Det har vært et ønske om å gjennomføre et prosjekt «fra grunnen av», hvor undervisningsopplegg planlegges, gjennomføres og evalueres med utgangspunkt i lærerhverdagen. Dette har krevd en åpen tilnærming hvor utforsking og utprøving av undervisningstilnærminger har stått sentralt og det er gjort valg som man i ettertid kan

kritisere eller finne bedre løsninger til. Valgene er likevel tatt, og jeg ser på det som viktig å kunne trekke lærdom fra det som er gjort.

1.3.1.1 Programmering som redskap

Hvordan et redskap benyttes i matematikkundervisningen generelt, og gjennom oppgavedesign spesielt, kan sees på som et utrykk for hvilke tanker man som lærer har på hvordan elever tar til seg eller konstruerer kunnskap (Watson & Thompson, 2015b). Ser man for eksempel på læring som *tilegnelse* eller læring som *deltakelse*? Jeg ønsker å utforske hvorvidt bruken av programmering kan sees på som et verktøy som kan legge til rette for matematisk aktivitet knyttet til argumentasjon og resonnering, noe som fordrer et sosialt aspekt (jamfør deltakelse). I tillegg er Python tenkt å fungere som et redskap med konseptuell forståelse av matematisk innhold som mål – et såkalt *epistemisk fokus*.

1.3.1.2 Oppgavedesign

Fokus på *oppgavedesign* er sett på som viktig da dette hevdes å være avgjørende for hvorvidt digitale verktøy fungerer i undervisningen slik de er tiltenkt (Drijvers, 2015). Bray og Tangney (2017) viser til fordelene ved å bruke digitale verktøy i oppgavedesign når oppgavene avhenger av verktøyet. Dersom man imidlertid like godt kunne løst oppgavene uten, mister de sitt fokus. Digitale verktøy antas i tillegg å være svært gunstig når oppgavene er designet med fokus på undersøkende virksomhet (*inquiry based*) (Bray & Tangney, 2017), men det må presiseres at begrepet utforsking i denne oppgaven er brukt noe annerledes enn *inquiry* som kan tolkes i retning av *åpne oppgaver*. Begrepet utforsking er brukt som en tilnærmingsmåte til oppgaver preget av å prøve ut, observere, leter etter mønster, eksperimentere, granske eller evaluere (Utdanningsdirektoratet, 2019b).

1.3.1.3 Elevens handlinger – matematisk aktivitet

I studien fokuserer jeg i vid forstand på elevenes handinger og interaktivitet for å forsøke å beskrive matematisk aktivitet, fremfor å beskrive elevers læring eller læringsutbytte. I tillegg undersøker jeg hvordan et programmeringsmiljø påvirker denne aktiviteten. Fører tilbakemelding fra programmet til gode matematiske diskusjoner? Bruker elevene koden, output eller oppgaveteksten når de argumenterer? I hvilken grad lar det seg gjøre, gjennom oppgavedesign og bruk av spesifikk programvare, å legge til rette for at elevene deltar aktivt i argumentasjon? Ved å se på samspillet mellom de ulike aktørene og elementene kan jeg

ikke si noe direkte om elevers læring, men man kan postulere at aktiviteter som inneholder for eksempel argumentasjon *legger til rette for* elevers læring (Brousseau, 2006).

Da det spørres hvordan oppgavedesignet, sammen med verktøyet, påvirker elevenes argumentasjon og resonnering er dette først og fremst et utgangspunkt for å kunne si noe om elevenes matematiske aktivet og kvaliteten på denne interaksjonen slik den manifisterer seg mellom elev, oppgave og programvare. For å holde et spisset fokus er det tatt utgangspunkt argumentasjon og resonnering, en viktig del av matematikkfaget og et av kjerneelementene i ny læreplan for matematikk (Utdanningsdirektoratet, 2019b). Klassiske rammeverk knyttet til argumentasjon i matematikkundervisning, blant annet Lithner, Lavy, Toulmin og Krumheuer, legges derimot til side og det taes utgangspunkt i argumentasjon og ressonering slik det beskrives av Brousseau og Gibel (2005). Dette blir gjennomgått i kapittel 2.

1.3.1.4 Didaktisk funksjon

Det spørres også hvilken didaktisk funksjon programvaren spiller i de valgte situasjonene. Dette innebærer hvordan programmeringsverktøyet blir brukt som redskap for at elevene skal sitte igjen med ønsket matematisk kunnskap i slutten av undervisningssituasjonen. På hvilken måte bidrar programmeringsverktøyet i elevenes arbeid med å nå målet for undervisningen og hvilken rolle spiller programmeringsverktøyet relatert til oppgavedesignet i praksis?

1.3.1.5 Tekstbasert programmering

Valget om å bruke tekstbasert programmering er rotet i et ønsket om å fokusere på andre emner i matematikken, egen kompetanse og tidligere erfaring med å bruke Python med elever på 7. trinn. Sistnevnte indikerte et potensiale i mer ordinær undervisning som jeg ønsker å se nærmere på i denne oppgaven. Da flere bøker som tar for seg programmering i Python hovedsakelig retter seg mot ungdomsskole og videregående skole (for eksempel Downey, 2017; Saha, 2015), er det funnet lite konkret materiale som kan brukes når en slik tilnærming skal prøves ut. Utgangspunktet har derfor vært å bruke inspirasjon fra nettsteder som kidsakoder.no sammen med egen erfaring som matematikklærer og lage et opplegg jeg tror kan virke.

Med en visshet om at tekstbasert programmering kan være krevende og innebærer utfordringer, vil jeg med denne oppgaven prøve å belyse hvorvidt disse utfordringene er overkommelige og hvilke mulighetene en slik tilnærming gir. Det er i den sammenheng viktig å presisere at analysen av elevpar som finner sted belyser *spesifikke utfordringer og muligheter* med de gitte oppgavene i den *aktuelle konteksten*.

1.4 Del av et forskningsprosjekt

Denne masteroppgaven er en del av forskningsprosjektet *Learning about teaching argumentation for critical mathematics education in multilingual classrooms* (LATACME). Forskningsprosjektet fokuserer på hvordan lærerstudenter lærer og underviser i argumentasjon og kritisk matematikk undervisning i flerspråklige klasserom. Denne oppgaven tilhører undergruppen argumentasjon og IKT som blant annet utforsker hvordan digitale læremidler størrer elevers utvikling av kritisk-matematisk argumentasjon.

1.5 Disposisjon

I kapittel 1 gjøres det rede for valg av tema og problemstilling, samt relevant forskning på temaet. Nye læreplaner, fokus på algoritmisk tenkning, og en generelt større interesse for bruk av digitale verktøy (programmering inkludert) det siste tiåret beskrives som bakgrunn for prosjektet.

Det teoretiske rammeverket som danner grunnlag for oppgavens videre analyse er presentert i kapittel 2. Sentrale begreper fra Brousseau (2006) sin Teori om Didaktiske Situasjoner (TDS) utdypes, spesielt de ulike situasjonsformene aksjons-, formulerings- og valideringssituasjoner. I tillegg presenteres et alternativt perspektiv, SAMR-modellen.

Metoden som er brukt for å besvare forskningsspørsmålene og for å belyse fokuset i oppgaven presenteres i kapittel 3. Det legges her vekt på beskrivelser av prosessen fra tidlig planlegging, til datainnsamling og analyse, samt valg som er gjort i tilknytning til oppgavedesign.

Resultatene fra utvalgte kasus presenteres i kapittel 4 gjennom seks utdrag fra elevers arbeid med oppgaver. Totalt tre elevpar i arbeid med henholdsvis to oppgaver hver (hvorav

noen delvis overlapper) er tatt med. Utdragene er ikke presentert kronologisk men knyttet til de respektive elevparene.

I kapittel 5 diskuteres resultatene opp mot det teoretiske rammeverket samt annen relevant litteratur. Analysene brukes for å besvare forskningsspørsmålene stilt innledningsvis, samt refleksjoner rundt muligheter og utfordringer med valgte tilnærming til matematikkundervisning.

Til slutt, i kapittel 6, avrundes oppgaven med refleksjoner rundt studiens relevans og behov for videre forskning.

2 Teoretisk rammeverk

Målet med rammeverket er å skape et utgangspunkt hvorfra datamaterialet kan forstås, samt å belyse fokuset og svare på forskningsspørsmålene som stilles. Som beskrevet innledningsvis er møtet mellom elev, oppgave og et programmeringsverktøy det primære fokuset i denne oppgaven – dette er analyseenheten. For å kunne beskrive og sette ord på samspillet mellom disse elementene er det tatt utgangspunkt i Brousseau (2006) sin *Teori om Didaktiske Situasjoner (TSD)*, en teori som fokuserer på hvordan matematisk kunnskap bygges opp i et samspill mellom eleven, lærer og omgivelsene. Videre er det inkludert et perspektiv knyttet til hvordan verktøyet blir brukt i undervisningsopplegget – hvilken matematikk elevene blir bedt om å gjøre og hva som er overlatt til programvaren. Dette spørsmålet belyses ved å bruke en modell utviklet av Puentedura (2006) med begrepene Substitution, Augmentation, Modification, and Redefinition, også kalt SAMR-modellen.

2.1 Teorien om didaktiske situasjoner (TDS)

> *«A student isn't really doing mathematics unless she is asking herself questions and solving problems» (Brousseau, 2006, s. 79)*

TDS setter søkelys på hvordan matematisk kunnskap utvikles og undervises, samt forhold som spiller en rolle i denne prosessen (Brousseau, 2006). Ifølge Brousseau må didaktikken først og fremst ta utgangspunkt i det matematiske målet, fremfor å være generell didaktikk som tilpasses. Ifølge Artigue, Haspekian og Corblin-Lenfant (2014) er TDS først og fremst rettet mot å forstå under hvilke forhold didaktiske systemer legger til rette for eller hindrer elevers læring og hvordan slike systemer kan utvikles og forbedres. Dette inkluderer blant annet studier av situasjoner som legger til rette for læring av spesifikk matematisk kunnskap (fundamentale situasjoner). Som teoretisk rammeverk bygger TDS på blant annet kognitiv teori (Kieran, 2019), noe som tydelig preger beskrivelsene av begreper som *læring* og *kunnskap*. Det sosiale samspillet mellom ulike aktører i undervisning kommer likevel tydelig frem og utgjør sentrale elementer i teorien, spesielt gjelder dette mellom lærer, elev og omgivelsene (*milieu*).

Bruk av TDS som teoretisk ramme i denne oppgaven baserer seg nettopp på et behov for et begrepsapparat knyttet til samspillet mellom elev og omgivelsene. I dette tilfellet er

omgivelsene først og fremst knyttet til oppgavene, digital programvare (programmering) og medelever. TDS kan gi et nyttig perspektiv for å si noe om elevers handlingsmønstre og matematiske aktivitet (argumentasjon og resonnering) i samhandling med gitte oppgaver og programmeringsverktøy.

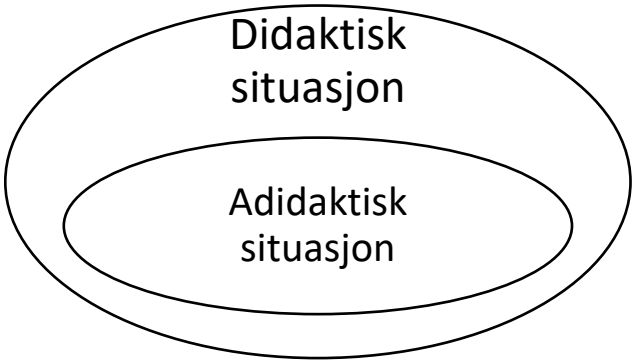
Ettersom jeg først og fremst fokuserer på forholdet mellom elev og omgivelsene (milieu) i spesifikke avgrensede situasjoner fremfor matematikkundervisning generelt (lærer-elev-omgivelser-tid), har jeg valgt å bruke deler av rammeverket for å belyse det som er mitt fokus, fremfor å beskrive TDS i sin helhet. Dette utvalget er først og fremst inspirert av Joubert (2017) som tar i bruk Brousseau (2006) sine begreper knyttet til *modes of production*, videre referert til som spesifikke didaktiske situasjoner: aksjon-, formulering- og validering, samt *motstand* (*obstacles*). Begreper knyttet til undervisning på et overordnet plan, som *den didaktiske kontrakt* (samspill og forventninger mellom lærer og elev) og *institusjonalisering* (en fase i undervisning) nevnes, men fokuseres mindre på. Valget om å kun beskriver deler av begrepsapparatet fra TDS er gjort for å kunne holde et spisset fokus mot det som er oppgavens primære mål; samspillet mellom elever, oppgave og programvare. Videre følger en gjennomgang av de begrepene som er ansett som mest sentrale for å kunne belyse et slikt perspektiv: didaktiske og a-didaktiske situasjoner, aksjons-, formulerings- og valideringssituasjoner, dialektikk, og motstand. I tillegg er det valgt å se nærmere på Brousseau og Gibel (2005) sine beskrivelser av elevers argumentasjon og resonnering i didaktiske situasjoner, samt noen refleksjoner knyttet til bruk av TDS som rammeverk.

2.1.1 Sentrale begreper

2.1.1.1 Didaktiske- og adidaktiske situasjoner

Didaktiske og a-didaktiske situasjoner spiller en viktig rolle i forståelsen av samspillet mellom lærer, elev og utvikling av kunnskap i TDS. *Didaktiske situasjoner* kan sees både som et begrep for undervisningssituasjoner på et overordnet plan, og for å referere til spesifikke underkategorier (*subsystemer*) – typer situasjoner som kan oppstå avhengig av omgivelsene, for eksempel oppgaven(Brousseau, 2006). Når elevene overlates til oppgaven, uten lærers interaksjon, bruker Broussau begrepet *a-didaktisk situasjon* og det er ifølge Brousseau (2006) først her grunnlaget for konstruksjon av kunnskap (læring) skjer. De to ulike situasjonsformene må sees i sammenheng, hvor a-didaktiske situasjoner er en underkategori

(*subsystem*) av den større helheten som er undervisningssituasjonen (*den didaktiske situasjonen*) (Figur 4).



Figur 4: Didaktiske og A-didaktiske situasjoner

Brousseau (2006) beskriver didaktiske situasjoner generelt som et samspill mellom lærer, elev og oppgaven (*problemer*) i kontekster hvor kunnskap og læring er målet. Undervisning sees på som et forsøk på å legge til rette for elevers deltakelse og involvering i en faglig kontekst, hvor elevens konstruksjon av kunnskap avhenger av hvordan man legger til rette for og presenterer problemer på en måte som tillater dette. For at dette samspillet skal fungere optimalt er det ifølge Brousseau (2006) viktig at elevene aksepterer utfordringene oppgavene tilbyr som sine egne, og er indre motivert til å løse oppgavene og diskutere mulige løsninger. En slik aksept og anerkjennelse av et problem kalles *matematisk motivasjon*. Dette kan sees i kontrast til *didaktisk motivasjon* hvor elever handler motivert av ytre mål, som om å «bli ferdig», «få rett svar», «få ros fra lærer» eller «gjøre som man får beskjed om» (Brousseau, 2006).

Ifølge Brousseau (2006) vil det med ulike former for matematisk kunnskap kunne lages oppgaver/problemer (derav også situasjoner) som ved å løses krever at eleven tar i bruk kunnskapen som forsøkes å formidles. Han kaller dette for *fundamentale situasjoner*, og ser på disse som mer abstrakte ideer av problemer i en undervisningskontekst. All matematisk kunnskap vil kunne knyttes til slike fundamentale situasjoner, og ifølge Artigue, Haspekian og Corblin-Lenfant (2014) kan et eksempel Brousseau (2006) bruker, «The race to 20» (Utdypes nærmere i kap 2.1.2.2), beskrives som en fundamental situasjon for euklidsk divisjon –

divisjon med heltall og rest. For å kunne «vinne» spillet, krever det at man tar del i en matematisk prosess som inkluderer utvikling av teorier (formulering) og argumentasjon av teorienes gyldighet (validering). En slik form for deltakelse vil ifølge Brousseau (2006) resultere i matematisk kunnskap knyttet til divisjon.

Oppgavene (problemene) som gis må derfor ta utgangspunkt i den målkunnskapen man ønsker eleven skal sitte igjen med, og eleven må ved å løse oppgaven bruke denne kunnskapen og gjøre den til sin egen.

> *«The modern conception of teaching therefore requires the teacher to provoke the expected adaptation in her students by a judicious choice of "problems" that she puts before them. These problems, chosen in such a way that students can accept them, must make the students act, speak, think, and evolve by their own motivation» (Brousseau, 2006, s. 30)*

Oppgavene, og ikke minst oppgavedesignet, er med andre ord en kritisk faktor for elevers tilegnelse av matematisk kunnskap slik Brousseau (2006) ser det. Gode oppgaver må ta utgangspunkt i spesifikk matematisk kunnskap og fordre motiverte fagrelaterte handlinger, tanker og samtale hos eleven. Gjennom analyse av oppgavene må også elevenes handlinger og reaksjon i møte med oppgavene til en viss grad reflekteres over på forhånd, dette for å kunne forutse elevenes reaksjoner på oppgavene. Ikke ulikt det Smith og Stein (2018) kaller *anticipating*, som går på å forestille seg mulige elevresponser på en oppgave, for eksempel ved å løse den på så mange måter man klarer. Læreren er med dette ett steg foran elevene, og kan lettere tilpasse oppgavene videre slik at elevenes misoppfatninger eller problemer adresseres.

Brousseau (2006) beskriver det også som et mål for undervisningen å fjerne elementer som forstyrrer elevenes selvstendige utvikling av kunnskap for på denne måten å bidra til at eleven heller er motivert av et ønske om å forstå og løse problemet i seg selv. For at dette skal kunne skje, kan man ikke simpelthen «overføre» kunnskap til eleven gjennom demonstrasjon (jf. Tabula Rasa eller «tradisjonell» undervisning), men eleven må selv, gjennom nøye planlagt undervisning, få mulighet til å konstruere kunnskap nødvendig for å løse problemet/situasjonen som gis på en tilfredsstillende måte. Dette kan videre bidra til å fostre et selvstendig og konstruktivt samspill mellom elev og mål-kunnskapen.

I disse beskrivelsene av undervisning som delt inn i situasjoner ligger en forståelse av selvstendig problemløsning som essensielt for tilegnelse av kunnskap (læring). Undervisning handler med andre ord om å analysere den matematiske kunnskapen man ønsker at elevene skal utvikle for deretter å lage oppgaver, problemer (situasjoner) som krever bruk av denne kunnskapen for å lykkes. Som Brousseau (2006, s. 31) skriver: «*In modern didactique, teaching is the devolution to the student of an adidactical, appropriate situation; learning is the student's adaptation to this situation*» Her kommer det epistemologiske fokuset nevnt innledningsvis mer tydelig frem – kunnskapen man ønsker å fokusere på, og metodene man ser som gunstige for å erverve denne kunnskapen, må analyseres nøye og legges til grunn i valg av oppgaver og problemer som gis elevene.

2.1.1.2 Devolusjon, didaktisk kontrakt og institusjonalisering

Det at konstruksjon av kunnskap ifølge Brousseau (2006) først og fremst skjer i a-didaktiske situasjoner, må ikke forstås som at ansvar for læring og utvikling av kunnskap kun faller på eleven. Snarere tvert i mot, det tydelige fokuset på «problemet» i TDS, kombinert med rollen lærer har for å presentere problemet på en måte som elevene aksepterer, viser at undervisning er et gjensidig samspill mellom flere aktører (Brousseau bruker begrepet *devolusjon* for å beskrive denne prosessen). Lærers valg av oppgaver og situasjoner som presenteres for elever er sjeldent tilfeldig, noe man kan anta at elevene også er inneforstått med. Elevenes forventning fra lærer om å tilby situasjoner som gir grunnlag for kunnskapsutvikling, kombinert med lærers forventning til elevene om å involvere seg personlig i disse situasjonene kalles *den didaktiske kontrakt* (Brousseau, 2006, s. 31).

For å sette a-didaktiske situasjoner i et større perspektiv, brukes begrepet *institusjonalisering*, en prosess for å sette kunnskapen elevene utvikler i en større kontekst. Det handler om å bruke elevresponsen fra undervisningen til å endre kunnskapen fra situasjonen til noe elevene kan ha bruk for også utenfor situasjonen (Brousseau, 2006). Målet er å inkludere elevene i et fellesskap (matematikk-historisk) hvor egen kunnskap og egne erfaringer blir satt i en større sammenheng og tilpasses det formelle matematiske språket og kulturen. Institusjonalisering kan sammenlignes med det Smith og Stein (2018) kaller *connecting*; prosessen med å trekke linjer mellom ulike elevers løsninger og grunnleggende matematiske konsepter. Her kan læreren sees på som en sentral aktør for å hjelpe elever å knytte det hele sammen. Da utdragene som analyseres i denne

masteroppgaven først og fremst er avgrensede tilfeller som retter seg mot oppgaveløsning, og datamaterialet ikke inneholder felles gjennomganger eller oppsummering, vil det være mindre behov for nærmere utdyping. Lærerens rolle i datamaterialet slik det presenteres er preget av veiledning og teknisk hjelp. Det er likevel relevant å trekke frem institusjonalisering som begrep, da dette kan knyttes til flere av utfordringene som påpekes i tilknytning til matematikk og programmering generelt som å skape sammenheng mellom matematikkfaglig innhold og programmering. Dette er utfordringer som også adresseres i forbindelse med eget datamateriale.

2.1.2 Elevers handling i didaktiske situasjoner

Tre sentrale begreper for denne oppgaven beskrives av Brousseau i «The race to 20», hvor ulike faser av et spill blir brukt som utgangspunkt for å illustrere spesifikke didaktiske situasjoner: *aksjons-*, *formulerings-* og *valideringssituasjoner*. Brousseau (2006, s. 62) beskriver de ulike didaktiske situasjonene som handlingsmønstre mellom eleven og elementer i omgivelsene. Situasjonene kan kort beskrives som deling av informasjon uten språk, direkte handlinger (aksjon), deling av informasjon gjennom språk (formulering) og deling av vurderinger gjennom språk (validering). Disse begrepene utdypes i kapittel 2.1.2.2 til 2.1.2.4.

Det er også en tett sammenheng mellom de ulike didaktiske situasjonene og elevers argumentasjon og ressonnering. Brousseau og Gibel (2005) bruker begrepet *reasoning*, som jeg har tolket som en samlebetegnelse på både resonnering og argumentasjon, hvor argumentasjon kan sees som en integrert del av – men mer spesifikk form for resonnering med mål om å overbevise (Hanna, 2014). Det er først og fremst kvaliteter i elevers argumentasjon som knytter utsagnene til de ulike situasjonsformene, men som Brousseau og Gibel (2005, s. 17) påpeker, må man også ta situasjonen i betraktning når man vurderer kvaliteten på elevers argumentasjon. Brousseau (2006) mener at det kan være elevenes argumentasjon som avgjør hvorvidt en situasjon er preget av aksjon- formulering- eller validering. Til sammen tolker jeg dette som et gjensidig forhold hvor beskrivelser av situasjoner kan være medvirkende til hvordan man skal tolke elevens argumentasjon, men også hvor elevers argumentasjon er med å klassifisere situasjoner.

2.1.2.1 «The race to 20» – et eksempel

Spillet i seg selv er enkelt og finnes i flere varianter[2]. Det går kort fortalt ut på å spille mot hverandre med mål om å være den første som sier tallet 20. Man starter med å telle fra 0, og hver runde legge spillerne til enten 1 eller 2. Dersom den første spilleren begynner med 2, kan den andre for eksempel velge å legge til 1, slik at totalsummen er 3. Slik fortsetter det helt til en av spillerne legger til 1 eller 2 slik at totalsummen blir 20 – den spilleren som gjør dette, vinner. Utfordringen ligger i å finne strategier som «alltid» vinner. Spillet kan sees på som en fundamental situasjon for euklidsk divisjon ved at vinnerstrategier kan relateres til mer generelle egenskaper med divisjon og delelighet. For å vinne spille, ved å formulere en vinnende strategi, må dette gjøres med bakgrunn i kunnskap om divisjon og delelighet – kunnskap som regnes som sentralt i denne spesifikke undervisnings-situasjonen.

2.1.2.2 Aksjonssituasjoner

Etter at lærer har gitt instruksjoner og forsikret seg om at elevene forstår reglene, starter en fase hvor elevene spiller mot hverandre. Denne fasen er preget av det Brousseau (2006) beskriver som *aksjonssituasjoner*; eleven spiller spillet og utfører handlinger. Elevene spiller mot hverandre, gjør seg kjent med reglene og mekanismene i spillet, og utforsker hvordan man kan vinne mot den andre eleven. En aksjonssituasjon beskrives av Brousseau (2006) som et forløp handlinger mellom en elev og *relevante elementer i omgivelsene*. Brousseau (2006) bruker begrepet *milieu*, som kan bety medelever, oppgaven, læreren eller ulike verktøy – det i omgivelsene som er av betydning for undervisningen og elevens utvikling av kunnskap. Kort fortalt kan man si at eleven handler ut fra omgivelsene og responsen som blir gitt fra omgivelsene. Dette forløpet sees på som en prosess hvor elevens handlinger danner grunnlag for implisitte og eksplisitte strategier knyttet til det matematiske innholdet i situasjonen. Eleven lærer *av* og *i* situasjonen, gjennom handling. Aksjonssituasjoner preges av at elever handler direkte mot og og blir påvirket av de umiddelbare omgivelsene rundt seg, på en måte som inkluderer det matematikkfaglige innholdet.

Det å umiddelbart finne en løsning på et problem, kan ifølge Joubert (2017) være tegn på en aksjonssituasjon. Hun illustrerer dette med elevers bruk av grafisk kalkulator; hvis elever blir bedt om å tegne en graf, vil det å umiddelbart plotte inn funksjonen på en grafisk kalkulator

[2] En variant fra matematikksenterets oppgavebank, mattelist: https://www.mattelist.no/65

tilsvare en aksjonssituasjon – eleven reagerer med en handling. Basert på forforståelse og tidligere innlærte strategier trer eleven inn i et handlingsmønster som kan beskrives som en dialog[3] med omgivelsen, men hvor strategier og forklaringer forblir tause/implisitt.

I min studie vil elevenes umiddelbare handling ved å kjøre programmet når de programmerer, etterfulgt for eksempel av en forandring og et nytt forsøk, kunne være en aksjonssituasjon. De fleste elevene vet at programmet starter når de trykker «F5/Run» og «noe» vil skje, for eksempel at et output vil vises i konsollen, en tekst, en verdi eller en feilmelding. Det vil derimot variere i hvilken grad tilbakemelding fra handlingene bygger opp under en dialog mellom elevene og omgivelsene og skaper progresjon – om elevene bruker tilbakemeldingene de får etter en slik handling til å justere påfølgende handlinger. Brousseau (2006) hevder at nesten alle undervisningssituasjoner til en viss grad er aksjonssituasjoner, men at visse handlingsforløp preges av at elevene er i stand til å forvente resultatet av egne handlinger eller bygger en handling på den foregående (implisitte strategier). Dette kaller Brousseau for *aksjonsdialektikk*.

2.1.2.2.1 Situasjoner og dialektikk

Begrepet dialektikk må i denne sammenheng sees i lys av hvordan det brukes av Brousseau. Antakelsen til Brousseau (2006) er at elevene i den første fasen av spillet vil begynne å danne seg meninger om hvordan man vinner, for eksempel hvilke tall som er gode å «satse på», om man alltid må gå for partall eller lignende. Når elevens erfaringer er utgangspunkt for *implisitte strategier*, som igjen påvirker elevens handlinger i neste trekk eller påfølgende runde, oppstår sekvenser av aksjonssituasjoner som bygger på hverandre og som kan kalles *aksjonsdialektikk*, Brousseau utdyper:

> *«We use the word "dialectic" rather than the word "interaction" because, on the one hand, the student is capable of anticipating the results of her choices and, on the other hand, her strategies are, in a way, propositions confirmed or invalidated by experimentation in a sort of dialogue with the situation». (Brousseau, 2006, s. 9)*

Skillet kan illustreres ved ulike tilnærminger i elevers prøving og feiling som problemløsningsmetode i matematikk. Dersom man ut fra et datamateriale tolker en elevs prøving og feiling som basert på og informert av tidligere forsøk, kan man argumentere for

[3] Brousseau bruker begrepet *dialog* med omgivelsene, selv om begrepet *dialektikk* blir brukt i sammenheng med situasjoner og progresjon.

at dette er *aksjonsdialektikk*. Denne tolkningen må nødvendigvis basere seg på eksemplene elevene bruker, og innhold elevene refererer til muntlig. Begrepet dialektikk kan således sees i lys av en bredere definisjon, som en metode for å trenge inn i et problem[4]. Ifølge Brousseau kan dette også skje gjennom handlinger basert på implisitte strategier i tillegg til gjennom samtale. Det er også relevant å se begrepet i lys av motstand og tilbakemelding elevene møter i de ulike situasjonene;

> *«[Dialectic] is the term Brousseau uses to emphasise the notion that the student interactions can be seen as a 'dialogue' with the milieu […] in Didactique, it means the method by which the student manages contradictions between the expectations from the milieu and the feedback»* (Sierpinska, A. (2000) i Joubert, 2013, s. 73).

Sierpinska (2000, i Joubert, 2013) påpeker at dialektikk som begrep brukes tilnærmet synonymt med en «dialog med omgivelsene», og at denne dialogen innebærer visse forventninger til omgivelsene. Elevene kan for eksempel ha forestillinger om hva som kommer til å skje når de kjører et program. Kombinert med hvordan eleven håndterer motstand, er dialektikk med andre ord en vekselvirkning mellom elevens handlinger og tilbakemeldinger fra omgivelsene (informasjon) som kan bunne ut i elevens utvikling av kunnskap. Denne forståelsen av begrepet er ikke helt uproblematisk; Hvordan skal man for eksempel, basert på observasjon, kunne hevde at elever har «utviklet kunnskap»? Som Brousseau (2006, s. 64) antyder, vil nettopp situasjonen (konteksten) røpe hvorvidt elevens samhandling med de relevante omgivelsene (oppgaven/verktøyene) innebærer en utvikling hos eleven. Elevenes språklige aktivitet vil være med på å indikere nettopp dette. Dersom aktiv deltakelse innebærer ytringer knyttet til omgivelsene (for eksempel elementer i oppgaveteksten eller programvaren), kan dette indikere en form for utvikling av kunnskap.

2.1.2.2.2 Argumentasjon og ressonnering i aksjonssituasjoner

Ifølge Brousseau og Gibel (2005) kan argumentasjon og resonnering forekomme enten eleven selv er dette bevisst, eller ikke. Sistnevnte kan tillegges et individ basert på observasjon. Det betyr at ved å analysere elevenes handlinger i situasjonen kan man tillegge elevens handlinger argumenterende egenskaper. For eksempel vil det at en elev sier «Jeg vet

[4] Dialektikk defineres i SNL som *«en metode som gjennom samtale – spørsmål og svar, bevis og motbevis, argument og motargument – søker å bestemme begrepenes innhold eller, mer allment, trenge inn i et problem.»* https://snl.no/dialektikk

det er rett, se da!», og deretter kjører et program (Eksempel fra elevpar 2), kunne sees på som en form for argumentasjon der eleven prøver å overbevise motparten/medeleven om en løsning og bruker handlingen (og tilbakemeldingen som følger) som argument (middel for å overbevise). Den bakenforliggende resonneringen, beskrivelsen av hvorfor løsningen er korrekt, forblir implisitt, men kan tillegges eleven basert på konteksten. Man kan for eksempel anta at eleven sier «Jeg vet det er rett» fordi *elementet* eleven viser til i konteksten kan sees på som selvsagt eller som allmenn kunnskap. Målet om å overbevise seg selv eller andre, «Se da!», gjør at man også kan forstå handlingen som argumentasjon. Argumentasjon og ressonnering i denne formen kalles av Brousseau og Gibel (2005) *reasoning of level 1 (N1)* og kan være med på å utgjøre hvorvidt et tilfellet er aksjonssituasjon eller ei. Brousseau (2006) hevder også at resonnering kan sees som totalt implisitt, i det som kalles «an implicit model of action». En elevs umiddelbare og korrekte løsningsforslag i møte med en oppgave, vil kunne tillegges en handlingsmodell hvor man antar at det ligger en form for ressonnering til grunn. Dette vil i så fall være resonnering av første nivå (N1).

2.1.2.3 Formuleringssituasjon

I «the race to 20» er reglene i spillet motivert av kunnskapen det er ønsket elevene skal utvikle (knyttet til divisjon). Tanken er at elevene i spillfasen danner seg mer eller mindre implisitte teorier og hypoteser om hvordan man kan vinne. I neste fase av spillet ornaiserer læreren en ny situasjon der elevene skal sette ord på strategiene sine, prøve dem ut og observere utfallet. Når situasjonen krever av elevene at strategiene, meningene, observasjonene eller lignende gjøres eksplisitt, kaller Brousseau (2006) det for en *formuleringssituasjon*. Formuleringssituasjoner er preget av prosesser hvor det utvikles et forståelig språk rundt situasjonene og elementene i situasjonen – en slags meningsutveksling. Målet når en elev *formulerer* er ikke nødvendigvis å overbevise, men å utrykke og teste ut språk knyttet til situasjonen. Dette kan skje eksplisitt ved at elevene begynner å sette ord på det som skjer, ved å gradvis forsøke å forklare eller stille spørsmål til det som skjer.

Brousseau (2006, s. 12) skriver at "*a dialectic of formulation would consist of progressively establishing a language that everyone could understand, which would take into account the objects and the relevant relationsship of the situation in an adequate way*". Elevene går fra

implisitte strategier til å utvikle et felles språk, til å begynne å utrykke forståelse for situasjonen og forholdet mellom relevante elementer i omgivelsene. Progresjon syntes å være sentralt for hvorvidt situasjonen har dialektisk preg eller om den kan sees som en enkelthendelse mer frigjort fra det som skjer før og etter. Ifølge Brousseau (2006) kommer dialektikken til syne i prosessen når eleven mottar tilbakemelding basert på hvor forståelig utsagnet er og hvordan eleven gjør justeringer eller forandringer basert på tilbakemeldingen.

Ifølge Joubert (2017) kan *formuleringsdialektikk* beskrives som en prosess som foregår når elever møter et problem i en matematisk situasjon, er matematisk motiverte for å løse problemet og går i gang med en løsningsprosess som bunner ut i hypoteser, strategier, forklaringer eller bevis. I kontekster (milieu) der digitale verktøy er inkludert, kan tilbakemelding fra det digitale verktøyet spille en viktig rolle i formuleringsprosessen. Det kan skje ved at tilbakemeldingen fører elevene fra tilfeldig gjetting over til et mønster hvor de systematisk gjetter og forbedrer eller utvikler strategier (Joubert, 2017). I denne studien vil en formuleringssituasjon for eksempel innebære at elever diskuterer innholdet i en kode, forsøker å beskrive hva de forskjellige variablene i koden betyr og gjetter eller beskriver hvilken effekt dette kan ha for det som skjer etterpå når programmet kjøres. I en slik setting vil utspill som kommuniserer informasjon om situasjonen/omgivelsene, men som ikke krever svar, være eksempler på formuleringer (Brousseau, 2006). Ifølge Brousseau (2006) kan formuleringssituasjoner skilles fra valideringssituasjoner ved et fravær av diskusjoner som inkluderer bevis eller tydelige forklaringer.

2.1.2.3.1 Argumentasjon og ressonnering i formuleringssituasjoner

Det andre nivået for argumentasjon og ressonnering, *reasoning of level 2 (N2)*, knyttes til formuleringssituasjoner som er mer preget av eksplisitt kommunikasjon hvor elevene setter ord på handlinger og observasjoner, men hvor logiske forbindelser, spørsmål om hvorfor, forklaringer og eksplisitte begrunnelser uteblir (Brousseau & Gibel, 2005). En elev som sier «Hvis vi skriver inn 3 der, får vi 3-gangen», er et eksempel på en slik form for resonnering, og vil kunne være et argument avhengig av konteksten. I utsagnet verken begrunner eller forklarer eleven påstanden om at resultatet av handlingen blir «3-gangen», men det trenger heller ikke være målet for eleven. Resonnering av typen hvis A så B, uten at det etterfølges av et «fordi», vil med andre ord være typisk for argumentasjon og ressonnering i formuleringssituasjoner. Brousseau og Gibel (2005, s. 57) skriver også at *«In a situation of*

formulation the student has to adapt his language to express his thinking, his implicit model of action. In a situation of communication he has to adapt it also to his interlocutor». Dette betyr at formuleringssituasjoner impliserer et sosialt aspekt hvor argumenter og resonnement gradvis tilpasses eller justeres basert på tilbakemelding fra samtalepartner.

2.1.2.4 Valideringssituasjoner

Den siste fasen av «The race to 20» handler om at elevene blir nødt til å ta stilling til strategiene eller hypotesene som er foreslått, samt vurdere holdbarheten og gyldigheten for disse. I eksempelet har læreren tilrettelagt for at elevene får komme med utsagn og blir møtt med respons fra resten av elevgruppen. I motsetning til tidligere i spillet da et utsagn kunne komme uimotsagt, vil det ifølge Brousseau (2006) nå være et mål at elevene gransker utsagnene i lys av logikk og eksempler, og i den grad elevenes utvikling tillater det, beviser og argumenterer for teoriene som er lagt frem. Slik aktivitet klassifises som *valideringssituasjoner*.

Valideringssituasjoner omhandler elevers bekreftelse av påstander, teorier og utspill som dukker opp gjennom å sette ord på det matematiske innholdet i situasjonen. Brousseau (2006) hevder at elevers argumentasjon, påstander og teorier i utgangspunktet kan være mangelfulle, feilaktige og klumsete, og at de godtar mangelfulle og feilaktige påstander fra deres medelever. Situasjonen bør imidlertid «tvinge» elevene til å omformulere, revidere og justere utsagn slik at de forstås og godtas av mottakerne (først og fremst medelever). Det er ifølge Brousseau (2006) i situasjoner hvor elever argumenterer og søker bekreftelse at de får mulighet til å oppdage egne misoppfatninger, gjøre nødvendige justeringer og tilegne seg kunnskap fra en situasjon. Brousseau (2006) mener oppgavedesign som legger til rette for eller oppfordrer til validering bær være et mål for matematikkundervisning da det først er i slike situasjoner genuin matematisk aktivitet finner sted.

Joubert (2017) beskriver *valideringsdialektikk* som noe som skjer når en samhandling mellom elev og situasjon eksplisitt krever forklaringer, bevis eller teorier – og samtidig oppfattes slik av den som deltar i situasjonen. Lærerens og medelevens rolle blir fremhevet som viktig for å kunne fasilitere situasjoner som inneholder validering fordi slike situasjoner sjeldent er spontane, de kan være vanskelige å fremprovosere, og de kommer sjeldent fra påvirkning fra omgivelsene som for eksempel oppgavedesign. Brousseau (2006) presiserer at *«it is clear, at any rate, that a dialectic of validation is itself a dialectic of formulation and therefore a*

dialectic of action» (s. 17). Valideringsdialektikk kan dermed sees som en dypere form for handling og formulering, som inkluderer eksplisitt og målrettet argumentasjon og resonnering. Den er målrettet i den forstand at den tar utgangspunkt i det matematiske innholdet som er relevant for oppgaven/undervisningen. I min studie vil valideringsdialektikk være situasjoner hvor elever argumenterer for påstandene sine i et forsøk på å overbevise partneren, for eksempel om at programmet vil gi et bestemt utfall. I en slik argumentasjon krever også tydelige begrunnelser med referanser til det matematiske innholdet eller programmeringskonsepter.

2.1.2.4.1 Argumentasjon og ressonnering i valideringssituasjoner

Brousseau (2006, s. 63) skiller mellom korrekte påstander (uavhengig av hvor formelt korrekte de måtte være) og det som beskrives som *validering*. Valideringer består av både beskrivelser, såkalte modeller, uttrykt i et relevant språk i henhold til elementer i omgivelsene, nivå, oppgave og andre forhold) og *vurderinger* av disse beskrivelsene. Sistnevnte omhandler hvilke forhold beskrivelsene gjelder for, hvorvidt det som beskrives alltid stemmer og hvorvidt beskrivelsene stemmer overens med de observerbare forhold. En valideringssitusjon vil følgelig kunne sies å inneholde to komponenter der kompleksiteten er avhengig av elevens utvikling og faglige kunnskap; en teori (forklaring eller modell uttrykt med språk) og en vurdering av teorien i lys av holdbarhet og/eller gyldighet. Sammenhengen mellom begrepene validering og ressonering er med andre ord tett overlappende og en elevs validering vil ha form som et argument dersom det er ment å overbevise samtalepartner.

Argumentasjon knyttet til valideringssituasjoner beskrives som en komplett argumentasjon med logisk konsekvente, tydelige referanser til elementer i situasjonen. Dette er i følge Brousseau og Gibel (2005) *reasoning of level 3 (N3)*. Her klarer eleven å påpeke og bruke relevante sammenhenger som enten har dukket opp i situasjonen eller som baserer seg på tidligere kunnskap, på en logisk korrekt måte; «Hvis A, så B, fordi C». Det skilles mellom elevers argumentasjon sett fra en lærers perspektiv i en undervisningssituasjon hvor elevenes utsagn ofte tolkes i retning av det matematiske målet, og sett fra en observatør eller forskers synspunkt hvor man må vise til elevers argumentasjon som intensjonell og meningsfull for det matematiske aspektet ved situasjonen (Brousseau & Gibel, 2005).

2.1.3 Motstand

Konstruksjon av kunnskap kan ifølge Brousseau (2006, s. 82) beskrives som det som skjer i en dialektikk mellom elev og spesifikke problem-situasjoner. Det er i disse situasjonene didaktikken har sitt mål, ved å studere hvilke vilkår i situasjoner og problem som fostrer en slik dialektikk. Hva interesserer eleven seg for (i oppgaven/situasjonen)? Hva prøver eleven ut? Hvilken innsats er eleven interessert i å legge ned for å løse problemet? Brousseau (2006) hevder at de mest interessante og nyttige problemene er de hvor elever utsettes for en viss form for matematisk motstand. Totalt fremheves tre former for motstand som man kan forvente i et forsøk på å legge til rette for elevers konstruksjon av kunnskap gjennom undervisning (Brousseau, 2006); Motstand relatert til elevers utvikling (*ontogenic origin*), motstand relatert til didaktiske valg (*didactic origin)* og motstand relatert til det matematiske innholdet (*epistemological origin*).

Motstand knyttet til elevers utvikling vil i en programmeringskontekst for eksempel være knyttet til elevenes evne til abstraksjon og hukommelse i møte med variabler og funksjoner. Forståelse av relevante begreper og konsepter som benyttes i programmeringskonteksten vil derfor kunne være emne for slik motstand, men her kan det også være snakk om motstand knyttet til didaktiske valg. Gir man elever oppgaver som krever kunnskap elevene ikke har fått mulighet til å tilegne seg, eller man strukturerer undervisningen på en måte som hindrer elevene i å tilegne seg kunnskap, kan dette sies å være didaktiske årsaker til elevenes motstand i situasjonen. Det er ifølge Brousseau (2006) et mål å redusere begge disse formene for motstand.

Det er derimot et mål å utsette elevene for motstand knyttet til matematisk innhold (*epistemologial origin*). Det er i møte med slik motstand man legger til rette for formulerings- og valideringssituasjoner og matematisk kunnskap utvikles (Brousseau, 2006). Joubert (2013) skriver at matematisk motstand kjennetegnes ved at eleven må konstruere ny eller reorganisere kunnskap for å løse problemet. Man kan med dette forstå motstand ut fra det som skjer i a-didaktiske situasjoner; et problem som krever at elevene revurderer eksisterende kunnskap der de bruker tilbakemeldingene fra omgivelsene i de gitte situasjonene til å tilpasse seg og skape en ny forståelse av det faglige innholdet. Det handler med andre ord om å gjøre elevene bevisst sine misoppfatninger eller ufullstendige forestillinger om det matematiske innholdet. Brousseau (2006, s. 82) mener at matematisk

motstand kan sees på som en «bit kunnskap» som tidligere ble antatt å være sann, men som viser seg utilstrekkelig eller feil. Matematisk motstand kan derfor også sees på som nærliggende med begrepet misoppfatninger, bare at dette dette også gjelder elevers «naive» forestillinger – deres forståelse av konsepter de strengt tatt ikke har blitt gitt mulighet til å forstå enda (for eksempel dersom innholdet ikke har vært gjenstand for undervisning enda).

Det sees av Brousseau (2006) som uungåelig i et samspill mellom eleven (og den kunnskap eleven har) og omgivelsene (milieu), at det oppstår slike misoppfatninger – konflikter mellom tidligere kunnskap og den aktuelle situasjonen. Men som det påpekes, er det en av didaktikkens primære beskjeftigelser å forstå vilkårene for hvordan slike misoppfattelser dannes og hvordan man kontrollerer dem. Implikasjoner for undervisning er at elever må eksponeres for gjentatte situasjoner hvor de møter motstand hvor disse misoppfatningene adresseres, og disse situasjonene må i tillegg være ulike nok til at elevene har behov for å utvikle nye strategier/ny kunnskap (Brousseau, 2006, s. 85). Et ekte problem tillater og motiverer en dialektikk mellom elevers tidligere oppfatninger og setter disse på prøve, med mål om at eleven overkommer motstanden og derav konstruerer ny kunnskap.

Joubert (2017) viser til at begrepet motstand (*obstacles*) hos Brousseau (2006) kan sees som et viktig begrep knyttet til design av oppgaver. For å fremme elevers matematiske motivasjon bør oppgaver lages på et vis som legger til rette for eller tvinger frem den matematiske aktiviteten man ønsker hos elevene. Ønsker man at elevene skal utvikle ny kunnskap eller en dypere forståelse, for eksempel knyttet til divisjon og rest, må de møte motstand i situasjonene (oppgaven) som utfordrer elevenes (antatt feilaktige/ufullstendige) forståelse av begrepene involvert. De må gjennom oppgaven tvinges til å resonnere seg frem til, formulere eller uttrykke bedre eller nye forklaringer på konseptene.

2.1.4 TDS som rammeverk – noen refleksjoner

Verken studien eller undervisningsopplegget som er gitt elever under datainnsamling i dette masterprosjektet er laget med utgangspunkt i TDS. Artigue et al. (2014) deler dette utgangspunktet og nevner potensielle utfordringer med å bruke TDS som rammeverk for å analysere undervisning. Blant annet nevnes ulik oppgavetype og annerledes situasjoner. Artigue et al. (2014, s.54) viser for eksempel til egne studier hvor oppgavene er mer rettet mot utforsking, fremfor klassisk problemløsning brukt som eksempler hos Brousseau (2006).

Det kan derfor være utfordrende for elevene å vite hvilke kriterier som kjennetegner en fullført oppgave – når er problemet løst? Tilsvarende er flertallet av oppgavene i dette masterprosjektet (i utdragene som presenteres spesielt) rettet mot å undersøke, utforske og gjette utfall, for å nevne noe. Det finnes eksempler i datamaterialet som viser at elevene anser en oppgave som løst, men hvor løsningsforslaget verken er utrykt eksplisitt eller forklart. Artigue et al. (2014) viser også til at det kan være utfordrende å klassifisere situasjoner, spesielt fordi oppgavene og omgivelsene påvirker elevenes handlinger og yttringer. Eksemplifisert kan aksjonsdialektikk fort gli over i mer eksplisitt formulering dersom elementer i oppgaven legger til rette for det.

Joubert (2017) bruker begreper fra TDS som et rammeverk for å forstå elevers aktivitet i møte med oppgaver, og det blir blant annet spurt hvilken rolle digitale verktøy og teknologi spiller og hvilken rolle motstand spiller i digitalt oppgavedesign hvor matematisk læring er målet. Begrepene *aksjons-, formulerings- og valideringsdialektikk* blir brukt for å beskrive elevers matematiske aktivitet og *motstand* (obstacles) blir deretter brukt som et utgangspunkt for å analysere hvilken elevaktivitet oppgavedesignet legger opp til (Joubert, 2017). Hun viser til egne studier, hvor det først og fremst fokuseres på oppgavedesignets rolle i en digital kontekst, og det vises til data fra undervisningssituasjoner hvor oppgavene ikke i tilstrekkelig grad er et godt utgangspunkt for valideringssituasjoner. I min studie bruker jeg en lignende tilnærming, men hvor TDS er tiltenkt å beskrive aspekter knyttet til interaksjon mellom elev og omgivelsene.

Leung og Bolite-Frant (2015) viser til TDS i tilknytning til bruk av ulike redskaper i matematikkundervisning og hevder verktøy kan legge til rette for de ulike situasjonsformene; aksjon, formulering og validering. Brousseau skriver selv at teorien kan brukes som rammeverk for å forstå situasjoner hvor man forsøker å undervise «noe» (spesifikt) til noen (Sierpinska i Måsøval, 2011, s. 64). Spesielt i lys av spørsmålene som stilles av Joubert (2017), mener jeg begrepene i TDS er relevante for mitt fokus og de gir et nyttig perspektiv på samhandling og argumentasjon hos elever i interaksjon med omgivelsene (mer spesifikt programvare).

På bakgrunn av hvordan undervisningsoppleggene i denne studien er lagt opp, vil situasjonene som analyseres i masteroppgaven være mer eller mindre a-didaktiske. Lærer interagerer kun sporadisk i elevenes selvstendige arbeid i par. De didaktiske situasjonene,

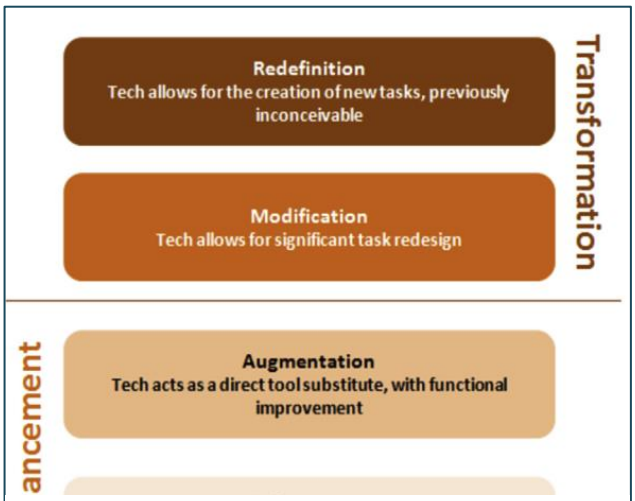
undervisningskonteksten på et mer overordnet plan, vil likevel være relevant og utdypes i metodekapittelet.

2.2 SAMR-modellen

Selv om TDS er en helhetlig teori er det valgt å i tillegg inkludere et alternativt perspektiv for å bedre kunne si noe om bruk av programmering som digitalt verktøy i dette prosjektet. Flere teorier/perspektiv er relevante til nettopp det, og et av rammeverkene som er vurdert som hjelpemiddel for å forstå bruken av programmering som redskap i undervisning er TPACK-modellen (Technological Pedagogical Content Knowledge) som beskrevet av Koehler og Mishra (2009), men da fokuset mitt i større grad retter seg mot forholdet oppgaven-elev (heller enn lærer-oppgave) har jeg valgt å heller supplere med SAMR-modellen (Puentedura, 2013b).

2.2.1 SAMR-modellen – Redskapers didaktiske funksjon

Som nevnt innledningsvis er SAMR forkortelse for Substitution, Augmentation, Modification og Redefinition, som i denne oppgaven er oversatt til Erstatning, Funksjonsøkning, Modifisering og Redefinering. Modellen retter seg mot hvordan teknologi integreres i klasserommet og er ment som et hjelpemiddel for lærere som ønsker å bruke digitale verktøy (Hilton, 2016). Bray og Tangney (2017) beskriver modellen som et hierarki som definerer et system hvor ny bruk av teknologi klassifiseres etter hvilken nyskapning teknologien tilfører i forbindelse med oppgavedesign og bruk (Figur 5). Modellen beskrives også som godt egnet til å beskrive hvordan teknologi brukes i sammenheng med spesifikke oppgaver (Bray & Tangney, 2017). SAMR-modellen tilbyr med andre ord et nyttig perspektiv for å forstå programvaren som redskap for matematisk aktivitet og hvorvidt programmeringskomponenten i oppgavene som er brukt kan sies å ha endret matematikken (eller om keiseren bare er gitt nye klær). Bruken av SAMR-modellen i denne



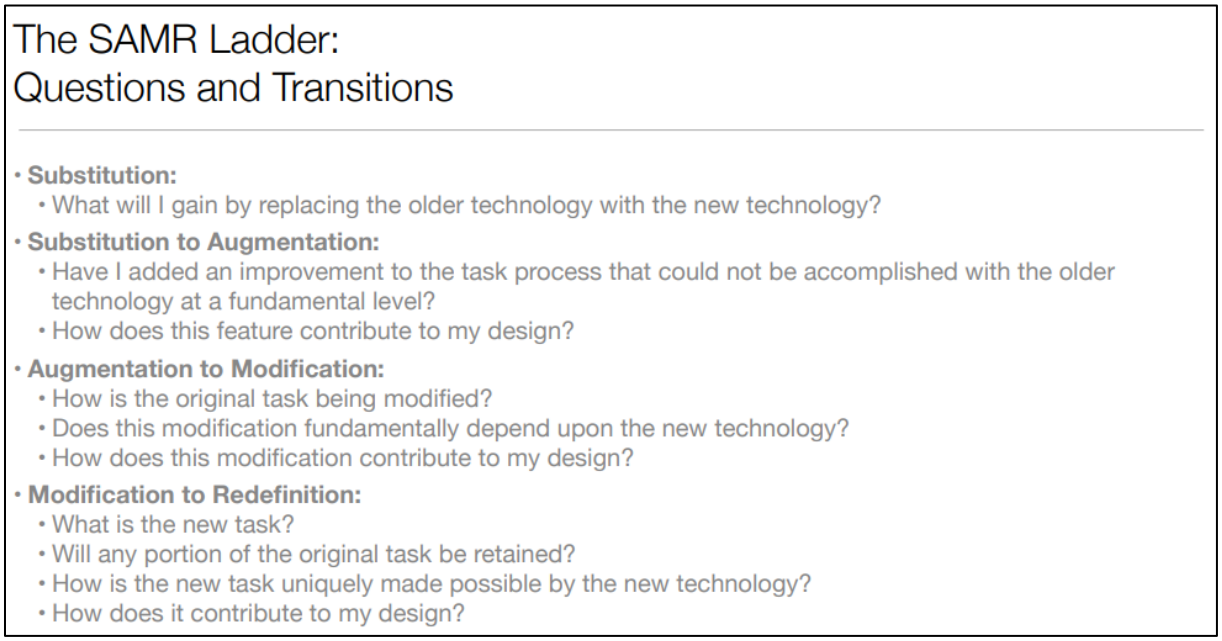
Figur 5: SAMR-modellen. Hentet fra Bray og Tangney (2017)

oppgaven er først og fremst ment som et supplement for å gi et bedre grunnlag for å kunne si noe om hvilken rolle programvaren (Python) spiller i de ulike utdragene som analyseres.

Erstatning handler om at et teknologisk verktøy erstatter et annet, uten nevneverdig funksjonsendring (Puentedura, 2013b). Et eksempel kan være en håndholdt kalkulator versus en kalkulatorapp på en datamaskin eller en mobil. Foruten å være på en datamaskin, vil det å bruke en applikasjon tilføre situasjonen minimalt sammenlignet med å bruke et fysisk eksemplar av en håndholdt kalkulator. Det ene verktøyet simpelthen erstatter det andre, men bruken forblir lik. Har kalkulator-appen derimot noen funksjoner som den håndholdte kalkulatoren ikke har, for eksempel mulighet til å tegne grafer, kan man diskutere hvorvidt bruken fungerer som *funksjonsøkning*. Appen (verktøyet) fungerer fortsatt som en direkte erstatning for en håndholdt kalkulator, men med noen forbedringer som øker funksjonaliteten. Et annet eksempel kan være å bruke en app på et nettbrett i innlæring av gangetabellen. Oppgaven forblir den samme (regne ut gangestykker), men appen tilbyr raskere tilbakemelding og ubegrenset øving. Man kan si at bruk av en slik app i en slik situasjon tilsvarer en *funksjonsøkning* sammenlignet med å øve gangetabellen med blyant og papir. Begge disse formene for bruk av teknologi kan ifølge Puentedura (2013b) fungere som *forsterkende* (*enhancement*), selv om de store endringene i didaktisk bruk uteblir.

Teknologi/verktøy kan også ha den funksjonen at det *endrer* oppgavene som gis eller gir rom for å skape nye oppgaver som uten teknologien ikke er mulige. I SAMR-modellen tilsvarer dette hovedkategorien *transformation*, som involverer at teknologien enten gir muligheter for å *modifisere* undervisning eller lage *nyskapende* undervisning hvor oppgavene som gis ikke er mulige å gi uten bruk av gitt teknologi (*redefinition*) (Puentedura, 2013b). Bray og Tangney (2017) viser til dynamisk geometri-programvare som eksempel på teknologi som kan bidra til å *modifisere* oppgaver i matematikkundervisning. Ved at beregninger tidligere utført av eleven blir gjort av datamaskinen (programvaren), endres elevenes aktivitet fra manuell plotting av grafer og beregninger, til for eksempel vurdering av grafenes egenskaper sett i forhold til funksjonsuttrykkene. Dette kan også bety at en i større grad bør bruke oppgaver som er mer fokusert på problemløsning og samarbeid, sammenlignet med arbeid uten et tilsvarende verktøy. Man bruker med dette teknologi slik at oppgavene kan lages på nye måter.

Det siste steget i modellen, *redefinering*, handler om at teknologien gir muligheter for oppgavedesign og undervisning som tidligere, uten teknologien, ikke ville vært mulig. Dette kan for eksempel innebære oppgaver som krever bruk av dynamisk geometriprogram kombinert med opptaksfunksjoner for å dokumentere og dele løsninger med resten av gruppen. En oppgave kan for eksempel be elevene lage egne problemer basert på et datasett, lage en instruksjonsvideo til hvordan man løser oppgaven, og dele denne med resten av klassen. Her kan man si at teknologien, nettbrett/data (programvare knyttet til opptak) gjør mulig en form for oppgave som avhenger av teknologien. Puentedura (2013a) viser til en serie spørsmål som kan fungere som en metode for å undersøke i hvilken grad teknologi kan brukes for å endre oppgaver som gis i undervisning (Figur 6).



Figur 6: Spørsmål for utvikling av oppgaver med teknologiske verktøy (Puentedura, 2013a)

2.2.2 Ulik bruk av SAMR-modellen

SAMR-modellen blir i flere andre sammenhenger brukt som et verktøy for utvikling av oppgaver og undervisning, noe som kanskje er mest i tråd med Puentedura (2013b) slik modellen legges frem. Eksempelvis bruker Clark og Lee (2019) det de kaller et SAMR-pensum i en case-studie hvor det forskes på bruk av nettbrett i matematikkundervisning. Oppgavene som skildres i studien er bevisst laget med tanke på de fire nivåene i SAMR-modellen. Personlig tilpasning i innlæring av gangetabellen ved hjelp av en app, er brukt som eksempel

for hvordan en oppgave er *modifisert*. SAMR-modellen er også brukt som et analyseverktøy for å klassifisere bruk av teknologi i oppgavedesign og undervisning (Hilton, 2016; Romrell, Kidder & Wood, 2014).

Modellen er også brukt i kombinasjon med andre rammeverk, for eksempel TPACK (jf. Koehler & Mishra, 2009). I en artikkel som tar for seg hvordan barnetrinnslærere bruker digital teknologi i matematikk, bruker Loong og Herbert (2018) SAMR modellen for å analysere hvordan lærere benytter seg av teknologi i matematikkundervisning. Det blir vist til eksempler hvor elever jobber med divisjon og læreren bruker et spill for å øve på delelighet. Forfatterne konkluderer med at slik teknologien brukes og måten oppgavene gis på, *erstatter* datamaskinene analoge alternativer, men at lite nytt tilføres (*substitution*).

2.2.3 Utfordringer ved bruk av SAMR-modellen

Som man kan se på eksemplene ovenfor, brukes modellen på ulike måter. Dette i seg selv trenger ikke å være negativt, men som Hamilton, Rosenberg og Akcaoglu (2016) påpeker, er ikke modellen i særlig grad teoretisk forankret, noe som kan være med på å forsterke ulike/forskjellige tolkninger. Enkelte hevder for eksempel at SAMR-modellen skal forstås som et hierarki; at innføring og bruk av teknologi som læringsverktøy handler om å skape muligheter (gjennom oppgaver) til læring som ellers ikke ville vært mulig (Hilton, 2016). Andre vil hevde modellen heller representerer et spektrum hvor de ulike elementene er likeverdige og representerer ulik bruk av teknologi som verktøy (Hamilton et al., 2016). Erstatning og funksjonsøkning kan med andre ord være et like «verdige» mål i innføring av ny teknologi, avhengig av hva man ønsker å oppnå. I tillegg nevner Hamilton et al. (2016) utfordringer knyttet til manglende fokus på kontekst i selve modellen, og et overdrevent fokus på verktøy og prosess, fremfor det som gjerne er målet med undervisning, nemlig læring.

I denne masteroppgaven vil modellen først og fremst brukes som et alternativt perspektiv, som et supplement til TDS hvor refleksjon over programmeringsverktøyets rolle i oppgavedesignet er hovedfokus. For å kunne undersøke muligheter og utfordringer ved bruk av tekstbasert programmering på mellomtrinnet, er spørsmålene til Puentedura (2013a) (Figur 6) sett på som gode utgangspunkt for refleksjon etter at datamaterialet er analysert. Utfordringene som nevnes av Hamilton et al. (2016) er kommet i møte ved at (1) kontekst allerede spiller en viktig rolle i denne studien og vil bli beskrevet mer utfyllende, (2)

modellen brukes som supplement til teori av mer «omfattende» karakter (jamfør. TDS), og (3) modellen er ikke brukt som et statisk hierarki for utvikling av oppgaver, men mer som et perspektiv for å tolke noe som allerede er gjort og for å undersøke hvorvidt denne bruken er hensiktsmessig.

3 Metode

I denne studien er fokuset generelt rettet mot muligheter og utfordringer ved å bruke tekstbasert programmering i matematikkundervisning, spesielt på samspillet mellom elever, oppgave og programvare. Ved å kombinere rollen som mastergradsstudent og faglærer på 7. trinn, ble det tidlig laget en plan for å prøve ut nye undervisningsmetoder hvor programmering brukes i matematikkundervisningen for egne elever. Dette ble gjort gjennom et aksjonsforsknings-inspirert prosjekt hvor det først ble undervist i programmering og deretter forsøkt å rette fokus mot matematikkfaglig innhold. I undervisningen ble elevene kjent med programmeringsspråket Python gjennom aktiviteter som å lage små spill, styre skilpadder (ref. LOGO) og lage andre enkle programmer. Varigheten av prosjektet var rundt 8 uker og det var i denne perioden problemstillingen tok form ved at en rekke spørsmål dukket opp; Hvordan reagerer elevene når programmeringsaktivitetene får et matematikkfokus? Hvordan påvirker oppgavene og programmerings-aspektet elevenes matematiske aktivitet? Hvilke utfordringer dukker opp og hvordan kan disse løses? I undervisningen oppstod flere situasjoner som indikerte både utfordringer og et potensiale for å bruke programmering inn mot matematikkundervisningen.

Basert på erfaringer fra undervisningen ble det laget to undervisningsopplegg med mål om å legge til rette for matematisk aktivitet knyttet til tallforståelse. For å kunne belyse problemfokuset og svare på forskningsspørsmålene er det valgt en kvalitativ tilnærming med fokus på oppgavedesign og utprøving der målet har vært å studere elevens arbeid. Prosessen fra tidlig planlegging, undervisning og lek med Python til to eksperimentelle undervisningsopplegg, kan sies å være inspirert av aksjonsforskning da jeg som mastergradsstudent selv har gjennomført undervisning, planlegging og evaluering. Da data skulle analyseres, ble det valgt å fokusere på deler av datamaterialet som omfatter temaene delelighet og rest, samt begrense omfang ved å kun bruke data fra noen av elevparene. Data fra tre ulike elevpar blir i følgende behandlet som avgrensede kasus for å kunne svare på spørsmålene som stilles innledningsvis.

3.1 Valg av metode

3.1.1 Forskningsdesign

Et forskningsdesign beskrives som en kontinuerlig plan som inneholder forskerens mål, teorier, problemstilling, metode og begrunnelser (Maxwell, 2005, s. 3). Problemstillingen min har forandret seg underveis i arbeidet med oppgaven. Målet har likevel hele veien vært å prøve noe nytt, og gå i dybden i situasjoner i egen undervisning hvor programmering brukes for å forstå hvordan programmering som verktøy kan påvirke læringsprosessen og undervisningen i klasserommet. En kvalitativ tilnærming har derfor vært det mest aktuelle.

I valg av metode generelt nevner Cohen, Morrison og Manion (2007) prinsippet om «Fitness for purpose» som grunnleggende prinsipp når man skal velge metode for å nå forskningsmålet sitt. Kort fortalt handler det om at hvilket forskningsparadigme man plasserer seg i må avhenge av formålet med forskningen og problemstillingene man tar for seg. For å studere situasjoner og fenomener mer i dybden kreves en etnografisk eller kvalitativ tilnærming (Cohen et al., 2007). Ettersom forskningsdesignet er preget av å være eksperminentelt – å lete etter potensiale i nye arbeidsmetoder/verktøy i eget undervisningsarbeid, samt utfordringer og muligheter med en gitt tilnærming, vil en kvalitativ tilnærming være egnet. Temaet i seg selv kan nok også utdypes gjennom kvantitative metoder, for eksempel ved å spørre et større utvalg lærere om deres forkunnskaper knyttet til programmering. En kvalitativ tilnærming vil derimot i større grad belyse de spørsmål jeg som lærer er interessert i; hvordan dette kan foregå i praksis – i eget klasserom, med fokus på elevene.

I kombinasjon med prinsippet «fitness for purpose» kan det også argumenteres for en kvalitativ tilnærming da problemstillingen som stilles ser ut til å være mindre belyst. Creswell (2014, s. 50) viser til kvalitativ forskning som godt egnet til å belyse temaer som det er forsket mindre på. Dette begrunnes med at man da kan være åpen for hvilke variabler som legges vekt på når fenomenet undersøkes og hvilke teorier som er relevante for å kunne si noe om tilfellene som studeres (Creswell, 2014). Valget for kvalitativ tilnærming i min studie kan dermed også begrunnes med et ønske om å undersøke noe som tilsynelatende få har undersøkt før i en tilsvarende kontekst. Det kan være naturlig å kombinere forskningstilnærminger, avhengig av hva man forsker på og hva man ønsker å få innsikt i

(Cohen et al., 2007). Jeg har valgt å kombinere en kasus-studie med et aksjonsforskningsperspektiv.

3.1.2 Aksjonsforskningsperspektiv

Ettersom problemstillingen tar utgangspunkt i en reell kontekst (egen arbeidsplass) samt handler om å prøve ut ny praksis, har et aksjonsforskningsperspektiv stått sentralt som metode i prosessen både før og under datainnsamlingen. Prosessen bærer preg av aksjonsforskning da datainnsamling har foregått stegvis i sykluser med problemstillingen basert i reelle behov. Dette er utgangspunktet for aksjonsforskere, slik det beskrives av McNiff (2006). I tillegg beskrives aksjonsforskning som et forsøk på å forstå, endre, reformere og forbedre ens egen praksis (Cohen et al., 2007, s. 297), noe som også er tilfellet i denne studien. Det er forsøkt å se etter potensiale i måter å undervise på basert på hva jeg ser som relevant med tanke på nye læreplaner og fremtidig yrkesutøvelse. Aksjonsforskning brukes for å studere hva som skjer når man forandrer undervisningsmetode, utvikler nye undervisningsmetoder eller forsøker å utvikle seg som lærer gjennom kompetanseheving (Cohen et al., 2007, s. 297). McNiff (2006) beskriver aksjonsforskning som en syklisk prosess hvor man beskriver nåværende tilstand, reflekterer over hva man ønsker å forandre, prøver ut nye måter, samler data for å evaluere effekt og justerer etter behov. Med forarbeidet inkludert kan denne studien sies å ha foregått gjennom tre sykluser over en periode på tre måneder (Som vist de tre første punktene i Figur 10, side 50).

3.1.3 Kasus studie

Både kasus-studier og aksjonsforskning har som mål å få dypere forståelse for enkeltfenomener i en reell kontekst (Blichfeldt & Andersen, 2006, s. 3). I en kasus-studie vil forskerens interesse for et gitt fenomen styre hva man undersøker, mens i aksjonsforskning styres valg av problemstilling og forskningsspørsmål mer av hvilke utfordringer deltakerne har (Blichfeldt & Andersen, 2006). Ettersom jeg i denne studien kombinerer min rolle som forsker og lærer, føles det naturlig å fremheve denne «doble» rollen også i metoden.

En kasus-studie gir forskeren innblikk i en avgrenset situasjon for å forstå sammenhenger mellom teori og praksis, og søker å beskrive hendelser og situasjoner gjennom fortolking (Cohen et al., 2007, s. 254). Studien min kan sees på som en kasus-studie da det er valgt å besvare spørsmålene som stilles ved å beskrive spesifikke utvalgte situasjoner, fortolke disse ut fra et teoretisk rammeverk og prøve å si noe om hva som skjer, samt lete etter muligheter

og utfordringer som viser seg når gitte verktøy benyttes. Fordelen med kasus-studier er blant annet at de inneholder virkelighetsnære beskrivelser som er lette å kjenne seg igjen i og man får en unik innsikt i hendelser som i større undersøkelser ofte går tapt (Cohen et al., 2007). Ved gode beskrivelser vil et kasus kunne tale for seg selv, og beskrivelsene vil ofte kunne være lettere tilgjengelige eller lesbare for et ikke-akademisk publikum. Spesielt det siste punktet er et viktig poeng da studien kan sees som en del av en utviklingsprosess på en skole og fokuserer på utfordringer som flere lærere står ovenfor i møte med fagfornyelsen høsten 2020.

Siden det er valgt å fokusere på enkeltutdrag fra datamaterialet i denne oppgaven, og det i tillegg tas utgangspunkt i Brousseau (2006) sin teori om didaktiske situasjoner, er det viktig å skille mellom utdragene, som representerer ulike *kasus*, og *situasjoner* som er mer avgrensede analyseenheter innad i utdragene. Dette er gjort fordi fokuset i studien underveis har blitt mer rettet mot de spesifikke situasjonene knyttet til elevenes problemløsning i de ulike oppgavene. En *situasjon* vil heretter omhandle et elevpar sitt spesifikke arbeid med å løse en spesifikk oppgave i et av oppgavesettene. På den måten vil en *situasjon* kunne beskrives som «et kasus i kasuset». Siden fokuset etter hvert har rettet seg mot slike situasjoner, har aksjonsforsknings-perspektivet blitt mindre vektlagt i teksten som presenteres.

3.2 Kontekst og deltakere

Prosjektet har funnet sted på egen arbeidsplass, i en klasse jeg underviser matematikk i. Før datainnsamlingen startet ble prosjektet godkjent av rektor på skolen samt Norsk senter for forskningsdata (NSD). Totalt tolv elever meldte interesse for å delta i prosjektet gjennom egen- og foresattes samtykke. Planen var å velge ut tre par basert på kartleggingsprøver og ferdighetsnivå, men da såpass mange elever var interessert, ble det bestemt å la alle elevene få delta. Totalt seks elevpar ble filmet ved gjennomføring av første runde datainnsamling. Etter dette valgte et av parene å trekke seg og to elevpar var fraværende på grunn av sykdom. Resterende tre elevparene ble filmet i påfølgende økt.

Deltakerne i utdragene som analyseres kan karakteriseres som en sammensatt gruppe, og gjenspeiler mangfoldet i klassen på en god måte. Deltakergruppen består både av gutter og

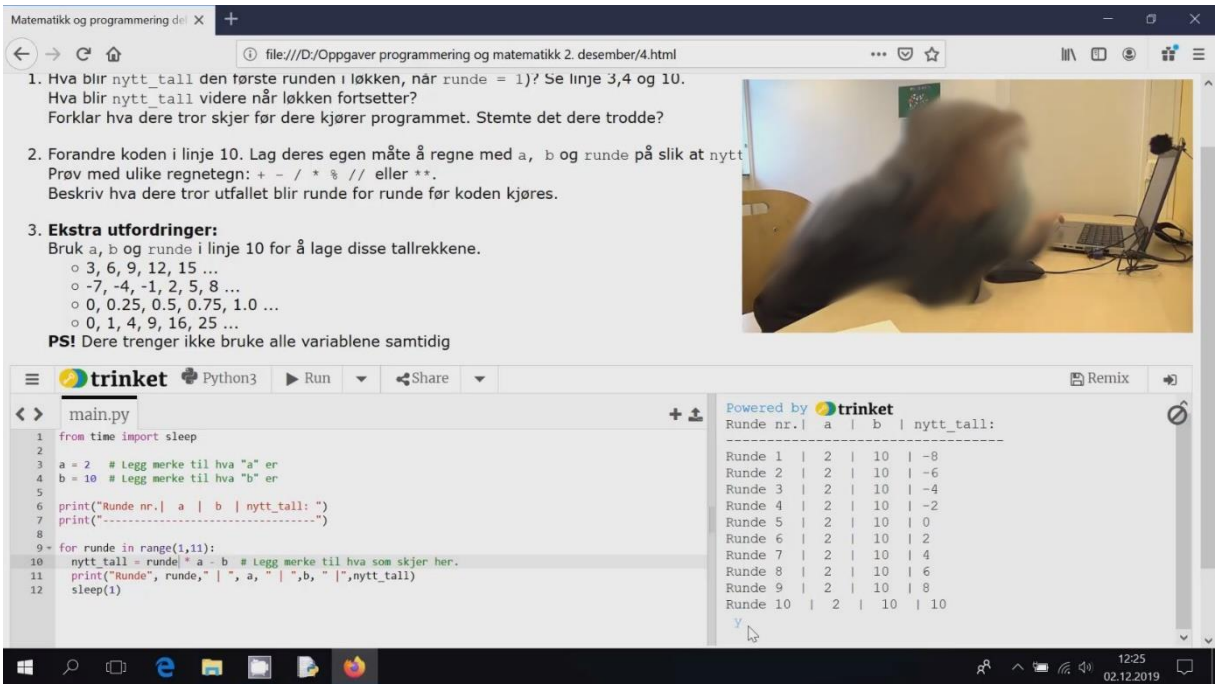
jenter og representere elever med ulike forutsetninger knyttet til matematikkfaglige- og programmeringstekniske kunnskaper. Noen av elevene viste i forarbeidet stor interesse for programmering, andre har vært mer nøytrale. Sammensetning av par ble basert på skjønn, med mål om at parene samarbeidet godt. Selv om elevparene representerer et begrenset utvalg, er det sett på som interessant å se hvordan ulike elever håndterer situasjonene og oppgavene.

Det første elevparet består av to gutter, her referert til som Geir og Harald. Begge kan karakteriseres som middels presterende i matematikkfaget, men har gjennom prosjektet vist stor interesse for programmering. Det andre elevparet består av en gutt og en jente, kalt Dag og Kari. Begge kan karakteriseres som middels presterende i matematikkfaget. De har gjennom prosjektperioden vært positive til programmering, men ved flere anledninger uttrykt at de synes det er vanskelig. Det tredje elevparet, kalt Ask og Brage, kan begge karakteriseres som høytpresterende og særdeles aktive i matematikkfaget. Underveis i prosjektet har også disse vist stor interesse for programmering.

3.3 Datainnsamling

For å få best mulig innsikt i elevers arbeid med programmering, samspillet mellom elevene, oppgave og programvare, ble det valgt å gjøre både film-, skjerm- og lyd-opptak av elevarbeidet (jamfør Alrø & Kristiansen, 1997). Fra prosjektets startfase frem til første runde med filming, ble undervisningsøktene i programmering planlagt og det ble skrevet logg i etterkant av øktene. Denne prossen bar preg av kontinuerlig evaluering og små justeringer av samarbeidsform, grad av lærers involvering etc. Loggen ble skrevet så detaljert og kronologisk korrekt som det lot seg gjøre i praksis. Inneholdet i loggen bærer preg av å være subjektivt da det ikke var utarbeidet noen fast mal, men noen holdepunkter går igjen. Totalinntrykk av undervisningsøkten, elevers reaksjoner på det faglige innholdet slik de ble erfart av undertegnede, vurdering av elevarbeid undervis og undervisningsplan er alle elementer som preget loggføringen. Gjennom to undervisningsøkter ble elevene filmet mens de jobbet med programmeringsoppgaver på datamaskin (referert til som økt 9 og økt 10, se tabell 2, side 46).

For å få best mulig oversikt over elevenes arbeid, ble det plassert et kamera på siden av elevene med mikrofon plassert på skjermen. I tillegg til video og lyd av elevene, ble gratis-programmet CamStudio™ brukt for å gjøre opptak av all aktivitet på PC-skjermen. Videofilene fra kamera ble deretter satt sammen med skjermopptaket til en sammensatt film hvor elevene, elevenes arbeid og elevenes verbale kommunikasjon fremkom (se figur 7)



Figur 7: Skjermdump fra oppgavesettet i økt 10 (Se vedlegg V)

3.4 Beskrivelse av prosjektet

I følgende gis en kortfattet beskrivelse av prosessen fra det ble valgt å prøve ut programmering på trinnet, frem til det ble designet og prøvd ut undervisningsopplegg med spesifikke matematiske mål.

For at elevene skulle få erfaring med ulike programmeringskonsepter som løkker, IF-setninger, funksjoner og variabler ble det laget en plan som skulle sikre variert eksponering for de ulike konseptene. Innholdet i øktene ble evaluert og dokumentert underveis (Tabell 1). Tematisk er øktene spesielt rettet mot små spill, styring av skilpadder og interaktivitet mellom bruker og program. Totalt endte det opp med ti programmeringsøkter over to måneder hvorav de to siste ble dokumentert med film og skjermopptak av utvalgte elevpar.

Det er gjennom prosjektet brukt to forskjellige programvarer hvor elevene kan skrive inn kode og kjøre programmer; Python IDLE og Trinket.io med Python.

Tabell 1: Oversikt over undervisning i prosjektperioden

Økt	Innhold	Programmeringskonsepter	Data
Økt 1 Skilpadder Python IDLE	Lage et program som styrer en skilpadde. Mål om å lage streker, bokstaver og enkle former.	Grunnleggende (lagring, kjøring av skript), løkker, Variabler, Skilpadde-funksjoner (`forward`, `speed`, `left` etc)	Logg og dagbok
Økt 2 Hvor gammel er du Python IDLE	Lage et program som tar imot innput fra bruker og gir respons basert på input.	Input-funksjon, `if`–setninger, Variabler, Operatorer `< > == !=`	Logg og dagbok
Økt 3 IF-setninger Python IDLE	Lage et program som tar imot innput fra bruker og gir respons basert på input.	Variabler, Regneoperasjoner, Bruker-Input, `if`–setninger	Logg og dagbok
Økt 4 Skilpaddekunst / Skilpaddeskolen Python IDLE	Lage program som bruker løkke for å lage ulike former og figurer.	Regneoperasjoner, Import av bibliotek, Funksjoner (Random/Randint), Løkker, Strenger og Variabler, `if`–setninger	Logg og dagbok
Økt 5 Stjerner og galakser Python IDLE	Lage program som bruker løkke for å lage ulike former og figurer.	Turtle, Funksjoner (Random/Randint), Funksjoner(`def ..`), løkker, variabler, kommentarer	Logg og dagbok
Økt 6 Mattespill (Steg 1-4) Python IDLE	Lage et program hvor man skriver inn svar på enkle regnestykker og får respons basert på svaret.	Regneoperasjoner, Import, Random/Randint, For Loops, Strenger og Variabler, `if`–setninger	Logg og dagbok
Økt 7 Oppgavesekvens på itslearning, innebygd trinket – Fokus på matematikk (Pilot)	En serie tekstoppgaver. Elevene redigerer kode for å løse oppgavene	Variabler, Funksjoner, Regneoperasjoner	Logg og dagbok, Elevarbeid,
Økt 8 Oppgavesekvens på itslearning, innebygd trinket – Fokus på matematikk (Pilot)	En serie tekstoppgaver. Elevene redigerer kode for å løse oppgavene	Variabler, Funksjoner, Regneoperasjoner, Løkker	Logg og dagbok, Elevarbeid,
Økt 9 Oppgavesekvens gitt på ark. Elevene koder i Python-IDLE	En serie oppgaver hvor elevene skriver av/fullfører koder for å utforske tall	regneoperasjoner (inkludert modulo-operator), løkker, variabler, hvis-setning, print-funksjon, lister, funksjoner (`def`),	Logg og dagbok, Elevarbeid, Film/Lyd og opptak av skjerm
Økt 10 Oppgavesekvens gitt via et HTML-dokument med interaktiv Python-editor (trinket.io)	En serie oppgaver hvor elevene bruker delvis skrevne koder/program for å utforske tall	Sleep-funksjon, løkker, hvis-setning, print-funksjon, variabler, regneoperasjoner (inkludert modulo-operator)	Logg og dagbok, Elevarbeid, Film/Lyd og opptak av skjerm

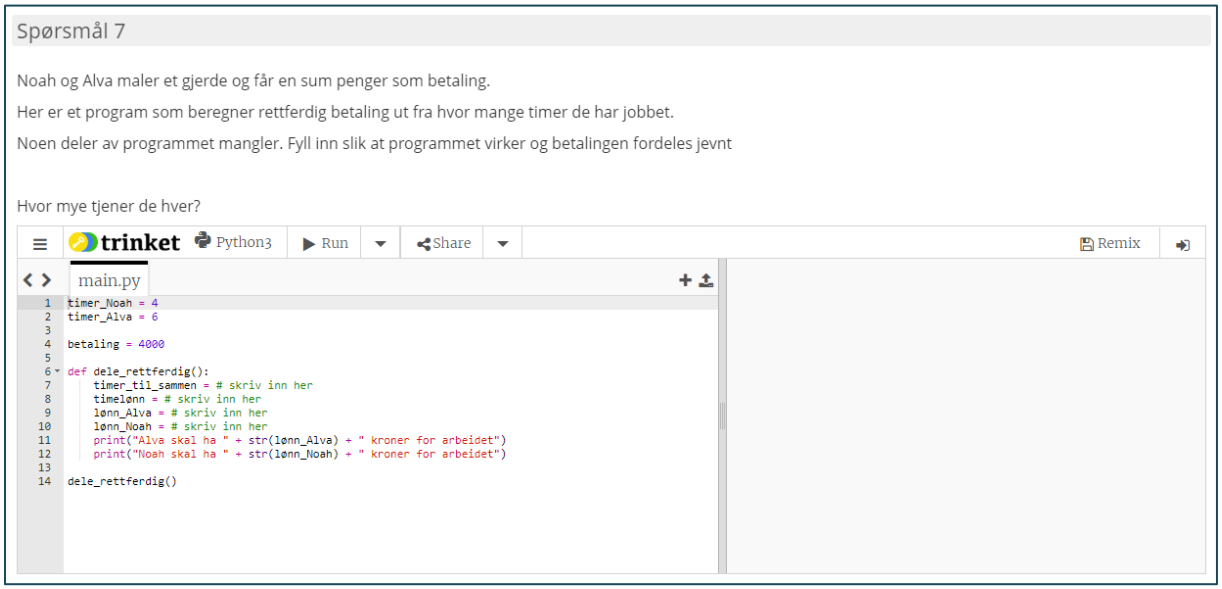
3.4.1 Undervisning i programmering (økt 1-8)

Målet i starten var at elevene skulle få erfaring med å programmere og lære seg grunnleggende konsepter samtidig som de fikk et positivt forhold til det å programmere. Oppgaver fra Kidsakoder.no ble valgt ettersom disse fremstår som gode for nybegynnere. I tillegg er ressursene lett tilgjengelig på nett med god støtte, tydelige instruksjoner og visuelle elementer, sammen med en tilsynelatende grei progresjon. En foreløpig plan ble utarbeidet med 5–6 økter i tankene.

Figur 8: Oppgave fra kidsakoder: https://oppgaver.kidsakoder.no/python/skilpadder/

Noen uker ut i prosjektet ble det eksperimentert med å gi elevene oppgaver knyttet mer spesifikt til matematikkfaget (økt 7 og 8). Dette ble gjort ved å bruke programvare («Trinket.io») som tillaterer lærer å lage ferdige kodesnutter og deretter bygge disse inn (embedded) i læringsplattformen itslearning (Figur 9). Oppgaven til elevene gikk dermed i større grad ut på å endre eksisterende kode fremfor å skrive egen. Oppgaveformatet er inspirert av oppgavene elevene allerede kjente til fra kodeklubben, kombinert med nettbaserte læringsmoduler for programmeringsinnhold som codecademy.com,

khanacademy.com og codewarriors.org. Elevene fulgte en progresjon laget på forhånd med blant annet tekstoppgaver som matematisk innhold.



Figur 9: Tekstoppgave i Python. Trinket bygget inn i itslearning

Det ble gjort mange interessante erfaringer i den første delen av undervisningsprosessen (økt 1-8), men da jeg først og fremst er interessert i resultatene fra øktene som ble filmet (økt 9-10), blir kun en oppsummering presentert i metodekapittelet for å gi et innblikk i konteksten til utdragene som presenteres i analysen. Sett bort fra tekniske utfordringer som å installere programvare på skole-PCer, få programvaren til å virke på samtlige datamaskiner og lagring av koder kan perioden oppsummeres i positive ordelag. Etter hvert som elevene ble kjent med programvaren, bar undervisningen preg av motiverte elever og god arbeidsflyt. Disse beskrivelsene er naturlig nok subjektive, men er likevel ansett som relevante da de la grunnlag for hvordan jeg utarbeidet oppgavesettene i økt 9 og 10. Erfaringene fra perioden kan oppsummeres i fem punkter:

(1) Løkker og variabler kan være vanskelig. Elevene hadde ingen større utfordringer med å lage kode som for eksempel skriver ut en tallrekke til 10 ved å bruke løkke. Plenumsdiskusjon i klassen røpte derimot at flere av programmeringskonseptene var vanskelig å forstå, spesielt løkker. Hva betyr ordene i koden «`for i in range(10)`»? Hva er en «runde»? Hvordan kan tallene både telle runder og være variabler? Arbeidet med

skilpadder ble valgt ut for å gjøre begrepet «løkke» mer konkret, da elevene gav uttrykk for å kunne forklare at løkken får skilpadden til å gjenta noe x antall ganger.

(2) Frihet eller styring? Da oppgavene ble åpnet opp, og elevene ble gitt større frihet til å leke seg, hendte det fort at noen elever fikk store ambisjoner. Det er i og for seg bra, men med begrensede lærerressurser fungerte det dårlig. Noen ville lage programmer som de ikke hadde forutsetninger for å lage på daværende tidspunkt.

(3) Fokus på matematikk? Det ble ikke fokusert på matematikkfaglig innhold de første ukene. Elevene ønsket først og fremst å leke seg, lage spill og utforske ekstreme tilfeller, ikke arbeide med matematikk. De gangene det ble forsøkt å vri fokuset mot matematikkfaglig innhold krevde det tydelig styring fra lærer da matematikkspesifikke sammenhenger i liten grad så ut til å bli vektlagt av elevene på egen hånd.

(4) Hardkoding og variabler? Da elevene fikk oppgaver å behandle variabler, for eksempel ved å lage et program som løste en enkel tekstoppgave med addisjon og subtraksjon, løste de fleste elevene dette ved å skrive inn verdiene direkte (såkalt hardkoding). Dette ble gjort i stedet for å bruke variablene slik at løsningen på problemet blir generalisert til å også gjelde for andre verdier. Her var veiledning fra lærer nødvendig. Behovet for veiledning/styring er videre imøtekomt ved å ha mye tekst i oppgavesettene, som en nettleksjon. I ettertid kan en spørre seg hvorvidt dette har medført at undervisningen i de siste øktene har vært preget av "the planning paradox" – desto mer detaljert man beskriver eller planlegger læringsmål og forventet utfall, desto vanskeligere er det for elever å se på programmering som et personlig og nyttig redskap (Ainley et al., 2006, i Misfeldt & Ejsing-Duun, 2015).

(5) Samarbeid som arbeidsform. Det ble erfart at elevene arbeidet mye bedre da de fikk jobbe sammen i par. De fleste elevene delte arbeidsoppgavene på en naturlig måte. En skrev kode mens den andre leste oppgaven og sjekket korrektur. Programmeringsutfordringene med syntaks/grammatikk i Python ble redusert betraktelig da elevene jobbet sammen. Elevparene trengte fremdeles oppfølging, men ved å ha en samarbeidspartner så det ut til å redusere behovet for lærer betraktelig.

3.4.2 Undervisning som kombinerer programmering og matematikk (økt 9)

Med utgangspunkt i foregående undervisning ble det laget et mer omfattende undervisningsopplegg. Målet var at elevene nå skulle utforske matematiske sammenhenger

knyttet til faktorisering, divisjon og rest ved å bruke Python som verktøy. Undervisningsopplegget ble gjennomført og data samlet inn. Video-, lyd- og skjermbilde ble redigert til sammensatte filmer, som deretter ble gjennomgått og evaluert med fokus på innhold. Resultatene ble sett på som interessante, men med et forbedringspotensiale. Noen av oppgavene ble vurdert til å være preget av uheldig design, med rom for misforståelser flere steder (steg 5, Vedlegg IV), men det var også enkeltoppgaver som viste potensiale ved å kombinere programmering og matematikk. Enkelte aspekter knyttet til selve programmeringen (lister og funksjoner) ble også vurdert for å være noe vanskelig, da elevene tydelig ikke hadde nok erfaring med dette. Det ble valgt å lage og prøve ut nytt undervisningsopplegg da en rekke situasjoner viste seg å legge til rette for matematisk aktivitet, mellom annet arbeid med modulo-operator for å finne restverdien når et tall divideres med et annet.

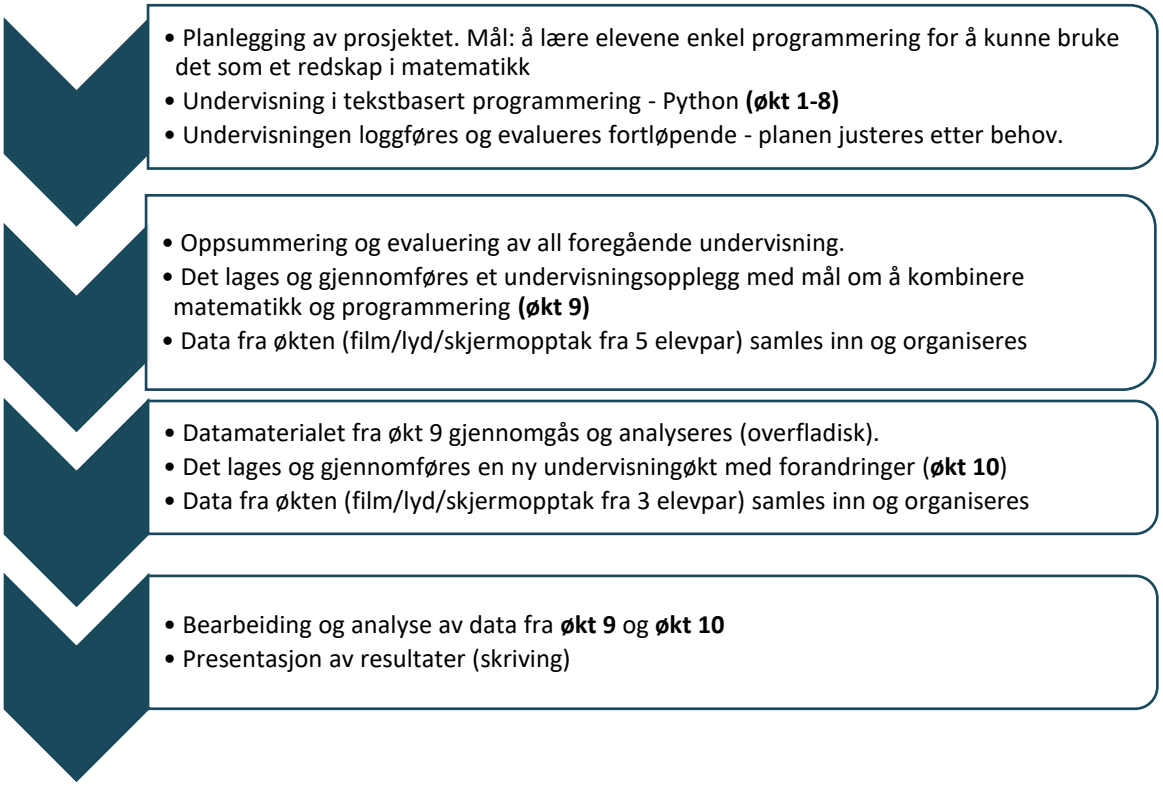
3.4.3 Nytt undervisningsopplegg (økt 10)

Oppgavene i økt 10 ble laget med bakgrunn i erfaringer fra de foregående øktene og som en respons på økt 9. I stedet for å la elevene skrive inn koden manuelt ble det valgt å bruke såkalte trinkets, en nettbasert kode-editor som kan bygges inn, for eksempel i en nettside. Dette ble gjort fordi elevene tidligere brukte mye av tiden på å skrive inn koden, fremfor å jobbe med matematikk. Erfaringene tilsa også at det å lage tallrekker ved å bruke løkker gav et spennende utgangspunkt for ulik utforsking av egenskaper ved tall og mønster: dette preget oppgavene laget til økt 10 stort ettersom de ulike kodesnuttene elevene presenteres for skriver ut ulike tallrekker og tabeller. For elevene ble fokus rettet mot å *lese* kode, *forklare* forventet utfall, og å gjøre *modifiseringer* med mål om å enten få et gitt utfall eller for å undersøke effekt.

Det ble satt som mål at elevene kunne gjennomføre oppgavene på egenhånd med lærer som fasilitator. Dette ble gjort både for å i større grad kunne bistå med tekniske spørsmål og for å beholde et fokus på utforsking. Som i økt 9, ble oppgavene også her svært rike på tekst, blant annet fordi jeg opplevde et behov for å gi elevene informasjon og retning i arbeidet, uten at jeg som lærer måtte gripe inn i elevparenes samtaler. Dette kan dog ha noen fordeler. Watson og Thompson (2015a) påpeker blant annet muligheten skriftlige oppgaver i matematikk gir og hevder de kan engasjere til matematisk refleksjon, demonstrere hvordan arbeid skal gjøres, invitere til å dele kunnskap, foreslå handlingsmåter for å løse oppgaven,

foreslå rekkefølge (gi tips?), gjennomtenkt rekkefølge på hvordan informasjon presenteres – påvirke hva som skal fokuseres på. I tillegg kan oppgaver som gis skriftlig introdusere nye måter å jobbe med matematisk innhold, gi informasjon på en strukturert og oversiktlig måte og innby til egenvurdering og refleksjon (Watson & Thompson, 2015a). Erfaringene fra økt 9 pekte spesielt på et behov for å gi konkrete føringer på eksplisitt refleksjon i møte med kodene og programmeringsinnholdet.

Oppsummert kan perioden før og under datainnsamlingen beskrives gjennom ulike faser (figur 10):



Figur 10: Oversikt over ulike stadier i prosjektet

3.5 Bearbeiding og analyse av data

Analyse av kasus-studier og etnografisk forskning innebærer ofte detaljerte beskrivelser av situasjoner, deltakere og analyse av relevante temaer. Creswell (2014) viser til seks steg for analyse av kvalitativ data som også er fulgt i denne studien: (1) Organisere data (Transkribere, scanne materiale og renskrive feltnotater), (2) se gjennom all data og beskrive hvilke generelle temaer og emner som dukker opp i materialet, avgjøre hvor troverdig

materialet er og beskrive hvilket «inntrykk» materialet gir, (3) koding av data, (4) detaljerte beskrivelser, ordnet tematisk, (5) beskrive hvordan temaene presenteres, og (6) tolkning av resultatene.

3.5.1 Steg for analyse

I første omgang ble undervisningsnotater og dokumenter renskrevet og lagt i tabeller (jamfør steg 1). Film, lyd og skjermopptak fra undervisningsøkter ble satt sammen til en bilde-i-bilde film, som er brukt videre i analysene. Grunnet stort omfang i datamaterialet ble transkribsjoner laget etter hvert som det ble klart hvilke situasjoner jeg skulle fokusere på.

Datamaterialet fra filmene ble gjennomgått i flere runder (jamfør steg 2). Den første gjennomgangen av materialet ble gjort direkte i etterkant av første runde med datainnsamling, og på bakgrunn av erfaringene derfra ble det bestemt å modifisere undervisningsopplegget før neste runde. Etter å ha gjennomført et nytt undervisningsopplegg, samlet og organisert data, ble filmene gjennomgått mer systematisk for å lete etter gjennomgående temaer, spesielt interessante hendelser. Spørsmål som «hva skjer egentlig her?», «hva handler dette om?» og «hvordan kan dette materialet belyse problemstillingen?» stod sentralt i denne fasen av analysearbeidet. Etter hvert ble det viet mer oppmerksomhet til spesifikke situasjoner og detaljer, for eksempel situasjoner hvor elever diskuterte løsninger, møtte utfordringer eller misforstod oppgavene. Notater ble laget for å oppsummere spesielt interessante hendelser underveis i de ulike filmene. Disse ble markert med tidsreferanser slik at det var lett å gå tilbake til de ulike situasjonene.

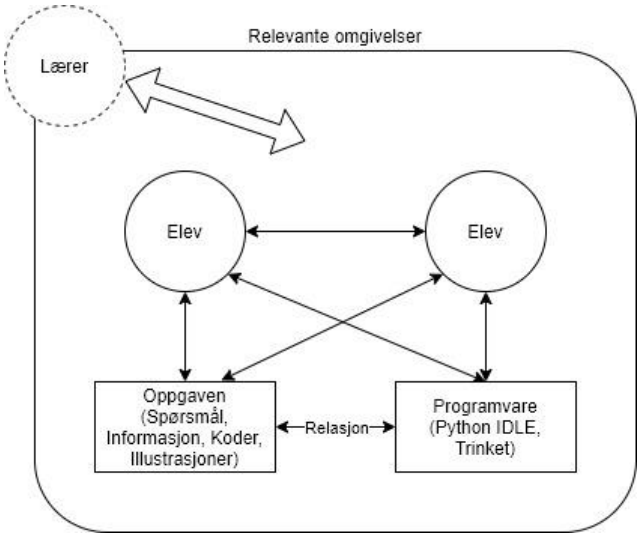
Etter en rekke gjennomganger hvor datamaterialet ble gruppert/klassifisert blant annet etter matematisk innhold, oppgaver og elevgrupper, ble det valgt å fokusere spesielt på oppgaver knyttet til divisjon, rest og modulo. I tillegg ble det valgt å fokusere på tre elevpar med begrunnelse i elevenes *ulike tilnærming* til oppgavene. Deretter ble det laget transkribsjoner av deler av filmene med skjermbilder inkludert for hver gang elevene forandret kodene eller kjørte programmene. Mindre forandringer, for eksempel i koden, ble beskrevet med firkantparentes. Transkribsjonene av utvalgte grupper i arbeid med et utvalg oppgaver dannet grunnlag for videre analyse.

Etter at utdragene ble kodet (jamfør steg 3) og innholdet analysert (grovt) med bakgrunn i valgt teori (TDS og SAMR-modellen), ble det valgt ut seks utdrag som videre ble beskrevet,

tolket og analysert mer inngående (jamfør steg 4). Elevenes ytringer og handlinger ble brukt for å avgjøre hvorvidt utdragene representerer aksjons-, formulerings- eller valideringssituasjoner, kvaliteten på argumentasjonen og hvorvidt elevene møtte ulik motstand. Presentasjon og tolkning av resultatene (jamfør steg 5 og 6) presenteres i kapittel 4.

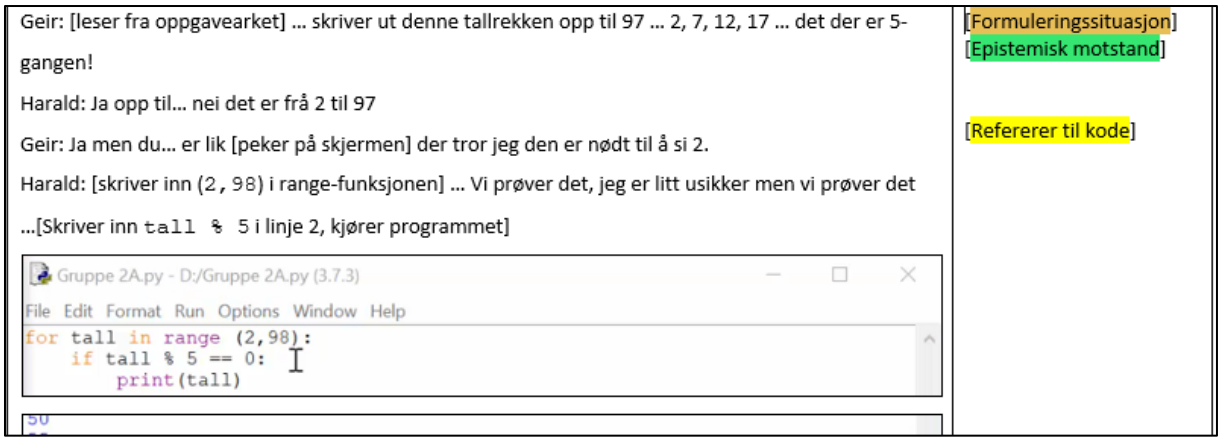
3.5.2 Koding og analyse av valgte utdrag

For å studere samspillet mellom elever, oppgave og programvare er datamaterialet blant annet analysert for å se hvorvidt situasjoner som sammenfaller på henholdsvis aksjon-formulerings- eller valideringssituasjoner har funnet sted. Utfordringen nevnt av Artigue et al. (2014) om å klassifisere undervisning i lys av situasjonsformene blir dermed et mål for analysen i denne masteroppgaven (dog fremdeles utfordrende). Denne tilnærmingen skiller seg også noe fra eksemplene Brousseau skildrer, for eksempel i «the race to 20», hvor hele økter klassifises for eksempel som en formuleringssituasjon. Jeg har også måtte tolke situasjonen litt annerledes enn Brousseau da de fleste utdragene i datamaterialet viser situasjoner som er kontinuerlig a-didaktiske. Det er ikke planlagt lærer-intervensjon verken i eller mellom oppgavene. I visse tilfeller spør elevene om hjelp, og det kan føre elevene fra den a-didaktiske ut i den didaktiske situasjonen (generelle), men det er i undervisningsoppleggene ikke planlagt for at lærer hjelpe elevene til å formulere eller validere. Oppgavene i seg selv er laget med tanke på å fremprovosere diskusjon og matematisk aktivitet, men hvorvidt dette faktisk har ført til formulering eller validering er gjenstand for analysen. Dette betyr at spørsmål om hvorvidt elevene går inn i de ulike situasjonene er avgjort på bakgrunn av innholdet i utdragene (kontekst).



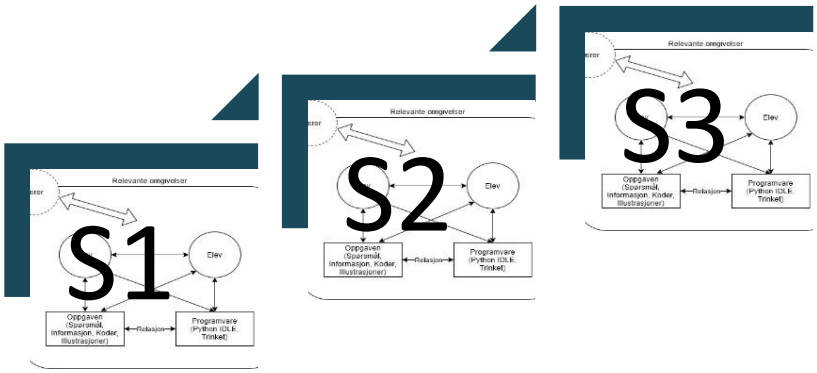
Figur 11: Analyseenhet laget med inspirasjon fra Brousseau (2006)

I alle utdragene som presenteres er omgivelsene preget av en spesifikk ramme. Brousseau (2006) bruker begrepet *milieu* mer spesifikt enn det dagligdagse ordet «omgivelser», på den måten at milieu omhandler de *relevante elementene* for den *faglige situasjonen* lagt frem av læreren, og disse trenger ikke bare inkludere de rent fysiske. Omgivelsene er først og fremst preget av eleven selv og en medelev (et elevpar), som sitter i en mer eller mindre symmetrisk relasjon til hverandre. I tillegg er datamaskinen med den aktuelle programvaren (Python) og oppgavene (presentert fysisk eller digitalt) de viktigste faktorene inkludert i omgivelsene. Relasjonen mellom elev, oppgave og et programmeringsverktøy (analyseenheten) illustreres i figur 12. Læreren er en del av omgivelsene i et av tilfellene hvor elevene ber om hjelp. Elevenes sosiale forhold til hverandre er også relevant for situasjonene, kanskje spesielt i tilknytning til argumentasjon (bruk av retoriske grep), men det er valgt å ikke gå i dybden på disse.



Figur 12: Utdrag fra analysedokument. Transkribsjon med skjermdump og koder

Under analysen ble det valgt å dele inn utdragene i mer avgrensede deler basert på situasjonene (aksjon, formulering og validering) samt oppgaver elevene jobbet med. I figur 11 vises et utdrag fra transkribsjonene hvor elevene har startet på en ny oppgave og fokus flyttes fra foregående oppgave til noe nytt. Her er det valgt å si at elevene trer inn i en ny *situasjon*. Utdragene ble delt opp i flere situasjoner (**S1**, **S2, S3** …) basert på oppgavene og innholdet elevene fokuserte på (Figur 13). Ordet situasjoner er her tett forbundet med de ulike didaktiske situasjonene beskrevet i teorikapittelet. Basert på elevenes ytringer ble eksempelet i figur 12 tolket som en *formuleringssituasjon*. Elevene er språklig engasjert, utveksler informasjon knyttet til oppgaven, og formidler løsningsforslag. I tillegg tolkes elevenes reaksjon på oppgaven som at elevene engasjerer seg, men de har ikke en umiddelbar løsning på problemet, noe som også gir elevene matematisk motstand.



Figur 13: Analyseenhet i serie. Påfølgende situasjoner danner grunnlag for dialektikk

Underveis i analyseprossen ble det også et behov for å lage et sett koder for å markere når elevene referte til ulike elementer knyttet til *kode* (det de skrev inn), *oppgave* (tekst, bilde og symboler) og *utfall* (output av program, vises i skallet eller i konsollen). Dette ble gjort da problemstillingen omhandler samspillet mellom elever, oppgave og programvare. Hvordan elevene refererer til og bruker omgivelsene deres (*milieu*) i oppgaveløsing og argumentasjon er av stor interesse for å beskrive dette samspillet. Disse kodene ble brukt hver gang elevene *aktivt refererte* til enten kode, oppgave eller utfall.

3.6 Beskrivelse av oppgavene og det matematiske innholdet

Brousseau (2006) beskriver det matematiske innholdet som en viktig komponent i didaktikken. I denne studien har det hele veien vært et uttalt mål å ha fokus på det matematiske innholdet, men i motsetning til situasjonene som beskrives hos Brousseau (2006), er oppgavene klart mer rettet mot utforsking fremfor generell problemløsning, noe som gir føringer for hvordan innholdet kan analyseres. I oppgavene er det matematiske innholdet rettet mot divisjon, rest og delelighet, samt en underliggende forståelse av begreper som modulær divisjon, divident, divisor og kvotient. Utforskingen er tenkt å bunne ut i en forståelse for sammenhengen mellom disse begrepene, samt at elevene får erfaringer med programmeringskonsepter som løkker, variabler og betingelser.

I det følgende presenteres kort oppgavene som de valgte data-utdragene baserer seg på. For fullstendig oversikt over alle oppgavene, se vedlegg IV og V.

3.6.1 Oppgave 1: Utforske modulo-operator

I denne oppgaven ble elevene bedt om å bruke modulo-operatoren (%-tegnet) i Python-skallet for å sjekke rest og delelighet ved ulike tall (Figur 14). Python-IDLE (programvaren) består av et vindu hvor man skriver kode, og et skall; et interaktivt vindu hvor resultatet fra koden visees, eller hvor man kan skrive inn korte Python-kommandoer som kjøres direkte (når man trykker enter). Python skallet vil i tillegg til å tolke funksjoner, variabler og andre programmeringselementer, også tolke matematiske utrykk. Man får dermed umiddelbar tilbakemelding, akkurat som i en kalkulator, men med mulighet for å skrive programmer i tillegg.

Steg 2: Utforske modulo-operatoren

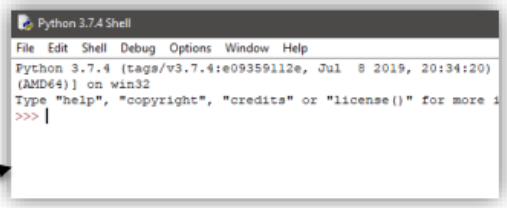
Modulo-operatoren (%-tegnet) gir oss det som blir til overs (**rest**) når et tall deles med et annet.

```
>>> 15 % 3
0
>>> 15 % 4
3
>>> 10 % 3
1
>>> 10 % 5
0
```

15 delt på 3 gir 0 i rest. Det betyr at 15 er **delelig** med 3.

Det betyr også at 3 er en **faktor** i 15 (fordi 3 multipliseret med 5 blir 15)

15 er **ikke delelig** med 4, fordi da er det **3 i rest**.



Oppgave 2: Skriv i «Skallet»

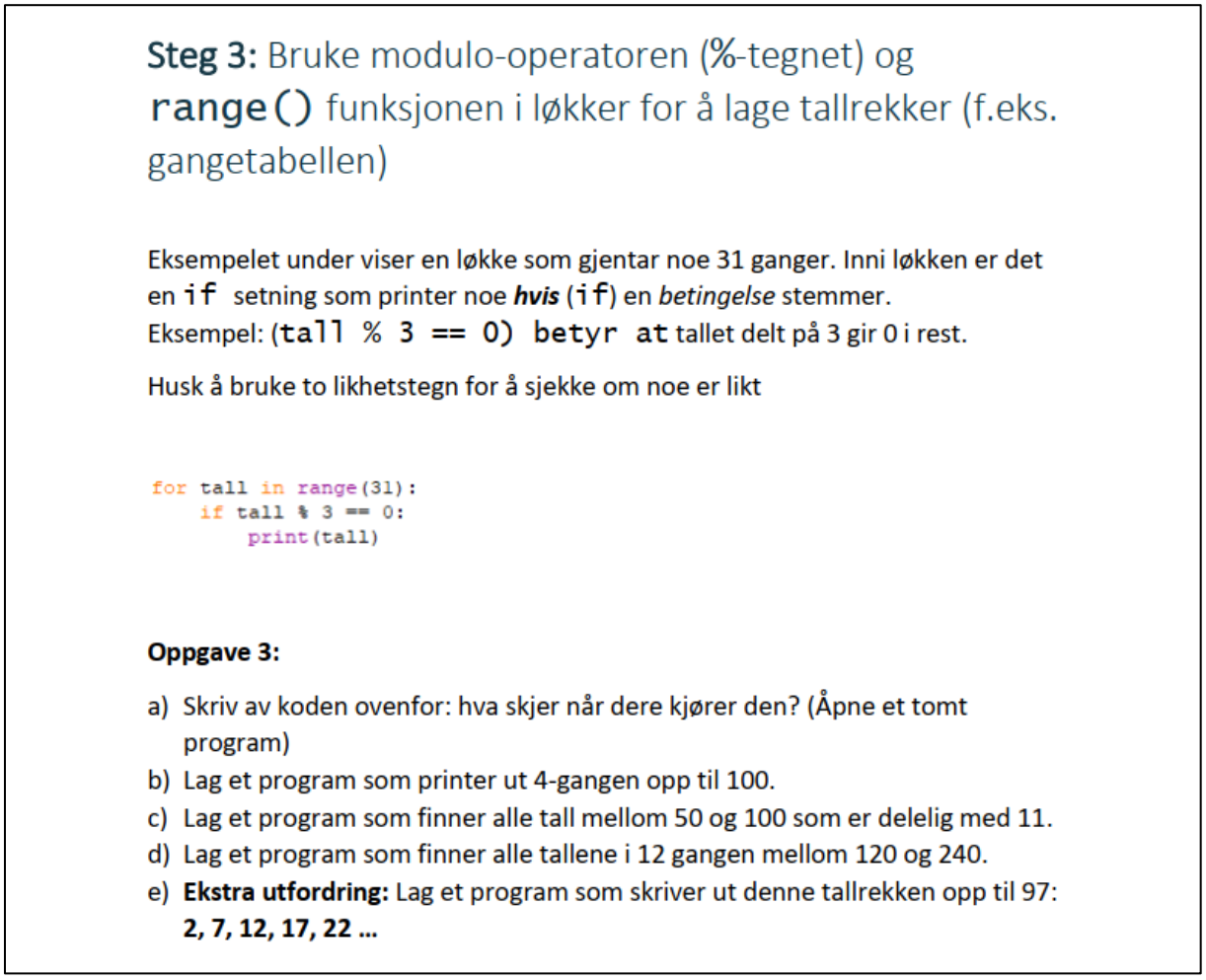
a) Hva får du i rest hvis du deler 45 på 6? Bruk modulo-operatoren (%-tegnet) til å sjekke.
b) Bruk modulo-operatoren til å finne tall som tall er delelige med 13. Er 39 delelig med 13? Er 113 delelig med 13?
c) Er 41 en faktor i 533? Dere kan også bruke deletegnet (/) for å sjekke.
d) Er 818 delelig med 8? Hvorfor / hvorfor ikke?
e) Er 13113 delelig med 13? Hvorfor / hvorfor ikke?

Figur 14: Oppgave 2, økt 9

3.6.2 Oppgave 2: Løkker og tallrekker

Oppgaven viser en kodesnutt som elevene skal skrive av i kodevinduet (en tom «tekstfil» som kan kjøres slik at resultatet vises i python-skallet). Deretter blir elevene bedt om å forklare hva som skjer når programmet kjøres. Koden i oppgaven inneholder en `for`-løkke som gjentas 31 ganger (fra 0 til 30), hvor det i hver runde av løkken skrives ut hvilket rundenummer man er på dersom (`if`) rundenummeret (`tall`) delt på `3` gir `0` i rest (`if tall % 3 == 0`). Dersom rundenummeret delt på `3` ikke gir `0` i rest, skjer ingenting.

Utfallet av å kjøre programmet slik det er gitt i oppgaven vil derfor være tallrekken 0, 3, 6 … opp til 30, altså 3-gangen. I de følgende deloppgavene ble elevene bedt om å lage ulike tallrekker ved å modifisere programmet.



Steg 3: Bruke modulo-operatoren (%-tegnet) og `range()` funksjonen i løkker for å lage tallrekker (f.eks. gangetabellen)

Eksempelet under viser en løkke som gjentar noe 31 ganger. Inni løkken er det en `if` setning som printer noe ***hvis*** (`if`) en *betingelse* stemmer.
Eksempel: (`tall % 3 == 0`) **`betyr at`** tallet delt på 3 gir 0 i rest.

Husk å bruke to likhetstegn for å sjekke om noe er likt

```
for tall in range(31):
    if tall % 3 == 0:
        print(tall)
```

Oppgave 3:

a) Skriv av koden ovenfor: hva skjer når dere kjører den? (Åpne et tomt program)
b) Lag et program som printer ut 4-gangen opp til 100.
c) Lag et program som finner alle tall mellom 50 og 100 som er delelig med 11.
d) Lag et program som finner alle tallene i 12 gangen mellom 120 og 240.
e) **Ekstra utfordring:** Lag et program som skriver ut denne tallrekken opp til 97: **2, 7, 12, 17, 22 …**

Figur 15: Oppgave 3, økt 9

3.6.3 Oppgave 3: Utforske tabeller med rest

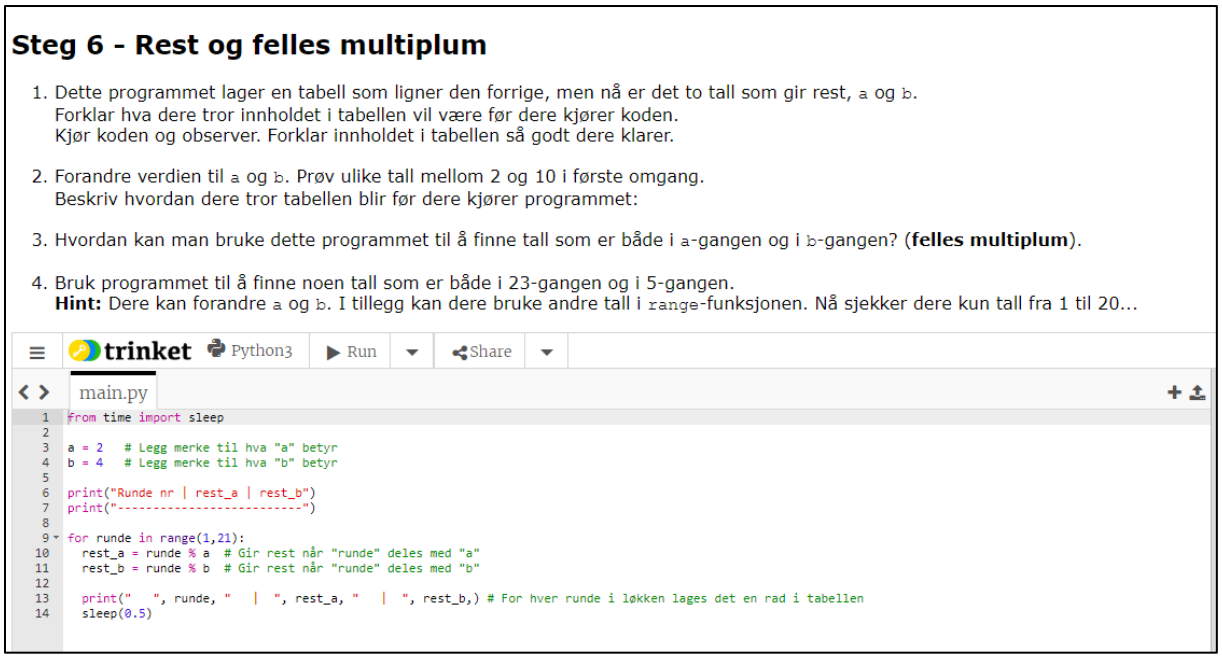
I økt 10 fikk elevene presentert et sett oppgaver med en trinket. På grunn av den tette sammenhengen mellom to av oppgavene ble de i analysen slått sammen og beskrevet sammenhengende. I trinketen ble det på forhånd skrevet inn en kode av lærer som del av oppgaven.

I steg 5 av oppgavesettet (Figur 16) ble elevene bedt om å forutse utfallet når koden kjøres, samt prøve å si hvilke verdier variabelen `rest` (linje 9) får underveis mens koden kjører. `Rest`-variabelen får verdien av å dele rundenummeret (`runde`) med variabelen `a`.

Programmet skriver ut en tabell, en rad for hver runde i løkken, hvor kolonnen til høyre viser verdien til `rest`. I tillegg er det lagt inn en funksjon (`sleep(0.5)`) som får programmet til å vente et halvt sekund for hver linje som skrives ut. Etterpå ble elevene først bedt om å forandre `a` og så forutse utfallet, deretter ble de bedt om å modifisere programmet slik at det kan brukes til å finne tall som er delelig med 33 og 44.

Figur 16: Oppgave 5, økt 10.

I den påfølgende oppgaven (Figur 17) er oppgaveteksten og koden noe forandret. På samme måte som i forrige oppgave skriver programmet ut en tabell, men nå inneholder tabellen to variabler som tilegges verdier ved å dele rundenummeret på henholdsvis 2 og 4. Igjen brukes modulo-operatoren, slik at variablene `rest_a` og `rest_b` får nye verdier som tilsvarer rest-verdien når rundenummeret deles på henholdsvis `a` og `b`.



Steg 6 - Rest og felles multiplum

1. Dette programmet lager en tabell som ligner den forrige, men nå er det to tall som gir rest, `a` og `b`.
   Forklar hva dere tror innholdet i tabellen vil være før dere kjører koden.
   Kjør koden og observer. Forklar innholdet i tabellen så godt dere klarer.

2. Forandre verdien til `a` og `b`. Prøv ulike tall mellom 2 og 10 i første omgang.
   Beskriv hvordan dere tror tabellen blir før dere kjører programmet:

3. Hvordan kan man bruke dette programmet til å finne tall som er både i `a`-gangen og i `b`-gangen? (**felles multiplum**).

4. Bruk programmet til å finne noen tall som er både i 23-gangen og i 5-gangen.
   Hint: Dere kan forandre `a` og `b`. I tillegg kan dere bruke andre tall i `range`-funksjonen. Nå sjekker dere kun tall fra 1 til 20...

trinket | Python3 | Run | Share

main.py

```
from time import sleep

a = 2   # Legg merke til hva "a" betyr
b = 4   # Legg merke til hva "b" betyr

print("Runde nr | rest_a | rest_b")
print("-------------------------")

for runde in range(1,21):
  rest_a = runde % a  # Gir rest når "runde" deles med "a"
  rest_b = runde % b  # Gir rest når "runde" deles med "b"

  print("   ", runde, "   |  ", rest_a, "   |  ", rest_b,) # For hver runde i løkken lages det en rad i tabellen
  sleep(0.5)
```

Figur 17: Oppgave 6, økt 10.

3.7 Validitet og reliabilitet

Validitet handler om hvorvidt resultatene, funnene og påstandene som blir fremhevet i studien er nøyaktige og til å stole på (Creswell, 2014). Det blir nevnt flere grep innen kvalitativ forskning som kan styrke studiers validitet, blant annet rike beskrivelser, datatriangulering, redegjørelse av forskers forforståelse (bias), motstridende resultater og varighet i felt/tidsspenn i datainnsamling (Creswell, 2014). I denne studien er det forsøkt å skape troverdighet ved å følge flere av disse grepene. Det har for eksempel vært et mål å være åpen om egen forforståelse og forskers nære tilknytning til deltakerne. I kvalitativ metode står forskeren sentralt (Creswell, 2014) og i problemformuleringen ligger forskerens perspektiv tett under overflaten: Hvordan kan *jeg* legge til rette for matematisk aktivitet ved å bruke programmering som undervisningsverktøy? Som forsker påvirker også min forforståelse for temaet valg av forskningsdesign, datainnsamling samt tolkning og bearbeiding av data.

Ettersom arbeidet med prosjektet har foregått over lang tid og dette har hatt betydning for valg som er tatt, har det vært et mål å gi detaljerte og rike beskrivelser av hele prosessen, også av valg og refleksjoner som er gjort i forkant av datainnsamlingen. Det er også forsøkt å gjøre rede for og belyse det som ikke har gått etter planen. For eksempel er behovet for å

diskutere oppgavene i plenum – trekke slutninger, oppsummere og kontekstualisere (*institusjonalisering*) – fokusert mindre på. Valg som i utgangspunktet virket naturlige, logiske eller fornuftige, men som i ettertid kan kritiseres, kan imidlertid bidra til å belyse mulige utfordringer ved undervisningsmetodikken som er beskrevet, og med dette bidra til ny kunnskap og utvikling av fagfeltet. Jeg vil påstå at de transperente og detaljerte beskrivelsene av prosessen underveis styrker studiens troverdighet, samtidig som de viser kompleksiteten i undervisnings-landskapet og utfordringene med å eksperimentere og prøve ut nye ting i en undervisningssituasjon.

Med utgangspunkt i det kvalitative forskningsdesignet som ligger til grunn for studien, har det ikke vært et mål å generalisere funnene, men påpeke noen muligheter og utfordringer i de *spesifikke* kasuistikkene som presenteres. Alle elever og situasjoner er ulike og resultatene må sees i lys av at mange ulike faktorer påvirker situasjonene som er beskrevet.

Videre er det innenfor kalitativ forskning viktig å påse at transkripsjonene har høy reliabilitet (Creswell, 2014). Dette handler om hvor nøyaktige transkripsjonene er og i hvilken grad de representerer virkeligheten. Transkribsjoner er fortolket og representerer kun begrenset data fra konteksten (Cohen et al., 2007, s. 367), men det er viktig å påse at disse er så nøyaktige og reliable som mulig. Ettersom datamaterialet har bestått av både video- og lydopptak av elever, samt skjermopptak av arbeidet, har dette gitt et godt (rikt og informativt) utganspunkt for grundige transkripsjoner og analyser. Jeg har med dette fått mulighet til å beskrive et komplett bilde av situasjonen ved å inkludere observasjoner av elevenes handlinger i rommet, det de sier, samt gester og handlinger på datamaskinen (musepeker, markering, inndata etc.). I den grad det fremkommer tydelig er det også inkludert deltakeren sitt humør eller fremtoning (irritabel, sint, ler). Skjermbilder av hendelser er også viktige for å forstå hendelsene, og disse er flettet inn i transkribsjonene på en kronologisk måte.

3.8 Etiske hensyn

I denne oppgaven er forskningsprosjektet satt i konteksten av egen arbeidsplass. Creswell (2014) viser til slik forskning som «backyard-research» og påpeker både fordeler og utfordringer. Av utfordringer nevnes blant annet det ubalanserte maktforholdet mellom

forsker og deltakere og at informasjonen som samles kan være preget av unøyaktighet. I tillegg kan slik forskning forringe forholdet mellom forsker og deltaker. Flere valg er imidlertid tatt for å holde en etisk ryddig standard gjennom prosjektet.

Som faglærer for klassen har det viktigste vært å skille mellom ordinær undervisning og undervisningssituasjoner med datainnsamling som del av forskningsprosjektet. De innledende øktene med programmering (økt 1-8) er en naturlig del av oppgaven, men er regnet som ordinær undervisning. Dette begrunnes med at det faglige innholdet til en viss grad har vært dekket i nåværende læreplan (LK06). I tillegg har det å jobbe med programmering blitt godkjent fra ledelsen. Det er også viktig å påpeke at data brukt fra den innledende perioden først og fremst har omhandlet undervisningsplanlegging og faglig innhold, ikke elever. Elevresponsen som blir beskrevet i forbindelse med disse øktene er av mer generell karakter og ikke rettet mot enkeltelever eller elevpar.

Planene for prosjektet ble først avklart med avdelingsleder og rektor, før de ble presentert for elevene. Denne informasjonen ble gitt i starten av skoleåret, før noen form for undervisning hadde funnet sted. Elevene ble forklart at alle elevene i klassen skulle jobbe med det samme innholdet som del av ordinær undervisning. Ved å delta i masterprosjektet, ville de i et par økter mot slutten bli filmet når de jobbet med utvalgte oppgaver parallelt med klassen på et grupperom hvor det ble tatt opp lyd-, film- og skjermopptak.

Det ble påpekt at deltakelse var frivillig og at informasjon fra prosjektet ikke ville bli brukt som del av skolens arbeid, verken utviklingsarbeid, som materiale til utviklingssamtale eller i fagets vanlige vurderingsarbeid. Siden jeg som elevenes faglærer umulig kan behandle datamaterialet helt objektivt, kan det tenkes at dette er problematisk, men da elevene ikke får karakterer på 7. trinn er dette regnet som akseptabelt så lenge elevene og foresatte er informert om hvordan datamaterialet blir behandlet og hva det brukes til. Ingen øvrige lærere, verken kontaktlærere eller andre på skolen er gitt tilgang til datamaterialet, noe elevene ble informert om. Elevene ble også informert om at dersom de ville være med, kunne de når som helst velge å trekke seg og få slettet informasjon om seg selv, selv om de tidligere hadde sagt ja til å delta. Elevene fikk også mulighet til å stille spørsmål om prosjektet før informasjons- og samtykkeskjema til slutt ble delt ut.

Å bruke elever (barn) som informanter fører med seg et betydelig ansvar (Tangen, 2010). Samtykke både fra barnet selv og foresatte har vært en selvfølge og en forutsetning. Søknad som inkluderer informasjonsskriv til foreldre og elever (Vedlegg I og II) er sendt til NSD gjennom LATACME-prosjektet og ble godkjent før datainnsamlingen startet. Det ble her presisert at datamaterialet behandles konfidensielt og anonymisert. Filmer og lydopptak som ikke lett lar seg anonymisere er lagret på ekstern hardisk og behandlet uten tilgang til nett.

4 Analyse

For å kunne si noe om muligheter og utfordringer med tekstbasert programmering, samt samspillet mellom elev, oppgave og programvare, er det spesifisert tre forskningsspørsmål:

1. Hva kjennetegner elevens handlingsmønster i interaksjon med oppgave og programvare (Python)?
2. Hvordan påvirker oppgavedesignet, sammen med verktøyet, elevenes argumentasjon og resonnering?
3. Hvilken didaktisk funksjon spiller programvaren i de valgte situasjonene?

For å besvare forskningsspørsmålene er det valgt å fokusere på seks utdrag fra datamaterialet hvor elever interagerer med matematiske oppgaver i et programmeringsmiljø. Utdragene er valgt blant annet fordi de belyser mangfoldet i datamaterialet for øvrig, noe som fremkom etter den første gjennomgangen av materialet. De første analysene viste at gruppene reagerte ulikt på konteksten og hadde ulike måter å samhandle på i arbeidet med oppgavene. I tillegg representerer gruppene elever med varierende kunnskaper og ferdigheter innen programmering. Det store mangfoldet i datamaterialet bidrar til rike observasjoner som er nyttige i det videre arbeidet med å påpeke både muligheter og utfordringer med å bruke Python-programmering i matematikkundervisning.

Utdragene er et resultat av en analyseprosess (beskrevet i metodekapittelet) hvor et omfattende datamateriale bestående av film, lyd og skjermopptak er redusert til kortere dialogsekvenser med enkelte skjermdumper. Spesielt i prosessen hvor datamaterialet er kodet med bakgrunn i et teoretisk perspektiv, har noen sekvenser skilt seg ut som særlig interessante. Dette kan for eksempel innebære dialogsekvenser rike på matematisk innhold (formulering/validering), men også sekvenser med et tydelig fravær av dette. I forkant av utdragene skildres deler av konteksten sett som relevant for utdraget, for eksempel hva elevene har jobbet med, og hvis det er aktuelt, en kort forklaring av tidligere arbeid. Utdragene presenteres hver for seg som enkeltstående kasus, men det må tas i betraktning at presentasjonen baserer seg på en større kontekst (beskrevet i metodekapittelet), som grunnet oppgavens begrensede omfang ikke utdypes videre.

4.1 Geir og Harald

Det er valgt å inkludere to utdrag fra elevparet Geir og Harald. Ingen av guttene skiller seg ut som høytpresterende i matematikkfaget, men i lys av programmeringsaktivitetene i tilknytning til dette prosjektet har guttene vist stor glede og engasjement. Utdragene er inkludert først og fremst ettersom de inneholder situasjoner med matematikkfaglige diskusjoner og -innhold. Enkelte av utdragene illustrerer også en spennende utvikling i elevenes forhold til programvaren og oppgaven.

4.1.1 Utforsking av modulo-operator

I starten av det første oppgavesettet skal elevene utforske modulo-operatoren (kapittel 3.6.1). På oppgavearket er det gitt eksempler hvor operatoren er brukt etterfulgt av et par oppgaver hvor de selv må bruke operatoren for å sjekke hvorvidt ulike tall er delelig med hverandre eller om et tall er en faktor i et annet. Første oppgave spør hva man får i rest dersom man deler 45 på 6. Geir og Harald starter med å lese eksemplene på arket og er så ivrige at de foreløbig glemmer oppgaven:

Geir: [Leser eksempelet på oppgavearket] Okey, det gav ikke mening!
Harald: Å ja, ja, det er sånn deling...
Geir: 15 prosent 3. Å ja, det er sant, det er under 0 så da kommer det på 0.
Harald: Mhm [Nikker]
Geir: 15 prosent av 4 er jo også ... hæ? Hvis vi prøver bare 1. [skriver inn `1 % 1` i skallet]. Da får vi 0! [Trykker enter (**Figur 1A**)]
Harald: Ja, da får vi 0!
Geir: Hvis vi tar 2 av 1... [skriver inn `2 % 1` (**Figur 1B**)]
Harald: Så blir det ... [Trykker enter]
Begge: 0! [i kor]
Geir: Så hvis vi tar 1 av 2 … [skriver inn `1 % 2` (**Figur 1C**)]
Harald: Så blir det 1! [Trykker enter]
	1A
	`>>> 1%1`
	`0`
	1B
	`>>> 2%1`
	`0`
	1C
	`>>> 1%2`
	`1`

Geir: Åja ja! Det divideres på det! [peker på arket]
Harald: Ja, jeg vet det... men 15 divideres på 3. Da er det ikke 0, da er det 5!
Geir: Ja, jeg vet ... men vi bare ...
Harald: Nei, nei, nei! Det vises jo hva det er i rest. Det er ingen rest her, med 15 delt på 3. Men 15 delt på 4 det går ikke, da er det litt rest igjen.
Geir: Ja, og 10 delt på 3 er 1 i rest ... og 10 delt på 5 det går! [viser til eksemplene på oppgavearket (**Figur 2A**)]
Harald: Og det er 0!
Geir: Det viser rest!
Harald: Mhm! Det er når et tall deles på …
	2A
	`>>> 15 % 3`
	`0`
	`>>> 15 % 4`
	`3`
	`>>> 10 % 3`
	`1`
	`>>> 10 % 5`
	`0`

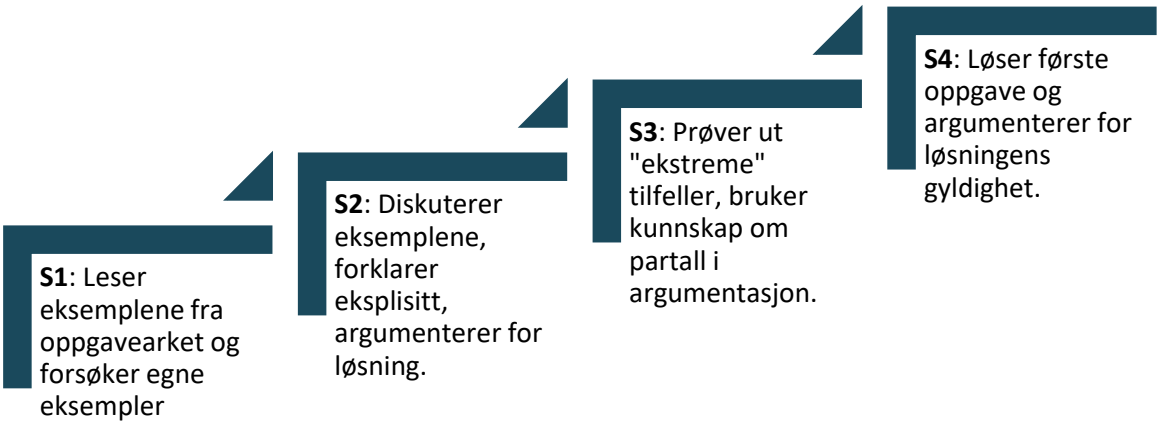
Guttene velger videre å prøve ut noen «ekstreme» tilfeller:
 	
Geir: Vi tar sånn … salamanana … sånn masse tall [skriver inn 64782682 % 2, Harald leser opp tallet med ord]
Harald: Hva blir det? [Geir trykker enter (**Figur 3A**)]
Geir: Null! Går det?! [kikker på skjermen] … Å ja, det er sant. Det er partall!
	3A
	`>>> 64782682%2`
	`0`

Geir skriver på nytt inn et lengre tall og guttene får bekreftet at tallet delt på 2 gir 1 i rest. Først nå retter guttene oppmerksomheten fra eksemplene til selve oppgaven som ber guttene bruke modulo-operatoren til å sjekke hva man får i rest når 45 deles på 6:
 	
Geir: 45 % 6! Ikke sant? [Geir skriver inn `45 % 6`, trykker enter (**Figur 4A**)]
Harald: Ja, det blir 3 i rest!
Geir: … ehm, hvordan? Hvis vi… ? Ja, det gir mening, når vi tar 3 minus 45 så får vi jo 42, som da er 6 ganger 7! Så da går jo det opp!
	4A
	`>>> 45 % 6`
	`3`

I møte med oppgaven er Geir raskt ute med å påpeke: «… det gav ikke mening!» Harald følger opp med at «Det er sånn deling» og viser at han har en formening om hvordan modulo (prosent-tegnet) virker, men han sier foreløpig ikke hva denne sammenhengen måtte innebære. Geir ser også ut til å ha en viss formening om hvordan modulo virker da han tidlig uttrykker «15 prosent 3, åja det er sant, det er under 0 så da kommer det på 0», men det er usikkert hva han referer til. Når Geir deretter leser av neste eksempel på oppgavearket «15 prosent av 4 er jo også ... hæ?», viser situasjonen at guttenes forståelse av begrepet kommer til kort og de får et behov for å utforske sammenhengen nærmere.

Guttene innleder en fase hvor de prøver ut egne eksempler, uavhengig av oppgavetekst eller instruks fra lærer. Først prøver guttene ut tre eksempler som ser ut til å bygge på en strategi; Det starter med at Geir sier: «Hvis vi prøver bare 1» etterfulgt av «Hvis vi tar 2 av 1...» og «Så hvis vi tar 1 av 2…». Før guttene trykker enter på det siste eksempelet spår Harald korrekt «Så blir det 1!». Etter å ha konstantert at «Det viser rest!» og «Det er når et tall deles på …», forsøker guttene to nye eksempler som kan kalles ekstreme da de inneholder en rekke tilfeldige siffer: «Vi tar sånn… salamanana … sånn masse tall». Når det så vises 0 i rest resonnerer Geir seg frem til at «åja det er sant, det er partall!». Til slutt blir guttene oppmerksomme på oppgaveteksten som ber dem om å bruke modulo-operatoren til å finne rest når 45 deles på 6. Geir skriver inn `45 % 6` og Harald konstaterer raskt «Ja, det blir 3 i rest!». Geir ser først ut til å være usikker på om output gir mening «… ehm, hvordan?», men svarer seg selv ved å resonnere høyt: «… når vi tar 3 minus 45 så får vi jo 42, som da er 6 ganger 7!».

Utdraget deles i fire situasjoner (**S1-4**) basert på innholdet i diskusjonene og oppgavene elevene jobber med.



Figur 18: Oversikt over situasjoner. Geir og Harald utforsker modulo-operator

I første situasjon (**S1**) leser elevene eksemplene på arket og prøver ut egne eksempler. Situasjonen starter som *aksjonssituasjon* ved at guttene leser eksemplene, en klart innarbeidet strategi som går igjen hos dette elevparet. Guttene har kun sett modulo-operatoren ved èn tidligere anledning, men da ble det ikke fokusert på fra lærers side. Guttene erkjenner tidlig at deres nåværende kunnskap kommer til kort for å beskrive modulo-operator. Dette kommer til syne da Geir i møte med det andre eksempelet (`15 % 4 = 3`) sier «... hæ?» og er tydelig forvirret. For Geir og Harald representerer situasjonen en utfordring eller et problem av matematisk karakter (motstand): Hvordan virker modulo-operatoren? Geir kommer umiddelbart med et forslag til hva de kan gjøre: «hvis vi prøver bare 1 [skriver inn 1 % 1 i skallet]. Da får vi 0!»

Verken oppgaveteksten eller lærer har på dette tidpunkt bedt elevene prøve ut egne tall, men for Geir ser dette ut til å være en naturlig strategi i møte med problemet (*å forstå modulo*). At guttene også velger en omvei ved å gjøre noe som ikke kreves av dem i møte med oppgaven tyder på at guttene er *matematisk motivert* (motivasjon til å løse problemet) fremfor *didaktisk motivivert* (motivasjonen er først og fremst er rettet mot å fullføre en oppgave). Ifølge Joubert (2017, s. 21) er det slike situasjoner som kan lede til utvikling av hypoteser, strategier, begrunnelser, bevis og forklaringer. Dette er en *formulerissituasjon* da begge guttene er tydelig fokusert på et problem – hvordan virker modulo-operatoren? – og begge ser ut til å ha en systematisk tilnærming som de jobber seg gjennom samtidig som de er i aktiv dialog med hverandre med fokus på problemet. Språkbruken elevene imellom er

også preget av det Brousseau (2006, s. 61) kaller deling av informasjon. Etter hvert som elevene skriver inn og prøver ut ulike kombinasjoner sier Geir: «Hvis vi prøver bare 1 [skriver inn 1 % 1 i skallet], da får vi 0!» Guttene klarer også å forutse noen av utfallene (Geir skriver inn 1 % 2, hvorpå Harald sier «Så blir det 1!»). En slik utveksling av informasjon hvor det implisitt ikke stilles krav om eller følger med begrunnelser, forklaringer eller lignende, er ifølge Brousseau (2006) typisk for elevers *formulering.* De bygger opp et formålstjenlig språk for situasjonen basert på relevante elementer i omgivelsene; hverandre, oppgaveteksten og programvaren.

At guttene helst spesifikt bruker tallene `1` og `2`, indikerer at de er bevisst flere sammenhenger knyttet til bruken av modulo, blant annet i tilknytning til begrepet deling og det faktum at rekkefølgen ser ut til å spille en rolle (De skriver inn både `1 % 2` og `2 % 1`). I tillegg kan situasjonen sies å være progressiv i den forstand at elevene ser ut til å basere en handling på en tidligere en, eksemplifisert ved at Geir skriver inn `1 % 2` umiddelbart etter å ha skrevet `2 % 1`.

Når Harald forklarer videre at «Det vises jo hva det er i rest» går de over i det jeg har valgt å se på som den andre situasjonen i utdraget, videre benevnt som **S2.** Den foregående utforskingen ser ut til å ha dannet et grunnlag for utforsking som guttene bygger videre på i en utveksling av påstander og begrunnelser av disse. Guttene forklarer for hverandre hva de tror foregår, bruker fagbegreper («rest», «divideres») og Harald påstar at «Det vises jo hva det er i rest». Påstanden kan tyde på at erfaringene fra de foregående forsøkene dannet grunnlag for det som nå blir en mer eksplisitt teori om at modulo-operatoren viser *hva det er i rest*. Harald argumenterer for og søker bekreftelse for hvorvidt teorien holder vann (validerer) ved å vise til eksemplene fra oppgavearket: «Men 15 delt på 4 det går ikke, da er det litt rest igjen». Geir annerkjenner forslaget ved å si «Ja!», og viser til et annet eksempel som er med på å bekrefte teorien: «og 10 delt på 3 er 1 i rest». Argumentasjonen til Harald er basert på det Brousseau (2006, s. 17) kaller intellektuelle begrunnelser da eksemplene han bruker kan sies å være allment akseptert kunnskap knyttet til divisjon og rest for elever i denne alderen: «da er det litt rest igjen». I utdraget er teorien sett på som en forklaring og eksemplene sett på som elevenes måte å avgjøre hvor gyldig eller holdbar teorien/forklaringen er. Det faktum at både en forklaring og en vurdering av forklaringen

finner sted, indikerer komplett resonnering – *reasoning of level 3*, og gjør at jeg tolker utdraget dithen at guttene trer inn i en *valideringssituasjon*.

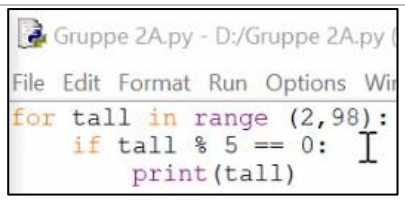
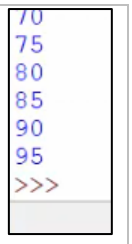
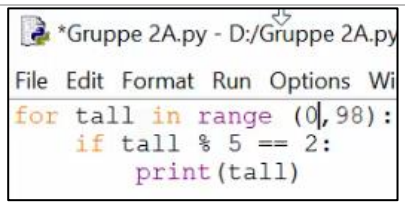
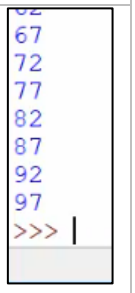
Innholdet i **S3** og **S4** kan karakteriseres som en forlengelse av *valideringssituasjonen* i **S2** da guttene nå har en eksplisitt teori for hvordan modulo virker og de ser ut til å bruke eksemplene som et middel for å bekrefte hvorvidt teorien holder vann. Når Geir foreslår et «ekstremt tilfelle» (**S3**), bruker han kunnskap om partall til å vurdere testeksempelet og bekrefte teorien: «Null! Går det?! [kikker på skjermen] … åja det er sant, det er partall!». Utsagnet kan imidlertid betraktes som ufullstendig argumentasjon siden det ikke blir forklart eksplisitt hvorfor et output `0` indikerer partall. Fra situasjonen kan man bare anta at argumentet består i et resonnement av typen *når et tall deles på to og gir 0 i rest, er det et partall* eller mer formelt; *et heltall som delt på to gir et heltall er et partall*. Når Geir til slutt resonnerer «når vi tar 3 minus 45 så får vi jo 42, som da er 6 ganger 7!» (**S4**) kan dette sees som et nytt belegg eller en forklaring på teorien om at modulo viser rest – Denne gang som et fullstendig argument. Utsagnet viser en mer komplett resonnering ved at flere sammenhenger gjøres eksplisitt; *45 er 3 mer enn et tall i 6-gangen, (42 = 6 * 7)*. Sitatet viser en logisk konsekvent resonnering med tydelige referanser til relevante elementer i omgivelsene (oppgaveteksten) og kan også sies å være *reasoning of level 3* (Brousseau & Gibel, 2005). Fra det punktet guttene først går inn i en valideringssituasjon ved at de uttrykker en forklaring på problemet (**S2**) til Geir uttrykker en komplett forklaring på den første oppgaven (**S4**), kan man se tydelig progresjon. Situasjonene kan derfor sies å inngå i en *dialektikk* hvor situasjonene bygger på hverandre.

Python-programvaren (IDLE) bestående av et skall og en tekst-editor er sentralt i situasjonen ovenfor, men det kan diskuteres hvorvidt den kan erstattet med andre digitale hjelpemidler. Elevene bes bruke modulo-operatoren i Python-skallet for å regne ut ulike rest-verdier når et tall deles med et annet, noe elevene gjør uten problemer. Hvis man ser på programmering som aktivitet, slik det beskrives innledningsvis, inneholder ikke situasjonen ovenfor noen form for programmering. Elevene simpelthen bruker et programmeringsverktøy som en kalkulator, og med utgangspunkt i SAMR-modellen kan man si at teknologien blir brukt som et direkte substitutt uten nevneverdig funksjonsforandring – Elevene kunne like godt gjennomført oppgaven og jobbet med det matematiske innholdet på samme måten ved å bruke en kalkulator. Bruken av modulo-operator i Python spiller likevel en viktig rolle i en

større kontekst da oppgaver presentert for elevene på et senere tidspunkt krever kjennskap til divisjon med modulo. I tillegg er det tydelig at tilbakemelding fra programvaren og eksemplene på oppgavearket brukes som referanser i elevenes argumentasjon og ressonering. Geir viser blant annet til et eksempel på oppgavearket da han sier «Ja, og 10 delt på 3 er 1 i rest», og til output i Python-skallet når han bekrefter teorien om modulo-operatoren: «når vi tar 3 minus 45 så får vi jo 42…».

4.1.2 Skrive ut tallrekker

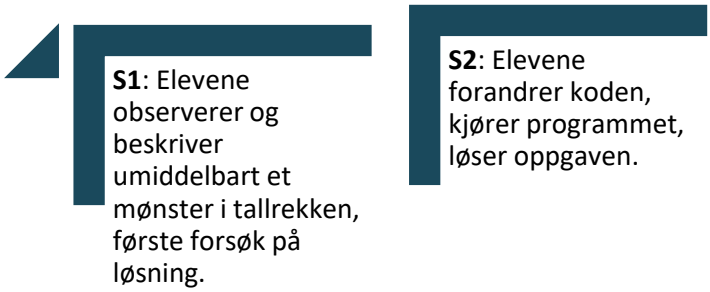
Som eneste elevpar klarer Geir og Harald å løse en oppgave knyttet til tallrekker og modulo hvor elevene bes om å lage et program som produserer en spesfikk tallrekke (kapittel 3.6.2). Elevene har i forkant fått programmet deres til å skrive ut tallrekker som 11-gangen. I denne oppgaven skal elevene fullføre en tallrekke som starter med 2 og øker med 5 for hver gang (2,7,12,17…). Det går kort tid fra elevene har lest oppgaveteksten til Geir kommer med en observasjon:

Geir: … det der er 5-gangen! **Harald**: Ja opp til… Nei det er frå 2 til 97 **Geir**: Ja men du… er lik [peker på skjermen (**Figur 1A**)] der tror jeg den er nødt til å si 2. **Harald**: [skriver inn (`2, 98`) i range-funksjonen (**Figur 1A**)] … Vi prøver det, jeg er litt usikker men vi prøver det … [Skriver inn `tall % 5 == 0` i linje 2, kjører programmet (**Figur 1B**)] **Geir**: Dette kommer aldri til å gå … **Harald**: Jo se! [peker på skjermen]	 **1A**  **1B**
Geir: Det går ikke … det kommer opp det vi ikke skal ha… det er derfor jeg tror vi kanskje må forandre den til 2 [skriver inn `tall % 5 == 2` i linje 2 (**Figur 2A**)] **Harald**: … og så tror jeg vi må ha 0 der [Forandrer range-funksjonen til `(0, 98)`] **Geir**: Hvorfor det? **Harald**: Bare prøv nå! **Geir**: [kjører programmet (**Figur 2B**)] **Harald**: Ja, se! Det virket! **Geir**: Nailed it!	 **2A**  **2B**

Geir beskriver umiddelbart mønsteret som «5-gangen» etterfulgt av Harald som korrigerer: «… nei det er fra 2 til 97». Her ser Geir et mønster som har noe med 5-gangen å gjøre, men som varierer noe. Geir følger opp med at «…den er nødt til å si 2» og Harald forandrer range-verdien til (`2, 98`). Når de kjører programmet blir ikke tallrekken som forventet, men skriver i stedet ut den «vanlige» 5-gangen (0,5,10,15…). Geir presiserer: «Det går ikke … det

kommer opp det vi ikke skal ha…», og forklarer videre at «… det er derfor jeg tror vi kanskje må forandre den til 2». Harald forandrer `If`-setningen i koden (`if tall % 5 == 2`). Dette kan tyde på at Geir allerede fra starten hadde en forståelse av tallrekken som *5-gangen pluss 2*, men at dette ikke ble uttrykt eksplisitt, og at de to guttene snakket forbi hverandre i starten av utdraget – de refererte til ulike observasjoner. Når guttene kjører programmet får de opp tallrekken og Harald bekrefter umiddelbart at oppgaven er løst «Ja, se! Det virket!»

Utdraget er i følgende delt inn i to situasjoner (**S1-2**) basert på de to løsningsforslagene elevene har:



Figur 19: Oversikt over situasjoner. Geir og Harald skriver ut tallrekker

For elevparet representerer oppgaven et matematisk problem som til en viss grad tilbyr *matematisk motstand*. Dette er synlig da Geir umiddelbart observerer og setter ord på en matematisk sammenheng relevant for oppgaven «… det der er 5-gangen!». Selv om oppgaven løses på relativt få steg, er heller ikke løsningen gitt på forhånd, og utdraget kan sees på som noe mer enn en aksjonssituasjon ved at elevene aktivt setter ord på (formulerer) observasjoner og deltar i en meningsutveksling med hverandre. Harald svarer Geir nokså umiddelbart ved å bekrefte «Ja opp til …» for deretter å utfordre påstanden med en egen observasjon «Nei det er fra 2 til 97», og Geir følger opp med «… der tror jeg den er nødt til å si 2».

Geirs første påstand om at tallrekken representerer 5-gangen kan sees på som starten på en *formuleringssituasjon* hvor guttene engasjerer seg i et matematisk problem og gradvis utvikler et språk for å løse problemet. I motsetning til en aksjonssituasjon som er mer preget

av direkte handling, prøving og feiling, viser de to første utsagnene til Geir en hypotese som blir formidlet til samtalepartner gjennom ord og valget om å kjøre programmet blir brukt som et middel for bekreftelse på om hypotesen stemmer. En slik sosial interaksjon og hvor partene formidler informasjon tydelig ment for den andre parten, illustrerer det Joubert (2017) beskriver som formuleringssituasjoners sosiale dimensjon. Dette vises også blant annet ved at Harald følger opp forslagene til Geir: «… Vi prøver det, jeg er litt usikker men vi prøver det…»

Den siste delen av utdraget, her kalt **S2**, inneholder elevenes respons på første forsøk og de justeringene elevparet (først og fremst Geir) gjør for å løse oppgaven. Det at Geir uttrykker misnøye med output («det kommer opp det vi ikke skal ha…») og ber på ny om å forandre «den til 2» viser at han allerede i S1 hadde en forståelse av hva som skulle til for å løse problemet, men at han ikke klarte å formidle eller uttrykke dette tydelig nok. Han forandrer selv koden slik at det står `tall % 5 == 2` før de kjører programmet, og umiddelbart ser at koden gav ønsket output. Da verken Geir eller Harald i det følgende utdyper, forklarer eller begrunner hvordan den spesifikke kodelinjen `tall % 5 == 2` gav ønsket resultat, er også **S2** klassifisert som en formuleringssituasjon.

Guttene forklarer ikke hvorfor løsningen (koden) gav ønsket output (tallrekken) og kunnskapen om hvorfor gitt kode gav ønsket resultat blir dermed ikke gjort eksplisitt. Elementene som skal til for at oppgaveprosessen bunner ut i *validering* uteblir, noe som kan ha konsekvenser for elevenes utvikling av kunnskap; I situasjonen er det Geir som står for de sentrale bidragene til at paret klarer å produsere tallrekken («… det er derfor jeg tror vi kanskje må forandre den til 2 [skriver inn `tall % 5 == 2` i linje 2]»). Det er derimot lite i utdraget som indikerer hvorvidt Harald forstår hvorfor programmet produserer tallrekken annet enn at han gjenkjenner output som «korrekt» («Ja, se! Det virket!»). Å anta at Harald sitter på nødvendig kunnskap for å løse problemet på egenhånd da de klikker seg videre, vil uten videre belegg være naivt. Utdraget viser at han er mer opptatt av hva som må stå i `range()`-funksjonen, enn hva betingelsen i `if`–setningen må være. I tillegg vil det for Geir være en «tapt mulighet» for å reorganisere kunnskapen sin, utvikle et bedre språk knyttet til relevante elementer i situasjonen og ikke minst dele denne kunnskapen med Harald. Situasjonen viser slikt sett utfordringer med oppgavedesignet sett i lys av et ønske om *validering*.

I motsetning til forrige utdrag hvor Geir og Harald simpelthen bruker Python-skallet som en kalkulator, er det i denne oppgaven tatt utgangspunkt i at elevene skal manipulere/forandre betingelser i en kode for å oppnå et ønsket utfall (en spesifikk tallrekke). Oppgaven krever mer direkte arbeid med programmering da elevene skriver inn kode (riktignok avskrift fra en oppgave laget av lærer), modifiserer koden med mål om at programmet skriver ut ønsket output i skallet (en tallrekke) og gjennomfører en feilsøking og forandring når koden ikke gir tilfredsstillende resultat.

Med utgangspunkt i SAMR-modellen kan man kalle bruken av Python i oppgaven som en *modifisering* av matematikkundervisning. I oppgaven taes det i bruk et verktøy som tillatter en ny måte å jobbe med tallrekker på sammenlignet med mer tradisjonell tilnærming. Innholdet i matematikken derimot forandrer ikke fokus vesentlig sammenlignet med annet arbeid med tallrekker. Arbeid med funksjoner, enten analogt eller ved hjelp av andre digitale verktøy, vil kunne stå som et reelt alternativ til denne måten å utforske divisjon og rest på. *Konteksten* kan derimot sies å være relativt nyskapende (relativt i den forstand en ikke har innsyn i hva andre lærere har gjort som ligner), og tilbyr elevene omgivelser som byr på erfaringer med programmering. Å utvikle programmeringskompetanse er ikke det primære målet med denne undervisningssituasjonen, men kan sees på som en ønsket bi-effekt. I utdraget er det først og fremst `range()`-funksjonen og betingelsen (`if`–setningen) som blir gjenstand for modifisering i koden. Hvorvidt elevene utvikler konkret kunnskap om programmering ved en slik tilnærming er det mindre grunnlag for å si noe om, men ved at de eksponeres for variabler, løkker og betingelser – viktige komponenter i programmering generelt – kan man anta at møte med slike konsepter i en senere anledning vil bli noe lettere, eller i alle fall oppleves som kjent.

4.2 Dag og Kari

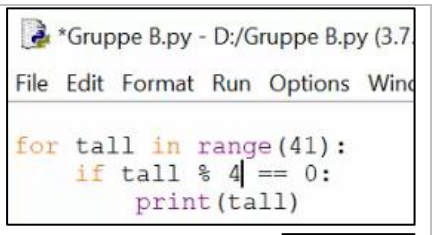
Dag og Kari representerer to elever som i matematikkfaget kan karakteriseres som «middels presterende». I datamaterialet har flere av utdragene vist seg som særlig interessante, både fordi de illustrerer motstand flere elever møtte underveis i arbeid med programmering, men også fordi Dag og Kari viste en tydelig progresjon gjennom arbeidet med et oppgavesett. I begge øktene som ble filmet (økt 9 og 10) viste en tidlig gjennomgang av datamaterialet to elever som strevde, men likevel forsøkte, ikke gav opp, og som trolig endte opp med en bedre forståelse av den relevante matematikken.

4.2.1 Skrive ut tallrekker

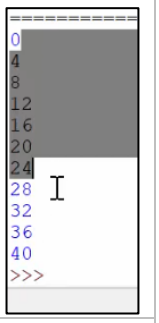
I motsetning til eksempelet med Geir og Harald (kapittel 4.1.1), hvor guttene klarer å sette ord på og beskrive hvordan modulo-operatoren fungerer, hadde Dag og Kari større utfordringer med å utvikle et godt begrepsapparat i møte med oppgaven. De møtte generelt mye motstand og de strevde med å forklare hvordan modulo-begrepet fungerte og hvorfor de ulike resultatene ble som de ble. Dag prøvde å begrunne hvordan `13 % 39` ble 13 med at «det tar ikke sånn komma-greier». Kari fikk det ikke til å stemme og spør lærer om hjelp: «Hvordan er det 13 igjen hvis vi tar 13 prosent 39 … da står det at det er 13 igjen?». Lærer forsøker å hjelpe og påpeker blant annet at rekkefølgen på tallene spiller en vesentlig rolle når de bruker modulo-operatoren som i vanlig divisjon. De forsøker et par ulike tall, blant annet `50 % 50`, hvorpå begge bare nikker og sier «ja!» når resultatet viser `0`. Det er lite i dialogen som tyder på at Kari eller Dag virkelig har forstått hvordan modulo-operatoren fungerer i det de fortsetter arbeidet. Følgende utdrag er fra arbeid med påfølgende oppgave hvor det er et mål å *bruke* modulo-operatoren for å lage ulike tallrekker. Dag skriver av koden på oppgavearket og kjører den mens Kari leser oppgaven høyt. De bruker noe tid på å få på plass alle tegnene i koden før de begynner å diskutere:

Kari: Nå kan vi ta … fordi nå kan vi skrive 41 og 4 der … [peker på koden i oppgavearket] For da blir det gangetabellen i 4.
Dag: Så det vi gjør er … at det viser gangetabellen.
Kari: Ja!
Dag: Men hvorfor 41? Det er ikke i gangetabellen?
Kari: For da blir det opp til 40! Bare se … [Forandrer koden (**Figur 1A**)]
Dag: Ja! Jeg skjønner … [Kjører programmet]
Kari: Ser du? [markerer output med musen (**Figur 1B**)]
Dag: Ja! [Tøyser:] Hvis vi hadde hatt 40, hadde det stoppet på 36, og da mangler du jo et tall i gangetabellen
Kari: Ja det er ikke bra! [ler]

Elevene forandrer deretter koden slik at 4-gangen skrives ut opp til 100 før de går videre til neste oppgave.



1A

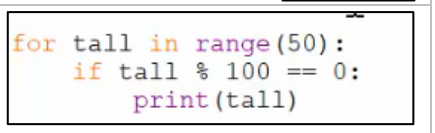


1B

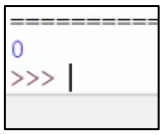
Kari leser av fra neste oppgave:

Kari: Lag et program som finner alle tall mellom 50 og 100 som er delelig med 11!
Dag: Delelig med 11? … [lengre pause]
Kari: ehm …
Dag: okey! [Skriver inn `(50)` i range-funksjonen og `tall % 100` (**Figur 2A**)]
Dag: … jeg bare sjekker!
Kari: [ler] Det er ikke riktig! … Det kommer bare til å bli sånn … 0! [Dag kjører programmet (**Figur 2B**)]
Kari: Du må skrive 11 der! [peker på skjermen]
Dag: Hvorfor det?
Kari: … fordi det skal være delelig med 11!
Dag: [Skriver inn `tall % 11` i linje 3 (**Figur 2C**), skal til å kjøre programmet, men venter] … men det står [leser fra oppgaveteksten] «lag et program som finner alle tall mellom 50 og 100 som er»
Begge: [i kor] …delelig med 11!
Dag: Ja, da må det stå 50 …
Kari: [Avbryter] Da må vi ha delelig med 11 der! Sjekk, jeg skal vise deg! [Skriver inn, Dag kjører programmet (**Figur 2D**)]

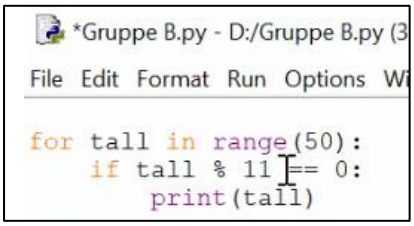
Kari: Ser du? Det er korrekt!
Dag: Nei!
Kari: Men opp til 50 så blir det bare …
Dag: [avbryter] det står
Kari: [avbryter] det blir bare 55 …
Dag: [Rynker på nesen, kikker på oppgavearket, leser med hviskestemme]
Kari: Nei, vi må skrive sånn … 50 komma 100?
Dag: Akkurat!
Kari: … men det er oppe!
[Dag forandrer verdiene i `range`-funksjonen mens Kari resiterer (**Figur 2E**)]
Kari: Nå har vi rett? [ser spørrende på Dag] … vi bare … [Dag kjører programmet (**Figur 2F**)]
Kari: Vi klarte det!



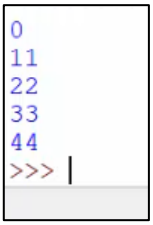
2A



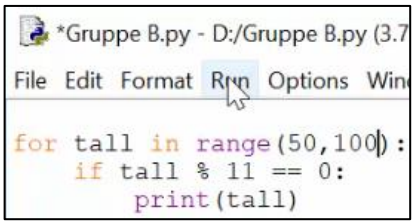
2B



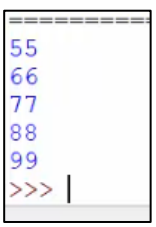
2C



2D



2E



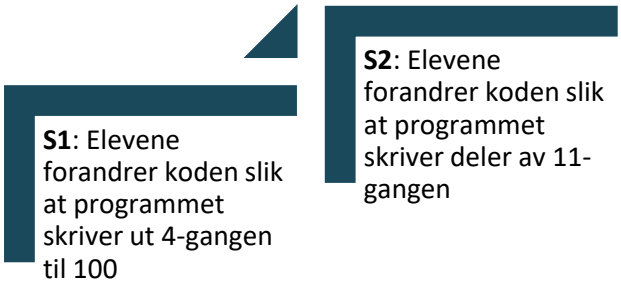
2F

Videre bes elevene forandre programmet slik at det skriver ut alle tallene i 12-gangen mellom 120 og 240, noe elevene nå gjør på første forsøk uten problemer.	

I starten av utdraget leser elevene oppgaven og Kari kommer tidlig med et forslag om å skrive inn 41 og 4 i koden og begrunner dette med: «For da blir det gangetabellen i 4». Dag spør: «Men hvorfor 41? Det er ikke i gangetabellen?». Kari forandrer koden, kjører programmet og bruker output som svar på spørsmålet til Dag: «Ser du? [markerer output med musen]». Dag gir uttrykk for at han er fornøyd med forklaringen allerede før Kari har kjørt programmet, og de går videre til neste deloppgave.

Kari fortsetter i rollen som leseansvarlig og formidler oppgaveteksten til Dag, som blir tydelig forvirret av ordlyden i oppgaven: «Delelig med 11? … [lengre pause]». Kari ser også ut til å være usikker på hvordan de skal løse oppgaven («ehm…»). Dag foreslår å forandre koden (`range(50)` og `tall % 100`): «… jeg bare sjekker!». Kari svarer umiddelbart «det er ikke riktig! … Det kommer bare til å bli sånn … 0!», noe som viser seg å stemme når Dag kjører programmet. Kari utbryter «Du må skrive 11 der! [peker på skjermen]» og følger opp med «… fordi det skal være delelig med 11!» når Dag ber om begrunnelse. Dag er ikke fornøyd med begrunnelsen («… men det står») og ser ut til å være mer opptatt av innholdet i `range()` funksjonen: «Ja, da må det stå 50 …». Kari svarer «Da må vi ha delelig med 11 der!», og forandrer på ny koden og bruker output som begrunnelse for påstanden sin: «Sjekk, jeg skal vise deg…!». Output viser tallene i 11-gangen fra 0 til 49 (`range(50)`), noe Kari på sin side hevder er korrekt uten å spesifisere hvorfor: «Ser du? Det er korrekt!» Dag sier bastant «Nei!», rynker på nesen, kikker på oppgavearket og leser videre med hviskestemme. Han viser med kroppsspråket at han er usikker.

Etter en stund foreslår Kari å skrive inn «vi må skrive sånn … 50 komma 100?» og refererer til `range()`-funksjonen i koden, noe Dag er enig i: «Akkurat!». Selv om Kari tilsynelatende er sikker på at koden fungerer «Nå har vi rett!», viser kroppsspråket tendenser til tvil, mellom annet ser hun spørrende på Dag. De kjører programmet for å få antakelsen bekreftet. Kari sier umiddelbart «Vi klarte det!» og tolker dermed output som at de har løst oppgaven.



Figur 20: Oversikt over situasjoner. Dag og Kari skriver ut tallrekker

Utdraget deles inn i to situasjoner, **S1** og **S2**, basert på de to utfordringene eller problemene elevene møter. I arbeidet med den første oppgaven (**S1**), er Kari raskt i gang med å forklare hvordan de kan gå frem og begrunner fremgangsmåten «For da blir det gangetabellen i 4». Tatt i betraktning hvor raskt Kari kommer med et løsningsforslag som fungerer, kan oppgaven for henne neppe kalles et *problem* da den i liten grad gir elevene motstand og Kari raskt beskriver en fremgangsmåte som gir ønsket resultat. For Dag derimot, representerer deler av koden, nærmere bestemt innholdet i `range()`-funksjonen, et problem og han ber Kari forklare: «Men hvorfor 41? Det er ikke i gangetabellen?». Videre er det tydelig at Dag tøyser, men han formidler likevel relevant informasjon når han sier «Hvis vi hadde hatt 40, hadde det stoppet på 36, og da mangler du jo et tall i gangetabellen». Spørsmålet Dag stiller preger situasjonen positivt ved at Kari nå får et insentiv til å forklare deler av koden og den sosiale dimensjonen blir forsterket - elevene diskuterer og utvikler et språk basert i og formålstjenelig for situasjonen. Vekselvirkningen mellom Kari sitt løsningsforslag, Dag sitt spørsmål og responsen Kari igjen gir etterpå, er preget av utveksling av informasjon knyttet til et matematisk problem, og representerer i så måte en *formuleringssituasjon*.

I resten av utdraget, **S2**, forsøker elevene å lage en kode som skriver ut tallene i 11-gangen mellom 50 og 100. Oppgaven viser seg umiddelbart som et problem med *matematisk motstand,* men som elevene aksepterer og jobber sammen med for å finne en løsning. Som med forrige oppgave, kan utdraget karakteriseres ved at elevene hyppig utveksler informasjon i tillegg til at elevene nå også gjennom hele sekvensen prøver ut og modifiserer koden basert på den umiddelbare tilbakemeldingen de får fra hverandre og programvaren. Først foreslår Dag en variant (**Figur 2A**) som Kari umiddelbart ser ikke stemmer: «Det er ikke

riktig! … Det kommer til å bli sånn … 0!» Deretter forandrer Kari koden og forsøker på nytt (**Figur 3A**), men denne gangen vurderer Dag resultatet (**Figur 3B**) som feilaktig: Kari sier «Det er korrekt!», Dag svarer «Nei!». Til slutt resiterer Kari mens Dag skriver inn koden (**Figur 4A**) og de klarer å produsere ønsket resultat (**Figur 4B**).

Denne formen for prøving og forbedring kan i følge Joubert (2017) også sees på som en *formuleringsdialektikk* da elevene bruker kunnskap og erfaringer fra en situasjon i forsøk på å løse problemet i den neste. Situasjonen preges av oppgaven, som tydelig utgjør et matematisk problem for begge elevene denne gangen, og tilbakemeldingen programvaren gir underveis i form av tallrekkene som produseres (**Figur 2B**, **3B**, **4B**). Man kan anta at tabellene som programvaren produserer gir en viss form for mening hos elevene (og derav prøving og forbedring i motsetning til vilkårlig gjetning), da responsen deres ser ut til å adressere output fra programmet på en koherent måte. For eksempel ser Kari med en gang hva output blir etter første forsøk (**Figur 2A**) og kommer med nytt forslag – en forbedring – «Du må skrive 11 der!» og begrunner dette med «fordi det skal være delelig med 11!». Etter andre forsøk (**Figur 3A**) ser også tilbakemeldingen fra output ut til å forbedre neste forsøk når elevene ser at tallrekken ikke treffer riktig «range»; Kari sier «det blir bare 55». Når elevene til slutt løser oppgaven, kan de sies å ha gjennomført til sammen tre sykluser av prøving og forbedring, hvorav siste forsøk gav tilfredsstillende resultat.

Gjennom hele andre del av utdraget (**S2**) skjer resonneringen både i form av muntlige ytringer, men også handlinger hvor deler av resonneringen forblir implisitt. Eksempelvis ser man ved tre tilfeller at Kari bruker koden og output som middel for å vise Dag at forslagene hennes løser oppgaven eller for å svare på spørsmål Dag stiller underveis. Første gang dette skjer, er når Dag spør hvorfor en forandring vil gi 4-gangen som output. Kari kjører programmet, markerer output og sier «Ser du?» Forklaringen på hvorfor man må skrive 41 i `range()`–funksjonen blir ikke gitt, men kan sees som et resultat av Kari sine handlinger når hun kjører programmet og markerer koden. På samme måte bruker Kari output til å argumentere for seg når hun skal forklare hvorfor de må skrive inn `tall % 11` i koden for å kunne skrive ut 11-gangen: «Sjekk, jeg skal vise deg…!». Den tredje gangen dette skjer, er mot slutten av utdraget (**S2**) når elevene er usikre på om koden stemmer. Ved å kjøre programmet bekrefter de for hverandre at de har løst oppgaven, og bruker således tilbakemeldingen fra programvaren som et ledd i resonneringen, selv om dette ikke utrykkes

muntlig. Svaret på spørsmålet om hvorfor tallrekken er løsning på oppgaven forblir også implisitt – elevene begrunner ikke løsningen muntlig for hverandre. Kari sine forklaringer kan derfor beskrives som delvis resonnering og argumentasjon supplert med handling, såkalt *Reasoning of level 2* (Brousseau & Gibel, 2005). Et viktig poeng i denne sammenheng er at en slik muntlig vurdering eller begrunnelse verken bes om eller spesifiseres i oppgaven.

De forklaringene som dukker opp i utdraget er ikke ansett å bidra til en *valideringssituasjon*, da forklaringene ikke er tilstrekkelige verken fra et matematisk perspektiv eller med hensyn til hvordan programmerings-komponentene virker. Eksempelvis forklarer Kari at «nå kan vi skrive 41 og 4 der … for da blir det gangetabellen i 4», men forklaringen gir ikke noe videre svar på hvorfor dette er tilfellet. I stedet er det et resonnement av typen hvis A, så B med en manglende årsaksforklaring (for eksempel i form av et «fordi»).

Oppgavene Dag og Kari jobber med i utdraget ovenfor er del av samme steg som i eksempelet med Geir og Harald (kap 4.1.2), og bruken av programmering som redskap i oppgaveløsningen fungerer derfor tilsvarende og kan ut fra SAMR-modellen sees på som *modifisering*. Utdraget illustrerer hvordan programmeringsverktøyet legger til rette for det jeg har tolket som formuleringssituasjoner, spesielt gjennom elevenes prøve- og forbedre-tilnærming til oppgavene. Oppgavenes mål om å få et program til å produsere tallrekker med et spesifikt omfang, og det mer generelle målet om å utforske delelighet og rest, ser ut til å treffe delvis. Eksempelvis trigges en kort diskusjon om gangetabellen når elevene må skrive inn i `range()`-funksjonen (**S1**). Når elevene skal løse neste oppgave (**S2**) blir koden gjenstand for diskusjon knyttet til delelighet: Kari sier «Du må skrive 11 der!», Dag spør «Hvorfor det?», og Kari svarer «Fordi det skal være delelig med 11!». Selv om den eksplisitte valideringen man ønsker uteblir, kan det sies at programmeringskomponenten har en konstruktiv rolle, og med noen justeringer kan oppgaven således gi et godt utgangspunkt for matematisk aktivitet knyttet til divisjon og rest.

Selve arbeidsformen, at elevene arbeider sammen to og to, preger også situasjonen positivt, da det gir elevene mulighet til å spørre spontane spørsmål, samtidig som det gir den andre mulighet til å svare. I dette tilfellet viser for eksempel Kari til mekanismer i programmet og setter ord på sammenhenger som respons til spørsmål fra Dag. Dag ser igjen ut til å se dra nytte av responsen når han forklarer «Hvis vi hadde hatt 40, hadde det stoppet på 36, og da mangler du jo et tall i gangetabellen».

4.2.2 Utforske tabeller med rest

Kari og Dag jobber med en oppgave gitt elevene i økt 10 (se Kapittel 3.6.3). Oppgaven er gitt i et HTML-format (nettside) med oppgavetekst og en trinket. Elevene blir i oppgaven bedt om å studere en kodesnutt, beskrive hva de tror kommer til å skje basert på innholdet i koden og tolke resultatet når programmet kjøres; *Hva tror du verdien til* `rest` *kommer til å være*? Kari begynner å lese oppgaveteksten høyt, men rekker ikke lese ferdig før Dag kommer med et forslag:

Dag: … resten sin verdi vil være ... 0, tror jeg!
Kari: Ja... Ja det vil den …
Dag: Jeg tror det blir 0, men jeg vet egentlig ikke hva som skjer. Nei, jeg tror det får verdi ... 3! [Peker til koden på skjermen]
Kari: Jeg tror det får verdi … [Dag kjører programmet (**Figur 1A**)]
Kari: Den fikk verdi 1?
Dag: … og 3!
Kari: Den fikk ikke 3 et eneste sted!
Dag: Jo, den fikk her… [Peker på kolonne i midten]
Kari: Nei den fikk... men se da, dette er et mønster; 1, 2, 0, 1, 2, 0 [peker på kolonnen til høyre]
Dag: Ja, 1 pluss 0 [holder musen over 2, høyre kolonne], det er 3. 3, 0, 3, 0 [fortsetter ned tabellen] sant?
Kari: Det gir ikke helt mening ...
Dag: Nei det gjør ikke det… [begge ler]

2	3	2
3	3	0
4	3	1
5	3	2
6	3	0
7	3	1
8	3	2
9	3	0
10	3	1
11	3	2
12	3	0
13	3	1
14	3	2
15	3	0
16	3	1
17	3	2
18	3	0
19	3	1
20	3	2

1A

De går videre til neste oppgave men arbeidet blir utfordrende, og etter en del prøving og feiling (tilsynelatende ureflektert gjetting) velger de å spørre lærer om hjelp. Lærer tilbakestiller koden, vender fokus tilbake til første oppgave og ber elevene forklare hva de så i tabellen:

Kari: Der ja, da var det liksom sånn mellom [...] først tok vi 3, da var det 0, 1, 2! ...
Lærer: Hva viser denne her nå da? [peker på hele tabellen i konsollen]
Dag: Den viser en rekkefølge mellom 0 og 3 ... eller 2!
Lærer: [Lærer markerer rad 6 i tabellen som skrives ut (**Figur 2A**)] Her da, hva forteller dette oss?
Dag: At det er noe med 3?
Kari: At det går an å gange 3 med 6? eller ... at 3 ganger 2 blir jo 6, så 6 er i 3-gangen.
Lærer: Det er jeg med på, hva viser denne da? [Lærer markerer rad 8 i tabellen (**Figur 2B**)]
Kari: At 8 er 2 i fra ... 2 over å være med i ... Altså, det er 2 over 6, så det er ikke med i gangetabellen.
Lærer: Hva viser denne da? [Lærer markerer øvereste rad i tabellen (**Figur 2C)]**
Dag: Den viser hva tall det er … og den og den [peker på kolonnene med blyant]
Kari: Åja... om det går an å gange det ... i gangetabellen
Lærer: Okei, for eksempel i runde nummer 10 [Markerer rad 10], så bruker vi 10. 10 delt på 3, hva gir det i rest?

6	3	0
7	3	1

2A

7	3	1
8	3	2

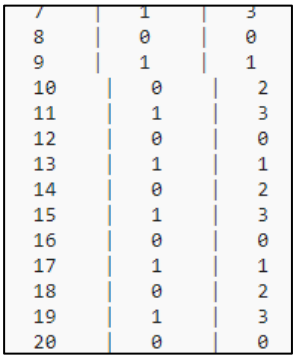
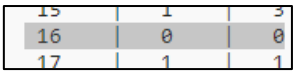
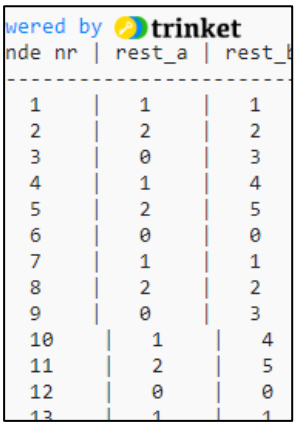
2B

unde nr.	delt på	Rest
1	3	1

2C

8	3	2
9	3	0
10	3	1

2D

Begge: [i kor] 1! **Lærer**: Nettopp! **Kari**: [markerer rad 9 (**Figur 2D**)] Og 9 delt på 3 blir 0!	
I arbeid med neste steg (oppgave 2.6) skal elevene på ny skrive ut en tabell, men denne gangen med to variabler (`a` og `b`) som utgangspunkt for en tabell som skrives ut. Kari leser oppgaveteksten og begge to blir sittende i lengre tid og studere koden. De ser på dette tidspunktet ut til å være ekstra fokuserte da de veksler mellom å lese deler av oppgaven på nytt og lese deler av koden for hverandre. Kari er den første til å komme med et forslag: **Kari**: Jeg tror at det går fra 1 til 20... og at det går 2 ganger 4. **Dag**: Jeg tror det hopper med 4 hver gang… **Kari**: jeg tror det hopper med 2 hver gang... eller 3! Jeg tror det hopper med 3! **Dag**: ja... **Kari**: Jeg tror det er det som skjer jeg. Og så går det opp til 20. [Kjører programmet (**Figur 3A**)] **Kari**: Nei... jeg skjønner nå! [lengre pause] skal vi se... Det går opp til 20, det hadde vi rett med. Åja! Det er 20 delt på de forskjellige tallene! Så hvis begge to er delelig med 4 og 2... **Dag**: … ja? **Kari**: Så blir det 0! [Ser bort på Dag] Så 20 er både delelig med 2 og 4! Det samme med ... [leter på skjermen med musen, markerer rad 16 (**Figur 3B**)] ... med 16, og 8, og 4... og 12! Det er delelig med alle de!	 **3A**  **3B**
De fortsetter med neste deloppgave og gir variablene `a` og `b` nye verdier: **Dag**: Bytt til 8 og 2! **Kari**: Jeg tror … **Dag**: Bytt til ULIKE tall… **Kari**: Åja… vi tar 6 og 3! [skriver inn a = 3 og b = 6] Nå vil det barevære de tallene mellom 1 og 20 at … som både 6 og 3 er delelig med, så vil det være 0 på siden… det VET jeg! [kjører programmet (**Figur 4A**)] **Kari**: Ser du? Jeg er ganske smart! **Kari**: [leser fra oppgaveteksten] Hvordan kan man bruke dette programmet til å finne tall som både er i a-gangen og i b-gangen? … FORDI, når man ser … det er liksom 0 på siden hvis det er delelig med begge gangetabellene!	 **4A**

Utdraget deles inn i fire situasjoner (**S1-S4**) basert på oppgavene elevene arbeider med. Etter å ha lest oppgaveteksten starter en utveksling hvor Kari og Dag diskuterer hva de tror kommer til å skje, kjører programmet og diskuterer output (**S1**). Dag gjetter at output vil representere noe som har med rest å gjøre: «resten sin verdi vil være … 0, tror jeg!», men røper like etterpå «men jeg vet egentlig ikke hva som skjer» – at han ikke har oversikt. Kari på sin side er også usikker, men støtter seg på Dag sin gjetning «Ja det vil den». Etter å ha kjørt programmet diskuterer elevene tabellen som vises i konsollen (tilsvarende Python-skallet). Umiddelbart har Kari og Dag ulikt fokus. Kari er tydelig opptatt av kolonnen til høyre

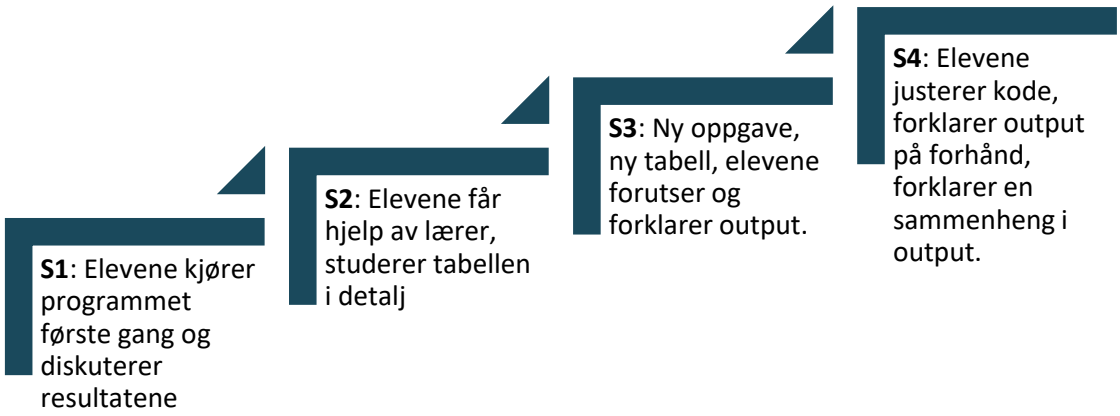
som viser tallene 0,1,2 nedover. Dag derimot, viser til midterste kolonne i tabellen og tilføyer «og 3!». Kari trekker oppmerksomheten til høyre kolonne og påpeker «men se da, dette er et mønster; 1, 2, 0, 1, 2, 0». Dag aksepterer skiftet i fokus til høyre kolonne, men holder på tallet 3 som viktig, og argumenterer for at tallrekken veksler mellom 0 og 3 nedover: «Ja, 1 pluss 0 [holder musen over 2, høyre kolonne], det er 3. 3, 0, 3, 0 [fortsetter ned tabellen]». Kari svarer «Det gir ikke helt mening ...» noe Dag sier seg enig i. Elevene klarer foreløpig ikke å forklare på hvilket grunnlag mønsteret er laget – verken med referanse til koden eller matematisk. **S1** er preget av at både Kari og dag er uvitende om hvordan programmet fungerer, men de klarer å se at tabellen representerer en regelmessighet – et mønster.

Etter å ha mislykkes med den neste oppgaven får elevene hjelp av lærer som vender fokus tilbake til første oppgaven og tabellen (**S2**): Lærer ber elevene først beskrive observasjonene som ble gjort tidligere (**S1**), og Kari svarer «først tok vi 3, da var det 0, 1, 2!...». I en serie spørsmål vender lærer oppmersomheten mot tabellen og ber elevene forklare hva de ser: «Hva viser denne her nå da?», «hva forteller dette oss?», «Hva viser denne da?». Først henviser læreren til tabellen generelt (**Figur 1A**), deretter høyre kolonne, rad 6 i tabellen, rad 8 i tabellen og til slutt øverste rad i tabellen (**Figur 2C**). Underveis gir Kari gode forklaringer på innholdet i tabellen. Når rad 6 er markert sier hun «at 3 ganger 2 blir jo 6, så 6 er i 3-gangen» og når rad 8 er markert sier hun «Altså, det er 2 over 6, så det er ikke med i gangetabellen». Uten å eksplisitt referere til koden, forklarer Kari nå deler av matematikken som ligger til grunn for hvordan tabellen er laget og viser god forståelse for det matematiske innholdet sammenlignet med i dialogen i det tidligere arbeidet (**S1**). Når lærer markerer øverste rad av tabellen, får Kari innholdet i tabellen til å gi mening; «Åja… om det går an å gange det…!». Etter at lærer har vist et nytt eksempel «for eksempel runde nummer 10», markerer Kari raskt en ny rad i tabellen og forklarer «Og 9 delt på 3 blir 0!».

Videre går elevene i gang med en ny oppgave hvor de på lignende vis skal tolke en kode og studere output (**S3**). Nå baseres tabellen som skrives ut blant annet på variablene `a = 2` og `b = 4`. Kari foreslår «Jeg tror at det går fra 1 til 20... og at det går 2 ganger 4». Dag følger opp; «Jeg tror det hopper med 4 hver gang…». Kari justerer seg noe, og imiterer Dag sin bruk av ordet «hopper»; «jeg tror det hopper med 2 hver gang... eller 3! Jeg tror det hopper med 3!». Denne gangen oppsummerer Kari output på følgende måte når de kjører programmet: «Det er 20 delt på de forskjellige tallene! Så hvis begge to er delelig med 4 og 2 […] Så blir

det 0!». Hun endrer ordlyden noe: «Så 20 er både delelig med 2 og 4! Det samme med [...] med 16, og 8, og 4... og 12! Det er delelig med alle de!». I siste setning ser Kari at forklaringen, *delt på de forskjellige tallene*, også gjelder for andre tall enn 20.

(**S4**) Etter elevene har forandret variablene til `a = 3` og `b = 6`, uttrykker Kari selvsikkert: «Nå vil det bare være de tallene mellom 1 og 20 at … som både 6 og 3 er delelig med, så vil det være 0 på siden». Kari oppsummerer på nytt denne sammenhengen når hun svarer på et av spørsmålene i oppgavesettet: «… det er liksom 0 på siden hvis det er delelig med begge gangetabellene!». Bortsett fra å komme med forslag til verdier i variablene, noe Kari nærmest ignorerer, forholder Dag seg taus resten av situasjonen.



Figur 21: Oversikt over situasjoner. Dag og Kari utforsker tabeller

Dag og Kari sitt arbeid med den første oppgaven (**S1**) er i likhet med forrige utdrag preget av at elevene snakker sammen og utveksler informasjon i tilknytning til oppgaven. De starter med relativt vilkårlig gjetning (Dag: «resten sin verdi vil være…») og fortsetter med å sette ord på observasjoner etter å ha kjørt programmet (Kari: «Den fikk verdi 1?»). Denne vekselvirkningen mellom elevenes forslag og påfølgende observasjoner er typisk en *formuleringssituasjon*. De begynner å diskutere mulige løsninger på det antatte problemet *hva vises i rest* når begge to ser ut til å anta at «resten sin verdi vil være» et spesifikt tall (entall). I oppgaveteksten presiseres det «hvilke verdier» (flertall), noe som implisierer flere verdier. Elevene diskuterer likevel mulige utfall og kan sies å være involvert i prossessen med å finne en løsning. Som typisk for en formuleringssituasjon (Brousseau, 2006, s. 12) har elevene begynt å utvikle et språk spesifikt for situasjonen som inkluderer en bevissthet rundt

sammenhenger og egenskaper ved problemet – her at de skal finne rest. Denne utviklingen av språk skjer gjennom observasjoner som gjøres eksplisitt, for eksempel når Kari like etter å ha kjørt programmet (**S1**) sier: «Den fikk verdi 1!» og Dag like etter sier: «og 3». Når Kari fortsetter med å beskrive det hun oppfatter som et mønster, er dette i tillegg til å utvikle situasjonsspesifikt språk, et resonnement eller et fremlegg av en hypotese/teori hun ser som relevant for oppgaven.

Denne vekselvirkningen fortsetter i neste situasjon (**S2**), bare at nå har elevene bedt lærer om hjelp og lærer fungerer som en samtalepartner ved å stille elevene spørsmål knyttet til tabellen. Lærers intervensjon fører elevene fra den a-didaktiske, ut i den bredere didaktiske sammenhengen. Man kan ifølge Brousseau (2006) anta at elevene dermed i større grad blir bevisst situasjonen som en «kunstig» situasjon fasilitert av lærer – i motsetning til i mer genuin problemløsning hvor elevene i øyeblikket godtar situasjonen som «reell». Lærers bruk av output (tabellen) som referanse i en rekke spørsmål ser likevel ut til å hjelpe elevene med å se sammenhengen mellom de ulike kolonnene. I slutten av **S2** bryter Kari inn, markerer en rad i tabellen og sier «Og 9 delt på 3 blir 0». Hun setter ord på en relevant observasjon og viser en forståelse for hva de ulike tallene i tabellen representerer.

I arbeidet med neste oppgave (**S3**) fortsetter mønsteret med gjetninger og observasjoner, men det utarter seg noe annerledes; I større grad enn tidligere, antakeligvis basert på erfaring fra **S1** og **S2**, har Dag og Kari andre teorier/formodninger/gjetninger om hva som kommer til å skje når de kjører programmet. Eksempelvis ser Dag at det nå «hopper med 4 hver gang» sammenlignet med tidligere påstand (*jeg tror det blir X).* De ser også ut til å i større grad bruke elementer i koden, noe man ser på tallene elevene bruker i gjetningene da disse er de samme som i koden. Kari innleder med å si «Jeg tror det går fra 1 til 20», noe som indikerer at hun leser av `range`-funksjonen, og «at det går 2 ganger 4», som indikerer at hun leser av startverdiene til variablene `a` og `b` i koden. Elevene klarer nå å benytte seg av elementer i koden for å gjøre en mer informert antakelse.

Progresjonen fra mer eller mindre uinformerte antakelser til mer korrekte, gjør at situasjonene har et *dialektisk* preg. Oppgavene elevene arbeider med bærer mange likheter og måten elevene responderer på oppgavene viser denne progresjonen. I starten (**S1)** gjetter både Kari og Dag at programmet skriver ut en enkeltverdi. De har her ikke lest/studert selve koden og ser ut til å basere gjettingen på enkeltverdier som dukker opp (f.eks: `a = 3`). I

arbeidet med neste oppgave (**S3**) er det fremdeles gjetting som råder når elevene møter oppgaven, men denne gangen ser gjettingen ut til å være mer informert. Kari bruker nå elementer fra koden når hun sier «at det går fra 1 til 20 … og at det går 2 ganger 4», en mer presis formulering sammenlignet med **S1** hvor hun gjettet et enkeltsiffer og ikke fikk tabellen til å gi mening. Dag sier nå at det «hopper med 4 hver gang» sammenlignet med **S1** hvor gjetningen var et enkeltsiffer etterfulgt av «men jeg vet egentlig ikke hva som skjer!». På bakgrunn av en slik sammenligning kan det sies å være en progresjon fra en situasjon til den neste, og det vil være rimelig å anta at erfaringen fra **S1** og fra lærereintervensjonen i **S2** spiller en rolle i denne utviklingen, og således påvirker elevenes strategier i møte med oppgavene. I motsetning til aksjonssituasjoner, hvor slike strategier forblir implisitte, er både Dag og Kari muntlig aktive og formidler hva de tror skjer underveis. De tre første situasjonene utgjør dermed *formuleringsdialektikk*.

Ettersom Kari etter hvert blir bevisst sammenhengen mellom tallene i tabellen og sier «Åja det er 20 delt på de forskjellige tallene…», er det ikke lenger snakk om gjetting. Kari er nå rimelig sikker på hva output viser, hun forklarer at «de tallene mellom 1 og 20 at … som både 6 og 3 er delelig med, så vil det være 0 på siden… det VET jeg!». Denne beskrivelsen er på flere vis nokså ufullstendig, men likevel korrekt, og Kari uttrykker en forståelse for matematikken og programmet, som tydelig er en forbedring/progresjon sammenlignet det man kunne observere i starten av utdraget. Forklaringen til Kari kan også sees på som en løsning til det antatte problemet *hva viser tabellen*, og man kan nå diskutere hvorvidt elevene trer inn i en *valideringssituasjon*. Som Brousseau (2006) påpeker vil elevers teorier, argumenter og bevis i en valideringssituasjon ofte være feilaktige og mangelfulle, men de må forstås som del av konteksten. Her vil progresjonen fra utsagn som «det gir ikke helt mening» til «Det er liksom 0 på siden hvis det er delelig med begge gangetabellene» tyde på at elevene (i alle fall Kari) har bygd opp en viss kunnskap relevant for situasjonen. Spørsmålet kan derimot stilles hvorvidt Dag oppfatter resonnementet til Kari som en forklaring, og om det forekommer en vurdering av hvorvidt denne forklaringen gir mening. Joubert (2017) påpeker at samtalepartner spiller en viktig rolle i valideringssituasjoner ved å gi tilbakemelding på utsagnet, enten gjennom bekreftelse, spørsmål eller motstand. Ettersom Dag forblir taus og verken han eller Kari begrunner hvorfor denne påstanden eventuelt er korrekt, må det vurderes hvorvidt denne forklaringen kan sidestilles med en

«observasjon» eller «påstand», fremfor mer komplett resonnering av typen *reasoning of level 3*. I det Kari til slutt sier «det er liksom 0 på sidene hvis det er delelig med begge gangetabellene», fungerer dette som en (korrekt) påstand, blir heller ikke den videre begrunnet eller forklart. Også den siste situasjonen i utdraget (**S4**) blir dermed definert som formuleringssituasjon.

Selve oppgaven, som innebærer å vurdere verdien(e) til en variabel i koden, ser ut til å spille en rolle for elevenes formuleringer da observasjonene knytter seg til hva «rest» betyr. Problemet for situasjonen som helhet ser derimot ut til å være knyttet til *hva innholdet i tabellen faktisk betyr*? Dette kommer til uttrykk når Kari påpeker at hun ser et mønster («1,2,0,1,2,0»), men innrømmer at hun ikke får det til å gi mening. Tatt i betraktning at lærer blir nødt til å lede oppmerksomheten tilbake til tabellen (**S2**), kan man se på dette problemet som utgangspunktet for motstanden elevene møter videre i utdraget. Motstanden kan sies å være av matematisk karakter da problemet innebærer å tolke et mønster. At elevene ser på dette som et matematisk problem er rimelig å anta da innholdet i samtalen gjenspeiler den relevante matematikken («gangetabell», «delelig med», «rest»).

I lys av spørsmålet hvilken rolle programmeringsverktøyet spiller i oppgaven, ser man at elevene i stor grad refererer til aspekter ved programmet men de underliggende konseptene får mindre betydning for elevenes argumentasjon og resonnement. Verken Dag eller Kari ser ut til å bry seg om hvilke mekanismer som er utgangspunkt for tabellen som skrives ut. Det ser ut til være tatt for gitt at utfallet er en tabell og at innholdet gjenspeiler variablene, men blir ikke påpekt hvordan dette skjer. Forståelsen for programmet (selve koden) ser for Dag og Kari dermed ut til å være verken nødvendig eller relevant for å løse oppgaven, og med en viss ambivalens kan man si at situasjonene er preget av at programmering *kun* fungerer som et redskap. Tabellene som for elevene er blitt gjenstand for undersøkelse har helt klart lagt til rette for matematisk undersøkende virksomhet, men det er ikke gitt at valg av Python (trinket) i oppgavedesignet er avgjørende for at denne virksomheten finner sted. Totalt sett er arbeidet i dette utdraget knyttet opp til tre ulike tabeller, og arbeidet med å produsere tabellene har bestått i å velge verdier på variabler og kjøre et program.

4.3 Ask og Brage

Sammenlignet med de to andre elevparene har Ask og Brage en ganske annen tilnærming til arbeidet, noe som vises spesielt godt i de to utdragene nedenfor. Ask og Brage kan begge karakteriseres som høytpresterende og særdeles aktive i matematikkfaget, men gjennomgang av videomaterialet viser to nokså rastløse gutter med mindre behov for å forklare eller utbrodere. Det kan nevnes at begge elevene underveis i prosjektet har vist stor interesse for å programmere, men helst på egne premisser.

4.3.1 Utforske modulo-operator

I arbeidet med oppgaven (se kapittel 3.6.1) starter elevene uten særlig betenkningstid. Brage leser oppgaven høyt, skriver umiddelbart `15 % 5` i et tomt kodevindu, men ingenting skjer:

Brage: *Nei, nei… jeg vet [skriver 15 % 7 i skallet, trykker enter (**Figur 1A**)]*
Brage: *[slår i bordet] Boom! [«feirer» med å vise tegn med fingrene. Ask viser tommel opp]*
	1A
	`>>> 15 % 7`
	`1`

Brage: *[leser videre fra oppgaveteksten] Hva får du i rest hvis du deler 45 på 6? Så da må vi bruke den da [peker på skjermen]*
Ask: *da må vi …*
Brage: *[strekker seg over tastaturet og skriver inn 45 % 6 i skallet] 45 … sånn?*
Ask: *Vi må først tenke! … 3! Vi får 3! [trykker enter (**Figur 2A**)]*
Brage: *jepp!*
	2A
	`>>> 45 % 6`
	`3`

Brage: [leser fra oppgaven, mumler deler av teksten] Bruk … tall som er delelig med 13… 39… prøv 39 og 13… eller 113 og 13… [Ask skriver inn 39 % 13, trykker enter *(**Figur 3A**)*]
Brage: Ja, det er delelig! Prøv 113, 13!
Ask: Det går jo ikke!
Brage: Jeg vet, men prøv så ser vi hvor mye som er i rest … [Ask skriver inn 113 % 13, trykker enter *(**Figur 3B**)*]
	3A
	`>>> 39 % 13`
	`0`
	3B
	`>>> 113 % 13`
	`9`

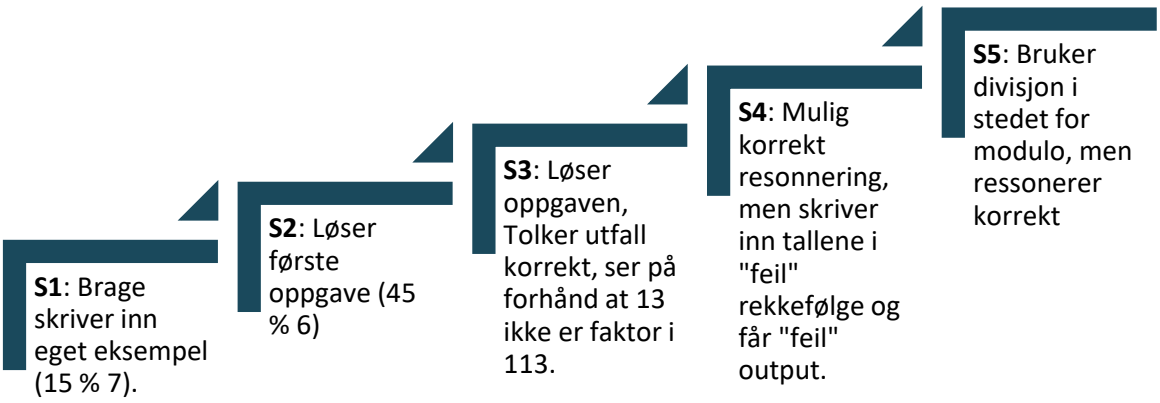
Brage: … Og så … [leser fra arket] 41 og 533
Ask: Da blir det vell 0 i rest, siden det er mer … [Skriver inn 41 % 533, trykker enter *(**Figur 4A**)*]
Brage: 41!
Ask: Ups!
	4A
	`>>> 41 % 533`
	`41`

Brage: [leser fra oppgaven] Er 13113 delelig med 13? prøv [sier sifrene en og en] 13113 og 13
Ask: … Det går ikke!
Brage: Jo!
Ask: nei!
Brage: nei, det gjør ikke det…
Ask: Nei det går ikke, det blir 9!
Brage: Det blir masse kommaer! … [Ask trykker enter *(**Figur 1A**)*]
Brage: Så nei det er ikke delelig med 13! eller jo… med komma…
	`>>> 13113/13`
	`1008.6923076923077`
	5A

Etter å ha skjønt hvor «skallet» er, er guttene raskt i gang med å prøve ut et eget eksempel (`15 % 7`). Brage gir med en gang uttrykk for at han får det til «Boom!» men sier ikke noe mer om *hva* som virket. Brage fortsetter med å lese oppgaven, skriver inn `45 % 6` og ber

Ask om bekreftelse før han kjører programmet: «sånn?». Ask tenker seg om og forutser at output vil vise 3, noe de får bekreftet like etterpå. I den neste oppgaven endrer spørsmålet form (*finn et tall som er delelig med*) og elevene tilpasser seg deretter. Brage leser fra oppgaven «tall som er delelig med 13…» og ber umiddelbart Ask prøve «39 og 13». Det at Brage umiddelbart endrer rekkefølge på tallene kan tenkes å være et bevisst valg da begge guttene har god forståelse for det matematiske innholdet oppgavene sikter mot. Ask konkluderer for eksempel like etterpå med at 113 ikke er delelig med 13 «Det går jo ikke!». Brage sier seg enig, men ber Ask om å prøve «så ser vi hvor mye som er i rest…». Når elevene skal finne ut om 41 er en faktor i 533 klarer Ask raskt å regne seg frem til eller anta at output viser 0 i rest, noe som faktisk er tilfellet (`533 % 41 = 0`), men denne gangen skriver Ask inn tallene i den rekkefølgen Brage leser opp og output viser `41` (i rest). Output viser her hvor mye man får i rest om 41 deles på 533. I stedet for å reflektere over svaret eller sjekke hvorvidt dette stemmer, godtar Ask umiddelbart output som fasit: «Ups!». Til slutt diskuterer guttene hvorvidt 13113 er delelig med 13. Etter en kort utveksling er guttene enige om at dette ikke er tilfellet; Ask hevder «det blir 9!» (i rest, min tolkning), og Brage legger til at «Det blir masse kommaer!». Denne gangen skriver de inn vanlig divisjon (13113 / 13) og med (antatt) referanse i output (1008.69230…) konkluderer Brage raskt: «Så nei det er ikke delelig med 13! Eller jo… med komma …».

Basert på oppgavene de arbeider med underveis kan utdraget deles inn i fem situasjoner (**S1-5**):



Figur 22: Oversikt over situasjoner. Ask og Brage utforsker modulo-operator

Elevenes handlinger kan i situasjonen ovenfor beskrives ved at elevene raskt identifiserer hva de må gjøre og handler deretter. Tiden det går fra Brage leser oppgaveteksten, Ask kommer med et forslag («vi får 3!»), til de får skrevet inn et korrekt utrykk i skallet, er kort. Denne vekslingen mellom å lese oppgaveteksten og umiddelbart komme med et korrekt løsningsforslag er også tydelig, spesielt i **S2**, **S3** og **S5**.

Det at elevene gjennom arbeidet med de ulike oppgavene snakker sammen og utveksler informasjon, kan tale for å klassifisere deler av utdraget som formuleringssituasjon. For eksempel når Brage ber Ask om å prøve 133 og 13 (*er 113 delelig med 13?*), resonnerer Ask raskt «Det går jo ikke». Når Brage deretter sier «Jeg vet det, men prøv så ser vi hvor mye som er i rest», så ligger det implisitt en forståelse av både modulo-begrepet og divisjon i utsagnet; dersom det ikke går (blir 0) vises mengden rest. Selv om dette kan betraktes som formuleringer i den forstand at de setter ord på relevante matematiske sammenhenger og bygger opp et språk rundt situasjonen, mangler oppgavene motstand slik at formuleringene er betinget et problem. **S5** kan derimot sees som en formuleringssituasjon da elevene i større grad møter motstand samtidig som de i diskusjonen påpeker relevante sammenhenger, som når Brage sier «Det blir masse kommaer!»

Det er ifølge Joubert (2017) først når elever møter motstand og engasjerer seg matematisk i en utfordring/et problem at elevene går inn i en *formulerings*- og *valideringssituasjoner*. Avhengig av elevenes forkunnskaper og tidligere erfaringer, vil det variare i hvilken grad elevene opplever en oppgave som et problem eller en utfordring (Joubert, 2017). Fra dette perspektivet kan man lese situasjonen med Ask og Brage dithen at oppgavene ikke byr på nok motstand til at elevene trenger å utvide kunnskapen sin – de trenger ikke gå inn i en formuleringssituasjon for å sette ord på det som skjer, da de fra starten ser ut til å vite hva som skjer og utsagnene simpelthen bekrefter det de allerede vet. Mulige unntak er da de ikke løser oppgaven korrekt (**S4**), ved at de blander om rekkefølgen tallene som skrives i modulo-funksjonen (Ask skriver inn 41 % 533). I situasjonen ser verken Ask eller Brage ut til å være oppmerksom på at dette er et problem – de behandler det ikke som det. Ask ser ut til å godta output og guttene går videre til neste oppgave. I tillegg setter Brage ord på en sammenheng i **S5** i det han forklarer hva utfallet blir.

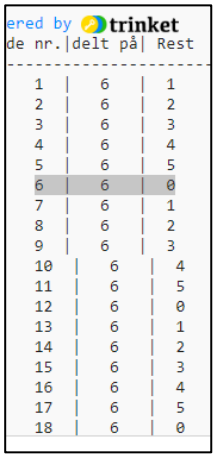
Oppsummert kan man karakterisere sekvensen ovenfor ved at manglende *matematisk motstand* gjør arbeidet rutinepreget. Sekvensen er dermed tolket som i hovedsak preget av

aksjonsdialektikk. Ask og Brage raskt tilpasser seg situasjonen og finner stort sett korrekte løsninger på oppgavene, uten å tilsynelatende møte *motstand*. Elementer i programvaren, som modulo-operatoren, blir kun gjenstand for observasjon innledningsvis når Brage sier «Så da må vi bruke den da [peker på skjermen]». Dialektikken kommer til syne ettersom elevene tydelig har implisitte strategier i møte med oppgavene, og disse påvirker handlingsløpet. Dette vises for eksempel ved at Ask tilsynelatende ser ut til å bruke kunnskap fra **S3** (`113 % 13 = 9`) når han senere (**S5)** påstår at 13113 ikke er delelig med 13, da «det blir 9!».

4.3.2 Utforske tabeller med rest

I økt 10 arbeider Ask og Brage med oppgaven der de skal forutse ulike verdier en variabel får når et program kjøres (se kapittel 3.6.3). Brage begynner med å lese oppgaveteksten, sier «`a` er lik `3`, hva skal vi gjøre?» og viser at han er noe i tvil om hvordan de skal gå frem. Han begynner nesten umiddelbart å lese fra neste oppgave. Ask forandrer a-verdien i koden til 6, og forklarer hva han tror kommer til å skje:

Ask:	Nå er det jo det samme, bare dobbelt så mye, fordi 6 er dobbelt så mye …
Brage:	6, ja, så skriver den ut … det der [peker på koden] rundenummer … delt på… rest, så kommer det sånn stiplet linje under. Og rundene går fra 1 til 20… ja? [nikker] Og resten er lik … runden og så sånn … rest … med sånn a, som er 6, og så skriver den ut … ja! [nikker] [kjører programmet (**Figur 1A**)]
Brage:	[Leser av i tabellen] 1, 2, 3, 4, 5, 0 … [fortsetter] Det er hvor mye det er i rest da!
Ask:	Vent litt… hvorfor blir det bare 1, 2, 3, 4, 5, 0 … ?
Brage:	… Fordi det er så mye det er i rest!
Ask:	Men det går vel ikke an å dele 1 på 6? [Ironisk? Peker på skjermen]
Brage:	Runde delt … 1 … Ja, hva er det som ikke går da?
Ask:	… [svarer ikke]
Brage:	Men hvis det går til et tall som går… [markerer linje 6 i tabellen, 6 delt på 6 som gir 0 i rest] Ja, null i rest! Se her da, og så blir det 1 i rest, 2 i rest [markerer videre i tabellen med musen] Det gir jo mening!



1A

Nesten parallelt med at Brage begynner å lese den andre oppgaven, skriver Ask inn en ny `a`-verdi i koden (`a = 6`) og trekker slutningen «Nå er det jo det samme, bare dobbelt så mye, fordi 6 er dobbelt så mye». Ettersom den første koden viste at `a = 3`, kan man tolke det som at Ask mener verdiene i tabellen nå vil være dobbelt så høye som i utgangspunktet. Brage følger opp med å forklare steg for steg hva han tror skjer ved å følge koden. Han leser koden og sier noen av ordene høyt. Ved å lese koden linje for linje viser Brage at han har god forståelse for hvordan koden fungerer: «… så kommer det sånn stiplet linje under. Og

rundene går fra 1 til 20…». Til slutt kjører de programmet. Brage leser av et par rader i tabellen og konkluderer «Det er hvor mye det er i rest da!».

Ask stopper opp og spør «hvorfor blir det bare 1, 2, 3, 4, 5, 0 … ?». Brage gjentar det han nettopp sa: «Fordi det er så mye det er i rest!» men Ask følger opp med «Men det går vel ikke an å dele 1 på 6?». At Ask her tydelig er ironisk kan tolkes i flere retninger. Det er mulig Ask ikke hørte etter når Brage forklarte og var mer opptatt av at `a`-verdien nå var 6, uten å ha reflektert over hva det fører til. Video-opptakene viser derimot at Ask tøyser mye i forkant, er ufokusert og mest sannsynlig kun får med seg deler av det Brage sier i utdraget. Han virker å være mindre interessert i en faktisk løsning. Videre stiller Brage spørsmålet tilbake: «Hva er det som ikke går da?». Ask svarer ikke og Brage følger opp med en ny forklaring hvor han markerer rad 6 og sier «Ja, null i rest! Se her da, og så blir det 1 i rest, 2 i rest…». Brage viser igjen god forståelse ettersom han i sted hadde tydelig fokus på koden og nå klarer nokså presist å forklare output som et resultat av koden «så blir det 1 i rest, 2 i rest».

Brage ser ut til å være den som er motivert for å løse oppgaven; Han går ikke mer i dybden på de underliggende mekanismene som produserer tabellen. Han forklarer ikke hvorfor og hvordan output er som det er. Situasjonen bærer likevel preg av et sosialt aspekt som involverer utvikling av språk knyttet til det matematiske innholdet i situasjonen; Brage leser/forklarer koden, Ask stiller spørsmålet «Hvorfor blir det bare 1, 2, 3, 4, 5, 0…?», noe som er tolket dithen at tabellen i seg selv tilbyr noe matematisk motstand, i alle fall for Ask. Brage responderer på spørsmålet, forsøker på nytt å forklare ved å gjenta «det er så mye det er i rest!». For å være komplett burde en slik forklaring være begrunnet med elementer i koden som modulo-tegnet, rundenummer og matematiske begreper som divisjon (delt på). I tillegg, for å kunne kalle utsagnet validering, måtte eleven på en logisk konsekvent måte knytte sammen disse begrepene med output, eksempelvis: *Rad 6 i tabellen representerer den sjette runden i løkken i programmet. I denne runden (representert som en rad i tabellen) regner programmet ut* `6 % 6` *da variablen* `runde` *og variabelen* `a`*, nå begge tilsvarer verdien* `6`*. Høyre kolonne viser resultatet av utregningen, det som er i rest;* `0`.

Alternativt kan man også begrunne det med at 6 delt på 6 er lik 1 hel. Med elevenes kunnskapsnivå og tidligere presteasjoner i mente, er ikke en slik forklaring utenfor rekkevidde. Argumentasjonen til Brage sammenfaller med det Brousseau og Gibel (2005)

kaller *reasoning of level 2* som igjen skjer i formuleringssituasjoner; Argumentasjonen er ikke komplett, selv på et nivå man kunne forvente av elever på dette nivået (ref. eksempelet ovenfor), men handlingene kombinert med visse utsagn viser en form for resonnement «så blir det 1 i rest, 2 i rest [markerer videre i tabellen med musen]». Her ser man også at output (tabellen) blir brukt som et middel for å illustrere løsningen på spørsmålet Ask stilte; *hvorfor blir mønsteret slik det blir?*

I dette utdraget ser man også mer tydelig at programvaren er medvirkende for hvordan situasjonen utspiller seg. Brage leser koden direkte, og man kan antyde at han får den til å gi mening ved at han nikker og sier «ja» samtidig som han leser. Brage viser også forståelse når han leser av koden og eksempelvis sier «og **så** skriver den ut». Brage viser til programflyten hvor Python-skallet tolker koden ovenfra og ned, først linje 1, deretter neste linje og så videre. Han presiserer også at «`a`, som er 6», noe som viser at han forstår hva variablene i koden refererer til, i alle fall før koden kjøres (og variabelen blir gitt nye verdier). Dette kan være medvirkende til at tabellen gir mening for Brage og at han umiddelbart kan forklare Ask at tabellen viser det som er i rest. I lys av SAMR-modellen viser dette utdraget mer tydelig at oppgaven byr på muligheter som er unike for en programmeringskontekst, og derav kan sees som *redefinering*. Koden tilbyr elevene et mønster, et objekt som kan studeres, og som har en klar logisk oppbygging. Koden får også en konsekvens når den kjøres i form av output og elevene kan når som helst velge å se resultatet.

5 Diskusjon

I analysen skildres forsøk på å bruke programmering som utgangspunkt for matematisk utforskende aktivitet knyttet til divisjon og rest. De to første forskningsspørsmålene omhandler hva som kjennetegner elevenes handlinger i interaksjon med oppgave og programvare og hvordan oppgavedesignet sammen med verktøyet påvirker elevenes argumentasjon og resonnering. For å besvare disse spørsmålene er det først og fremst tatt utgangspunkt i TDS (Brousseau, 2006). De ulike didaktiske situasjonene samt begrepet motstand er brukt som analyseverktøy for å kunne si noe om elevenes handlinger i møte med oppgave og programvare. Videre er SAMR-modellen (Puentedura, 2013) brukt som et perspektiv for å kunne belyse hvilken didaktisk funksjon programvaren spiller i de valgte situasjonene.

5.1 Elevenes handlinger, argumentasjon og resonnering

I flere tilfeller kan elevenes handlinger sees på som et resultat av en umiddelbar trang til å prøve ut programmene de møter i oppgaven, men handlingene kan også sees på som en følge av eksplisitt resonnering eller argumentasjon. Eksempelvis er Geir og Harald sin teori om at modulo-operator viser *det som er i rest* utgangspunkt for videre utforsking med «ekstreme» tall for å sjekke om teorien holder vann. Elevenes mer direkte handlinger fører også til matematisk diskusjon (som inneholder resonnering eller elementer av argumentasjon). For eksempel diskuterer Dag og Kari hvorvidt en tallrekke (som er resultat av å skrive inn kode og kjøre et program) er svar på oppgaven om å skrive gitte deler av 11-gangen. Ettersom handlinger rettet mot oppgave og programvare tett kobles opp mot argumentasjon og resonnering, er det sett på som hensiktsmessig å se begge disse forskningsspørsmålene i sammenheng når resultatene fra analysen diskuteres videre.

I lys av TDS kan man oppsummere majoriteten av utdragene beskrevet i analysen som preget av situasjoner hvor elevene først og fremst *utveksler informasjon* knyttet til oppgaven og programvaren uten å mer inngående forklaringer og vurderinger. Disse situasjonene er derfor klassifisert som formuleringssituasjoner. Et av utdragene, Ask og Brage i arbeid med modulo-operator, er beskrevet i hovedsak som aksjonsdialektikk, med begrunnelse i at elevene tilsynelatende møter mindre matematisk motstand. Når Geir og Harald utforsker

samme oppgave (modulo-operator), skiller det seg imidlertid ut som i hovedsak preget av valideringssituasjoner med komplette resonnement og gode begrunnelser.

Det er tydelig at de samme oppgavene treffer elevene ulikt (både matematikkfaglig og programmeringsteknisk), noe jeg kommer tilbake til nedenfor. Sistnevnte utdrag (valideringssituasjon) eksemplifiserer hvordan elevene bruker programvaren som redskap for å resonnere og argumentere seg frem til en forståelse av det de oppfatter som problemet. Ettersom valideringssituasjoner og valideringsdialektikk ifølge Brousseau (2006) er den formen som i størst grad representer genuin matematisk aktivitet, er det grunn til å spørre seg hvordan man skal forholde seg til de resterende tilfellene. Hvilken verdi har de resterende situasjonene som matematisk aktivitet og i hvilken grad er manglende validering en utfordring man kan gjøre tilpasninger for å imøtekomme?

5.1.1.1 Manglende insentiv for validering?

I litteraturen som er gjennomgått pekes det på lærers sentrale rolle i å gjøre matematikken eksplisitt (Forsström & Kaufmann, 2018; Misfeldt & Ejsing-Duun, 2015). Dette kan sies å være analogt med innholdet i *validering* som begrep da validering krever eksplisitt bruk av matematisk kunnskap. At elevene sjelden trer inn i valideringssituasjoner på egenhånd illustrerer dermed en utfordring. I utdragene er det utfordrende for elevene å nettopp påpeke sammenhenger og begrunne resultatene de får ved å kjøre koden, selv om oppgavene er laget på en måte som forsøker å legge til rette for dette.

Joubert (2017) mener at man kan ha som en grunnleggende antakelse i elevers arbeid med oppgaver at de vil finne den enklest mulige måten å løse en oppgave på, og det er på ingen måte nødvendigvis den måten som fører til mest læring, eller til kunnskap lærer har planlagt for. Selv om oppgavene inneholder spørsmål som fordrer refleksjon over den underliggende matematikken, finner elevene måter å unngå dette på. Dette bygger også opp under Brousseaus (2006, s. 7) påstand om at man ikke kan forvente at valideringssituasjoner skjer uten at det er lagt opp til dette helt spesifikt. I tillegg beskriver Joubert (2017) datamaskinen som verdifull i forbindelse med rask tilbakemelding, men viser til utfordringer med at elevene gjennom denne tilbakemeldingen får bekreftet/avkreftet teoriene sine uten å selv måtte gjøre dette eksplisitt.

Geir og Harald som helt korrekt løser et problem (kapittel 4.1.2) eksemplifiserer denne utfordringen. Ettersom oppgaven ikke eksplisitt etterspør refleksjoner rundt svarene, utelater Geir og Harald å *validere* løsningen de har produsert ved å bevise eller forklare hvordan eller hvorfor koden produserer en gitt tallrekke. En konstantering av at svaret er korrekt blir sett på som godt nok. Det er tydelig at tilbakemeldingene fra programmet hjelper elevene med å sette ord på og bekrefte hypoteser underveis i arbeidet (formuleringsfaser), men elevene går ikke nærmere inn på hvorfor koden deres fungerte. I eksempelet kan man ikke fastslå hvorvidt Harald faktisk forstod hvordan koden fungerte, og man kan i værste fall anta at Harald ikke forstod noe av selve koden og at «læringseffekten» i oppgaven er betraktelig redusert. Her er det tydelig at det må andre virkemidler til for å få elevene til å gjøre det matematiske innholdet eksplisitt.

Det er også verdt å merke seg at i utdraget som klassifiseres som valideringssituasjon (kapittel 4.1.1), bes ikke elevene verken forklare, begrunne eller bevise noe som helst. Det spørres simpelthen hva som blir i rest når et tall deles på et annet. Her kan *motstanden* elevene møter i form av det antatte problemet *hva betyr modulo-operatoren*, være en forklaring på valideringsdialektikken som karakteriserer dette utdraget. Joubert (2017) hevder tilbakemelding fra datamaskiner kan ha stor påvirkning på elevenes handlingsmønster, nærmere bestemt hvorvidt de er i en aksjon-, formulerings- eller valideringssituasjon. Den umiddelbare tilbakemeldingen fra Python-skallet trigger en interesse og nysgjerrighet hos guttene som fører til en dialektikk hvor elevene steg for steg nærmer seg et punkt hvor de produserer en god forklaring for hvordan modulo-operatoren virker og begrunner denne med referanser i oppgavetekst og matematisk kunnskap.

Elevenes handlinger i situasjonene er også preget av at elevene kombinerer kroppsspråk med korte utsagn som referer til elementer i omgivelsene i stedet for å bruke fulle setninger med begreper i det de argumenterer for løsninger eller forandringer i koden – såkalt deiktisk kommunikasjon (Wyndhamn & Säljö, 2009). Selv om elevene bruker koden og output som referanse, for eksempel ved at de peker på koden, bruker de sjelden kunnskap om programmering eksplisitt som referanse i argumentasjonen eller resonnementene sine. Et slikt resonnement ville for eksempel vise til egenskapene til modulo-operatoren; *Modulo viser det som er i rest, hvis vi skriver inn 11 der, vil det vise hva som er i rest når tallet deles på 11*. Referansene er derimot basert på det umiddelbare output, det som vises på

skjermen. I eksempelet ovenfor uttrykker Geir og Harald enighet, nikker og snakker sammen, men hvorvidt de underliggende konseptene knyttet til modulo og rest, løkker og betingelser forstås forblir utydelig.

Ved at det kun forekommer ett tilfelle av valideringssituasjon i utdragene som er analysert, kan man spørre seg hvorvidt oppgavene og konteksten for øvrig gir elevene tilstrekkelig insentiv for *validering*. Oppgavene byr også åpenbart på ulik matematisk motstand for de forskjellige elevparene. Det som for Geir og Harald er oppgaver som skaper gode diskusjoner; formulering og validering, ser hos Ask og Brage ut til å være trivielt (utforske modulo-operator). Dette tyder på et behov for at et slikt opplegg er godt tilpasset elevenes faglige nivå (jamfør Vygotskys nærmeste utviklingssone) – både matematikkfaglig og programmeringsfaglig/-teknisk. I tillegg må oppgavene følges opp med tydeligere og mer tilrettelagte validerings- og institusjonaliserings-situasjoner. Lærers rolle ser også ut til å være viktig for å hjelpe elevene å utvikle et presist språk for konteksten og fremheve viktige faglige sammenhenger.

Brousseau og Gibel (2005) påpeker at elevens erfaringer i problemløsningssituasjonen må fremheves og settes inn i en faglig kontekst. Dette blir også påpekt som viktig i arbeid med programmering og matematikk av Benton et al. (2016) gjennom begrepene *Exchange* og *bridgE*: Å *utveksle* kunnskap i en sosial sammenheng og skape *koblinger* mellom ulike fagfelt (relevante konsepter). Ettersom oppgavene og programvaren (konteksten) helt klart legger til rette for elevers formuleringer, indikerer dette et potensiale. Det må imidlertid følges opp med situasjoner som i større grad legger til rette for utveksling av relevant underliggende informasjon i løsningene og eksplisitte vurderinger av gyldighet i løsningsforslagene.

En tilnærming for å få elevene til å utdype kan være tilleggsspørsmål, enten i oppgaveteksten eller i diskusjon med lærer. Dette kan potensielt være nyttig, noe man ser tendenser til hos Dag og Kari (kapittel 4.2.2). Her er det inkludert et oppfølgingsspørsmål om å forklare hvordan man kan bruke programmet til å finne tall som både er i `a`-gangen og i `b`-gangen? Forklaringen som følger er ikke sett på som tilstrekkelig for å være validering, ettersom Kari resonnerer og oppsummerer for seg selv uten respons fra Dag. Spørsmål av typen «forklar hvordan programmet virker», enten fra lærer underveis eller i oppgaveform, vil kunne hjelpe elevene å sette ord på hva det var som gjorde at koden deres faktisk

produserte ønsket tallrekke og således øke potensiale for at elevene tilegner seg ønsket kunnskap.

Som påpekt av Smith og Stein (2018) er spørsmål som forsøker å gjøre underliggende matematikk eksplisitt både krevende å stille og krevende å håndtere, spesielt i diskusjoner med hele klasser. Det forutsetter at lærer både har god forståelse for matematikken, elevenes utgangspunkt – og i denne sammenhengen også elementer knyttet til programmeringsspråket. Å stille spørsmålene skriftlig kan også være problematisk da man mister muligheten til å omformulere eller presisere dersom spørsmålet oppfattes på en måte som ikke var tiltenkt.

5.1.1.2 Formuleringsdialektikk – prøving og feiling som problemløsningsmetode

For å møte problemer på en konstruktiv måte, hevder Brousseau (2006, s. 85) at elevene må utsettes for stadig nye situasjoner hvor eventuelle misoppfatninger adresseres (ved epistemologisk motstand), og først ved å kommer forbi disse hindringene korrigeres misoppfatningene. Arbeidet til Dag og Kari illustrerer det Brousseau (2006) kaller formuleringsdialektikk; en faglig progresjon og utvikling i elevenes tilnærming til problemene de møter i oppgavene. Et viktig grunnlag skapes i dialogen, og det er tydelig at elevene møter motstand, bygger kunnskap og bruker denne underveis i prosessen. Dette ser vi spesielt i hvordan elevene går fra å gjette nærmest vilkårlig (**S1**, kapittel 4.2.2) til å benytte elementer i koden til å gi mer presise forutsigelser (**S4**, kapittel 4.2.2). Eksempelet viser hvordan oppgavedesignet, bestående av mange tilsynelatende like oppgaver, gjør at elevene forandrer, justerer, gjetter og forventer på bakgrunn av tilbakemeldingen som gis når elevene kjører kodene. Elevenes tilnærming til oppgavene karakteriseres av det Hana (2014) kaller «metoden med gjentatte tilnærminger», og innebærer en mer eller mindre systematisk prøving og feiling.

Det at elevene overlates til seg selv (a-didaktiske situasjoner) i problemløsningsfasen er et poeng for Brousseau (2006), da dette er nødvendig for å personliggjøre kunnskapen eleven utvikler underveis. Tilbakemeldingen elevene får fra lærer kan imidlertid virke korrigerende og konstruktiv, noe man ser hos Dag og Kari gjennom det dialektiske preget på situasjonene. Intervensjon fra lærer kan dermed sees på som en konstruktiv innblanding som hjelper elevene og en slik interaksjon behøver ikke nødvendigvis å være en motsetning til forståelse av a-didaktiske situasjoner beskrevet i TDS.

5.2 Programmeringsmiljø som utgangspunkt for matematisk utforsking

Det siste forskningsspørsmålet adresserer hvilken didaktisk funksjon programvaren spiller i de valgte situasjonene. Dette er viktig ettersom det hevdes at innføring av programmering og AT i matematikkundervisningen bør føre til endringer i måten matematikk undervises på (Disessa, 2018; Olive et al., 2009). Dette handler om å lage oppgaver og legge til rette for situasjoner som ikke ville vært mulig uten en programmeringskontekst. Å bruke programmering som redskap til å utforske begreper i matematikken vil i et slikt perspektiv også være et potensielt verktøy for det mye omtalte dybdelæringsbegrepet (Utdanningsdirektoratet, 2019a). Spesielt gjelder dette å erfare og bruke faglige konsepter på tvers av fagområder og i nye situasjoner. Ettersom Bray og Tangney (2017) konkluderer med at bruken av teknologi generelt sett ikke utnytter sitt fulle potensialle til å forandre hvordan matematikk undervises, illustrerer dette en utfordring for lærere. Ulike teknologiske redskaper brukes ofte for å forsterke tradisjonell undervisning og ny teknologi er i seg selv ikke nok til å skape ønsket forandring. Bruken må også tilpasses teknologien/redskapet.

5.2.1.1 Python som utgangspunkt for formuleringer

Programvaren (Python) helt klart hatt en viktig didaktisk funksjon i situasjonene som er analysert ettersom oppgavene i stor grad baserer seg på å modifisere koder og utforske resultatet. Analysene viser også at det valgte designet i stor grad har gitt grunnlag for utforsking og diskusjoner. Samspillet mellom elever, oppgaver og program er preget av det Brousseau (2006) kaller *formulering*, at elevene observerer, setter ord på og bygger opp et formålstjenlig språk i møte med konteksten. Hos enkelte elever er dette språket mer «rafinert», som hos Geir og Harald, mens hos andre er det mer preget av en utvikling fra mer «naive» forklaringer til mer presise. Det er også tydelig at diektisk språk (Wyndhamn & Säljö, 2009) spiller en viktig rolle i elevenes resonnering og argumentasjon ettersom referansene i stor grad blir *vist til* fremfor å bli forklart eksplisitt.

I arbeidet med oppgavene blir både resultatet (output) og koden grunnlag for utforsking og diskusjon i elevparene. Begge elevene ser samme output og disse referansene fungerer som fasilitatorer for matematiske diskusjoner. Som beskrevet av Herheim (2010), fungerer datamaskinen som en felles arena for elevenes arbeid (common ground). Det er derimot ikke gitt at elevene observerer og diskuterer på et nivå som involverer eksplisitt validering, beskrevet som *reasoning of level 3* av Brousseau og Gibel (2005). Det er heller ikke gitt at

konteksten gir likt utgangspunkt for de ulike elevparene. Her spiller elevenes tidligere kunnskap og forforståelse en viktig rolle.

Interaktiviteten som tilbys ved at elevene når som helst kan kjøre programmet og skape sammenheng mellom kode og resultat (et logisk mønster bestående av funksjoner, variabler, betingelser og så videre), gir grunnlag for å hevde at en slik tilnærming *redefinerer* oppgaver knyttet til utforsking av divisjon og rest. På samme måte som dynamiske geometri-program (for eksempel GeoGebra) kan sies å gi rom for å redefinere hvordan man utforsker grafer og funksjoner, vil arbeid med programmering kunne redefinere hvordan man leter etter mønster og utforsker sammenhenger mellom kode og output (her: tallrekker og tabeller). Denne vekselvirkningen mellom kode og output kan nærmest minne om behandling av konkreter. Eksempelet med Dag og Kari (preget av formuleringsdialektikk) kan i så måte illustrere en vekselvirkning mellom Manipulering, få en Forståelse for og Artikulering (MFA-modellen). Denne beskrives av John Mason, Leone Burton og Kaye Stacey (2010) som en nyttig pedagogisk konstruksjon i matematikkundervisning. Det at objektene for analyse (konkretene) er kode – som igjen er rike på underliggende matematisk innhold – gjør verktøyet unikt og konteksten nyskapende.

5.2.1.2 Utfordringer med programmeringskonteksten

I flere situasjoner ser elevene ut til å ta egenskaper ved koden og programmeringsmiljøet for gitt. Selv om elevene bruker koden og output som referanse, bruker de i mindre grad kunnskap om programmering som referanse i argumentasjon og resonnering underveis. Det er tydelig at Dag og Kari diskuterer matematikk (kapittel 4.2) og har fokus på det matematiske innholdet, men de ser tilsynelatende forbi aspekter ved programmering som for eksempel hvilken rolle modulo-operatoren spiller når de ulike tallrekkene skrives ut i konsollen som en tabell. De verken diskuterer eller nevner hvilken betydning modulo spiller for betingelsen i koden. De reflekterer heller ikke selvstendig hvorfor output blir som det blir. Referansene til elevene er i større grad basert på det umiddelbare output, nemlig tabellen som produseres, fremfor de underliggende mekanismene som fører til output (jamfør koden). Hvorvidt ferdigskrevne kodesnutter er en årsak til dette blir bare spekulasjon, men oppgaver hvor elever lager koder fra bunn (enten med tekst eller blokkbasert programmering) for å produsere spesifikke tallrekker, ville vært interessant å arbeide med videre. Når Benton et al. (2016) beskriver *bridgE* som et konsept i forbindelse

med programmering og matematikkundervisning er det nettopp for å formidle og gjøre sammenhengen mellom overlappende fagdisipliner eksplisitt. For eksempel, hvordan påvirker koden de ulike komponentene (betingelser og modulo-operator) det matematiske innholdet som diskuteres?

At kodene i stor grad er gitt i eksemplene fra datamaterialet gjør at man kan stille spørsmål om hvorvidt elevene i det hele tatt «programmerer» eller «koder». På den ene siden ser man at noen elever leser koden bevisst og bruker den for å skape mening i konteksten, som Brage når han leser gjennom koden tilhørende oppgaven (kapittel 4.3.2). På den andre siden virker koden av og til å være irrelevant, mer som en «motor» som utfører «noe», men akkurat hvordan det skjer er mindre viktig. For Dag og Kari virker sistnevnte å være tilfelle, ettersom de primært diskuterer output.

Tidlig i prosessen opplevde jeg et behov for mer lukkede oppgaver på grunn av elevenes generelle utfordringer med å lage programmer fra bunn med Python, samt for å kunne fokusere eksplisitt på matematikken. Det er imidlertid vanskelig å treffe alle elevene når oppgavene er detaljerte og progresjonen bestemt på forhånd. For Ask og Brage kan det virke som om en streng progresjon virker negativt. Som to høytpresterende elever med stor interesse for programmering viste de overraskende lite engasjement og lite faglige diskusjoner sammenlignet med andre elevpar i studien. Arbeidet deres med modulo-operatoren blir beskrevet ved en rekke aksjonssituasjoner, noe som begrunnes med umiddelbare løsninger på oppgavene og lite motstand. Utfordringen med for detaljert planlegging beskrives av Misfeldt og Ejsing-Duun (2015) som *the planning paradox*, hvor lærers ønske om matematisk måloppnåelse eller fokus på matematikk fører til «for nøye» planlegging slik at det motvirker elevers opplevelse av programmering som et nyttig redskap. Her kan man spørre seg hvorvidt de strenge rammene det er lagt opp til i oppgavedesignet faktisk virker hemmende for enkelte elever.

Sfard og Leron (1996) beskriver hvordan datalabben og matematikk-klasserommet kunne se ut som to forskjellige verdener, med forskjellige holdninger og handlingsrom. Problemer som i matematikkundervisningen viste seg å være vanskelige, ble til og med i mer komplekse former løst av elevene når de jobbet i datalabben. Forskerne forklarer fenomenet blant annet med kulturen som ofte setter sitt preg på matematikk-klasserommet. Ved å skifte miljø ble blant annet elevenes toleranse for å gjøre feil større. Det som i matematikkrommet

var skummelt og negativt, ble i et digitalt miljø en viktig del av arbeidet. I et slikt perspektivt kan man spørre seg hva som skjer med de positive aspektene ved programmering, for eksempel økt motivasjon (Forsström & Kaufmann, 2018), når koding og programmering blir brukt som et redskap i matematikkundervisning i stedet for som aktivitet i seg selv. Gjøres programmering om fra noe nytt og spennende, til noe som «skolsk» (i negativ forstand) og snevert for elevene?

5.2.1.3 Tekstbasert programmering på mellomtrinnet?

Når det gjelder utfordringer i forbindelse med tekstbasert programmering, er det helt klart flere aspekter som kan nevnes. De første fasene av prosjektet (økt 1-8) var preget av en rekke didaktiske og tekniske utfordringer knyttet til bruk av tekstbasert programmering. De mest utpregede har vært utfordringer i forbindelse med syntaks; elevene får feilmeldinger når de skriver inn feil ord, feil notasjon og lignende. Enkelte konsepter, spesielt bruk av variabler og løkker viste seg utfordrende å jobbe med for flere av elevene. Ikke minst viste det seg hos en del elever at veien fra en idè til et fungerende program var lang, spesielt når kunnskapen om programmeringsspråket var begrenset.

Arbeidet med masteroppgaven har vært preget av å finne måter å imøtekomme og redusere disse utfordringene på slik at programmeringskonteksten kan være et utgangspunkt for matematisk utforsking. For eksempel er det i det siste undervisningsopplegget (økt 10) valgt å gå fra Python IDLE hvor elevene skriver inn koden selv, til en trinket hvor elevene blir presentert for en ferdig kode. Ettersom fokuset på dette stadiet først og fremst var på det matematiske innholdet, ble dette sett på som en pragmatisk og uproblematisk tilnærming. Valget i seg selv illustrerer at mange typer arbeid ville vært uaktuelt dersom man benytter seg av Python som programmeringsspråk med yngre elever. Arbeid som tar sikte på å skape programmer fra bunnen ville sannsynligvis vært lettere tilgjengelig dersom de i stedet hadde arbeidet med blokkbasert programmering. Jeg mener likevel bruken av Python har vist seg som en spennende innfallsvinkel ved at elevene utforsker matematikk, samtidig som de ser og «tukler med maskineriet» i et program.

Det finnes også indikasjoner på at en dobbel tilnærming, hvor man både tar i bruk blokkbasert- og tekstbasert programmering, kan være en god måte å forsterke forståelsen for konsepter på (Weintrop & Wilensky, 2019). Spesielt tenker jeg tilnærmingen beskrevet i økt 10 (vedlegg V), med ferdigskrevne kodesnutter i en trinket viser at man kan bruke et

tekstbasert programmeringsspråk selv om elevene har begrenset kunnskap. En interessant vinkling ville vært å gjennomføre lignende oppgaver med elever som har tidligere erfaring med for eksempel Scratch. Spesielt er dette interessant ettersom det å lese kode er en aktivitet som i seg selv er nyttig i forbindelse med å lære programmering (Busjahn & Schulte, 2013). Sett i en kontekst av matematikk-klasserommet vil man også på et lavere nivå få mange gode diskusjoner rundt matematisk innhold (tolke mønster og løse problemer), noe jeg mener resultatene fra denne studien illustrerer. Her vil det i så fall være gode muligheter for å diskutere programflyt og hvordan løkker og variabler spiller inn i koden, i tillegg til det matematiske innholdet. Som Tabet, Gedawy, Alshikhabobakr og Razak (2016) viser, kan også erfaring med blokkbasert programmering forsterke elevers forståelse av Python, noe som taler til fordel for å la elevene få erfaringer med flere programmeringsspråk.

5.2.1.4 Digitale verktøy og oppgavedesign

Oppgavedesignet som er laget i denne studien aktualiserer også spørsmål som matematikklærere må stille seg fremover; Vil for eksempel lesing av, utforsking av og modifisering av kode være en del av begrepet «programmering»? Ifølge læreplanen skal elevene etter 7. årstrinn «bruke programmering til å utforske data i tabellar og datasett» (Utdanningsdirektoratet, 2019b). I hvilken grad vil en tilnærming lik den som er valgt i oppgavesettene mine dekke dette målet? Dersom man imidlertid mener en slik tilnærming *ikke* er dekkende, illustrerer dette ekstra tydelig utfordringen lærere står ovenfor i møte med de nye læreplanene når disse skal tolkes og overføres til klasserommet.

Joubert (2017) spør hvordan bruk av datamaskiner som digitalt vertkøy påvirker prosessen med å lage gode oppgaver i matematikkundervisningen, og konkluderer blant annet med at man som lærer og oppgavedesigner må stille seg selv noen viktige spørsmål når redskaper velges ut: Hvilken matematikk er overlatt til elevene? Hva er overlatt til datamaskinen? Legger programvaren til rette for at elevenes handlingsmønstre kan karakteriseres som god matematisk aktivitet, eller brukes programvaren på en måte som gjør elevene passive og tause? Resultatene fra denne studien viser at elevarbeidet i hovedsak er preget av å lete etter mønstre, modifisere kode, vurdere, gjøre antakelser, observere og diskutere (noe som inkluderer argumentasjon og resonnering). Dette kan absolutt karakteriseres som matematisk aktivitet. Det tradisjonelle arbeidet med å gjøre beregninger er imidlertid overført til datamaskinen.

5.3 Refleksjoner rundt rammeverk

Som nevnt er Brousseau (2006) sin teori brukt som et teoretisk rammeverk for analyse av data i etterkant, fremfor som utgangspunkt for selve oppgavedesignet. I motsetning til å designe matematiske situasjoner spesifikt for å fremprovosere aksjons-, formulerings- og valideringssituasjoner, er det laget et oppgavesett med fokus på utforsking og utprøving. Dette kan ifølge Artigue et al. (2014) være nyttig, men også utfordrende, ettersom en må tilpasse begrepene en kontekst som er noe forskjellig fra de som beskrives i TDS. Dette er en viktig distinksjon ettersom situasjonene kunne vært annerledes om de hadde vært laget på bakgrunn av det Brousseau (2006) kaller *didactical engineering*. En slik tilnærming vil handle om å lage undervisningsopplegg som i større grad sikter spesifikt inn på de ulike situasjonsformene, slik som skissert i «the race to 20»; En aktivitet rettet mot aksjon (prøve ut), en aktivitet med fokus på å observere, forklare og dele (formulering) og til slutt en økt hvor man skal legge frem teorier og overbevise andre. Det er også først ved å utvide oppleggene som presenteres her, jeg vil ta høyde for Brousseau sitt begrep *institutionalization*, som handler mer om å gjøre elevenes teorier, hypoteser, formaninger formelle, sette de inn i en matematikk-teoretisk sammeheng. En slik tilnærming vil i så fall være med å adressere utfordringene med at matematikken forblir implisitt eller skjult (Forsström & Kaufmann, 2018).

SAMR-modellen gir et interessant perspektiv på hvordan teknologien (programvaren) brukes, men modellen oppleves også relativ. Hva er for eksempel «gammel» teknologi? Hva er en «ny» oppgave og hva skal man sammenligne med? Det kan også spørres hvorvidt modellen kommer til kort når utgangspunktet blir sirkulær argumentasjon: det faktum at oppgaven tar i bruk programmering gjør at undervisningen redefineres. Kunne *målet* med oppgaven vært gjennomført like godt ved å levere ut tabeller på ark og be elevene lete etter og beskrive mønstre? Ut fra et SAMR-perspektiv vil jeg likevel hevde det faktum at elevene jobber i et programmeringsmiljø (python-trinket) helt klart *transformerer* og til en viss grad *redefinerer* undervisningen. Hvorvidt lignende aktivitet hadde vært mulig med andre virkemidler er nærmest umulig å svare på da innsikt i *alle* tilgjengelig verktøy ikke er reelt. Spørsmålet blir snarere hvorvidt en endring ved tilnærmingen fører til er positiv og konstruktiv forandring, eller om det i det hele tatt er et skritt i riktig retning med tanke på bruk av digitale verktøy? Basert på resultatene i denne studien mener jeg svaret på

sistnevnte helt klart er ja, men det krever nøye planlegging og ikke minst oppfølging for å sikre at både det matematiske innholdet og programmeringskonseptene blir gjort eksplisitt.

6 Konklusjon

Masteroppgaven min kan sees som et svar på oppfordringene fra blant annet Disessa (2018) og Bray og Tangney (2017) om å utforske mulighetene programmering har som verktøy i matematikkfaget. I studien har jeg utviklet flere oppgavesett rettet mot utforsking av tall i en programmeringskontekst med det tekstbaserte språket Python.

Gjennom oppgavedesignet og undervisningstilnærmingen som er valgt, viser resultatene et tydelig potensiale for å redefinere aspekter ved tradisjonell matematikkundervisning. Ved å bruke TDS som ramme for å studere samspillet mellom oppgave, elev og programvare, er det tydelig at Python tilbyr en interessant og spennende ramme/kontekst for utforsking av tallrekker, spesielt knyttet til divisjon og rest. Oppgavedesignet og programvaren har skapt et utgangspunkt hvor eleveparene har vært aktivt involvert i matematiske prosesser, først og fremst preget av det Brousseau (2006) kaller formuleringssituasjoner.

Utfordringene knyttes i hovedsak til at elevene ikke på egenhånd bygger bro mellom programmering og matematikk, og at valideringssituasjoner i hovedsak uteblir – en utfordring som også blir påpekt i litteraturen. Situasjonene som er analysert viser imidlertid elever som diskuterer, observerer, ser etter mønstre, beskriver, resonnerer og argumenterer. Situasjonene er i tillegg rike på matematiske sammenhenger og det antas at man gjennom veiledning og mer fokusert arbeid kan belyse disse sammenhengene og knytte dem opp mot relevante programmeringsaspekter som løkker, betingelser og variabler. Særlig bruk av Modulo-operator har vist seg interessant i kombinasjon med Python for å utforske divisjon og rest. Her ligger også en tydelig implikasjon om at man ved hjelp av videre tilrettelegging og oppfølging av lærer, kan legge til rette for valideringssituasjoner på bakgrunn av det elevene observerer, diskuterer og setter ord på underveis – særlig gjelder dette forholdet mellom matematisk innhold, kode og output.

Nye fagplaner fordrer forandring og nytenkning, særlig i tilknytning til digitale verktøy. Masteroppgaven min er en liten brikke i en større sammenheng hvor det søkes innsikt i hvordan programmering skal forstås i norsk skole. Det er forsøkt å operasjonalisere og implementere noe som formelt sett er innført (fra 2020), men som for mange lærere oppleves som uhåndgripelig og fjernt – nemlig programmering og AT. Ved at prosjektet har foregått på mellomtrinnet og har tatt i bruk Python som programmeringsspråk, er studien

også et bidrag til å belyse tema det er forsket mindre på, men som fortsatt er relevant i forbindelse med fagfornyelsen. Denne oppgaven bidrar gjennom tilnærmingen som er valgt til å belyse *noen* av utfordringene og mulighene, men det trengs ytterligere forskning for å belyse disse videre.

Referanser

Aasen, M., Waaler, N. & Vinje, K. (2018). Er vi klare for programmering i skolen? Hentet 13.04 2020 fra https://khrono.no/marianne-aasen-programmering-programmering-i-skolen/er-vi-klare-for-programmering-i-skolen/247840

Alrø, H., & Kristiansen, M. (1997). Mediet er ikke budskabet: video i observation af interpersonel kommunikation. I Alrø, H. : Dirckinck-Holmfeld, L. (red.) (red.), *Videoobservation* (s. 73-99). Aalborg Universitetsforlag. Interpersonel Kommunikation i Organisationer, Nr. 3.

Artigue M., Haspekian M., Corblin-Lenfant A. (2014) Introduction to the Theory of Didactical Situations (TDS). I: Bikner-Ahsbahs A., Prediger S. (red.) *Networking of Theories as a Research Practice in Mathematics Education. Advances in Mathematics Education*. Springer, Cham

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2016). Building mathematical knowledge with programming: insights from the ScratchMaths project. Suksapattana Foundation. Hentet 12.03 2020 fra https://discovery.ucl.ac.uk/id/eprint/1475523/1/The%20ScratchMaths%20Project%20accepted.pdf

Benton, L., Saunders, P., Kalas, I., Hoyles, C. & Noss, R. (2018). Designing for learning mathematics through programming: A case study of pupils engaging with place value. *International Journal of Child-Computer Interaction, 16*, 68-76. Hentet 16.04.2020 fra https://doi.org/10.1016/j.ijcci.2017.12.004

Blichfeldt, B. S. & Andersen, J. R. (2006). Creating a wider audience for action research: Learning from case-study research. *Journal of Research Practice, 2*(1), 2.

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The nordic approach to introducing computational thinking and programming in compulsory education*. National Research Council of Italy, Institute for Educational Technology (CNR-ITD). Hentet fra https://www.itd.cnr.it/doc/CompuThinkNordic.pdf

Bray, A. & Tangney, B. (2017). Technology usage in mathematics education research – A systematic review of recent trends. *Computers & Education, 114*, 255-273. https://doi.org/10.1016/j.compedu.2017.07.004

Brousseau, G. (2006). *Theory of didactical situations in mathematics: Didactique des mathématiques, 1970–1990.* Springer Science & Business Media.

Brousseau G., Gibel P. (2005) Didactical Handling of Students' Reasoning Processes in Problem Solving Situations. I: Laborde C., Perrin-Glorian MJ., Sierpinska A. (red.) *Beyond the Apparent Banality of the Mathematics Classroom*. Springer, Boston, MA

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

Busjahn, T. & Schulte, C. (2013). The use of code reading in teaching programming. *Proceedings of the 13th Koli Calling international conference on computing education research* (s. 3-11).

Clark, S. & Lee, L. (2019). Technology Enhanced Classroom for Low-Income Children's Mathematical Content Learning: A Case Study. *International Journal of Information and Education Technology, 9*(1).

Cohen, L., Morrison, K. & Manion, L. (2007). *Research methods in education* (6. utg.). London: Routledge.

Creswell, J. W. (2014). *Research design : qualitative, quantitative, and mixed methods approaches* (4. utg.). Los Angeles, Calif: SAGE.

Disessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning, 20*(1), 3-31. Hentet 24.04 2020 fra: https://doi.org/10.1080/10986065.2018.1403544

Downey, A. B. (2017). *Modeling and Simulation in Python* (Versjon 3.4.0). Needham, Massachusetts: Green Tea Press.

Drijvers, P. (2015). Digital technology in mathematics education: Why it works (or doesn't). *Selected regular lectures from the 12th international congress on mathematical education* (s. 135-151). Springer.

Emmy-Charlotte Förster, Klaus-Tycho Förster & Löwe, T. (2018). *Teaching programming skills in primary school mathematics classes: An evaluation using game programming*. Innlegg presentert ved 2018 IEEE Global Engineering Education Conference (EDUCON), Tenerife, Spain. Abstract hentet fra http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8363411&isnumber=8363090

Feurzeig, W., Papert, S. A. & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487-501. https://doi.org/10.1080/10494820903520040

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

Gjøvik, Ø., Torkildsen, H. A. (2019). Algoritmisk tekning. *Tangenten – tidsskrift for matematikkundervisning*, 30*(3)*, 31–37.

Hamilton, E. R., Rosenberg, J. M. & Akcaoglu, M. (2016). The substitution augmentation modification redefinition (SAMR) model: A critical review and suggestions for its use. *TechTrends, 60*(5), 433-441.

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

Hanna, G. (2014). Mathematical Proof, Argumentation, and Reasoning. I S. Lerman (Red.), *Encyclopedia of Mathematics Education* (s. 404-408). Dordrecht: Springer Netherlands.

Herheim, R. (2010). Communication and learning at computers: An overview. *Nordic Studies in Mathematics Education, 15*(2), 69-94.

Hilton, J. T. (2016). A case study of the application of SAMR and TPACK for reflection on technology integration into two social studies classrooms. *The Social Studies, 107*(2), 68-73.

Johan, L., Inge Olav, H. & Tamsin Jillian, M. (2017). Computer programming in the lower secondary classroom: learning mathematics. *Italian Journal of Educational Technology, 25*(2). https://doi.org/10.17471/2499-4324/911

John Mason, Leone Burton & Kaye Stacey (2010). *Thinking mathematically* (2. utg.). Harlow: Prentice Hall. (Opprinnelig utgitt: London: Addison-Wesley, 1982)

Joubert, M. (2013). Using computers in classroom mathematical tasks: revisiting theory to develop recommendations for the design of tasks. *Task Design in Mathematics Education. Proceedings of ICMI Study 22*, 69.

Joubert, M. (2017). Revisiting theory for the design of tasks: Special considerations for digital environments. I *Digital Technologies in Designing Mathematics Education Tasks* (s. 17-40). Springer.

Khasawneh, A. A. (2009). Assessing Logo programming among Jordanian seventh grade students through turtle geometry. *International Journal of Mathematical Education in Science and Technology, 40*(5), 619-639. https://doi.org/10.1080/00207390902912845

Kieran, C. (2019). Task Design Frameworks in Mathematics Education Research: An Example of a Domain-Specific Frame for Algebra Learning with Technological Tools. I G. Kaiser & N. Presmeg (Red.), *Compendium for Early Career Researchers in Mathematics Education* (s. 265-287). Cham: Springer International Publishing.

Koehler, M. & Mishra, P. (2009). What is technological pedagogical content knowledge (TPACK)? *Contemporary issues in technology and teacher education, 9*(1), 60-70.

Leung, A. & Bolite-Frant, J. (2015). Designing mathematics tasks: The role of tools. I *Task design in mathematics education* (s. 191-225). Springer.

Maxwell, J. A. (2005). *Qualitative research design: An interactive approach* (2. utg.) London: Sage publications.

McCoy, L. P. (1996). Computer-Based Mathematics Learning. *Journal of Research on Computing in Education, 28*(4), 438-460. https://doi.org/10.1080/08886504.1996.10782177

McNiff, J. (2006). All you need to know about action research. I J. Whitehead (Red.),(s. IV, 274 s.). London: Sage publications.

Melby-Lervåg, M. (2018). Er det lurt å "lære kidsa koding" i skolen? Ny kunnskapsoppsummering. Hentet 27.05 2020 fra https://utdanningsforskning.no/artikler/er-det-lurt-a-lare-kidsa-koding-i-skolen-ny-kunnskapsoppsummering/

Misfeldt, M. & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. *CERME 9 - Ninth Congress of the European Society for Research in Mathematics Education*, Charles University in Prague, Faculty of Education; ERME, Feb 2015, Prague, Czech Republic. s. 2524-2530.

Måsøval, H. S. (2011). *Factors constraining students' establishment of algebraic generality in shape patterns: A case study of didactical situations in mathematics at a university college*. Doktorgradsavhandling, Universitetet i Agder.

Nordseth, L. K. (2017). IKT-Norge: – Koding må inn i læreplanen for grunnskolen. Hentet 13.04 2020 fra https://www.tu.no/artikler/ikt-norge-koding-ma-inn-i-laereplanen-for-grunnskolen/398172

Olive J., Makar K., Hoyos V., Kor L.K., Kosheleva O., Sträßer R. (2009) Mathematical Knowledge and Practices Resulting from Access to Digital Technologies. I: Hoyles C., Lagrange JB. (red.) *Mathematics Education and Technology-Rethinking the Terrain*. New ICMI Study Series, vol 13. Springer, Boston, MA.

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

Popat, S. & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education, 128*, 365-376. https://doi.org/10.1016/j.compedu.2018.10.005

Puentedura, R. R. (2013a). The SAMR Ladder: Questions and Transitions. Hentet 23.04 2020 fra http://www.hippasus.com/rrpweblog/archives/2013/10/26/SAMRLadder_Questions.pdf

Puentedura, R. R. (2013b). SAMR: A contextualized introduction. Hentet 23.04 2020 fra http://www.hippasus.com/rrpweblog/archives/2013/10/25/SAMRAContextualizedIntroduction.pdf

Romrell, D., Kidder, L. & Wood, E. (2014). The SAMR model as a framework for evaluating mLearning. *Online Learning Journal, 18*(2).

Saha, A. (2015). *Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More!* No Starch Press.

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. Journal of Educational Psychology, 111(5), 764–792. Hentet 20.04.2020 fra https://doi-org.galanga.hvl.no/10.1037/edu0000314

Sevik, K. (2016). *Programmering i skolen*. Senter for IKT i utdanningen. Hentet 18.01.2020 fra https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Sfard, A. & Leron, U. (1996). Just give me a computer and I will move the earth: Programming as a catalyst of a cultural revolution in the mathematics classroom. *International Journal of Computers for mathematical learning, 1*(2), 189-195.

Smith, M. S. & Stein, M. K. (2018). *5 practices for orchestrating productive mathematics discussions* (3. utg.). Reston: National Council of Teachers of Mathematics.

Tabet, N., Gedawy, H., Alshikhabobakr, H. & Razak, S. (2016). From alice to python. Introducing text-based programming in middle schools. *Proceedings of the 2016 ACM Conference on innovation and Technology in Computer Science Education* (s. 124-129).

Tangen, R. (2010). "Beretninger om beskyttelse" ; etiske dilemmaer i forskning med sårbare grupper - barn og ungdom. *Norsk Pedagogisk tidsskrift, 94*(4), 318-329.

Tedre, M. & Denning, P. J. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (s. 120-129): ACM.

Udirbloggen. (2017). Kjerneelementer i matematikk, men hvorfor programmering? Hentet 26.11.2019 fra https://udirbloggen.no/kjerneelementer-i-matematikk-men-hvorfor-programmering/

Utdanningsdirektoratet. (2019a). Dybdelæring. Hentet 01.04.2020 fra https://www.udir.no/laring-og-trivsel/dybdelaring/

Utdanningsdirektoratet. (2019b). *Læreplan i matematikk 1.–10. trinn (MAT01-05)*. Hentet 01.02.2020 fra https://www.udir.no/lk20/mat01-05

Watson, A. & Thompson, D. R. (2015a). Design Issues Related to Text-Based Tasks. I A. Watson & M. Ohtani (Red.), *Task Design In Mathematics Education - an ICMI study 22* (s. 157-204). Springer.

Watson, A. & Thompson, D. R. (2015b). Design Issues Related to Text-Based Tasks. I A. Watson & M. Ohtani (Red.), *Task Design In Mathematics Education - an ICMI study 22* (s. 205-). Springer.

Weintrop, D. & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education, 142*. https://doi.org/10.1016/j.compedu.2019.103646

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. Hentet 25.10.2019 fra https://doi.org/10.1145/1118178.1118215

Wyndhamn, J. & Säljö, R. (2009). Meaning-Making and the Appropriation of Geometric Reasoning: Computer Mediated Support for Understanding the Relationship between Area and Perimeter of Parallelograms. I R. J. Krumsvik (Red.). *Learning in the Network Society and the Digitized School* (s. 21–40). New York: Nova Science Publishers, Inc.

Vedlegg

Vedlegg I: Samtykkeskjema foresatte

Vil du delta i forskningsprosjektet
«Argumentasjon og kritisk matematikkundervisning i flerspråklige klasserom»?

Dette er et spørsmål om å delta i et forskningsprosjekt om argumentasjon og kritisk tenkning i matematikkundervisning i flerspråklige klasserom. I dette skrivet informerer vi kort om innholdet i prosjektet og hva deltakelse innebærer.

Bakgrunn og formål
Prosjektet handler om å fremme lærerstudenters kompetanse i å legge til rette for argumentasjon og kritisk matematikkundervisning for elever i flerspråklige klasserom på barnetrinnet. Dette kan være å kritisk kunne vurdere matematikkforklaringer og å se matematikkens rolle i argumentasjon om aktuelle samfunnsspørsmål. Skolene som er med i prosjektet er partnerskoler eller praksisskoler som allerede er en del av et samarbeid mellom Bergen kommune og Høgskulen på Vestlandet (HVL). Prosjektet varer i fire år og er et forskningssamarbeid mellom lærere og elever ved partnerskoler og tilsatte og studenter ved matematikklærerutdanningen ved HVL.

Som en del av dette forskningsprosjektet ønsker en masterstudent (Fredrik Eidsvåg) å samle inn datamateriale der elever arbeider med tekstbasert programmering i matematikkundervisningen. Målet er å få innsikt i hvordan programmering som verktøy påvirker matematisk argumentasjon.

Hvem er ansvarlig for forskningsprosjektet?
Høgskolen på Vestlandet er ansvarlig for prosjektet, og det er ledet av Professor Tamsin Meaney. Prosjektet gjennomføres i samarbeid med Bergen Kommune, og det er støttet av Norges forskningsråd.

Hvorfor får du spørsmål om å delta?
Vi spør om å la ditt/deres barn få delta i prosjektet fordi vi ønsker å observere/studere hvordan bruk av programmering i matematikkundervisningen påvirker elevers læring. Timelærer i matematikk/naturfag på 7. trinn, Fredrik Eidsvåg, er også masterstudent og deltar i forskningsprosjektet. Skolen v. rektor stiller seg for øvrig positiv til prosjektet.

Hva innebærer det for deg å delta?
Det vil bli gjennomført undervisning der elevene arbeider i par/grupper med matematikk og tekstbasert programmering (Python). Det er særlig fokus på matematisk argumentasjon der elevene oppfordres til å reflektere, diskutere og dele kunnskap. Undervisningen vil være lik for hele klassen, men et utvalg elever som ønsker å delta i prosjektet vil bli spurt om å jobbe parallelt på grupperom hvor det vil bli gjort lyd- og video-opptak.

Det er frivillig å delta
Det er frivillig å delta i studien, og du/dere kan uten grunngiving når som helst trekke ditt/deres samtykke. Hvis du/dere trekker barnet fra prosjektet vil alle opplysninger om barnet bli anonymisert, og lyd- og video-opptak av barnet vil bli slettet. Det vil ikke få negative konsekvenser hvis du/dere ikke ønsker at barnet skal delta, eller senere velger å trekke ditt/deres barn fra prosjektet.

Ditt/deres barns personvern – hvordan vi oppbevarer og bruker barnets opplysninger
Alle personopplysninger blir behandlet konfidensielt og personidentifiserbart materiale lagres på HVL sin forskningsserver, sikret med brukernavn og passord. Kun deltakere i prosjektgruppen og eventuelt transkriberingsfirma har tilgang til materialet. Deltakere vil ikke kunne bli identifisert i publikasjoner.

Prosjektet skal avsluttes 31.12.2023. Etter denne dato vil alle personidentifiserende data slettes og materialet vil ikke lengre være lagret på HVL sin forskningsserver. Videre bruk av dataene blir i presentasjoner, undervisning, eventuelle oppfølgingsstudier og senere forskning basert på transkribert og anonymisert materiale.

Dine/deres rettigheter
Så lenge ditt barn kan identifiseres i datamaterialet, har du/dere rett til:

- innsyn i hvilke personopplysninger som er registrert om barnet ditt,
- å få rettet personopplysninger om barnet ditt,
- å få slettet personopplysninger om barnet ditt,
- å få utlevert en kopi av ditt barns personopplysninger (dataportabilitet), og
- å sende klage til personvernombudet eller Datatilsynet om behandlingen av ditt barns personopplysninger.

Hva gir oss rett til å behandle personopplysninger om ditt/deres barn?
Vi behandler opplysninger om ditt/deres barn basert på ditt/deres samtykke. På oppdrag fra HVL har Norsk senter for forskningsdata (NSD) vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

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

- Prosjektleder Tamsin Meaney på tlf.: 55 58 55 69 eller epost: Tamsin.Jillian.Meaney@hvl.no
- Masterstudent:
  - Fredrik Eidsvåg på tlf.: 98428392 eller epost: 132057@stud.hvl.no
- HVL sitt personvernombud: Advokat Halfdan Mellbye, personvernombud@hvl.no
- NSD – Norsk senter for forskningsdata AS, på epost: personverntjenester@nsd.no eller telefon: 55 58 21 17.

✂ -

Samtykkeerklæring forskningsprosjektet

Jeg/vi har mottatt og forstått informasjon om prosjektet «Argumentasjon og kritisk matematikkundervisning i flerspråklige klasserom» og fått anledning til å stille spørsmål. Jeg/vi samtykker til at barnet mitt/vårt kan:

☐ delta i videopptak
☐ delta i lydopptak

Jeg/vi samtykker til at mitt/vårt barns opplysninger behandles frem til prosjektet er avsluttet, 31.12.2023.

(Signert av prosjektdeltakers foresatte, dato)

Samtykkeerklæring for videre bruk av videoer:

☐ Jeg samtykker i at videosnutter der mitt/vårt barn er med kan vises i presentasjoner og undervisning ut over prosjektets avslutning i 2023. Disse vil ikke være anonyme.

(Signert av prosjektdeltakers foresatte, dato)

Vil du delta i forskningsprosjekt om programmering i matematikkfaget?

Hei! Jeg heter Fredrik Eidsvåg og i tillegg til å undervise matematikk på trinnet, studerer jeg ved Høgskolen på Vestlandet. Jeg ønsker å undersøke hvordan man kan bruke programmering i matematikkundervisningen. Da trenger jeg å filme noen av dere og ta skjermopptak av datamaskinen. Dette gjør jeg fordi jeg skal skrive en masteroppgave om programmering og matematikk.

Du kan velge om vi skal få lov til å filme deg. Dersom du ønsker å være med vil det kun være vi som jobber med prosjektet som får se og høre opptakene av deg. Dette er helt frivillig, og du kan ombestemme deg når som helst. Ingen vil kunne gjenkjenne deg når de leser masteroppgavene våre.

Hvis du lurer på noe, ta gjerne kontakt:

- Fredrik Eidsvåg på tlf.: 98428392 eller på epost: 132057@stud.hvl.no

✂ -

Samtykkeerklæring forskningsprosjektet

Jeg har mottatt og forstått informasjon om prosjektet og fått anledning til å stille spørsmål. Jeg synes det er greit at jeg:

☐ deltar i videopptak

Jeg synes det er greit at mine opplysninger behandles frem til prosjektet er avsluttet, 31.12.2021.

(Min signatur (elev), dato)

Samtykkeerklæring for bruk av videoer:

☐ Jeg synes det er greit at videosnutter der jeg er med kan vises i presentasjoner og undervisning.

(Min signatur (elev), dato)

Vedlegg III: Godkjenning av rektor til å gjennomføre prosjekt?

22.9.2019 [redacted]

SV: Masteroppgave og datainnsamling på 7.trinn

[redacted]

ti 17.09.2019 11:02

Til:Eidsvåg, Fredrik [redacted]

Hei,

Jeg bekrefter at vi er positivt innstilt til dette prosjektet, og at det kan gjennomføres på skolen som beskrevet.

Med vennlig hilsen
[redacted]
Rektor
[redacted]

Fra: Eidsvåg, Fredrik [redacted]
Sendt: tirsdag 17. september 2019 10:58
Til: [redacted]
Emne: Masteroppgave og datainnsamling på 7.trinn

Hei,

Referer til tidligere samtale angående innsamling av data til masteroppgave på [redacted]skole. I forbindelse med masteroppgave i undervisningsvitenskap ønsker jeg å filme og ta lydopptak av enkelte elever på 7.trinn mens de arbeider med matematikk. Elevene som blir spurt om å delta vil bli informert om at deltakelse er frivillig, og det vil bli sendt eget samtykke- og informasjonsskriv til foresatte. Målet med prosjektet er å undersøke hvordan tekstbasert programmering kan brukes som verktøy i matematikkundervisning. Oppgaven er forøvrig en del av et pågående forskningsprosjekt ved HVL (LATACME). For å få godkjent søknad til NSD trenger jeg en skriftlig bekreftelse fra rektor om at skolen er informert om og godkjenner at prosjektet finner sted på skolen. Positivt svar på denne mailen fungerer godt nok på bekreftelse.

Mvh. Fredrik Eidsvåg

https://[redacted].no/owa/#path=/mail 1/1

Steg 1: Utforske `range()` funksjonen

Funksjonen `Range()` gir oss en rekke tall.

En `for`-løkke kan brukes til å printe ut tallene i `range()` funksjonen

```
for tall in range(10):
    print(tall)
```

Oppgave 1:

a) Her er det valgt `range(10)`. Prøv med forskjellige tall og kjør koden. Hva legger dere merke til?
b) `Range(5,10)` viser til tallene mellom 5 og 9. Hva må skrives inn for å printe ut tallene fra 10 til 30?
c) Hva med tallene mellom 900 og 1000?

Steg 2: Utforske modulo-operatoren

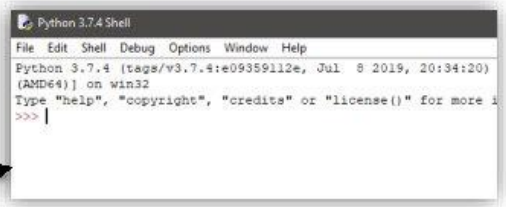
Modulo-operatoren (%-tegnet) gir oss det som blir til overs (**rest**) når et tall deles med et annet.

```
>>> 15 % 3
0
>>> 15 % 4
3
>>> 10 % 3
1
>>> 10 % 5
0
```

15 delt på 3 gir 0 i rest. Det betyr at 15 er **delelig** med 3.

Det betyr også at 3 er en **faktor** i 15 (fordi 3 multipliseret med 5 blir 15)

15 er **ikke delelig** med 4, fordi da er det **3 i rest**.



Oppgave 2: Skriv i «Skallet»

a) Hva får du i rest hvis du deler 45 på 6? Bruk modulo-operatoren (%-tegnet) til å sjekke.
b) Bruk modulo-operatoren til å finne tall som tall er delelige med 13. Er 39 delelig med 13? Er 113 delelig med 13?
c) Er 41 en faktor i 533? Dere kan også bruke deletegnet (/) for å sjekke.
d) Er 818 delelig med 8? Hvorfor / hvorfor ikke?
e) Er 13113 delelig med 13? Hvorfor / hvorfor ikke?

OBS! Hvis dere skal skrive **komma**, må dere bruke **punktum** i stedet i Python

Steg 3: Bruke modulo-operatoren (%-tegnet) og `range()` funksjonen i løkker for å lage tallrekker (f.eks. gangetabellen)

Eksempelet under viser en løkke som gjentar noe 31 ganger. Inni løkken er det en `if` setning som printer noe ***hvis*** (`if`) en *betingelse* stemmer.
Eksempel: `(tall % 3 == 0) betyr at` tallet delt på 3 gir 0 i rest.

Husk å bruke to likhetstegn for å sjekke om noe er likt

```
for tall in range(31):
    if tall % 3 == 0:
        print(tall)
```

Oppgave 3:

a) Skriv av koden ovenfor: hva skjer når dere kjører den? (Åpne et tomt program)
b) Lag et program som printer ut 4-gangen opp til 100.
c) Lag et program som finner alle tall mellom 50 og 100 som er delelig med 11.
d) Lag et program som finner alle tallene i 12 gangen mellom 120 og 240.
e) **Ekstra utfordring:** Lag et program som skriver ut denne tallrekken opp til 97: **2, 7, 12, 17, 22 ...**

Steg 4: Legge elementer til en liste

Det går an å lage tomme lister som fylles med f.eks. tall:

```
listen_min = []

for tall in range(11):
    if tall % 2 == 0:
        listen_min.append(tall)

print(listen_min)
```

Programmet bruker funksjonen `.append()` på listen for å legge til visse tall i `listen_min`. Til slutt printes listen ut.

OBS: Legg merke til at vi bruke firkant-parenteser i lister: `[ ]`

Oppgave 4:

a) Skriv av koden og kjør den: Hvilke tall blir lagt til i listen?
b) Lag en liste med alle tallene i 5-gangen opp til 50.
c) Klarer dere å forklare hvordan koden virker steg for steg?
d) **Ekstra utfordring:** Lag en liste med alle tallene i 13 gangen mellom 130 og 300.

Steg 5: Finne ut om et tall er en faktor i et annet tall

```
def er_faktor(a,b):
    if [redacted] % [redacted] == 0:
        print(a, " er en faktor i ", b)
    else:
        print(a, " er IKKE en faktor i ", b)

er_faktor(13,104)
```

Dette programmet inneholder funksjonen `er_faktor(a,b)` som avgjør om et tall, *a*, er en faktor i et annet tall, *b*.

Deler av koden er sjult...

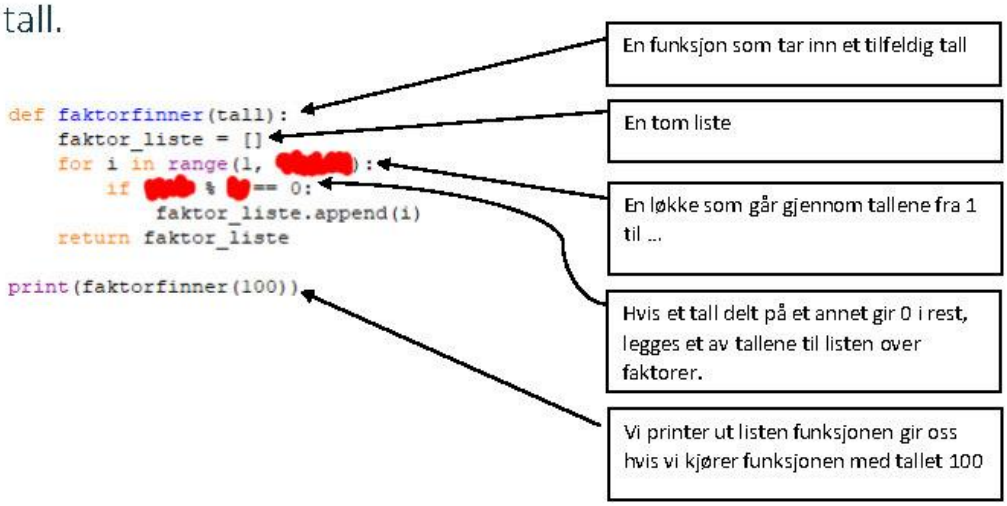
Oppgave:

a) Skriv av koden ovenfor: Hva må stå i `if`-setningen for at vi skal finne ut om *13* er en faktor i tallet 104?
b) Er 53 en faktor i 104?
c) Gjør om koden slik at funksjonen kun inneholder variabler (*a* og *b*), og **ingen** tall.

Hint: Funksjonen trenger to tall for å virke, *a* og *b*. Disse kan man regne med i funksjonen, i stedet for å bruke tall. Da slipper man å forandre alle tallene hver gang man vil prøve noe nytt.

d) Bruk programmet for å finne ut om 55 er en faktor i 30415?

Steg 6: Funksjon som lager en liste med faktorene til et tall.



Oppgave 6:

a) Skriv av koden ovenfor: Hva må stå i `range()`-funksjonen for at løkken går helt opp til tallet vi velger?

b) Hva må stå etter `if`-setningen slik at vi kun legger til faktorer av tallet i listen?

c) Prøv å kun bruke variabler i funksjonen (ikke tall), slik at vi slipper å skrive koden på ny hvis vi bytter ut tallet 100 med noe annet.

d) Hvilke faktorer har tallet 1889?

e) Hvilke faktorer har tallet 1200?

f) Prøv å forklare hvordan programmet fungerer, linje for linje.

g) **Ekstra:** Utvid programmet ved å lage en `for`-løkke ved å bruke funksjonen `faktorfinner()` slik at dere finner alle faktorene i alle tallene opp til 20.

```
def faktorfinner(tall):
    faktor_liste = []
    for i in range(1, tall+1):
        if tall % i == 0:
            faktor_liste.append(i)
    return faktor_liste

for i in range([redacted]):
    print(faktorfinner([redacted]))
```

Steg 7: Lag en funksjon som avgjør om tall er primtall eller ikke basert på hvor mange tall en liste inneholder (lengden på listen)

`Funksjonen len()` kan brukes til å finne lengden på en liste.

```
>>> min_liste = [4,3,50,28]
>>> print(len(min_liste))
4
```

Oppgave 7:

a) Lag noen lister med tall i skallet og prøv ut `len()` –funksjonen. Forklar hva funksjonen gjør.
b) Hvorfor kan dette være nyttig for å lage et program som finner primtall? *(Hint: Hva er definisjonen på et primtall?)*
c) Programmet nedenfor skal finne ut om et tall er primtall eller ikke. *X* er en variabel som inneholder lengden på en liste med faktorer til tallet vi velger.
d) Er 1697 et primtall?
e) Hva med 19934?
f) Og 131917?

```
def faktorfinner(tall):
    faktor_liste = []
    for i in range(1, tall+1):
        if tall % i == 0:
            faktor_liste.append(i)
    return faktor_liste

def er_primtall(tall):
    x = [redacted]

    print(faktorfinner(tall))
    if x == 2:
        print("... ")
    else:
        print("... ")

er_primtall(1697)
```

Hvordan finne antall faktorer i tallet?

Hva bør stå i disse to feltene?

Matematikk og programmering - Runde 2

Tallrekker og tabeller

I denne modulen skal dere utforske tall og tallrekker ved å bruke løkker i Python.

Dere kan når som helst spørre om hjelp hvis det er noe dere ikke forstår.

De som blir filmet kan velge å avbryte når som helst hvis de ønsker.

VIKTIG!

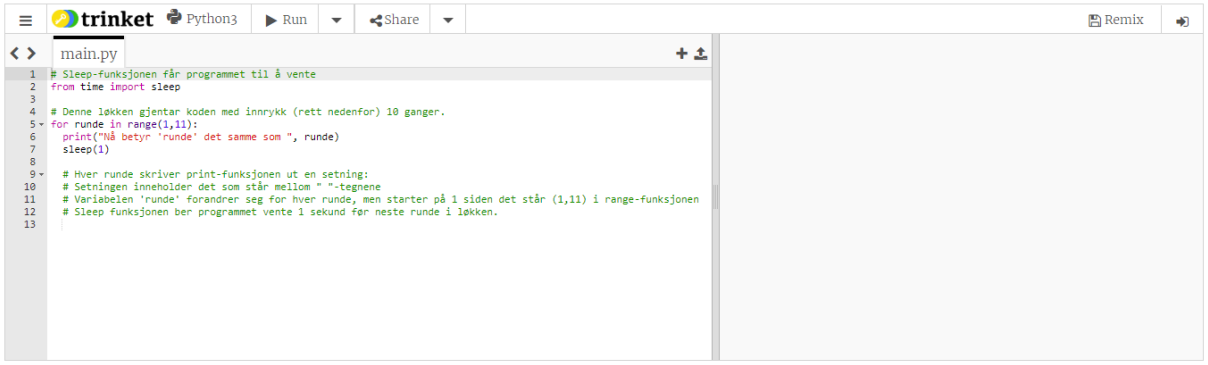
Når dere jobber med oppgavene, VENT ALLTID med å kjøre programmene til dere har lest oppgavene og gjort dere opp en mening om hvordan programmet virker.
Før dere kjører programmet skal dere alltid lese koden og forklare hva dere tror kommer til å skje, så detaljert som dere klarer
Hvis dere gjør forandringer i koden, forklar hvordan dere tror forandringen påvirker programmet før dere kjører koden på ny

Neste

Start 1 2 3 4 5 6 7 8

Steg 1 - for-løkken

1. Les koden og kommentarene bak `#`-tegnet. **Kommentarene påvirker ikke koden.**
2. Hva tror dere vises på skjermen når programmet kjøres?
3. Kjør programmet. Stemte det dere trodde?
4. Trykk **'Neste'** nederst på siden for å gå videre

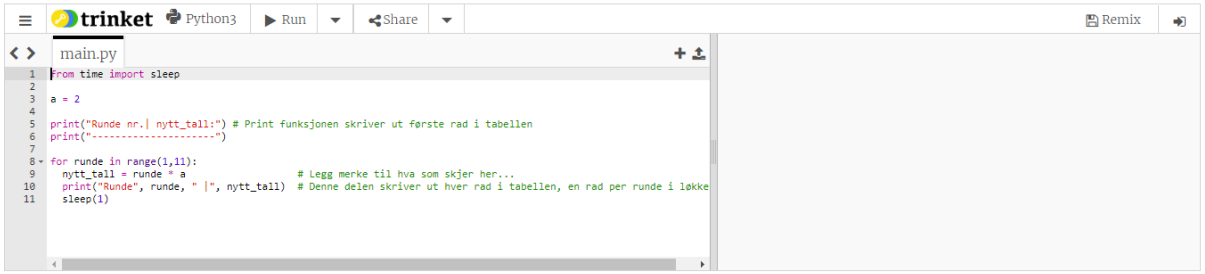


Forrige Neste

Start 1 2 3 4 5 6 7 8

Steg 2 - Variabler

1. I linje-9 lages en ny variabel `nytt_tall` som printes for hver runde i løkken.
   Hva tror dere `nytt_tall` blir den første runden? (når `runde = 1`).
   Kjør koden og observer. Stemte det dere trodde?
2. Endre til `a = 3` (i linje 3).
   Forklar hva dere tror skjer når koden kjøres (Husk at `runde = 1` den første runden, deretter 2, så 3 osv...).
   Kjør koden og observer. Stemte det dere trodde?
3. Prøv det samme med noen andre tall... (f.eks 11, 99, 1000?)



Forrige Neste

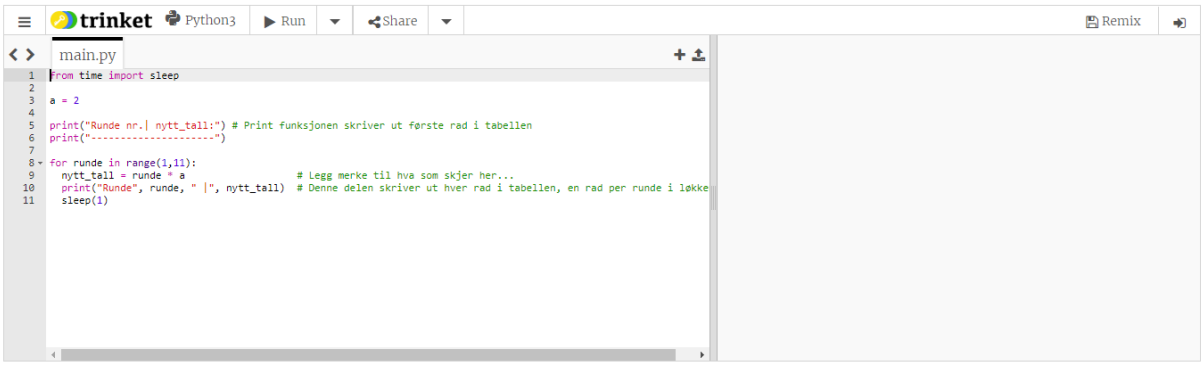
Start 1 2 3 4 5 6 7 8

Steg 3 - Forandre regnetegn

1. I linje 9 blir `nytt_tall` lagd ved å **gange** `runde` med `a`.
   Sett `a = 10` (linje-3) og prøv ut ulike regnetegn (`+ - * /`) i linje-9.
   Beskriv hva dere tror programmet skriver ut før dere kjører. Prøv med ulike regnetegn.

2. Sett `a = 2` (i linje 3) og endre linje-9 til: `nytt_tall = runde ** a`.
   Kjør programmet. Hva tror dere `**` betyr?
   Hva tror dere skjer hvis `a = 3`? Forklar hva dere tror skjer før dere endrer `a` til `3` og kjører programmet.

3. Forandre koden i linje 9 slik at dere lager deres egen tallrekke.
   Forklar hva dere tror `nytt_tall` blir for hver runde FØR dere kjører koden.



Forrige Neste

Start 1 2 3 4 5 6 7 8

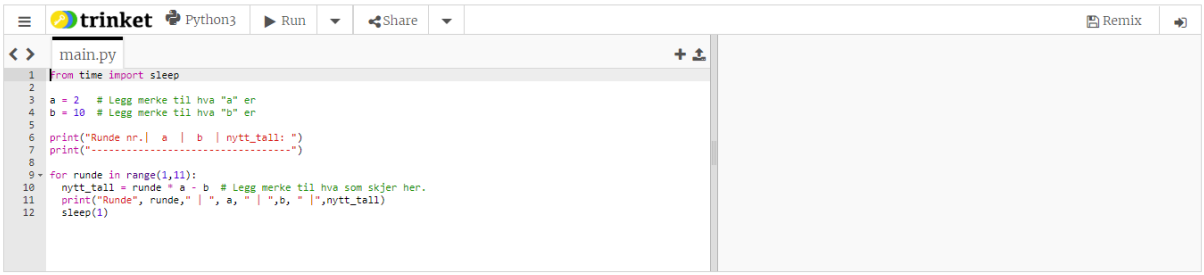
Steg 4 - Tabell

1. Hva blir `nytt_tall` den første runden i løkken, når `runde = 1`)? Se linje 3,4 og 10.
   Hva blir `nytt_tall` videre når løkken fortsetter?
   Forklar hva dere tror skjer før dere kjører programmet. Stemte det dere trodde?

2. Forandre koden i linje 10. Lag deres egen måte å regne med `a`, `b` og `runde` på slik at `nytt_tall` får ny verdi.
   Prøv med ulike regnetegn: `+ - / * % //` eller `**`.
   Beskriv hva dere tror utfallet blir runde for runde før koden kjøres.

3. **Ekstra utfordringer:**
   Bruk `a`, `b` og `runde` i linje 10 for å lage disse tallrekkene.
   - 3, 6, 9, 12, 15 ...
   - -7, -4, -1, 2, 5, 8 ...
   - 0, 0.25, 0.5, 0.75, 1.0 ...
   - 0, 1, 4, 9, 16, 25 ...

   PS! Dere trenger ikke bruke alle variablene samtidig



Forrige Neste

Start 1 2 3 4 5 6 7 8

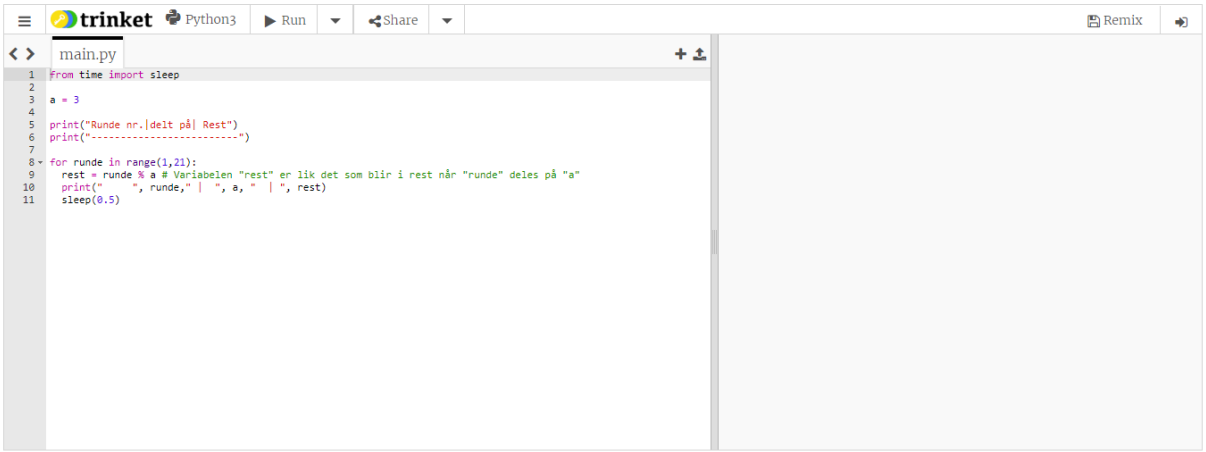
Steg 5 - Divisjon og rest

1. Hvilke verdier vil `rest` (linje 9) få når `runde = 1, runde = 2` osv. Forklar hva dere tror skjer før dere kjører koden.

2. Forandre verdien til `a`. Prøv med tall mellom 4 og 10.
   Forklar hva dere tror vil vises i tabellen, deretter kjør koden.

3. Hvilke tall mellom 150 og 200 er delelige med 33? Hvordan kan dere bruke programmet til å sjekke dette?
   Prøv å regne i hodet først og sjekk med programmet etterpå.
   Hvordan ser dere om tallet er delelig med 33? Bruk kalkulator til å sjekke om dere er i tvil.

4. Hvilke tall mellom 500 og 600 er delelige med 44? Regn ut i hodet først og bruk programmet til å sjekke etterpå
   PS! Forandre verdien i sleep-funksjonen til et lavere tall hvis det går for tregt (f.eks 0.1 eller 0.01)




```
from time import sleep

a = 3

print("Runde nr.|delt på| Rest")
print("------------------------")

for runde in range(1,21):
  rest = runde % a # Variabelen "rest" er lik det som blir i rest når "runde" deles på "a"
  print("     ", runde," |  ", a, "  | ", rest)
  sleep(0.5)
```

Forrige Neste

Start 1 2 3 4 5 6 7 8

Steg 6 - Rest og felles multiplum

1. Dette programmet lager en tabell som ligner den forrige, men nå er det to tall som gir rest, `a` og `b`.
   Forklar hva dere tror innholdet i tabellen vil være før dere kjører koden.
   Kjør koden og observer. Forklar innholdet i tabellen så godt dere klarer.

2. Forandre verdien til `a` og `b`. Prøv ulike tall mellom 2 og 10 i første omgang.
   Beskriv hvordan dere tror tabellen blir før dere kjører programmet:

3. Hvordan kan man bruke dette programmet til å finne tall som er både i `a`-gangen og i `b`-gangen? (**felles multiplum**).

4. Bruk programmet til å finne noen tall som er både i 23-gangen og i 5-gangen.
   Hint: Dere kan forandre `a` og `b`. I tillegg kan dere bruke andre tall i `range`-funksjonen. Nå sjekker dere kun tall fra 1 til 20...


```
from time import sleep

a = 2   # Legg merke til hva "a" betyr
b = 4   # Legg merke til hva "b" betyr

print("Runde nr | rest_a | rest_b")
print("-------------------------")

for runde in range(1,21):
  rest_a = runde % a  # Gir rest når "runde" deles med "a"
  rest_b = runde % b  # Gir rest når "runde" deles med "b"

  print("   ", runde, "   |  ", rest_a, "   |  ", rest_b,) # For hver runde i løkken lages det en rad i t
  sleep(0.5)
```

Forrige Neste

Start 1 2 3 4 5 6 7 8

Steg 7 - Finne felles multiplum

1. **Dette programmet skal finne noen tall opp til 1000 som er både i 9-gangen og i 11-gangen.**
   Kjør programmet. Virker det slik det skal?

2. Forandre programmet slik at det finner tall som er både i 9-gangen og i 11-gangen.
   Hint: Dere kan forandre `a`, `b`. I tillegg kan dere forandre tallene i `range`-funksjonen

3. Les gjennom koden; Hva skjer i linje 10 og 11?

4. **Ekstra utfordring:** Kan dere legge til en variabel `c` og en variabel `rest_c` slik at dere finner tall som er i 3 forskjellige gangetabeller?
   Husk at dere også må forandre innholdet i `if`-setningen også...
   Finnes det tall mellom 1000 og 2000 som er både i 7-gangen, 8-gangen og i 9-gangen?
   Sjekk med kalkulator på et par av tallene for å være sikre på at programmet fungerer slik det skal...


```
1  from time import sleep
2
3  a = 2
4  b = 5
5
6  for runde in range(1,51):
7    rest_a = runde % a  # Gir rest når "runde" deles med "a"
8    rest_b = runde % b  # Gir rest når "runde" deles med "b"
9
10   if rest_a == 0 and rest_b == 0: # HVIS både "rest_a" er 0 OG "rest_b" er 0, så:
11     print(runde, "er både i", a, "gangen, og i", b, "gangen... ")
```

Forrige Neste

Start 1 2 3 4 5 6 7 8