«Dette kan vi jo bruke i matematikken!»

En studie av oppstarten med programmering i matematikkfaget.

av

Marita Orvedal

Erfaringsbasert master i undervisning med fordypning i matematikk

Matematisk institutt, Universitetet i Bergen

1.juni 2023

Forord

Programmering ble en del av læreplanen LK20 fra høsten 2020. Det ble ikke opprettet som et eget fag, men ble innlemmet i eksisterende fag som matematikk, naturfag, musikk og kunst og håndverk. Flere lærere, meg selv inkludert, hadde ikke noen bakgrunn med å bruke programmering i undervisningssammenheng før denne skolereformen. Læringskurven har derfor vært bratt for mange, og det vil den fortsatt være i årene som kommer. Flere har allerede kommet godt i gang med å innlemme programmering i sin undervisning, noe som tydelig kommer frem av de utallige gruppene i sosiale medier som drøfter og deler opplegg innen programmering i skolen. Å ta del i en delingskultur og lese om andres erfaringer kan være verdifullt. Det er også blitt opprettet ulike videreutdanninger hvor både lærere og andre kan fordype seg mer i programmering.

Da jeg skulle begynne på en erfaringsbasert master med fordypning i matematikk ønsket jeg å fordype meg i et tema innenfor programmering i matematikkfaget. Dette ble valgt fordi jeg så et behov for å lære mer om programmering og hvordan jeg kan innlemme det i matematikkundervisningen på en god måte. Jeg hadde ingen erfaring med bruk av programmering i undervisningssammenheng fra før, og det ble naturlig for meg å gjennomføre dette prosjektet som en aksjonsforskning.

Det har vært svært lærerikt for meg å jobbe innenfor det valgte temaet og det har også vært berikende å delta på innholdsrike og spennende masterseminarer.

Det har vært krevende å være i masterutdanning samtidig som jeg har vært i jobb og opplevd flere alvorlige sykdomstilfeller i nær familie. Jeg vil rette en stor takk til min veileder Johann Christoph Kirfel, som har gitt meg konstruktive innspill gjennom hele prosessen. Videre er jeg takknemlig for å ha gode kolleger ved min arbeidsplass som har vært forståelsesfulle for mitt fravær ved studiesamlinger og lesedager, og en arbeidsgiver som har lagt til rette for at jeg kunne kombinere studiet med jobb. En takk rettes også til elevene som deltok i studien. Uten deres deltakelse ville ikke denne studien vært mulig å gjennomføre. Jeg er glad for at jeg fikk tid og mulighet til å gjennomføre studiet og er takknemlig for all støtte underveis.

Sammendrag

Denne masteroppgaven presenterer en studie som ser på en 8. klasse som har startet opp med programmering i matematikkfaget. Etter å ha kartlagt elevenes erfaringer, holdninger og forventninger til programmering, vil oppgaven vise ulike utfordringer som kan oppstå når elevene møter variabler i programmeringssammenheng.

Studien er gjennomført som en aksjonsforskning der jeg har besøkt en 8. klasse, og elevene har deltatt i åtte undervisningsøkter i programmering. Elevene har deltatt i spørreundersøkelser, levert logger og det er blitt tatt skjerm- og lydopptak i flere av undervisningsøktene som de deltok i. Til slutt har elevene besvart en avsluttende diagnostisk kartleggingstest med fokus på variabelbruk i programmeringen.

Gjennom aksjonsforskningen har jeg kommet frem til at det finnes mange ulike vansker og misoppfatninger som kan knyttes til variabelbruk innen programmering. I denne oppgaven vil jeg presentere teori og observasjoner som viser hvordan vanskene som oppstår innen syntaktisk, konseptuell og strategisk kunnskap i programmeringen kan komme til syne. Noen av vanskene kan oppstå på grunn av ikke-kongruente konverteringer mellom naturlig tale- og skriftspråk, algebraiske uttrykk og programmeringsspråk. At variabelbegrepet forstås noe ulikt i matematikk og i programmering kan også skape noen vanskeligheter. I denne oppgaven er det et spesielt fokus på ni utvalgte misoppfatninger som er knyttet til variabelbruk i Scratch, hentet fra Sorvas oversikt med 162 misoppfatninger (Sorva, 2012, s.358-368). I tillegg fremkommer det i datamaterialet åtte andre typer vansker som ligger utenfor det valgte rammeverket, og disse vil bli presentert og kategorisert som T1-T8.

Det viser seg at eventuelle misoppfatninger og andre vansker som elever kan ha ofte er vanskelige å identifisere og rette opp. Det er derfor verdifullt at lærere får kunnskap innen dette området, slik at man kan planlegge og tilrettelegge for undervisningsøkter som kan hjelpe elevene med å overvinne barrierene som skaper vanskene. Gjennom erfaringene fra aksjonsforskningen har jeg kommet frem til noen konkrete grep som matematikklærere kan bruke i undervisningen for å støtte læringsprosessen til nybegynnere innen programmering. Avslutningsvis har jeg bygget på disse erfaringene og gjort noen pedagogiske grep og endringer i undervisningsopplegget som ble gjennomført i denne studien. Jeg håper at grepene kan være til støtte for lærere med å hjelpe sine elever til å overvinne eventuelle barrierer som skaper vansker i programmeringen, og at dette kan hjelpe elevene med å forstå variablenes roller i programkoder. Å forstå variablenes roller er spesielt viktig for å kunne generalisere en programkode til å kunne løse en hel klasse av problemer.

Innholdsfortegnelse

Figurliste

1 Innledning

I dette kapittelet vil jeg presentere bakgrunn og valg av tema. Jeg vil vise til tidligere forsøk på å iverksette programmering i skolen, beskrive hvordan andre land har innført programmering i både matematikkfaget og som eget fag, og forklare hvordan Norges nye skolereform LK20 vektlegger programmering i skolen. Videre kommer jeg til å vise til noen resultater fra TIMMS som handler om norske elevers kompetanse innen algebra, og jeg avslutter kapittelet med å presentere problemstillingen og forskningsspørsmålene for masteroppgaven.

1.1 Bakgrunn og valg av tema

1.1.1 Innføring av programmering i skolen

Å bringe programmering inn i skolesammenheng er ikke en ny tanke, da det allerede i 1960 var planer om at skolebarn skulle ta i bruk programmering. Noen skoler introduserte LOGO på 1980-tallet, men det ble faset ut etter bare en kort tid. (Popat & Starkey, 2018, s.365). Selv om viljen til å innføre programmering i skolen var til stede kan det virke som tidspunktet ikke var riktig. Det var kanskje mangel på både utstyr og lærere med programmeringsbakgrunn. Ellers kan man peke på at «digital technology had not become a natural part of people's daily lives as it has today» (Bråting & Kilhamn, 2021, s. 170).

Utfordringene i dag kan fremdeles være mangel på programmeringskompetanse blant lærere, men det er opprettet mange kurs og videreutdanninger som gjør at man kan få faglig påfyll innen programmering i undervisningen. Jeg ser også at det er kommet mange gode læringsressurser som man kan velge å benytte seg av, både i lærebøker og på nett, slik at man ikke begynner på bar bakke. Det som imidlertid kan være en utfordring er at ulike skoler bruker ulike digitale verktøy og programmeringsspråk. Det kan virke som om det er opp til hver enkelt å finne ut hvilket programmeringsspråk man vil starte med, og hver enkelt må også i stor grad løse tekniske problemer som oppstår underveis. Det kan være bagateller som ulike tastetrykk på en Chromebook og en PC som skaper problemer, eller at valgt programmeringsspråk ikke er kompatibelt med enheten som skolen bruker. Da jeg var på besøk hos en valgfagsklasse i programmering skoleåret 2020/21 ble jeg presentert for disse ulike utfordringene. Noen av elevene hadde valgt å ta med sin egen private PC til valgfagstimene, siden skolens Chromebook hadde flere begrensninger som gjorde at det ikke gav dem like mange muligheter når de skulle jobbe med programmering.

Programmering har vært i bruk i grunnskolen i Estland, Hellas, England og Australia siden 2010. (Popat & Starkey, 2018, s.365). I England har programmering blitt opprettet som et eget fag «computing», mens i Finland og Sverige har programmeringen blitt innlemmet i eksisterende fag som matematikk. (Bråting & Kilhamn, 2021, s.170). Norge har også landet på sistnevnte løsning. Vi burde se til disse landene som har erfaring på området, og bygge videre på deres lærdom. Det er verdifullt å se på hva som har fungert bra og hva som ikke har fungert like godt ved innføring av programmering i disse landene. Har de brukt blokkbasert eller tekstbasert programmeringsspråk i oppstarten? Har de hatt programmering som et eget fag eller innlemmet det i matematikkfaget? Har de noen erfaringer knyttet til elevers misoppfatninger og andre vansker innen programmering? Har de funnet at bruk av variabler i programmering styrker eller svekker elevers forståelse for variabelbruk innen matematikk?

Det å se til andres erfaringer kan være nyttig for oss som er i en startfase sammenlignet med mange av de landene som er nevnt.

1.1.2 Programmering i læreplanen

Med skolereformen LK20 kom programmering for alvor inn i skolen og ble en del av flere fag. Matematikk er et fag hvor programmering passer naturlig inn, men også i naturfag, kunst og håndverk og andre fag vil det være flere gode muligheter for å anvende programmering. I matematikk er programmering kommet inn som en grunnleggende ferdighet som skal brukes til å utforske og løse matematiske problemer. (Utdanningsdirektoratet, 2020). Om man leser videre på kompetansemålene for hvert trinn oppdager man at ordet programmering ikke dukker opp før på 5. trinn. Det vil si at i de første skoleårene kan man jobbe med algoritmisk tenking og lage algoritmer uten å ta i bruk et programmeringsspråk. De neste utdanningsårene blir det mer spesifikt i læreplanen hva man skal bruke programmeringen til, selv om målene er veldig runde og åpne for tolkning. På 10. trinn er kompetansemålet å "utforske matematiske egenskaper og sammenhenger ved å bruke programmering" nevnt. Likevel vil det være mulig å forsøke å utforske og løse matematiske problemer på det nivået man er gjennom hele utdanningsløpet, og ikke vente helt til 10. trinn før man opplever denne enorme muligheten programmering kan åpne for oss.

Slik jeg ser det vil arbeid med programmering i matematikkfaget kunne heve kompetansen innenfor alle de 6 kjerneelementene i læreplanen for matematikk. De 6 kjerneelementene er inndelt som dette i læreplanen: 1) utforsking og problemløsing, 2) modellering og anvendelser, 3) resonnering og argumentasjon, 4) representasjon og kommunikasjon, 5) abstraksjon og generalisering og 6) matematiske kunnskapsområder. (Utdanningsdirektoratet, 2020). Som matematikklærer vil jeg naturligvis foretrekke at programmeringen som skal foregå i matematikktimene skal være relevant for faget og gå hånd i hånd med de matematiske kunnskapsområdene fremfor å komme som et eget "kurs" som oppleves adskilt fra resten av matematikkundervisningen. Om man opplever det sistnevnte kunne man like gjerne ha opprettet programmering som et eget fag. I dette forskningsprosjektet vil jeg derfor i stor grad tilstrebe å holde et blikk på matematiske begreper som vinkler, mangekanter, tallrekker, omkrets og areal i arbeidsøktene med programmeringen. Samtidig som disse matematiske kunnskapsområdene vil bli vektlagt i oppgavene, legges det matematiske innholdet på et nivå som forventes å være kjent for elevene. Slik vil hovedfokuset for elevene på dette tidspunktet bli å lære seg programmeringsspråket og kunne bruke dette i arbeidet i tråd med de 5 første kjerneelementene.

1.1.3 TIMMS – norske elevers algebrakunnskaper

Da dette prosjektet i stor grad kommer til å handle om bruk av variabler i programmeringen kan det være aktuelt å kikke litt på hvordan det står til med norske elevers algebrakunnskaper. Dette er ikke valgt fordi det alltid er en sterk kobling mellom algebra og programmering, men fordi notasjonene som brukes i de to ulike systemene ofte kan ligne på hverandre, men likevel kan de gjerne ha ulik betydning. Da det ligger utenfor rammene til dette prosjektet å kartlegge norske elevers algebrakunnskaper viser jeg til resultater som er avdekket ved TIMMS. «TIMSS er en internasjonal undersøkelse, som måler elevers kompetanse i matematikk og naturfag på 5. og 9. trinn. Gjennom spørreskjemaer samles det i tillegg inn relevant informasjon om elevene, lærerne og skolene som brukes i analysene av datamaterialet.» (Utdanningsdirektoratet, 2022). I de fleste land er det elever fra 4. og 8. trinn som deltar,

mens Norge deltar med 5. og 9. trinn for at elevene skal være på alder med de andre elevene som deltar.

Et av funnene fra TIMSS 2011 er beskrevet slik: «Resultatet for de norske elevene på området Algebra utmerker seg internasjonalt som spesielt svakt. Av de landene som deltok på 8. trinn i 2011, var det bare typiske utviklingsland, med en helt annen ressurssituasjon enn Norge, som lå på eller i underkant av det norske nivået på dette emneområdet.» (Grønmo et al, 2012, s.25). Også i TIMSS 2015 er algebra trukket frem som et område Norge ikke scorer særlig høyt på: «På 9. trinn kan norske elevers prestasjoner i matematikk karakteriseres som middels gode i et europeisk perspektiv. Det er særlig svake prestasjoner i emneområdet Algebra, som her trekker gjennomsnittsscoren ned.» (Bergem et al, 2016, s.22). Også TIMSS-rapporten fra 2019 viser at norske elever presterer lavere enn Sverige og Finland når det gjelder algebra. (Kaarstein et al, 2020)

Det er med dette ikke sagt at lave prestasjoner innen algebra betyr at norske elever har et dårligere grunnlag for å drive med programmering enn andre land som scorer høyere, men det er vanskelig å se at det kan være noen fordel for de norske elevene å ha lave prestasjoner innen algebra. Det er uansett greit å være bevisst over denne trenden, slik at man kan være forberedt på de ulike vanskene dette kan medføre, særlig når man skal kombinere algebra med programmering.

1.2 Problemstilling og forskningsspørsmål

Hensikten med denne studien er at jeg gjennom aksjonsforskning skal få erfaring med å bruke programmering i matematikkundervisningen, og at jeg skal lære mer om hvilke misoppfatninger og andre vansker elever kan møte i programmeringen. Siden programmering i undervisningssammenheng er relativt nytt for min egen del synes jeg at denne vinklingen er både lærerik og svært nyttig for min egen undervisningspraksis.

Flere har påpekt viktigheten av å ha kunnskap og bevissthet knyttet til misoppfatninger og andre vansker som kan oppstå i programmering. Qian og Lehman (2017) har forklart at "Raising awareness of students' misconceptions and other difficulties, and strategies for addressing them, can help computer science teachers to better teach their students." (Qian & Lehman, 2017, s.2). Clancy et al (2001) er også blant dem som har påpekt nytten av dette og påstår at: «Identification of common misconceptions can guide teachers to help students counter or avoid them.» (Clancy et al, 2001, s. 324).

Med dette som motivasjon ønsker jeg å undersøke hvilke misoppfatninger og andre vansker som kan oppstå når elever møter variabler i programmeringssammenheng. Jeg velger å holde fokuset på bruk av variabler, for jeg ser for meg at å undersøke absolutt alle vansker på en gang kan bli for omfattende. Før jeg påbegynner datainnsamlinger knyttet til dette valgte temaet er det viktig for meg å få noe kjennskap til hvilke erfaringer, holdninger og forventninger elevene som deltar i prosjektet har til programmering fra før. Avslutningsvis i prosjektet vil jeg, om mulig, presentere noen forslag til pedagogiske grep som kanskje kan hjelpe elevene med å bryte barrierene som skapte vanskene innen programmeringen. Siden programmering var relativt nytt for både elevene og meg selv før oppstarten av dette prosjektet har jeg landet på følgende tema og problemstilling for masteroppgaven:

Oppstart med programmering i matematikkfaget.

For å få kjennskap til elevenes bakgrunn, vansker som kan oppstå og mulige pedagogiske grep som kan følges i et klasserom, har jeg formulert tre forskningsspørsmål som vil være selve grunnlaget for masteroppgaven. De tre forskningsspørsmålene er formulert slik:

1) Hvilke erfaringer, holdninger og forventninger har elever på 8. trinn til programmering?
2) Hvilke misoppfatninger og andre vansker kan elever på 8. trinn oppleve i møte med variabler i programmering?
3) Hvilke pedagogiske grep kan matematikklærere gjøre i undervisningen for å hjelpe elevene med å overvinne misoppfatninger og andre vansker knyttet til variabelbruk i programmering?

1.3 Oppgavens oppbygging

I dette kapittelet har jeg forklart bakgrunn og valg av tema samt presentert oppgavens problemstilling og forskningsspørsmål. I neste kapittel vil jeg belyse relevant teori for oppgaven, og videre vil studiens metode presenteres. I kapittel 4 besvares første forskningsspørsmål gjennom analysen av spørreundersøkelsene, mens forskningsspørsmål to belyses gjennom transkripsjoner og den avsluttende kartleggingstesten. Etter å ha drøftet noen generelle aspekter i kapittel 5, vil jeg besvare det siste forskningsspørsmålet i kapittel 6. De tre forskningsspørsmålene vil dermed bli belyst i ulike deler av masteroppgaven, og har til hensikt å til sammen belyse oppgavens tema og problemstilling: Oppstart med programmering i matematikkfaget.

2 Teori

I dette kapittelet vil jeg presentere teori som har vært relevant for dette prosjektet. For å belyse problemstillingen og forskningsspørsmålene vil jeg starte med å beskrive begrepene programmering og algoritmisk tenkning, og deretter begrunne valget av programmeringsspråket i dette prosjektet. Jeg vil vise til teori som beskriver hvilke aktiviteter man kan gjøre med nybegynnere i programmering, og i hvilken rekkefølge det er fornuftig å gjennomføre aktivitetene. Videre belyses teori knyttet til variabelbruk innen matematikk og programmering. Neste del består av en gjennomgang av hvordan litteraturen jeg har lest presenterer ulike typer kunnskaper innen programmering, og hva som kan kategoriseres som misoppfatninger og andre vansker innen programmering. Kapittelet avsluttes med en gjennomgang av de misoppfatningene som jeg kommer til å holde fokus på i denne oppgaven.

2.1 Oppstart med programmering for nybegynnere

2.1.1 Hva er programmering

Store norske leksikon definerer programmering som "utformingen av et dataprogram som avgjør hvordan en datamaskin, en robot, en mobiltelefon, en vaskemaskin eller annet elektronisk apparatur skal fungere mens programmet er aktivt eller kjører." (Store norske leksikon, 2023). Å utforme et dataprogram innebærer mye mer enn selve kodingen, som er selve reformuleringen inn i et programmeringsspråk. Programmering som arbeidsprosess består av "forarbeid, planlegge, skrive, tegne, bearbeide, undersøke om problemet er løst, eksperimentere, revurdere, argumentere for at algoritmen alltid virker, generalisere algoritmen til å løse en hel klasse av problemer, sjekke effektivitet osv." (Gjøvik, 2019, s.34)

Er det også mulig å drive med programmering uten å utforme et dataprogram? Om vi tenker oss at det er et menneske som utfører instruksjonene, kan dette også defineres som programmering? Da kan begrepet programmering inkludere det å følge og gjennomføre en matoppskrift, å utføre en bestemt koreografi, å utføre gjetteleker med tall osv. "En trenger likevel ikke tenke seg at programmering nødvendigvis skal skje på en datamaskin med et nytt språk eller notasjonssystem" (Gjøvik, 2019, s.34). Gjøvik (ibid) beskriver videre en "analog" programmeringsøkt med sorteringsalgoritmer, og argumenterer for at en slik økt kan høre hjemme i både matematikk og programmering. Til tross for at selve kodingen ikke finner sted, dekker man i en slik økt mange av de andre prosessene som kreves for å kunne drive med programmering. Man kan på denne måten jobbe med problemløsning og lage algoritmer som virker på en hel klasse av problemer før man tar i bruk et programmeringsspråk.

2.1.2 Hva er algoritmisk tenkning

"Algoritmisk tenkning er innført på norsk som det samme som Computational Thinking" (Gjøvik, 2019, s.32). At det ikke er en direkte oversettelse mellom engelsk og norsk kan skape forvirring knyttet til begrepet. En mulig forklaring kan være: "Computational thinking is regarded as a thought process entailed in designing solutions that can be executed by a computer, a human, or a combination of both" (Bocconi & Chioccariello, 2018, s.7). En annen formulering for begrepet er gitt slik: "Computational thinking is taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing" (Wing, 2008, s.3717). Gjøvik (2019) beskriver

innholdet i algoritmisk tenkning og kategoriserer det i fem ulike underkategorier: abstraksjon, algoritmebehandling, generalisering, automatisering og dekomponering. Hver underkategori beskrives slik:

1) Abstraksjon handler om å kunne trekke ut essensen av flere eksempler eller tilfeller og se bort fra irrelevante opplysninger.
2) Algoritmebehandling, som vi ser er en underkategori av algoritmisk tenkning, samsvarer med det engelske begrepet Algoritmic Thinking. Dette handler om å kunne forklare og følge stegvise instrukser.
3) Generalisering er å kunne gjenkjenne mønstre og sammenhenger og lage allmenne regler og metoder som fungerer på en hel klasse av eksempler.
4) Automatisering er å kunne implementere løsningen av problemer i programmeringsspråk eller å gjøre det menneskelige bidraget minimalt.
5) Dekomponering er å kunne bryte opp et problem i mindre bestanddeler og håndtere hovedproblemet i mindre biter. (Gjøvik, 2019, s.33)

Wing (2008) beskriver at essensen i algoritmisk tenkning er abstraksjon, og at abstraksjon i programmeringssammenheng er rikere og mer kompleks enn abstraksjon som skjer i matematikk og fysiske vitenskaper. Videre omtaler hun databehandling (computing) som en aktivitet hvor vi automatiserer våre abstraksjoner, men vi behøver ikke nødvendigvis en datamaskin for dette. "A computer could be a machine, but more subtly it could be a human. Humans process information; humans compute. In other words, computational thinking does not require a machine" (Wing, 2008, s.3719). Når man tar i bruk et programmeringsspråk, må man i tillegg lære seg selve programmeringsspråket. "Our field of computing is in a unique situation since not only are there computational concepts to teach but also there is a tool to teach" (Wing, 2008, s.3721). Det er viktig å lære seg å forstå det valgte programmeringsspråket, men også begreper og logikk som ligger til grunn. Wing (2008) foreslår å bruke programmeringsspråket som et verktøy for å styrke forståelsen for begreper og algoritmisk tenkning. Et av begrepene som er nødvendig for å kunne løse en hel klasse av problemer er begrepet variabel, og i kap. 2.2 kommer jeg tilbake til dette begrepet. "In order to create an algorithm to solve a general mathematical problem, we need to create and use variables." (Bråting & Kilhamn, 2021, s.176).

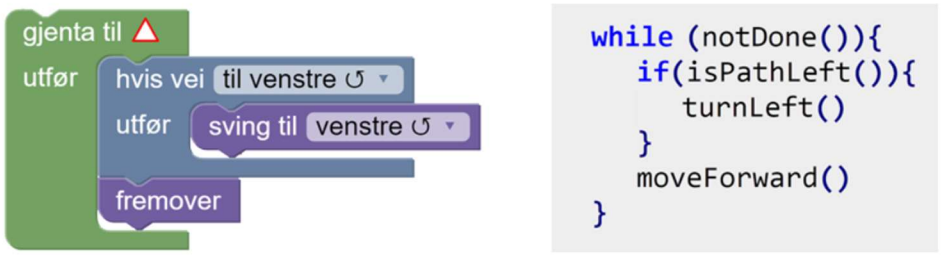
2.1.2 Valg av programmeringsspråk

Da jeg skulle velge mellom blokkbasert og tekstbasert programmeringsspråk for dette prosjektet var jeg lenge i tvil om hva som ville passe best for den valgte elevgruppen. I videreutdanningstilbud for lærere og andre kurs som finnes på nettet virker det tekstbaserte Python som mest brukt. I samtale med lærere som har undervist i valgfaget programmering i noen år blir blokkbasert programmering omtalt som barnslig og med begrensede muligheter. Likevel landet jeg på å bruke Scratch som programmeringsspråk i mitt prosjekt, og vil i det følgende forklare hvorfor.

Læreverk

Det heldigitale læreverket Campus Inkrement har valgt å bruke blokkbasert programmeringsspråk i sine leksjoner for ungdomstrinnet. Dette gjør at mange elever vil ha tilgang på videoleksjoner og annet materiell knyttet til et blokkbasert programmeringsspråk i

sitt læreverk. Nedenfor vises et eksempel på et program skrevet i både blokkbasert og tekstbasert programmeringsspråk.



Figur 1: Blokkbasert vs tekstbasert programmeringsspråk (Inkrement as, 2023)

Ut ifra hva som vises på figur 1 og forklaringen til Inkrement (Inkrement as, 2023), kan det tyde på at det kan oppleves lettere for en nybegynner innen programmering å både lese, forstå og produsere egne koder dersom de er blokkbasert. I blokkbasert programmering, slik som Scratch, vil man finne de ulike kommandoene i en logisk og lettlest meny. Kommandoene er sortert på en oversiktlig måte med fargekoder som skal hjelpe brukeren til å orientere seg riktig frem ut ifra hvilken kommando en søker etter. Brukeren kan enkelt lete etter og finne kommandoene for å dra de inn i arbeidsområdet. Skrivefeil vil ikke være mulig, og koder som ikke passer sammen vil ikke være mulig å kombinere. Likevel er det lett å skrive et program som ikke gir ønsket resultat. Som regel vil man oppdage dette under kjøring av programmet, og deretter vil man gjerne forsøke å gjøre korrigeringer slik at programmet oppfører seg slik man ønsker.

Blokkbasert programmering er av nevnte grunner mye enklere å starte opp med og har lavere brukerterskel enn å bruke tekstbasert programmering som f.eks. Python. I tekstbasert programmering må man i større grad huske de ulike kommandoene selv, skrive de inn uten skrivefeil og kombinere de på riktig vis. Siden jeg ikke hadde noen forventning til at deltakerne i dette prosjektet skulle ha noen særlige forkunnskaper innen programmering, var det naturlig for meg å starte med blokkbasert programmering på dette tidspunktet. Jeg ser for meg at det vil være enklere å begynne med tekstbasert programmering på ungdomstrinnet i de neste årene fremover, for da vil forhåpentligvis flere av elevene som begynner i 8. klasse ha jobbet med programmering på barneskolen. Man kan si at i Scratch lærer elevene grunnleggende ferdigheter innen algoritmisk tenking, de blir kjent med hvilke kommandoer som finnes og hvordan de kan kombineres, og de kan ganske raskt komme i gang med å bygge opp sine egne programmer. Dette vil kunne bidra til mestringsfølelse og glede blant elevene og nysgjerrighet for å lære mer om programmering senere.

<u>Mestring</u>

Etter å ha jobbet i ungdomsskolen i mange år, har jeg erfart at mange elever har en lav terskel for å gi opp dersom de møter noe som virker vanskelig. Jeg ser også at det i de aller fleste klasser finnes elever som har lese- og skrivevansker. Disse erfaringene støtter opp under idéen om å starte med Scratch på nåværende tidspunkt. Brukerterskelen er lav, og jeg er ganske trygg på at alle vil kunne oppleve mestring fremfor motgang i startfasen. Jeg vil forsøke å gi oppgaver som er av en art som legger til rette for utforskning og sikrer at alle får til noe. Jeg vil forsøke å unngå å gi ensporede oppgaver som mange opplever for rigide og for vanskelige til å klare. Jeg ser for meg at programmering i noen tilfeller kanskje kan være en

ny motivasjon for elever som er litt skoletrøtte i utgangspunktet. Det er ikke nødvendigvis de som har vist gode resultater tidligere som blir best i programmering. Av de ulike grunnene jeg har nevnt landet jeg på å bruke Scratch som programmeringsspråk for mitt forskningsprosjekt.

2.1.3 Å lese kode og å skrive kode

Schulte og Busjahn (2013, s.3) påpeker viktigheten av å bygge opp programmeringsferdighetene i følgende rekkefølge: «A casual relationship between code reading and writing skills would require to focus on reading and understanding of code before writing it.» (Schulte & Busjahn, 2013, s.3). Dette betyr ikke at eleven nødvendigvis må lese mange lange og kompliserte programmer før hen kan gå i gang å programmere selv, men hen må vite om hvilke kommandoer som finnes og hva kommandoene gjør før hen kan bruke de til noe nyttig selv. I programmeringen vil det være rom for å utforske de ulike kommandoene samtidig som de introduseres. Eleven vil altså lese/finne en kommando fra menyen i Scratch, prøve den og finne ut hva den gjør. Senere kan eleven bruke kommandoen i andre sammenhenger. Lesing og skriving av kode går dermed som hånd i hånd i en oppadgående spiral.

Schulte og Busjahn (2013) er ikke de eneste som har påpekt rekkefølgen og sammenhengen mellom de ulike programmeringsferdighetene. Lister et al (2009) har skrevet om sammenhengene mellom ulike ferdigheter. De konkluderer med følgende: “We found that students who cannot trace code usually cannot explain code, and also that students who tend to perform reasonably well at code writing tasks have also usually acquired the ability to both trace code and explain code.” (Lister et al, 2009, s.161).

Med bakgrunn i dette vil jeg bygge opp aktivitetene i en rekkefølge som gir elevene mulighet til å følge en progresjon slik som det er anbefalt i de to nevnte artiklene. I vedlegg D beskrives undervisningsopplegget som er gjennomført i klassen.

2.1.4 PRIMM som arbeidsmetode

Nybegynnere innen programmering kan møte en barriere når de får i oppgave å skrive egne programmer før de er i stand til å lese programmer. (Sentance et al, 2019, s. 137). For å gjøre oppstarten og arbeidet med programmering på en god måte utarbeidet Sentance og Waite i 2017 et forslag til en arbeidsmetode som mange har hatt gode erfaringer med å bruke innen programmering. Arbeidsmetoden forkortes med PRIMM, som står for predict, run, investigate, modify, make. (Sentance & Waite, 2017). Ved å følge denne metoden vil man legge til rette for flere aktiviteter enn bare å lese og skrive kode, noe som kan bidra til økt læring. Stegene i PRIMM-modellen er nærmere beskrevet slik:

P=Predict: Beskrives som en aktivitet hvor elevene får utlevert et ferdig program og hvor de leser dette og antar hvordan programmet oppfører seg. De kan f.eks. tegne eller skrive hva de mener at programmet vil gjøre. R=Run: Run blir en naturlig fortsettelse av predict, hvor man i run skal kjøre det gitte programmet og teste om det oppførte seg slik som en hadde antatt eller om det oppfører seg på en annen måte. Resultatet kan deretter diskuteres i klassen. I=Investigate: I dette steget er det lagt vekt på å utforske programmet nærmere. Dette kan skje gjennom ulike aktiviteter som læreren legger til rette for. Det kan være f.eks. å spore verdien til en variabel gjennom hele kjøringen av programmet, forklare programmet med egne ord, å kommentere hver eneste linje eller blokk i programmet eller å feilsøke et program

som inneholder en feil eller unødvendige koder. (Matematikksenteret, 2023). Legg merke til at elevene ikke har skrevet noen koder selv i disse tre første stegene. M=Modify: Dette steget i prosessen handler om at elevene skal gjøre endringer i eksisterende kode og sjekke hvordan dette påvirker programmet. I dette steget øker elevens eierskap til programmet, siden de går aktivt inn og endrer koden i programmet. M=Make: Dette er det siste steget i PRIMM, og går ut på at elevene skal kunne lage et nytt program som gjerne ligner på det som de har jobbet med i de foregående stegene, men som likevel løser et nytt problem. (Sentance et al, 2019, s, 148).

Når en ser på hvordan Schulte og Busjahn (2013) og Lister et al (2009) beskriver sammenhengen mellom å kunne lese kode og det å kunne skrive egne koder i lys av arbeidsmetoden PRIMM, ser en at det å lese og forstå kode er første steget i PRIMM, mens å skrive sine egne koder er det siste steget i PRIMM. Ved å følge PRIMM-modellen i undervisningen vil man i tillegg til å lese og skrive kode også arbeide med aktiviteter som legger til rette for utforsking av eksisterende koder. Gjennom disse aktivitetene kan man kanskje hjelpe elevene bedre til å komme fra å lese kode til å skrive egne programmer.

2.2 Hva er en variabel?

I dette forskningsprosjektet vil jeg undersøke hvilke misoppfatninger og andre vansker elever kan oppleve når de møter variabler i programmering. I den sammenheng er det hensiktsmessig å se nærmere på selve variabelbegrepet. Jeg vil beskrive hvordan en variabel brukes innen matematikk og hvordan den brukes innen programmering. Jeg vil også vise til en kjent metode som mange lærebøker og lærere har brukt for å introdusere variabelbegrepet for elever. Den britiske matematikeren og filosofen Bertrand Russell hevdet at “The variable is perhaps the most distinctively mathematical of all notions; it is certainly also one of the most difficult to understand” (Russell, 1937, s. 89). Med dette som bakgrunn vil jeg likevel gjøre et forsøk på å forklare hvordan vi kan forstå konseptet variabel.

2.2.1 Ulike definisjoner av variabel

Det ligger i ordets natur at en variabel er noe som kan variere. Hvilken intensjon det er med å innføre variabelen vil gjerne gjenspeiles i hvordan man forklarer og forstår ordet. Innen kvantitative forskningsmetoder kan begrepet defineres slik: «Termen variabel refererer til egenskaper og fenomener som forventes å innta mer enn én verdi. (…) I en skolekontekst er for eksempel alder, kjønn, motivasjon og mestring relevante variabler. Når alle som inngår i en undersøkelse er jenter, vil derimot egenskapen kjønn være en konstant. (…) En variabel er en egenskap som på en meningsfull måte kan gi uttrykk for variasjoner i det utvalget som blir studert. Ved målinger beskrives variasjonene ved variabelverdier» (Befring, 2015, s. 122). Statistisk sentralbyrå definerer en variabel som “en egenskap ved en statistisk enhet, og den kan anta mer enn én verdi, i form av en numerisk verdi (kvantitativ variabel) eller en kategori fra en klassifikasjon (kategorisk variabel).” (Statistisk sentralbyrå, 2022). Som jeg vil vise i delkapittel 2.2 vil en variabel i matematikk være en hvilken som helst verdi innenfor en gitt definisjonsmengde, mens i programmering blir den sett på som et lagringssted. Å finne en felles definisjon for begrepet kan virke krevende, og et forslag er å bruke ulike ord i de ulike sammenhengene.

2.2.2 Variabel i matematikk

Doukakis et al (2007) beskriver at "In mathematics a variable is a name (usually one alphabet letter) that can stand for numerical values or other algebraic objects." (Doukakis et al, 2007, s.2). Ifølge denne definisjonen er en variabel en bokstav eller et symbol som kan tilknyttes numeriske verdier. Variabelen kan ifølge den gitte definisjonen også være et algebraisk objekt. Da kan variabelen være et punkt, en linje, et plan, en figur, og algoritmene som kan anvendes på slike variabler kan være speiling, rotasjon og translasjon. Variabelen kan også være en vektor eller en matrise. Videre er det mulig å dele variabelbegrepet inn i to ulike aspekter: "Det første aspektet er oppfatningen om at noe varierer – i motsetning til å være konstant (…). Det andre aspektet er måten en bruker bokstaver på til å representere generaliserte tall i matematikk." (Brekke et al, 2000, s.9). Man kan også bruke følgende definisjon for en variabel: «generalized numbers and functionally related varying quantities" (Gray et al, 2009, s. 60).

Bertrand Russell beskriver en variabel som "something which successively assumed all values of a certain class. (…) In fact, n just denotes any number, and this is something quite distinct from each and all numbers." (Russell, 1937, s. 90-91). Det betyr at en variabel til enhver tid kan ha én verdi gitt fra et bestemt definisjonsområde, uten at det er avklart eksakt hvilken verdi den har. Variabelen har ikke en bestemt verdi og den har heller ikke alle verdiene samtidig. Russell (1937) bruker begrepet "any value" for å beskrive dette.

Ukjent eller variabel:

I likningen x+2=5 er x en bokstav tilknyttet en numerisk verdi. Her vil x ha en ukjent verdi inntil likningen er løst og man får beregnet en tallverdi for den ukjente. Denne bruken av x er ikke det samme som å bruke bokstaven x for en variabel. Betydningen av x kan dermed være svært ulik i ulike oppgaver. "Å introdusere algebra som et språk som uttrykker sammenhenger mellom to variable størrelser er ikke det samme som å finne verdien til en ukjent." (Brekke et al, 2000, s.73). I en likning vil det noen ganger være flere mulige løsninger, som i en andregradslikning hvor man vil kunne ha to mulige løsninger. Man kan også oppleve at løsningen til et problem er en funksjon. Da kan man finne variabler i den ukjente. Det kan se ut til at variabler oppstår på forskjellige nivå i matematikken. Det er i alle fall avklart, etter Brekke et al (2000) sin definisjon, at en ukjent ikke er det samme som en variabel, men selve variabelbegrepet er nok fremdeles litt diffust.

Uavhengig og avhengig variabel

Sammenhengen mellom variable størrelser kan vises gjennom et eksempel. Dersom man ser på et kvadrat hvor sidene har lengden x vil man kunne uttrykke omkretsen som 4x og arealet som x^2. Lengden, omkretsen og arealet til et kvadrat er alle variabler, og det finnes sammenhenger mellom disse variable størrelsene. Endrer vi lengden, vil omkretsen og arealet også endres. Det er vanlig å skille mellom uavhengige og avhengige variabler, og i dette tilfellet vil lengden x være en uavhengig variabel mens omkrets og areal er avhengige variabler. Variabelen x kan ha "any value", etter Russells (1937) definisjon. Lengden har ikke en bestemt verdi, og den har heller ikke alle verdiene samtidig. Den kan bare ha én verdi av gangen, og dersom verdien blir kjent for oss blir den å regne som en konstant deretter. Da kan man enkelt beregne en tallverdi for omkretsen og arealet også. At en variabel er uavhengig betyr at den ikke lar seg påvirke av andre faktorer, mens en avhengig variabel har en bestemt relasjon til en annen variabel. I GeoGebra kan dette illustreres med punkter som er løse og

punkter som er faste. De løse punktene er det mulig å flytte på, mens de faste punktene kan ikke flyttes på og posisjonen deres avhenger av de løse punktenes posisjon.

Variabel eller konstant

I mange sammenhenger vil en variabel på et tidspunkt få en verdi, men en matematisk variabel er ikke lenger en variabel når den har fått en bestemt verdi. Da er den blitt til en konstant. "Every value of a variable is a constant." (Russell, 1937, s.91). Slik Russell beskriver en variabel ser man at det er nesten lettere å si noe om hva en variabel ikke er enn å si hva den er: "Though it is always connected with some class, it is not the class, nor a particular member of the class, nor yet the hole class, but *any* member of the class. On the other hand, it is not the *concept* "any member of the class", but it is that (or those) which this concept denotes." (Russell, 1937, s.351). En variabel har altså tilknytning til en verdi, men ikke til en bestemt verdi. For dersom det er en bestemt verdi, vil det ikke lenger være snakk om en variabel, men en konstant.

Flere måter å bruke bokstaver i matematikken på

Bokstavbruken i matematikken har flere roller enn jeg har beskrevet til nå, og antagelig enda flere enn jeg vil nevne i denne masteroppgaven. Den generelle likningen for en rett linje $y=ax+b$ inneholder fire ulike bokstaver, men er det snakk om fire variabler? Om vi i dette tilfellet lar x være en uavhengig variabel og y en avhengig variabel, vil a og b kalles parametere. "a og b er det en kaller parametere. De står for vilkårlige, men faste tall fra gang til gang. Setter en inn ulike verdier for a og b, får en forskjellige linjer" (Brekke et al, 2000, s. 9). Her ser vi altså tre ulike måter å bruke bokstaver på i en og samme likning: uavhengig variabel, avhengig variabel og parametere. Bertrand Russell (1937) beskriver parametere i rette linjer $ax+by+c=0$ slik: "And thus what are called parameters are simply variables. (…) But unless we are dealing with one absolutely particular line, say the line from a particular point in London to a particular point in Cambridge, our a, b, c are not definite numbers, but stand for any numbers, and are thus also variables." (Russell, 1937, s. 6). Med denne forståelsen vil a, b og c også oppfattes som variabler, selv om vi gjerne vil oppfatte a, b og c på en annen måte enn vi gjør med x og y som variabler i denne likningen. Det som er avgjørende for om a, b og c er variabler eller ikke er om vi har en bestemt linje, eller om det er en hvilken som helst linje man behandler. Det viser seg at variabelbegrepet kan være noe vanskelig å begripe, og Russell (1937) beskriver det slik: "It appears from the above discussion that the variable is a very complicated logical entity, by no means easy to analyze correctly." (Russell, 1937, s.93).

Bokstavbruk

Som vist kan bokstavene i matematikk kan være uavhengige variabler, avhengige variabler, ukjente eller parametere. I noen tilfeller brukes det også bokstaver for konstanter. Gravitasjonskonstanten er et eksempel på dette, og den forkortes til g. Den har en bestemt verdi på ethvert sted man befinner seg. π og tallet e er også konstanter. I tillegg brukes bokstaver for å vise måleenheter som for eksempel gram (g), liter (l) og meter (m). Prefikser kommer i tillegg. At bokstaver brukes på flere ulike måter i matematikken kan virke forvirrende for mange elever, og kan være rot til flere misoppfatninger, særlig i oppgaver hvor man finner flere bokstaver med ulik mening.

Bokstaver brukes mye i matematikken, og det er ikke alltid at en bokstav er en variabel. Selv om variabelbegrepet viser seg å være noe komplisert å forklare, håper jeg likevel at jeg i dette delkapittelet har klart å få frem hva en variabel er, og hva er variabel ikke er, sett i fra et matematisk synspunkt.

2.2.3 Variabel i programmering

I programmering brukes variabler hyppig, men er det egentlig noen forskjell på en variabel innen matematikk og innen programmering? Doukakis et al (2007) forklarer at "The concept of variable in programming shares similarities with the matemathical variable but also has many different differences." (Doukakis et al, 2007, s.2). At det finnes både fellestrekk og forskjeller mellom en matematisk variabel og en programmeringsvariabel kan skape forvirring og gi rom for flere vansker knyttet til bruk av variabler.

Definisjon av en variabel innen programmering

Hva er en variabel innen programmering, og hvordan skiller den seg fra en matematisk variabel? En enkel definisjon av en programmeringsvariabel kan formuleres slik:

> "Variabler er lagringsplasser for verdier." (Mughal et al, 2003, s. 6).

Videre beskrives at "En variabel lar programmereren sette av plass i minnet til en verdi og gi denne plassen et navn. Dessuten har hver variabel en gitt datatype. En datatype er definert ved et verdiområde og et sett av operasjoner. Datatypen brukes til å bestemme hvor mye plass verdien i variabelen vil trenge i minnet, og hvilke verdier som kan lagres i den". (Mughal et al, 2003, s. 6). Disse formuleringene viser tydelig hvordan man forstår variabler innen programmering. Formuleringene støtter også opp Doukakis et al (2007) sin idé om at en programmeringsvariabel har fysisk eksistens, siden den refererer til et fysisk lagringssted. I programmering kan man for eksempel bruke variabler til å gi poeng i et spill, sjekke om noe er true/false eller lagre informasjon som navn, fødested, fødselsdato osv. Det som ikke kommer veldig tydelig frem av den nevnte definisjonen er at variabelen bare kan ha en verdi til enhver tid.

I boken "Lær å kode" brukes en mer lettfattet definisjon: "En variabel er som en spesiell boks som vi har lagret noe viktig inni." (Wainewright, 2018, s. 54). I en startfase kan det være tilstrekkelig med denne definisjonen, men det vil være viktig å poengtere at denne "spesielle boksen" bare kan lagre én verdi av gangen, og at den kan endre verdi i løpet av programmet. Jeg vil komme nærmere inn på bruken av denne forklaringsmodellen i neste delkapittel.

Type, deklarering, initialisering

For å bruke en variabel i programmering vil man starte med å deklarere en variabel av en bestemt type. I Scratch vil man bare finne en type variabel. Av den grunn går det ikke an å velge feil type variabel i Scratch, mens i andre tekstbaserte programmeringsspråk velger man fra starten om variabelen skal inneholde heltall, desimaltall, boolske verdier eller tekst. Det skal legges til at i tillegg til disse variabeltypene finnes det mange flere andre typer variabler også som f.eks. matriser og vektorer. Datatypen som velges er med på å avgjøre hvor mye plass i dataminnet som skal settes av til variabelen. Når variabelen er opprettet vil den i noen programmeringsspråk være tom inntil den blir gitt en verdi, mens i andre programmeringsspråk vil den automatisk få verdien null inntil dette endres via koden eller i

brukergrensesnittet. Det siste er tilfellet for Scratch, som er programmeringsspråket som vil bli brukt i dette prosjektet.

Sett i lys av hvordan en matematisk variabel ble presentert, vil variabelen som har blitt initialisert i grunnen være en konstant, men den kan likevel endre verdi senere. Her er det tydelig at begrepet variabel i matematikk og i programmering ikke samsvarer med hverandre. I matematikk vil en variabel som har fått en verdi regnes som en konstant videre, mens i programmering vil en variabel som har fått en verdi fremdeles regnes som en variabel.

Ulike variabler i programmering

Variabler som brukes i programmering kan ha ulike roller. For grunnleggende programmering finnes det ti ulike roller for variablene som dekker 99% av behovet. (Kuittinen & Sajaniemi, 2004, s. 57). Jeg vil gjøre et skille mellom 1) variabler som er ren input enten fra koden selv eller fra brukergrensesnittet og som holder seg konstant gjennom kjøringen av programmet og 2) variabler som endrer verdi i løpet av programmet. I førstnevnte bruk av variabler kan det for eksempel være at programmet trenger input på lengde og bredde for å beregne arealet til et rektangel. Lengde og bredde endres ikke i løpet av kjøringen av programmet, men kan endres dersom man kjører programmet på nytt. Matematisk er variablene lengde og bredde å regne som konstanter, men likevel defineres de som programmeringsvariabler. Dette kan virke forvirrende, men fordelen er at programmet kan kjøres på nytt flere ganger, og man kan gjøre nye beregninger ganske kjapt bare ved å endre verdien på disse variablene. Lengde og bredde kan altså endre verdi hver gang programmet kjøres på nytt, men ikke i en og samme kjøring. Denne typen variabler går under kategorien «fixed value» i Kuittinen og Sajaniemi (ibid) sin beskrivelse. Et eksempel på sistnevnte bruk av variabler kan være poenggivning i et spill. I løpet av spillet kan variabelen poeng endre verdi flere ganger når en bestemt hendelse inntreffer. Da hendelsene i et spill er avhengig av hvordan spillet spilles, er det mulig at poengsummen ender opp ulikt hver gang man spiller spillet. I et spill kan det være tilfeldig hva som skjer underveis og hvordan en variabel for poeng ender opp ved spillets slutt.

Programmering sammen med matematikk

Noen ganger kan en variabel fungere som et hjelpetall i en iterasjonsprosess. Slik kan variabler utføre svært mange skifter på kort tid og for eksempel beregne arealet under en kurve eller volumet av en pyramide ved å dele det aktuelle området opp i stadig mindre og flere fragmenter. Vanligvis vil man benytte seg av integrasjon for å gjøre slike beregninger, men i Tangenten 3/2020 deler Eidsvig et undervisningsopplegg hvor elevene kan utlede formelen for volumet til en pyramide ved bruk av figurtall, programmering og enkel volumberegning. (Tangenten 3/2020, s. 31-36).

Ved å dele en pyramide med kvadratisk grunnflate inn i mange små kuber vil man kunne utlede følgende formel for volumet til pyramiden:

$$V = l^2 \cdot h \cdot \frac{1 + 2^2 + 3^2 + \cdots + n^2}{n^3}$$

Ved å bruke programmering vil man kunne beregne brøkens verdi lynkjapt, og man kan velge ganske store verdier for n, altså dele pyramiden opp i så små kuber som man måtte ønske. Man vil se at brøken nærmer seg 1/3 og kommer slik frem til formelen for volumet til en pyramide med kvadratisk grunnflate.
(Tangenten 3/2020, s. 31-36)

I det beskrevne undervisningsopplegget brukes variabler av begge kategoriene som er nevnt i ett og samme program. Det er klart at beregningene som utføres i programmet kunne like gjerne ha blitt utført for hånd, men det ville ha tatt mye lenger tid. Programmeringen kan ikke erstatte matematikken, men bespare oss for utallige og krevende gjentatte utregninger. Tenk om Arkimedes (ca. 240 fvt.) hadde hatt slike hjelpemidler da han beregnet verdien av π ved å omskrive og innskrive polygoner på en sirkel. Det må ha vært strevsomt å beregne omkretsen av en 96-kant, men han gjorde det og fant slik en god tilnærmet verdi for π.

Det kan være fristende å påstå at programmering er enklere enn matematikk fordi man i programmering kan løse ganske store problemer ved å bruke bare én variabel, mens man i matematikk behøver mange indekserte variabler for å kunne utføre det samme. "That is, in programming we can use a single variable that changes in the execution of the program, while in mathematics we need a series of indexed variables to accomplish the same thing." (Bråting & Kilhamn, 2021, s.178). Likevel er selve utregningene som foregår akkurat de samme. Når man i matematikken benytter flere indekserte variabler kan man holde kontroll på hvor langt man er kommet i utregningen og man kan når som helst avbryte utregningene og fortsette senere. I et dataprogram som skal utføre de samme beregningene vil man kunne oppnå det samme ved å bruke bare en variabel og en løkke. Løkken kjøres så mange ganger som trengs for å komme i mål med utregningene. Fordelen er at utregningene skjer lynkjapt, slik at programmering blir et kraftig verktøy som kan lette utregninger som man ellers vil utføre i matematikken. For at programmering skal kunne bli brukt på en slik måte må programmereren først ha den matematiske kompetansen som trengs for å kunne løse slike problemer. Det er altså ikke nok med kompetanse i programmering eller matematikk hver for seg for å kunne løse problemstillinger som beskrevet i undervisningsopplegget ovenfor, men om man kombinerer kunnskap innen programmering og matematikk vil det være mulig å løse denne typen problemer ved å bruke programmering som et verktøy i matematikken.

Bruk av variabler i programmering vil kunne forenkle utregninger mye og vil kunne gi et mye større spillerom og muligheter enn ellers. Lister et al (2009) beskriver det slik: "We belive that one promising technique for helping students to see "the forest" is the concept of the roles of variables." (Lister et al, 2009, s. 165). Her er det å se skogen brukt som et bilde på å forstå helheten i et program fremfor bare å se hver enkelt kode alene.

2.2.4 Bruk av analogier for variabler

Det er viktig at elever som skal begynne å bruke variabler har en forståelse for hva en variabel er og hva den kan brukes til. Mange har derfor forsøkt å illustrere hva en variabel er ved å bruke bilder fra vårt hverdagsliv som elevene kan knytte variabelbegrepet opp til. Dette kan være nyttig, men kan også skape ytterligere misoppfatninger dersom man ikke er tydelig nok. "The most common analogy for a variable is to some kind of box or drawer with a label on it. Without further explanation this can be confusing" (Boulay, 1986, s.65). La oss se nærmere på denne analogien som ofte benyttes ved innføring av variabler og hva som kan være lurt å være oppmerksom på.

I en eske eller skuff som beskrevet av Boulay (1986) vil navnelappen utenpå esken/skuffen representere selve variabelnavnet. Variabelnavnet er som oftest valgt ut ifra hva variabelen skal si noe om. Det kan være for eksempel "jentenavn", "temperatur", "ukedag", "partall"

eller "x". Variabelnavnet endres ikke underveis selv om innholdet gjør det. Dersom man skal registrere to fenomener samtidig må man ha to ulike variabler med ulike variabelnavn.

Det som ligger oppi esken/skuffen vil være variabelens verdi. I virkeligheten vil en eske eller skuff kunne inneholde flere gjenstander, men dette er ikke tilfelle for en variabel. En variabel kan bare ha én verdi til ethvert tidspunkt. Dersom variabelen får en ny verdi, vil den gamle verdien kastes og bli glemt. Dersom gamle verdier ikke skal bli glemt må de bli lagret et annet sted, for eksempel i en tabell. Da må det være med i programkoden at dette skal skje. Det er også viktig å huske på at man ikke kan legge verdier for temperatur i esken for jentenavn eller omvendt. Variabelens verdi må passe overens med hvilken type innhold esken er bygget for. Innen matematikk vil det oppstå en konflikt med en gang esken åpnes og man ser variabelens verdi. For som kjent er variabelen i samme øyeblikk blitt å regne som en konstant, om man tenker ut ifra et matematisk perspektiv. Man kan si at det vil være nærmest umulig å legge inn en ny verdi i esken og kalle det en variabel, for man kan se hva som står på papirlappen før man legger det i esken, og da er det allerede en kjent verdi. Det er mulig at analogien om at en variabel er som en eske eller skuff passer bedre til programmering enn til matematikk, siden man i programmering viser til et fysisk lagringssted, men flere kritiserer denne forklaringsmodellen også i denne sammenhengen. Clancy (2004), sitert i Qian og Lehman (2017) forklarer det slik: «Describing a variable as a box, which is something many teachers of introductory programming courses do, is an example of misapplication of analogy in programming.» (Qian & Lehman, 2017, s. 10). Det viser seg at det kan være lett å kritisere eksisterende analogier, påpeke fallgruver og kilder til misoppfatninger, men det kan være utfordrende å finne bedre alternativer for å introdusere variabelbegrepet for nybegynnere.

Dersom man velger å bruke hverdagslige bilder for å illustrere en variabel er det viktig å gjøre det såpass grundig at det ikke skaper ytterligere forvirring blant elevene. Siden måten å gjøre nytte av en variabel på er ulik i ulike situasjoner, kan man kanskje bruke ulike ord og underkategorier for begrepet. For "by making our x always an unrestricted variable, we can speak of *the* variable, which is conceptually identical in Logic, Arithmetic, Geometry, and all other formal subjects." (Russell, 1937, s. 91) Det kan godt tenkes at man behøver denne forståelsen av en variabel også innen programmering, som når man bruker matematiske funksjoner som en del av programmeringen, og da vil en matematisk variabel være noe annet enn en programmeringsvariabel.

Å bruke analogier som beskrevet ovenfor kan være både til hjelp og til besvær. Det er mulig at esken og skuffen ikke er gode nok bilder på hva en variabel faktisk er, selv om noen elever kanskje vil ha nytte av en slik introduksjon i starten. Det har vist seg at misoppfatninger skjer uansett om man bruker slike forklaringer eller ikke: "It is true that inappropriate analogies like the box or the plate analogy have been used by textbooks and teachers but problems seem to exist even in cases where no analogies are used." (Doukakis et al, 2007, s.1). Påstanden om at analogien er upassende begrunnes ikke nærmere, men jeg har belyst noen punkter knyttet til dette ovenfor. For programmering, og i de tilfellene hvor variabelen tildeles en verdi og egentlig behandles som en konstant, vil nok forklaringen med esken kunne være god i en oppstartsfase, men det er viktig at det presiseres at variabelen bare kan ha en verdi av gangen. Esken blir et forenklet bilde av et lagringssted. Innen matematikk derimot, hvor variabelen representerer "any number" innenfor et definisjonsområde, kan forklaringen med esken skape forvirring, nettopp fordi elevene kan få en forestilling om at

variabelen alltid skal ha en bestemt verdi. Dersom man tar i bruk forklaringsmodeller for å billedliggjøre variabelbegrepet er det viktig å ha tatt disse utfordringene med i betraktning og gjerne diskutert dem med elevene.

Selv om variabelbegrepet omfavner mye og kan være vanskelig å forstå, håper jeg likevel at jeg i dette kapittelet har klart å sette lys på hva en variabel er og hva den kan brukes til innen matematikk og programmering. Jeg har forsøkt å vise til hva som er likt og ulikt med variabler innen matematikk og programmering, og utfordringer knyttet til dette. Siden problemstillingen for oppgaven er knyttet til variabler innen programmering, vil ordet variabel i fortsettelsen referere til en programmeringsvariabel.

2.3 Misoppfatninger og andre vansker

I dette delkapittelet vil jeg belyse teori knyttet til misoppfatninger og andre vansker som kan oppstå når man skal ta i bruk variabler i programmering. Begreper som syntaktisk kunnskap, konseptuell kunnskap og strategisk kunnskap vil være med på å forklare skillet mellom misoppfatninger og andre vansker, og jeg vil beskrive hvorfor det er utfordrende å spore og identifisere en misoppfatning. Jeg vil vise til ulike aspekter som kan skape hindringer for å lære og forstå programmering, og til slutt vil jeg vise et utvalg av spesielle misoppfatninger som jeg kommer til å ha et søkelys på videre i denne oppgaven.

2.3.1 Hva er en misoppfatning

"Et begrep er sjelden fullstendig utviklet ved at en har gjort erfaringer på et avgrenset felt. En kaller ufullstendige tanker om et begrep for misoppfatninger" (Brekke et al, 2000, s. 10). Misoppfatninger kan også forklares som feilaktige idéer som kommer i konflikt med teorier som det er bred enighet om innen forskningsmiljøer. (Qian & Lehman, 2017, s.2). Da ordet misoppfatning kan virke noe negativt ladd, er det vanlig å også kunne bruke andre begreper for det samme. Av denne grunn vil misoppfatninger i noen kilder kunne bli kalt for alternative oppfatninger, og innen programmering kan man finne begreper som «difficulties», «errors», «bugs», «mistakes» «partial understandings», «incorrect understandings», «student-constructed rules» osv. Sorva (2013) bruker ordet misoppfatninger i en ganske vid forstand innen programmering som «understandings that are deficient or inadequate for many practical programming contexts» (Sorva, 2013, s.5). Innen programmering kan man også definere en misoppfatning som «an incorrect understanding of a programming concept or a set of related concepts, typically affected by prior knowledge from domains other than programming such as mathematics and natural languages.» (Swidan et al, 2018, s.151).

2.3.2 Å identifisere misoppfatninger

Når man skal forske på misoppfatninger er det naturlig å studere elevers feilaktige besvarelser og analysere disse i detalj. Hvilken feil var det som ble gjort, hvordan var oppgaven utformet, og hvilken type misoppfatning hadde eleven? Det er verdt å huske på at feil også kan skje selv om man ikke har noen som helst misoppfatning omkring temaet. Man kan ha lest oppgaven feil, det kan være at oppgaveformuleringen var utydelig og kunne bli tolket på flere ulike måter, man kan ha gjort en tastefeil, eller man var rett og slett ikke oppmerksom nok i øyeblikket. Feil kan oppstå av mange ulike grunner, og det er ikke alle feil som kan spores tilbake til en bestemt misoppfatning. Av og til kan det være diffuse skiller mellom feil som oppstår på grunn av misoppfatninger og feil som oppstår av andre grunner.

“Difficulties that students encounter in programming are not always neatly identifiable, and misconceptions may contribute to other kinds of difficulties or errors that students encounter” (Qian & Lehman, 2017, s.3). De uklare linjene mellom misoppfatninger og andre vansker kan også være en faktor som bidrar til at mange velger å bruke andre begreper enn ordet misoppfatning. I kapittelet med resultater og drøfting brukes ordet misoppfatning hyppig, som en type forklaring på ulike feil som elevene har gjort. Da er det verdt å ha dette synet på misoppfatninger, vansker og feil klart for seg. En bestemt type feil i en bestemt type oppgave kan fortelle oss at en elev sannsynligvis har misoppfatningen, men det kan også være andre grunner til at feilen oppsto. I tillegg kan man i noen tilfeller oppleve at elever som har svart riktig på en oppgave likevel har en misoppfatning omkring teamet. Disse ulike aspektene er det viktig å ta hensyn til når man bruker diagnostiske oppgaver for å avdekke misoppfatninger, og at man bruker gjennomtenkte og velformulerte oppgaver som er vanskelig å tolke i flere ulike retninger. Brekke (2002) har belyst hvordan man kan ta i bruk diagnostiske oppgaver i klasserommet for å identifisere misoppfatninger, og hvordan man kan oppklare misoppfatningene ved å skape situasjoner hvor elevene kommer i en kognitiv konflikt. En slik prosess kalles for diagnostisk undervisning. I denne studiens avsluttende kartleggingstest har jeg benyttet diagnostiske oppgaver, og da er det viktig å huske på de ovenfornevnte betraktninger. Det er mulig at de feilaktige svarene som ble gitt i testen har oppstått av andre grunner enn jeg har klart å belyse.

2.3.3 Ulike kunnskaper og ulike vansker

Før jeg kan komme nærmere inn på hvilke vansker og misoppfatninger elever kan ha innen programmering, er det nødvendig å belyse noe teori om elevers ulike kunnskapsområder innen programmering. Gjennom denne teorien vil det bli mulig å beskrive hvilke vansker og misoppfatninger elever kan ha og innen hvilken gren de utspringer fra.

Det finnes ulike modeller som tilstreber å beskrive hvilke typer kunnskaper og ferdigheter man behøver for å mestre og forstå et fagområde fullt ut. Kilpatrick et al (2001) beskriver fem ferdighetsområder som er nødvendig innen matematikk: «conceptual understanding, procedural fluency, strategic competence, adaptiv reasoning and productive disposition.» (Kilpatrick et al, 2001, s.116). Qian og Lehman (2017, s.3-4) beskriver tre ulike kunnskaper innen programmering: «syntactic, conceptual and strategic». McGill og Volet (1997) knytter den sistnevnte modellen sammen med tre kunnskaper som man gjerne bruker innen kognitiv psykologi: «declarative, procedural and conditional knowledge». De har konstruert et todimensjonalt rammeverk som beskriver ulike typer kunnskaper innen programmeringskunnskap. Jeg har valgt å bruke Qian og Lehman (2017) sin modell videre i denne oppgaven, da denne modellen tydeliggjør forskjellen mellom misoppfatninger og andre vansker på en oppklarende måte. I det følgende vil jeg beskrive hvordan de deler programmeringskunnskap inn i tre ulike grener. Jeg vil videre beskrive hvilke typer vansker som kan oppstå innen de ulike kategoriene.

1) Syntaktisk kunnskap

> «Syntactic knowledge is the knowledge of the language features, basic facts, and rules. For example, in Java, double quotation marks are required to define strings, and a semicolon is required to end a statement.» (Qian & Lehman, 2017, s.3)

Syntaktisk kunnskap kan beskrives som en type kunnskap om et valgt programmeringsspråk, om å kunne forstå hvilke regler som gjelder i språket, som for eksempel hvor man skal bruke dobbelt anførselstegn og hvor man skal bruke semikolon og vite forskjellen på symboler som et likhetstegn og et dobbelt likhetstegn. Om man bruker slike symboler på en feil måte vil ikke programmet virke slik som man ønsker. Denne typen feil er hyppig observert blant nye programmerere, og det skal være ganske lett å oppklare og korrigere denne typen feil. (Qian & Lehman, 2017, s.5). I programmeringsspråket Python vil man ved kjøring av et program kunne få ut feilmeldinger som viser hvilken linje i koden som inneholder en syntaktisk feil. Feilen kan være liten, som et innrykk for lite eller for mye. Når dette blir korrigert for vil programmet virke slik som man ønsker. I Scratch er blokkene konstruert slik at man unngår denne typen syntaktiske feil. Blokkene er som legoklosser, og det er ikke alle som passer sammen. En blokk som er formet som en sekskant kan kun plasseres inn et sted hvor det mangler en sekskant. Forsøker man å sette inn noe annet som ville gitt syntaktisk feil ved kjøring av programmet, vil ikke Scratch tillate det, og man blir på denne måten oppmerksom på hva som er tillatt og ikke tillatt allerede før kjøring av programmet. Det er likevel mulig å skrive et program som oppfører seg ganske annerledes enn hva man hadde planlagt. Syntaktisk kunnskap er dermed kunnskap om hvilke regler som gjelder i et valgt programmeringsspråk. Man kan ha en idé om hvordan et program skal utformes, men dersom man mangler syntaktisk kunnskap innen valgt programmeringsspråk vil man ikke klare å skrive programkoden til programmet.

2) Konseptuell kunnskap

> «Conceptual knowledge refers to the knowledge of how programming constructs and principles work and what happens inside the computer.» (Qian & Lehman, 2017, s.3)

Vi ser av nevnte definisjon at konseptuell kunnskap beskrives som en kunnskap om de underliggende prinsipper i programmeringen. Eksempler på hva som inngår i konseptuell kunnskap kan være forståelse for betingelser, variabler, inn- og utdata og løkker. (Qian & Lehman, 2017, s.5).

Boulay (1986) beskriver at vansker og misoppfatninger innen programmering kan stamme fra at eleven har problemer med å forstå «the notional machine». Dette innebærer at man ikke har forståelse for de underliggende prinsippene og har problemer med å forstå hva som skjer inne i datamaskinen. I noen tilfeller kan slike vansker føre med seg en antagelse av at datamaskinen er «intelligent» og kan hjelpe programmereren med å få ønsket resultat. (Sorva, 2013, s.7). For å forstå hvordan et program virker er det viktig å forstå tidsaspektet gjennom programmets kjøring og evne å se den dynamiske siden. «Many misconceptions, if not most of them, have to do with aspects that are not readily visible, but hidden within the executive-time world of the notional machine: references, objects, automatic updates to loop control variables, and so forth.» (Sorva, 2013, s.6-7). Det er først når man har oppnådd en viss grad av konseptuell og syntaktisk kunnskap samtidig at det vil bli mulig å planlegge, skrive, forklare og feilsøke programmer. (Qian & Lehman, 2017, s.6).

Som tidligere beskrevet er det mange måter å tolke ordet misoppfatning på, og det finnes mange synonymer til ordet misoppfatning som er mer eller mindre korrekte, men litt mer spissformulert kan misoppfatninger defineres som «errors in conceptual understanding». (Qian & Lehman, 2017, s.3). Dette betyr at syntaktiske feil kan oppstå uavhengig av

misoppfatninger, og denne typen feil kan ofte lett spores og oppklares. Men dersom en programmerer har gjort noe feil med løkker eller tildeling av verdier til variabler, noe som er knyttet til konseptuell kunnskap, da kan man si at det er sannsynlig at programmereren kan ha en misoppfatning knyttet til dette, selv om det ikke alltid er direkte knyttet til en misoppfatning. Misoppfatninger er vanskeligere å spore og tar lengre tid å oppklare enn syntaktiske vansker. I dette prosjektet har jeg hatt søkelys på hvilke vansker som oppstår når elever skal tildele verdier til variabler. Dette er innenfor konseptuell kunnskap, og feil som blir gjort kan i mange tilfeller være på grunn av at eleven har en type misoppfatning omkring dette.

Til forskjell fra vansker innen syntaktisk kunnskap, som er lett å oppdage og lett å korrigere, vil vansker innen konseptuell kunnskap være vanskeligere å spore og de kan føre til «deep and significant misconceptions that are related to students' mental models of code execution and the computer system» (Qian & Lehman, 2017, s.5).

3) Strategisk kunnskap

> "Strategic knowledge refers to how to apply syntactic and conceptual knowledge of programming to solve novel problems. Planning, writing, and debugging programs including tracing or explaining code require stategic knowledge, in addition to knowing syntax and concepts of a programming language." (Qian & Lehman, 2017, s.4)

Strategisk kunnskap forklares som en «expert-level knowledge» (Qian & Lehman, 2017, s.6) hvor man skal kunne planlegge en strategi for å løse problemet, dekomponere problemet, skrive og forklare programmet i tillegg til å kunne feilsøke programmet og forbedre den. Man bruker da både syntaktisk og konseptuell kunnskap for å løse nye og ukjente problemer.

Videre beskriver Qian og Lehman (2017, s.6) noen eksempler på hva vansker innen strategisk kunnskap kan innebære. Det kan være vansker med å initialisere variabler, vansker med å bruke variabler i en løkke, vansker med å velge riktig type løkke i en gitt kontekst og vansker med å opprette betingelser som hindrer feil som divisjon med null. Disse problemene oppstår ikke nødvendigvis fordi eleven har en misoppfatning av en bestemt type, men kanskje fordi problemet er av et omfang som gjør det nødvendig å dekomponere det og gjøre ulike steg i ulike bolker, og da kan det være vanskelig for eleven å se helheten av programmet. «Students' difficulties in strategic knowledge are highly correlated to their difficulties in syntactic knowledge and conceptual knowledge.» (Qian & Lehman, 2017, s.6). Vanskene innen strategisk kunnskap går ofte ut på å forstå selve oppgaven, å dekomponere problemet, å kunne teste og forbedre koden. (Qian & Lehman, 2017, s.6-7)

2.3.4 Feil, vansker og misoppfatninger

Etter gjennomgangen av de ulike kunnskapene blir det forståelig at det er mulig å gjøre feil med variabler både på grunn av manglende syntaktisk kunnskap, manglende konseptuell kunnskap og manglende strategisk kunnskap. «In introductory programming courses, students exhibit various misconceptions and other difficulties in syntactic knowledge, conceptual knowledge, and strategic knowledge.» (Qian & Lehman, 2017, s.23). Syntaktiske feil kan være knyttet til at man for eksempel ikke bruker selve variabelen i koden, men feilaktig bruker et objekt med samme navn i stedet. Feil som er knyttet til manglende konseptuell kunnskap, og som dermed kan være grunnet misoppfatninger, kan være for

eksempel at man har en oppfatning om at variabelen kan holde flere verdier samtidig, eller at man har to variabler samtidig og tildeler verdiene til feil variabel. Feil som er knyttet til manglende strategisk kunnskap kan være for eksempel at man glemmer å initialisere variabler. I denne studien har jeg hatt et fokus på å studere utfordringer som elever kan oppleve knyttet til variabelbruk innen programmering. Det er mulig at noen av funnene som beskrives i kapittelet «resultater og drøfting» kan spores tilbake til manglende syntaktisk eller strategisk kunnskap og dermed ikke er knyttet til en bestemt type misoppfatning. Alle funnene som jeg presenterer i kapittel 4 vil likevel kunne vise til ulike aspekter som kan gjøre det utfordrende for elevene å anvende variabler i programmeringen, og det kan være på grunn av både misoppfatninger og andre vansker og noen ganger kanskje en kombinasjon. Qian og Lehman (2017) har presentert et samlet fokus på misoppfatninger og andre vansker, og jeg har valgt å gjøre det samme i denne oppgaven. I kapittelet med resultater og drøfting vil det derfor presenteres funn av både misoppfatninger og andre vansker.

2.3.5 Vansker knyttet til semiotiske representasjoner

"In mathematics, variables have only symbolic existence and not physical existence" (Doukakis et al, 2007, s.3). At variabelen ikke har fysisk eksistens gjør at vi hverken kan se den eller ta på den. Variabelen eksisterer likevel som et matematisk konsept, og symboler som brukes for variabler er en måte å representere idéen om selve variabelkonseptet. Steinbring (2006) beskriver et epistemologisk triangel som viser hvordan matematiske objekter kan representeres ved hjelp av symboler, og hvordan dette knyttes til matematiske konsepter. Tallet fem er et matematisk konsept som kan representeres ved symbolet 5. Tallet kan oppstå i ulike situasjoner, og kan for eksempel være knyttet til en enkelt situasjon hvor man har fem epler. På samme måte kan man bruke en eske for å representere en variabel, eller man kan skrive variabelen som x. Å klare å koble den symbolske representasjonen til en konkret situasjon er viktig for forståelsen, enten det er snakk om tallet 5 eller variabelen x. Når man har mestret å koble den symbolske representasjonen til en konkret situasjon kan et nytt problem oppstå: Ikke-kongruente konverteringer mellom ulike systemer.

Selv om programmering utspringer fra algebra bringer programmeringen inn en ny type semiotisk representasjon som bryter med de syntaktiske reglene som gjelder i symbolsk algebra. (Bråting & Kilhamn, 2021, s.173). Både syntaks, som er selve oppbygningen av språket, og semantikken, som er symbolene som brukes, kan være ulik fra det man kjenner igjen i algebra. I noen tilfeller kan man oppleve at et og samme symbol kan ha ulik betydning i de ulike systemene. Av denne grunn kan et symbol bare ha en betydning innen et bestemt semiotisk system. (Duval, 2006, s.110). Likhetstegnet er et eksempel på et symbol som har ulik betydning i de ulike systemene. I algebra vil et likhetstegn alltid referere til en ekvivalens, mens i programmeringsspråket Python vil et likhetstegn kunne benyttes for å tildele verdier til variabler. I Python vil man bruke dobbelt likhetstegn i en situasjon med ekvivalens. Ellers er det ofte flere måter å symbolisere det samme på innad i et og samme system. Om en variabel skal øke verdien sin med én, kan man i JavaScript skrive dette både som a = a + 1 og som a++. I neste avsnitt beskrives overgangen mellom slike notasjoner nærmere.

2.3.6 Transformasjoner og misoppfatninger

Duval (2006, s.111) gjør et skille mellom transformasjoner som skjer innad i samme system (bearbeidinger) og transformasjoner som skjer mellom ulike system (konverteringer).

Eksempelet med at koden a = a + 1 betyr det samme som a++ viser til en bearbeiding, siden transformasjonen skjer innad i samme system. Konverteringer, derimot, er transformasjoner som skjer mellom ulike system. En kongruent konvertering kan være når naturlig språk som «fire pluss fem er ni» omformes til 4+5=9, mens en ikke-kongruent konvertering kan være når naturlig språk som «summen av fem og fire er ni» blir omformet til 5+4=9. (Bråting & Kilhamn, 2021, s.174). Det er de ikke-kongruente konverteringene mellom ulike representasjonsformer som vanligvis skaper mest problemer for elevene. (Duval, 2006, s.123). Et annet eksempel på en ikke-kongruent konvertering er når man skal konvertere likningen for en rett linje over til en grafisk representasjon eller en tabell. Elever kan tro at det er snakk om ulike matematiske objekter, når det egentlig bare er ulike representasjonsformer for det samme objektet. Ifølge Duval (2006, s.124) skjer denne feilaktige oppfatningen hyppigere dersom konverteringen går mellom to ulike semiotiske representasjoner hvor ingen andre representasjonsformer er tilgjengelig for eleven. «Conversion of representation requires the cognitive DISSOCIATION of the represented object and the content of the particular semiotic representation through which it has been first introduced and used in teaching. But there is a cognitive IMPOSSIBILITY OF DISSOCIATING any semiotic representation content and its first represented object when there is no other possible access to the mathematical object than semiotic. That conflict leads to the consideration of two representations of the same object as being two mathematical objects.» (Duval, 2006, s.124). Av denne grunn er det viktig at man knytter den semiotiske representasjonen opp mot noe konkret og kjent for elevene. Også Brekke et al (2000, s.81) poengterer viktigheten av disse transformasjonene slik: "For å forstå at symboler kan representere variable størrelser, trenger elevene å få bred erfaring i å "oversette" fra en situasjon til et algebraisk uttrykk og fra et algebraisk uttrykk til en situasjon." I dette tilfellet er transformasjonen mellom algebra og en situasjon, mens i programmeringen er transformasjonen ofte mellom algebra og programmeringsspråk, hvilket byr på litt større utfordringer siden begge systemene er basert på symbolske registre. Da behøves kanskje en tredje representasjonsform samtidig for å oppnå ønsket kompetanse. På et masterseminar presenterte Morten Munthe (personlig kommunikasjon, 01.12.22) et forslag om at man kan bruke et flytdiagram som en overgang mellom algebra og programmering. Dette kan for noen være et verktøy som kan være til hjelp i prosessen med å skifte mellom ulike representasjonsformer.

«The ability to change from one representation system to another is very often the critical threshold for progress in learning and for problem solving.» (Duval, 2006, s.107). Dette tyder på at selv om overgangene mellom ulike representasjonsformer kan være strevsomme, kan mestring på dette området også kunne virke svært positivt for læringsprosessen.

2.3.7 Dobbel barriere

Munthe (personlig kommunikasjon, 01.12.22) forklarte på masterseminar begrepet «dobbel barriere», hvor man tenkte seg at elevene møtte utfordringer både innenfor programmeringen og innenfor matematikken samtidig. Han poengterte viktigheten av at elevene har nok forkunnskaper til å løse oppgavene og ikke møter slike doble barrierer. Man kan lage dataprogrammer uten å trekke inn noe særlig matematikk, men noen ganger skal man bruke matematikk i programmeringen, og da må matematikken være kjent på forhånd. I andre tilfeller kan det hende at man vil bruke programmeringen som et verktøy til å utforske

matematiske sammenhenger, og i dette tilfellet er det viktig at elevene har nok kunnskaper innen programmering på forhånd. På denne måten hindrer man å møte på en såkalt dobbel barriere hvor både matematikken og programmeringen er ukjent for eleven, og da vil også overgangen mellom de ulike representasjonene være mer overkommelig.

2.3.8 Endre på notasjoner?

Som nevnt vil programmering bringe inn en ny type semiotisk representasjon for elevene som ikke alltid samsvarer med det de kjenner til fra før. TIMSS-rapporten fra 2019 viser at norske elever presterer lavere enn Sverige og Finland når det gjelder algebra (Kaarstein et al, 2020), og derfor er det ekstra viktig for oss å sette fokus på hvordan en ny type representasjonsform kan virke forstyrrende eller støttende til allerede eksisterende kunnskap. Bråting og Kilhamn (2021) mener at bevissthet eller ubevissthet omkring dette kan styrke eller svekke innlæring av algebra, og ønsker mer forskning på dette området. De påpeker også at det er viktig å vente med å iverksette forslag om at algebraisk notasjon må vike til fordel for programmeringsspråk. Dette fordi man først må forske mer på hvordan en slik endring kan påvirke læring i algebra. Matematikken har lange historiske røtter, og dersom notasjoner skal omgjøres for å forenes mer med programmering må det gjøres med omhu. Det er kanskje også en mulighet at programmeringsspråkene kunne ha gjort noen justeringer for å tilnærme seg matematikken bedre. Mange problemer kan løses både ved bruk av algebra og ved bruk av programmering, og ofte i en kombinasjon, men siden det er en ikke-kongruent konvertering mellom disse systemene, kan dette være rot til mange av vanskene som oppstår. Swidan et al (2018, s.151) påpeker at i tillegg til at kjent kunnskap innen matematikk vil kunne komme i konflikt med programmeringsspråket vil også vårt naturlige talespråk kunne komme i samme konflikt. Det ser uansett ut til at de ikke-kongruente konverteringene kan være en faktor for at ulike vansker kan oppstå. Mer forskning på dette området vil være mulig i årene fremover, nå som programmering er kommet inn i våre læreplaner.

2.3.9 Sorvas 162 misoppfatninger

Det finnes mange studier om misoppfatninger innen programmering. Sorva (2012, s.358-368) har samlet en omfattende liste med 162 slike misoppfatninger, og denne listen tar utgangspunkt i og refererer til kjent litteratur innen forskningsmiljøet. Mange av de 162 misoppfatningene har opptil flere referanser, og jeg ser på denne listen over misoppfatninger som et solid, grundig og gjennomført arbeid. Av denne grunn har jeg valgt å bruke denne listen som et utgangspunkt for mitt prosjekt.

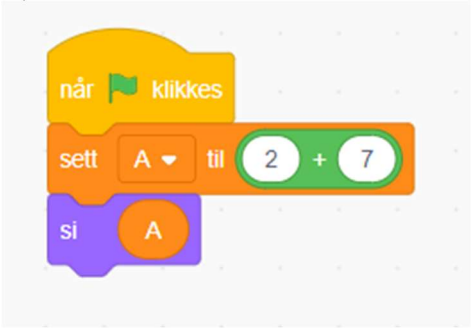
De 162 misoppfatningene som Sorva (2012, s.258-368) har med i sin oversikt er sortert etter ulike kategorier, og jeg har valgt å se nærmere på de som er aktuell for Scratch-brukere og som handler om å tildele en variabel verdi (VarAssign). I oversikten nedenfor viser jeg de 9 misoppfatningene som jeg vil se nærmere på i dette prosjektet. Som det vises har jeg laget en diagnostisk oppgave til hver av disse utvalgte misoppfatningene. I noen tilfeller har jeg splittet vanskene ytterligere opp og laget flere oppgaver på en og samme misoppfatning. I andre tilfeller har jeg sett at det kan forekomme flere misoppfatninger på en og samme oppgave. Oppgavene er ment å være utformet på en slik måte at de kan avdekke misoppfatninger av de ulike typene. Da mange misoppfatninger kan være knyttet sammen på ulikt vis, er det viktig å poengtere at selv om en elev svarer feil på en bestemt oppgave er det likevel ikke sikkert at hen har akkurat denne typen misoppfatning, men at det er sannsynlig. I noen tilfeller kan det være andre grunner til at de har svart som de har gjort, for eksempel kan

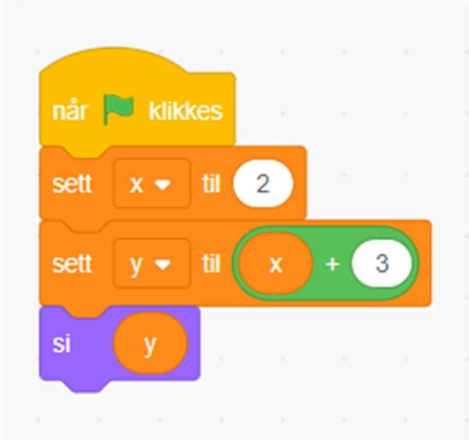
de ha kopiert svaret til den som sitter ved siden av eller det var noe annet som gjorde at de endte opp med akkurat dette feilaktige svaret. Om jeg i fortsettelsen skulle beskrive misoppfatninger som elever har på en diagnostisk måte, er det viktig å ha i minne at misoppfatninger kan være komplekst sammensatt og ikke alltid er enkel å hverken spore eller forklare. Oppgavene som jeg har designet til hver misoppfatning er inspirert av artikkelen "Programming Misconceptions for School Students" (Swidan et al, 2018), men jeg har valgt et litt annet fokus og har tilpasset oppgavene til mitt prosjekt. Jeg vil også nevne at misoppfatninger om variabler er nevnt i Sorva (2012, s.358-368) sin liste under andre kategorier enn VarAssign også, men jeg har avgrenset min oppgave til å ha fokus på de misoppfatningene som er under VarAssign og som er relevant for Scratch-brukere. Jeg har likevel valgt å ta med M23, som er plassert i en annen kategori av misoppfatninger som kalles Control. M23 handler om vansker med å forstå betydningen rekkefølgen på koden har for programmet. Der har jeg valgt å ta med en oppgave hvor variabler inngår og der man samtidig blir utfordret på forståelsen av hvilken betydning rekkefølgen av koden har. En utvidet studie som omfatter absolutt alle misoppfatningene som handler om variabler i programmering ville vært interessant, men jeg ville anbefalt å gjennomføre det med elever som har litt mer erfaring med programmering enn min elevgruppe hadde, og det ville dessuten krevd mye mer tid og ressurser fra den som skulle gjennomført det. Å gi elevene en diagnostisk test med 162 oppgaver ville nok ikke vært den beste løsningen, og prosjektet måtte nok ha blitt delt opp, gjerne med flere forskere og flere elevgrupper. Jeg har valgt å avgrense mitt prosjekt til et ganske konkret og avgrenset område som er mulig å gjennomføre i en enkelt klasse med den tid som jeg har tilgjengelig med dem.

Det er verdt å merke seg at selv om misoppfatninger kan være uavhengige av hvilket programmeringsspråk man bruker, vil de kunne opptre ulikt i ulike programmeringsspråk. Dette beskrives av Kaczmarczyk et al slik: "However, many of the misconceptions we found are generally belived to be universal, but play out differently in different languages, and as such will need to be dealt with in the inventory". (Kaczmarczyk et al, 2010, s.110). Det er dermed grunn til å anta at funnene i denne studien kan ha verdi for brukere av andre programmeringsspråk også, men at vanskene kanskje kan komme til syne på en annen måte.

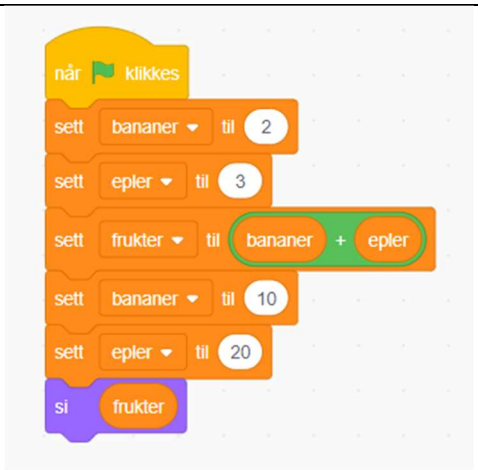
I tabell 1 beskriver jeg misoppfatningene som er valgt ut for prosjektet og hvordan jeg vil søke etter disse blant elevene. Etter tabellen kommer en nærmere beskrivelse av hver enkelt misoppfatning.

Misoppfatning	Beskrivelse	Hvordan søke etter dette?
M9	En variabel kan ha flere ulike verdier på samme tid / kan huske gamle verdier.	 Riktig svar: 2 Feil svar pga. M9: 1,2

M11	Tildeling av verdi virker i motsatt retning.	 Riktig svar: 20, 20 Feil svar pga. M11: 10, 10
M12	Tildeling av verdi virker i begge retninger.	Undersøker dette ved å bruke samme oppgave som for M11. Feil svar pga. M12: 20, 10
M15	Variabler som tildeles en verdi via et uttrykk lagrer likninger eller uløste uttrykk. Jeg deler denne misoppfatningen opp i to ulike vansker, og har tre oppgaver for å kartlegge misoppfatningen. M15a: Gjengir variabelens navn. M15b: Gjengir et uløst regnestykke eller en likning, men kan ha klart å erstatte variabler med verdier.	A)  Riktig svar: 3 Feil svar pga. M15a: AntallLiv B)  Riktig svar: 9 Feil svar pga. M15a: A Feil svar pga. M15b: 2+7 eller A=2+7

		C)  Riktig svar: 5 Feil svar pga. M15a: y Feil svar pga M15b: x+3 eller y=x+3 eller 2+3 eller y=2+3
M16	Tildeling av en verdi flytter en verdi fra en variabel til en annen.	 Riktig svar: 2, 2 Feil svar pga. M16: 0, 2
M17	Det valgte variabelnavnet vil påvirke hvilken verdi som går inn i hvilken variabel.	 Riktig svar: 1000, 1000 Feil svar pga. M17: 1000, 1

M18	Rekkefølgen man oppretter og navngir variablene vil påvirke hvilken verdi som går til hvilken variabel.	 Riktig svar: 3, 0 Feil svar pga. M18: 0, 3
M20	Misoppfatninger knyttet til å endre en variabel.	A)  Riktig svar: 8 Feil svar pga. M20: 3 B)  Riktig svar: 2 Det kan være aktuelt å undersøke om elevene forstår oppgave A eller B best.

M23	Vansker med å forstå betydningen av rekkefølgen på koden. Denne misoppfatningen er ikke presentert under kategorien VarAssign, men jeg velger å ta den med likevel, da jeg finner den aktuell innen tildeling og endring av flere variablers verdier, samtidig som forståelse for rekkefølge spiller en viktig rolle.	 Riktig svar: 5 Feil svar pga. M23: 30

Tabell 1

M9: En variabel kan ha flere ulike verdier på samme tid / kan huske gamle verdier

Misoppfatning M9 handler om at man har en feilaktig oppfatning om at en variabel kan inneholde flere verdier på samme tid. Dersom man har introdusert variabler ved å bruke en eske kan dette være en mulig årsak til denne misoppfatningen. En eske kan inneholde flere lapper med ulike verdier, men en variabel i programmering kan bare ha en verdi til enhver tid.

M9 handler også om en feilaktig oppfatning om at en variabel kan huske gamle verdier. Man kan feilaktig anta at datamaskinen er smart nok til å huske gamle verdier, og at den automatisk velger den verdien som det er passende å bruke i en gitt setting. Slik er det ikke, og et dataprogram må alltid få en detaljert beskrivelse av hva som skal skje. Om en variabel får tildelt en ny verdi vil denne overskrive den tidligere verdien. Gamle verdier blir ikke lagret noe sted med mindre man har kodet at programmet skal lagre dette på et nytt sted, for eksempel i en liste. "It is possible that seeing past and future values in role images migth give students the impression that all these values are available and could be accessed using some special syntax. Therefore, the teacher should stress constantly during animation sessions that only one of the values exist in reality and that the others are shown for visualization purposes only. In our experience, this is sufficient for preventing misunderstandings" (Kuittinen & Sajaniemi, 2004, s.60).

M11: Tildeling av verdi virker i motsatt retning

M11 handler om en manglende forståelse for hvordan en variabel får tildelt en verdi via en annen variabel. Man kan da streve med å forstå hvilken av variablene som får ny verdi. Dette gjør at koder som [sett tall1 til tall2] blir problematisk å forstå. Man vil da ha problemer med å forstå om det er tall1 som får verdien til tall2, eller om det er tall2 som får verdien til tall1.

M12: Tildeling av verdi virker i begge retninger

M12 handler om en manglende forståelse for hvordan en variabel får tildelt en verdi via en annen variabel. Dersom en variabel får en verdi via en annen variabel, kan elever feilaktig anta at variablene bytter verdi, altså at tildelingen virker i begge retninger.

M15: Variabler som tildeles en verdi via et uttrykk lagrer likninger eller uløste uttrykk
M15 handler om vansker med å forstå variabelens verdi når den får en ny verdi via et uttrykk. Man antar da at det er selve uttrykket som lagres, og ikke selve verdien. Jeg har valgt å kalle denne misoppfatningen for M15b, mens en misoppfatning hvor man antar at det er variabelens navn som lagres har jeg kalt for M15a.

I programmering vil [x=x+2] bety at x øker med 2. På venstre side av likhetstegnet behandles x som en lokasjon for å lagre noe, mens x på høyre side behandles som en verdi. Siden x behandles ulikt på venstre og høyre side kan dette skape misforståelser for nybegynnere i programmering. At notasjonen som brukes i programmeringen forstås annerledes enn i matematikken kan skape forvirring. I Scratch vil den tilsvarende koden bli skrevet som [sett x til x+2], og ved å bruke ord fremfor tegn som gir ulik mening i matematikk og programmering kan man håpe at denne vansken ikke blir dominerende.

M16: Tildeling av en verdi flytter en verdi fra en variabel til en annen
M16 handler om en manglende forståelse for hvordan en variabel får tildelt en verdi fra en annen variabel. Dersom en variabel A får samme verdi som en annen variabel B fra koden [sett A til B], kan man feilaktig anta at variabelen B blir stående tom etterpå. Man kan ha en oppfatning om at verdien ikke kopieres over, men flyttes fra en variabel til en annen. Da oppfatter man at variabelen B blir tom eller gis verdien 0.

M17: Det valgte variabelnavnet vil påvirke hvilken verdi som går inn i hvilken variabel
Denne misoppfatningen går ut på at programmereren antar at variabelnavnet vil påvirke hvilken variabel som får hvilken verdi. Man tenker da at datamaskinen automatisk velger hvilken verdi som går til hvilken variabel, ut ifra variabelens navn. Om man for eksempel har en variabel StortTall og en variabel LiteTall, tenker man seg at den største verdien automatisk går til StortTall og den minste verdien til LiteTall. En slik oppfatning tilsvarer misoppfatning M17.

M18: Rekkefølgen man oppretter og navngir variablene vil påvirke hvilken verdi som går til hvilken variabel
Misoppfatningen handler om at rekkefølgen man oppretter og navngir variabler i vil påvirke hvilken verdi som går til hvilken variabel. Dette kan man se i output, der man feilaktig vil oppgi verdiene i den rekkefølgen som variablene ble deklarert.

M20: Misoppfatninger knyttet til å endre en variabel
M20 handler om vansker med å forstå hvordan en variabel endrer verdi. Denne vansken kan oppstå når en variabel endrer verdi via et uttrykk, men ikke når en variabel oppdateres eller initialiseres.

M23: Vansker med å forstå betydningen av rekkefølgen på koden
Denne misoppfatningen handler ikke direkte om variabler, men om hvilken rolle kodens rekkefølge har å si. I dette prosjektet har jeg tatt med M23 og knyttet den sammen med bruk av variabler.

3 Metode

I dette kapittelet vil jeg presentere valg av metode og forskningsdesign for forskningsprosjektet, og jeg vil gjøre rede for valg som er tatt underveis i prosessen. Jeg vil beskrive hvordan datamaterialet er innhentet og hvordan dette er blitt bearbeidet og analysert. Jeg vil kommentere studiens datakvalitet og har et eget delkapittel for metodekritikk. Til slutt vil jeg vise hvordan jeg har fulgt de etiske retningslinjer for personvern og gjennomføring av prosjektet.

3.1 Valg av metode og forskningsdesign - Aksjonsforskning

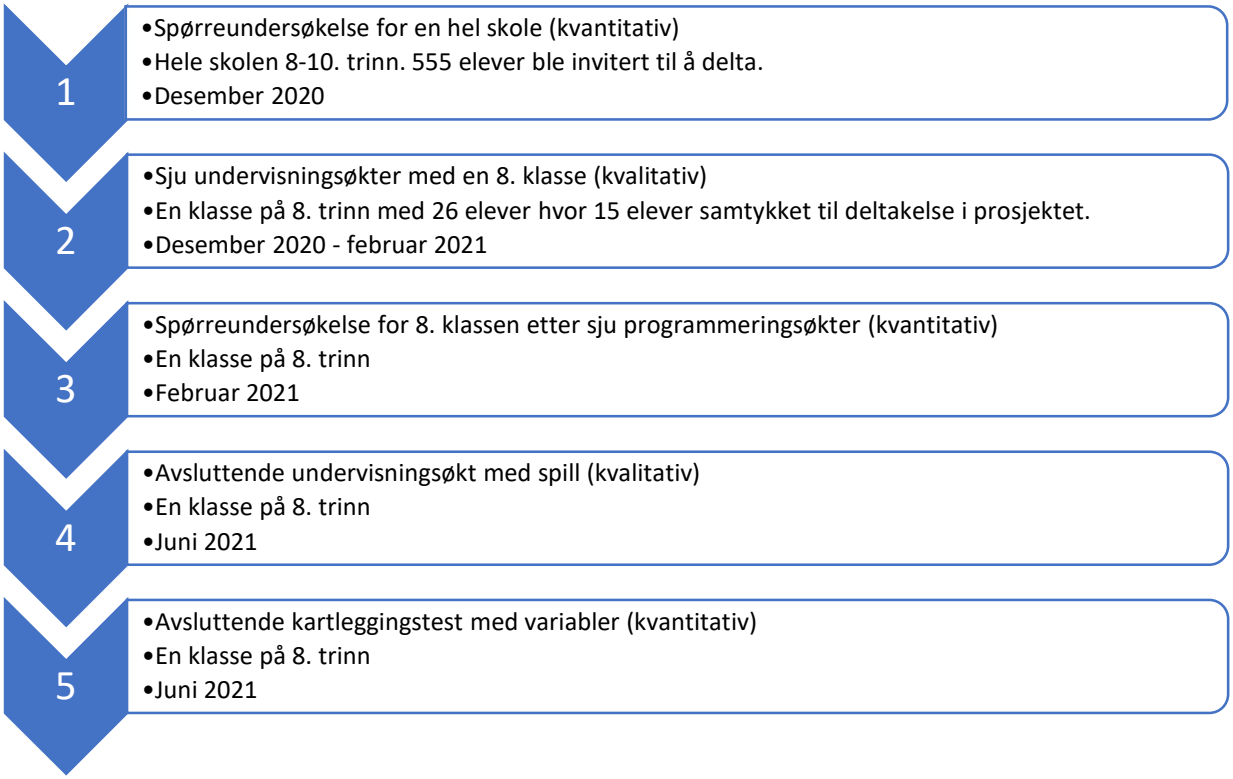
Creswell (2012, s.577) beskriver aksjonsforskning som en forskningsmetode som passer godt for lærere som ønsker å forske på egen praksis og elevers læring. «Gjennom aksjonsforskning granskes undervisning og læring fra innsiden». (Roness, 2016, s.17). Siden jeg jobber som lærer, og ønsket et masterprosjekt som var relevant for egen undervisningspraksis, landet jeg på å bruke aksjonsforskning i mitt masterprosjekt. Ønsket om å utvikle meg selv og lære mer om programmering i matematikkundervisning passet godt sammen med å bruke aksjonsforskning som forskningsdesign. Dette innebar først og fremst at jeg måtte lære meg litt om programmering selv. Deretter tenkte jeg å planlegge og gjennomføre et undervisningsopplegg i en klasse, og deretter analysere og drøfte resultatene. I et slikt forskningsdesign kan man ha stor metodefrihet og justere underveis som man ser behov for det. Dette ble nødvendig, for jeg visste lite om hva som ventet da jeg startet opp med den første programmeringstimen. Erfaringer fra hver programmeringsøkt og fra hver datainnsamling ble dermed med på å forme neste ledd i forskningsprosessen.

Jeg har samlet inn data på flere ulike måter, noe som gjør at prosjektet inneholder både kvantitative og kvalitative forskningsmetoder. Prosjektet har på ingen måte fulgt et strengt og rigid forskningsdesign, men jeg har tillatt meg selv å ha stor metodefrihet og tilpasset prosjektet etter hvert som jeg har gjort noen observasjoner og erfaringer. For å kunne arbeide etter en slik metode krevde det at jeg hadde muligheten til å gjøre innsamlingen av data over en lengre tidsperiode, noe jeg heldigvis fikk muligheten til. En slik veksling mellom kvalitative og kvantitative metoder resulterer i det som kalles metodetriangulering, og denne kombinasjonen vil til sammen kunne danne et bredere og mer allsidig bilde av problemstillingen. «Triangulation is the process of corroborating evidence from different individuals (e.g., a principal and a student), types of data (e.g., observational fieldnotes and interviews), or methods of data collection (e.g., documents and interviews) in descriptions and themes in qualitative research. The inquirer examines each information source and finds evidence to support a theme». (Creswell, 2012, s. 259). I denne studien har jeg benyttet både kvantitative og kvalitative metoder for å belyse samme fenomen. Datamaterialet består av spørreundersøkelser, skjerm- og lydopptak, logger og en diagnostisk kartleggingstest. Jeg håper at helheten til sammen skal kunne være med å belyse valgt problemstilling og forskningsspørsmålene som er formulert innledningsvis på en hensiktsmessig måte. Creswell beskriver i tillegg at metodetriangulering vil øke nøyaktigheten til studien, siden datamaterialet har ikke bare én kilde med informasjon. (Creswell, 2012, s. 259).

Aksjonsforskning er "en strategi der læreren setter fokus på noe man ønsker å utvikle eller forbedre i sin praksis". (Roness, 2016, s.59). I mitt tilfelle, hvor jeg ikke hadde undervist noe

i programmering før jeg startet med denne studien, vil jeg naturlig nok ikke kunne trekke inn alle ledd i å forbedre egen undervisningspraksis innenfor dette området. Likevel vil forskningen være nyttig for min egen undervisningspraksis, og gi meg erfaringer knyttet til bruk av programmering i matematikkfaget og hvilke misoppfatninger og andre vansker som kan oppstå når elever møter variabler i programmeringen. Å bli kjent med elevenes misoppfatninger vil gjøre meg selv i bedre stand til å hjelpe dem videre i læringsprosessen. "Identifying and addressing students` misconceptions is a key part of a computer science teacher`s competence". (Qian & Lehman, 2017, s.1).

Etter å ha gjort de ovenfornevnte betraktninger påbegynte jeg planleggingen av et forskningsprosjekt med et bredt innsamlingsmateriale. I delkapittel 3.2 går jeg nærmere inn på de ulike datainnsamlingene som ligger til grunn i prosjektet. Før jeg beskriver hvert enkelt trinn i prosessen vil jeg vise til figur 2, som kortfattet illustrerer hvilke data jeg har samlet inn, hvem som har deltatt og når det er blitt gjennomført.



Figur 2: Skjematisk oversikt over innhenting av datamateriale

3.2 Utvalg og innhenting av data

Dette delkapittelet viser hvordan utvalg og innhenting av datamaterialet har funnet sted.

3.2.1 Spørreundersøkelse for en hel skole

I desember 2020 gjennomførte jeg en spørreundersøkelse om programmering for en hel ungdomsskole. Dette var en kvantitativ undersøkelse hvor de aller fleste spørsmålene var avkrysningsspørsmål. Undersøkelsen var frivillig, anonym og digital. Jeg benyttet meg av Google Skjema fordi dette er et verktøy som skolen ellers bruker i sin skolehverdag og som

elevene er kjent med. Dersom jeg skulle spurt om personsensitive opplysninger ville jeg ha undersøkt om det fantes en enda sikrere metode å samle inn data på, men til dette formålet vurderte jeg at Google Skjema kunne anvendes til både elevenes og min egen fordel. Jeg synes at innsamlingen av data på den måten ble effektiv og enkel for meg å håndtere i etterkant, og det var enkelt å dele spørreskjema med valgt målgruppe.

Målet med undersøkelsen var å undersøke hvilke erfaringer elevene hadde med programmering og kartlegge deres generelle oppfatninger om programmering. Det var fokus på om de hadde noe erfaring med programmering fra før, eventuelt hvilke programmeringsspråk de kjente til, og hvilke holdninger de hadde til programmering. Ved å kartlegge elevenes tidligere erfaringer med programmering kunne jeg få et grunnlag for å planlegge passende undervisningsøkter for en tilfeldig utvalgt klasse. Se vedlegg C for utformingen av spørreskjemaet.

3.2.2 Sju undervisningsøkter med en 8. klasse

I perioden desember 2020 – februar 2021 var jeg på besøk i en 8. klasse og gjennomførte sju undervisningsøkter à 60 minutter innen blokkbasert programmering. Vi brukte Scratch som programmeringsspråk. Se vedlegg D for detaljert oversikt over hva innholdet i disse øktene var, og tabell 2 for en kort gjennomgang av innholdet i øktene.

Undervisningsøkter	Innhold
Økt 1 – Window patterns	Elevene klippet ut regulære mangekanter og fulgte bestemte algoritmer for å lime dem sammen. Resultatet ble ulike stjerner og mønster som elevene dekorerte vinduene med.
Økt 2 – Geometriske mønstre	Elevene lagde programmer i Scratch som tegnet regulære mangekanter og andre mønstre. Et av målene var å få programmet til å tegne de samme mønstrene som kom frem i økt 1.
Økt 3 – Geometriske mønstre fortsettelse	Fortsettelse og forlenging av økt 2.
Økt 4 – Puslespill. Omkrets av et rektangel	Elevene fikk utdelt programkoder som de skulle pusle sammen til et program som kunne beregne omkretsen til et rektangel.
Økt 5 – Kunnskapsspill	Elevene lagde et spill hvor en avatar forflyttet seg fra start til mål og fikk spørsmål underveis. Dersom man besvarte spørsmålene korrekt, ville avataren forflytte seg et steg nærmere mål.
Økt 6 – Kunnskapsspill fortsettelse	Fortsettelse og forlenging av økt 5.
Økt 7 – Tallrekker	Elevene lagde programmer som listet opp tall fra ulike tallrekker.

Tabell 2: Oversikt over undervisningsøktene

Elevene ble inndelt i grupper med tre deltakere på hver gruppe. På grunn av fravær og andre sosiale grunner ble gruppene noen ganger satt sammen litt annerledes, men i utgangspunktet tilstrebet vi å ha tre deltakere på hver gruppe. I timene ble det tatt skjermopptak av deres arbeid med programmeringen og det ble tatt lydopptak av deres samtaler. Jeg valgte å ikke ta med videoopptak av elevene for jeg tenkte at det ville være færre som hadde lyst til å delta i

prosjektet om dette skulle være et krav. Cirka halve klassen samtykket til å delta i prosjektet, men hele klassen gjennomførte undervisningsopplegget. Vi benyttet oss av muligheten til å dele klassen opp i to ulike rom, slik at de som tok opptak og de som ikke tok opptak satt i ulike rom. Klassens faste matematikklærer var med hele tiden, slik at dette lot seg gjennomføre. Jeg gikk mellom rommene for å hjelpe elevene i gang med arbeidet og for å veilede underveis. I timene hvor det ikke ble gjort opptak satt elevene samlet i ett klasserom. Etter hver undervisningsøkt leverte hver gruppe inn en skriftlig logg med svar på noen spørsmål knyttet til økten og ellers generelle tilbakemeldinger på hva som hadde vært vanskelig eller motiverende.

3.2.3 Spørreundersøkelse for 8. klassen etter sju programmeringsøkter

Etter de sju undervisningsøktene var jeg interessert i å vite mer om hvordan elevene selv hadde opplevd programmeringsøktene. Et kort spørreskjema knyttet til dette ble gjennomført for den aktuelle klassen. Undersøkelsen ble gjennomført via Google Skjema og resultatene ble anonymisert. Deltakelse var frivillig, men alle elevene i klassen valgte å svare. Se vedlegg E for utformingen av spørreskjemaet. En del av resultatene fra spørreundersøkelsen var med på å utforme en siste og avsluttende undervisningsøkt med klassen.

3.2.4 Avsluttende undervisningsøkt med variabler i spill

Etter å ha sett gjennom datamaterialet fra de sju undervisningsøktene, ble jeg oppmerksom på at mange elever hadde problemer med å ta i bruk variabler og forstå variablenes rolle i programmeringen. Av den grunn planla jeg en siste undervisningsøkt med klassen som kunne spisses ytterligere inn mot bruk av variabler. Siden logger og samtaler viste at elevene foretrakk øktene med spill, så valgte jeg å legge opp til at den siste undervisningsøkten skulle være knyttet til et spill. Utformingen av undervisningsøkten finnes i vedlegg D.

Elevene fikk utlevert et ferdig programmert spill hvor de skulle forbedre koden ved å legge til poenggivning. For å klare dette ville de måtte benytte seg av variabler som endret verdi hver gang en ape i spillet spiste bananer. I tillegg var det mulig å utvide programmet til at man startet med tre liv, og at man mistet et liv hver gang apen berørte et insekt. Da ville elevene måtte opprette enda en variabel og arbeide med flere variabler på samme tid.

Som i de forrige undervisningsøktene ble det også denne gangen tatt skjermopptak, lydopptak og levert logger på slutten av timen.

3.2.5 Avsluttende kartleggingstest med variabler

Siste datainnsamling i dette masterprosjektet var en avsluttende kartleggingstest med 13 diagnostiske oppgaver. Alle elevene i besøksklassen deltok på denne testen. Den var utformet som en diagnostisk prøve med to formål:

1) Å kartlegge om elevene hadde noen av de kjente misoppfatninger innen variabelbruk i programmering som nevnt i tabell 1 i kap. 2.3.9.
2) Å gi elevene og læreren deres en vurderingssituasjon som kunne gi en tilbakemelding på hva de mestret godt og hva de burde øve mer på.

Testen ble gjennomført i Google Skjema og var ikke anonym. Begrunnelsen for dette var at klassens faglærer skulle få en tilbakemelding på hvordan hver enkelt av elevene hadde scoret på testen. Elever som ikke hadde samtykket til deltakelse i prosjektet fikk muligheten til å gjøre testen uten at deres resultat ville bli tatt med i prosjektet. Jeg landet på denne løsningen

for at alle elevene skulle kunne få en tilbakemelding på testen og samtidig kunne velge om de ville at deres resultater skulle inkluderes i prosjektet eller ikke. Testen ble anonymisert før jeg startet med å analysere resultatene.

Diagnostiske oppgaver har til hensikt «å identifisere og fremheve misoppfatninger som elevene har utviklet, også *uten* at det trenger å ha vært noen formell undervisning i det en vil undersøke, å gi læreren informasjon om løsningsstrategier elevene bruker for ulike typer oppgaver, å rette undervisningen mot å framheve misoppfatningene, for på den måten å overvinne dem og de delvise begrepene, å utvikle elevenes eksisterende løsningsstrategier og å måle hvordan undervisningen har hjulpet elevene til å overvinne misoppfatningene ved å bruke de samme oppgavene før og etter undervisningssekvensen.» (Brekke, 2002, s. 4). De 13 diagnostiske oppgavene som ble brukt i dette prosjektet var planlagt og utformet slik at det var vanskelig å få riktig svar uten at man hadde en korrekt forståelse for hvordan variabelens verdi ble tildelt og fikk endret verdi. Det er viktig å «unngå å stille spørsmål der elevene kan få riktig svar selv om de har feilaktige ideer knyttet til begrepet.» (Brekke, 2002, s. 5). Utformingen av kartleggingstesten som ble brukt i dette prosjektet finnes i vedlegg F.

3.3 Bearbeiding og analyse av data

3.3.1 Hvilke data skal vektlegges?

Mye data skulle bearbeides og analyseres etter at besøkene i 8. klassen var avsluttet. Det var naturlig å gå raskt gjennom alt datamaterialet underveis i prosessen, siden jeg tok utgangspunkt i funnene jeg gjorde for å drive prosjektet videre. Likevel var det først i etterkant av besøkene at jeg startet en strukturert gjennomgang av datamaterialet. Jeg har vært gjennom alt datamaterialet, men har i denne oppgaven valgt å presentere det som er relevant for problemstillingen som er valgt for prosjektet. Datamaterialet som blir lagt mest vekt på i denne studien er det som viser noe om elevenes bruk av og forståelse for variabler i programmeringen.

3.3.2 Spørreundersøkelser

De to spørreundersøkelsene som ble gjennomført i Google Skjema har vært viktig for å belyse det første forskningsspørsmålet: «Hvilke erfaringer, holdninger og forventninger har elever på 8. trinn til programmering?» De fleste resultatene fra undersøkelsene vil derfor vektlegges i oppgaven.

Resultatene fra Google Skjema ble overført til Excel for å kunne bearbeides. For oppgavene hvor det var forhåndsgitte svaralternativer kunne resultatene veldig enkelt oppsummeres i tabeller, mens i oppgaver hvor det var mulig med tekstsvar måtte jeg gjøre en manuell sortering av data. Der det var hensiktsmessig lagde jeg diagrammer for å illustrere resultatene på en ryddig måte.

3.3.3 Skjermopptak, lydopptak og transkripsjon

Skjerm- og lydopptakene har vært viktig for å belyse forskningsspørsmålet om hvilke vansker elevene kan oppleve knyttet til variabelbruk i programmering. I denne oppgaven velger jeg å presentere et utvalg av transkripsjoner som er blitt gjort. Utvalget har til hensikt å belyse noen ulike vansker som kom frem i datamaterialet.

Etter hver undervisningsøkt leste jeg gjennom loggene og så på opptakene. Opptak som ikke var relevante for oppgavens forskningsspørsmål ble lagt til side. Dette gjaldt for eksempel opptak hvor en gruppe ikke arbeidet med oppgaven, men pratet om fotballtreningen som skulle være senere på dagen. Jeg noterte ned tidspunkter i opptakene som var av særlig interesse og hvor jeg ville gå tilbake senere for transkripsjon og analyse.

Selve transkriberingen forsøkte jeg å ta noen snarveier på. Det vil si at jeg forsøkte å få det gjort via et program på pc-en, men det viste seg at oversettelsen ikke ble god nok. Det var for mange ord som var for utydelige til å bli tolket, og elevene hadde ofte avbrytelser og overlappende samtaler, noe som kompliserte arbeidet en del. Jeg bestemte meg for å gjøre transkripsjonen selv for å ha full kontroll på arbeidet. Det viste seg at ved å gjøre dette arbeidet selv oppdaget jeg mer enn jeg ville ha gjort ved å lese en ferdig transkripsjon. Jeg kunne for eksempel høre når noe ble sagt med ironi. I et tilfelle sa en elev «det skjønte jeg», men ut ifra settingen og tonefallet var det tydelig at eleven egentlig mente det motsatte. Jeg er derfor veldig glad for at jeg endte opp med å gjøre transkriberingen selv, selv om det var veldig tidkrevende, fordi det gav meg en mye mer riktig forståelse av det som faktisk skjedde i rommet.

Til å gjennomføre transkriberingen ryddig og oversiktlig valgte jeg å bruke Excel regneark. Jeg var på forhånd blitt kjent med at UiB har tilgang på et transkripsjonsprogram som jeg kunne ha benyttet meg av, men jeg følte at jeg hadde best kontroll ved å behandle data på en måte som jeg var kjent med fra før. I første kolonne la jeg inn en kode for nummerering av observasjonen. Slik ville det alltid være lett å sortere tilbake til kronologisk rekkefølge. I andre kolonne hadde jeg en kode for hvem som snakket eller om det var en hendelse av viktig betydning på skjermopptaket. Jeg brukte kode 1, 2, 3 og L for de tre elevene på gruppen og meg selv. Kode 0 brukte jeg når det skjedde noe viktig av betydning på skjermen til gruppen. I tredje kolonne noterte jeg hva som ble sagt eller hva som skjedde på skjermen. I fjerde kolonne la jeg inn mulighet for å notere hvilke misoppfatninger og andre vansker jeg oppdaget. Videre hadde jeg en kolonne for hvilken type strategi elevene brukte. Jeg førte opp kolonner som gjorde det oversiktlig når elevene kom med spørsmål, svar, refleksjoner om egen kunnskap og oppklaringer. I siste kolonne førte jeg inn andre kommentarer som kunne ha betydning. Med denne strukturen på transkriberingen var det mulig for meg å sortere data etter flere kriterier i ettertid. Det ville også være mulig å legge inn flere kolonner med enda flere sorteringsnøkler dersom det skulle bli nødvendig på et senere tidspunkt.

3.3.5 Logger

Etter hver undervisningsøkt fylte elevene ut logger. Disse ble levert på papir. Dette ble valgt fordi det skulle bare være én pc på hver gruppe, og slik kunne man ha mulighet til å jobbe med oppgaven i Scratch samtidig som en annen på gruppen kunne fylle inn i loggen. Data fra loggene ble manuelt sortert i ettertid. Jeg skrev alt over i et dokument hvor man kan se hva alle gruppene svarte på hvert enkelt spørsmål. Slik ble det enkelt for meg å sammenligne gruppene, se om det var noe som var felles for gruppene, om de hadde problemer med det samme osv. Resultatene fra loggene vil ikke bli vektlagt i denne masteroppgaven, men de var et nyttig verktøy for meg. Gjennom loggene fikk jeg bli bedre kjent med hvordan elevene opplevde øktene, og med denne kunnskapen hadde jeg mulighet til å planlegge de neste undervisningsøktene tettere mot elevgruppens behov.

3.3.6 Avsluttende kartleggingstest med variabler

Siste del av forskningsprosjektet var en avsluttende kartleggingstest hvor elevene i 8. klassen fikk 13 diagnostiske oppgaver knyttet til variabelbruk i programmering. Resultatene fra testen er i høy grad vektlagt i masteroppgaven, og resultatene presenteres i sin helhet i kapittel 4.5. Resultatene er med på å belyse det andre forskningsspørsmålet: «Hvilke misoppfatninger og andre vansker kan elever på 8. trinn oppleve i møte med variabler i Scratch?»

Hver av de diagnostiske oppgavene hadde til hensikt å avdekke om en spesiell misoppfatning forekom eller ikke. Misoppfatningene som ble lagt til grunn handlet om tildeling og endring av verdier til variabler i Scratch, og er navngitt M9-M23 slik de er presentert av Sorva. (Sorva, 2012, s.358-368). I alle oppgavene var variablene allerede deklarert, slik at man vil ikke kunne avdekke elevers vanskeligheter med å forstå når og hvorfor variabler opprettes. Testen vil kunne avdekke misoppfatninger og andre vansker som kan oppstå når en elev skal lese og forstå et program, man den vil ikke kunne finne vansker knyttet til å skrive egne programmer, teste, feilsøke, korrigere og forbedre programkoder.

I den avsluttende kartleggingstesten har jeg tilstrebet å lage de diagnostiske oppgavene enkle og entydige. Hver oppgave har til hensikt å avdekke hver sin særegne misoppfatning. Gjennom analyse av kartleggingstesten viste det seg imidlertid at å studere bare én misoppfatning om gangen ikke alltid var mulig. Det fremkom ofte flere ulike typer misoppfatninger i en og samme oppgave, og i mange tilfeller dukket det også opp svar som ikke kunne kategoriseres under noen av de misoppfatningene som jeg hadde satt søkelyset på. Det ble derfor nødvendig for meg å opprette noen nye typer kategorier T1-T8, som kom i tillegg til de misoppfatningene som jeg hadde planlagt å studere. Beskrivelse av T1-T8 vil presenteres fortløpende når de kommer frem i resultatene, og de er også å finne i tabell 3 i kapittel 4.5.

3.4 Datakvalitet

3.4.1 Reliabilitet

«Med begrepet reliabilitet reiser vi spørsmålet om graden av målepresisjon eller målefeil.» (Befring, 2007, s.116). En studies reliabilitet sier noe om selve målingene som er gjort er presise eller ei, og dermed om selve datamaterialet som ligger til grunn er pålitelig. Creswell (2012) beskriver det slik: «Reliability means that scores from an instrument are stable and consistent.» (Creswell, 2012, s.159). Man kan bruke et termometer for å beskrive dette, slik Dag Roness (personlig kommunikasjon, 08.05.20) beskrev det på en forelesning i vitenskapelig metode. Han forklarte at dersom man bruker et gammelt og ødelagt termometer for å måle temperatur, kan man si at målingene har liten grad av reliabilitet. Man vil derfor ønske seg et nøyaktig termometer som gir riktige målinger, og i forskning vil man også ønske å ha forskningsmetoder som gir nøyaktige og pålitelige data.

I denne aksjonsforskningen har jeg forsøkt å gjøre innhentingen av datamaterialet på en presis og nøyaktig måte. I spørreskjemaene og loggene har jeg derfor tilstrebet å formulere spørsmålene slik at de ikke skulle kunne tolkes på ulike måter. Jeg har forsøkt å begrense antall spørsmål slik at elevene ikke skulle miste interessen og begynne å svare tilfeldig. Feilaktige tolkninger av spørsmålene og tilfeldige avkrysninger vil gi datamaterialet dårlig reliabilitet, og jeg har derfor forsøkt å unngå slike situasjoner.

I skjermopptak og lydopptak ville det vært mulig å få et enda bedre bilde av virkeligheten ved å legge til videoopptak. Siden videoopptak ikke var med mistet jeg muligheten til å se elevenes ansiktsuttrykk og non-verbal kommunikasjon som for eksempel peking. Jeg valgte likevel bort videoopptak, for jeg ville ha med flest mulig fra klassen i prosjektet. Det viste seg senere at dette valget kunne by på utfordringer for meg med å høre forskjell på stemmene til elevene på gruppene. Videoopptak ville ha kunne hjulpet meg til å være enda sikrere på å vite hvem det var som sa hva. Likevel mener jeg at dialogene som lydopptakene viser gir gode eksempler på elevers arbeid med programmering, og særlig når en ser lydopptakene sammen med skjermopptakene. Siden opptakene likevel skulle anonymiseres spiller det ikke noen stor rolle for meg om hvem som sa hva, bare jeg klarer å skille de ulike stemmene fra hverandre. Selv om videoopptak kunne vært med på å gjøre selve datamaterialet mer reliabelt, mener jeg å ha klart å hente inn gode nok data for det denne studien har til hensikt å avdekke. Jeg tror at jeg hadde fått mindre elever til å delta i prosjektet om det skulle være videoopptak, og er derfor fornøyd med denne avgjørelsen.

3.4.2 Validitet

Validiteten sier oss noe om gyldigheten av slutninger man har gjort på bakgrunn av resultatene i datamaterialet. Validitet er altså knyttet til slutninger og ikke metode. Målingene fra et termometer kan for eksempel ikke brukes til å gjøre slutninger om frykt. (Roness, personlig kommunikasjon, 08.05.20). Det er viktig at datamaterialet som benyttes er gyldig for den valgte problemstillingen, og at man kan vurdere om slutningene som blir gjort er gyldig i alle kontekster eller bare i noen enkelte særtilfeller.

Det finnes ulike metoder man kan benytte for å øke en studies validitet. «Validating findings means that the researcher determines the accuracy or credibility of the findings through strategies such as member checking or triangulation.» (Creswell, 2012, s. 259). Triangulering nevnes her som en metode som kan øke validiteten, og triangulering kan forklares slik: «Triangulation is the process of corroborating evidence from different individuals (e.g., a principal and a student), types of data (e.g., observational fieldnotes and interviews), or methods of data collection (e.g., documents and interviews) in descriptions and themes in qualitative research.» (Creswell, 2012, s.629). I denne studien har jeg gjennomført triangulering ved å innhente data på flere ulike måter. Dette kan være med på å øke studiens nøyaktighet og pålitelighet. Jeg har i flere tilfeller gjort samsvarende observasjoner fra de ulike datamaterialene, noe som øker studiens pålitelighet. For eksempel har jeg observert funn av M23 (Problemer med å forstå betydningen av rekkefølgen på kode) både i den avsluttende kartleggingstesten og i opptakene. I opptakene er det mulig å følge elevenes resonnering og se hvordan de endret oppfatning underveis som de arbeidet med oppgaven, mens i spørreundersøkelsene er det kun sluttsvarene deres som kommer frem. Dette gir veldig ulike datamaterialer som beskriver samme fenomen, og som til sammen gir et bedre bilde av situasjonen enn det ville vært å bruke kun et datamateriale. Å innhente ulike typer data, slik jeg har gjennomført i dette prosjektet, vil altså kunne være med på å øke studiens validitet.

Noe som kunne ha økt studiens validitet ytterligere kunne ha vært å inkludere intervju i datainnsamlingen. I den avsluttende kartleggingstesten oppsto det flere tilfeller hvor elever oppgav svar hvor det var vanskelig å forstå hvordan de hadde resonnert. En kort samtale med disse elevene i etterkant ville vært en fin metodetriangulering som kunne ha validert resultatene fra den avsluttende testen i enda større grad. På grunn av at avtaler var gjort på

forhånd, og at det var tett på sommerferie for elevene, var det vanskelig å få til en intervjurunde i etterkant.

3.4.3 Utfordringer

Opptak

Underveis i forskningsprosessen oppsto det noen utfordringer. I starten brukte jeg en del tid på å finne ut hvordan jeg skulle få gjennomført skjermopptak og lydopptak. Det var nødvendig å gjøre tester på opptak i forkant av undervisningsøktene med 8. klassen for å sikre at opptak ville fungere. Dette ble gjort sammen med noen andre elever som ikke var med i forskningsprosjektet, men som stilte seg villig til å hjelpe meg med å finne ut av dette samtidig som de fikk en litt annerledes matematikktime. Vi testet opptak både på skolens PC-er og på en privat PC. Til opptak brukte vi programmet WeVideo, et videoredigeringsprogram som skolen hadde lisens på å bruke.

Fravær

Da vi startet opp med undervisningsøktene i forskningsprosjektet brukte besøksklassen PC-er, og tre av fem grupper klarte å gjøre opptak. Neste økt fikk alle dette til å virke. Det viste seg at det var de samme elevene som tok ansvar for gruppenes opptak hver gang. Dette skapte en utfordring når vedkommende var borte og noen andre på gruppen måtte ta ansvar for opptaket. Om gruppen manglet eleven som hadde tatt ansvar for opptaket ble det ikke tatt noe opptak. Undervisningsøkt 6 gjennomførte hele klassen uten opptak, men elevene delte programmene sine med hverandre i klassegalleriet i Scratch og leverte logg. Selv om det ikke ble gjort opptak på alle gruppene hver gang, så mener jeg at det likevel er innsamlet et tilstrekkelig datamateriale til å kunne si noe om oppgavens forskningsspørsmål.

Bytte fra PC til Chromebooks

I perioden da jeg var hos besøksklassen foregikk et bytte fra PC til Chromebooks, og dette gjorde at man måtte bruke litt andre tastetrykk enn det man var vant til. Elevene brukte dermed litt ekstra tid på det tekniske fremfor det faglige i denne overgangen. Dette skal ikke ha noen betydning for resultatene i denne studien, men det tok noe av tiden som vi hadde tilgjengelig.

Korona

Koronapandemien satte sitt preg på forskningsprosessen. I perioden hvor jeg gjennomførte undervisningsøkt 1-7 vekslet skolen mellom å følge koronarestriksjoner på gult nivå, rødt nivå og deretter tilbake til gult nivå. I tillegg var det til enhver tid en risiko for at det kunne bli hjemmeskole. Dette gjorde at planleggingen av undervisningstimene, sitteplassene for elevene, fordeling i ulike rom og smittevernregler preget planleggingen av skolehverdagen. I økt 3 var klassen fordelt i to rom og de som skulle ta opptak måtte bruke munnbind siden det ikke var mulig å overholde 1-meters regelen dersom de skulle få til et samarbeid på gruppene. Slike hensyn måtte vurderes gjennom hele prosjektet fordi skolen hadde ulike retningslinjer til ulike tider. Til tross for de begrensningene som er nevnt synes jeg at vi likevel fikk gjennomført øktene på en god måte, men det er klart at gjennomføringen ble noe mer krevende enn det ville vært uten koronaepidemien. Jeg er svært takknemlig for at jeg i det hele tatt fikk lov til å besøke klassen, og på den måten kunne gjennomføre forskningsprosjektet med de begrensningene som var.

3.5 Metodekritikk

I ettertid kan det pekes på noen sider ved dette prosjektet som kunne ha vært planlagt og utført på andre og kanskje bedre måter.

Reliabilitet til adjektiver

I den første spørreundersøkelsen erfarte jeg at noen elever krysset av for motstridende svaralternativer. Dette gjaldt for spørsmålene som innebar forhåndsgitte adjektiver. Det kan være at noen elever ikke tok testen seriøst, svarte tilfeldig eller syntes at spørsmålet var meningsløst og signaliserte det ved å gi motstridende tilbakemeldinger. Jeg opplevde ikke dette i andre sammenhenger. Spørsmålene med adjektivene er kanskje det som har minst reliabilitet i denne studien, men jeg mener likevel at ved å se på adjektivene som opptrer hyppigest og minst kan man trekke noen slutninger som er pålitelige. Dersom jeg hadde gjennomført en pilotstudie eller fått en kvalitetssjekk på spørreskjema i forkant ville jeg kanskje oppdaget dette og omformulert disse to spørsmålene.

Å trekke gyldige slutninger fra diagnostiske oppgaver

I den avsluttende kartleggingstesten har jeg i stor grad tilstrebet å isolere hver misoppfatning, søkt etter denne hos elevene og presentert resultatene i lag med en tilhørende analyse. Gjennom analysen viste det seg i flere tilfeller at det kunne være utfordrende å trekke presise slutninger fra de diagnostiske oppgavene. Å lage entydige diagnostiske oppgaver som bare avdekker én bestemt misoppfatning av gangen, viste seg å være vanskelig. Allerede i den første diagnostiske oppgaven kunne svarene fra elevene tyde på fire ulike misoppfatninger i elevgruppen. I de neste oppgavene kunne noen av svarene tyde på at enkelte elever hadde flere misoppfatninger samtidig. «Det er viktig å forstå forskjellen på de feil elevene gjør, og de misoppfatningene de har. En feil kan komme mer eller mindre tilfeldig, fordi en ikke er oppmerksom nok eller leser oppgaven godt nok. Misoppfatninger er ikke tilfeldige.» (Brekke, 2002, s.10).

Det er mulig at noen av de diagnostiske oppgavene kunne ha vært forbedret. Særlig ønsker jeg å fremheve oppgaven for M17, hvor linjen [sett StortTall til LiteTall] gjør at svarene elevene gir kan utspringe fra ulike typer misoppfatninger. Jeg har likevel valgt å beholde oppgaven i gjennomgangen av resultatene og drøfter de ulike svarene elevene har gitt. I analysen vil jeg presentere misoppfatninger som det er sannsynlig at elevene kan ha. Som Brekke (ibid) poengterer er det likevel mulig at elevene har gitt de feilaktige svarene av andre grunner som jeg ikke har klart å belyse. Jeg mener likevel at oppgavene som er brukt og resultatene viser interessante funn som er knyttet til vansker ved bruk av variabler.

Utfordringer i presentasjon av data

Siden jeg har benyttet ulike former for datainnsamlinger og har vekslet mellom disse flere ganger, har det blitt noen utfordringer knyttet til hvordan jeg skulle få presentert alle dataene på en ryddig måte. Jeg kunne ha presentert kvantitative og kvalitative data hver for seg, men siden jeg i aksjonsforskningen stadig benyttet mine funn til å drive prosjektet videre, har jeg valgt å presentere resultatene i en tidsmessig kronologisk rekkefølge.

3.6 Etikk

I dette forskningsprosjektet har jeg fulgt de gjeldende retningslinjer for innsamling av data. Først la jeg inn mitt forskningsprosjekt til godkjenning hos UiB Rette. Da dette var godkjent, fikk jeg kontakt med en rektor på en ungdomsskole hvor jeg kunne få gjennomføre prosjektet. Rektor samtykket til at jeg kunne besøke skolen og gjennomføre dette prosjektet.

Videre sendte jeg ut en forespørsel til alle matematikklærere på 8. og 9. trinn på den aktuelle skolen om deres klasse ville delta i prosjektet. Det var en 8. klasse og en 9. klasse som stilte seg positiv til deltagelse, og jeg måtte velge ut en av dem. På grunn av at den første spørreundersøkelsen hadde høyest svarfrekvens på 8. trinn valgte jeg å besøke 8. klassen. Alle elevene i klassen ble informert om prosjektets gang og fikk et info- og samtykkeskriv med hjem til foresatte. De ble også informert om at de når som helst kunne trekke seg ut av prosjektet og de fikk anledning til å stille spørsmål. 15 elever samtykket til å delta i skjerm- og lydopptak.

Skjermopptak og lydopptak ble behandlet varsomt. Etter at jeg fikk lastet ned opptakene ble disse lagret på en ekstern harddisk uten tilgang til internett. Elevene ble bedt om å slette sine opptak innen kort tid, slik at de ikke hadde opptak liggende på sine datamaskiner eller i en lagringssky.

Jeg synes det var flott å få besøke en klasse hvor jeg ikke underviser selv. Dette gjorde at min rolle som forsker ikke ble forstyrret av elevenes forhold til meg som lærer. Elevene ble gjort kjent med at de ikke skulle få karakter på arbeidet sitt, men at de ville få en poengsum og tilbakemelding på en avsluttende kartleggingstest.

4 Resultat og analyse

I dette kapittelet vil jeg presentere resultatene som jeg har fått fra de gjennomførte datainnsamlingene. Jeg vil presentere resultatene i den tidsmessige kronologiske rekkefølgen slik de ble utført. Dette velger jeg å gjøre fordi i mange tilfeller var observasjoner som ble gjort underveis med på å styre hvordan de neste timene ble planlagt.

Underveis i presentasjonen av hver enkelt datainnsamling vil det komme drøftinger knyttet til funnene som er gjort. Jeg vil forsøke å analysere dette i lys av teori som er lagt til grunn i kapittel 2. Det er viktig å ha i minne at drøftingen og analysen i mange tilfeller er mine egne tanker og er knyttet til et lite utvalg elever. Denne studien alene vil dermed ikke kunne si noe om hva som gjelder generelt i landet for alle elever, men jeg håper likevel at jeg vil klare å belyse noen områder som kan være av interesse for andre også.

4.1 Spørreundersøkelse for en hel skole

I spørreundersøkelsen for hele skolen fikk jeg inn 217 svar av totalt 555 elever. På 8. trinn fikk jeg inn 102 svar. På 9. trinn fikk jeg inn 26 svar. På 10. trinn fikk jeg inn 89 svar. På grunn av lav svarprosent på 9. og 10. trinn har jeg valgt å bruke data fra kun 8. trinn i denne analysen. Det er dermed 102 svar fra elever på 8. trinn som ligger til grunn i denne første spørreundersøkelsen.

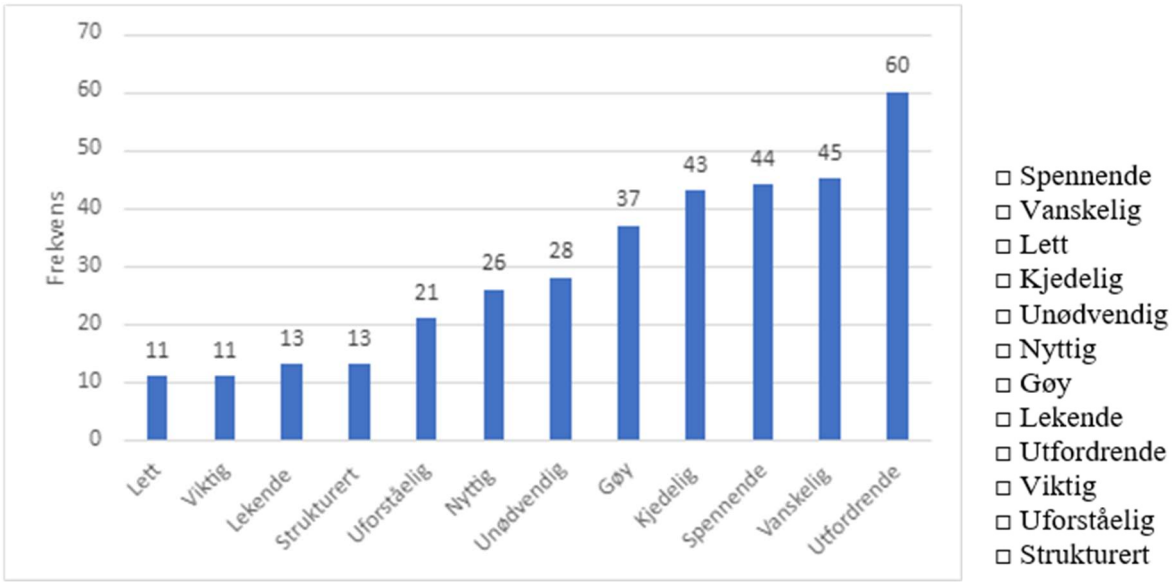
Formålet med denne spørreundersøkelsen var å finne svar på første forskningsspørsmål: Hvilke erfaringer, holdninger og forventninger har elever på 8. trinn til programmering? Denne spørreundersøkelsen har også vært med på å gjøre en del kartlegginger for at jeg selv skulle bli i bedre stand til å planlegge undervisningsøkter for elevene og legge oppgavene på et passende faglig nivå. Det var viktig for meg å gå bredt ut i starten for å senere spisse meg inn mot et mer spesifikt tema.

De to første spørsmålene i undersøkelsen handlet om klassetrinn og kjønn, slik at jeg senere hadde mulighet til å velge ut ulike elevgrupper. Disse to spørsmålene utelater jeg når jeg presenterer resultatene, da de kun var ment for å gi meg større valgmuligheter i sortering av data.

Spørsmål 3

Tenk på ordet programmering. Hvilke av adjektivene nedenfor synes du passer til programmering? Kryss av for alle ordene som du synes passer.

Resultat:



Figur 3: Resultater fra spørsmål 3 i spørreundersøkelsen

Drøfting

Dette spørsmålet hadde som hensikt å kartlegge om elevene hadde noen bestemte holdninger til programmering. Jeg valgte å avdekke dette ved å be dem krysse av for adjektiver som de syntes passet overens med programmering. Figur 3 viser at adjektivene som flest elever assosierer med programmering er «utfordrende», «vanskelig», «spennende» og «kjedelig». Adjektivene som færrest forbinder med programmering, er «lett» og «viktig». Jeg synes at det kan være nyttig å kjenne til slike eksisterende holdninger i møte med elever som skal jobbe med programmering. Det er verdt å bemerke at i dette spørsmålet var det mulig å krysse av for flere adjektiver, og at det er 102 elever fra 8. trinn som har svart. Det er også et poeng at adjektivene var forhåndsgitte, og det var ikke mulig for elevene å lage egne adjektiver. Likevel synes jeg at resultatene gir noen interessante funn.

Blant adjektivene som handler om elevenes forventninger om vanskelighetsgrad er det 11 elever som krysser av for «lett», 21 for «uforståelig», 45 for «vanskelig» og 60 for «utfordrende». Det er tydelig at det er mange flere elever som forventer at programmeringen skal være «vanskelig» og «utfordrende» enn elever som tenker at det skal være «lett». Av denne grunn tror jeg at det vil være viktig å planlegge undervisningstimer på en enkel måte slik at absolutt alle skal kunne få positive mestringsfølelser helt fra starten av. Samtidig er det flott om de som faktisk synes at det er lett får mulighet til å få noen utfordringer videre, slik at alle får noe som er tilpasset sine forutsetninger. Dette er det tatt høyde for i undervisningsopplegget som presenteres senere, hvor det er noen felles og grunnleggende oppgaver i starten, men hvor man ganske raskt kan gå videre til vanskeligere nivå og mer åpne oppgaver.

Videre er det noen adjektiver som går ut på nyttegraden av å drive på med programmering. 11 elever krysser av for «viktig», 26 for «nyttig» og 28 for «unødvendig». Minst 37 elever har ikke valgt noen av disse adjektivene, og dermed kan vi anta at denne gruppen ikke har gjort seg opp en mening om hvorvidt programmering er viktig eller ikke, eller at det ikke var noen mulige passende adjektiver som de ønsket å krysse av for. Adjektivene som var forhåndsbestemte var ganske bestemte og spissformulerte og det kan godt tenkes at mange elever ville valgt noe som ligger et sted imellom de valgte ytterpunktene. Det er også mulig at de ville valgt noen helt andre adjektiver som ikke er med i det mulige utvalget. Dataen tyder på at elevgruppen har ulike synspunkt angående om programmering i det hele tatt er nyttig for oss å holde på med. Av denne grunn vil jeg anta at det vil være viktig å gi elevene erfaringer med hvordan programmering kan være et nyttig verktøy for oss, slik at enda flere ser nytteverdien av programmeringen, og dermed kanskje også får økt motivasjon til å gjøre oppgaver innen programmering.

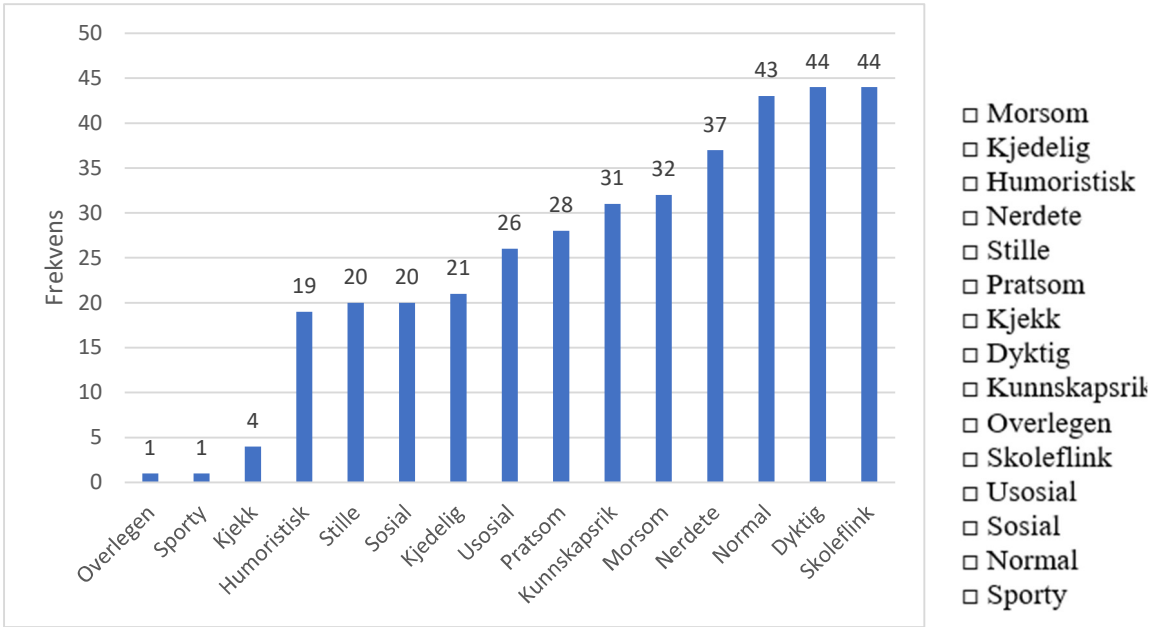
Videre er det 43 elever som har krysset av for «kjedelig», noe som kan stå i motsetning til 13 svar for «lekende», 37 for «gøy» og 44 for «spennende». Det kan dermed se ut til at elevgruppen er noe splittet i sin motivasjon for programmering, og jeg håper å kunne være med til å bidra til at flere skal synes at programmering er gøy.

Jeg har i ettertid sett at spørsmål 3 kunne med fordel ha vært inndelt i tre ulike spørsmål, noe som ville ha gjort analysen enklere og mer entydig. Valget om å holde antall spørsmål lavt overskygget denne vinklingen da spørreundersøkelsen ble laget.

Spørsmål 4

Tenk på en person som driver med programmering. Hvilke av adjektivene nedenfor synes du passer til en person som driver med programmering? Kryss av for alle ordene som du synes passer.

Resultat:



Figur 4: Resultater fra spørsmål 4 i spørreundersøkelsen

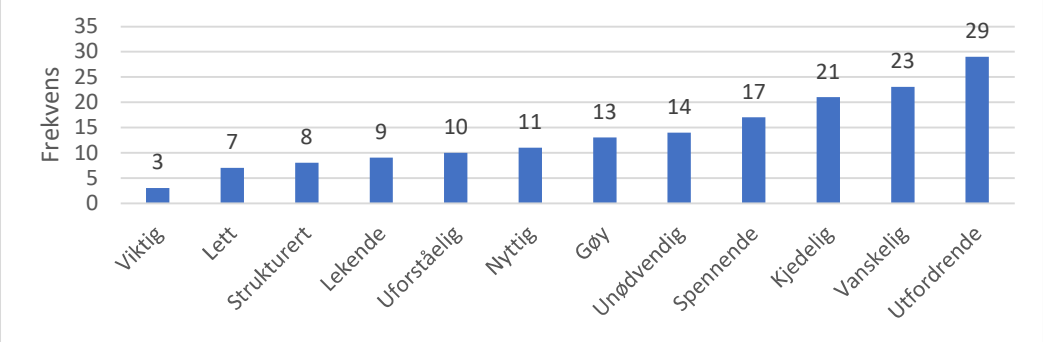
Drøfting

Dette spørsmålet har mye av de samme kriteriene som spørsmål 3, hvor det er forhåndsgitte adjektiver og mulig å krysse av for flere alternativer. Her ser vi nærmere på elevers syn på en person som driver med programmering.

Adjektivene som scorer høyest i dette spørsmålet er «skoleflink», «dyktig» og «normal», mens adjektivene som elevene knytter minst til en programmerer er «overlegen», «sporty» og «kjekk». Ved å se nærmere på noen enkeltbesvarelser dukket det opp noen motstridende svar. Det viste seg at det var flere som hadde krysset av for «usosial» og «sosial» samtidig. Flere hadde også krysset av for «morsom» og «kjedelig» samtidig. En mulig forklaring på dette kan være at noen elever kan ha funnet spørsmålet noe meningsløst og dermed valgt å krysse av for motstridende adjektiver. På en måte er dette ganske greit å erfare, for jeg ønsker ikke at elever skal ha stereotypiske holdninger ovenfor programmerere. Samtidig kan man undre seg over om det i det hele tatt er mulig å trekke noen slutninger som er valide når de samme elevene svarer slike motstridende svar.

Det er likevel verdt å trekke frem at 43% av elevene krysset av for at en som driver med programmering forbindes med å være «skoleflink». Jeg ønsker at alle skal føle at de har noe å bidra med i programmeringsøktene, og vil forsøke å legge opp timene slik at alle skal oppleve mestring, uansett om de anser deg selv som «skoleflink» eller ikke. Det er underlig at mange av elevene har en forventning om at en programmerer må være «skoleflink», for programmering har ikke vært i norsk skole veldig lenge, og mange av elevene har manglende erfaring med programmering. At programmering i stor grad knyttes til skole skulle på ingen måte være en selvfølge, da det frem til LK20 ikke har vært en del av læreplaner i hverken grunnskole eller videregående skole. Av denne grunn har barn og ungdom som har drevet med programmering før denne tid gjort det uavhengig av skole og kun ut ifra egen interesse. I denne undersøkelsen ser det likevel ut til at elevene forbinder en programmerer med adjektivet «skoleflink».

Videre har jeg sett nærmere på gruppen som krysset av for adjektivet «skoleflink», siden dette adjektivet hadde en uventet høy score. Av de 44 elevene som knyttet egenskapen «skoleflink» til en programmerer, var det 29 som mente at programmering var «utfordrende», 23 som mente at det var «vanskelig» og 21 som krysset av for «kjedelig». Bare 3 av elevene som koblet programmering til adjektivet «skoleflink» mente at programmering var «viktig». Hele oversikten vises i figur 5. Resultatene tyder på at mange elever knytter programmering til skole, men at de likevel ikke synes at det er viktig.



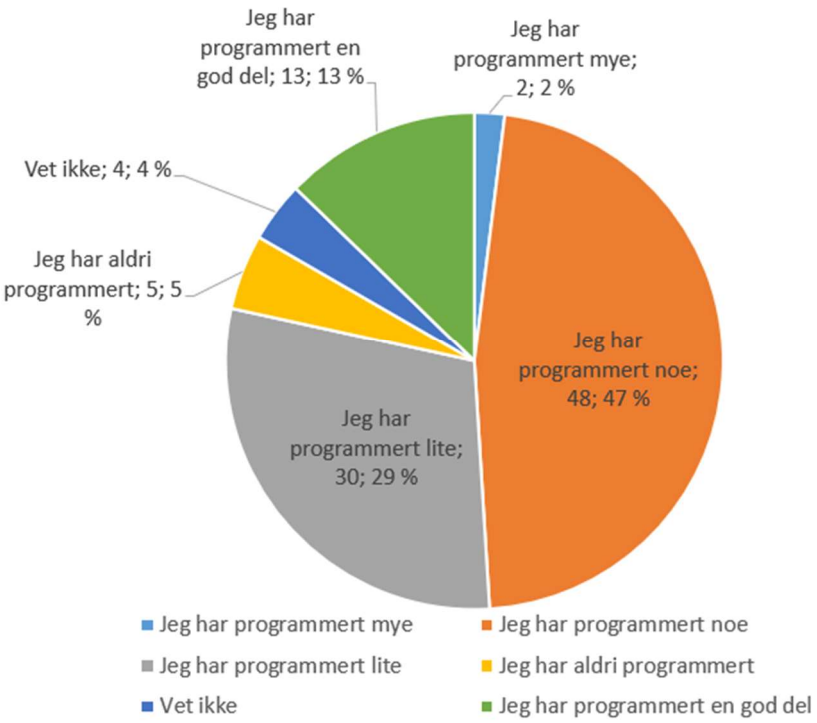
Figur 5: Resultater fra elevene som valgte «skoleflink» i spørsmål 4

Spørsmål 5

Har du selv noe erfaring med programmering? Kryss av for det alternativet som passer best for deg.

□ Jeg har aldri programmert.
□ Jeg har programmert lite.
□ Jeg har programmert noe.
□ Jeg har programmert en god del.
□ Jeg har programmert mye.
□ Vet ikke.

Resultat:



Figur 6: Resultater fra spørsmål 5 i spørreundersøkelsen

Drøfting

Av de 102 elevene som har svart på undersøkelsen var det bare én elev som hadde hoppet over dette spørsmålet. Denne eleven har jeg plassert under "vet ikke" slik at alle 102 elevene er representert i sektordiagrammet.

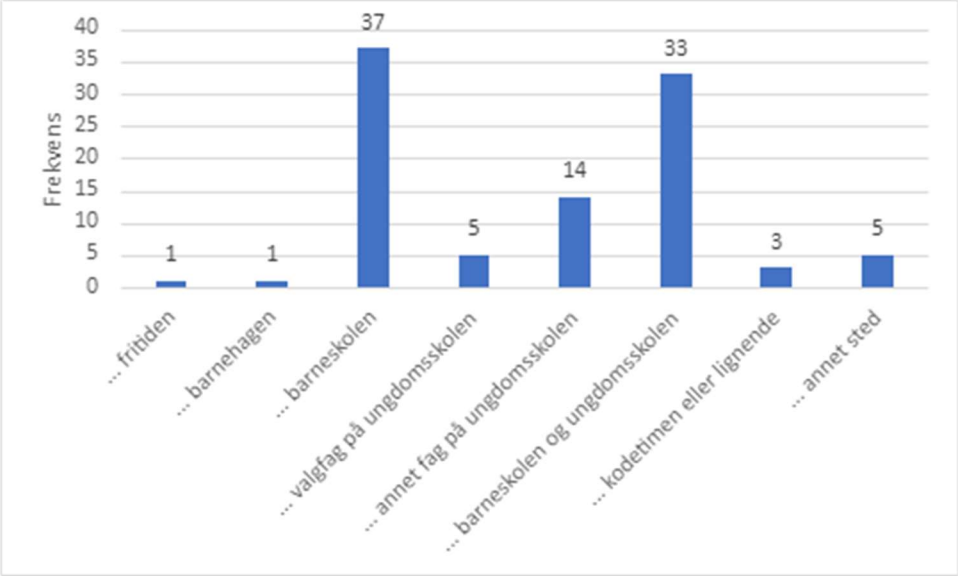
39 av 102 elevene på 8. trinn, noe som tilsvarer 38% av elevene, har programmert lite, aldri eller vet ikke om de noensinne har programmert. Dette viser at over en tredjedel av elevene som startet på 8. trinn høsten 2020 i stor grad ikke har tidligere erfaringer med programmering. Dette må det tas høyde for i planleggingen av undervisningen. Jeg forventer at prosentandelen som har programmert før oppstart på ungdomsskolen vil øke kraftig i årene fremover.

Spørsmål 6

Hvis du har noe erfaring med programmering, hvor og når jobbet du med dette? Det er mulig å krysse av på flere alternativer. Jeg har programmert…

□ … på fritiden
□ … i barnehagen
□ … på barneskolen
□ … i valgfag på ungdomsskolen
□ … i et annet fag på ungdomsskolen
□ … i kodetimen eller lignende
□ … et annet sted

Resultater:



Figur 7: Resultater fra spørsmål 6 i spørreundersøkelsen

Drøfting

Figur 7 viser at blant de elevene som hadde noen erfaring med programmering var det 60,8% som hadde denne erfaringen fra barneskolen. Dette viser at barneskolene allerede på dette tidspunktet hadde begynt å implementere programmering i sin undervisning. Diagrammet viser også at noen elever har begynt å bruke programmering på ungdomsskolen. Det er å på sin plass å påpeke at resultatene avhenger av hvordan elevene selv definerer ordet programmering, noe de ikke ble spurt om i denne undersøkelsen, og at det er mulig at resultatene ville sett annerledes ut om man hadde spurt deres lærere fra både barneskole og ungdomsskolen.

Det skapte noen utfordringer at jeg hadde åpnet opp for at elevene kunne oppgi flere svar på dette spørsmålet. Det ble derfor i ettertid nødvendig å lage en egen kolonne for de som hadde programmert både på barneskolen og ungdomsskolen. I denne kolonnen er det noen som har programmert i valgfag, noen i et annet fag og noen i begge, i tillegg til å ha programmert på barneskolen. For de som hadde programmert både på fritiden og på et skolenivå valgte jeg å plassere de på skolenivå. For de som hadde programmert kun i barnehagen, kun på fritiden

eller kun i kodetimen ble de plassert i sine respektive kolonner, men om de hadde disse i kombinasjon med programmering på skole ble de plassert på skolen.

Det er totalt 99 elever som har svart på dette spørsmålet. Det vil si at det er 3 elever som ikke har svart, og som jeg antar at ikke har programmert noe før. I spørsmål 3 er det 5 elever som krysser av for at de aldri har programmert, og det kan tenkes at to av de har ombestemt seg underveis i undersøkelsen. Dette viser meg at noen elever sannsynligvis har glemt at de har programmert før, eller kanskje de ikke har skjønt at de har programmert, men at det kan komme tilbake til minnet deres når de får tenkt seg litt om. Det kan også være at noen elever har trykket fort og tilfeldig gjennom spørreskjema. 37 elever har hatt programmering kun på barneskolen, 19 elever har hatt det kun på ungdomsskolen, mens 33 har hatt det både på barneskolen og ungdomsskolen. Dette betyr at 70 elever har hatt programmering på barneskolen, ca. 69% av elevene. Sett i lys av forrige spørsmål er det gjerne variasjon i hvor stor grad elevene har jobbet med programmering på barneskolen. Uansett vil det på dette tidspunktet være viktig å møte elevgruppen med programmeringsøkter på nybegynnernivå og samtidig ha muligheter for utvidelser for de som mester det.

Spørsmål 7
Hvis du har erfaring med programmering, utdyp her hvilket programmeringsspråk du kjenner til:

Resultat:
Elevene fra 8. trinn som har programmert før nevner at de har kjennskap til disse programmeringsspråkene:

Micro:bit, make code, Scratch, JavaScript, Python, HEX, BASIC, c#, lego mindstorm.

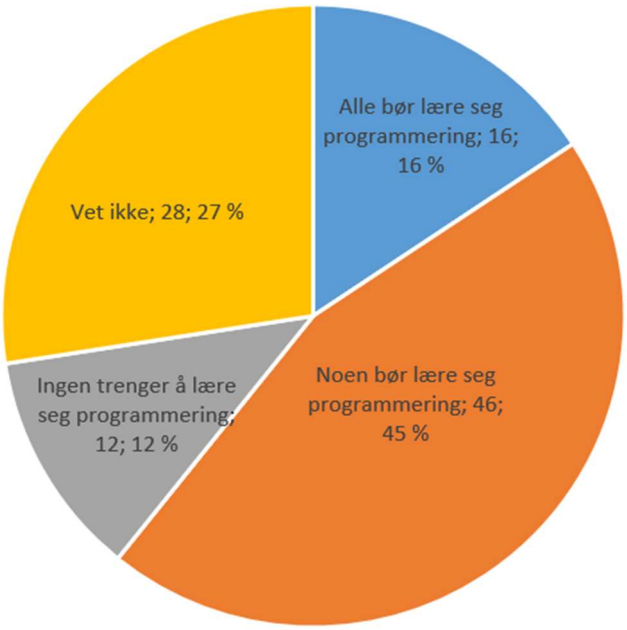
Drøfting
Av de elevene som har programmert fra før opplyser de å ha de kjennskap til ulike programmeringsspråk. Det hadde vært enklere å starte opp med programmering på ungdomsskolen dersom elevgruppen i større grad hadde noe mer likt utgangspunkt fra barneskolene. Dette var en viktig observasjon for meg med tanke på videre planlegging av undervisningsøkter og i møte med elever.

Spørsmål 8

Hvilken av påstandene nedenfor synes du er mest riktig?

□ Alle bør lære seg programmering.
□ Noen bør lære seg programmering.
□ Ingen trenger å lære seg programmering.
□ Vet ikke.

Resultat:



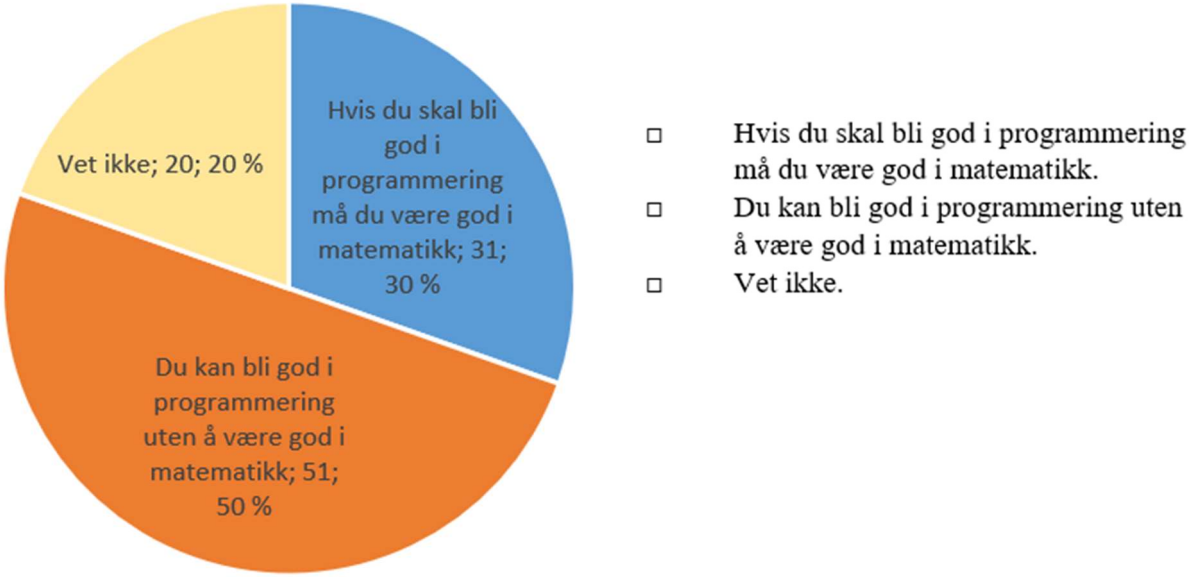
Figur 8: Resultater fra spørsmål 8 i spørreundersøkelsen

Drøfting

Dette spørsmålet handler om elevenes syn på om alle, noen eller ingen trenger å lære seg programmering. Nå som programmering er innført for alle tenkte jeg at det kunne være spennende å finne ut hva elevene selv synes om dette. Mener elevene at alle behøver programmering, slik dagens læreplan legger opp til? Figur 8 viser at 15,7% av elevene på 8. trinn mener at programmering er nødvendig for alle. 45,1% mener at noen bør lære seg programmering, og det er 11,8 % som mener at ingen trenger å lære seg programmering. 27,5%, altså over en fjerdedel av 8.klassingene, tar ikke stilling til spørsmålet og svarer at de ikke vet. Dette indikerer at de ikke har bakgrunn nok til å kunne gjøre seg opp en mening, og at de i utgangspunktet er åpen for å lære om programmering selv. At elever flest mener at programmering ikke er for alle er greit å ha med seg i møte med elever som skal jobbe med programmering. Det kan være et poeng å gi elevene erfaringer med hvordan programmering kan være nyttig for alle.

Spørsmål 9
Hvilken påstand nedenfor mener du er riktig?

Resultat:



Figur 9: Resultater fra spørsmål 9 i spørreundersøkelsen

Drøfting
Det siste spørsmålet i denne spørreundersøkelsen handler om elevenes tanker rundt sammenhengen mellom matematikk og programmering. Dette synes jeg er interessant siden vi i stor grad har lagt kompetansemålene for programmeringen i matematikkfaget. Jeg forsøkte å formulere påstandene som elevene skulle velge mellom på en entydig og forståelig måte for elevene. 30,4% av elevene mente at man må være god i matematikk for å bli god i programmering, mens 50% mente at man kan bli god i programmering uten å være god i matematikk. 19,6% av elevene svarte «vet ikke».

Det er interessant å se at 50% av elevene i denne undersøkelsen mener at man kan bli god i programmering uten å være god i matematikk. Det kan godt tenkes at en programmerer kan produsere ganske artige og kompliserte spill og andre programmer uten å ta i bruk særlig avansert matematikk. Når programmering likevel har blitt plassert i matematikkfaget kan man tenke seg og håpe på at en etter hvert også skal kunne bruke programmeringen som et verktøy i matematikkfaget. Vi skal senere se at elevene i denne undersøkelsen var mest motivert for å jobbe med spillutvikling i programmeringstimene. Det kan antyde at å knytte programmeringen til matematikk ikke var det elevene først og fremst forventet og ønsket å bruke programmeringen til.

4.2 Sju undervisningsøkter med en 8. klasse

Datamaterialet som ligger til grunn i denne studien er hentet fra skoleåret 2020-2021 da jeg besøkte en 8. klasse i åtte undervisningstimer. I seks av timene ble det tatt skjerm- og lydopptak av arbeidet deres. Det var stort sett fem grupper som deltok med opptak, men i den ene økten var det bare fire grupper som deltok på grunn av sykdom. Selv om undervisningstimene var på 60 minutter, ble opptakene stort sett på en varighet på rundt 20

minutter. Dette kan forklares med at i timene der jeg besøkte klassen var det alltid deres kontaktlærer som ønsket dem velkommen og brukte en del tid på gjennomgang av ukeplanen og annen felles informasjon. Deretter brukte jeg litt tid på å forklare dagens oppgave, og elevene brukte også en god del tid på å finne frem utstyr og logge seg inn på de ulike nettsidene som var nødvendig. Likevel har jeg totalt fått inn ganske mye skjerm- og lydopptak som er blitt gjennomgått, sortert og transkribert. I dette kapittelet vil jeg vise et utvalg av observasjoner som er gjort i disse opptakene. Selv om det var svært mye spennende å observere i disse opptakene er jeg blitt nødt til å holde et blikk på det som er relevant for denne oppgaven, og jeg viser derfor til observasjoner som kan knyttes til vansker ved bruk av variabler i programmeringen. Jeg viser her noen observasjoner fra økt 4 og økt 7 og senere i kap. 4.4 viser jeg også noen resultater fra økt 8.

Økt 4 – Omkrets puslespill - gruppe 1

I denne undervisningsøkten skulle elevene pusle sammen forhåndsgitte koder til et program som skulle beregne omkretsen til et rektangel. For å se detaljert beskrivelse av undervisningsopplegget se vedlegg D. I det følgende vises en samtale som fant sted på gruppe 1.

J1: Okey, vi tar først ... vi tok først "flagg". Så spurte vi "hva er bredden på rektangelet". Og så tok vi "sett bredden til svar", og nå skal vi ha "sett lengde til svar".

Her leste J1 koden som allerede var tastet inn, og foreslo hva neste kode måtte være.

G1: Da må vi sette ... da må vi hente en ny en da.

Eleven mente at de måtte opprette en ny variabel, men manglet ord for dette.

Figur 10: Økt 4, gruppe 1 sin begynnelse til et program

Videre opprettet de en ny variabel og navnga den til lengde. Som vist på figur 10 valgte de å gi variabelen lengde en verdi uten å ha spurt om denne først. I dette tilfellet vil både bredden og lengden bli tildelt verdien som brukeren gir som svar for bredden. Gruppen opprettet variablene lengde og bredde, som skal gis to ulike verdier, men de tildelte begge variablene samme verdi. Det er tydelig at gruppen prøvde og feilet seg frem. Det som viser seg å være vanskelig i dette tilfellet er å forstå hvordan en variabel får tildelt en verdi i brukergrensesnittet.

Figur 11: Økt 4, gruppe 1 sin fortsettelse til et program

På figur 11 kan vi se at det som skjer videre er at gruppen ba om bredden to ganger, og at koden står i en ulogisk rekkefølge. Vi ser tydelig at de ikke skjønte betydningen av rekkefølgen kodene står i. Denne observasjonen er et funn av M23.

Vi kan også se at gruppen har bedt om input for bredde to ganger i stedet for en bredde og en lengde. Videre kjørte elevene programmet og testet hvordan resultatet ble. I kjøring av

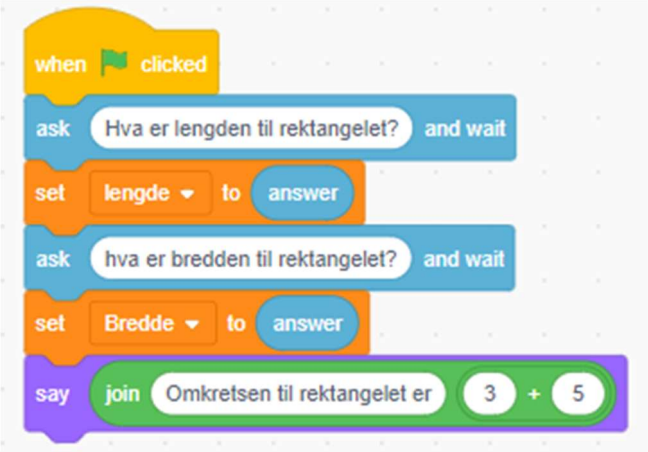
programmet fikk de opp spørsmål om bredden to ganger, og de kjørte programmet flere ganger uten å skjønne hva som var feil. Siden de ikke hadde lagt inn kode som gav noe utdata, ble det ekstra vanskelig for gruppen å spore hvor feilen var.

Omtrent på dette tidspunktet kom det informasjon fra lærer til hele klassen, slik at de som ikke var kommet veldig langt fikk noen tips til å komme i gang med arbeidet. Etter dette skjønte gruppen at man måtte spørre etter både lengde og bredde, og de endret derfor koden sin til det som visers på figur 12.

Figur 12: Økt 4, gruppe 1 sitt videre arbeid med et program

Elevene testet ikke dette programmet, og virket ganske trygg på at dette var korrekt. De opprettet deretter en variabel med navn "sett sammen" fordi denne kommandoen visste de fra puslespillet at de behøvde, men de fant den ikke i menyen. Dette viser at de opprettet en variabel som de ikke behøvde. Denne observasjonen er funn av en type vanske som går ut på at det er vanskelig å vite når man i det hele tatt skal opprette en variabel. I noen tilfeller, slik som jeg vil vise senere for gruppe 3 i økt 7, opprettes ikke variabler når det hadde vært hensiktsmessig å gjøre det, mens i dette tilfellet som vises her opprettes en variabel som ikke behøves. Variabelen [sett sammen] ble ikke brukt videre i programmet, og når de fikk vite at [sett sammen] tilsvarer det engelske [join], som var lett å finne i menyen, ble alt mye lettere for dem. Dette viser at vårt naturlige språk og oversettelser mellom engelsk, norsk og programmeringsspråk lett kan skape ekstra utfordringer for elevene. I dette tilfellet kunne vanskene vært unngått ved at elevene hadde byttet til å bruke norsk i Scratch, men av ulike grunner var det noen grupper som ønsket å ha det på engelsk uansett.

Etter at gruppen hadde fått hjelp fra lærer til å stille kodene opp i riktig rekkefølge var siste steget å få katten til å si omkretsen, og da måtte gruppen legge dette inn i koden. Resultatet deres ble som vist på figur 13.



Figur 13: Økt 4, gruppe 1 sitt ferdige program

Figur 13 viser at elevene ikke brukte variablene «lengde» og «bredde» i regnestykket på den siste linjen, men de brukte tallene som de selv hadde gitt som input i brukergrensesnittet. Her ser vi at elevene strever med å se fordelen med å bruke variabler. Dette programmet vil alltid gi samme omkrets uansett hvilke verdier som blir valgt for lengde og bredde. Dette oppdaget ikke elevene, fordi de aldri testet dette programmet videre, men følte seg trygg på at det de hadde gjort var riktig. Det som de brukte litt tid på derimot var at 3 og

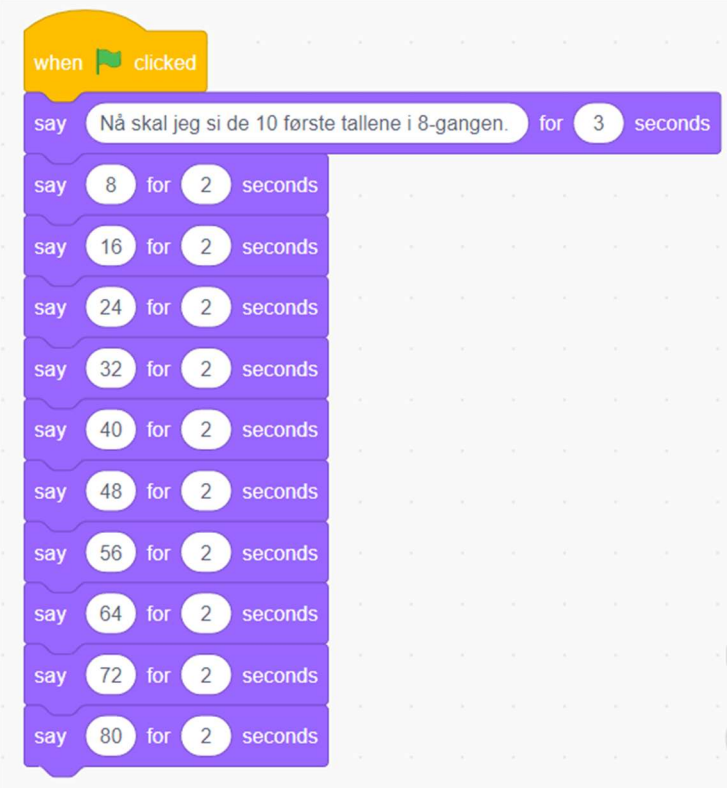
5, som var deres valg for lengde og bredde, måtte multipliseres med 2. De spurte en medelev fra en annen gruppe om hjelp for å løse dette. Medeleven forklarte steg for steg hvordan man kom i mål, og etter en kjapp test av det endelige programmet, hvor variablene ble brukt på en hensiktsmessig måte, utsprang denne samtalen:

J1: Å, så det er på en måte en kalkulator?
G1: Ja.
J1: Da kan vi jo bruke den i matten.

Dialogen tyder på at gruppen ikke skjønte poenget med å bruke variabler før helt til slutten av oppgaven. Når programmet var ferdig og fungerte som det skulle oppdaget gruppen fordelen med å bruke variabler. Elevene på gruppen vil sannsynligvis kunne lage andre slike "kalkulatorer" i fremtiden, for eksempel til å beregne areal og volum av forskjellige geometriske figurer.

Oppsummering fra økt 4: I denne økten har jeg gjort observasjoner på at det kan være vanskelig for elevene å vite når det er hensiktsmessig å opprette variabler og hvordan språket kan by på utfordringer. Jeg har også sett utfordringer med å sette opp kodene i korrekt rekkefølge og at elevene ikke testet programmet sitt godt nok etter hver endring som de hadde gjort. Det ble også gjort funn av at elevene ikke anvendte variabler i output, men heller brukte konstante tall, noe som gjør at hele poenget med å bruke variabler forsvinner. Heldigvis oppdaget gruppen, helt på slutten av økten, hvordan variablene kunne generalisere koden og gjøre et enkelttilfelle om til å kunne løse en gruppe av samme typen problemer. Det var spesielt kjekt å høre dem si at programmet kunne bli brukt som en type «kalkulator». Dette sitatet kan tyde på at de har skjønt hvordan programmering kan brukes til å generalisere et problem til å omfatte en hel gruppe av samme typen problemer, og hvordan programmering kan brukes som et verktøy i matematikkfaglige problemstillinger.

<u>Økt 7 – tallrekker – gruppe 1</u>
Denne gruppen besto av en gutt og to jenter. Jeg kaller dem for G1, J1 og J2, men samtalene som jeg vil trekke frem er kun mellom G1 og meg selv (L). Elevene snakket mye utenom faget og diskuterte lite sammen om selve oppgaven som de skulle løse sammen. Det var G1 som styrte det som skjedde på skjermen hele tiden, og det som ble forsøkt ut var ikke diskutert på gruppen. Han opprettet ganske tidlig en variabel «tall», men siden denne ikke ble brukt noe videre opprettet han den antagelig kun fordi det sto i oppgaven at dette skulle gjøres. Deretter tastet han inn koder uten at de hadde noen dialog om dette på gruppen, og han endte opp med programkoden som er vist i figur 14.



Figur 14: Økt 7, gruppe 1 sin tallrekke uten bruk av variabel

Som vi ser av figur 14 har gruppen laget et program som vil skrive ut de 10 første tallene i 8-gangen. Variabelen «tall» som ble opprettet innledningsvis er ikke tatt i bruk, men gruppen følte seg fornøyd med å ha løst oppgaven. Når jeg etter hvert kom til gruppen, skjedde følgende samtale:

G1: Vi er ferdig vi.
L: Ja, la meg se.
G1: Men jeg tror vi skal legge inn.. hm det er en forenklet versjon.
L: Ja, dere har laget en forenklet versjon.

Samtalen kan tyde på at G1 har en idé om at programmet ikke er helt som bestilt, og at det skal være mulig å lage en bedre versjon. Dette kan tyde på at han visste at variabelen «tall» skulle være en del av koden i programmet, men ikke helt skjønte hvordan den skulle brukes. Det viser seg, også her, at det kan være en utfordring for elevene å forstå når det er hensiktsmessig og når det er nødvendig å opprette variabler, og hvordan variabler skal kunne være med på å generalisere koden. I samtalen videre gav jeg gruppen tips om å generalisere programmet, slik at man kunne ha spurt etter de 100 eller 1000 første tallene. Jeg beskrev med ord hvordan de kunne benytte en variabel inni en løkke for å få ønsket resultat, men etter at jeg hadde gått til neste gruppe fikk ikke gruppen gjort noe mer på oppgaven sin annet enn å bytte bakgrunn. Neste gang jeg kom til gruppen fortsatte samtalen slik:

L: Har dere noen spørsmål?
G1: Ja. Hvordan skal vi få tallet til å øke seg?
L: Ja, da må dere gå inn på variabler, og så tar du «change my variable».
Gruppen drar inn kodene som vist på figur 15.

Figur 15: Økt 7, gruppe 1 sin tallrekke med bruk av variabel

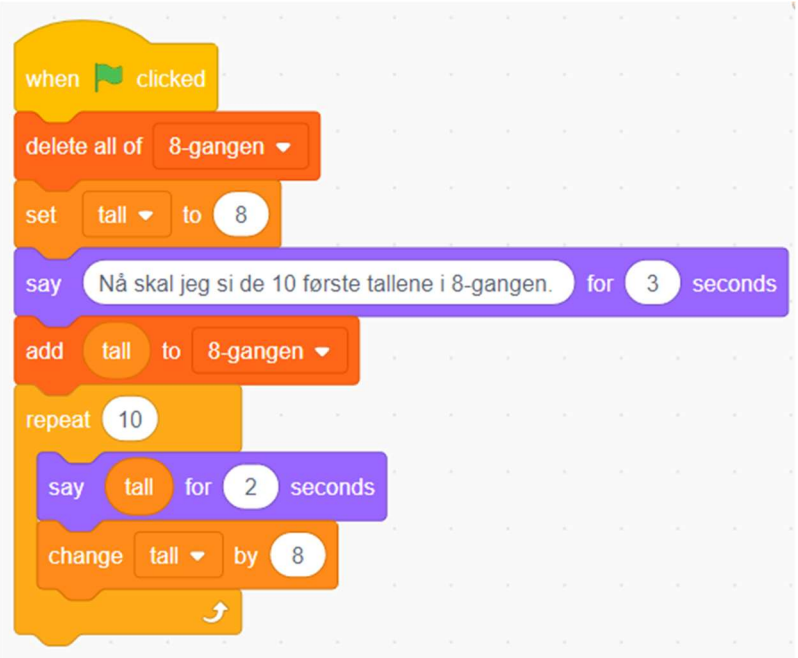
Når de kjørte dette programmet, så det ut til å virke fint. Elevene var veldig stødig på 8-gangen og hva de ville ha programmet til å gjøre, og de møtte ingen matematiske barrierer. Utfordringene deres var å få skrevet riktig kode for å få ønsket resultat. Gruppen måtte ha mye veiledning for hvert steg de skulle videre i prosessen, og det skjedde liten progresjon når jeg var opptatt med å veilede andre grupper.

Videre gav jeg dem en oppgave om å legge tallene inn i en liste. Jeg hadde tenkt at slike lister ville hjelpe elevene med å forstå tidsaspektet i kjøringen av programmet, og at det kunne være en visualisering av hvordan variabelen endret verdi under kjøringen. De opprettet en liste, men fikk ikke tallene inn i den. Etter å ha kjørt programmet noen ganger oppdaget gruppen også at programmet ikke begynte på 8 hver gang. Neste gang jeg kom til gruppen skjedde denne samtalen:

G1: Men jeg begynner på 265.
L: Ja, fordi, her oppe, før du starter på den løkken, så må du på en måte gi tallet en startverdi.
G1: Men den starter på 200.
L: Du må nesten sette tallet til ett eller annet før du starter.
G1: Ja
0: Eleven legger inn koden [set tall to 0]

Her ser vi at elevene har vansker med å initialisere variabler ved starten av et program. Dette er ifølge teorien som er presentert i kapittel 2.3.4 ikke å regne som en misoppfatning, men en vanske innen strategisk kunnskap. Likevel er initialisering av variabler noe som kan skape utfordringer for elevene i arbeidet med variabler.

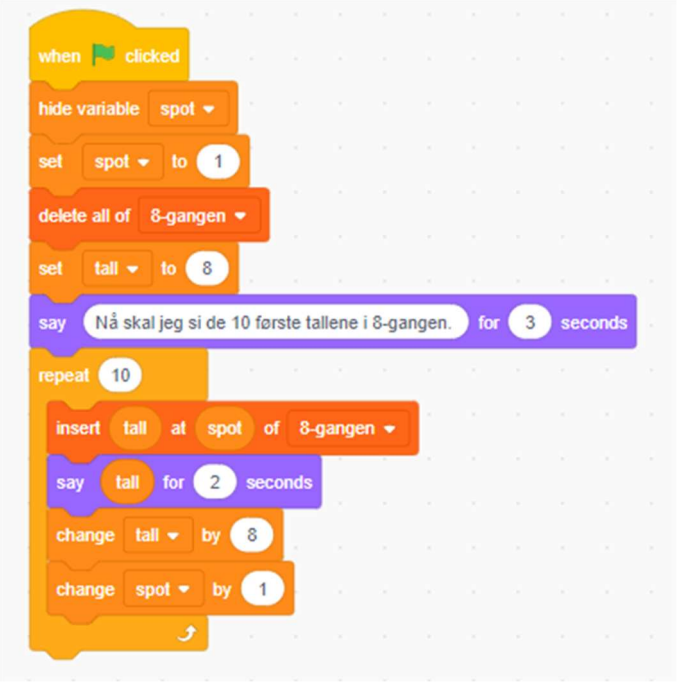
Etter denne samtalen, og etter at de har forsøkt å legge tall inn i en liste, utvikler programmet seg til slik som vist på figur 16.



Figur 16: Økt 7, gruppe 1 sin tallrekke med variabel og et forsøk på å lage liste

Programmet virket greit, men de fikk fremdeles ikke tallene inn i listen. De hadde ikke plassert kodene i riktig rekkefølge for å få ønsket resultat, og dette kan være et resultat av misoppfatning M23.

Helt til slutt spurte gruppen en medelev fra en annen gruppe om hjelp, og han gjorde endringer på programmet deres slik at de endte opp med programkoden som vises på figur 17.



Figur 17: Økt 7, gruppe 1 sin tallrekke med variabel og liste

Eleven som hjalp gruppen med å lage dette programmet hadde programmering som valgfag, og han var en ressurs for de andre elevene i klassen. Han løste oppgaven for dem, og elevene på gruppen skjønte egentlig ikke hva som ble gjort. Likevel var det tilfredsstillende for dem å endelig komme helt i mål. Dette endelige programmet listet opp alle de 10 tallene og la dem inn i en liste samtidig. Vi ser at det er tatt i bruk enda en variabel «spot» som skal holde rede på hvor i listen tallet skal plasseres. I dette tilfellet ville ikke det vært nødvendig, men flott å se at han tok i bruk en variabel på enda en ny måte.

Oppsummering av økt 7 gruppe 1: I denne økten har jeg observert at det er utfordrende for gruppen å forstå når det er nødvendig å opprette variabler og på hvilken måte variabler kan hjelpe oss med å generalisere koden. De opprettet variabelen «tall» antagelig kun fordi det sto i oppgavebeskrivelsen, og ikke fordi de hadde en plan om å gjøre nytte av denne variabelen i programkoden sin. Videre var det vanskelig for dem å endre verdien til variabelen, og å legge disse verdiene inn i en tabell. De hadde også vansker med å initialisere variabelen på starten av programmet, samt å sette koden i riktig rekkefølge. (M23).

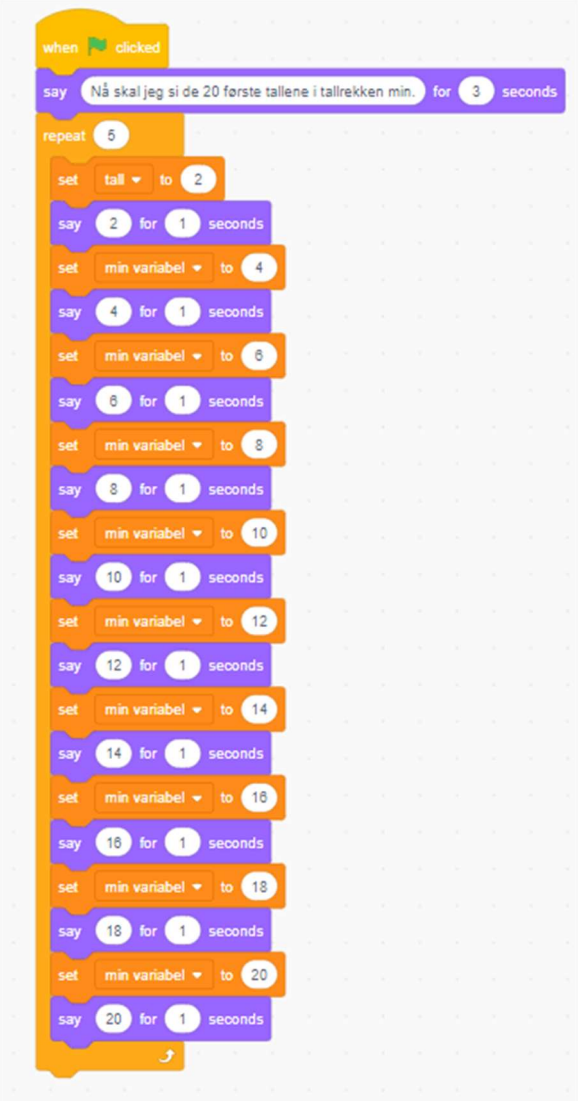
Økt 7 - tallrekker - gruppe 2

Denne gruppen besto av tre jenter som samarbeidet godt. Alle var med på å bidra positivt i gruppen, og de stilte hverandre mange faglige spørsmål underveis i arbeidet. Spørsmålene og svarene de gav hverandre var med på å drive prosjektet fremover. Noen av spørsmålene var av matematisk art, slik som når de skulle bestemme hvilken tallrekke de ville velge. De gikk gjennom ulike tallrekker, slik som partall, kvadrattall og trekanttall, og de ble enig om at de ville ta en enkel først.

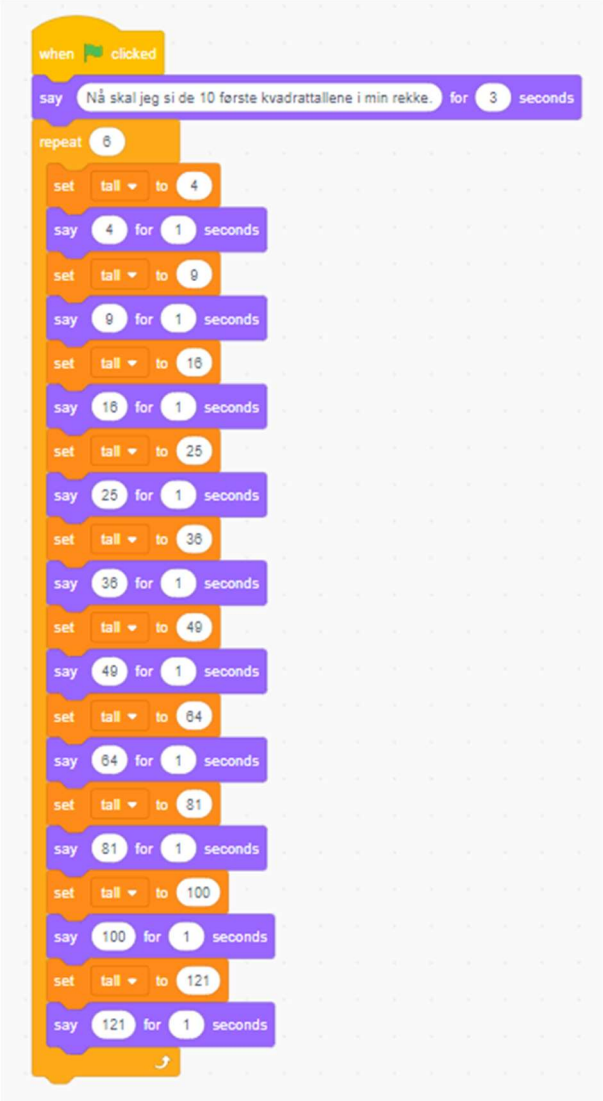
J1: 2-4-6-8, vi begynner med den, og så utvider vi videre til kvadrattall for eksempel, eller trekanttall.

Andre spørsmål de gav hverandre handlet om hvordan de skulle skrive koden, og hvor man kunne finne kodeblokkene i menyen. Underveis i arbeidet deres vises det tydelig at de ikke skjønte hvordan de kunne bruke programmeringen til å forenkle en oppgave, eller hvordan programmering kan brukes som et nyttig hjelpemiddel. De opprettet en variabel og en løkke kun fordi de hadde fått beskjed om det, og ikke fordi de selv så noen verdi av å benytte seg av det. Figur 18 og 19 er bildeutsnitt fra arbeidet deres som viser dette tydelig.

I gruppens første program, som vises i figur 18, ønsket elevene at programmet skulle skrive ut partallene opp til 20. Når de kjørte programmet, fikk de ønsket resultat og de var svært fornøyde med å ha løst oppgaven. Programmet så ut til å virke som det skulle, siden katten sa alle partallene slik som elevene hadde ventet seg. Det er likevel flere feil med dette første programmet som de har laget. For det første sies det innledningsvis at de skal si 20 tall, mens det blir sagt bare 10 tall. Hele sekvensen blir repetert 5 ganger i en løkke, hvilket er ganske unødvendig og også litt tilfeldig valgt. For det andre har de opprettet en variabel [tall] som aldri brukes i programmet. Videre endres verdien til en annen variabel med navn [min variabel] uten at dette gir noen effekt på programmet. De brukte altså to variabler i koden sin, uten at variablene gav noen reell virkning på programmet. Gruppen har ikke generalisert koden sin, slik at det ville blitt svært krevende for de om oppgaven ba om de 100 første, eller 1000 første, partallene. Det samme gjelder for den neste oppgaven med kvadrattallene, som vises i figur 19, men der har de riktig antall tall i utskriften og de har bare med en variabel.



Figur 18: Økt 7, gruppe 2 sin tallrekke med to variabler som ikke brukes



Figur 19: Økt 7, gruppe 2 sin tallrekke med en variabel som aldri brukes

J1 *Det er veldig mye trykking.*
J2 *Mhm*
J1 *Jaja, det er verdt det*
J3 *Eller er det det?*
J1 *Det er kjempegøy (ironisk)*
J1 *Å trykke sånn anbefales på høyt nivå*
0 *Alle ler*

Elevene kommenterte selv at det ble mye trykking, og tenkte nok at det burde vært en enklere måte å løse oppgaven på. Det er tydelig at variabelen [tall] aldri brukes til noe nyttig i koden deres, og henger bare med fordi de har fått beskjed om at det var lurt å opprette en variabel. I koden med partallene har de i tillegg glemt å erstatte [min variabel] med [tall], og det er uansett ingen av disse to variablene som brukes på en meningsfull måte i disse programmene.

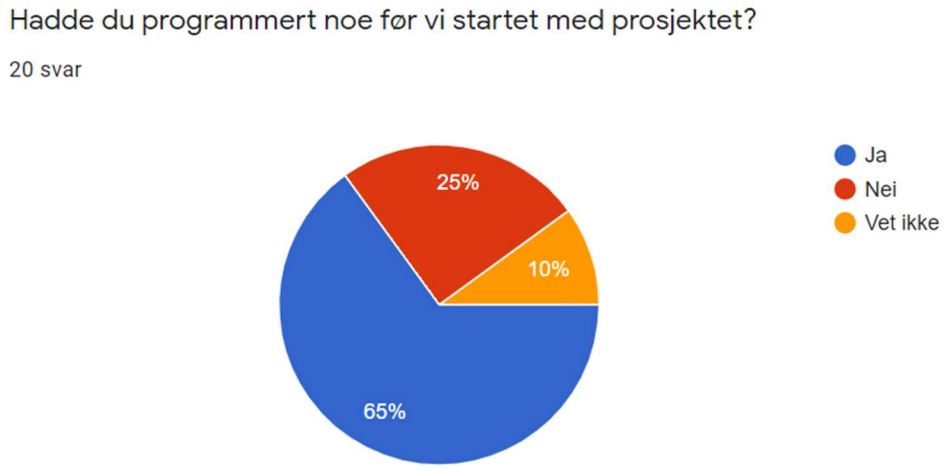
Videre ser vi at gruppen har opprettet en løkke rundt det hele, og den repeteres 5 ganger med partallene og 6 ganger med kvadrattallene, hvilket kan virke litt tilfeldig og uten mening i denne sammenhengen. Elevene syntes det var fint at programmet begynte på nytt etter å ha kommet til 20, men erfarte aldri å se at programmet faktisk stopper opp etter 5 og 6 runder. De så aldri hele programkjøringen ferdig. Elevene lærte nok litt om løkker likevel, men fikk ikke brukt hverken variabler eller løkker på den måten som det var tenkt.

Oppsummering økt 7 gruppe 2: Observasjonene fra denne økten viser at det kan være vansker med å vite når man i det hele tatt skal opprette en variabel og hvordan den skal kunne brukes på en fordelaktig måte. Gruppen har opprettet variabler og løkker kun fordi de har blitt bedt om det, og uten å ha en plan om å bruke det på noen hensiktsmessig måte. Likevel får de ønsket resultat av koden sin, og i de tilfellene der de ikke skal repetere veldig mange tall kan deres måte å løse oppgaven på virke greit. Likevel vil denne metoden være unødvendig tungvint, og som de kommenterer selv blir det «mye trykking». Det bør være et mål å få elevene til å gjøre erfaringer med hvordan variabler kan forenkle ulike problem, slik at programmeringen faktisk oppleves som et hjelpemiddel fremfor en unødvendig og tung prosess.

4.3 Spørreundersøkelse for 8. klassen etter sju programmeringsøkter

Etter å ha gjennomført sju undervisningsøkter i klassen som jeg besøkte i dette prosjektet ville jeg undersøke nærmere hvilke kommentarer klassen hadde til øktene. Jeg var interessert i å finne ut hvilken økt de likte best, hva som var vanskelig og hva de lærte mest av. Det var viktig for meg å vite hvordan elevene opplevde timene slik at jeg kan videreutvikle og forbedre undervisningen i fremtiden. Spørreundersøkelsen var frivillig og ble gjennomført av faglærer i klassen. Jeg fikk inn 20 av 26 svar med jevn fordeling av kjønnene. Den inneholdt 13 spørsmål, men jeg velger å presentere kun de resultatene som jeg synes er relevant for temaet i denne oppgaven. Noen av spørsmålene var rene sorteringsnøkler, slik som kjønn og om eleven var med på videoopptak eller ikke.

Spørsmål 2



Figur 20: Resultat fra spørsmål 2 i spørreundersøkelsen etter sju økter

Vi ser at i denne klassen hadde 35% av elevene ikke programmert tidligere eller visste ikke om de noensinne hadde programmert, og dermed kan alle disse kategoriseres som nybegynnere. 65% hadde programmert tidligere. Av disse 65% vet jeg at noen av de hadde programmering som valgfag, så andelen som hadde erfaring med programmering fra før ungdomsskolen er nok litt lavere. Likevel; resultatet viser at det er ulikt hvilke forkunnskaper elevene har innenfor programmering. Tallene for denne klassen passer godt overens med det som gjaldt generelt for trinnet. 10% av de 20 svarene jeg fikk inn, altså 2 av elevene, svarer at de ikke vet om de hadde programmert noe før vi startet med prosjektet.

Spørsmål 3
Hvis ja; hvilket programmeringsspråk har du brukt tidligere?

Resultat: Her fikk jeg en god blanding av svar og det var følgende programmeringsspråk som ble nevnt: scratch, html, blocks, microbit, java, bitpot, sphero og husker ikke.

Av de som hadde programmert tidligere og som husket navnet på programmeringsspråket ser vi at de har jobbet med programmering på ulikt vis tidligere. Det er altså stor variasjon i både forkunnskaper og hvilket programmeringsspråk de kjenner fra før. Dette gjør at man kan ha et bredt kunnskapsfelt i klassen totalt sett, og at de kan lære hverandre mye gjennom samarbeid, men at det kan bli utfordrende å lage et felles opplegg som skal være passende for alle.

Spørsmål 4

Figur 21: Resultat fra spørsmål 4 i spørreundersøkelsen etter sju økter

I forkant av øktene forsøkte jeg å legge til rette for at oppgavene skulle være av en type som gjorde at alle elevene ville kunne mestre noe uavhengig av forkunnskaper, men samtidig med muligheter for utfordringer for de som ønsket det. 45% av elevene synes det var varierende nivå og 25% mente at det var et passe nivå, og det samsvarer godt med nivået jeg forsøkte å legge dette prosjektet på. Likevel er det 10% som syntes at det var for lett, og 20% som syntes det var for vanskelig. Det viser seg at det kan bli utfordrende å jobbe med programmering på et nivå hvor alle føler mestring på sitt nivå.

Spørsmål 6
Hva var det gøyeste i timene?

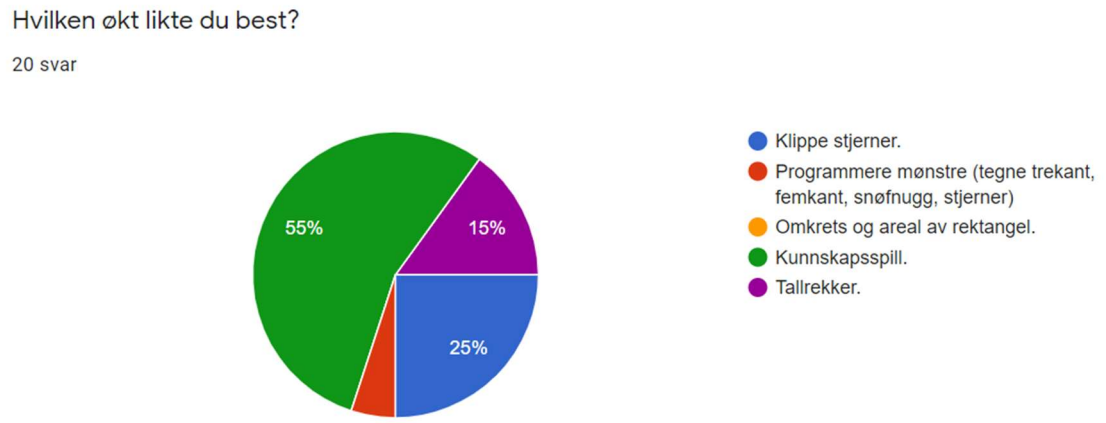
Resultat: Å lære nye ting, egentlig alt, å gjøre det sammen med vennene på gruppen, når jeg faktisk fikk programmet mitt til å fungere, å lage kunnskapsspillene, lage spill, å bruke veldig mye hodet for å tenke hva man skal gjøre, programmere, ingenting var gøy, det som var gøyest i timen det var når vi lagde et spill, å få ting til som en gruppe, spill-laging, å lage spill, jeg vet ikke, vet ikke, det gøyeste var at vi lagde spill, å lage spill, de gøyeste timene var når vi laget et valgfritt kunnskapsspill, når vi fikk til det vi skulle gjøre, det gøyeste var å lage spillet og lage de greiene i første time (arkene).
Det er tydelig at elevene foretrekker å bruke programmeringen til å lage spill.

Spørsmål 7
Hva kunne vært gjort på en bedre måte? Kom gjerne med forslag.

Resultat: Vet ikke, vet ikke helt, egentlig ikke, forklare litt mer hva vi skulle gjøre, kommer ikke på noe, vet ikke, gi lettere oppgaver og gi en oppskrift, vet ikke helt, skrive logg, at når vi filmet brukte det lang tid på å laste inn, blitt enige med en gang og ikke brukt tid på å finne ut hvem som skulle gjøre hva, kanskje ikke brukt Scratch noen ganger, jeg vet ikke, vet ikke alt var bra synes jeg, konsentrert meg bedre, vi kunne fått velge litt mer selv kanskje forskjellige oppgaver til de som har programmert før og lettere til de andre, vet ikke, vi kunne gjort gøyere ting og ikke bare lage figurer.

Det er nyttig å se tilbakemeldinger fra elevene om hva de mener kunne ha vært gjort på en bedre måte. Det viser seg å være forbedringspotensialer både når det gjelder programvaren, samarbeidsevner elevene imellom, elevers egen konsentrasjon og ikke minst lærers tydeliggjøring av de ulike valgene og vanskelighetsnivåene som elevene kunne velge blant.

Spørsmål 9

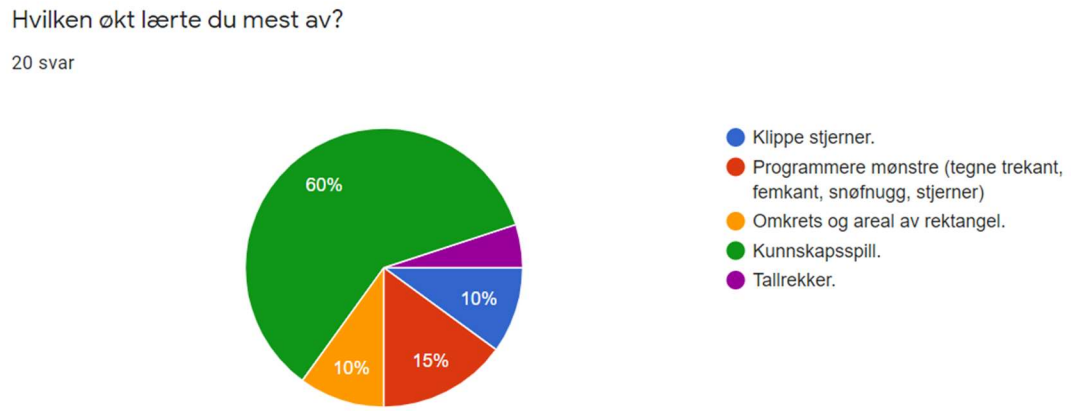


Figur 22: Resultat fra spørsmål 9 i spørreundersøkelsen etter sju økter

Av programmeringsøktene elevene likte best var det øktene med produksjon av kunnskapsspill som helt tydelig pekte seg ut. Ingen elever foretrakk økten hvor man brukte programmering til å beregne omkrets og areal av et rektangel. Disse resultatene viser tydelig at elevene som deltok i denne studien foretrakk programmering som ikke trekker inn altfor mye matematikk. Jeg skal være ærlig å innrømme at for min del var dette en skuffende observasjon, da jeg ønsker å bruke programmering som et verktøy i matematikkfaget og ikke

ønsker å ha det som et eget emne som oppleves adskilt fra de matematiske kunnskapsområdene. Det er likevel lov å håpe at elever som lærer algoritmisk tenking og programmering via spill vil kunne knytte disse erfaringene til matematikkfaglige problemstillinger senere. Det er absolutt verdt å undersøke nærmere hvilke muligheter produksjon av spill kan gi i et klasserom.

Spørsmål 11



Figur 23: Resultat fra spørsmål 11 i spørreundersøkelsen etter sju økter

Til tross for at ingen elever foretrakk å jobbe med økten med omkrets og areal svarte 10% av elevene at det var nettopp denne økten som gav dem best læringsutbytte. Det er interessant å se at noen elever opplevde at de lærte mest av en økt som de ikke foretrakk å jobbe med. 15% mente at de lærte mest av økten med geometriske mønstre. Likevel er det øktene med kunnskapsspill elevene både foretrekker og hevder å ha aller best læringsutbytte av. Det er klart at dersom motivasjonen er til stede vil det være større muligheter for læring også, noe som kan tyde på at å benytte en spillramme kan være en god måte å jobbe med programmering for denne klassen. Jeg var på forhånd og er fremdeles av den oppfatningen at det er en fordel at lærer styrer programmeringsoppgavene inn mot matematiske tema, slik at programmeringen ikke bare består i spillutvikling og valgfrie emner. Likevel er det tydelig at spillrammen fungerer bra for elevenes motivasjon og dette er absolutt noe en kan benytte seg av i planlegging av undervisningsopplegg.

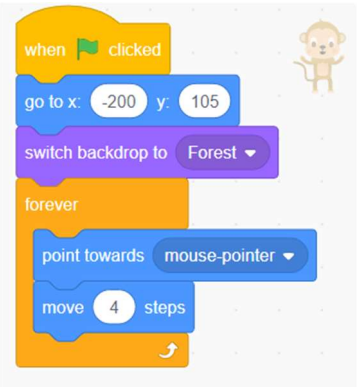
4.4 Avsluttende undervisningsøkt med spill

Denne undervisningsøkten var den siste timen jeg hadde med klassen. Etter å ha vært gjennom sju økter tidligere tenkte jeg at elevene hadde fått et grunnlag for å kunne jobbe med en litt større oppgave på en litt mer selvstendig måte.

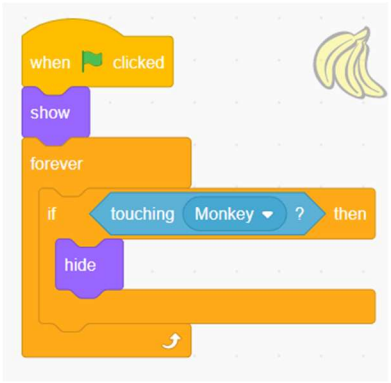
Siden spørreskjemaet som ble gjennomført etter sju økter viste at elevene foretrakk å jobbe med programmering innen produksjon av spill, valgte jeg å gjennomføre den siste programmeringsøkten med å lese, forbedre og skrive programkode til et spill.

Da det hadde vist seg i de foregående øktene at det ofte var svært vanskelig for mange å komme i gang med arbeidet og å avgjøre når man skulle ta i bruk variabler og ikke, valgte jeg denne gangen å gi elevene et ferdig programmert spill med mange muligheter for utvidelser og forbedringer. På denne måten tenkte jeg at alle ville kunne klare å komme i gang, og i det

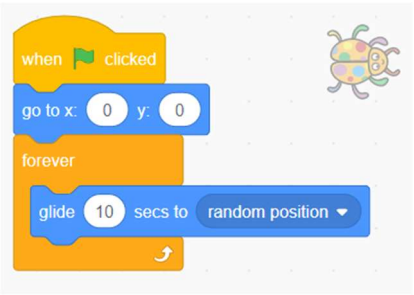
minste kunne lese eksisterende kode. Som beskrevet i teorikapittelet er det viktig å ikke bare vektlegge skriving av kode i læringsprosessen, men også legge til rette for lesing av kode. De fikk tilgang til et spill hvor en ape kunne bevege seg rundt på skjermen ved bruk av piltastene på tastaturet. Det var bananer og insekter med i spillet også, og elevene skulle utvide koden slik at apen fikk poeng hver gang den tok en banan, og deretter utvide programmet slik at apen mistet et liv hver gang den berørte et insekt. For å kunne oppnå dette måtte de ta i bruk variabler. En detaljert oppgavebeskrivelse finnes i vedlegg D. Kodene som var forhåndsgitt i spillet er vist på figur 24, 25 og 26, og det var en ape, ti bananer og fire insekter med i spillet.



Figur 24: Gitt kode for Monkey



Figur 25: Gitt kode for Banana



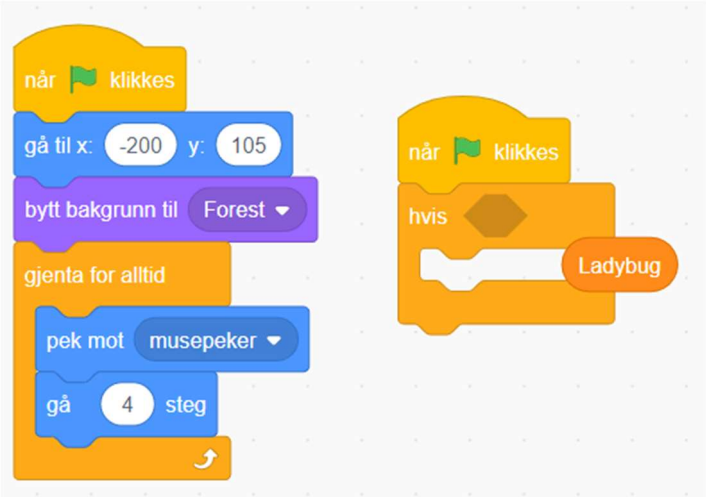
Figur 26: Gitt kode for Ladybug

På gruppe 1 var det denne gangen en jente og en gutt som var til stede. En av samtalene som fant sted var slik:

G1: Ok, hvis vi skal få han til å miste liv når han treffer biene så må vi få sånn «hvis treffer bie», så må vi kanskje lage en variabel, hva heter de da…
G1: Ladybug.
J1: Hehe, Ladybug.

Etter denne dialogen opprettet elevene en variabel kalt Ladybug. Dette er et unaturlig variabelnavn for å holde kontroll på hvor mange liv man har i spillet, og det ville nok ha vært mer naturlig å kalle denne variabelen for «liv» eller «AntallLiv». Dette kan tyde på at elevene har vansker med å velge passende variabelnavn. Videre forsøkte de å plassere variabelen Ladybug inn i en hvis-løkke slik som vist på figur 27, og samtidig sa en av elevene:

G1: Hvis Ladybug treffer… eh de… å nei det går ikke.



Figur 27: Økt 8, gruppe 1 resonnerer korrekt men gjør feil med syntaks

På grunn av at variabelen Ladybug ikke passet inn i sekskanten, brukte eleven bare få sekunder på å forstå at dette ikke ville fungere. Dette viser et eksempel hvor programvaren

hindrer programmereren til å gjøre syntaktiske feil. Ellers er det interessant å se at elevene snakket om Ladybug som et objekt, men ønsket å bruke variabelen Ladybug i koden. Samtalen antyder at elevene forveksler de fire objektene med navn Ladybug1, Ladybug2, Ladybug3 og Ladybug4 med variabelen Ladybug.

Videre observerer jeg at gruppen har vansker med å opprette korrekte betingelser. Først lagde de en kode hvor man mister liv når et insekt berører musepeker, deretter korrigerte de dette til at man mister liv hver gang insekt nummer 1 berører insekt nummer 2. Ingen av disse alternativene er korrekte ut ifra oppgavebeskrivelsen. Det er fint å se at gruppen testet programmet flere ganger underveis, og endret på koden når de oppdaget at resultatet ikke ble som ønsket. I denne prosessen jobbet gruppen med flere av de midterste stegene i PRIMM-metoden. De kom til slutt i mål med at man mistet liv hver gang når apen berørte et insekt.

Videre oppsto det en samtale som viser at manglede kunnskap om matematisk notasjon kan være med på å skape noen vansker. Gruppen ville lage en programkode slik at dersom man hadde null liv igjen skulle man få opp teksten «game over», men de var usikre på hvordan de skulle skrive denne koden på en korrekt måte. På dette tidspunktet hadde de for øvrig forkastet variabelen Ladybug og laget en ny variabel med navn Liv. De oppsøkte hjelp fra lærer, og da oppsto følgende samtale:

L: *Men når du bruker «hvis» så må du bruke operatorer, og så f.eks. hvis poeng er lik 10… kanskje, da skjer det noe?*
G1: *Åja, jeg vil ha hvis liv er lik null.*
L: *Ja, da må du trykke inn liv…*
G1: *Der kan jeg ta den inn.*
L: *Den kan du hente derfra ja. Hvis liv er større enn null…*
G1: *Er lik.*
L: *Nå står det ikke er lik. Nå står det hvis liv er større enn null.*
G1: *Åja, større enn 3.*
L: *Hvis… det er det jo aldri, for du skulle starte med 3 liv. Men hva er det du vil gjøre egentlig?*
J1: *Du må ta er lik hvis du skal få det til å bytte bakgrunn.*
G1: *Ja, men jeg vil at når det blir til null da skal det komme sånn…*
J1: *Ja, da skal det være er lik.*
L: *Ja, da tar du vekk det, og så bruker du operator er lik. Det er den. Og hvis liv er lik null, da skjer det noe.*
G1: *Åja, ja.*
0: *Skriver koden som det er blitt enighet om.*
0: *Lager en bakgrunn med tekst «game over» og får dette inn i koden slik at dette kommer opp dersom det er null liv igjen.*

Figur 28: Økt 8, gruppe 1 forveksler likhetstegn med ulikhetstegn

I dette tilfellet kan det være manglende kunnskap om likhetstegnet og ulikhetstegnet som gjør at det blir vanskelig å mestre kodingen. Siden likhetstegnet i Scratch har samme betydning som i matematikken er ikke denne vansken noe som oppstår på grunn av at et symbol har ulik betydning i ulike registre. Her brukes matematiske symboler direkte inn i kodingen, og for å få programmet til å fungere som man ønsker må man først ha de nødvendige matematiske

forkunnskapene. Det er interessant å se hvordan G1 poengterer muntlig at han ønsker at det skal stå er lik, og han ønsker at det skal skje noe når liv blir til null, men at han likevel bruker et ulikhetstegn i koden. Det kan dermed se ut til at eleven har god evne til å planlegge og beskrive hva som skal skje, og at han kan forklare hvordan variabelen skal spille noen rolle i dette programmet, men det ser ut til at det er noe manglende kunnskap om matematisk notasjon som hindrer eleven i å skrive koden korrekt.

På gruppe 2, hvor tre jenter deltok, mestret elevene etter hvert å gi poeng i spillet hver gang apen tok en banan. De glemte imidlertid å nullstille denne variabelen ved hver ny kjøring av programmet, slik at poengene fortsatte fra der forrige runde avsluttet. Denne samtalen viser deres reaksjon på dette:

J1: Jeg fikk 11 poeng, men det er bare 10 bananer. Sånn, trykker vi på den (flagget) igjen, så får vi 12, 13, 14, hehe, 15…
J2: Det der er juks.
J1: Neeei.
J3: Det er ikke noe juks.

Det er verdt å merke seg at elevene helt tydelig observerte at det var noe som ikke stemte her, uten at de brukte noe energi på å forsøke å løse dette. De nevnte ikke noe mer om poengene i programmet senere, selv om poengsummen etter hvert kom opp til 31. Dersom de hadde satt inn en startverdi på variabelen poeng helt i starten av programmet ville dette ha løst seg ganske enkelt.

Oppsummering fra økt 8:
I denne undervisningsøkten har jeg sett at elevene har opprettet variabler med ufornuftige navn og blandet disse sammen med objekter som hadde samme navn. Dette medførte at de forsøkte å trekke inn koder som bryter med syntaktiske regler, og her er Scratch godt oppbygget siden blokkene er av ulike former og ikke tillater dette. Om forvirringen knyttet til objekter og variabler er grunnet manglende syntaktisk kunnskap eller om det er snakk om andre typer misoppfatninger er vanskelig å avgjøre, men det viser uansett en vanske med å forstå bruken av variabler fullt ut.

Jeg har også sett at elever kan streve med å endre verdien til en variabel når det er en betinget hendelse som skal utløse dette. De kan også streve med å opprette betingelsene på korrekt måte. Variabelbruk i kombinasjon med betingelser og løkker viser seg å være noe vanskelig å forstå.

Ellers har det vist seg gjennom denne økten at elevene lett kan glemme å initialisere en startverdi på variabler. Dette er knyttet til manglende strategisk kunnskap.

Manglende kunnskap om matematisk notasjon, slik som vi ser her at et ulikhetstegn forveksles med et likhetstegn, kan hindre eleven i å komme i mål med kodingen.

Elevene var ganske trygge på hva de ville at programmet skulle gjøre i denne økten, og det var heller ingen store matematiske barrierer som hindret dem i å løse akkurat denne oppgaven. Det som viste seg å være utfordrende i denne oppgaven var bruken av variabler, løkker og betingelser. Dette trenger elevene mer kunnskap om, og dersom de oppnår

kompetanse innen disse områdene, vil programmering i enda større grad kunne bli brukt som et verktøy for å løse nye og ukjente problemer for dem.

Ved poenggivningen observerer jeg at det er ulike måter å løse det på, og noen metoder er mer effektive enn andre. En gruppe valgte å legge alle kodene på apen, og måtte dermed trykke inn ganske mye. Det kunne ha vært enklere å ha lagt en liten kode på en banan og kopiert dette til alle bananene. Det samme gjelder ved tap av liv. Dette kan tyde på at det kan være vanskelig for elevene å vite på hvilket objekt koden bør plasseres på.

4.5 Avsluttende kartleggingstest med variabler

Siste innsamling av data i dette prosjektet var en avsluttende kartleggingstest med 13 diagnostiske oppgaver hvor 26 elever fra 8. klassen deltok. Denne testen ville gi elevene og læreren deres en tilbakemelding på hva de mestret godt og hva de burde øve mer på videre. Den vil også gi meg et bilde av hvilke misoppfatninger elevene hadde knyttet til bruk av variabler i Scratch.

I det følgende viser jeg oppgavene som jeg gav elevene i denne kartleggingstesten, hvilke mulige misoppfatninger jeg forventet å finne ut ifra kjent teori, og deretter vises resultatene og en analyse av resultatene. Det forventes at leser allerede er kjent med de ulike misoppfatningene som er presentert i kap. 2.3.9. I tillegg har jeg gjort funn av andre typer misoppfatninger som ikke er nevnt i kap. 2.3.9, og som dermed kommer utenfor det forhåndsbestemte rammeverket. For å holde oversikten har jeg navngitt de nye misoppfatningene som type T1-T8. Beskrivelse av disse kommer underveis i drøftingen der de er blitt registrert. Tabell 3 viser en oversikt over alle misoppfatningene som nevnes i dette delkapittelet. Det kan være en fordel å ha oversikten tilgjengelig ved lesing av resultatene.

	Beskrivelse av misoppfatningene/vanskene
M9	En variabel kan ha flere ulike verdier på samme tid / kan huske gamle verdier.
M11	Tildeling av verdi virker i motsatt retning.
M12	Tildeling av verdi virker i begge retninger.
M15	Variabler som tildeles en verdi via et uttrykk lagrer likninger eller uløste uttrykk.
M16	Tildeling av en verdi flytter en verdi fra en variabel til en annen.
M17	Det valgte variabelnavnet vil påvirke hvilken verdi som går inn i hvilken variabel.
M18	Rekkefølgen man oppretter og navngir variablene vil påvirke hvilken verdi som går til hvilken variabel.
M20	Misoppfatninger/vansker knyttet til å endre en variabel.
M23	Vansker med å forstå betydningen av rekkefølgen på koden.
T1	En variabel kan være alle tall innenfor et bestemt intervall.
T2	Man generaliserer et mønster og får output deretter.
T3	Ignorerer deler av koden.
T4	Reduksjon/sammenslåing av variabler.
T5	Riktig svar, men output er utvidet til noe mer enn hva som forventes. F.eks. svarer man med en hel setning hva som blir output fremfor å bare gi output.
T6	Svarer på mer enn det spørres om. F.eks. oppgis alle variablene som output, mens man bare skulle hatt med en av dem.
T7	Feilaktig tolkning av syntaks/semantikk.
T8	En variabel husker og adderer alle tidligere verdier.

Tabell 3

Oppgave 1 - M9

Hva sier Katten?

Figur 29: Diagnostisk oppgave 1 for å undersøke M9

Misoppfatning M9: En variabel kan ha flere ulike verdier på samme tid / kan huske gamle verdier.

Eleven vil da svare 1, 2

Resultater:

Svar fra elevene	Frekvens	Type
2	10	Riktig svar
1, 2	11	M9
Et tall fra 1 til 2	1	T1
Først 1, så 2, så neste tall	1	T2
Tall	3	M15a

Diskusjon

I oppgave 1 undersøker jeg om elever har misoppfatninger som samsvarer med M9. Det var 10 av 26 elever som svarte riktig på denne oppgaven. 16 elever klarte ikke denne første oppgaven, og svarene deres kan tyde på ulike typer misoppfatninger. 11 elever mente at variabelen ville ha to verdier samtidig, og svarte at katten ville si både 1 og 2. Dette tyder på at elevene kan ha misoppfatning M9. Det er overraskende at mange elever gir dette feilaktige svaret etter å ha deltatt i åtte programmeringsøkter, og de har i tillegg hatt videolekse om variabler. Det er tydelig at denne misoppfatningen må det legges mye større fokus på fra min side når jeg skal jobbe med programmering i en klasse på ungdomstrinnet. M9 gir ikke bare utslag i oppgave 1, men vil vise seg i flere av de følgende oppgavene, noen ganger i kombinasjon med andre misoppfatninger samtidig.

Én elev gjør oss oppmerksom på en annen type misoppfatning; at variabelen nå er et hvilket som helst tall mellom 1 og 2. Eleven har kanskje tenkt at dersom variabelen først er 1, og deretter endres til 2, kan variabelen være et hvilket som helst tall mellom de to gitte tallene. Jeg navngir denne typen misoppfatning for T1. Misoppfatningen kan være en direkte konsekvens av at man bruker variabelbegrepet ulikt i matematikk og i programmering, og om eleven har en oppfatning om at en variabel kan være «any number», slik Russell (1937) definerer det, vil dette svaret være rimelig.

Videre er det én elev som svarte at variabelen først er 1, så 2, så neste tall. Eleven kan ha sett på de to første verdiene for variabelen og forestilt seg en type generalisering som gjør at output vil fortsette i det samme mønsteret, selv om det ikke er noe i koden som tilsier at dette skal skje. Denne typen misoppfatning kaller jeg for T2.

Til slutt ser vi at det er 3 elever som svarte at katten sier «tall». Dette er en type misoppfatning hvor man antar at output ikke er variabelens verdi, men variabelens navn. Denne oppfatningen samsvarer med misoppfatning M15a. Jeg hadde planlagt at oppgave 3

skulle avdekke denne formen for misoppfatninger senere, men som vi ser inntreffer denne misoppfatningen allerede i oppgave 1.

Jeg ser i ettertid at oppgave 1 kan avdekke flere typer misoppfatninger enn jeg hadde kunnet forutse. Den er en fin oppgave å gi til nybegynnere i programmering, slik at man ved å drøfte denne oppgaven grundig kan avdekke og oppklare mange typer misoppfatninger helt fra starten av. Å avdekke og oppklare misoppfatninger tidlig kan bidra til økt læring og motivasjon blant elevene. En slik aktivitet kan tilsvare det Brekke (2002, s. 16) kaller for en konfliktdiskusjon i diagnostisk undervisning, hvor elever med misoppfatninger blir satt i en kognitiv konflikt og denne løses gjennom diskusjon. Oppgaven er lett å forstå så lenge man har riktig forståelse for hvordan en variabel får og endrer verdi i et dataprogram.

Oppgave 2 – M11 og M12

Hva sier Katten?

Figur 30: Diagnostisk oppgave for å undersøke M11 og M12

Misoppfatning M11: Tildeling av verdi virker i motsatt retning. Eleven vil da svare 10, 10

Misoppfatning M12: Tildeling av verdi virker i begge retninger. Eleven vil da svare 20, 10

Resultater:

Svar fra elevene	Frekvens	Type
20, 20	6	Riktig svar
10, 10	0	M11
20, 10	1	M12
10, 20	8	T3
10	1	T4
20	2	T4
10, 20, 30	1	M9 og T7
10, 20, 20	1	M9
10, 20, 20, 20	1	M9
10, 20, 20, 10, 20	1	M9
Tall1, tall2	2	M15a
tall2	1	T4 og M15a
Et tall fra 10 til 20	1	T1

Diskusjon

I denne oppgaven hadde jeg planlagt å avdekke misoppfatninger innen M11 og M12. Det var interessant at absolutt ingen elever svarte 10,10, som ville vært funn av misoppfatning M11. Ellers var det kun én elev som svarte 20, 10, som tyder på at eleven kan ha misoppfatning M12. 6 elever svarte riktig, mens resten viser oss andre typer misoppfatninger som jeg vil forsøke å beskrive nærmere i det følgende.

8 elever svarte 10, 20. Disse elevene kan ha ignorert koden [sett tall1 til tall2]. Følgene av dette blir selvsagt feil svar. Å ignorere deler av koden kaller jeg misoppfatning av type T3.

Én elev har svart 10, og to elever har svart 20. De har gjort to variabler om til en. En slik feilaktig sammenslåing /reduksjon av antall variabler kaller jeg for en misoppfatning av type T4. Det er mulig at denne sammenslåingen av variabler er knyttet til en annen type misoppfatning, M16, hvor man tenker seg at tildeling av en verdi flytter en verdi fra en variabel til en annen, og at det deretter kanskje ikke er bruk for mer enn en av dem. Jeg velger å la M16 representere misoppfatninger der begge variablene fremdeles eksisterer, mens i T4 er det blitt en reduksjon i antall variabler.

Én elev har svart 10, 20, 30. En mulig forklaring er at eleven antar at en variabel kan ha flere verdier på samme tid (M9). For å vite helt nøyaktig hva eleven har tenkt for å komme frem til dette svaret skulle jeg ha snakket med eleven. Uten en slik samtale kan jeg bare komme med en antagelse om elevens mulige resonnement. Eleven kan ha tenkt at tall1 er 10 og tall2 er 20 og 30. Da kan eleven ha tolket koden [sett tall1 til tall2] som at tall1 skal legges til tall2, og i kombinasjon med M9 vil dette kunne resultere i det feilaktige svaret. En feilaktig tolkning av syntaks kategorisere jeg som T7.

Eleven som svarte 10, 20, 20 kan ha tenkt at tall1 er 10 og 20 og at tall2 er 20, og dette kan tyde på at eleven har misoppfatning M9.

Én elev har svart 10, 20, 20, 20 og én elev har svart 10, 20, 20, 10, 20. Det er vanskelig for meg å vite nøyaktig hva de har tenkt, men dette viser uansett mangfoldet av misoppfatninger man kan ha som nybegynner i programmering. Det er uansett tydelig at de gir flere verdier i output, noe som kan tyde på misoppfatning M9.

To elever har svart tall1, tall2, og er funn av M15a, hvor man gjengir variabelens navn i stedet for variabelens verdi.

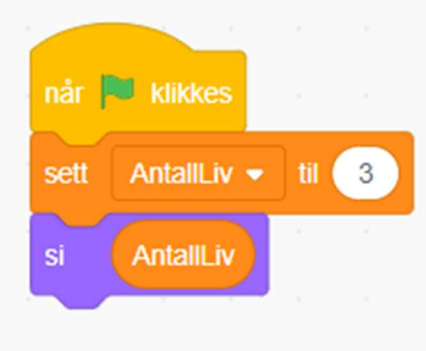
En elev har svart tall 2. Siden det er reduksjon av antall variabler viser eleven et svar som er forenlig med misoppfatning T4, og siden det er variabelens navn og ikke variabelens verdi som oppgis viser eleven i tillegg M15a.

Én elev mener at vi vil få en variabel som kan være alt mellom 10 og 20. Dette er funn av misoppfatningen som jeg tidligere har navngitt T1, at en variabel kan være alle tall innenfor et bestemt intervall. Når jeg kikker nærmere på datamaterialet kommer det frem at det er den samme eleven som viser denne misoppfatningen i begge tilfellene.

Også i denne oppgaven observeres noen misforståelser som jeg ikke hadde søkt etter i denne bestemte oppgaven. Vanskene som viser seg, kan være knyttet til misoppfatninger som ble observert allerede i oppgave 1.

Oppgave 3 - M15a

Hva sier Katten?



Figur 31: Diagnostisk oppgave 3 for undersøke M15a

Misoppfatning M15a: Gjengir variabelens navn.

Eleven vil da svare AntallLiv.

Svar fra elevene	Frekvens	Type
3	22	Riktig
4	1	Skrivefeil?
AntallLiv	3	M15a

Diskusjon

I oppgave 3 kartlegger jeg om elevene har M15a. Det er én elev som har svart 4, og dette ser jeg på som en skrivefeil, og 22 elever har svart det riktige tallet som er 3. De siste tre elevene har svart AntallLiv, som er variabelens navn. De har ikke skjønt at det er variabelens verdi som vil komme til syne når programmet kjøres. Etter en nærmere kikk på resultatene har jeg oppdaget at det er de samme tre elevene som har gjort denne misoppfatningen gjennom alle de tre første oppgavene. Selv om oppgave 1 og 2 ikke var ment for å avdekke M15a, så er det validerende at de samme tre elevene viser denne misoppfatningen konsekvent gjennom alle de tre første oppgavene. Det er sannsynlig at disse tre elevene har M15a og vil ha problemer med dette gjennom flere av oppgavene. Dersom jeg skulle fortsette å være med klassen ville disse funnene ha gjort det enkelt for meg å kunne veilede disse tre elevene for å klare opp i denne misoppfatningen. Oppgave 3 er ikke egnet for å avdekke andre misoppfatninger, og er dermed en god diagnostisk oppgave for å undersøke misoppfatninger innen M15a.

Oppgave 4 - M15b

Hva sier Katten?

Figur 32: Diagnostisk oppgave 4 for å undersøke M15b med en variabel

Misoppfatning M15a: Gjengir variabelens navn.
Eleven vil da svare A.

Misoppfatning M15b: Gjengir et uløst regnestykke eller en likning, men kan ha klart å erstatte variabler med verdier.

Eleven vil da svare 2+7 eller A = 2+7

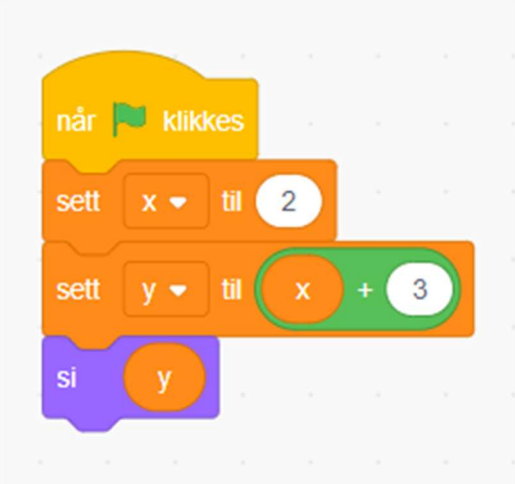
Elevenes svar	Frekvens	Type
9	15	Riktig
A	3	M15a
2+7	4	M15b
2+7=9	4	T5

Diskusjon

I denne oppgaven ser jeg etter misoppfatning M15b. Det er 15 elever som har svart riktig på denne oppgaven. Det er fremdeles tre elever som viser misoppfatning M15a og svarte variabelens navn A. 4 elever har svart 2+7, noe som er et uløst regnestykke dermed tyder på at elevene har misoppfatning M15b. 4 elever har svart 2+7=9. Dette kategoriserer jeg som T5, og T5 er tilfelle når man har løst regnestykket og får riktig svar, men utvider output til noe mer enn forventet. I tilfellet hvor elevene svarte 2+7=9 har de løst oppgaven, men gir mer output enn hva som er korrekt.

Oppgave 5 - M15b

Hva sier Katten?



Figur 33: Diagnostisk oppgave 5 for å undersøke M15b med to variabler

Misoppfatning M15a: Gjengir variabelens navn. Eleven vil da svare y.

Misoppfatning M15b: Gjengir et uløst regnestykke eller en likning, men kan ha klart å erstatte variabler med verdier. Eleven vil da svare noen av disse alternativene:

x+3 2+3 y=x+3 y=2+3

Elevenes svar	Frekvens	Type
5	8	Riktig
y=5	1	T5
Y	3	M15a
x+3	5	M15b
2+3	1	M15b
2, 5	2	T6
2, 2+3	1	M15b og T6
y, 3	1	M15a og T6
6	1	Brukt feil regnealgoritme?
52	1	?
Blank	2	

Diskusjon

I denne oppgaven ser jeg om det finnes misoppfatninger av typen M15b. Selv om oppgave 4 og oppgave 5 begge søker etter M15b, er oppgave 5 mer komplisert fordi den inneholder to variabler. I resultatet ser vi at dette medfører flere feilaktige svar.

Det er 8 elever som har svart riktig på denne oppgaven. De har klart å skille mellom variablene x og y, og har dessuten brukt verdien til x til å finne verdien til y.

Én elev har svart y=5. Dette er riktig verdi, men katten vil ikke si y=5, for den vil bare si 5. En slik utvidelse av output navngir jeg for T5.

3 elever har svart y. Det viser seg at det er de samme tre elevene som tidligere har gjengitt variabelens navn og har M15a.

5 elever har svart x+3, og en har svart 2+3. Disse elevene gjengir uløste oppgaver fremfor å regne ut svaret, og viser dermed et svar som er forenlig med misoppfatning M15b.

2 elever har svart 2, 5. Oppgaven ber kun om y og ikke om x. Likevel ser det ut til at de velger å opplyse om både x og y. Når eleven gir verdien til flere variabler enn det som koden ber om kaller jeg misoppfatningen for T6. Denne misoppfatningen kan muligens forveksles med M9, men jeg forsøker likevel å skille disse to.

Én elev har svart 2, 2+3. Dette blir det samme som det forrige svaret 2, 5, bare at 2+3 er ikke regnet ut. Dette kan være M15b kombinert med T6.

Én elev har svart y, 3. Dette kan være M15a kombinert med T6, men jeg synes ikke svaret kan forklares ut fra de to misoppfatningene alene. Mulig at det er en tredje misoppfatning her i tillegg, men det er vanskelig å si hva som er tenkt i dette tilfellet.

Én elev har svart 6. En mulig forklaring er at eleven kan ha brukt multiplikasjon i stedet for addisjon.

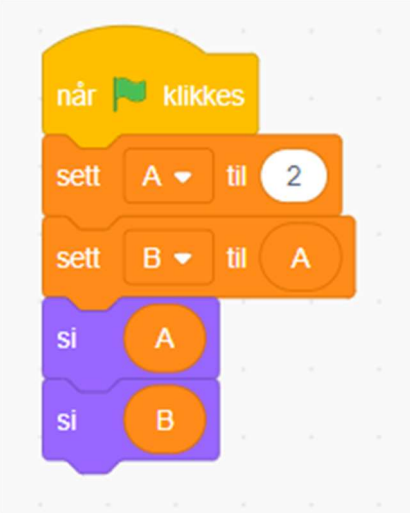
Én elev har svart 52. Det er mulig at hen har tenkt at 5 er verdien til y og 2 er verdien til x. Hvis dette skulle gi oss tallet 52 måtte koden ha lagt inn disse sifrene på tierplassen og enerplassen, men det er ingenting i koden som tilsier at dette skal skje.

2 elever har svart blankt på denne oppgaven.

I denne oppgaven ser jeg at det oppstår flere feilaktige svar enn i forrige oppgave. Siden det ikke er én variabel som endrer verdi flere ganger i denne oppgaven, men det er snakk om to variabler som tildeles verdier, bør ikke M9 forekomme her. Dersom eleven feilaktig har lest at det sto y på første linje kan svarene 2, 5 og 2, 2+3 likevel forklares med M9. Jeg erfarer at kartleggingsoppgave 4 er enklere for meg å tolke enn oppgave 5, for den gir ikke rom for å gjøre mange ulike misforståelser i samme oppgaven. Likevel er det verdifullt å registrere at M15b forekommer hyppigere når det er to variabler i oppgaven enn når det er bare én variabel.

<u>Oppgave 6 – M16</u>

Hva sier Katten?



Figur 34: Diagnostisk oppgave 6 for å undersøke M16

Misoppfatning M16: Tildeling av en verdi flytter en verdi fra en variabel til en annen.

Eleven vil da svare 0, 2.

Elevenes svar	Frekvens	Type
2, 2	10	Riktig
22	1	Tastefeil?
2	3	T4
0, 2	1	M16
A, B	2	M15a
2, A	2	M15a
2, b	1	M15a
2a	1	Tastefeil og M15a
B	1	T4 og M15a
ab, 2	1	?
2, 4	1	?
Blankt	2	

Diskusjon

Oppgave 6 undersøker om elevene har M16. Det er 10 elever som har svart det riktige svaret 2, 2. Det er én som har slått tallene sammen og ender opp på 22 til svar. Det kan være en tastefeil.

3 elever har svart 2, og dette kan være funn av T4 som går ut på at eleven har slått sammen de to variablene til en.

Bare én elev har svart 0, 2, som er funn av M16. Her ser det ut til at eleven har en oppfatning av at koden [sett B til A] vil gjøre at verdien til A blir flyttet over til B, og at A vil få verdien 0 etter denne overføringen. Dette er en misoppfatning som kategoriseres som M16. Dette skiller seg fra T4, fordi man i M16 tenker seg at variabelen A fremdeles eksisterer.

I ettertid ser jeg at output som gir variabelnavnet i stedet for variabelens verdi i noen tilfeller kan ha rot i både M15a og M16. Dersom eleven antar at variabelen er blitt helt tom etter en kommando som [sett B til A], men at variabelen fremdeles eksisterer, så kan eleven i enkelte oppgaver tenke at [si A] vil gi A i output, rett og slett fordi A har mistet sin verdi. Dette viser at elevenes svar på de diagnostiske oppgavene ikke alltid kan knyttes til bestemte misoppfatninger. Svarene deres kan oppstå av ulike grunner. Likevel mener jeg at det er verdifullt å analyse resultatene, og at det kan gi en pekepinn på utfordringer som kan fremheves og drøftes i fellesskap.

2 elever har svart A, B som antyder misoppfatning M15a.

2 elever har svart 2, A. Her har elevene forventet at koden [si A] gir 2 og [si B] gir A. Det kan se ut som at vanskene øker når en variabel får ny verdi via en annen variabel og ikke får tildelt en ny verdi direkte. Når B settes til A, vil B lagre verdien til A og ikke A sitt navn. Dette svaret viser funn av M15a, siden elevene forventer variabelnavnet som output.

Én elev har svart 2a. Om vi ikke skiller på stor og liten bokstav, er ikke dette svaret veldig ulikt det forrige, og misoppfattelsen kan ha samme opphav kombinert med at eleven mangler et komma.

Én elev har svart 2, b. Om vi ser bort fra at eleven har gjort B om til b, ser det ut til at denne eleven har skjønt deler av koden, men ikke alt. Hen har fått [si A] til å bli 2, noe som er korrekt. Det er også ganske tydelig å se verdien til A fra koden. Når det gjelder B har hen ikke skjønt hvilken verdi som ligger lagret i B og hen får dermed [si B] til å bli b. Dette er funn av M15a.

Én elev har svart B. Dette kan være funn av M15a, men siden hen bare gjengir én variabel er det T4 samtidig.

Videre er det én elev som har svart ab, 2 og én elev som har svart 2, 4. Dette synes jeg er vanskelig å finne en forklaring på.

2 har svart blankt.

Oppgave 7 – M17

Hva sier Katten?

Figur 35: Diagnostisk oppgave 7 for å undersøke M17

Misoppfatning M17: Det valgte variabelnavnet vil påvirke hvilken verdi som går inn i hvilken variabel.

Eleven svarer da 1000, 1.

Elevenes svar	Frekvens	Type
1000, 1000	7	Riktig
1, 1000	3	T3
1000, 1	3	M17
1100	1	T4
999	1	T4
1000	2	T4
1, 1000, 1000	1	M9
1, 1000, LiteTall	1	M9 og M15a/M16
StortTall, LiteTall	2	M15a
StortTall	1	T4 og M15a
Vet ikke / blankt	4	

Diskusjon

I denne oppgaven er det 7 elever som har svart det riktige svaret 1000, 1000. De har klart å holde oversikt på hvilken verdi som går inn i hvilken variabel.

3 elever har svart 1, 1000 og har dermed hoppet over [sett StortTall til LiteTall] i koden. Dette viser funn av T3.

3 elever har svart 1000, 1 og dette viser funn av M17, hvor man lar det valgte variabelnavnet påvirke hvilken verdi som går inn i hvilken variabel. I ettertid har jeg sett at svaret 1000, 1 også kan utspringe fra M12. Enten har elevene latt det valgte variabelnavnet påvirke hvilken verdi de lar gå inn i variablene, eller de har tenkt at tildelingen av verdi virker i begge retninger. Jeg ser at jeg burde tilstrebe å lage en kartleggingsoppgave hvor det var mulig å skille på disse to misoppfatningene bedre. I oppgave 2, som søkte etter M12, var det bare én elev som viste tegn til M12, mens i denne oppgaven var det 3 elever som svarte 1000, 1. Av denne grunn vil jeg påstå at denne oppgaven viser funn av at noen elever vil la det valgte variabelnavnet påvirke hvilken verdi som går i hvilken variabel, altså M17.

Én elev har svart 1100, én har svart 999, og to har svart 1000. Siden de bare gir ett tall som output kategoriserer jeg disse som T4, men de førstnevnte elevene ser ut til å ha andre misoppfatninger i tillegg. Det kan se ut til at de kan ha noen vansker knyttet til titallssystemet, og en forestilling om at verdien 1 fra StortTall skal adderes eller trekkes fra ulike posisjoner fra LiteTall. Dette er det vanskelig å si noe mer om da oppgaven ikke var ment for å kartlegge dette, men svarene deres kan tyde på at det kan være noen vansker knyttet til dette.

Én elev har svart 1, 1000, 1000 og dette kan være på grunn av M9. Hen kan ha tenkt at StortTall er 1, 1000 mens LiteTall er 1000.

Én elev har svart 1, 1000, LiteTall. Dette kan være en kombinasjon av M9 og M15a. Om eleven har tenkt at StortTall vil gi 1, 1000 og LiteTall vil gi LiteTall kan svaret også forklares med M9 kombinert med M16.

De to elevene som har svart StortTall, LiteTall viser tegn til M15a.

Én elev har svart StortTall og dette kan være en kombinasjon av T4 og M15a.

Oppgave 8 - M18

Hva sier Katten?

Figur 36: Diagnostisk oppgave 8 for å undersøke M18

Misoppfatning M18: Rekkefølgen man oppretter og navngir variablene vil påvirke hvilken verdi som går til hvilken variabel. Eleven vil da svare 0, 3.

Elevenes svar	Frekvens	Type
3, 0	12	Riktig
0, 3	6	M18
3	1	T4
Poeng	1	T4 og M15a
AntallLiv, poeng	2	M15a
Poeng 0, antallLiv 3	1	T5
Vet ikke / blank	3	

Diskusjon

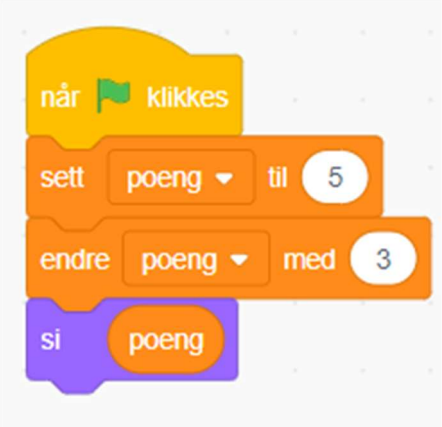
I denne oppgaven er det ingen variabler som endrer verdi gjennom programmet, og ingen muligheter for å svare feil på grunn av M9. Det er 12 elever som mestrer denne oppgaven og har gitt det korrekte svaret 3, 0.

6 elever har svart 0, 3, og dette kan være på grunn av M18, hvor man lar rekkefølgen variablene står i påvirke hvilken verdi som går til hvilken variabel.

Videre avslører denne oppgaven T4 (reduksjon av antall variabler), T5 (utvidet output) og M15a (gjengir variabelnavn fremfor variabelens verdi). Dette er vansker som har vist seg gjennom mange av de foregående oppgavene, og det er naturlig at de samme vanskene viser seg her også.

Oppgave 9 - M20

Hva sier Katten?



Figur 37: Diagnostisk oppgave 9 for å undersøke M20

Misoppfatning M20: Misoppfatninger knyttet til å endre en variabel. Eleven vil da svare 3.

Elevenes svar	Frekvens	Type
8	9	Riktig
2 eller 8	1	T7
2	2	T7
5, 3	3	M9
5	1	T3
3	2	M20
Poeng	2	M15a
Poeng 3	1	M20 og T5
Sier 5, endrer poeng med 3	1	T5 og M23
Vet ikke / blankt	4	

Diskusjon

Det er 9 elever som har svart riktig på denne oppgaven. De har skjønt at variabelen poeng blir tildelt verdien 5 og deretter øker med 3.

Én elev svarte 2 eller 8, og det viser at hen er usikker på om variabelen skal øke eller minke i verdi. Med litt erfaring vil man lære at i Scratch vil denne koden gi en økning av verdien. 2 elever har svart 2. De har høyst sannsynlig lest koden [endre poeng med 3] til at variabelen poeng skal avta med 3. Jeg kategoriserer disse som T7, en vanske som oppstår på grunn av feilaktig tolkning av syntaks/semantikk. Det er manglende syntaktisk kunnskap som gjør at vansken oppstår, noe som vil være lett å oppklare og korrigere for.

3 elever svarte 5,3 og jeg plasserer dette under M9, hvor man tenker at en variabel kan holde flere verdier samtidig.

Én elev svarte 5 og har dermed ignorert koden [endre poeng med 3]. Dette viser funn av T3.

2 elever har svart 3, og dette er funn av M20. De lar koden [endre poeng med 3] oppføre seg som [sett poeng til 3]. Om M20 oppstår som en konsekvens av manglende syntaktisk kunnskap, kan den kategoriseres som en vanske fremfor en misoppfatning.

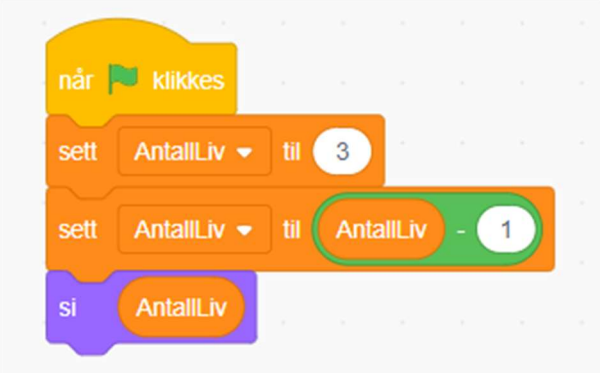
2 elever svarte poeng, som viser funn av M15a.

Én elev svarte poeng 3 og jeg antar dette er på grunn av en kombinasjon av M20 og T5.

Til slutt er det én elev som svarte “sier 5, endrer poeng med 3”. Dette er funn av T5, siden output er utvidet til noe mer enn forventet. Samtidig kan det se ut til at eleven har vansker med å forstå betydningen til rekkefølgen av koden, noe som tyder på M23.

Oppgave 10 - M20

Hva sier Katten?



Figur 38: Diagnostisk oppgave 10 for å undersøke M20

Misoppfatning M20: Misoppfatninger knyttet til å endre en variabel.

Programkoden i oppgave 9 og oppgave 10 vil oppføre seg likt. Oppgave 10 er tatt med for å undersøke om elevene klarer oppgave 9 eller 10 best.

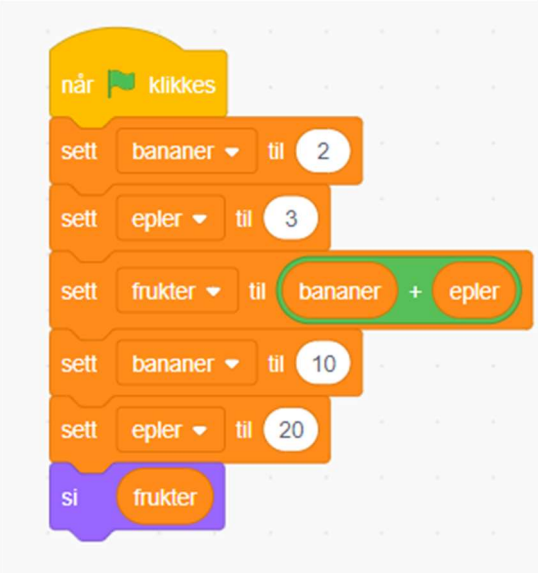
Elevenes svar	Frekvens	Type
2	15	Riktig
AntallLiv 2	1	T5
3, 3-1	1	M9 og M15b
3	1	T3
-1	1	T3
AntallLiv	2	M15a
3 liv på starten og så mister du et liv	1	T5
Vet ikke / blankt	4	

Diskusjon

Det er 15 elever som svarte riktig på denne oppgaven, mot 9 riktige svar i forrige oppgave. Dette viser at koden som er gitt i oppgave 10 gir mindre vansker enn koden i oppgave 9. Ellers kan svarene som kommer frem i denne oppgaven tyde på funn av M9, M15a, M15b, T3 og T5.

Oppgave 11 - M23

Hva sier Katten?



Figur 39: Diagnostisk oppgave 11 for å undersøke M23

Misoppfatning M23: Vansker med å forstå betydningen av rekkefølgen på koden.

Eleven vil da svare 30.

Elevenes svar	Frekvens	Type
5	4	Riktig
13	4	M23
35	2	M23 og T8
30	2	M23
20	1	T3 og M23
40	1	M23
10+20	1	M23 og M15b
10+3	1	M23 og M15b
Frukter	2	M15a
Bananer+epler	2	M15b
Vet ikke / blankt	6	

Diskusjon

4 elever svarte riktig på denne oppgaven. For å svare riktig må de ha skjønt betydningen av rekkefølgen på koden, og de forstår at selv om verdien til bananer og epler endres på slutten av programmet, vil ikke verdien til frukter endres.

De feilene som ellers er gjort viser flere funn av M23, men også M15a, M15b og T4 forekommer her. Jeg tenkte at dersom en elev hadde misoppfatning M23 ville hen ha gitt svaret 30 i denne oppgaven, men det viser seg at svaret 13 også dukker opp 4 ganger i denne oppgaven. Dette viser at 4 elever har hentet verdien til variabelen frukter fra tilfeldige steder i koden.

De to elevene som har svart 35 har antagelig summert alle tallene (2+3+10+20). Svaret tyder på at de har vansker både med å forstå betydningen av rekkefølgen på koden, men også har en oppfatning av at en variabel husker og adderer alle tidligere verdier. Den sistnevnte oppfatningen navngir jeg for T8.

2 elever svarte 30. Dette er funn av M23 og tyder på at de har vansker med å forstå betydningen av rekkefølgen på koden.

Én elev har svart 20. Eleven kan ha hatt fokus på nest siste linje i koden [sett epler til 20] og ignorert det meste av koden ellers. Ignorering av kode kategoriseres som T3. Ellers er det epler, og ikke frukter, som settes til 20 i koden. Dette viser at eleven kan ha vansker med å forstå betydningen av rekkefølgen til koden.

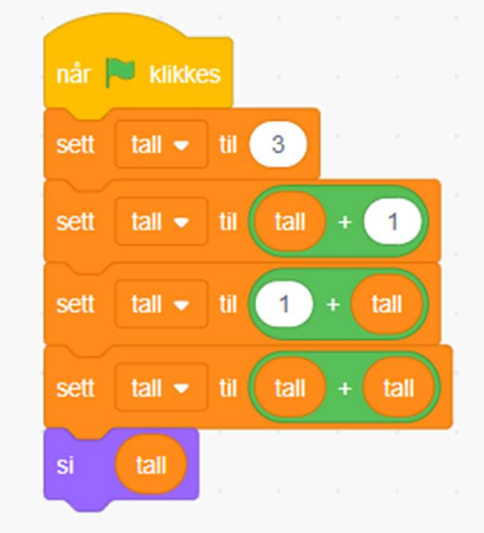
Én elev har svart 40. Hen kan ha brukt verdien 2 på bananer og verdien 20 på epler og samtidig lest addisjon som multiplikasjon. Om dette er tilfelle vises det en manglende forståelse for betydningen av rekkefølgen til koden, og dette hører til i kategorien M23.

De som svarte 10+20 og 10+3 har vansker med både betydningen til rekkefølgen på koden og at regnestykket skulle ha vært regnet ut. Dette er funn av M23 og M15b.

De to elevene som har svart frukter viser M15a, og de to som har svart bananer+epler viser M15b.

Oppgave 12 - sammensatt

Hva sier Katten?



Figur 40: Diagnostisk oppgave 12

Denne oppgaven er mer sammensatt og er tatt med siden den endrer verdien på en variabel flere ganger etter hverandre. Den er ikke ment å skulle avdekke noen nye typer misoppfatninger, men undersøke hvilke vansker som oppstår når en variabel endrer verdi flere ganger etter hverandre og dessuten bruker sin egen verdi i videre beregninger.

Elevenes svar	Frekvens	Type
10	10	Riktig
17	2	T8
3+1, 1+3, 3+3	1	M9, M15b, T3
3, 1+tall, tall+1	1	M9, M15b, T3
6	1	T3
3	2	T3
25	1	
Tall	2	M15a
Vet ikke / blank	6	

Diskusjon

I denne oppgaven har 10 elever svart riktig. De har en korrekt forståelse for hvordan en variabel endrer verdi flere ganger etter hverandre, og kan bruke variabelens verdi i videre beregninger.

2 elever har svart 17, og de kan ha regnet ut 3+4+5+5. Dette er ikke riktig og viser en misoppfatning hvor eleven antar at programmet adderer alle tidligere verdier som en variabel har hatt. En type misoppfatning som går ut på at en variabel husker og adderer alle tidligere verdier navngir jeg for T8.

Én elev har gitt svaret 3+1, 1+3, 3+3. Her vises tegn på flere misoppfatninger samtidig. Da eleven gjengir flere verdier på én variabel kan vi anta at M9 er til stede. Siden variabelens første verdi ikke er med i output kan man si at eleven har ignorert deler av koden og viser T3. Samtidig vises M15b fordi output blir gitt som uløste regnestykker. Videre oppdateres ikke variabelens verdi underveis i programmet.

Eleven som svarte 3, 1+tall, tall+1 har mye til felles med overnevnte elev. Misoppfatningene går ut på det samme, men det er litt ulike deler av koden som er ignorert og denne eleven har i tillegg ikke klart å sette inn en verdi for variabelen tall.

Én elev har svart 6. Det er mulig at hen har ignorert linjene [sett tall til tall +1] og [sett tall til 1+tall], for da ville dette svaret gitt mening. Ignorering av kode kategoriseres som T3.

2 elever har svart 3. Det er mulig at de har ignorert linjene [sett tall til tall+1], [sett tall til 1+tall] og [sett tall til tall+tall]. Ignorering av kode kategoriseres som T3.

Én elev har svart 25. Dette er det vanskelig å finne en naturlig forklaring på. Da det i andre oppgaver har vist seg å være vansker knyttet til titallssystemet og at en endring av verdien til en variabel kan knyttes til både økning og minking av ulike sifre, vil det være mulig at det er denne vansken som vises her.

2 elever har svart variabelens navn tall og viser M15a.

Oppgave 13 - sammensatt

Hva sier Katten?

Figur 41: Diagnostisk oppgave 13

Dette er en sammensatt oppgave og er ikke ment å skulle avdekke nye typer misoppfatninger. Den vil avdekke eventuelle vansker som oppstår når man har to variabler A og B samtidig, hvor B blir tildelt en verdi fra A, og når B endres enda en gang etter det igjen.

Elevenes svar	Frekvens	Type
2, 1	9	Riktig
A, B	1	M15a
2, 2, 1	1	M9
0, 1	1	M16
2, 2	1	M23 eller T3
2, 3	1	T8
1, 1	1	M11 og M23
2	2	T3 og T4
4	1	T3, T4 og T8
2, A, 1	2	M9, M15a
a2	1	M16 og T3
Vet ikke / blankt	5	

Diskusjon

I denne oppgaven var det 9 elever som svarte riktig. 5 valgte å ikke svare, og resten av elevene hadde veldig ulike svar.

I denne oppgaven var det én elev som svarte A, B. Dette er funn av M15a.

Én elev har svart 2, 2, 1 og viser funn av M9. Her har eleven sannsynligvis latt [si A] bli til 2, og [si B] bli til 2, 1.

Én elev har svart 0, 1 som viser funn av M16.

Én elev har svart 2, 2. Da har eleven kanskje tenkt at både A og B har fått verdien 2. Dersom man ignorerer [sett B til 1] vil dette være et naturlig svar. Ignorering av kode er navngitt som T3. Ellers kan eleven ha vansker med å forstå betydningen av kodens rekkefølge, som er kategorisert som M23.

Én elev har svart 2,3. Hen kan ha tenkt at A er 2 og at B er 3. Hvordan kan eleven komme frem til at B er 3? Det kan henge sammen med at B først ble satt til A, som var 2. Deretter endret B verdi til 1, men eleven har i stedet addert disse verdiene sammen. 2+1=3 og dermed er B=3. En oppfatning om at en variabel husker og adderer alle tidligere verdier er en misoppfatning av type T8.

Én elev har svart 1, 1. Dette kan være M11 kombinert med M23. Eleven har tenkt at både A og B har verdien 1 når programmet avsluttes. M11 kombinert med M23 gjør dette mulig. Hvis man tenker at A får sin verdi fra B, og B får verdien 1, og rekkefølgen ikke spiller noen rolle, da vil dette svaret kunne komme frem.

2 elever har svart 2. Her ser vi at output bare gir en verdi, noe som viser reduksjon av antall variabler, som kalles T4. Eleven har også ignorert kommandoen sett B til 1, noe som gjør at eleven samtidig viser T3.

Én elev har svart 4. Siden det er reduksjon i antall variabler kan man si at det er funn av T4, men det må være noe mer i tillegg siden verdien 4 fremkommer ikke noe sted i koden. Kanskje eleven har tenkt at A=2 og B=2, deretter ignorert at B=1, og til slutt summert A og B. Dette svaret kan tyde på at eleven har flere misoppfatninger på samme tid. Svaret kan forklares med T3, T4 og T8 samtidig.

2 elever har svart 2, A, 1. I dette tilfellet vil [si A] gi 2 og [si B] gi A, 1. Dette er funn av M9 sammen med M15a.

Én elev har svart a2. En mulighet er at eleven egentlig mente å skrive A, 2. Da vil [si A] gi A mens [si B] vil gi 2. En mulig tanke er at A ikke lenger har verdien 2 etter at B overtok denne verdien, og dette er tegn på M16. Samtidig har eleven ignorert at B senere får verdien 1, noe som tyder på T3.

5 Drøfting

Resultatene fra denne studien har blitt analysert fortløpende gjennom kapittel 4 etter hvert som de ble presentert. Noen generelle og gjennomgående aspekter er det likevel verdt å diskutere videre, og dette vil jeg belyse i dette kapittelet. For å se absolutt alle resultatene fra studien vises til kapittel 4, da jeg her i kapittel 5 kun har oppsummert og diskutert noen typiske og gjentagende trekk fra studien.

Elevers forkunnskaper, holdninger og forventninger til programmering

I den første spørreundersøkelsen hvor hele skolen ble invitert til å delta var et av funnene at det var mange som ikke hadde noe særlig erfaring med programmering fra tidligere, og blant de som hadde noe erfaring hadde mange av de brukt ulike programmeringsspråk. Dette gjør at man ikke kan møte en ny klasse på 8. trinn og ha noen forventing om at de har noe felles forkunnskaper innen programmering. Når man møter nye klasser på 8. trinn vil man alltid ha en forventning om at de har lært noe av det samme på barneskolene innen lesing, skriving, regning osv. Innen programmering ser det ut til at man ikke kan ha slike forventninger helt ennå, men det er sannsynlig at dette vil kunne endre seg i årene som kommer, og kanskje det allerede i skrivende stund vil være annerledes enn på det tidspunktet da jeg gjennomførte spørreundersøkelsen.

Videre viste det seg at 15,7 % av elevene mente at programmering er noe som alle elever bør lære seg, en holdning som viser sterk kontrast til dagens læreplaner hvor programmering er innført for alle. Om man skulle latt elevenes synspunkt være styrende ville programmering gjerne ha vært et valgfag på ungdomstrinnet og et programfag på videregående skole, hvor de med særlig interesse for dette faget kunne hatt mulighet til å velge det. I denne sammenhengen vil jeg komme med et par synspunkter. Jeg har gjennom en årrekke erfart at mange elever på ungdomstrinnet vil argumentere for at de ikke vil ha bruk for matematikk i livet sitt. Det er stort sett mulig å overbevise dem om at matematikk vil være nyttig for dem uansett hvilket yrke de velger, fordi de kan ha nytte av matematikkunnskapen i sitt eget privatliv. Det kan trekkes frem praktiske eksempler som matlaging, borddekking, personlig økonomi, planlegging av reiser, oppussing og innredning av en leilighet og mye mer. På denne måten vil man lett kunne overbevise de fleste elever om at matematikk på ungdomstrinnet er nyttig for alle. For veldig mange elever er disse praktiske sidene ved faget et tilstrekkelig fokus, og interessen for generaliseringer og formler er ikke alltid til stede. Når det gjelder programmeringen, som nå er blitt en del av matematikkfaget, ser jeg at det kan være utfordrende å finne like gode argumenter for å overbevise om at absolutt alle vil ha nytte av denne kunnskapen i sin fremtid. Selv om programmering vil være en større del av vår fremtid, og stadig flere yrker vil gjøre nytte av programmeringens fordeler, vil det antagelig være mange mennesker som fremdeles vil ha yrker der dette ikke er en nødvendig kompetanse. Av denne grunn ser jeg at det kan bli utfordrende å motivere absolutt alle for å lære om programmering, og tenker at det er ekstra viktig at elevene får positive erfaringer med programmering tidlig. Om man overkjører elevene med mye tung programmering er jeg redd for at det kan oppleves som skolen har innført «det nye latin», noe som ville vært veldig uheldig.

I dette prosjektet kom det frem at de fleste elevene i klassen likte best undervisningsøkten med programmering av spill, og de mente også at det var dette de lærte mest av. Det var

merkbart allerede i de første undervisningsøktene at elevene var mer interessert i å programmere spill enn å tegne geometriske mønstre, og det var flere elever som spurte om når de kunne få begynne å programmere spill. Selv om langt fra alle skal bli spillutviklere i fremtiden kan denne observasjonen være noe å bygge undervisningsoppleggene på. Det var også dette som førte meg inn på den siste undervisningsøkten med spillet «Sulten ape». Å bruke spill kan se ut til å ha en motiverende effekt på elevene, i hvert fall for elevene som deltok i dette prosjektet.

Vansker som kan oppstå knyttet til variabelbruk i programmering

Gjennom opptakene fra denne studien observerte jeg at flere elever hadde utfordringer med å forstå i hvilke situasjoner man burde opprette variabler. I noen tilfeller opprettet elevene ikke variabler når det hadde vært hensiktsmessig, og i andre tilfeller opprettet de variabler som ikke var hverken nødvendig eller i bruk senere. Videre observerte jeg at elevene kunne ta i bruk unaturlige navn på sine variabler, noe som igjen kunne skape mer forvirring for dem senere. Blant annet kunne dette medføre at variabler ble forvekslet med objekter som hadde samme navn. Det var flere tilfeller hvor elevene glemte å initialisere variabler, noe som medførte at programmet ikke oppførte seg som forventet. Disse funnene kan tyde på at allerede ved deklarering og initialisering av variabler vil det kunne oppstå vansker. I delkapittel 4.2 og 4.4 vises det flere tilfeller hvor disse ulike vanskene kommer til syne. Resultatene i denne studien tyder på at det kan være vansker med å forstå variablenes rolle i en programkode. Dette samsvarer med Kuittinen og Sajaniemi sin studie om undervisning i grunnleggende programmering (Kuittinen & Sajaniemi, 2004).

Resultatene fra denne studien avslører videre hvordan elevenes forkunnskaper fra vårt naturlige talespråk, engelsk og matematikk kan komme i veien eller forstyrre en korrekt oppfatning av programkodene, slik Bråting og Kilhamn (2021) og Qian og Lehman (2017) også har beskrevet. For å unngå at det skulle oppstå unødvendige vansker knyttet til det engelske språket oppfordret jeg elevene til å bruke norsk versjon av Scratch, men mange valgte å bruke engelsk likevel. Utfordringer knyttet til det engelske språket viste seg da noen elever ikke var klar over at det engelske [join] tilsvarte det norske [sett sammen], og det ble av denne grunn vanskelig for dem å skrive programkoden korrekt. Dette er beskrevet i kapittel 4.2. Videre hadde elevene ulike oppfatninger av norske koder som [Endre Tall med 1] og [Sett A til B], noe jeg vil utdype i neste avsnitt. Jeg observerte også at elevenes vansker innenfor matematiske fagområder gjorde det vanskelig for dem å skrive programkode til programmer som inneholdt matematikk. For eksempel var det i et tilfelle en elev som forvekslet ulikhetstegn med likhetstegn, noe som er beskrevet i kapittel 4.4. Et annet eksempel var at noen elever ikke visste hvordan man skulle finne omkretsen av et rektangel. Det viste seg at manglende matematisk kompetanse kunne være til hinder for programmeringen dersom programmeringen skulle brukes sammen med matematikk.

Videre viste resultatene at det oppsto mange misoppfatninger og vansker knyttet til tildeling og endring av verdien til variabler. Vanskene kom til syne gjennom opptakene og fra den avsluttende kartleggingstesten med diagnostiske oppgaver. For dette prosjektet hadde jeg valgt ut ni misoppfatninger fra Sorva sin liste med 162 misoppfatninger (Sorva, 2012, s. 358-368). Disse ni misoppfatningene var knyttet til tildeling og endring av verdien til variabler og var samtidig relevant for Scratch-brukere. Misoppfatningene ble navngitt M9, M11, M12, M15, M16, M17, M18, M20 og M23, i samsvar med Sorva (ibid) sin kategorisering, og disse

er nærmere beskrevet i tabell 3 i delkapittel 4.5. Av de misoppfatningene som jeg hadde et søkelys på var det flest observasjoner av M9, M15 og M23. M9 er en misoppfatning som går ut på at eleven antar at en variabel kan ha flere ulike verdier på samme tid eller at den kan huske gamle verdier. M15 går ut på at eleven gjengir uløste likninger eller uttrykk fremfor variabelens verdi. M23 handler om at man har vansker med å forstå betydningen av rekkefølgen på kodene. Ved M23 kan man ha en konkret plan på hva programmet skal gjøre, men plasserer kodene i en ulogisk rekkefølge. Det kan være at vansken oppstår på grunn av en manglende forståelse for tidsaspektet i kjøringen av et program. Kartleggingstesten i dette masterprosjektet viser dermed resultater som samsvarer med misoppfatningene som er presentert av Sorva (ibid). Videre leverte elevene i dette prosjektet inn svar som ikke passet hverken med en korrekt oppfatning eller samsvarte det forhåndsbestemte rammeverket. Jeg opprettet derfor åtte nye typer kategorier av vansker navngitt T1-T8. Disse er forklart i tabell 3 i delkapittel 4.5. Av disse åtte vanskene var det flest observasjoner knyttet til at elever kunne ignorere deler av koden (T3) og at de kunne redusere eller slå sammen variabler (T4). Én elev sine svar tyder på en misoppfatning om at en variabel kan være alle tall innenfor et bestemt intervall (T1). Siden definisjonen for en variabel er ulik i programmering og i matematikk, slik beskrevet i kapittel 2.2, vil denne forståelsen for en variabel innen programmering kunne kategoriseres som en misoppfatning. De andre kategoriene av vansker kommer også til syne i resultatene, og viser til sammen et bredt spekter av misoppfatninger og andre vansker som kan oppstå når man skal tildele og endre verdien til variabler. Hele oversikten er å finne i kapittel 4.5.

Videre drøfting av to kommandoer som tildeler verdier til en variabel

Jeg vil nå presentere to enkle koder som er knyttet til tildeling av verdier til variabler, og jeg vil vise hvilke misoppfatninger som kan oppstå i den samme koden. Jeg vil sette søkelys på kodene [Sett B til A] og [Endre A med 2], siden disse handler om å tildele en verdi til en variabel, og jeg vil beskrive hva studiens resultater viser om elevers oppfatning av disse kodene. Dette er mine egne betraktninger og er basert på resultatene og analysen som er presentert i kapittel 4.5.

[Sett B til A] ser ut til å kunne tolkes på mange ulike måter. Den riktige tolkningen er at B får samme verdi som A. Denne koden kan være opphav til mange ulike misoppfatninger, og jeg vil her oppsummere de tolkningene jeg har funnet i denne studien:

- A får samme verdi som B. (M11)
- A og B bytter verdier. (M12)
- B får A sin verdi og A blir til 0 eller blir helt tom. (M16)
- B får A sin verdi og behandles heretter som en A. Det er ikke lenger bruk for B og denne forsvinner vekk. (T4)
- B sin verdi legges til A, slik at A sin verdi øker med B. (T7)

Vi ser at den enkle koden [Sett B til A] kan tolkes på mange ulike måter for en som ikke har erfaring med å bruke programmering. Siden en kode alene kan være opphav til mange forskjellige tolkninger vil det være vanskelig å lage diagnostiske oppgaver som er entydig og kun kartlegger en misoppfatning av gangen. Likevel kan man gjennom slike oppgaver få verdifulle resultater som kan løftes frem i klasserommet og drøftes videre. Slik kan elever som har misoppfatninger bli satt i en kognitiv konflikt og få ryddet opp i misoppfatninger. En

slik undervisningsmetode er blitt beskrevet som diagnostisk undervisning av Brekke (2002, s.19).

Koden [Endre A med 2] kan også tolkes på ulike måter. Det riktige svaret er at verdien til A øker med 2. Mulige misoppfatninger for denne koden kan være:

- A beholder sin opprinnelige verdi, men får nå verdien 2 i tillegg. (M9)
- A minker med 2. (T7)
- A får verdien 2. (M20)

Det kommer tydelig frem at vårt tale- og skriftspråk er åpent for flere tolkninger enn man har i et programmeringsspråk. Der man i vanlig språk kan forstå et ord eller uttrykk ulikt i ulike situasjoner, vil man i programmeringen møte et mer rigid og entydig språk som alltid betyr det samme. Koder som [Sett B til A] og [Endre A med 2] vil alltid bety det samme, uansett hvilken kontekst man setter det inn i. Parallelt med det som er beskrevet av Bråting og Kilhamn (2021), tyder resultatene fra denne studien også på at de ikke-kongruente konverteringene mellom naturlig språk, matematikk og programmeringsspråk kan by på utfordringer.

6 Pedagogiske grep og endringer i egen undervisningspraksis

Som tidligere nevnt har jeg gjennomført denne studien med et mål om å utvikle og forbedre egen undervisningspraksis innen programmering i matematikkfaget. Siden jeg ikke hadde noe særlig erfaring med programmering i undervisningssammenheng før jeg begynte på masterstudiet vil jeg påstå at min egen læringskurve har vært bratt.

Hvordan har erfaringene fra denne studien betydning for min undervisningspraksis? Dette går på hvorvidt min aksjonsforskning har ført til endringer i undervisningsopplegget som jeg gjennomførte i 8. klassen som jeg besøkte. Ut ifra erfaringene fra prosjektet og i lys av den teori som er blitt kjent for meg har jeg laget et forslag til et nytt undervisningsopplegg. Det nye opplegget inneholder noen endringer og pedagogiske grep som jeg håper vil støtte elevene i læringsprosessen fra å lese kode til å skrive kode og hjelpe dem med å forstå variablenes rolle i en programkode.

Siden denne studien er bygget opp med utgangspunkt i aksjonsforskning skulle jeg helst ha gjennomført det nye undervisningsopplegget og sjekket om det faktisk viste en forbedring eller ikke. Dette rekker jeg ikke å gjennomføre på dette tidspunktet, men er noe jeg kan teste ut med nye elevgrupper senere. Man kan alltid forbedre egen undervisningspraksis som lærer, både underveis og i etterkant av studieløpet.

Nå vil jeg peke på noen sider ved dette prosjektets undervisningsøkter som kan forbedres.

6.1 Generelle endringer

I funnene fra denne studiens datamateriale fremkommer det at elevene stort sett hadde lite forkunnskaper om programmering, og derav også lite erfaring med bruk av variabler i programmeringen. Erfaringene deres var veldig ulike, og jeg kunne ikke møte dem med en forventning om at de hadde noe som helst felles forkunnskaper innen programmering. Deres forventninger og holdninger til programmering var også ganske delt. Ellers viste det seg at elevene både likte og, ifølge dem selv, også lærte mest av undervisningsøkten hvor de skulle lage et kunnskapsspill. Å utnytte dette i større grad i planlegging av undervisningen vil med høy sannsynlighet gi et bedre læringsutbytte, siden motivasjon er en viktig faktor for læringen. Å bruke spillutvikling som en innfallsvinkel til programmeringen kan altså være en faktor som kan være motiverende for elevene.

En annen side som jeg ønsker å trekke frem er å sette av mye tid til å snakke om begrepene før man skal ta i bruk begrepene i selve kodingen. Påstanden “most introductory programming classes do not put enough emphasis on fostering students' understanding of the concepts being used. As a consequence, the development of conceptual and strategic knowledge is almost entirely left to unguided discovery” (McGill & Volet, 1997, s.279) viser hvor viktig akkurat dette er. I mitt undervisningsopplegg hadde elevene en kort videoleksjon i hjemmelekse hvor de fikk en forelesning om variabler i programmering før de skulle ta denne kunnskapen i bruk på skolen. Likevel kunne skjerm- og lydopptakene avdekke at det var mange misoppfatninger og vansker knyttet til bruk av variabler, og dette tyder på at elevene kan trenge en enda grundigere og muligens flere gjennomganger av slike begreper. Jeg vil derfor ha et større fokus på dette når elever møter programmeringsoppgaver som krever bruk av variabler, løkker og betingelser, og jeg vil i større grad snakke med elevene om begrepene

i både forkant og underveis i læringen. Brekke (2002, s.19) foreslår diagnostisk undervisning, hvor man fremhever misoppfatninger som er observert blant elevene og løser disse ved å skape en kognitiv konflikt som gjennom diskusjoner og refleksjoner i undervisningen blir oppklart. Slike konfliktdiskusjoner vil jeg gjerne iverksette, for å øke sjansen for at eventuelle misoppfatninger blir ryddet av veien og at elevene får en korrekt konseptuell forståelse for de begrepene som de møter i programmeringen.

Elevene kan også i større grad få bruke analoge programmeringsøkter, i tråd med det som Gjøvik (2019, s.34) beskriver, før de skal ta i bruk et programmeringsspråk. På denne måten har de mer bakgrunn og erfaring før de selv skal skrive egne programmer. Dette er noe jeg gjerne vil bruke mer tid på sammen med elevene og vil forsøke å prioritere enda mer enn jeg fikk muligheten til gjennom denne studien.

Videre ønsker jeg å ta i bruk flytdiagram. Dette gjennomføres gjerne i etterkant av en analog programmeringsøkt eller etter at man har blitt presentert for et nytt problem. Morten Munthe formidlet på masterseminar (personlig kommunikasjon, 1.des 2022) at det kan være nyttig å lage et flytdiagram for hvordan man ønsker å løse et problem, slik at konverteringen mellom de ulike systemene blir mer tydelig for elevene. Ifølge Duval (2006, s.124) kan arbeid med slike konverteringer skape flere konflikter, men om det gjennomføres slik at programmeringsspråket blir en utvidelse av etablert kunnskap vil det kunne være et svært lærerikt steg å kunne mestre flere typer representasjoner på samme tid. (s.107). På samme måte som singaporemodellen (Ng & Lee, 2009) vil kunne støtte en konvertering mellom situasjon og algebra, vil et flytdiagram kunne være en støtte for konverteringer mellom løsningsstrategier uttrykt ved språk, algebraiske uttrykk og figurer over til et programmeringsspråk.

I det opprinnelige undervisningsopplegget, som jeg har gjennomført i en 8. klasse og som er å finne i vedlegg D, var det lagt opp til aktiviteter som lot elevene lese og tolke koder før de skulle skrive sine egne programmer. Denne rekkefølgen var planlagt med bakgrunn i teori fra Schulte og Busjahn (2013) og Lister et al (2009), som alle beskriver viktigheten av å kunne lese og forstå koder før man skal skrive egne programmer. Etter å ha blitt kjent med arbeidsmetoden PRIMM (Sentance & Waite, 2017) ønsker jeg å forbedre undervisningsopplegget ved å gi elevene flere situasjoner som ikke er knyttet til kun det første og det siste steget i PRIMM-modellen. I det reviderte undervisningsopplegget vil elevene i større grad få gjøre endringer i eksisterende programmer og teste hvordan endringene vil påvirke kjøringen av programmet. De vil også få oppgaver om å feilsøke programmer og rette opp feil i programkoden slik at programmet virker som ønsket. Denne typen aktiviteter gjør at man arbeider tettere på flere av stegene i PRIMM-modellen. Målet er å hjelpe elevene bedre i læringsprosessen fra å lese koder til å skrive egne programmer, og hjelpe dem til å forstå variablenes roller i programkoden.

6.2 Forslag til nytt undervisningsopplegg med endringer

I dette delkapittelet vil jeg vise til konkrete endringer som jeg vil gjøre i det opprinnelige undervisningsopplegget. Øktene er delt inn i fire adskilte opplegg med ulike tema. De ulike inndelingene er som følger: geometriske mønstre, omkrets av et rektangel, tallrekker og spill.

Undervisningsopplegg med geometriske mønstre

I de tre første undervisningsøktene som jeg hadde med klassen var temaet geometriske mønstre. De passer derfor naturlig sammen som et helhetlig undervisningsopplegg. Jeg vil i det følgende presentere en mer detaljert beskrivelse og forslag til endringer.

Første økt, en analog programmeringsøkt hvor elevene limte sammen regulære mangekanter og fikk frem flotte mønstre, ønsker jeg å beholde i det nye undervisningsopplegget. I tillegg ønsker jeg å legge inn en liten forelesning på starten av økten hvor elevene blir presentert for variabelbegrepet. De to neste øktene gjennomføres som en naturlig fortsettelse til denne aktiviteten.

I andre økt ønsker jeg å gjøre en del endringer. I stedet for å begynne med Scratch i denne timen ønsker jeg at elevene bruker timen til å lage et flytdiagram som beskriver detaljert hva som ble utført i økt 1. Elevene bytter deretter flytdiagrammer med hverandre og forsøker å utføre flytdiagrammene så nøyaktig som mulig. På denne måten vil de tegne hverandres mønstre. Til slutt velger klassen et av flytdiagrammene som viste seg å fungere godt. Dette flytdiagrammet deles med alle elevene i klassen og drøftes felles. I diskusjonen kan man stille spørsmål som: Hvorfor fungerte dette flytdiagrammet godt? Hvilke variabler ble tatt i bruk, og hvilken rolle spiller variablene gjennom hele flytdiagrammet? Flytdiagrammet lagres til neste økt og kan lette overgangen til programmeringsspråket. Det er mulig at flytdiagrammet kan gjøre det enklere for elevene å se når og hvorfor det er fordelaktig å opprette variabler.

Først i tredje økt vil elevene få utdelt et program hvor de skal lese koder og tolke hvilket mønster programmet vil tegne. Jeg vil endre litt på oppgavearket som de opprinnelig fikk utdelt i økt 2, slik at programmet som de får utdelt ikke tegner et digitalt 8-tall, men noe som minner mer om mønstrene som de jobbet med i de to første øktene. Å lese og tolke programmet vil være en aktivitet som tilsvarer det første steget P (predict) i arbeidsmetoden PRIMM. Deretter kjøres programmet og elevene undersøker om programmet oppfører seg slik de hadde forventet. Denne aktiviteten tilsvarer det andre steget R (run). Videre vil jeg be elevene om å forklare hver linje i programmet med egne ord, og der skal være lagt inn en feil som elevene må søke etter og rette opp. Feilen kan være av en simpel type, for eksempel at mønsteret ikke kommer helt rundt, men at det stopper opp litt for tidlig. Disse aktivitetene vil tilsvare I (investigate) i PRIMM. Videre får elevene oppgaver hvor de skal vurdere og teste hva som skjer dersom man gjør små endringer i eksisterende program. Hva skjer dersom man endrer på lengden og/eller rotasjonsvinkelen, og hva skal til for at mønsteret skal «bite seg selv i halen»? Slike endringer er situasjoner som dekker M (modify) i PRIMM. Elevene kan kjapt undersøke disse endringene ved å teste det ut i eksisterende program. Til slutt får elevene i oppgave å lage sine egne programmer som genererer de ønskede mønstrene, og de vil ha tilgjengelig flytdiagrammet fra økt 2. Dette er siste steget M (make). Da kan elevene ta utgangspunkt i det som allerede er gjort eller lage noe helt nytt. Ved å følge denne fremgangsmåten vil undervisningsopplegget innen geometriske mønstre ha aktiviteter som dekker alle stegene i PRIMM-modellen.

Undervisningsopplegg med omkretsen av et rektangel

Den fjerde økten var planlagt slik at elevene skulle pusle sammen gitte koder til et program som beregnet omkretsen til et rektangel. Jeg ønsker å utvide økten til å vare over minst to timer. Slik får elevene bedre mulighet til å mestre oppgaven, og de som blir fort ferdig kan

utvide programmet til å dekke flere figurer enn rektangelet. Jeg vil også gi elevene tilgang til, eller utarbeide sammen med elevene, et flytdiagram som kan være til hjelp for å plassere kodene i riktig rekkefølge. Det å sette opp koder i en feil og kanskje tilfeldig rekkefølge (M23) var en av vanskene som ble observert i denne studien, og et flytdiagram kan være til hjelp for å pusle kodene sammen i en logisk rekkefølge. Det er naturlig å gjennomføre denne økten i samme periode som klassen arbeider med omkrets innen plangeometri.

Undervisningsopplegg med tallrekker

Videre har jeg byttet litt om på rekkefølgen til de neste øktene, og ønsker nå å se på økten med tallrekker. Denne økten vil jeg beholde, men vil gjerne gjøre en del endringer. Erfaringene fra denne økten var at noen elever opprettet en variabel «tall» kun fordi det sto i oppgaveteksten at de skulle gjøre det, og deretter brukte de ikke variabelen på noen hensiktsmessig måte videre. Dette er beskrevet nærmere i kap. 4.2. For å minske sjansen for å havne i denne situasjonen vil jeg unngå å bruke oppgaveformuleringer som: «Lag en variabel som heter tall», og jeg vil i stedet gi elevene en aktivitet som i større grad dekker alle stegene i arbeidsmetoden PRIMM.

I det nye undervisningsopplegget vil elevene få utdelt et program som skal generere alle partallene opp til 20. Programmet som de får utdelt inneholder noen feil. Elevenes første oppgave blir å finne feilene og rette de opp slik at programmet fungerer som planlagt. Feilene innebærer at programmet ikke gir partallene i output, og at programmet dessuten starter på feil partall. Se figur 42 for utformingen av programmet som inneholder de nevnte feilene. Ved å feilsøke programmet (I) vil elevene lese og tolke kodene som ligger i programmet (P), de vil kjøre programmet gjentatte ganger for å sjekke om det oppfører seg som ønsket (R), og de kan gjøre endringer i koden som forbedrer programmet (M). Det finnes flere måter å løse oppgaven på, og de ulike løsningene kan drøftes i fellesskap. Til slutt lager elevene et eget program med en helt ny tallrekke som de selv har valgt ut.

Figur 42: Forslag til programkode som elevene skal feilsøke og korrigere.

Undervisningsopplegg med spill

Til slutt har jeg slått sammen det som opprinnelig var økt 5, 6 og 8. Ved å la denne økten gå over flere undervisningstimer vil jeg kunne finne mer tid til å gjennomgå begreper felles for klassen. Elevene får utdelt kodene til spillet, leser og tolker kodene, kjører programmet og tester om det oppfører seg som forventet. Videre får de noen nye oppgaver, som ikke var med i det opprinnelige opplegget, som handler om å gjøre små endringer i koden som å bytte ut apen med et annet dyr, endre startposisjonen for insektene, lage fem bananer ekstra osv. Elevene får i oppgave å legge til poeng i spillet for hver gang apen spiser en banan, og gjøre slik at man mister liv i spillet når apen berører insektene. Til slutt er det mulig å legge inn en tekst «game over» dersom man har mistet alle livene sine, og et nytt spillebrett kan komme til syne dersom man oppnår en viss poengsum. Det kan være aktuelt å se på et flytdiagram sammen, vansker som har oppstått underveis og oppklare begreper og koder som benyttes.

Endringene som er beskrevet ovenfor er oppsummert i tabell 4.

Opprinnelig undervisningsopplegg	Revidert undervisningsopplegg
Økt 1: Window patterns. Elevene fikk utdelt mangekanter som ble limt sammen til flotte mønstre.	**Økt 1:** Beholder økt 1 som den var, men legger inn en liten forelesning om begreper som variabel, løkke, input og output. De to neste øktene bør gjennomføres som en naturlig fortsettelse til denne økten.
Økt 2: Geometriske mønstre. Elevene fikk utdelt et program som skulle leses, tolkes og kjøres. Deretter skulle elevene lage sine egne programmer.	**Økt 2:** Starter med å lage et flytdiagram som viser detaljert hva som ble utført i økt 1. Elevene bytter på hverandres flytdiagram og utfører en analog programmeringsøkt. Til slutt velger klassen et av flytdiagrammene som viste seg å være velfungerende og dette deles og drøftes med hele klassen. Flytdiagrammets variabler og deres rolle gjennom kjøringen blir diskutert i plenum.
Økt 3: Geometriske mønstre - forts.	**Økt 3:** Elevene får utdelt et program som inneholder noen små feil. De leser programmet og antyder hva programmet gjør, og de tester deretter programmet. Videre får de i oppgave å beskrive hver eneste linje i programmet, og feilsøke for å finne hvor feilen befinner seg. Denne feilen skal de selvsagt korrigere for. Det blir lagt inn noen oppgaver hvor elevene skal vurdere og teste hva som skjer dersom man gjør små endringer i eksisterende program. Hva skjer dersom lengden/vinkelen økes/minkes, og hva skal til for at mønsteret skal «bite seg selv i halen»? Til slutt lager de sitt eget program hvor målet er å få frem noen av de samme mønstrene som ble introdusert i økt 1 og 2. Elevene har tilgjengelig flytdiagrammet fra økt 2.
Økt 4: Puslespill - Omkrets av et rektangel	**Økt 4**: Denne økten beholdes som den er, men kan med fordel utvides over flere økter. I tillegg til at elevene får utdelt puslespillet kan de også få utdelt, eller utarbeide selv, et flytdiagram som viser hvordan man beregner omkretsen til et rektangel. De som blir ferdig tidlig kan lage flere programmer som beregner omkrets og areal av andre figurer eller utvide eksisterende program til å omfatte flere figurer. Det er naturlig å gjennomføre dette opplegget i samme periode som klassen jobber med plangeometri.
Økt 7: Tallrekker	**Økt 7:** I denne økten vil jeg ta bort den instruerende formuleringen «Lag en variabel som heter Tall», og jeg vil endre økten slik at den står tettere i tråd med arbeidsmetoden PRIMM. I denne økten får elevene utdelt et program som inneholder noen feil. Oppgaven blir å finne feilene og rette opp for dette. Elevene får altså lese og tolke et gitt program om tallrekker, teste om kjøring av programmet gir forventet resultat, gjøre endringer i eksisterende program og til slutt lage sin egen tallrekke og et program som gir ut tallrekken.
Økt 5: Kunnskapsspill	**Økt 5, 6 og 8:** Disse tre øktene samles til et samlet undervisningsopplegg. Tar utgangspunkt i spillet «sulten ape», og lar elevene jobbe med dette over flere timer. Slik blir det lettere å finne tid til å drøfte vanskelige begreper og vansker som oppstår underveis. Elevene får utdelt kodene til spillet, leser og tolker kodene, kjører programmet og tester om spillet oppfører seg som forventet. Videre får de noen nye oppgaver om å gjøre små endringer i koden som å bytte ut apen med et annet dyr, endre startposisjonen for insektene, lage 5 bananer ekstra osv. Videre får elevene i oppgave å legge til poeng i spillet for hver gang apen spiser en banan, og at man mister liv når apen berører insektene. Til slutt er det mulig å legge inn en «game over» dersom man har mistet alle livene sine og et nytt spillebrett kan komme frem dersom man når 10 poeng. Det er mulig å benytte flytdiagram i denne økten også.
Økt 6: Fortsette kunnskapsspill	
Økt 8: Spill med poeng – sulten ape	

Tabell 4

7 Avslutning og konklusjon

Gjennom aksjonsforskningen i denne masterstudien har målet vært å opparbeide erfaringer med å bruke programmering i matematikkundervisningen og utforske hvilke utfordringer elever kan oppleve når de tar i bruk programmering i matematikkfaget. Det har vært et særlig fokus på hvilke misoppfatninger og andre vansker som kan oppstå når man tar i bruk variabler i programmeringen.

Tre forskningsspørsmål ble formulert for prosjektet. De handlet om elevenes erfaringer, forventninger og holdninger til programmering, hvilke misoppfatninger og andre vansker elever opplever i møte med variabler i programmeringen, og til slutt hvilke pedagogiske grep lærere kan gjøre i undervisningen for å hjelpe elevene til å bryte eventuelle barrierer som skaper vansker i programmeringen.

Datamaterialet som ligger til grunn i denne studien er innhentet gjennom åtte programmeringsøkter som jeg har gjennomført i en 8. klasse. I løpet av programmeringstimene ble det samlet inn en mengde datamateriale gjennom både kvalitative og kvantitative metoder. Det ble gjennomført skjerm- og lydopptak, spørreundersøkelser og elevene skrev logg i etterkant av undervisningstimene. I etterkant av de åtte øktene gjennomførte klassen en diagnostisk kartleggingstest med fokus på noen utvalgte misoppfatninger hentet fra Sorvas oversikt med 162 misoppfatninger (2012, s. 358-368).

Funnene fra datamaterialet i denne studien viser at elevene hadde lite og sprikende forkunnskaper innen programmering. Over en tredjedel av elevene hadde ingen eller liten erfaring med bruk av programmering. Det viste seg at elevene hadde delte holdninger til programmering og ulikt syn på programmeringens nytteverdi. Det kom også frem at elevene som deltok i denne studien foretrakk å bruke programmering til å utvikle spill. Ellers viser resultatene at det kan være mange ulike misoppfatninger og vansker knyttet til variabelbruk i programmeringen. Først og fremst viser det seg at elever kan ha vansker med å forstå når og hvorfor det er nødvendig å ta i bruk variabler. Å forstå variablene sin rolle i en programkode har vist seg å være en utfordring for flere av elevene som deltok i denne studien. Studien viser eksempler på hvordan vårt naturlige språk og forkunnskaper innen matematikk kan være med på å skape vansker med å forstå det valgte programmeringsspråket, og i denne oppgaven er det først og fremst vanskene knyttet til tildeling og endring av verdien til variabler som er trukket frem.

Blant misoppfatningene som jeg har observert i denne studiens diagnostiske kartleggingstest finnes blant annet M9 (En oppfatning om at en variabel kan ha flere ulike verdier på samme tid / kan huske gamle verdier), M15 (En oppfatning om at variabler som tildeles en verdi via et uttrykk lagrer likninger eller uløste uttrykk) og M23 (Problemer med å forstå betydningen av rekkefølgen på koden). I tillegg til å observere misoppfatninger innenfor det forhåndsbestemte rammeverket kom det også frem noen andre typer misoppfatninger. Disse har jeg navngitt T1-T8. Av særlig interesse er funnet av misoppfatning T1 (En variabel kan være alle tall innenfor et bestemt intervall), da dette kan være en direkte konsekvens av at variabelbegrepet har ulik betydning innen matematikk og programmering. Ellers har jeg observert at elever kan ignorere deler av koden (T3), redusere eller slå sammen variabler (T4) og mangle forståelse for verdien til en variabel når den endrer verdi underveis i kjøringen av

et program. Til slutt viser det seg at i mange tilfeller gis det flere ulike feilaktige svar i de diagnostiske oppgavene, noe som kan tyde på at det kan eksistere flere ulike typer misoppfatninger innenfor en og samme oppgave. Det er også noen elever som viser tegn til å ha flere misoppfatninger samtidig. Selv om det ikke alltid er mulig å trekke en entydig slutning mellom de feilaktige svarene og de nevnte misoppfatningene, så vil resultatene likevel belyse ulike vansker som er knyttet til bruk av variabler og vise hvilke misoppfatninger det er sannsynlig at elevene kan ha.

For å lage gode undervisningsøkter i programmering har jeg kommet frem til en del avsluttende idéer og pedagogiske grep som jeg gjerne vil prøve ut i videre praksis. Kort oppsummert handler det om å følge arbeidsmetoden PRIMM tettere og legge til rette for flere oppgaver som styrer elevene inn mot de midterste stegene i PRIMM-modellen. Ved å jobbe mer systematisk med aktiviteter som bygger på I (investigate) og M (modify) i PRIMM-modellen, vil man kunne få en bedre læringsprosess enn å gå direkte fra å lese kode til å skrive egne programmer. Aktiviteter som er spesielt rettet mot dette kan være kodesporing, feilsøking, gjøre justeringer i eksisterende program og å forklare kodene i et program linje for linje. Videre ønsker jeg å ta i bruk flytdiagrammer i etterkant av analoge programmeringsøkter, som en tredje type representasjon som binder sammen programmeringsspråket med løsningsstrategien for problemet. Da det viste seg at elevene mente at de både likte og lærte mest av øktene med spillutvikling, kan det være verdt å la elevene få arbeide med programmering knyttet til spill, men jeg vil også ha økter som har fokus på matematiske begreper som for eksempel omkrets og tallrekker. Til slutt vil jeg prioritere å sette av mer tid til å gjennomgå begreper som variabel og løkke før elevene behøver disse for å løse nye problem, og fremheve vanlige misoppfatninger slik at disse kan oppklares på et tidlig tidspunkt.

Gjennom denne studien har jeg erfart hvordan programmering i matematikkundervisningen kan være både til glede og til ergrelse for elevene. Jeg vil trekke frem igjen et sitat fra en jente på gruppe 1, som i økt 4 oppdaget at programmet som de hadde laget kunne brukes for å løse en hel klasse av problemer. Jenten som oppdaget dette virket veldig tilfreds med denne oppdagelsen hun hadde gjort og hun uttrykte: «Å, så det er på en måte en kalkulator? Da kan vi jo bruke den i matten.» Slike øyeblikk viser at eleven virkelig har skjønt både matematikken som ligger til grunn og hvordan programmeringen kan brukes som et nyttig verktøy. Å lære å bruke matematikk sammen med programmering vil kunne gi elevene muligheten til å løse mer komplekse problemer på en effektiv måte. Når man har opparbeidet en viss kompetanse innen valgt programmeringsspråk kan man også bruke programmering til å utforske matematikk. I dette prosjektet var oppgavene som elevene jobbet med på et grunnleggende nivå, og de matematiske fagområdene som skulle tas i bruk var kjent for dem på forhånd. Likevel var det et stort steg for denne elevgruppen å oppdage denne måten å bruke programmering sammen med matematikk, og hvordan de kunne gjøre nytte av å bruke programmering i matematikkfaget.

Ellers har jeg gjennom denne studien lært mye om misoppfatninger og andre vansker som kan oppstå knyttet til variabelbruk i programmeringen. Jeg har utvidet min kompetanse innen hvordan man kan planlegge, gjennomføre og evaluere programmeringsøkter. Jeg skjønner samtidig at dette er et område som det er mulig å forske mye mer på enn hva jeg har hatt mulighet til i denne avgrensende oppgaven. Jeg vil fortsette å lære mer innen dette området i

min videre praksis. Det nye undervisningsopplegget vil bli prøvd ut i egne klasser og jeg vil helt sikkert oppdage enda flere muligheter til forbedringer. Jeg håper og tror at erfaringene jeg har gjort gjennom denne studien vil kunne gjøre meg i bedre stand til å planlegge gode undervisningsøkter for mine fremtidige elever samt å kunne hjelpe dem gjennom de misoppfatninger og vansker som måtte oppstå knyttet til bruk av variabler i programmeringen.

«If we are to expand and improve computer science education, more research on students' misconceptions, especially on the evolution of (mis)conceptions, is needed and appropriate instructional strategies and tools to address them must be developed and disseminated.» (Qian & Lehman, 2017, s.18). Med dette sitatet vil jeg avslutte masteroppgaven, og jeg vil meddele at jeg føler meg heldig som har fått en sjelden mulighet til å fordype meg i et tema som jeg finner interessant og som samtidig er relevant for min yrkesutøvelse. Jeg er veldig takknemlig for at jeg fikk denne muligheten, og gleder meg til å fortsette å lære enda mer om bruk av programmering i matematikkfaget i fremtiden.

8 Litteraturliste

Befring, E. (2007). *Forskingsmetode med etikk og statistikk* (2. utg., p. 240). Samlaget.

Befring, E. (2015). *Forskningsmetoder i utdanningsvitenskap* (p. 176). Cappelen Damm akademisk.

Bergem, O. K., Kaarstein, H., Nilsen, T., Blömeke, S., Scherer, R., & Frøyland, M. (2016). *Vi kan lykkes i realfag - Resultater og analyser fra TIMSS 2015.* Universitetsforlaget.

Brekke, G., Kvalitet i matematikkundervisningen, & Læringssenteret. (2002). *Introduksjon til diagnostisk undervisning i matematikk* (Bokmål[utg.]. ed., p. 25). Læringssenteret.

Brekke, G., Grønmo, L. S., Rosén, B., & Nasjonalt læremiddelsenter. (2000). *Veiledning til algebra: F, H og J* (Bokmål[utg.]. ed., p. 93). Nasjonalt læremiddelsenter.

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning, 23(2)*, 170–185. https://doi.org/10.1080/10986065.2020.1779012

Bocconi, S., Chioccariello, A. and Earp, J. (2018). The Nordic approach to introducing Computational Thinking and programming in compulsory education. Report prepared for the Nordic@BETT2018 Steering Group. https://doi.org/10.17471/54007

Clancy, M., Stasko, J., Guzdial, M., Fincher, S., & Dale, N. (2001). Models and Areas for CS Education Research. Computer Science Education, 11(4), 323–341. https://doi.org/10.1076/csed.11.4.323.3827

Creswell, J. (2012). *Educational research: planning, conducting, and evaluating quantitative and qualitative research.* (4.edition). University of Nebraska-Lincoln.

Doukakis, D., Grigoriadou, M., and Tsaganou, G. (2007). Understanding the Programming Variable Concept with Animated Interactive Analogies. In: *Proceedings of the The 8th Hellenic European Research on Computer Mathematics & its Applications Conference, HERCMA '07.*

Du Boulay, B. (1986). Some Difficulties of Learning to Program. Journal of Educational Computing Research, 2(1), 57–73. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

Duval, R. (2006). A Cognitive Analysis of Problems of Comprehension in a Learning of Mathematics. *Educational Studies in Mathematics, 61(1/2),* 103–131. https://doi.org/10.1007/s10649-006-0400-z

Gibbs, W. (1999). *Mathematical Window Patterns the art of creating translucent designs using geometric principles.* Stradbroke: Tarquin Publications.

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

Gray, S. S., Loud, B. J., & Sokolowski, C. P. (2009). Calculus Students' Use and Interpretation of Variables: Algebraic vs. Arithmetic Thinking. *Canadian Journal of Science,*

Mathematics and Technology Education, 9(2), 59–72. https://doi.org/10.1080/14926150902873434

Grønmo, L., Onstad, T., Nilsen, T., Hole, A., Aslaksen, H., & Borge, I. (2012). *Framgang, men langt fram : Norske elevers prestasjoner i matematikk og naturfag i TIMSS 2011*. Oslo: Akademika.

Inkrement as. (2023, 20.mars). *Bli trygg i programmering med Campus Matte.* Campus Inkrement. https://campus.inkrement.no/Blogg/Bli-trygg-i-programmering-med-Campus-Matte

Kaarstein, H., Radišić, J., Lehre, A.C., Nilsen, T. & Bergem, O.K. (2020). *TIMSS 2019. Kortrapport.* Institutt for lærerutdanning og skoleforskning, Universitetet i Oslo.

Kaczmarczyk, L., Petrick, E., East, J., & Herman, G. (2010). Identifying student misconceptions of programming. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 107–111. https://doi.org/10.1145/1734263.1734299

Kilpatrick, J., Swafford, J., Findell, B., & National Research Council. (2001). Adding it up : Helping children learn mathematics. Washington, DC: National Academy Press.

Kuittinen, M., & Sajaniemi, J. (2004). Teaching roles of variables in elementary programming courses. *ITiCSE 2004 : Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education : June 28-30, 2004, Leeds, United Kingdom, 36(3),* 57–61. https://doi.org/10.1145/1026487.1008014

Lister, R., Fidge, C., & Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE Bulletin, 41(3)*, 161–165. https://doi.org/10.1145/1595496.1562930

Matematikksenteret. Nasjonalt senter for matematikk i opplæringen. (2023, 23.mars). *Didaktisk grunnlag.* https://www.matematikksenteret.no/l%C3%A6ringsressurser-og-undervisningsopplegg/programmering/anbefalt-arbeidsmetodikk

McGill, T., & Volet, S. (1997). A Conceptual Framework for Analyzing Students' Knowledge of Programming. *Journal of Research on Computing in Education, 29(3)*, 276-297. https://doi.org/10.1080/08886504.1997.10782199

Mughal, K.A., Hamre, T., Rasmussen, R. W. (2003). *Java som første programmeringsspråk.* Cappelens Forlag, Oslo 2003.

Ng, S. F., & Lee, K. (2009). The Model Method: Singapore Children's Tool for Representing and Solving Algebraic Word Problems. *Journal for Research in Mathematics Education, 40(3),* 282–313. https://doi.org/10.5951/jresematheduc.40.3.0282

Popat, S., & Starkey, L. (2018). Learning to code or coding to learn? A systematic review. *Computers and Education, 128*, 365–376. https://doi.org/10.1016/j.compedu.2018.10.005

Steinbring, H. (2006). What Makes a Sign a Mathematical Sign? An Epistemological Perspective on Mathematical Interaction. *Educational Studies in Mathematics, 61(1/2)*, 133–162. https://doi.org/10.1007/s10649-006-5892-z

Qian, Y. & Lehman, J. (2017). Students`Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ. 18*, 1, Article 1 (October 2017), 24 pages. https://doi.org/10.1145/3077618

Roness, D. (Red.). (2016). *Å forske på egen praksis.* Bergen: Fagbokforlaget.

Russell, B. (1937). *The principles of mathematics* (2nd ed.). London: Allen & Unwin.

Schulte, C. & Busjahn, T. (2013). The use of code reading in teaching programming. *Conference: Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. 3–11. https://doi.org/10.1145/2526968.2526969

Sentance, S. & Waite, J. (2017). PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. *In Proceedings of WiPSCE`17, Nijmegen, Netherlands, November 8-10*, 2017, 2 pages. https://doi.org/10.1145/3137065.3137084

Sentance, S. & Waite, J. & Kallia, M. (2019) Teaching computer programming with PRIMM: a sociocultural perspective, *Computer Science Education, 29*:2-3, 136-176, https://doi.org/10.1080/08993408.2019.1608781

Sorva, J. (2012). *Visual program simulation in introductory programming education*. PhD Thesis, Aalto University. (2012). http://urn.fi/URN:ISBN:978-952-60-4626-6

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education, 13(2)*, 1-31. https://doi.org/10.1145/2483710.2483713

Statistisk sentralbyrå. (2022. 7.august). Variabeldefinisjoner. Hentet fra https://www.ssb.no/a/metadata/definisjoner/variabler/main.html

Store norske leksikon (2023. 29.mars). programmering – IT – Store norske leksikon (snl.no)

Swidan, A., Hermans, F. & Smit, M. (2018). Programming Misconceptions for School Students. *Proceedings of the 2018 ACM Conference on international computing education research, 2018-08-08*, p.151-159. https://doi.org/10.1145/3230977.3230995

Tangenten, tidsskrift for matematikkundervisning, 3/2020, 31.årgang, Caspar Forlag AS

Utdanningsdirektoratet. (2022, 9.februar). *Den internasjonale studien TIMSS.* Hentet 15.02.2023 fra https://www.udir.no/tall-og-forskning/internasjonale-studier/timss/

Utdanningsdirektoratet. (2020). *Læreplan i matematikk 1.-10.trinn (MAT01-05).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nob

Wainewright, M. (2018). *Lær å kode. Datakoding trinn for trinn*. Cappelens forlag.

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

Vedlegg A - Samtykker

Vil du delta i forskningsprosjektet
”Programmering i matematikkfaget”

Dette er et spørsmål til deg som er elev om å delta i et forskningsprosjekt hvor formålet er å finne ut mer om hvordan programmering kan brukes på en god måte i matematikkfaget. I dette skrivet gir jeg deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål

I tillegg til å være lærer er jeg (Marita Orvedal) også masterstudent ved Universitetet i Bergen. Jeg følger studieprogrammet “Erfaringsbasert master i undervisning med fordypning i matematikk” og skal nå i gang med å skrive en oppgave om programmering i matematikkfaget.
I denne forbindelse ønsker jeg å samle inn datamateriale fra elever som jobber med programmering. Målet er å finne ut mer om hvordan programmering kan brukes i matematikkfaget på en måte som fremmer elevers kritiske tenkning og legger til rette for problemløsing, samarbeid og faglige samtaler.

Hvem er ansvarlig for forskningsprosjektet?
Universitetet i Bergen er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta, og hva innebærer det for deg å delta?
Din klasse er blitt valgt ut til å delta i dette prosjektet. Siden jeg jobber på 10. trinn og elevene der ikke skal ha programmering i år, så har din matematikklærer/kontaktlærer og rektor sagt at det er greit at jeg besøker din klasse og jobber med programmering sammen med dere.
Dere som deltar i prosjektet vil få være med i et undervisningsopplegg hvor dere jobber med noen utvalgte oppgaver innen programmering. I disse timene vil det bli tatt skjermopptak og lydopptak. Det skrives også en liten logg på slutten av timen.
Noen av dem som sier ja til å være med på undersøkelsen vil også bli intervjuet.

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

Ditt personvern – hvordan jeg oppbevarer og bruker dine opplysninger
Jeg vil bare bruke opplysningene om deg til formålene jeg har fortalt om i dette skrivet. Jeg behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Ingen elever vil kunne bli gjenkjent i oppgaven som jeg skal skrive.

Hva skjer med opplysningene dine når jeg avslutter forskningsprosjektet?
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er 30.06.22. Ved prosjektslutt slettes alle skjermopptak og lydopptak. Kun resultatene av prosjektet vil bli lagret, og disse vil ikke kunne spores tilbake til den enkelte deltaker.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Hva gir meg rett til å behandle personopplysninger om deg?
Jeg behandler opplysninger om deg basert på ditt samtykke.
Prosjektet er godkjent av UiB RETTE og det er vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan du finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:
- Marita Orvedal: tlf. 95844332, mail: marita.orvedal@student.uib.no
- Eventuelt kan du ta kontakt med min veileder ved Universitetet i Bergen: Christoph Kirfel, christoph.kirfel@uib.no

Med vennlig hilsen
Marita Orvedal

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om Marita Orvedal sitt forskningsprosjekt "programmering i matematikkfaget", og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ Å delta i skjermopptak
- ☐ Å delta i lydopptak
- ☐ At skjermopptak fra min skjerm kan brukes i presentasjoner.
- ☐ At lydopptak av meg kan brukes i presentasjoner.
- ☐ Jeg samtykker til at mine opplysninger kan behandles frem til prosjektet er avsluttet.

(Signert av prosjektdeltaker, dato)

(Signert av prosjektdeltakers foresatte, dato)

Vedlegg B – Skjematisk oversikt over datainnsamling i klassen

Tidspunkt for gjennomføring	**Hva**	**Datamateriale**
Desember 2020	Spørreundersøkelse på en hel skole.	Google Skjema.
Man 7.des 2020	Møte med klassens faglærer i matematikk. Gjøre avtaler for gjennomføring.	
Ons 9.des 2020	Elevene får utdelt samtykkeskjema.	
Man 14.des 2020	**Økt 1: Window patterns** Klipp og lim. Lag stjerner til å henge i vinduene. Lage kontoer i Scratch. Sjekke at elevene har tilgang på campus og WeVideo.	Observasjon og logg. Bilder av resultatet. Samle inn samtykkeskjema.
	Lekse: Se campus-video 8.1 Kommandoer og 8.4 Geometriske figurer.	
Man 21.des 2020	**Økt 2: Geometriske mønstre.** 1) Tolke og forklare hva et gitt program gjør. 2) Lage eget program som tegner en T, en regulær mangekant, og et eget mønster for eksempel stjerne, snøfnugg eller mønstre som vi har klippet i timen før.	Skjermopptak og lydopptak. Logg.
Man 18.jan 2021	**Økt 3: Geometriske mønstre - forts.** 1) Gi elevene kode for fullversjon på WeVideo 2) Hvorfor fikk dere ikke den mangekanten som dere planla å lage? 3) Lag flere regulære mangekanter 4) Lag mønster som vi har klippet. 5) Ekstra utfordring: Søk på nettet og finn ut hva en fraktal er. Kan du lage et program som tegner en fraktal?	
	Lekse: Se campus-video 8.3 Variabler for avstand.	
Man 25.jan 2021	**Økt 4: Puslespill - Omkrets av et rektangel.** Få oppgitt kommandoene som skal brukes i et program. Få oppgitt hva programmet skal gjøre. Pusle de sammen og teste om programmet virker som planlagt. Gjøres først med fysiske puslebrikker og deretter i Scratch. Hvis tid: Utvid programmet slik at det også beregner arealet. Lag gjerne tilsvarende program for andre figurer hvis det er mer tid igjen.	Skjermopptak og lydopptak. Logg.
	Lekse: Se campus-video 8.2 Hvis og løkker	
Man 8.feb 2021	**Økt 5: Kunnskapsspill.** Lage et spill hvor en figur skal gå fra start til mål og få oppgaver underveis. Dersom man svarer riktig, forflytter man seg nærmere mål. Ved mål kommer det opp en gratulasjon og kanskje en lyd.	Skjermopptak og lydopptak. Logg.
Man 15.feb 2021	**Økt 6: Fortsette kunnskapsspill**	Logg. Delte programmer. Ingen opptak.
Man 22.feb 2021	**Økt 7: Tallrekker**	Skjermopptak og lydopptak. Logg.
Man 22.feb 2021	**Spørreundersøkelse**	Google Skjema
Man 7.juni 2021	**Økt 8: Spill med poeng** Gi elevene spillet "Sulten ape". De skal bruke en variabel til å gi poeng. Dersom de får 10 poeng bytter bakgrunnen, og de kan lage sitt eget spill. Deles med klassen etterpå.	Skjermopptak og lydopptak. Logg.
Ons 9.juni 2021	**Kartleggingstest: Programmering med variabler**	Google Skjema

Vedlegg C – Spørreskjema for en hel skole

Spørreskjema om programmering

1. Hvilket klassetrinn går du på?

   - ◯ 8. trinn
   - ◯ 9. trinn
   - ◯ 10. trinn

2. Er du jente eller gutt?

   - ◯ Jente
   - ◯ Gutt

3. Tenk på ordet programmering. Hvilke av adjektivene nedenfor synes du passer til programmering? Kryss av for alle ordene som du synes passer.

   Merk av for alt som passer

   - ☐ Spennende
   - ☐ Vanskelig
   - ☐ Lett
   - ☐ Kjedelig
   - ☐ Unødvendig
   - ☐ Nyttig
   - ☐ Gøy
   - ☐ Lekende
   - ☐ Utfordrende
   - ☐ Viktig
   - ☐ Uforståelig
   - ☐ Strukturert

4. Tenk på en person som driver med programmering. Hvilke av adjektivene nedenfor synes du passer til en person som driver med programmering? Kryss av for alle ordene som du synes passer.

Merk av for alt som passer

- [ ] Morsom
- [ ] Kjedelig
- [ ] Humoristisk
- [ ] Nerdete
- [ ] Stille
- [ ] Pratsom
- [ ] Kjekk
- [ ] Dyktig
- [ ] Kunnskapsrik
- [ ] Overlegen
- [ ] Skoleflink
- [ ] Usosial
- [ ] Sosial
- [ ] Normal
- [ ] Sporty

5. Har du selv noe erfaring med programmering? Kryss av for det alternativet som passer best for deg.

- ◯ Jeg har aldri programmert.
- ◯ Jeg har programmert lite.
- ◯ Jeg har programmert noe.
- ◯ Jeg har programmert en god del.
- ◯ Jeg har programmert mye.
- ◯ Vet ikke.

6. Hvis du har noe erfaring med programmering, hvor og når jobbet du med dette? Det er mulig å krysse av på flere alternativer. Jeg har programmert...

Merk av for alt som passer

- [ ] ...på fritiden
- [ ] ...i barnehagen
- [ ] ...på barneskolen
- [ ] ...i valgfag på ungdomsskolen
- [ ] ...i et annet fag på ungdomsskolen
- [ ] ...i kodetimen eller lignende
- [ ] ...et annet sted

7. Hvis du har erfaring med programmering, utdyp her hvilket programmeringsspråk du kjenner til:

_____

8. Hvilken av påstandene nedenfor synes du er mest riktig?

- ◯ Alle bør lære seg programmering.
- ◯ Noen bør lære seg programmering.
- ◯ Ingen trenger å lære seg programmering.
- ◯ Vet ikke.

9. Hvilken påstand nedenfor mener du er riktig?

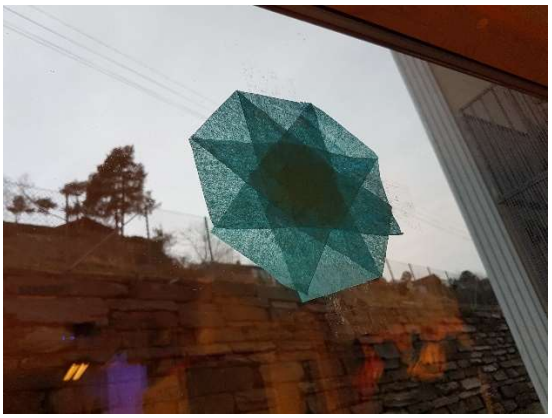
- ◯ Hvis du skal bli god i programmering må du være god i matematikk.
- ◯ Du kan bli god i programmering uten å være god i matematikk.
- ◯ Vet ikke.

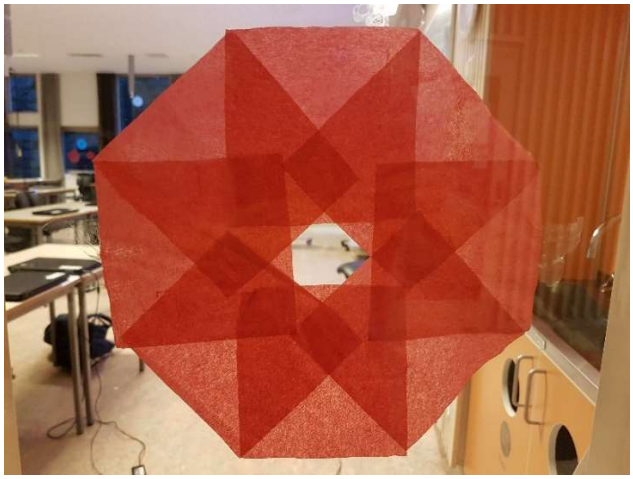
Vedlegg D - Undervisningsøkene

I vedlegg D vil jeg beskrive nærmere innholdet i de 8 undervisningsøktene. For økt 2, 3, 4, 5, 7 og 8 beskrives øktene ved å presentere elevarkene som ble gitt til elevene, mens økt 1 og 6 bare forklares kjapt siden det ikke var noe elevark som ble utdelt i disse to øktene.

Økt 1

Økt 1 var en introduserende bli-kjent økt hvor elevene fikk utdelt ferdig utklippede regulære mangekanter av gjennomskinnelig silkepapir. Disse limte elevene sammen slik som beskrevet i boken «Mathematical Window Patterns» (Gibbs, 1999). Resultatet ble flotte mønstre, og noen av mønstrene kunne minne om julestjerner. Bildene nedenfor er noen av resultatene fra denne økten. I denne økten ble elevene kjent med begrepet regulær mangekant, og de måtte gjøre en algoritme gjentatte ganger for å få ønsket resultat. Tanken var at dette skulle kunne være noe som de kunne dra nytte av til neste økt hvor de skulle lage et eget program med geometriske mønstre.





Økt 2 – Geometriske mønstre

I denne undervisningsøkten skal dere lese og tolke hva et program gjør og deretter lage deres eget program.

Oppgave 1:

Hva gjør programmet som er vist nedenfor? Skriv svaret i loggen. Lag en figur som viser hvordan du tenker at resultatet vil bli.

Test programmet i Scratch og sjekk om programmet oppfører seg slik som du trodde.

Oppgave 2: Lag et program som tegner en T.

Oppgave 3: Lag et program som tegner en regulær trekant, firkant, femkant og/eller sekskant.

Oppgave 4: Lag ditt eget mønster. Det kan være for eksempel en stjerne eller et snøfnugg. Husk at alle snøfnugg er sekskantete. Du kan også ta utgangspunkt i figurene som vi klippet ut sist.

Økt 3 – Geometriske mønstre fortsettelse

Vi fikk litt dårlig tid til å fullføre forrige økt skikkelig, så vi fortsetter med det samme tema i denne økten. Dere som skal ta opptak i WeVideo må sørge for å ha trykket "apply kode" i WeVideo. Dere har fått koden tidligere og kan få den på nytt i dag om det trengs.

Oppgave 1:

Flere av dere lagde flotte mangekanter sist. Det var trekanter, firkanter, femkanter og sekskanter. Likevel skjedde noe rart. Flere planla å lage en trekant, men resultatet ble en sekskant. Hva skjedde? Har dere en forklaring på dette? Skriv svaret i loggen.

Oppgave 2:

Lag to nye mangekanter som dere ikke har laget før.

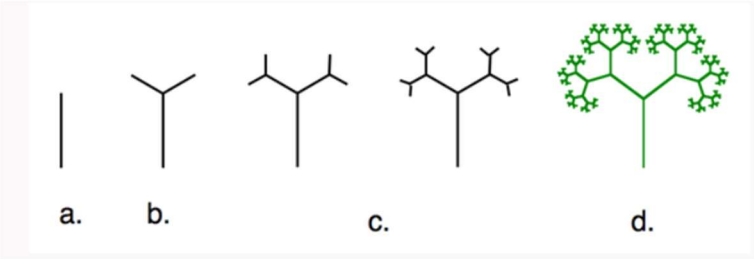
Oppgave 3:

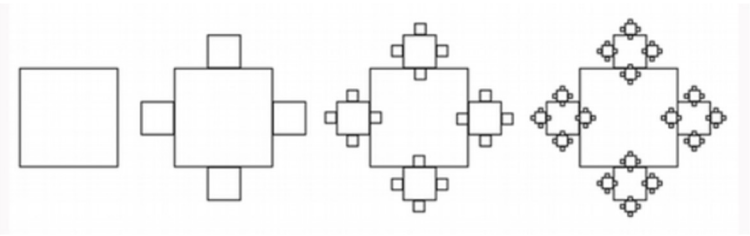
Programmer mønsteret som vi klippet og hang i vinduet.

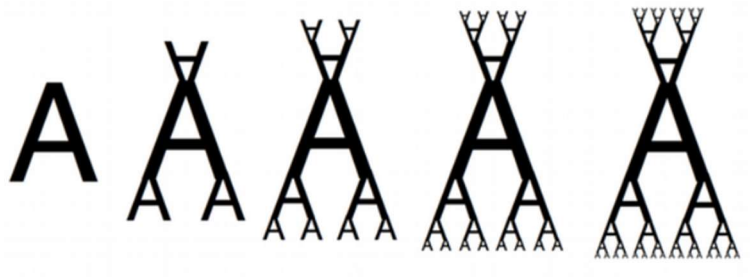
Oppgave 4:

Dette er en ekstra utfordring. Søk på nettet og finn ut hva en fraktal er. Kan du lage et program som tegner en fraktal? Du kan ta utgangspunkt i eksemplene som er lagt ved eller lage dine egne fraktaler. For mer info se lenken og studer bildene som er lagt ved.

https://www.matematikksenteret.no/l%C3%A6ringsressurser/videreg%C3%A5ende/fraktaler





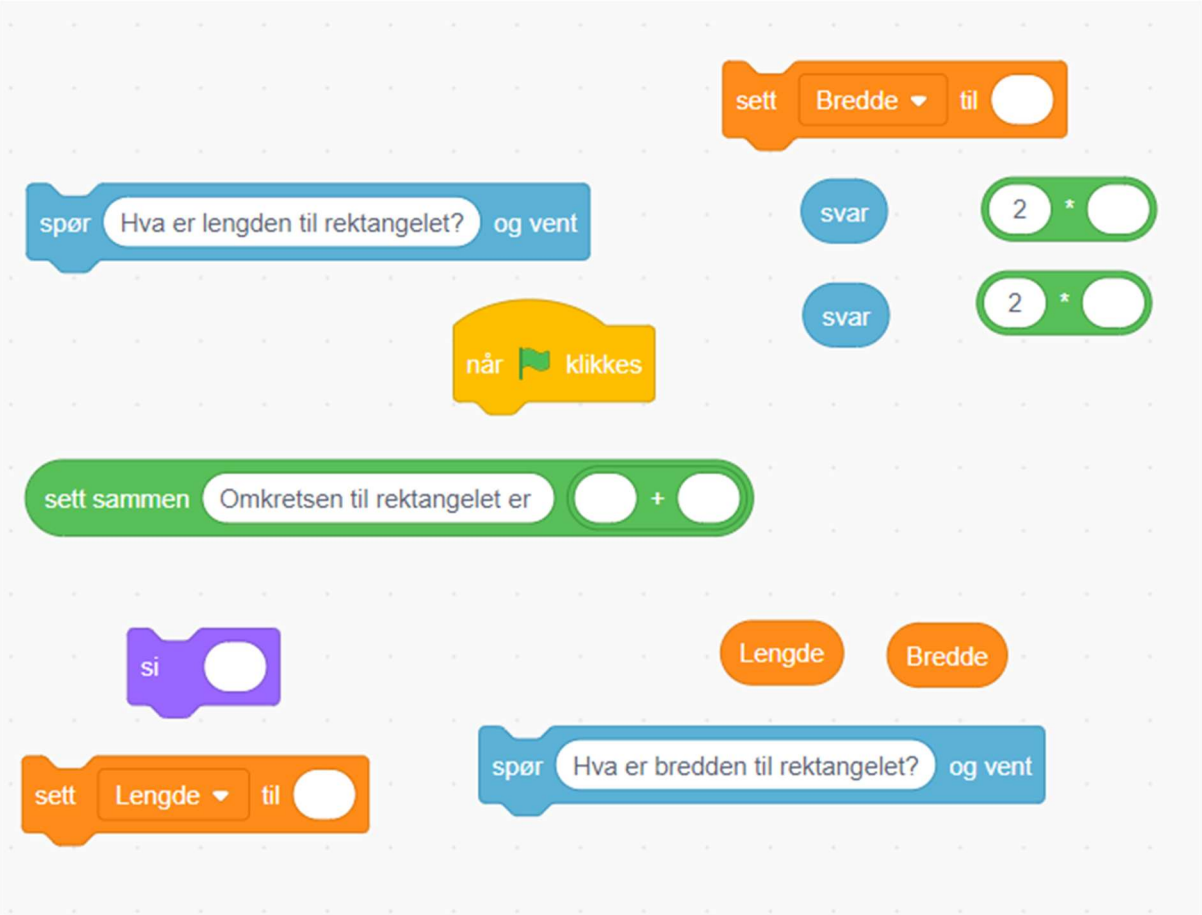


Økt 4: Omkrets og areal av et rektangel

Oppgave 1:
Nedenfor finner du alle kommandoene som skal brukes i et program.
Målet med programmet er å beregne omkretsen til et rektangel.
(Omkrets = hvor langt det er rundt en figur) Først spør programmet etter lengden og bredden til rektangelet. Lengde og bredde er variabler som brukeren av programmet må skrive inn. Deretter regner programmet ut omkretsen av rektangelet.
Pusle sammen blokkene slik du tror at det skal være for å få programmet til å virke slik som beskrevet. Test det deretter i Scratch om programmet fungerer slik det var tenkt.
Hint: Du må først lage en variabel som heter Lengde og en variabel som heter Bredde.

Oppgave 2:
Hvis du fikk til oppgave 1 kan du senere utvide programmet slik at det også beregner arealet av rektangelet.

Oppgave 3:
Velg en annen geometrisk figur og lag et program som beregner omkrets og arealet til figuren. Det er ekstra fint om selve figuren blir tegnet i programmet også.

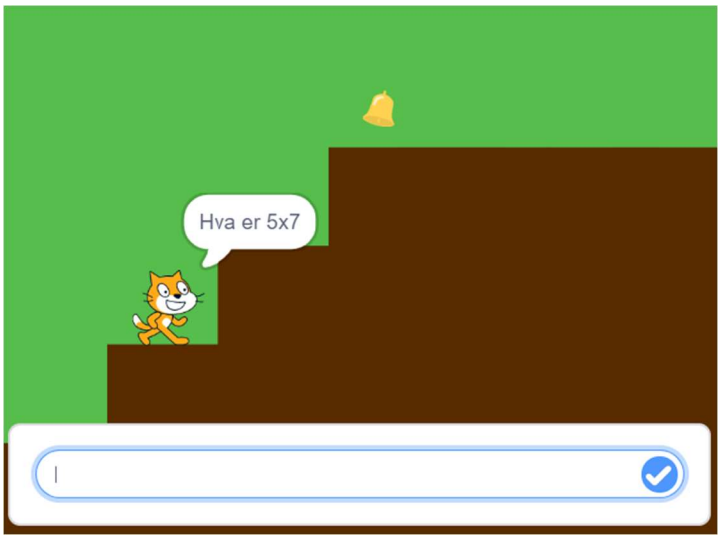


Økt 5 – kunnskapsspill

I dag skal dere lage et kunnskapsspill i Scratch. Jeg vil vise dere et eksempel med en trapp hvor katten stiller et spørsmål for hvert trappetrinn. Dersom spilleren svarer riktig, går katten opp et trinn. Dersom spilleren svarer feil, blir katten stående og stiller et nytt spørsmål. I mitt eksempel er oppgavene som gis tilfeldige regnestykker med multiplikasjon opp til 10x10. Dere kan bruke noe tilsvarende, eller lage helt andre kunnskapsspørsmål. Målet med spillet er å komme i mål på færrest mulige spørsmål.

Oppgave:

Lag et spillebrett for et kunnskapsspill. Det kan være en rett strek, en stige, en trapp, et tre eller noe annet som dere velger selv. Velg en spillefigur (katten kan erstattes med ønsket figur). Lag spørsmål til spilleren og dersom spilleren svarer riktig rykker spillefiguren frem i spillebrettet. Dere kan legge på lyd, poeng, tekst og hva dere måtte ønske selv for å avansere programmet. Til slutt deler dere programmet med klassen i Scratch i galleriet "Økt5– kunnskapsspill". Lykke til.



Økt 6

Økt 6 ble en forlengelse av økt 5 hvor elevene fikk fortsette arbeidet med kunnskapsspillet. På slutten av timen delte de programmene med hverandre i et klassegalleri.

Økt 7 – Tallrekker

I dag skal dere lage et program som lister opp en tallrekke.

En tallrekke kan være for eksempel partallene: 2-4-6-8-10-12-14-16 osv.

Oppgave:

1. Skriv ned en tallrekke som du vil programmere.
2. Innled programmet med at katten sier "Nå skal jeg si de 10 første tallene i tallrekken min"
3. Lag en variabel som heter tall.
4. Programmer slik at katten sier de 10 første tallene i tallrekken.
5. Dere kan også lage en liste slik at alle tallene blir stående i en liste.
6. Hva er tall nummer 100 i tallrekken deres? La katten avslutte programmet med å si hva tall nummer 100 er.
7. Del programmet med klassen i galleriet som heter "Økt 7".
8. Utfordring 1: Lag et program som lister opp de 10 første kvadrattallene.
9. Utfordring 2: Lag et program som lister opp de 10 første trekanttallene.

Økt 8 – Sulten ape

I denne økten får dere tilgang til et ferdig programmert spill. Logg inn i Scratch og finn spillet i klassegalleriet "Sulten ape".

Spillet fungerer fint som det er, men kan med fordel forbedres.

Oppgaver:

1) Forbedre spillet ved å legge på poeng for hver banan som blir spist. Når alle 10 bananene er spist opp har du fått 10 poeng og apen skal si: "Gratulerer"

2) Test at programmet virker som det skal. Test minst to ganger.

3) Det er også med noen giftige insekter i spillet. Hvis apen kommer borti insekter tre ganger er spillet slutt og apen sier: "Game over". Forbedre spillet slik at disse reglene også gjelder.

4) Test at programmet virker som det skal. Test minst to ganger.

5) Utvid spillet slik at når alle bananene er spist går man over på neste bakgrunn. Her lager dere et eget spill hvis dere har tid til det.

Vedlegg E – Spørreskjema for en 8. klasse etter 7 økter

Programmering i 8F

1. Er du jente eller gutt?

   - ◯ Jente
   - ◯ Gutt

2. Hadde du programmert noe før vi startet med prosjektet?

   - ◯ Ja
   - ◯ Nei
   - ◯ Vet ikke

3. Hvis ja; hvilket programmeringsspråk har du brukt tidligere?

   _____

4. Hvordan synes du nivået har vært i dette prosjektet?

   - ◯ For vanskelig
   - ◯ Passelig
   - ◯ For lett
   - ◯ Varierende

5. Var du med på gruppen som tok opptak?

   - ◯ Ja
   - ◯ Nei
   - ◯ Noen ganger

6. Hva var det gøyeste i timene?

7. Hva kunne vært gjort på en bedre måte i timene? Kom gjerne med forslag.

8. Liker du programmering mer eller mindre nå enn før prosjektet.

- ◯ Jeg liker programmering mindre nå.
- ◯ Ingen endring.
- ◯ Jeg liker programmering bedre nå.

9. Hvilken økt likte du best?

- ◯ Klippe stjerner.
- ◯ Programmere mønstre (tegne trekant, femkant, snøfnugg, stjerner)
- ◯ Omkrets og areal av rektangel.
- ◯ Kunnskapsspill.
- ◯ Tallrekker.

10. Hvorfor likte du den økten best?

11. Hvilken økt lærte du mest av?

- Klippe stjerner.
- Programmere mønstre (tegne trekant, femkant, snøfnugg, stjerner)
- Omkrets og areal av rektangel.
- Kunnskapsspill.
- Tallrekker.

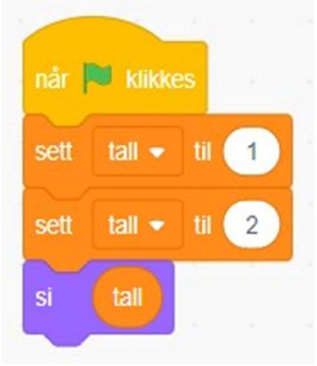
12. Hva lærte du i denne økten?

13. Har du andre kommentarer til prosjektet?

Vedlegg F – Avsluttende kartleggingsprøve

Programmering med variabler

1. Hva sier Katt når man trykker på flagget?

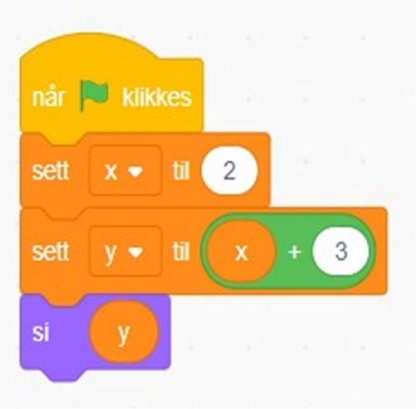


2. Hva sier Katt når man trykker på flagget?

3. Hva sier Katt når man trykker på flagget?

4. Hva sier Katt når man trykker på flagget?

5. Hva sier Katt når man trykker på flagget?



6. Hva sier Katt når man trykker på flagget?

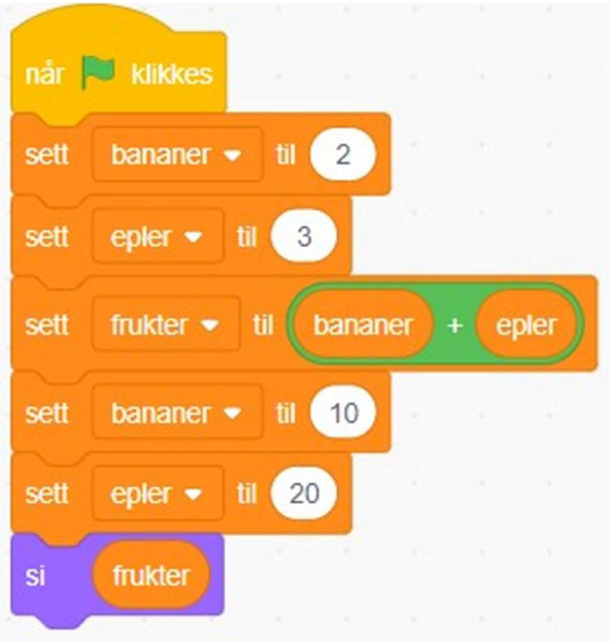
7. Hva sier Katt når man trykker på flagget?

8. Hva sier Katt når man trykker på flagget?

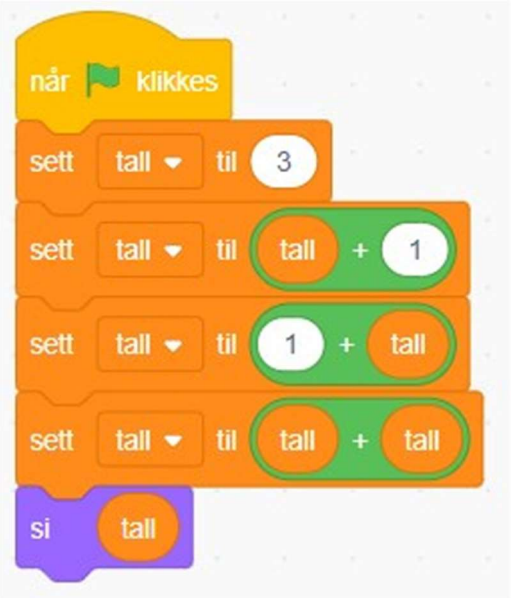
9. Hva sier Katt når man trykker på flagget?

10. Hva sier Katt når man trykker på flagget?

11. Hva sier Katt når man trykker på flagget?



12. Hva sier Katt når man trykker på flagget?



13. Hva sier Katt når man trykker på flagget?