Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Odin Løvås

Anvendelse av programmering i kjemi på videregående skole

Oppgaver, algoritmer og løsningsforslag som støtter kjemilærere ved utvikling av programmeringsbaserte undervisningsopplegg

Bacheloroppgave i Informasjonsbehandling
Veileder: Helge Hafting
Mai 2022

NTNU
Kunnskap for en bedre verden

Odin Løvås

Anvendelse av programmering i kjemi på videregående skole

Oppgaver, algoritmer og løsningsforslag som støtter kjemilærere ved utvikling av programmeringsbaserte undervisningsopplegg

Bacheloroppgave i Informasjonsbehandling
Veileder: Helge Hafting
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Sammendrag

I forbindelse med fagfornyelsen har det blitt utgitt nye læreplaner til alle fag i den norske skolen. Disse læreplanene legger vekt på at elevene skal lære det viktigste i faget, i tillegg til å skaffe nødvendig generell og faglig kompetanse til å gå videre inn i studier eller arbeidslivet. Læreplanene blir laget ut ifra sentrale kjerneelementer, og setter blant annet fokus på begrepet dybdelæring.

En av endringene til læreplanen i flere fag, spesielt da realfag, er at programmering skal brukes i undervisningen. Dette er et mål for at elevene skal utvikle nødvendige digitale ferdigheter, og at elevene skal utforske faget på nye måter, for å oppnå økt dybdeforståelse.

Med denne endringen har det oppstått en utfordring med at lærerne ikke har gode nok forkunnskaper innen programmering for å effektivt implementere programmering i undervisninger. For å møte dette problemet har flere institusjoner utviklet ressurser for å øke lærernes kompetanse innen programmering, men disse ressursene vil først ha effekt over lengre tid ettersom å lære programmering til et tilstrekkelig nivå tar tid.

Det er derfor behov for mer umiddelbare støttende ressurser som lærere kan bruke for å utvikle undervisningsopplegg, og dette prosjektet har som mål om å utvikle en slik ressurs for kjemifaget på videregående skole.

Prosjektet er utført med bruk av forskningsmetoden «Design Science», hvor det da er utført et forskningsarbeid som endte i et produkt.

Produktet utviklet igjennom prosjektet er et oppgavehefte med 12 oppgaver relatert til kjemifaget på videregående, både kjemi 1 og kjemi 2. Oppgavene presenteres med en problemstilling, en løsning, algoritmer man kan følge og et eksempel på programkode med resultat.

Det er forsøkt å samle inn oppgaver som samsvarer med de nye læreplanene, og som skal kunne støtte lærere i utvikling av undervisningsopplegg. Det er også forsøkt å samle oppgaver som er engasjerende, tillater kreativitet fra elevene og leder til økt dybdelæring.

Abstract

In accordance with the renewal of subject in the Norwegian school, has the Ministry of Education developed a new curriculum for all subjects in the Norwegian school. This new curriculum focuses on that the students learn the most important general and subject-related skills in order to succeed in future studies and work. The new curriculum is based on central core elements of the subject, and focus on concepts like in depth learning.

One of the changes to the curriculum in several subjects, especially STEM subjects, is that programming shall be used as a tool in the education. This is a change based on the students need to develop necessary digital skills, in addition to allowing students to explore the subject in new ways, which allows for in depth learning.

This change led to a challenge given that the majority of teachers does not have good enough knowledge within the field of programming to effectively implement programming into the education. In order to solve this problem has several institutions developed resources in order to aid teachers in learning programming, but these resources will take effect over time as learning programming to a satisfactory level takes time.

It is therefore a need for more immediate tools to aid teachers in adapting their teaching to use programming, and this project has a goal of developing such a resource for teachers in chemistry at a high school level.

The project is performed using “Design Science Research”, in order to produce a product.

The end product of the project is a collection of chemistry related exercises that use programming as a way to solve them, relevant to both the subjects “Kjemi 1” (Chemistry 1) and “Kjemi 2” (Chemistry 2). The exercises includes a problem, a solution, algorithms to follow and an example of a program including an result.

The goal of the exercises is that they follow the new curriculum and can aid teachers in their task to adapt programming to their teaching. It is attempted to choose tasks that are engaging, allow for creativity by students, and leads to increased in-depth learning.

Forord

Når jeg selv gikk på grunnskolen og videregående skole syntes jeg alltid at det var for lite fokus på data og programmering. Jeg var selv interessert i temaet, men det var ingen undervisning i verken data eller programmering som mente jeg måtte lære alt på egen hånd. Jeg kunne i flere tilfeller se for meg at data og programmering kunne hjelpe utforske fagene på en artigere og mer innholdsrik måte.

Dermed synes jeg at denne oppgaven, i tillegg til andre anvendelser av programmering, passet bra for meg og kunne være et motiverende tema å jobbe med. Jeg synes det er viktig at elever får tidlige erfaringer i programmering, og dette er da relevant for kjemi hvor jeg har sett at programmeringsspråk som MatLab og Python ofte blir bruk ved videre utdyping i faget.

Siden det er flere år siden jeg selv gikk på videregående skole og tok kjemi, måtte en god del av arbeidet gå til utdyping i faget, i tillegg til forskning relatert til programmering i kjemi og skolen.

Oppgaven er utført for NTNU-AIT og veileder til prosjektet har vært Helge Hafting.

Innhold

Tabeller

Programkoder

Forkortelser og symboler

CCSE: Center for Computing in Science Education

IS: Informasjonssystemer (Engelsk: Information Systems)

NOU: Norges Offentlige Utredninger.

NTNU: Norges Teknisk-Naturvitenskapelige Universitet.

UDIR: Utdanningsdirektoratet.

UiO: Universitetet i Oslo

Begreper

Algoritme: «En algoritme er en presis beskrivelse av en serie operasjoner som skal utføres for å oppnå et visst resultat» (Haraldsrud, 2021, s. 8)

Algoritmisk tenking: «Algoritmisk tenkning er en problemløsningsmetode. Algoritmisk tenkning innebærer å tilnærme seg problemer på en systematisk måte, både når vi formulerer hva det er vi ønsker å løse og når vi foreslår mulige løsninger.» (Utdanningsdirektoratet, 2019, b)

Dybdelæring: «Det å gradvis utvikle kunnskap og varig forståelse av begreper, metoder og sammenhenger i fag og mellom fagområder. Det innebærer at vi reflekterer over egen læring og bruker det vi har lært på ulike måter i kjente og ukjente situasjoner, alene eller sammen med andre.» (Utdanningsdirektoratet, 2019, a).

Fagfornyelsen: «Fagfornyelsen er navnet på prosessen med å utvikle og innføre nye læreplaner i Kunnskapsløftet. Nye læreplaner ble tatt i bruk fra 2020» (Utdanningsdirektoratet, udatert, c)

Kjerneelement: «Kjerneelementer er det viktigste og mest sentrale elevene skal lære i hvert fag. Det kan være kunnskapsområder, metoder, begreper, tenkemåter og uttrykksformer» (Kunnskapsdepartementet, 2018)

Modellering: «Modellering er en prosess for å finne en forenklet representasjon av et fenomen i virkeligheten, altså en modell» (Haraldsrud, 2021, s. 9)

Programmering: «Programmering går ut på å sette opp en serie instrukser som styrer maskinen og som avgjør hvordan den skal reagere på inndata, inntastinger, musebevegelser og annet» (Rossen, 2019)

Pseudokode: «Pseudokode er en uformell beskrivelse av et dataprogram eller en algoritme» (Dvergsdal, 2022)

1 Introduksjon

1.1 Bakgrunn for oppgaven

Fra høsten 2020 ble det begynt å innføre nye læreplaner i den norske skolen. Begrunnelsen for disse nye læreplanene ble basert på tilbakemeldinger fra rektorer og lærere om at de tidligere læreplanene dekket for mange temaer og at lærere dermed ikke fikk til å gå inn i dybden i fagene. (Kunnskapsdepartementet, 2018).

Det var bekymringer for at de gamle læreplanene ledet til mye overfladisk læring, som førte til at elevene ikke får den kompetansen de trenger for å gå inn i videre studer og arbeidslivet. I forbindelse med dette ble det derfor utført en fornyelse av læreplanene kalt fagfornyelsen. (Kunnskapsdepartementet, 2018).

1.1.1 Fagfornyelsen

Fagfornyelsen er en fornyelse av læreplanene, hvor det skal legges fokus på at elevene skal kunne gå inn i dybden i fagene. Elevene skal kunne få tid til å lære seg det viktigste delene av faget og utforske disse delene slik at de sitter igjen med et godt grunnlag for videre studier og arbeidslivet. (Kunnskapsdepartementet, 2018).

I første omgang, fra høsten 2020, skal det innføres nye læreplaner for alle fagene på grunnskolen og de gjennomgående fagene på videregående skole. Videre skal det innføres nye læreplaner for de studieforberedende programfagene og fordypningen i yrkesfag for andre år på videregående skole i høsten 2021, og til slutt skal det innføres nye læreplaner for de studieforberedende programfagene for tredje år på videregående skole i høsten 2022. (Utdanningsdirektoratet, 2022).

Målet med fagfornyelsen er at læreplanene skal legge vekt på mer dybdelæring og flere fag skal rettes mot en mer praktisk tilnærming av fagstoffet. I forbindelse med dette blir fagene endret slik at elevene får tid til å utforske fagstoff i større grad. For å klargjøre hva som skal være i fokus igjennom undervisningsopplegget ble det utviklet konkrete kjerneelementer som læreplanene skal utvikles ut ifra (Kunnskapsdepartementet, 2018).

Disse kjerneelementene skal være det viktigste og mest sentrale fagstoffet i fagene, som skal lede til bedre forståelse for fagene i sin helhet. De skal være konkrete og dekke det aller viktigste som elevene skal lære seg i fagene. De inkluderer ikke alt innholdet i faget,

men skal fungere som en pekepinn hva som er viktigst å få med i undervisningsopplegg. (Kunnskapsdepartementet, 2018).

I tillegg til de faglige kjerneelementene, blir det i fagfornyelsen også utviklet bestemte grunnleggende ferdigheter som alle fag har et særskilt ansvar for at eleven skal oppnå. Disse grunnleggende ferdighetene er (Kunnskapsdepartementet, 2018):

1. Muntlige ferdigheter
2. Å kunne skrive
3. Å kunne lese
4. Å kunne regne
5. Digitale ferdigheter

I forbindelse med digitale ferdigheter, og som et kompetansemål ved flere av fagene, legger de nye læreplanene vekt på at programmering skal tas i bruk ved undervisningen i relevante fag.

1.1.2 Programmering i skolen

En viktig del av skolen er grunnleggende ferdigheter innen hvordan bruke digitale ressurser. I nyere tid har det blitt aktuelt å vurdere om ferdigheter innen programmering også skal være en del av disse digitale ressursene, og det har blitt en større bevegelse som ønsker å innføre programmering og IT kunnskaper inn i den norske skolen. (Selvik, 2016)

Det å inkludere programmering som en del av undervisningen i skolen er noe flere andre land har begynt med, og det er blant annet utviklet tre forskjellige strategier for å knytte programmering inn i skolen (Selvik, 2016).

Den første måten innebærer å undervise i programmering som et eget IKT fag, slik at man får fokus på å lære programmering direkte. Dette er en metode som blant annet England har begynt med (Selvik, 2016).

Den andre måten innebærer å bruke programmering som et støttende verktøy ved undervisning av relevante fag, og dermed indirekte lærer programmering igjennom relevante aktiviteter. De relevante fagene i denne sammenheng kan være matematikk, kunst og håndverk, naturfag eller andre fag, som regel realfag. Dette er en metode som blant annet Finland bruker (Selvik, 2016).

Den tredje metoden innebærer å sette programmering som et tverrfaglig tema, slik at alle fagene har som mål å øke kompetansen innen programmering. Dette er en metode som blant annet Estland har tatt i bruk (Selvik, 2016).

Ved innføring av fagfornyelsen har den norske skolen begynt å bruke programmering som en del av eksisterende fag, blant annet i realfagene.

1.1.3 Valgfagene kjemi 1 og kjemi 2

Fagfornyelsen og de nye læreplanene innført i 2020 gjelder alle fag på grunnskolen, og obligatoriske fag på videregående skole. Videre skal programfagene i studiespesialisering og yrkesfaglige fordypningsfag få nye læreplaner fra høsten 2021 og høsten 2022. (Utdanningsdirektoratet, 2022).

Kjemi 1 og 2 er studieforbedrende programfag som elevene kan velge ved andre og tredje år i videregående skole. Disse fagene får dermed nye læreplaner fra høsten 2021 og høsten 2022, henholdsvis (Utdanningsdirektoratet, 2022). Kjemifagene bygger videre ved flere av kunnskapsområdene i de obligatoriske realfagene naturfag og matematikk. Fagene handler om en blanding av praktiske aktiviteter, blant annet ved utføring av forsøk, og teoretiske aktiviteter, blant annet ved analysering og bearbeid med resultater fra forsøk (Utdanningsdirektoratet, udatert, a).

I kjemifagene har det blitt definert følgende kjerneelementer (Utdanningsdirektoratet, udatert, a):

- Praksiser og tenkemåter i kjemi
- Kjemiske bindinger og strukturer
- Kjemiske reaksjoner
- Anvendt kjemi

1.1.4 Kjemifagene og digitale ferdigheter

Som en del av de grunnleggende ferdighetene definert av kunnskapsløftet, skal kjemi brukes til å forbedre elevenes kompetanse innen digitale ferdigheter. Ifølge læreplanen skal kjemifagene oppnå dette ved å bruke digitale ressurser til å utforske, illustrere og utdype kjemifaglig stoff. I tillegg innebærer dette å samle inn og bearbeide data, gjøre utregninger,

lage visualiseringer og presentere data ved bruk av digitale ressurser. (Utdanningsdirektoratet, udatert, a).

I forbindelse med dette vil det være naturlig å ta i bruk programmering og databeregninger i undervisningen, ettersom programmering og databeregninger kan bidra til å nå flere av disse målene.

1.2 Lærerkompetanse i programmering

Endringene som kom med fagfornyelsen har ført til at store deler av innholdet i fagene blir endret og at fagene skal undervisers på en ny måte, slik at elevene skal lære bedre (Kunnskapsdepartementet, 2018). Læreplanene er laget på en slik måte at lærerne skal kunne fokusere på de viktigste temaene innen hvert fag, men innføringen av de nye læreplanene har kommet med nye utfordringer for lærerne.

En av disse utfordringene er mangel på nødvendige forkunnskaper hos lærerne, slik at de ikke får undervist på en effektiv og god måte. Dette er blant annet et problem for lærerne når de skal nå målet om å undervise fagene ved bruk av programmering. Da de nye læreplanene ble innført, hadde majoriteten av lærere lite forkunnskaper innen programmering (Stenlund, E., 2021).

For å møte denne utfordringen må lærere bruke tid på å lære seg programmering, for så å tilpasse undervisningen slik at programmering blir bruk i undervisningsopplegget. Dette er en tidskrevende prosess som lærerne må finne tid til, i tillegg til å utføre lærerarbeidet. I en casestudie utført av Erling Stenlund blir det undersøkt i hvor stor grad dette er en utfordring, hvor flere lærere ble intervjuet om deres meninger og erfaringer rundt det å bruke programmering i undervisningen (Stenlund, E., 2021). Ifølge casestudien synes flere lærere at programmering er spennende og interessant, og ser verdien i ferdigheten, men at de også var bekymret for at programmering blir tidskrevende og forutsetter mye selvstendig arbeid. Studien viste også til at flere lærere med negativ innstilling til programmering endret mening til en mer positiv mening om programmering etter gjennomføring av kurs. Dette viser til hvor viktig det er at lærere har lett tilgang til nødvendig støtte i forbindelse med å lære programmering.

I et forsøk på å lette byrden for lærerne har flere institusjoner implementert tiltak som skal hjelpe lærere i å skaffe nødvendige forkunnskaper. UDIR har blant annet innført

internetbaserte kompetansepakker som lærere kan benytte seg av (CCSE, 2021), og flere organisasjoner ønsker å bidra i undervisning av programmering, som Lær Kidsa Koding (Lær Kidsa Koding, udatert). I tillegg til dette tilbyr flere høyskoler og universiteter etterutdanning som lærere kan melde seg til, for å øke egen kompetanse innen programmering.

På lengre sikt vil nok forkunnskapene til lærere forbedres innen programmering, blant annet på grunn av disse tilbudene og et større fokus på programmering som en nødvendig kompetanse ved utdanning av lærere. Men effekten av dette vil først komme fremover i tid, mens læreplanene har allerede begynt å bli innført fra høsten 2020. Lærere synes derfor det er et problem med å skaffe nok støtte for tilrettelegging av undervisning som de kan benytte seg av umiddelbart og bruke som referanse for videre utvikling av egne undervisningsopplegg som benytter seg av programmering. Blant annet klager flere lærere på mangel av støttelitteratur og fagdidaktikk (Stenlund, E., 2021).

1.3 Problemstilling

Som nevnt i tidligere avsnitt er det, etter lærernes mening, en mangel på støtte i å tilpasse programmering til undervisningen. Det er derfor behov for materialer som kan støtte lærerne i deres undervisning av programmering i fagene.

I relasjon til dette ble det utført et bachelor prosjekt som skulle utforske bruk av programmering i matematiske fag og produsere et oppgavehefte som kunne støtte lærere i å bruke programmering i undervisningen av matematiske fag (Mohammadi, S. 2021). Videre har NTNU utlyst flere nye bacheloroppgaver som har som mål å utforske bruk av programmering i andre fag, og forsøke å utvikle oppgavehefter som kan benyttes av lærere i forbindelse med undervisning av programmering i disse fagene.

I denne bacheloroppgaven forsøkes det å

1. utforske bruk av programmering i kjemifagene, og
2. utvikle et oppgavehefte som samsvarer med de nye læreplanene og støtter kjemilærere i undervisning av programmering i kjemi.

Hensikten med oppgaven er å finne ut hvilke oppgaver innen kjemi på videregående nivå som egner seg best for å øke dybdeforståelse, engasjement og kreativitet hos elever, og anbefalte fremgangsmåter for implementasjon av disse.

Den første delen av oppgaven handler om å utforske programmering i kjemifagene og går ut på å utforske hvordan man kan ta i bruk programmering i kjemifagene for å øke dybdeforståelsen til elevene.

Den andre delen av oppgaven handler om å ta i bruk resultatet fra den første delen, for å utvikle et oppgavehefte med programmerings-relaterte kjemioppgaver, som kan fungere som en støttende ressurs for kjemilærere ved anvendelse av programmering i kjemifagene.

1.4 Rapportens struktur

Rapporten inneholder følgende kapitler:

- **Introduksjon:** I dette kapittelet beskrives behovet og grunnlaget for oppgaven. Her blir det beskrevet nødvendig relevans for oppgaven, i tillegg til en beskrivelse av oppgaven.
- **Teori:** I dette kapittelet blir det gjennomgått sentrale begreper relatert til oppgaven.
- **Metode:** I dette kapittelet blir det gjennomgått hvilket programmeringsspråk som er tatt i bruk for oppgavene, i tillegg til en begrunnelse på hvordan oppgavene er valgt ut.
- **Resultater:** I dette kapittelet blir resultatene fra forskningsdelen gjennomgått, og det blir vist fram noen eksempler fra oppgaveheftet.
- **Diskusjon:** I dette kapittelet blir resultatene diskutert og begrunnet opp problemstillingen og teoridelen.
- **Konklusjon:** I dette kapittelet blir det konkludert hva som har blitt lært igjennom prosjektet, og det blir reflektert rundt muligheter for videre arbeid.

2 Teori

Fagfornyelsen setter fokus på at elevene skal lære de viktigste temaene i fagene, og at elevene skal kunne utvikle sine grunnleggende ferdigheter igjennom fagene. Dette fokuset er satt med et formål om at elevene skal kunne være best forberedt til videre studier og arbeidslivet.

I forbindelse med dette bruker fagfornyelsen begrepet dybdelæring som ett grunnleggende element i kjerneelementene. Læreplanene er laget på en slik måte at elevene skal kunne fordype seg innen fagenes viktigste temaer.

Som en del av de grunnleggende ferdighetene skal elevene også øke sin kompetanse innen digitale ferdigheter. I flere fag involverer dette da programmering og databeregninger, som vil være viktige ferdigheter å kunne bruke i videre studier og arbeidslivet.

Dette kapittelet tar for seg:

- Hva er dybdelæring og hvorfor er dybdelæring et viktig begrep for undervisning?
- Hvilke kjerneelementer er blitt definert i kjemi etter fagfornyelsen?
- Hva er programmering, hvorfor burde det bli brukt i undervisning i skolefag og hvordan kan man implementere programmering i undervisningen.

2.1 Dybdelæring

Fagfornyelsen er en prosess som ble påbegynt basert på tilbakemeldinger fra lærere og rektorer om at det var for mange temaer som lærere måtte igjennom og at det ikke ble nok tid til å gå inn i dybden i fagene (Kunnskapsdepartementet, 2018).

Dette problemet ble utforsket igjennom et arbeid som undersøkte hva som kreves av kompetanse i fremtidens skole. I forbindelse med dette arbeidet ble det utarbeidet to utredninger, av Ludviksen-utvalget: «Elevenes læring i fremtidens skole – Et kunnskapsgrunnlag» (NOU, 2014:7) og «Fremtidens skole – Fornyelse av fag og kompetanser» (NOU, 2015:8).

I NOU 2014:7 blir det påpekt at læringsforsking viser til at dybdelæring har betydning for elevenes utvikling i og på tvers av fag (NOU, 2014:7, s.10). Det ble også poengtert at mange av skolefagene hadde et for bredt omfang, som gjorde det vanskelig for lærere å ivareta dybdelæring i fagene (NOU, 2014:7, s. 78).

I NOU 2015:8 blir det konkludert at dybdelæring er tett knyttet til kompetanseoppnåelse og at dybdelæring er like viktig i alle fag (NOU, 2015:8, s. 10). Det ble også konkludert at det var behov for en fornyelse av fagene, med dybdelæring i sentrum (NOU, 2015:8, s.40)

Begge utredningene har altså presisert viktigheten for dybdelæring i skolen og hvor viktig det er å fornye fagene for å ta hensyn til dette. Disse utredningene ble en viktig del av grunnlaget for fagfornyelsen og er en av grunnene til at fagfornyelsen setter fokus på dybdelæring.

UDIR definerer dybdelæring som følgende:

«Vi definerer dybdelæring som det å gradvis utvikle kunnskap og varig forståelse av begreper, metoder og sammenhenger i fag og mellom fagområder. Det innebærer at vi reflekterer over egen læring og bruker det vi har lært på ulike måter i kjente og ukjente situasjoner, alene eller sammen med andre.» (Utdanningsdirektoratet, 2019, a).

2.2 Kjerneelementer i kjemi

Etter fagfornyelsen har kjemifagene fått fire sentrale kjerneelementer:

- **Praksiser og tenkemåter i kjemi**

«Kjerneelementet praksiser og tenkemåter i kjemi handler om hvordan naturvitenskapelige hypoteser, teorier, metoder og modeller innenfor fagfeltet utvikles og brukes, og hvordan disse er knyttet til eksperimenter og forsøk. Det handler også om praktisk laboratoriearbeid og utforskende aktiviteter, metodevalg og bearbeiding av innsamlede data. Fagets representasjonsformer, symboler og termer inngår også i kjerneelementet.» (Utdanningsdirektoratet, udatert a)

- **Kjemiske bindinger og strukturer**

«Kjerneelementet kjemiske bindinger og strukturer handler om krefter mellom partikler og hvordan disse har betydning for stoffers oppbygning, sammensetning og egenskaper. Det handler også om kriterier for klassifisering av stoffer og hvordan periodesystemet brukes til å se sammenhenger og trender i egenskaper hos grunnstoffene.» (Utdanningsdirektoratet, udatert a)

- **Kjemiske reaksjoner**

«Kjerneelementet kjemiske reaksjoner handler om hvordan og hvorfor stoffer reagerer. Det inkluderer reaksjonstyper, termodynamikk og kinetikk. Observasjoner og ulike typer data er, sammen med balanserte reaksjonsligninger, utgangspunkt for klassifisering, vurdering og beregning.» (Utdanningsdirektoratet, udatert a)

- **Anvendt kjemi**

«Kjerneelementet anvendt kjemi handler om å bruke kompetanse i kjemi til å forstå virkningen av kjemiske stoffer og prosesser på individ og samfunn. Det handler også om kjemiske analyser og om nytteverdien av og ressursbruk knyttet til ulike stoffer, materialer, prosesser og kjemisk teknologi. Anvendt kjemi handler også om å bruke kjemiske prinsipper og teorier for kritisk å vurdere informasjon og få innsikt i betydningen kjemi har for menneskers velferd. Vurderinger knyttet til helse, miljø og sikkerhet er en del av dette kjerneelementet.» (Utdanningsdirektoratet, udatert a)

2.2.1 Digitale ferdigheter

I tillegg til kjerneelementer skal kjemi også utdype elevenes grunnleggende ferdigheter. Disse innebærer (Utdanningsdirektoratet, udatert a):

- Muntlige ferdigheter
- Å kunne skrive
- Å kunne lese
- Å kunne regne
- Digitale ferdigheter

Som en del av fagfornyelsen skal programmering være en del av de digitale ferdighetene som elevene skal skaffe i relevante fag. Målet om å oppnå digitale ferdigheter i kjemi blir i læreplanen definert som:

«Digitale ferdigheter i kjemi innebærer å bruke digitale ressurser til å utforske, illustrere og utdype kjemifaglig stoff. Det innebærer også å bruke digitale ressurser til å samle inn og bearbeide data, å gjøre beregninger og lage visualiseringer og å presentere resultater fra eget og andres arbeid. Videre innebærer digitale ferdigheter å bruke modellering for å utforske kjemiske fenomener.» (Utdanningsdirektoratet, udatert a).

Programmering er en digital ressurs som kan oppnå flere av disse punktene, og derfor kan kjemi ansees som et av de fagene hvor det vil være relevant å bruke programmering i undervisningen.

2.3 Programmering

Programmering er arbeidet rundt å utarbeide et dataprogram som bestemmer hvordan en datamaskin eller et annet elektronisk produkt skal oppføre seg. Programmet er en serie med instrukser som bestemmer hvordan maskinen skal reagere på ulike former for inndata (Rossen, 2019).

Programmene skrives ved bruk av ulike programmeringsspråk som oversetter instruksene slik at maskinen kan forstå de. Disse programmeringsspråkene kan være i form av direkte maskinkode, et lavnivåspråk, høynivåspråk eller avanserte utviklingsverktøy. Ved maskinkode og lavnivåspråk vil instruksene skrives på en måte som ligner på hvordan maskinen leser instruksene, men som er vanskelig for mennesker å lese. Ved høynivå språk skrives koden på er mer lesbar måte for mennesker, mens ved bruk av utviklingsverktøy blir mye av skrivingen gjort av verktøyet (Rossen, 2019).

Måten man utvikler programvare handler om å først analysere oppgaven man ønsker gjennomført. Så bryter man ned oppgaven i trinn og utvikler en logikk for å løse disse trinnene. Deretter skriver man logikken, i form av instrukser, inn i programmet. Ofte opplever man at programmet ikke fungerer som det skal i første omgang, da må man prøve å finne feilen ved instruksene og rette på disse. Ofte når man produserer programmer kommer man fram til løsninger som kan brukes igjen i senere anledninger, som leder til at man sparer tid ved utvikling av fremtidige programmer (Rossen, 2019).

Programmering er altså mer enn kun prosessen om å skrive programinstrukser, det inkluderer også prosessen om å komme fram til hvordan man utvikler disse instruksene, som kan assosieres med utforsking, problemløsning og feilsøking. (Selvik, 2016)

2.4 Programmering som en undervisningsstrategi

Som nevnt i forrige delkapittel kan programmering assosieres med mer enn kun det å skrive programkode, men også med utforskning, problemløsning og feilsøking. Med tanke på dette

er det utviklet flere lærestrategier som tar i bruk konsepter innen programmering for å utforske og løse problemer.

2.4.1 Algoritmisk tenking

Algoritmisk tankegang er en problemløsningsmetode som går ut på å bryte ned store, komplekse problemer inn i mindre delproblemer. Problemløsningsmetoden baserer seg på måten man utvikler programkode ved at man organiserer og analyserer data på en logisk måte og utvikler fremgangsmåter, også kalt algoritmer, som man bruker til å løse mer komplekse problemer. Metoden innebærer også å generalisere løsninger på problemer slik at man effektivt kan utnytte samme eller lignende løsninger for å løse lignende problemer (Selvik, 2016).

Algoritmisk tenking handler altså om å skaffe seg en evne til å systematisk gå fram i et problem, steg for steg, og å skaffe nok teknologisk kompetanse til å forstå hva man må løse selv og hva en datamaskin kan løse i stedet (Utdanningsdirektoratet, 2019, b).

2.4.1.1 Algoritme

«En algoritme er en presis beskrivelse av en serie operasjoner som skal utføres for å oppnå et visst resultat» (Haraldsrud, 2021).

Som tidligere forklart er en viktig del av den algoritmiske tankegangen utviklingen av fremgangsmåter, også kalt algoritmer. Algoritmer forklarer hvert enkelt steg på vei mot løsningen, slik at man får en systematisk oversikt over hvordan man løser problemet (Utdanningsdirektoratet, 2019, b).

I forbindelse med programmering er det ofte mulig å gjenbruke algoritmer flere ganger i samme eller forskjellige programmer. Det er også mulig å importere ferdiglagde algoritmer. I forbindelse med læring er det dermed ikke anbefalt å importere algoritmer, som begrunnes med at å lage egne algoritmer gir bedre innsikt i hvordan programmet fungerer og tillater mer endring og utforsking av programkoden (Haraldsrud, 2021).

2.4.2 Pseudokode

Pseudokode er en strategi hvor man prøver å beskrive fremgangsmetoden for å løse et problem på en måte som man kan realisere i mange programmeringsspråk, men som også er enkelt å forstå. Det er ingen fastsatte definisjoner på hvordan man lager pseudokode, så lenge det oppfyller disse to punktene (Dvergsdal, 2022).

Å bruke pseudokode i undervisning kan være en god måte å introdusere elever til programmering. Ved å først produsere en pseudokode, så videre utvikle programkode fra dette kan elevene forbedre sine problemløsningsferdigheter innen algoritmisk tenking, og hjelpe de lettere forså hvordan instruksene i dataprogrammene fungerer (Olsen, 2005).

2.5 Modellering

«Modellering er en prosess for å finne en forenklet representasjon av et fenomen i virkeligheten, altså en modell» (Haraldsrud, 2021).

Modellering handler både om å lage modeller og å forstå de. I kjemi er dette et relevant tema ettersom en viktig del av faget handler om å bruke modeller for å forklare sammenhenger mellom kjemiske stoffer. I tillegg er det viktig å forstå at modellene kun er forenklinger og tolke modellene basert på dette (Haraldsrud, 2021).

2.6 Programmering i kjemi

Programmering kan være en nyttig digital ressurs for å forstå og utforske kjemi. Programmering kan blant annet brukes til å utføre beregninger, lage modeller og visualisere kjemisk fagstoff på nye måter som kan gi bedre innsikt i kjemifaget (Haraldsrud, 2021).

Programmering kan bidra til blant annet: (Haraldsrud, 2021)

1. Datahåndtering
2. Utforsking av kjemiske sammenhenger og stoffers egenskaper
3. Modellering og numeriske eksperimenter
4. Forståelse av matematikk og matematiske metoder i kjemi

I kjemien blir programmering dermed ofte brukt som er verktøy for å forenkle store deler av arbeidet i faget, slik at man lettere kan forstå og jobbe med kjemifaglige temaer (Haraldsrud, 2021).

3 Teknologi og metode

I dette kapittelet blir det begrunnet valg av teknologi og metode for prosjektet. Den første delen, teknologi, tar for seg hvilken teknologi blir valgt under prosjektet. Den andre delen, metode, beskriver hvilken metode som ble brukt for å skaffe data relatert til prosjektet.

3.1 Teknologi

Formålet med oppgaven, som definert i problemstillingen, er å utforske bruk av programmering i undervisningen i kjemi, og videre utvikle et oppgavehefte med kjemirelaterte oppgaver som er løst med programmering. I forbindelse med dette målet er det nødvendig å spesifisere hvilken type teknologi som er blitt benyttet for å nå målet med oppgaven.

Det blir derfor i dette delkapittelet beskrevet hvilket programmeringsspråk, hvilke biblioteker og moduler for dette programmeringsspråket og hvilket utviklingsmiljø som er brukt i forbindelse med dette prosjektet.

3.1.1 Programmeringsspråk

Det finnes mange forskjellige programmeringsspråk, alle med egne bruksområder og vanskelighetsgrad. De fleste programmeringsspråk bruker lignende logikk, og man vil derfor kunne ta i bruk hva man lærer i et språk, inn i et annet. Men hvordan logikken er utformet kan fort bli komplisert i noen språk, som gjør at noen språk egner seg bedre til å lære programmering enn andre. I tillegg vil flere av språkene gå i dybden på forskjellige felter, som gjør at de har egne stryker og svakheter for forskjellige bruksområder.

Ved undervisningsopplegg er det viktig å ta til betraktning bruksområdet og vanskelighetsgraden til språket slik at det egner seg til undervisningen og ikke blir for komplisert for elevene. Med dette i betraktning er det valgt å bruke programmeringsspråket Python i denne oppgaven.

3.1.2 Python

Python er et høynivå programmeringsspråk, laget av Guido van Rossum i 1991. Det er et populært programmeringsspråk som egner seg blant annet til matematiske utregninger og bearbeiding av data (w3School, a).

Det er flere styrker til Python, blant annet er Python (w3School, a):

- Skrevet med enkel syntaks som ligner på engelsk
- Skrevet med en syntaks som tillater å gjøre mye med få linjer
- Godt egnet til å håndtere data
- Godt egnet til å utføre matematiske kalkulasjoner
- Egnet for hurtig prototyping som gjør det lett å eksperimentere.

3.1.3 Relevante Python biblioteker

I likhet med mange andre programmeringsspråk, kan Python utvides ved å installere og importere biblioteker. Dette gjør mye av arbeidet med språket enklere og er ofte nødvendig for å få programmet til å kjøre slik man ønsker. I dette delkapittelet listes det opp alle bibliotekene som blir brukt i oppgaveheftet.

Matplotlib: Matplotlib er et bibliotek for å lage statiske, animerte og interaktive visualiseringer av data i Python. Det ble utviklet av John D. Hunter i 2002, og har siden da vært i aktiv utvikling. (Hunter, 2007).

Mendeleev: Mendeleev er et bibliotek som tillater programmet å koble seg opp mot en database fult med informasjon om grunnstoffer, ioner og isotoper i det periodesystemet. Biblioteket er utviklet av Lukasz Mentel i 2014 og har siden da vært i aktiv utvikling. (Mentel, 2014-).

NumPy: NumPy er et bibliotek som utvider Python med funksjoner som er godt egnet til numerisk databehandling. Biblioteket ble utviklet i 2005 og har siden da vært i aktiv utvikling (NumPy, udatert).

Py3Dmol: Py3Dmol er et bibliotek som tillater programmet å koble seg opp mot to database fylt med visualiseringer om molekyler og kjemiske bindinger. Disse visualiseringene kan bli hentet fram ved bruk av Py3Dmol sine funksjoner koblet opp mot IDen til visualiseringen man ønsker å hente fram (Py3Dmol, udatert).

pHcalc: pHcalc er et bibliotek som utvider Python med funksjoner for beregning av pH i syrer og baser. Biblioteket beregner pHen i både sterke og svake syrer og baser basert på metoden «Systematic Equilibrium», hvor den beregner pH basert på ladningen til forbindelsen (Nelson 2016).

chemlib: chemlib er et bibliotek som utvider Python med mange funksjoner relatert til ulike deler av kjemifaget. Biblioteket er under aktiv utvikling hvor nye funksjoner blir stadig lagt til (Ambethkar, 2020).

3.1.4 Jupyter

Jupyter er et «non-profit, open-source» prosjekt, som vil si det er et prosjekt som er gratis å ta i bruk, og som man kan endre og bruke så mye man vil, så lenge man følger prosjektlisensen. Prosjektet er en videreutvikling av prosjektet «IPython» for å støtte interaktiv datavitenskap og vitenskapelig databehandling over flere ulike programmeringsspråk (Project-Jupyter, 2022).

Jupyter utvikler flere verktøyer i forbindelse med interaktiv programmering, og deling av dokumenter og visualiseringer av data. Et av disse verktøyene er Jupyter Notebooks som er en filtype som egner seg for å lage og dele dokumenter relatert til datavitenskapelige beregninger. Man kan lage disse dokumenttypene igjennom flere forskjellige programmer, blant annet Jupyter sitt eget, nettbaserte utviklingsmiljø «Jupyterlab». (Project-Jupyter, 2022).

Siden Jupyter Notebook er tilgjengelig igjennom sitt egne nettleserbaserte utviklingsmiljø kan man kjøre Jupyter Notebooks lokalt eller over en ekstern server. Ved bruk av en ekstern server kan en lærer sette opp programvaren og gi elever tilgang igjennom en nettleser, slik at elevene ikke trenger å bekymre seg for å installere ting korrekt (Kluyver, mfl., 2016).

3.2 Metode

I prosjektet er det valgt å ta i bruk forskningsmetoden «Design Science Research».

«Design-Research er et paradigme som har røtter i ingeniørarbeid og vitenskapen rundt det kunstige. Det er i grunnen et problemløsningsparadigme. Det ønsker å utvikle innovasjoner som definerer ideer, praksiser, tekniske egenskaper og produkter som kan brukes for å oppnå effektiv og suksessfull analyse, design, implementasjon, ledelse og bruk av informasjonssystemer (IS).» (Hevner, March & Park, 2004, s. 76).

«Design Science» (designforskning) metoden produserer innovasjoner rundt et tema, kalt artefakter i denne sammenhengen, som bruker eksisterende kjerneteorier (kernel theories) som blir brukt, testet, endret og utvidet av forskerens erfaringer, kreativitet, intuisjon og problemløsningsegenskaper. (Hevner, March & Park, 2004, s. 76).

Design er et viktig tema for forskning innen IS, ettersom, ifølge Benbasat og Zmud (1999, s. 5), relevansen til IS-forskningen er direkte knyttet til dens anvendelighet i design (Hevner, March & Park, 2004).

Målet med forskningsmetoden er derfor å gi retningslinjer for forskningen rundt designprosessen av et informasjonssystem. Igjennom metoden blir det laget og evaluert artefakter som kan svare på problemstillingen til forskningen. Evalueringen av artefaktene blir en viktig del av designprosessen ettersom det vil gi et bedre innblikk i problemet, som kan videre lede til forbedringer av artefaktene og produktet (Hevner, March & Park, 2004).

Ifølge Hevner, March og Park (2004) finnes det syv retningslinjer for utføring av designforskning:

1. Utvikling av en artefakt – Designforskning må produsere en brukbar artefakt.
2. Relevansen til problemet – Formålet med forskningen skal være å utvikle en løsning til et viktig og relevant problem.
3. Evaluering av designet – Brukbarheten, kvaliteten og effektiviteten til design-artefaktene må demonstreres igjennom evaluering av artefaktet.
4. Forskningsbidrag – Designforskningen må lede til klare og verifiserbare bidrag til temaer rundt design-artefaktet
5. Forskningsgrunnlag - Designforskningen skal støttes til solide metoder for utvikling og evaluering av artefaktet.
6. Design som en søkeprosess – Søkeprosessen etter en effektiv artefakt skal bruke tilgjengelige ressurser for å nå formålet, samtidig som at det skal tilfredsstille lover i miljøet til problemet.
7. Forskningskommunikasjon – Forskningen skal presenteres på en effektiv måte, som kan forstås av både et publikum med teknisk-bakgrunn og et publikum med leder-bakgrunn.

I tillegg har Hevner (2007) beskrevet tre forskningssykluser som man må igjennom ved en effektiv utføring av forskningsmetoden:

1. «Relevance» syklusen – I relevanssyklusen blir det satt fokus på å definere problemet og dens relevansen innen sitt miljø. I syklusen blir det definert kravene til forskningen, i tillegg til at det blir definert akseptable kriterier for evalueringen av resultatet.
2. «Rigor» syklusen – I rigorsyklusen blir det satt fokus på å knytte forskningsarbeidet opp imot tidligere forskning for å styrke grunnlaget til forskningsarbeidet, og lede til nytenking og innovasjon i prosjektet.
3. «Design» syklusen – I designsyklusen blir det satt fokus på å utføre arbeid rundt å utvikle, evaluere og forbedre en artefakt, basert på input fra de to andre syklusene.

Begrunnelsen for valget av metoden er at det er en relevant metode for utføring av prosjektet, basert på prosjektet er en kombinasjon av et forskningsarbeid og produksjon av en ressurs som forsøker å løse et problem. Forskningen til prosjektet tar for seg anvendelse av programmering i kjemi og artefaktet fra prosjektet vil være oppgaveheftet.

Prosjektet begynte med rigorsyklusen, hvor det blir utforsket eksisterende løsninger og forskning relevant til temaet. Dette inkluderte relevant litteratur knyttet til fagfornyelsen, programmering i skolen og programmering i kjemi, hvor sistnevnte inkluderte å oppdage eksisterende oppgaver relatert til programmering i kjemi.

Videre gikk prosjektet inn i relevanssyklusen hvor resultatet fra rigorsyklusen ble brukt til å definere relevansen til problemet og utvikle krav til forskning og evaluering av resultatet. Resultatet fra relevanssyklusen definerte at oppgavene skulle samsvare med fagfornyelsens krav om at fagene skal lede til økt dybdelæring, i tillegg til at de skal samsvare med fagets kjerneelementer. Videre ble det også definert at oppgaveheftet skal kunne brukes av personer med mindre forkunnskaper innen programmering til støtte for å utvikle undervisningsopplegg.

Deretter begynte designsyklusen som involverte å vurdere eksisterende oppgaver, strukturere dokumentet og velge ut oppgaver som er engasjerende, motiverende og leder til kreativitet blant elevene, i tillegg til å samsvare med fagets krav om å oppnå økt dybdelæring.

På grunn av tidsbegrensinger og arbeidsmengde ble ikke oppgavene evaluert av eksterne testere, som vil si en viktig del av designsyklusen ble utelatt. For at prosjektet skal kunne kvalitetssikres burde denne evalueringen gjøres som videre arbeid, og ved behov gå tilbake til designsyklusen.

4 Resultater

Kjemifaget på videregående skole er delt opp i to fag, kjemi 1 og kjemi 2. De har ulike kompetansemål, men samme kjerneelementer. Elevene velger kjemi 1 på andre år i videregående, og kan gå videre med kjemi 2 i det tredje året om ønsket. Ingen av fagene er obligatoriske, men bygger videre på blant annet det obligatoriske faget naturfag.

I dette kapittelet blir det først beskrevet resultater fra forskningsarbeidet utført som den første delen av prosjektet, så blir det vist fram noen eksempler fra oppgaveheftet.

4.1 Resultat forskningsdel

Som definert i problemstillingen og i visjonsdokumentet var formålet med prosjektet å finne ut hvilke oppgaver som kjemilærere kunne bruke, for å lettere anvende programmering i kjemifaget på videregående skole. Videre ble dette formålet delt inn i to deler, først en forskningsdel hvor det skulle utforskes hvordan man kan bruke programmering for å løse kjemirelaterte oppgaver, deretter en produksjonsdel hvor det blir laget et oppgavehefte, basert på resultatet fra forskningsdelen.

For å løse den første delen av oppgave ble det dermed gjort et forskningsarbeid om bruk av programmering i kjemifaget, hvor det da ble undersøkt eksisterende ressurser som tok for seg det å bruke programmering relatert til kjemifaget.

Den første delen av forskningsarbeidet gitt ut på å finne ut hvordan programmering og digitale ferdigheter var en viktig del av kjemifaget, og hvordan programmering kunne brukes i relasjon til kjemifaget. Resultatet fra dette var at elevene er nødt til å utvikle sine digitale ferdigheter, for å være forberedt til videre studier og arbeid i et samfunn i stadig teknologisk utvikling. I tillegg til dette viser det seg også at programmering tillater elever å utforske fagstoff på nye måter, hvor de kan utforske mer i dybden, som er viktig for at elevene skal øke kompetanse innen faget.

I relasjon til kjemifaget er digitale ferdigheter noe elevene kan utvikle ved bruk av digitale ressurser til å utdype faglig stoff, i form av blant annet modellering, datahåndtering, og visualisering. Programmering er et verktøy som kan hjelpe med flere av disse punktene, på en måte som tillater elevene å utforske fagstoff på nye, kreative måter, et sentralt konsept

innen dybdelæring. I tillegg kan programmering brukes til å hjelpe med komplekse og tungvinte deler av faget, for å øke motivasjon hos elevene.

Den andre delen av forskningsarbeidet innebar å samle oppgaver for kjemifaget kunne bli løst ved bruk av programmering, utvikle algoritmer for disse oppgavene og lage programkode ut ifra disse algoritmene. Det ble forsøk å ta oppgaver som viste hvordan man kunne bruke programmering i kjemifaglig stoff, slik at lærere kan lage egne, lignende oppgaver basert på eget undervisningsopplegg.

4.2 Resultat oppgaveheftet

For å utvikle oppgaveheftet ble det undersøkt flere kilder om eksisterende løsninger til problemet. Oppgavene er dermed hentet fra nettbaserte og trykte ressurser:

- Trykte:
  - Aqua 1 Studiebok (Steen, Fimland & Juel, 2021)
- Nettbaserte:
  - Realfaglig programmering i skolen (CCSE, 2021)
  - Programmering i kjemi av Andreas Haraldsrud

Tabell 4.1 viser en oversikt over alle temaene og antall oppgaver under hvert tema.

Tema	Antall oppgaver
Modellering	3
Lesing og plotting av data	3
Kjemiske beregninger	3
Statistikk	3

Tabell 4.1: Fordeling av oppgaver

Her er noen eksempler på oppgaver i oppgaveheftet:

4.2.1 Periodesystemet

Oppgave: Lag et program som spør etter et atomnummer og som responderer med atomnummer, navn, symbol og gruppe.

Oppgave-konseptet er hentet fra «Realfaglig programmering i skolen» (CCSE, 2021)

Løsning: For å løse denne oppgaven bruker vi biblioteket «mendeleev», som gjør det mulig å hente data om grunnstoff fra en database. For å bruke mendeleev må vi først importere biblioteket før vi kan begynne å bruke biblioteket i programmet.

Først må vi få atomtallet fra brukeren. Deretter lager vi et objekt, vi kaller dette grunnstoff i denne oppgaven, som vi knytter opp mot mendeleev sin element()-funksjon, ved bruk av atomnummeret brukeren skriver inn.

Skriv så opp en variabel til navn, symbol og gruppe, og knytt disse opp til deres respektive verdier i grunnstoff-objektet. Skriv så ut resultatet.

Algoritme:

1. Importer element from mendeleev
2. Lag en variabel for atomnummer
    a. Knytt denne opp til en input() funksjon hvor bruker kan skrive inn atomnummer
3. Lag objektet «grunnstoff» for grunnstoffet som knyttes opp mot element(atomnummet)
4. Lag variabelen «navn» som knyttes opp mot grunnstoff.name
5. Lag variabelen «symbol» som knyttes opp mot grunnstoff.symbol
6. Lag variabelen «gruppe» som knyttes opp mot grunnstoff.group_id
7. Skriv en print()-funksjon som skriver ut verdiene «atomnummer», «navn», «symbol» og «gruppe».

Programkode:

```
#  Importer mendeleev biblioteket
from mendeleev import element

#  Lag input for atomnummeret
atomnummer = int(input('Skriv inn atomnummer: '))

# Lag objekt for grunnstoffet
grunnstoff = element(atomnummer)

# Lag variabler for navn, symbol og gruppe
navn = grunnstoff.name
symbol = grunnstoff.symbol
gruppe = grunnstoff.group_id

# Skriv ut resultatet
print("Atomnummer:", atomnummer, "--Navn:", navn, "--Symbol:", symbol, "--
Gruppe:", gruppe)
```

Programkode 4.1: Periodesystemet

Resultat:

```
Skriv inn atomnummer: 5
Atomnummer: 5 --Navn: Boron --Symbol: B --Gruppe: 13
```

Resultat 4.1: Periodesystemet

4.2.2 Plotting av lister med data

Oppgave: Vi har målt damptrykket til vann i et glassrør ved ulike temperaturer. Vi har fått følgende data:

Temperatur (˚C)	Damptrykk (kPa)
16	1.817
18	2.063
20	2.339
22	2.644
24	2.984

Tabell 4.2: Damptrykket til vann i et glassrør ved ulike temperaturer

Lag et program som leser inn disse verdiene og plotter de ut.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsningen: For å løse oppgaven bruker vi biblioteket Matplotlib for å få tilgang til det biblioteket sine plotte-funksjoner.

Lag en liste som inneholder temperaturene og en liste som inneholder damptrykket. Plott ut figuren med bruk av Matplotlib sin plot()-funksjon. Om man ønsker å gjøre figuren mer presenterbar ved å modifisere plottet med bruk av Matplotlib sine funksjoner.

Algoritme:

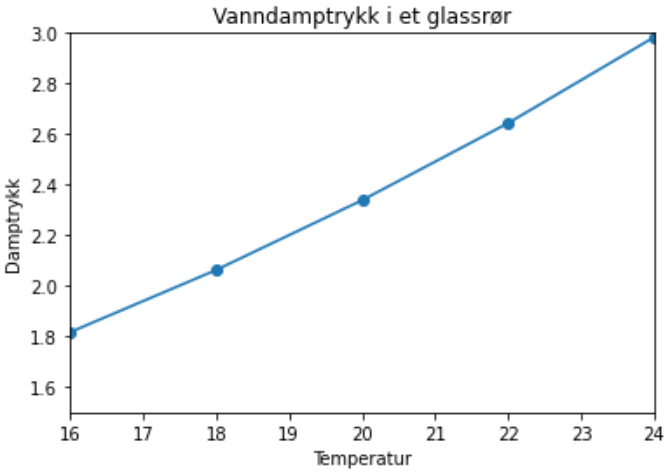
1. Importer matplotlib.pyplot som plt
2. Lag liste for temperaturene
3. Lag liste for damptrykket
4. Plott ut figuren med plt.plot(temperaturen, damptrykket)
    a. Gi plottet tittel med title()-funksjonen
    b. Gi plottet beskrivelser til x- og y-aksen med xlabel() og ylabel()
    c. Begrens plottet til min- og maksverdien til dataen med xlim() og ylim()
5. Vis figuren med plt.show()

Programkode:

```
# Importer Matplotlib.pyplot
import matplotlib.pyplot as plt

# Lag liste for temperaturene og damptrykket
temp = [16, 18, 20, 22, 24]
trykk = [1.817, 2.063, 2.339, 2.644, 2.984]

# Plott ut listene med Matplotlib
plt.plot(temp, trykk, marker="o")

# Gjør ønskede endringer på presentasjonen av figuren
plt.title("Vanndamptrykk i et glassrør")
plt.ylabel("Damptrykk")
plt.xlabel("Temperatur")
plt.ylim(1.5, 3)
plt.xlim(16,24)

# Vis figuren
plt.show()
```

Programkode 4.2: Plotting av lister med data

Resultat:



Resultat 4.2: Plotting av lister med data

4.2.3 pH-beregninger

Oppgave:

a) Lag et program som regner ut pH verdien til den sterke basen NaOH og den sterke syren HCl basert på en konsentrasjon man skriver inn.
b) Lag et nytt program som regner ut pH verdien til NaOH og HCl ved bruk av Python biblioteket pHcalc. Utvid programmet til å også inkludere den svare syren HF.

Oppgave-konseptet er hentet fra «Realfaglig programmering i skolen» (CCSE, 2021)

4.2.3.1 a)

Løsning: For å løse oppgaven bruker vi biblioteket NumPy.

Først må vi skrive inn konsentrasjonen til stoffene, enten ved bruk av en input. Deretter må vi skrive linjer som utfører beregningen av pH.

For NaOH må man først regne ut pOH som er lik -log10(OH^-). Siden NaOH er en sterk base vil konsentrasjonen av OH^+ være lik konsentrasjonen av NaOH. Deretter kan vi regne ut pH ved å ta i bruk formelen pH + pOH = 14, som vil si pH = 14 – pOH.

For HCl må man regne ut pH som er like -log(H^+). Siden HCl er en sterk syre vil konsentrasjonen av H^+ være lik konsentrasjonen av HCl.

Vi skriver så ut resultatet.

Algoritme:

1. Importer NumPy som np
2. Lag variabler for konsentrasjonen til NaOH og HCl
    a. Knytt disse opp mot en input()-funksjon hvor bruker skriver inn konsentrasjonen i M.
3. Regn ut pOH for NaOH ved bruk av formelen pOH = -log10[OH^-]
    a. OH^- har samme konsentrasjon som NaOH
4. Regn ut pH for NaOH ved bruk av formelen pH + pOH = 14
5. Regn ut pH for HCl ved bruk av formelen pH = -log10(H^+)
    a. H^+ har samme konsentrasjon som HCl
6. Skriv ut resultatene med print()

Programkode:

```
# Importer NumPy
import numpy as np

# Skriv inn konsentrasjonen
k_NaOH = float(input("Skriv inn konsentrasjonen til NaOH i M: "))
k_HCl = float(input("Skriv inn konsentrasjonen til HCl i M: "))

# Regn ut pOH og pH hos NaOH
pOH_NaOH = -np.log10(k_NaOH)
pH_NaOH = 14 - pOH_NaOH

#Regn ut pH hos HCl
pH_HCl = -np.log10(k_HCl)

# Skriv ut resultatet
print("pH-verdi for NaOH:", round(pH_NaOH, 2), "pH-verdi for HCl:",
round(pH_HCl, 2))
```

Programkode 4.3: pH-Beregninger a)

Resultat:

```
Skriv inn konsentrasjonen til NaOH i M: 0.1
Skriv inn konsentrasjonen til HCl i M: 0.1
Skriv inn konsentrasjonen til HF i M: 0.1
pH-verdi for NaOH: 13.0 pH-verdi for HCl: 1.0 , pH-verdi for HF: 1.0
```

Resultat 4.3: pH-Beregninger a)

4.2.3.2 b)

Løsning: For å løse oppgaven bruker vi biblioteket pHcalc, som er et bibliotek som hjelper med å utføre beregninger syrer og baser. pHcalc har tre objekter som man kan bruke: Acid(), Neutral() og System(). pHcalc justerer for H_30^+ og OH^- internt, så ved beregningene må man kun definere verdiene for resten.

Først må vi skrive inn konsentrasjonen inn i programmet. Deretter kan vi bruke pHcalc for å gjøre beregningen.

For den sterke basen NaOH må vi lage et nytt Neutral() objektet. Der trenger vi konsentrasjonen og ladningen som parametere, ladningen for NaOH blir da 1 siden pHcalc justerer for OH^- internt så man må bruke verdiene for Na^+. Så nå vi lage et nytt System() objekt som bruker Neutral() Objektet som parameter. Til slutt kjører vi System() objektet igjennom funksjonen pHsolve().

For den sterke syren HCl må vi gjøre det samme som for NaOH, men ladningen blir -1 i stedet, ettersom man må bruke verdien for Cl^-.

For den svake syren HF må vi i bruke Acid() objektet i stedet for Neutral() objektet, og vi må ha verdien for pKa, som for HF er 3.17, eller Ka som er 6.76e-4. Ellers er det lik som med de sterke syrene.

Algoritme:

1. Importer pHcalc.pHcalc med objektene Acid(), Neutral() og System()
2. Lag variabel k_NaOH satt lik en input for konsentrasjonen til NaOH
3. Lag variabel k_HCl satt lik en input for konsentrasjonen til HCl
4. Lag variabel k_HF satt lik en input for konsentrasjonen HF
5. Lag et nytt Neutral() objekt kalt Na
    a. Sett inn ladningen 1 og konsentrasjonen k_NaOH
    b. Lag et nytt System() objekt kalt pH_Na, gi den parameteren Na.
    c. Kjør pH_Na objektet igjennom pHsolve()-funksjonen
6. Lag et nytt Neutral() objekt kalt Cl
    a. Sett inn ladningen -1 og konsentrasjonen k_HCl
    b. Lag et nytt System() objekt kalt pH_Cl, gi den parameteren Cl.

   c. Kjør pH_Cl objektet igjennom pHsolve()-funksjonen
7. Lag et nytt Acid() objekt kalt HF
   a. Sett inn pKa 3.17, ladningen 0 og konsentrasjonen k_HF
   b. Lag et nytt System() objekt kalt pH_HF, gi den parameteren HF
   c. Kjør pH_HF objektet igjennom pHsolve()-funksjonen
8. Skriv ut pH_Na, pH_Cl og pH_HF
   a. Rund av til andre desimal med round()-funksjonen

Programkode:

```
# Importer pHcalc
from pHcalc.pHcalc import Acid, Neutral, System

# Skriv inn konsentrasjonene
k_NaOH = float(input("Skriv inn konsentrasjonen til NaOH i M: "))
k_HCl = float(input("Skriv inn konsentrasjonen til HCl i M: "))
k_HF = float(input("Skriv inn konsentrasjonen til HF i M: "))

# Definer Neutral objektet til Na
Na = Neutral(charge=1, conc=k_NaOH)
pH_Na = System(Na)
pH_Na.pHsolve()

# Definer Neutral objektet til Cl
Cl = Neutral(charge=-1, conc=k_HCl)
pH_Cl = System(Cl)
pH_Cl.pHsolve()

# Definer Acid objektet til HF
HF = Acid(pKa=3.17, charge=0, conc=k_HF)
pH_HF = System(HF)
pH_HF.pHsolve()

# Skriv ut resultatene:
print("pH-verdi for NaOH:", round(pH_Na.pH, 2), ", pH-verdi for HCl:",
round(pH_Cl.pH, 2), ", pH-verdi for HF:", round(pH_HF.pH, 2))
```

Programkode 4.4: pH-Beregninger b)

Resultat:

```
Skriv inn konsentrasjonen til NaOH i M: 0.1
Skriv inn konsentrasjonen til HCl i M: 0.1
Skriv inn konsentrasjonen til HF i M: 0.1
pH-verdi for NaOH: 13.0 , pH-verdi for HCl: 1.0 , pH-verdi for HF: 2.1
```

Resultat 4.4: pH-Beregninger b)

4.2.4 Grafisk framstilling av statistisk spredning:

Oppgave: Vi skal konstruere en standardkurve for magnesiumkonsentrasjonen i en vannprøve. Vi bruker en serie på 0.2, 0.3, 0.4, 0.5 og 0.6 µg/mL Mg^{2+} som vi analyserer tre ganger hver med et flammeatomabsorpsjonsspektrofotometer. Da har vi tre målinger for absorpsjon per konsentrasjon. Plott disse målingene og usikkerheten med disse målingene ved bruk av NumPy.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: Først må vi importere Matplotlib og NumPy. Deretter må vi skrive inn dataen inn i lister, først en for konsentrasjonen og en for absorbans. Lag må vi lage en liste som samler standardavviket og en liste som samler gjennomsnittet av absorpsjonsmålingene, og lage en løkke som regner ut dette ved bruk av NumPy sin std()- og mean()-funksjon. Deretter plottet vi ut figuren ved bruk av Matplotlib sin funksjon errorbar() som plotter dataen inn i en figur med usikkerhetsstolper.

Algoritme:

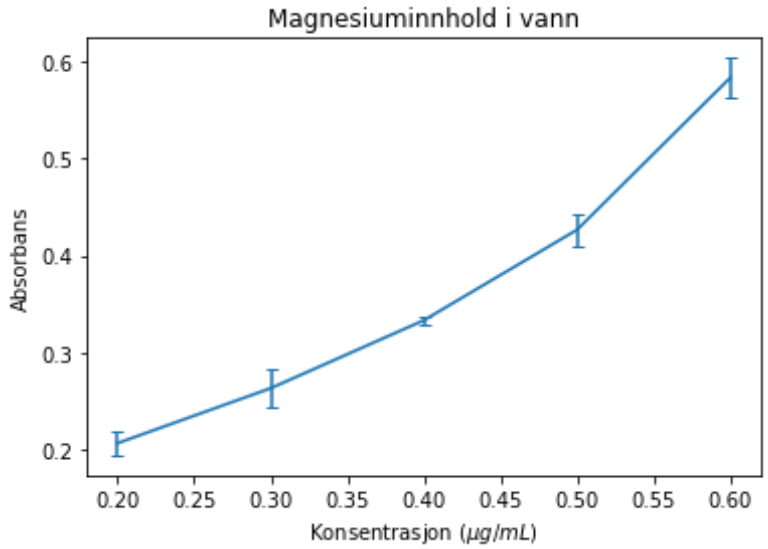
1. Importer matplotlib.pyplot som plt, og numpy som np
2. Lag en liste for konsentrasjonen og absorbans
3. Lag så en tom liste for standardavviket og gjennomsnittet
4. Skriv en for-løkke som utfører std() og mean() på hvert element i absorbanslisten
    a. Bruk append()-funksjonen for å legge resultatet inn i listen for standardavvik og gjennomsnittet
5. Plott ut figuren med bruk av plt.errorbar() funksjonen
    a. Legg til deskriptive funksjoner for å presentere funksjonen bedre
6. Vis figuren med bruk av plt.show()

Programkode:

```
# Importer Matplotlib og NumPy
import matplotlib.pyplot as plt
import numpy as np

# Lag liste for målignene
konsentrasjon = [0.2, 0.3, 0.4, 0.5, 0.6]
absorbans = [[0.21, 0.22, 0.19], [0.26, 0.29, 0.24], [0.33, 0.33, 0.34],
[0.41, 0.42, 0.45], [0.56, 0.61, 0.58]]

# Lag en tom liste for standardavviket og gjennomsnittet
standardavvik = []
snitt = []

# Lag en for-løkke som regner ut standardavviket og gjennsomsnittet for
målingene i absorbans
for element in absorbans:
    standardavvik.append(np.std(element))
    snitt.append(np.mean(element))

# Plott ut figuren med usikkerhetsstoler
plt.errorbar(konsentrasjon, snitt, yerr = standardavvik, capsize=3)
plt.errorbar
plt.title("Magnesiuminnhold i vann")
plt.xlabel("Konsentrasjon ($\mu g/mL$)")
plt.ylabel("Absorbans")
plt.show()
```

Programkode 4.5: Grafisk fremstilling av statistisk spredning

Resultat:



Resultat 4.5: Grafisk fremstilling av statistisk spredning

5 Diskusjon

I dette prosjektet ble problemstillingen del i to:

1. utforske bruk av programmering i kjemifagene, og
2. utvikle et oppgavehefte som samsvarer med de nye læreplanene og støtter kjemilærere i undervisning av programmering i kjemi.

I dette kapittelet blir det diskutert resultatet til de to delene, i tillegg til en reflekterende analyse av hvordan prosjektet har foregått.

5.1 Diskusjon forskningsdel

Den første delen av oppgaven handlet om å utføre et forskningsarbeid for å undersøke konseptet om å bruke programmering i kjemi og hvordan man best kan få oppnådd dette. For å oppnå målet med denne delen ble det undersøkt flere ressurser som tok for seg konseptet og lignende konsepter. En grundigere beskrivelse om hva som ble oppdaget i utforskningen er beskrevet i kapittel 1 og 2. I dette delkapittelet blir det gjennomgått en kortere oppsummering av hva som ble oppdaget.

I 2020 læreplanene i den norske skolen fornyet, på grunn av at de tidligere læreplanene ikke tillot elevene å gå i dybden på viktigste delene av fagene, og at de gamle læreplanene ikke lengre ga elevene god nok kompetanse til videre studier og arbeidsliv i et stadig økende teknologisk samfunn. De nye læreplanene satte blant annet fokus på begrepet dybdelæring.

For at elevene skal oppnå en tilfredsstillende kompetanse innen digitale ferdigheter har det blitt bestemt at programmering skal brukes i undervisningen i relevante fag. I tillegg er det konkludert med at programmering kan lede til økende grad av dybdelæring ved å la elevene løse problemer på nye måter og utforske fagene i ett nytt lys.

I forbindelse med kjemi kan programmering bli tatt i bruk for å undervise flere av kjemiens kjerneelementer, på en måte som kan oppnå dybdelæring hos elevene. Det tillater elevene å utforske faget på en ny måte, i tillegg til å hjelpe med å arbeide med mer komplekse og tidskrevende deler av faget.

Men implementering av programmering i undervisningen, både i kjemifaget og i andre fag, kommer med flere komplikasjoner. En av disse ligger i lærerkompetanse innen

programmering, et fåtall av lærere har gode nok forkunnskaper til å utvikle undervisningsopplegg som tar i bruk programmering, på en tilfredsstillende måte.

Det er utviklet flere ressurser for å løse problemet, blant annet kurs og kompetansepakker, men effekten av disse løsningene tar tid. Flere lærere mener at de ikke får nok støtte til å utvikle undervisningsopplegg, i forbindelse med dette kan det være nyttig å utvikle et oppgavehefte, med løsningsforslag, som lærere kan bruke som en referanse for eget undervisningsopplegg.

5.2 Diskusjon oppgavehefte

Den andre delen av prosjektet handlet om å utvikle et oppgavehefte, med kjemifaglige oppgaver som løses ved bruk av programmering. Utviklingen av oppgavehefte ble gjort igjennom tre faser: samling av oppgaver, utvalg av oppgaver og utvikling av løsninger, i form av algoritmer og programkode.

5.2.1 Samling av oppgaver

Det første steget i utviklingen av oppgaver var å finne eksisterende oppgaver som passet til målet. I forbindelse med dette ble det derfor utforsket eksisterende ressurser som man kunne hente oppgavene fra.

Samlingen av oppgavene ble gjort ut ifra kravene spesifisert i visjonsdokumentet. Disse var som følgende:

- Oppgavene må være beskrevet med fremgangsmåte og løsningsforslag
- Oppgavene må være relevante for fagstoffet
- Oppgavene burde gi et resultat (en output)
- Oppgavene burde bidra til å øke kreativitet og engasjement for elever.
- Oppgavene skal bidra til bedre dybdeforståelse.

I første omgang ble det forsøkt å finne eksisterende oppgaver for kjemifaget på videregående skole, hvor programmering ble brukt under løsningen av oppgaven. I forbindelse med dette ble det utforsket forskjellige trykte og nettbaserte ressurser for kjemiundervisning på videregående skole.

Det viste seg at det ikke eksisterte mange oppgaver hvor programmering ble brukt, i de ressursene som ble utforsket. Det ble ikke funnet mange oppgaver i de undersøkte lærebøkene hvor programmering ble tatt i bruk, og i forskjellige kurs-relaterte ressurser var det ikke utviklet oppgaver, men heller en beskrivelse om hvordan man kunne ta i bruk programmering i undervisningen av kjemi.

Grunnen til dette kan ligge i at programmering i kjemifaget på videregående nivå er et lite utviklet område, og at det derfor vil ta tid før undervisningsressurser lager tilfredsstillende oppgaver som bruker kjemi til å løse oppgaven.

Etter at det ikke ble oppnådd et tilfredsstillende samling av eksisterende oppgaver, ble det videre forsøk å finne løsninger på hvordan programmering kunne bli brukt i kjemi, og bruke dette til å løse kjemifaglige oppgaver. I forbindelse med dette ble det utforsket flere nettbaserte ressurser, både i form at etterutdanningskurs for lærere og programmeringskurs for kjemistudenter ved høgskoler og universiteter.

Disse ressursene ga innsikt i hvordan kjemi kunne egne seg i kjemifaget, som viste seg å være på flere punkter. En av de største fordelene med programmering i kjemisammenheng viste seg å være hvordan man kunne bruke programmering til å visualisere, modellere og utforske kjemifaglig stoff. I tillegg kunne programmering hjelpe med numeriske beregninger relatert til kjemi, spesielt da om beregningene ble tungvinte og kompliserte, noe som kan påvirke elevenes motivasjon.

Disse oppdagelsene passet inn i fokuset på dybdelæring ved at det tillater elevene å utforske og reflektere over resultatene på nye måter. I tillegg ledet denne bruken av programmering i kjemi til at elevene kunne bearbeide, analysere og presentere egne resultater fra forsøk gjort i undervisningen, ved bruk av programmering. Dette kan lede til økt engasjement og kreativitet hos elevene, i tillegg til at det tillater elevene å utforske fagstoffet dypere, på nye måter.

Ut ifra disse funnene ble det, blant annet, samlet og utviklet oppgaver som var lettere å tilpasse til utdanningsopplegget som læreren utvikler, slik at elevene kan arbeide med resultater de produserer selv fra forsøk.

5.2.2 Utvalg av oppgaver

Originalt var tanken at oppgaveheftet skulle fordeles inn i de to fagene Kjemi 1 og Kjemi 2. Denne tanken ble fort endret basert på to oppdagelser.

Den første var mangel på gode ressurser som tillot å samle oppgaver, løst med bruk av programmering, som var unike til kompetansemålene for de to fagene. Dette var et stort problem for Kjemi 2, ettersom flere av de trykte ressursene som var laget etter de nye læreplanene, var ikke utgitt enda. Den andre oppdagelsen var at noen av de sterkeste sidene for bruk av programmering i kjemifaget, var like for Kjemi 1 og Kjemi 2. En av grunnene til dette var at en av de mest praktiske bruksområdene for programmering i kjemi var i samsvar med kjerneelementene, som begge fagene delte.

Oppgaveheftet ble derfor delt på temaer, og oppgavene ble fordelt inn i fire temaer:

1. **Modellering** – Modellering er allerede et tema som står sentralt i kjerneelementene, og modellering ved bruk av digitale ressurser er en sentral del av de digitale ferdighetene. Programmering tillater elevene å utforske modellering av kjemifaglig stoff på nye måter, ofte på en enklere måte enn uten programmering. Dette kan lede til økt kreativitet og engasjement. Modellering er også et sentralt tema i flere kompetansemål både i kjemi 1 og kjemi 2.
2. **Lesing og plotting av data** – En av de største bruksområdene for programmering i kjemi viste seg å være det å samle inn, bearbeide og analysere data fra forsøk. Å gjøre dette ved bruk av digitale ressurser er også en sentral del av de digitale ferdighetene. Det å kunne skrive inn, bearbeide og presentere data som elevene selv har skaffet kan lede til engasjement og kreativitet. I tillegg vil det å la elevene reflektere om hvordan de skal bearbeide og presentere dataen, tillater elevene å gå mer i dybden på resultatene og utforske resultatene på nye måter, som leder til økt dybdelæring.
3. **Kjemiske beregninger** – Å kunne utføre ulike kjemiske beregninger er et sentralt tema innen kjerneelementene. Å kunne bruke digitale verktøy for å utføre beregninger er også en del av de digitale ferdighetene. Programmering kan brukes til å lage programmer som utfører beregninger, slik at man løser beregningene på en algoritmisk måte. I tillegg kan man bruke biblioteker og egne funksjoner for å gjøre oppgaver som ellers ville vært komplekse og tidskrevende å gjøre noe som vil ha en effekt på motivasjon og engasjement hos elevene.

4. **Statistikk** – Å kunne gjøre statistiske utregninger på kjemifagligstoff og resultatet er en viktig del av å bearbeide og analysere data. Programmering kan tillate elevene å gjøre slike utregninger på en enklere og mindre tidskravene måte som kan ha en effekt på engasjement. I tillegg tillater det elevene å analysere resultatene på nye måter.

Oppgavene er valgt og samlet for fagene kjemi 1 og kjemi 2. Det er ikke utviklet oppgaver relatert direkte til kjemiprosessfaget i yrkesfaglig fordypning, men flere av temaene og oppgavene kan være relevante for undervisningen i det faget og.

5.2.3 Utvikling av løsninger

En viktig mål for oppgaveheftet var at det skulle kunne være til støtte for lærere med lite forkunnskaper innen programmering. Det er derfor hentet eller utviklet løsninger for alle oppgavene i form av algoritmer og programkode.

Algoritmene har som formål å være en form for pseudokode, som vil si de skal være lesbare og fungere som instrukser på hvordan man løser oppgaven, ved å konvertere instruksene til programkode.

Programkoden er et eksempel kjørt igjennom Jupyter Notebooks og inkluderer resultatet til programkoden. Det er også inkludert kommentarer som beskriver programmet på en måte som følger algoritmen.

5.3 Refleksjon

Formålet med oppgaveheftet er at det skal kunne brukes som en støttende ressurs til kjemilærere når de skal utvikle et undervisningsopplegg som bruker programmering i kjemifaget. For å undersøke om dette formålet er oppnådd må oppgaveheftet testes, som kan lede til forbedringsmuligheter. Ved dette tilfellet kan oppgaveheftet bli endret, utvidet eller brukt som en del av et annet dokument.

Igjennom prosjektet er det erfart en mangel på kjemifaglige oppgaver og metoder for bruk av programmering i kjemifaget på videregående nivå. Dette er et problem som kan forbedres over tid, ettersom programmering blir i økende grad brukt i undervisningen av fagene. Dokumenter som dette oppgaveheftet kan spille en rolle i å hjelpe med å normalisere

programmering i undervisningen, slik at kvaliteten på undervisningsopplegget øker, og bedre oppgaver blir laget.

Denne mangelen på ressurser gjorde samling og utvikling av oppgaver en mer tidskravene prosess en antatt, som ledet til en mulig svakhet at det kunne blitt utviklet flere oppgaver til oppgaveheftet. De oppgavene som ble utviklet burde derimot ha et stort bruksområde, som vil kunne være til hjelp ved flere deler av undervisningsopplegget.

Prosjektet har krevd en god del tid til forskningsdelen, ettersom det krevde en grundig fordypning i temaer som kjemifaget, fagfornyelsen, programmering i skolen og programmering i Python. På grunn av mangel på tilgang til oppdaterte ressurser ble det mer tidskrevende enn antatt å utvikle egne oppgaver, der eksisterende løsninger mangler, spesielt da for fagstoff i kjemi 2.

5.3.1 Evaluering

En viktig del av forskningsmetoden som ble brukt under prosjektet var evaluering av oppgavene, men denne delen ble ikke utført i sin helhet på grunn av tidsmangel og arbeidsmengde.

Det er utført en grad av evaluering fra utvikler, ved å evaluere artefaktet opp mot resultatet fra forskningsdelen, da hovedsakelig ved å knytte det opp mot læreplanen i kjemi. Resultatet fra denne evalueringen er beskrevet i diskusjonsdelen rundt oppgaveheftet.

For å effektivt evaluere artefaktet må det testes av sluttbrukerne av produktet, som vil være kjemilærere med få til ingen forkunnskaper i programmering. Denne evalueringen kan gi innblikk i om løsningen, algoritmen og programkoden til oppgavene er forståelig, og om oppgavene er relevante for undervisningsopplegget.

Det er også mulig å få gode tilbakemeldinger om faglig relevans og muligheter for andre oppgaver ved å la interessenter med mer kjemifaglig kompetanse og kompetanse innen pedagogikk evaluere artefaktet.

6 Konklusjon

I dette prosjektet er det blitt utforsket bruk av programmering i kjemifaget på videregående skole, og det er blitt utviklet et oppgavehefte med kjemirelaterte oppgaver som tar i bruk programmering i løsningen.

Oppgaveheftet et utviklet etter behov for støttende ressurser til lærere ved utvikling av undervisningsopplegg i kjemi. Det er forsøkt å samle oppgaver som leder til økt dybdelæring, engasjement og kreativitet hos elevene. Målet med oppgaveheftet, er at oppgavene kan bli bruk av kjemilærere til å utvikle undervisningsopplegg som bruker programmering som et verktøy for å utforske fagstoffet. Målet er også at det skal være en brukbar ressurs for kjemilærere med lav eller ingen kompetanse innen programmering.

6.1 Videre arbeid

Videre arbeid med oppgaveheftet vil inkludere evaluering og forbedring av oppgaveheftet. I løpet av prosjektet har oppgaveheftet ikke blitt testet og evaluert på grunn av tidsbegrensinger og arbeidsmengde. Evaluering av oppgaveheftet kan innebære:

- Evaluering av algoritmene, blant annet testing om algoritmen blir godt nok presentert til at alle som følger den kan utvikle et dataprogram som løser oppgaven
- Evaluering av faglig relevansen, som innebærer å vurdere hvor oppgaver kan passe bedre med annen type data eller med andre lignede oppgaver

I tillegg kan det vurderes å bedre utvikle oppgaveheftet til å være rettet mot å støtte eksisterende undervisningsopplegg. Flere av oppgavene passer bra inn som videre arbeid fra praktiske forsøk i laboratoriet, for å analysere og presentere data fra resultater. Slike opplegg ble det ikke mulig for meg å utvikle på grunn av manglende kompetanse og erfaringer innen kjemifaglig pedagogikk.

7 Referanser

Ambethkar, H. (2020) *chemlib*. Hentet fra https://chemlib.readthedocs.io/en/latest/#

Benbasat, I., Zmud, R. W. (1999). *Empirical Research in Information Systems: The Practice of Relevance*. MIS Quarterly, vol. 23, no. 1, pp. 3-16. Hentet fra https://doi.org/10.2307/249403

Center for Computing in Science Education (CCSE), Universitetet i Oslo. (2021) *Realfaglig programmering i skolen*. Hentet fra https://realprog.no/docs/intro.html

Dvergsdal, H. (2022). Pseudokode. *Store Norske Leksikon*. Hentet fra https://snl.no/pseudokode

Haraldsrud, A. (2021) *Programmering i kjemi*. Hentet fra https://andreasdh.github.io/programmering-i-kjemi/docs/intro.html

Hevner, A. R., March S. T., Park, J., (2004) *Design Science in Information Systems Research*. MIS Quarterly, vol. 28, no 1, pp. 75-105, Hentet fra https://doi.org/10.2307/25148625

Hevner, A. R. (2007) *A Three Cycle View of Design Science Research*. Scandinavian Journal of Informational Systems: Vol. 19: Iss. 2, Article 4. Hentet fra https://aisel.aisnet.org/sjis/vol19/iss2/4/

Hunter, J. D. (2007) "*Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3m pp. 90-95*. Hentet fra https://matplotlib.org/3.5.2/index.html#

Kunnskapsdepartementet. (2018). Fornyer innholdet i skolen. Hentet fra https://www.regjeringen.no/no/aktuelt/fornyer-innholdet-i-skolen/id2606028/

Lær Kidsa Koding. (udatert) *Lær Kidsa Koding*. Hentet fra https://www.kidsakoder.no/om-lkk/

Mentel, L. M. (2014-), *mendeleev – A python resource for properties of chemical elements, ions and isotopes. ,* Hentet fra https://github.com/lmmentel/mendeleev

Mohammadi, S. (2021) *Anvendelse av programmering i matematikk: Matematisk modellering, algoritmer og programkoder som støtter matematikklærere i programmerings undervisning på den norske skolen.* Bacheloroppgave. Norges Teknisk-Naturvitenskapelige Universitet. Hentet fra https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2778046

7. Referanser

NOU (2014:7) *Elevenes læring i fremtidens skole – Et kunnskapsgrunnlag*. Oslo: Departementenes sikkerhets- og serviceorganisasjon, informasjonsforvaltning. Hentet fra https://www.regjeringen.no/no/dokumenter/NOU-2014-7/id766593/

NOU (2015:8). *Fremtidens skole. Fornyelse av fag og kompetanser*. Oslo: Departementenes sikkerhets- og serviceorganisasjon, informasjonsforvaltning. Hentet fra https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/

NumPy (Udatert) *NumPy*. Hentet fra https://numpy.org/

Olsen, A. L. (2005) *Using Pseudocode to teach problem solving*. J. Comput. Sci. Coll., 21(2), 231-236

Project-Jupyter (2022), *Jupyter*. Hentet fra https://jupyter.org/

Py3Dmol (udatert) *Py3Dmol*. Hentet fra https://3dmol.csb.pitt.edu/

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

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

Utdanningsdirektoratet (2022) *Innføring og overgangsordninger for nye læreplaner*. Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/innforing-og-overgangsordninger-for-nye-lareplaner/#

Utdanningsdirektoratet. (udatert, a) *Læreplan i kjemi (KJE01-02)*. Fastsatt som forskrift. Læreplanverket for kunnskapsløftet 2020. Hentet fra https://www.udir.no/lk20/kje01-02

Utdanningsdirektoratet (udatert, b) *Digitale ferdigheter som grunnleggende ferdighet*. Hentet fra https://www.udir.no/laring-og-trivsel/rammeverk/rammeverk-for-grunnleggende-ferdigheter/2.1-digitale-ferdigheter/

Utdanningsdirektoratet (udatert, c) *Fagfornyelsen*. Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/

Rossen, E. (2019) Programmering (IT). *Store Norske Leksikon*. Hentet fra https://snl.no/programmering_-_IT

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

7. Referanser

Steen, B. G., Fimland, N., Juel, L. A., (2021) *Aqua 1 Studiebok*. 3. Utgave. Oslo: Gyldendal Norsk forlag.

Stenlund, E. (2021) *Programmering og Fagfornyelsen*. Masteroppgave. Universitetet i Oslo. Hentet fra https://www.duo.uio.no/handle/10852/87187

W3Schools. (udatert, a) *Python Introduction*. Hentet fra https://www.w3schools.com/python/python_intro.asp

W3Schools. (udatert, b) *Python variables*. Hentet fra https://www.w3schools.com/python/python_variables.asp

W3Schools. (udatert, c) *Python comments*. Hentet fra https://www.w3schools.com/python/python_comments.asp

W3Schools. (udatert, d) *Python data types*. Hentet fra https://www.w3schools.com/python/python_datatypes.asp

W3Schools. (udatert, e) *Python operators*. Hentet fra https://www.w3schools.com/python/python_operators.asp

8 Vedlegg

Prosjektet inneholder to vedlegg:

A. Oppgavehefte
B. Visjonsdokument

Sammen med rapporten blir det også levert en fil som inneholder prosjekthåndboken og kildekode til oppgavene.

Vedlegg A

1 Introduksjon

Dette dokumentet er en samling av kjemifaglige oppgaver som løses med bruk av programmering.

Programmeringsspråket som brukes i oppgavene er programmeringsspråket Python. Dokumentet begynner med en grunnleggende beskrivelse av viktige konsepter og begreper i programmeringsspråket, i tillegg til lenker til nettressurser, om man ønsker å fordype seg i språket.

Oppgavene er strukturert på en slik måte at det først kommer en problemstilling, etterfulgt av et løsningsforslag, etterfulgt av en algoritme for å løse oppgaven og til slutt et eksempel på programkoden for oppgaven og resultatet fra den programkoden.

Oppgavene er fordelt i delkapitler som er rettet til relevante temaer for kjemifaget. Fagstoffet som blir brukt i oppgavene kan være relevant for både kjemi 1 og kjemi 2, og flere av oppgavene kan knyttes opp mot mer fagstoff enn hva oppgaven bruker.

Målet med dokumentet er at det skal fungere som en støttende ressurs til kjemilærere ved anvendelse av programmering i kjemi. Dokumentet dekker ikke alle temaene eller kompetansemålene i kjemi, men flere av løsningene til de forskjellige oppgavene kan brukes i andre temaer.

2 Programmering i Python

Python er et programmeringsspråk som er laget med fokus på å være lesbart. Man kan utvikle flere nyttige programmer ved bruk av det, blant annet er det et godt språk til håndtering og beregning av data.

I dette kapittelet blir det beskrevet kort noen grunnleggende begrep og funksjoner, men det går ikke mye i dybden på disse. For å lære mer om de forskjellige temaene er det anbefalt å se i Python sin dokumentasjon (Engelsk: https://www.python.org/doc/, Norsk: https://wiki.python.org/moin/NorwegianLanguage). I tillegg kan den engelske nettsiden W3Schools anbefales (https://www.w3schools.com/python/default.asp), om man ikke har mye forkunnskaper om programmering.

Oppgavene er skrevet ved bruk av utviklingsmiljøet Jupyter Notebooks (https://jupyter.org/).

2.1 Variabler

Variabler er måten man lagrer verdier i programkoden. De definerer med et navn, satt lik en verdi. Man kan deretter sette kalle på verdien ved å bruke navnet på variabelen. Man kan endre verdien til variablene i programkoden flere ganger, men den vil alltid ta i bruk siste satte verdi. Variabler kan defineres innenfor og utenfor funksjoner, men variabler opprettet i en funksjon kan kun brukes i den funksjonen.

2.1.1 Objekter

Objekter er en type variabel som er en samling av andre variabler. Objekter blir definert av en «Class», og brukes i objekt-orientert programmering. Objekter blir ikke lager i oppgavene, men et par biblioteker henter objekter som brukes i oppgavene.

2.1.2 Lister

Man kan også lagre data inn i lister, kalt arrays, i Python. Disse listene starter på 0, så om man vil hente ut første ledd i listen må man vise til ledd 0.

2.2 Kommentarer

Python har mulighet til å skrive kommentarer i koden. Det vil si at det man skriver ikke regnes som en kommandolinje for programmet og den vil ignorere linjen når programmer kjører. Slik kan man blant annet beskrive funksjoner for å lettere lese og forstå programkoden.

Man bruker kommentarer med å begynne kommando linken med en emneknagg, «#». Man kan også kommentere bort flere linjer ved å starte og slutte med tre anførselstegn, «"""». (W3Schools, udatert, c).

2.3 Datatyper

Datatyper er en dypere beskrivelse av verdien man setter til en variabel. Forskjellige typer data kan brukes forskjellig og må behandles forskjellige i koden. Følgende datatyper finnes i Python (W3School, udatert, d):

- Teksttyper (Streng): «str»
- Talltyper:
  - Heltall: «int»
  - Flyttall: «float»
  - Komplekse tall: «complex»
- Sekvenstyper: «list», «tuple», «range»
- Mappingtyper: «dict»
- Settyper: «set», «frozenset»
- Boolsk typer: «bool»
- Binære typer: «bytes», «bytearray», «memoryview»
- None type: «none»

2.4 Operatorer

Operatorer brukes til å utføre oppgaver med variabler og verdier. De deles opp i flere forskjellige grupper, blant annet (W3Schools, udatert, e):

- Regneoperatorer (Arithmetric operators): Disse brukes blant annet til å gjøre matematiske utregninger på variabler og verdier. Tabell 2.1 viser en oversikt over noen av de viktigste regneoperatorene.

Operator	Tegn
Addisjon	+
Subtraksjon	-
Multiplikasjon	*
Divisjon	/
Potens	**
Parantes	()

Tabell A.2.1: Regneoperatorer i Python

- Tildelingsoperatorer (Assignment operators): Disse brukes til å sette en verdi lik en annen gitt verdi, og blir blant annet bruk for å definere variabler. Tabell 2.2 viser noen av de viktigste tildelingsoperatorene og hva de betyr.

Operator	Eksempel	Betyr
=	X = 1	X = 1
+=	X += 1	X = X + 1
−=	X −= 1	X = X - 1

Tabell A.2.2: Tildelingsoperatorer i Python

- Sammenligningsoperatorer (Comparison operators): Disse brukes til å sammenligne to verdier. Tabell 2.3 viser en oversikt over noen av de viktigste sammenligningsoperatorene.

Operator	Tegn
Mindre enn	<
Større enn	>
Lik	=
Mindre enn, eller lik	<=
Større enn, eller lik	>=
Ulik	!=

Tabell A.2.3: Sammenligningsoperatorer i Python

- Logikkoperatorer (Logical operators): Responderer sant eller usant til utsagn som blir beskrevet med “AND”, “OR”, “NOT”.

2.5 Regnerekkefølge

Python har en bestemt rekkefølge for utregning av operatorer. Tabell 2.4 viser regnerekkefølgen hvor venstre blokk er prioritert først, og høyre blokk blir prioritert sist. Samme blokk har samme prioritering.

()	**	*, /, %, //	+, -

Tabell A.2.4: Regnerekkefølge i Python

2.6 Løkker

Python har to former for løkker:

For-løkke: En for-løkke fortsetter å gå et gitt antall ganger, definert av et uttrykk som man definerer i for-løkken.

Programkode 2.1 er en for-løkke som skriver ut hvert symbol i edelgass-listen.

```
edelgasser = ["He", "Ne", "Ar", "Kr", "Xe", "Rn", "Og"]
for x in edelgasser:
      print(x)
```

Programkode A.2.1: Eksempel for-løkke

While-løkke: En while-løkke fortsetter så lenge et gitt uttrykk er korrekt, eller en «break» kommando blir kjørt.

Programkode 2.2 er en while-løkke som skriver ut tallene 1-5.

```
i = 1
while i < 6:
      print(i)
      i += 1
```

Programkode A.2.2: Eksempel på en while-løkke

2.7 Betingelser

Python bruker betingelsessetningene «if», «if-else» og «if-elif-…-else». Disse brukes for å bestemme hvordan programmet skal oppføre seg ved når gitte utsagn er korrekte eller ukorrekte.

- if «utsagn»:

  Om utsagnet er korrekt, gjøre det som følger dette, om ikke gå videre

- elif «utsagn»:

  Om dette utsagnet er i stedet korrekt, gjør det som følger dette, om ikke gå videre
- else:

  Om ingen if eller elif utsagn var korrekte, gjør det som følger dette.

2.8 Innebygde funksjoner

Python har en del innebygde funksjoner som man kan bruke uten å definere de i forveien. Tabell 2.5 viser et par eksempler på innebygde funksjoner som blir brukt i oppgaveheftet

Funksjon	**Beskrivelse**
float()	Returnerer et flyttall
input()	Tillater bruker å skrive inndata til programmet
int()	Returnerer et heltall
len()	Returnerer antall verdier i en liste
print()	Skriver utdata
round()	Runder data til gitt desimal

Tabell A.2.5: Eksempler på innebygde funksjoner i Python

2.9 Egendefinerte funksjoner

Det er mulig å definere egne funksjonene i Python, slik at man lettere kan gjenbruke kode flere ganger i samme program. Disse funksjonene kjører når man kaller de i programmet. Man kan legge til data, kjent som parametere i denne sammenhengen, inn i funksjonen. En funksjon defineres med bruk av nøkkelordet «def» etterfulgt av et navn.

Programkode 2.3 er en funksjon som skriver ut «Jeg er en funksjon» når den blir kalt.

```
def min_funksjon():
      print("Jeg er en funksjon")

min_funksjon()
```

Programkode A.2.3: Eksempel på egendefinert funksjon

2.10 Biblioteker

Python har flere biblioteker og moduler man kan laste ned og importere inn i programkoden for å utvide funksjonene man kan utføre. For å ta i bruk bibliotekene må man først installere de, så importere de inn i programmet man ønsker å bruke de i. For mer informasjon om dette, besøk deres nettsteder eller PyPi side, som et oppslagsverk for Python biblioteker.

I dette oppgaveheftet blir det brukt flere biblioteker, det er anbefalt å lese igjennom dokumentasjonen til disse bibliotekene for å forstå bedre hvordan de fungerer:

- Matplotlib: https://matplotlib.org/
- Mendeleev: https://mendeleev.readthedocs.io/en/stable/
- NumPy: https://numpy.org/
- Py3Dmol: https://3dmol.csb.pitt.edu/
- pHcalc: https://github.com/rnelsonchem/pHcalc
- chemlib: https://chemlib.readthedocs.io/en/latest/#

3 Kompetansemål

Kompetansemål etter kjemi 1

Mål for opplæringen er at eleven skal kunne:

- forstå og bruke kjemisk terminologi og regler for navnsetting i faglig kommunikasjon
- planlegge og gjennomføre forsøk, estimere usikkerhet og vurdere feilkilder, presentere resultater og argumentere for gyldigheten av resultater og konklusjoner
- bruke informasjon fra sikkerhetsdatablad til å gjøre vurderinger knyttet til helse, miljø og sikkerhet i praktisk arbeid
- bruke data, simuleringer og beregninger i tolkninger og til å trekke konklusjoner
- bruke modeller til å forklare observasjoner og kjemiske fenomener, og argumentere for modellenes styrker og begrensinger
- gjøre rede for oppbygningen av periodesystemet, og bruke kjerneladning og elektronkonfigurasjon til å forklare periodiske trender
- gjøre rede for kjemisk binding som elektrostatiske krefter som virker mellom partikler, og bruke dette til å forklare molekylgeometri og organiske og uorganiske stoffers struktur, sammensetning og egenskaper
- utforske og gjøre beregninger på kjemiske reaksjoner, og bruke observasjoner og teoretiske vurderinger til å identifisere reaksjonstype
- gjøre beregninger med ulike enheter for konsentrasjon og bruke stoffkonsentrasjon i vurderinger av vann- og luftkvalitet
- gjennomføre volumetrisk og gravimetrisk titreranalyse og drøfte bruk av titreranalyse
- gjøre rede for sammenhengen mellom atomets oppbygning og grunnstoffers absorbsjons- og emisjonsspektre og bruke spektroskopiske metoder i kvalitativ og kvantitativ analyse
- gjøre rede for entalpi og bruke beregninger og forsøk til å utforske entalpiendringer i reaksjoner
- gjøre rede for kollisjonsteori og utforske faktorer som påvirker reaksjonsfart og kjemisk likevekt

- utforske løseligheten til stoffer, og gjøre rede for betydningen av ladning, polaritet og temperatur for løselighet
- gjøre rede for begrepene syre, base, protolyse og pH, og utforske egenskapene til sterke og svake syrer og baser
- gjøre rede for prinsipper for grønn kjemi og drøfte hvordan bruk av prinsippene kan bidra til bærekraftig utvikling
- presentere kjemifaglig innhold fra ulike kilder, kritisk vurdere kildene og bruke relevant teori til å drøfte innholdet

Hentet fra læreplanen i kjemi (utdanningsdirektoratet, udatert, a)

Kompetansemål etter kjemi 2

Mål for opplæringen er at eleven skal kunne:

- forstå og bruke kjemisk terminologi og fagspråk i faglig kommunikasjon
- planlegge og gjennomføre forsøk, drøfte metode og tiltak for å redusere risiko og vurdere usikkerhet og feilkilder i egne og andres forsøk
- gjøre rede for hvordan naturvitenskapelige modeller og teorier utvikles, og reflektere over hvordan samarbeid bidrar til kunnskapsutvikling i kjemi
- utforske redoksreaksjoner og bruke beregninger til å vurdere sammenhenger mellom masse, ladning, spenning og energi i elektrokjemiske reaksjoner
- utforske likevekter og bruke massevirkningsloven til å gjøre beregninger og forklare observasjoner
- gjøre rede for entropibegrepet og bruke entropi og entalpi til å vurdere spontanitet og endringer i likevektsystemer
- utforske og beregne pH i vannløsninger og drøfte betydningen av buffere for regulering av pH i naturlige og industrielle prosesser
- utforske og gjøre beregninger av løseligheten til stoffer og gjøre vurderinger av løselighet i biologiske og industrielle prosesser
- utforske katalyserte reaksjoner og gjøre rede for betydningen av katalysatorer i biologiske og industrielle prosesser

- gjøre rede for reaksjonstypene addisjon, eliminasjon, substitusjon, hydrolyse og kondensasjon og bruke elektrostatiske krefter til å forklare noen enkle reaksjonsmekanismer
- gjennomføre synteser og gjøre rede for faktorer som påvirker utbytte og renhet i synteser
- gjøre rede for prinsipper for kromatografi og bruke kromatografi for å separere og analysere organiske stoffblandinger
- beskrive oppbygningen til noen biologiske makromolekyler og vurdere hvordan ytre faktorer kan påvirke molekylenes struktur og egenskaper
- gi eksempler på produksjon, gjenvinning, deponering og nedbryting av noen metaller og noen typer plast, og drøfte tiltak som er i samsvar med prinsipper for grønn kjemi
- utforske en teoretisk eller praktisk problemstilling, og drøfte og presentere funn

Hentet fra læreplanen i kjemi (utdanningsdirektoratet, udatert, a)

4 Modellering

Modellering er en viktig del av kjemifaget og det å forstå modeller er en viktig ferdighet for elever. Programmering tillater elever å lage og utforske modeller, for å øke sin kompetanse om kjemiske modeller (Haraldsrud, 2021).

I dette kapittelet er det forsøkt å samle oppgaver som tar for seg hvordan man kan bruke programmering for å lage og utforske modeller.

4.1 Periodesystemet

Oppgave: Lag et program som spør etter et atomnummer og som responderer med atomnummer, navn, symbol og gruppe.

> Oppgave-konseptet er hentet fra «Realfaglig programmering i skolen» (CCSE, 2021)

Løsning: For å løse denne oppgaven bruker vi biblioteket «mendeleev», som gjør det mulig å hente data om grunnstoff fra en database. For å bruke mendeleev må vi først importere biblioteket før vi kan begynne å bruke biblioteket i programmet.

Først må vi få atomtallet fra brukeren. Deretter lager vi et objekt, vi kaller dette grunnstoff i denne oppgaven, som vi knytter opp mot mendeleev sin element()-funksjon, ved bruk av atomnummeret brukeren skriver inn.

Skriv så opp en variabel til navn, symbol og gruppe, og knytt disse opp til deres respektive verdier i grunnstoff-objektet. Skriv så ut resultatet.

Algoritme:

8. Importer element from mendeleev
9. Lag en variabel for atomnummer
    a. Knytt denne opp til en input() funksjon hvor bruker kan skrive inn atomnummer
10. Lag objektet «grunnstoff» for grunnstoffet som knyttes opp mot element(atomnummet)
11. Lag variabelen «navn» som knyttes opp mot grunnstoff.name

12. Lag variabelen «symbol» som knyttes opp mot grunnstoff.symbol
13. Lag variabelen «gruppe» som knyttes opp mot grunnstoff.group_id
14. Skriv en print()-funksjon som skriver ut verdiene «atomnummer», «navn», «symbol» og «gruppe».

Programkode:

```
#  Importer mendeleev biblioteket
from mendeleev import element

#  Lag input for atomnummeret
atomnummer = int(input('Skriv inn atomnummer: '))

# Lag objekt for grunnstoffet
grunnstoff = element(atomnummer)

# Lag variabler for navn, symbol og gruppe
navn = grunnstoff.name
symbol = grunnstoff.symbol
gruppe = grunnstoff.group_id

# Skriv ut resultatet
print("Atomnummer:", atomnummer, "--Navn:", navn, "--Symbol:", symbol, "--
Gruppe:", gruppe)
```

Programkode A.4.1: Periodesystemet

Resultat:

```
Skriv inn atomnummer: 5
Atomnummer: 5 --Navn: Boron --Symbol: B --Gruppe: 13
```

Resultat A.4.1: Periodesystemet

4.2 Elektronskall

Oppgave: Lag et program som spør etter et atomnummer mellom 1-20 og responderer med fordelingen av elektronene i skallene K, L, M og N.

Oppgaven er hentet fra «Aqua 1 Studiebok» (Steen, Fimland og Juel, 2021, s. 15).

Løsning: For å løse oppgaven må vi først spørre om et atomnummer imellom 1-20. I et normalt ladd grunnstoff vil atomnummeret være likt antallet elektroner, så vi kan dermed bruke det til å fortelle programmet antallet elektroner i grunnstoffet.

Deretter kan vi skrive if-elif-else-setninger for å si hvor mange elektroner det er i hvert skall, basert på at det er et maks antall elektroner i hver skall og i de første 20 grunnstoffene følger elektronfordelingen en logisk fordeling.

Algoritme:

1. Lag en variabel for atomnummer
    a. Knytt denne opp til en input() funksjon hvor bruker kan skrive inn atomnummer
2. Lag en if():-funksjon som sjekker om det er 2 eller mindre elektroner i grunnstoffet, om det er sant:
    a. Sett K skallet lik atomnummeret
    b. Sett L, M og N skallene lik 0
3. Lag en elif():-funksjon som sjekker om det er 10 eller mindre elektroner i grunnstoffet, om det er sant:
    a. Sett K skallet lik 2,
    b. Sett L skallet lik atomnummeret – 2
    c. Sett M og N skallet lik 0
4. Lag en elif():-funksjon som sjekker om det er 18 eller mindre elektroner i grunnstoffet, om det er sant:
    a. Sett K skallet lik 2
    b. Sett L skallet lik 8
    c. Sett M skallet lik atomnummeret – 10
    d. Sett N skallet lik 0
5. Lag en elif():-funksjon som sjekker om det er 20 eller mindre elektroner i grunnstoffet, som det er sant:
    a. Sett K skallet lik 2
    b. Sett L skallet lik 8
    c. Sett M skallet lik 8
    d. Sett N skallet lik atomnummeret - 18

6. Lag en print() funksjon som skriver ut resultatet

Programkode:

```
#  Lag input for atomnummeret
atomnummer = int(input('Skriv inn et atomnummer mellom 1 og 20: '))

# Lag betingelsessetningene
if (atomnummer <= 2):
    K = atomnummer
    L, M, N = 0, 0, 0
elif (atomnummer <= 10):
    K, M, N = 2, 0, 0
    L = atomnummer - 2
elif (atomnummer <= 18):
    K, L, N = 2, 8, 0
    M = atomnummer - 10
elif (atomnummer <= 20):
    K, L, M = 2, 8, 8
    N = atomnummer - 18

# Skriv ut resultatet
print("Elektroner i K-skallet: ", K, ". Elektroner i L-skallet: ", L, ".
Elektroner i M-skallet: ", M, ". Elektroner i N-skallet: ", N)
```

Programkode A.4.2: Elektronskall

Resultat:

```
Skriv inn atomnummer: 14
Elektroner i K-skallet:  2 . Elektroner i L-skallet:  8 . Elektroner i
M-skallet:  4 . Elektroner i N-skallet:  0
```

Resultat A.4.2: Elektronskall

4.3 Molekylvisualisering

Oppgave: Lag et program som lager en grafisk modell av molekylet for Paracetamol, ved bruk av biblioteket py3Dmol.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: For å løse oppgaven tar vi i bruk biblioteket py3Dmol. Dette er et bibliotek for Jupyter Notebooks som henter en grafisk visualisering fra de to databasene PDB (Protein Data Bank) og PubChem.

Først må vi finne ut IDen til Paracetamol i databasen PubChem. Denne er kalt «cid» og brukes sammen med kommandoen «query» for å hente fram visualiseringen.

Deretter importerer vi py3Dmol inn i programmet og bruker dens view()-funksjon med iden til Paracetamol som parameter. Så bruker vi setStyle()-funksjonen for å skrive ut figuren. setStyle()-funksjonen tillater flere typer modeller, i denne oppgaven bruker vi argumentet «stick» for modellen vår.

Først må man importere py3Dmol. Dette biblioteket gir tilgang til funksjoner som kan hente fram grafiske visualiseringer fra databasene. For å hente visualiseringen må man først finne IDen til molekylet i databasen man ønsker å hente den fra, her PubChem. Denne IDen heter i PubChem «cid» og har for Paracetamol IDen «1983».

Man bruker da funksjonen view() for å hente fram visualiseringen og bruker funksjonen setStyle() for å skrive ut visualiseringen.

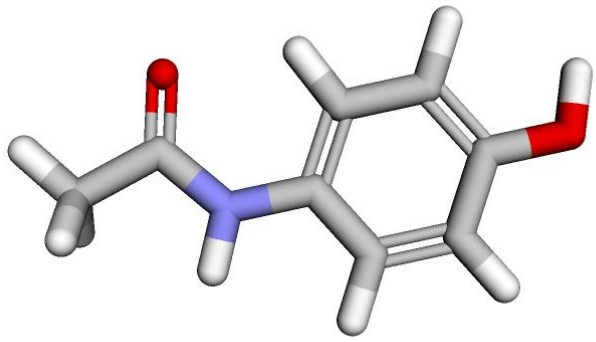
1. Finn ID for Paracetamol i PubChem
2. Importer py3Dmol
3. Lag en variabel paracetamol som er lik view(query="cid:IDen")
4. Skriv funksjonen setStyle() på variabelen paracetamol
    a. Sett inn argumentet for modelltypen «stick», eller hva man ønsker selv
    b. Sett inn argumentet for fargen «spectrum», eller hva man ønsker selv

Programkode:

```
# Importer py3Dmol
import py3Dmol

# Hent visualiseringsdataen fra databasen
paracetamol = py3Dmol.view(query="cid:1983")

# Skriv ut modellen
paracetamol.setStyle({"stick": {"color":"spectrum"}})
```

Programkode A.4.3: Molekylvisualisering

Resultat:



Resultat A.4.3: Molekylvisualisering

5 Lesing og plotting av data

En viktig del av kjemifaget er å kunne representere og behandle data. I forbindelse med dette kan programmering brukes til å bearbeide store mengder data og visualisere dette igjennom ulike plott (Haraldsrud, 2021).

I dette kapittelet er det forsøkt å samle oppgaver som tar for seg lesing og plotting av data, slik at man kan analysere og utforske dataen.

5.1 Plotting av lister med data

Oppgave: Vi har målt damptrykket til vann i et glassrør ved ulike temperaturer. Vi har fått følgende data:

Temperatur (˚C)	Damptrykk (kPa)
16	1.817
18	2.063
20	2.339
22	2.644
24	2.984

Tabell A.5.1:Damptrykket til vann i et glassrør ved ulike temperaturer

Lag et program som leser inn disse verdiene og plotter de ut.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsningen: For å løse oppgaven bruker vi biblioteket Matplotlib for å få tilgang til det biblioteket sine plotte-funksjoner.

Lag en liste som inneholder temperaturene og en liste som inneholder damptrykket. Plott ut figuren med bruk av Matplotlib sin plot()-funksjon. Om man ønsker å gjøre figuren mer presenterbar ved å modifisere plottet med bruk av Matplotlib sine funksjoner.

Algoritme:

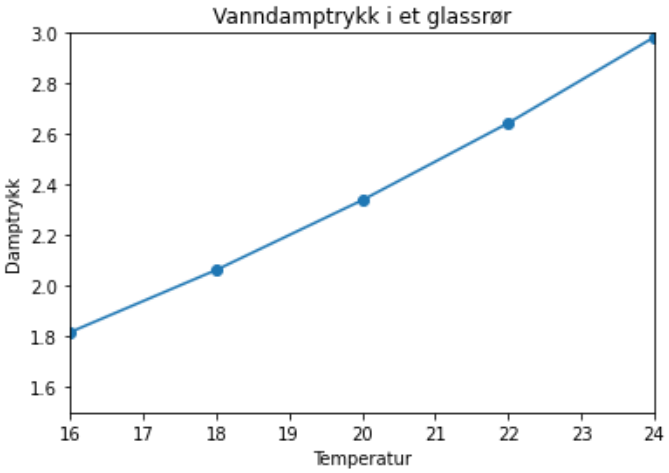
6. Importer matplotlib.pyplot som plt
7. Lag liste for temperaturene
8. Lag liste for damptrykket
9. Plott ut figuren med plt.plot(temperaturen, damptrykket)
    a. Gi plottet tittel med title()-funksjonen
    b. Gi plottet beskrivelser til x- og y-aksen med xlabel() og ylabel()
    c. Begrens plottet til min- og maksverdien til dataen med xlim() og ylim()
10. Vis figuren med plt.show()

Programkode:

```
# Importer Matplotlib.pyplot
import matplotlib.pyplot as plt

# Lag liste for temperaturene og damptrykket
temp = [16, 18, 20, 22, 24]
trykk = [1.817, 2.063, 2.339, 2.644, 2.984]

# Plott ut listene med Matplotlib
plt.plot(temp, trykk, marker="o")

# Gjør ønskede endringer på presentasjonen av figuren
plt.title("Vanndamptrykk i et glassrør")
plt.ylabel("Damptrykk")
plt.xlabel("Temperatur")
plt.ylim(1.5, 3)
plt.xlim(16,24)

# Vis figuren
plt.show()
```

Programkode A.5.1: Plotting av lister med data

Resultat:



Resultat A.5.1: Plotting av lister med data

5.2 Lesing av data fra en tekstfil

Ved bruk av blant annet biblioteket NumPy kan man lese inn data fra tekstfiler, som kan nyttig ved arbeid med resultater fra forsøk med mange resultater.

Oppgave: Vi har utført en titrering av 25 mL 0.12 M eddiksyre med 0.10 M lut (NaOH). Dataen fra titreringen er lagret i en tekstfil «titrering_eddiksyre_NaOH.txt». Lag et program som leser inn og plotter ut dataen. Tabell 5.2 viser et utdrag av tekstfilen.

Volum	pH
0.0	2.51
2.05	2.76
4.00	3.03
6.01	3.11
8.22	3.31

Tabell A.5.2: Utdrag fra tekstfilen titrering_eddiksyre_NaOH.txt

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: For å løse oppgaven tar vi i bruk bibliotekene Matplotlib og NumPy.

Først må vi lese inn dataen, som vi gjør ved bruk av NumPy sin loadtxt()-funksjon. Deretter må vi lage en liste for hver av kolonnene i tekstfilen.

Dette gjør vi ved bruk av data[], hvor man skriver inn hvilken kolonne man ønsker å hente ut ved å skrive inn parameteren [:, 0] for første kolonne og [:, 1]for andre. «:» tegnet i denne sammenheng forteller data[] at den skal hente ut kolonner. Første kolonne begynner på 0 siden lister begynner på 0 i Python.

Videre må vi plotte volumet og pHen inn i et plott ved bruk av Matplotlib sine funksjoner.

Algoritme:

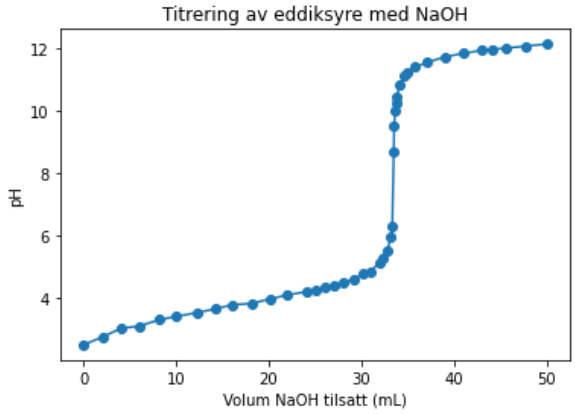
1. Importer matplotlib.pyplot som plt og numpy som np
2. Last inn dataen ved bruk av np.loadtxt()-funksjonen
3. Lag en liste for volumet ved bruk av data[] med parameter lik kolonnen til volumet
4. Lag en liste for pH ved bruk av data[] med parameter lik kolonnen pH-verdien ligger i
5. Plott ut volumet og pHen med bruk av plt.plot()-funksjonen
    a. Navngi plottet med title()-funksjonen,
    b. Navngi x- og y-aksen med xlabel() og ylabel()
6. Vis fram plottet med plt.show()

Programkode:

```
# Importer Matplotlib.pyplot og NumPy
import matplotlib.pyplot as plt
import numpy as np

# Last inn data fra tekstfilen ved bruk av numpy
data = np.loadtxt('titrering_eddiksyre_NaOH.txt', delimiter = ',',
skiprows=1)

# Lag liste for volum og pH
volum = data[:,0]
pH = data[:,1]

# Plott ut dataen med bruk av plt.plot()
plt.plot(volum, pH, marker='o')
plt.title("Titrering av eddiksyre med NaOH")
plt.xlabel("Volum NaOH tilsatt (mL)")
plt.ylabel("pH")
plt.show()
```

Programkode A.5.2: Lesing av data fra tekstfil

Resultat:



Resultat A.5.2: Lesing av data fra tekstfil

5.3 Regresjon

Oppgave: Vi har brukt et spektrometer for å måle absorbansen til ulike standardløsninger av Fe^{2+}. Dette har gitt oss følgende datasett (Tabell 5.2):

Standardløsning (mol/L)	Absorbans
0.0	0.0
0.010	0.021
0.030	0.120
0.060	0.139
0.080	0.186
0.100	0.208

Tabell A.5.3: Resultat fra å måle absorbansen til Fe^{2+}

Lag et program som gjør lineær regresjon på dataene og plotter datapunktene og den mest tilpassede regresjonskurven i samme koordinatsystem.

Oppgaven og løsningen er hentet fra «Programmering for kjemikere» (Haraldsrud, 2021a).

Løsning: For å løse oppgaven tar vi i bruk biblioteket Matplotlib og NumPy.

Først må vi skrive datasettet inn i programmet ved å skrive det inn i lister. For å utføre regresjonsanalyse må denne listene være i form av np.array(), slik at NumPy kan bruke listene i funksjonene sine.

Deretter bruker vi NumPy sin funksjon for polynomregresjon, np.polyfit, med listene som parameter etterfulgt av graden til polynomet. Denne funksjonen gir oss en liste med koeffisientene «a» og «b» for uttrykket «y= ax + b». Vi kan da bruke disse koeffisientene til å lage nye y-verdier basert på de opprinnelige x-verdiene.

Til slutt bruker vi Matplotlib for å plotte resultatet inn i en graf. Vi bruker scatter() funksjonen til å plotte inn datapunktene og plot() funksjonen til å plotte inn den tilpassede regresjonslinja. Til gjør vi deskriptive endringer på plottet for å presentere dataen på en god måte.

Algoritme:

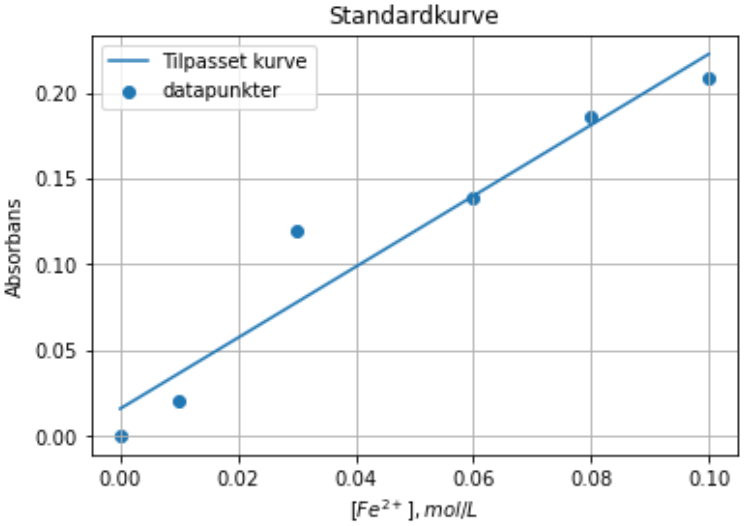
1. Importer matplotlib.pyplot som plt og numpy som np
2. Skriv data for standardløsningen inn i en np.array([…])) liste.
3. Skriv data for absorbans inn i en np.array([…]) liste.
4. Bruk np.polyfit(standardløsning, absorbans, grad) på listene, med polynomdivisjon av første grad.
5. Sett resultatet fra polyfit() inn i «y = ax + b» linje, hvor a og b er resultatene og x er standardløsningen
6. Plott datapunktene ved bruk av plt.scatter(…)
7. Plott regresjonslinja ved bruk av plt.plot(…)
    a. Endre deskriptive deler av plottet, som å gi tittel, navn på akser og rutenett.
8. Vis resultatet med plt.show()

Programkode:

```
# Importer Matplotlib og NumPy
import matplotlib.pyplot as plt
import numpy as np

# Skriv inn datasettet inn i lister
jern = np.array([0.0, 0.010, 0.030, 0.060, 0.080, 0.100])
absorbans = np.array([0.0, 0.021, 0.120, 0.139, 0.186, 0.208])

# Gjør første grads polynomdivisjon på datasettet
reg = np.polyfit(jern, absorbans, 1)

# Lag nye y-verdier med resultatet
y = (reg[0])*jern + reg[1]

# Plott resultatene
plt.scatter(jern, absorbans, label="datapunkter")
plt.plot(jern, y, label="Tilpasset kurve")
plt.legend()
plt.title("Standardkurve")
plt.xlabel("$[Fe^{2+}], mol/L$")
plt.ylabel("Absorbans")
plt.grid()
plt.show()
```

Programkode A.5.3: Regresjon

Resultat:



Resultat A.5.3: Regresjon

6 Kjemiske beregninger

Å utføre kjemiske beregninger er en sentral del av kjemifaget, for eksempel beregninger av kjemiske reaksjoner. Programmering kan brukes til å enklere utføre disse beregningene, som videre da tillater elever å utforske og tolke disse beregningene.

I dette kapittelet er det forsøkt å samle oppgaver som tar for seg ulike beregninger man utfører i kjemifaget, løst ved bruk av programmering.

6.1 Molar masse

Oppgave: Lag et program som beregner molar masse til en organisk forbindelse basert på antallet av grunnstoffene C, H og O.

Oppgaven er hentet fra «Aqua 1 Studiebok» (Steen, Fimland og Juel, 2021, s. 83).

Løsning: For å løse denne oppgaven må vi først skrive inn den molare massen til grunnstoffene, enten ved å skrive de manuelt inn eller med bruk av biblioteket mendeleev.

Deretter må vi spørre brukeren om antallet av hvert grunnstoff i forbindingen. Dette antallet ganger vi sammen med den molare massen til grunnstoffet og legger alle sammen inn en variabel for den totale molare massen.

Algoritme:

1. Importer element from mendeleev og hent dataen til karbon (6), hydrogen (1) og oksygen (8), eller finn og skriv inn den molare massen til grunnstoffene manuelt.
2. Skriv tre input()-funksjoner som spør etter hvor mange det er av hvert grunnstoff
3. Gang antallet av hvert grunnstoff med sin molare masse
4. Legg sammen alle de nye verdiene
5. Print() ut resultatet

Programkode:

```
# Importer mendeleev
from mendeleev import element

#  Hent data om hvert element fra mendeleev
karbon = element(6)
hydrogen = element(1)
oksygen = element(8)

# Spør bruker om antallet av grunnstoffene
C = int(input("Skriv inn antall karbonatomer: "))
H = int(input("Skriv inn antall hydrogenatomer: "))
O = int(input("Skriv inn antall oksygenatomer: "))

# Gang antallet av hvert grunnstoff med dens molare masse
C_mass = C * karbon.mass
H_mass = H * hydrogen.mass
O_mass = O * oksygen.mass

# Legg sammen de molare massene
mol_mass = C_mass + H_mass + O_mass

# Skriv ut resultatet
print(mol_mass, "g/mol")
```

Programkode A.6.1: Molar masse

Resultat:

```
Skriv inn antall karbonatomer: 6
Skriv inn antall hydrogenatomer: 12
Skriv inn antall oksygenatomer: 6
180.156 g/mol
```

Resultat A.6.1: Molar masse

6.2 Likevekt

Oppgave: Finn ut som reaksjonslikningen $2SO_2 + O_2 \rightarrow 2SO_3$ er i likevekt.

Oppgave-konseptet er hentet fra «Realfaglig programmering i skolen» (CCSE, 2021)

Løsning: For å løse denne oppgaven tar vi i bruk biblioteket chemlib. Chemlib har to nyttige objekter for denne oppgaven, «Compound()» og «Reaction()». Compound() gir programmet en representasjon av en forbinding, som Reaction() bruker for å utføre en reaksjonsliking.

Først lager vi en variabel for hver forbindelse og knytter disse opp mot Compound()-objektet. Deretter lager vi en variabel for reaksjonen og knytter denne opp til Reaction()-objekt og legger variablene for forbindelsene inn som parameter.

Deretter kjører vi en funksjon balance() som sjekker om reaksjonen er i likevekt. Skriv så ut likningen med funksjonen «reaksjon.formula» og svaret på om likningen er i likevekt med funksjonen «reaksjon.balance».

Algoritme:

1. Importer Compound og Reaction fra chemlib
2. Lag en variabel for hver av forbindelsene og knytt de opp mot Compound()-objektet
3. Lag en variabel for reaksjonen og knytt den opp mot Reaction()-objektet
4. Bruk reaction.balance() for å sjekke om reaksjonslikningen er i likevekt
5. Skriv ut formelen med reaction.formula() og svaret på om reaksjonen er i likevekt med reaction.is_balanced().

Programkode:

```
# Importer chemlib
from chemlib import Compound, Reaction

# Skriv inn de tre forbindelsene
SO2 = Compound("SO2")
O2 = Compound("O2")
SO3 = Compound("SO3")

#  Kjør reaksjonen igjennom Reaction funksjonen til chemlib
reaksjon = Reaction([SO2, O2], [SO3])
reaksjon.balance()
print(reaksjon.formula)
print("Reaksjonen er i likevekt =", reaksjon.is_balanced)
```

Programkode A.6.2: Likevekt

Resultat:

```
2S1O2 + 1O2 --> 2S1O3
Reaksjonen er i likevekt = True
```

Resultat A.6.2: Likevekt

6.3 pH-beregninger

Oppgave:

c) Lag et program som regner ut pH verdien til den sterke basen NaOH og den sterke syren HCl basert på en konsentrasjon man skriver inn.

d) Lag et nytt program som regner ut pH verdien til NaOH og HCl ved bruk av Python biblioteket pHcalc. Utvid programmet til å også inkludere den svare syren HF.

Oppgave-konseptet er hentet fra «Realfaglig programmering i skolen» (CCSE, 2021)

6.3.1 a)

Løsning: For å løse oppgaven bruker vi biblioteket NumPy.

Først må vi skrive inn konsentrasjonen til stoffene, enten ved bruk av en input. Deretter må vi skrive linjer som utfører beregningen av pH.

For NaOH må man først regne ut pOH som er lik -log10(OH^-). Siden NaOH er en sterk base vil konsentrasjonen av OH^+ være lik konsentrasjonen av NaOH. Deretter kan vi regne ut pH ved å ta i bruk formelen pH + pOH = 14, som vil si pH = 14 – pOH.

For HCl må man regne ut pH som er like -log(H^+). Siden HCl er en sterk syre vil konsentrasjonen av H^+ være lik konsentrasjonen av HCl.

Vi skriver så ut resultatet.

Algoritme:

7. Importer NumPy som np
8. Lag variabler for konsentrasjonen til NaOH og HCl
    a. Knytt disse opp mot en input()-funksjon hvor bruker skriver inn konsentrasjonen i M.
9. Regn ut pOH for NaOH ved bruk av formelen pOH = -log10[OH^-]
    a. OH^- har samme konsentrasjon som NaOH
10. Regn ut pH for NaOH ved bruk av formelen pH + pOH = 14
11. Regn ut pH for HCl ved bruk av formelen pH = -log10(H^+)
    a. H^+ har samme konsentrasjon som HCl
12. Skriv ut resultatene med print()

Programkode:

```
# Importer NumPy
import numpy as np

# Skriv inn konsentrasjonen
k_NaOH = float(input("Skriv inn konsentrasjonen til NaOH i M: "))
k_HCl = float(input("Skriv inn konsentrasjonen til HCl i M: "))

# Regn ut pOH og pH hos NaOH
pOH_NaOH = -np.log10(k_NaOH)
pH_NaOH = 14 - pOH_NaOH

#Regn ut pH hos HCl
pH_HCl = -np.log10(k_HCl)

# Skriv ut resultatet
print("pH-verdi for NaOH:", round(pH_NaOH, 2), "pH-verdi for HCl:",
round(pH_HCl, 2))
```

Programkode A.6.3: pH-Beregninger a)

Resultat:

```
Skriv inn konsentrasjonen til NaOH i M: 0.1
Skriv inn konsentrasjonen til HCl i M: 0.1
Skriv inn konsentrasjonen til HF i M: 0.1
pH-verdi for NaOH: 13.0 pH-verdi for HCl: 1.0 , pH-verdi for HF: 1.0
```

Resultat A.6.3: pH-Beregninger a)

6.3.2 b)

Løsning: For å løse oppgaven bruker vi biblioteket pHcalc, som er et bibliotek som hjelper med å utføre beregninger syrer og baser. pHcalc har tre objekter som man kan bruke: Acid(), Neutral() og System(). pHcalc justerer for H_30^+ og OH^- internt, så ved beregningene må man kun definere verdiene for resten.

Først må vi skrive inn konsentrasjonen inn i programmet. Deretter kan vi bruke pHcalc for å gjøre beregningen.

For den sterke basen NaOH må vi lage et nytt Neutral() objektet. Der trenger vi konsentrasjonen og ladningen som parametere, ladningen for NaOH blir da 1 siden pHcalc justerer for OH^- internt så man må bruke verdiene for Na^+. Så nå vi lage et nytt System() objekt som bruker Neutral() Objektet som parameter. Til slutt kjører vi System() objektet igjennom funksjonen pHsolve().

For den sterke syren HCl må vi gjøre det samme som for NaOH, men ladningen blir -1 i stedet, ettersom man må bruke verdien for Cl^-.

For den svake syren HF må vi i bruke Acid() objektet i stedet for Neutral() objektet, og vi må ha verdien for pKa, som for HF er 3.17, eller Ka som er 6.76e-4. Ellers er det lik som med de sterke syrene.

Algoritme:

9. Importer pHcalc.pHcalc med objektene Acid(), Neutral() og System()
10. Lag variabel k_NaOH satt lik en input for konsentrasjonen til NaOH
11. Lag variabel k_HCl satt lik en input for konsentrasjonen til HCl
12. Lag variabel k_HF satt lik en input for konsentrasjonen HF
13. Lag et nytt Neutral() objekt kalt Na
    a. Sett inn ladningen 1 og konsentrasjonen k_NaOH
    b. Lag et nytt System() objekt kalt pH_Na, gi den parameteren Na.
    c. Kjør pH_Na objektet igjennom pHsolve()-funksjonen
14. Lag et nytt Neutral() objekt kalt Cl
    a. Sett inn ladningen -1 og konsentrasjonen k_HCl
    b. Lag et nytt System() objekt kalt pH_Cl, gi den parameteren Cl.
    c. Kjør pH_Cl objektet igjennom pHsolve()-funksjonen
15. Lag et nytt Acid() objekt kalt HF
    a. Sett inn pKa 3.17, ladningen 0 og konsentrasjonen k_HF
    b. Lag et nytt System() objekt kalt pH_HF, gi den parameteren HF
    c. Kjør pH_HF objektet igjennom pHsolve()-funksjonen
16. Skriv ut pH_Na, pH_Cl og pH_HF
    a. Rund av til andre desimal med round()-funksjonen

Programkode:

```
# Importer pHcalc
from pHcalc.pHcalc import Acid, Neutral, System

# Skriv inn konsentrasjonene
k_NaOH = float(input("Skriv inn konsentrasjonen til NaOH i M: "))
k_HCl = float(input("Skriv inn konsentrasjonen til HCl i M: "))
k_HF = float(input("Skriv inn konsentrasjonen til HF i M: "))

# Definer Neutral objektet til Na
Na = Neutral(charge=1, conc=k_NaOH)
pH_Na = System(Na)
pH_Na.pHsolve()

# Definer Neutral objektet til Cl
Cl = Neutral(charge=-1, conc=k_HCl)
pH_Cl = System(Cl)
pH_Cl.pHsolve()

# Definer Acid objektet til HF
HF = Acid(pKa=3.17, charge=0, conc=k_HF)
pH_HF = System(HF)
pH_HF.pHsolve()

# Skriv ut resultatene:
print("pH-verdi for NaOH:", round(pH_Na.pH, 2), ", pH-verdi for HCl:",
round(pH_Cl.pH, 2), ", pH-verdi for HF:", round(pH_HF.pH, 2))
```

Programkode A.6.4: pH-Beregninger b)

Resultat:

```
Skriv inn konsentrasjonen til NaOH i M: 0.1
Skriv inn konsentrasjonen til HCl i M: 0.1
Skriv inn konsentrasjonen til HF i M: 0.1
pH-verdi for NaOH: 13.0 , pH-verdi for HCl: 1.0 , pH-verdi for HF: 2.1
```

Resultat A.6.4: pH-Beregninger b)

7 Statistikk

«Statistikk handler om samling, organisering og analyse av eksperimentelle data.» (Haraldsrud, 2021, s. 43)

I dette delkapittelet er det forsøk å samle oppgaver hvor man benytter programmering til å utføre statistiske operasjoner i kjemifaget.

7.1 Resultat fra kromatografi

Vi har ønsker å undersøke koffeininnholdet til en varm kopp te ved bruk av væskekromatografi. Vi pipetterer ut noen mL fra tekoppen, filtrerer teen og fortynner løsningen. Deretter bruker vi væskekromatografi (HPLC) for å finne konsentrasjonen. Vi gjør dette ti ganger og får resultatet i tabell 7.1.

Injeksjon	Konsentrasjon (mg/mL)
1	245
2	272
3	252
4	264
5	261
6	272
7	255
8	260
9	268
10	259

Tabell A.7.1: Resultat fra væskekromatografi

7.1.1 Gjennomsnitt

Oppgave: Lag et program som regner ut gjennomsnittet av resultatet, både manuelt og med bruk av NumPy.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: Først legger vi dataen inn i en liste. For å regne ut gjennomsnittet manuelt må vi lage en funksjon som legger sammen alle verdiene og deler den på seg selv. Dette kan vi gjøre

ved bruk av en løkke som plusser alle verdiene sammen, og etter løkken er ferdig deler den summen på antallet verdier i listen. For å regne ut gjennomsnittet med bruk av NumPy bruker vi funksjonen mean().

Algoritme:

1. Lag en liste «koffein» med verdiene fra resultatet
2. Definer en funksjon gjennomsnitt(liste)
    a. Lag en variabel for summen
    b. Lag en for-løkke som gjentar seg selv for hvert element i listen
        i. Gjør at for-løkken adderer verdien for hvert resultat i listen inn i summen
    c. Skriv en linje hvor en variabel for snittet blir lik summen delt på len(liste)
    d. gjør at funksjonen returnerer variabelen for snittet
3. Skriv en linje hvor en variabel kaller på funksjonen gjennomsnitt(koffein)
4. Skriv ut resultatet

NumPy:

1. Importer numpy som np
2. Lag en liste «koffein» med verdiene fra resultatet
3. Skriv en linje hvor en variabel kaller på funksjonen mean(koffein)
4. Skriv ut resultatet

Programkode:

```
# Importer NumPy for NumPy delen
import numpy as np

# Lag en liste for resultatene
koffein = [245, 272, 252, 264, 261, 272, 255, 260, 268, 259]

# Definer en funksjon gjennomsnitt
def gjennomsnitt(data):
    # Lag en variabel for summen
    summen = 0

    # Lag en for-løkke som legger adderer alle resultatene sammen i summen
    for element in data:
        summen += element

    # Regn ut snittet med bruk av summen
    snitt = summen/len(data)

    # Returner snittet
    return snitt

# Kall funksjonen og skriv ut resultatet
snitt = gjennomsnitt(koffein)
print("Gjennomsnitt:", snitt, "mg/mL (manuelt)")

# Numpy: Bruk mean() funksjonen til å regne ut snittet at listen
np_snitt = np.mean(koffein)
print("Gjennomsnitt:", np_snitt, "mg/mL (NumPy)")
```

Programkode A.7.1: Kromatografi, Gjennomsnitt

```
Gjennomsnitt: 260.8 mg/mL (manuelt)
Gjennomsnitt: 260.8 mg/mL (NumPy)
```

Resultat A.7.1: Kromatografi, Gjennomsnitt

7.1.2 Varians og standardavvik

Oppgave: Lag et program som regner ut gjennomsnittet, variansen og standardavvik til koffeindatasettet ved bruk av NumPy, for å regne ut hvor stor spredning det er datasettet.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: Importer NumPy. Lag listen for koffeinresultatene. NumPy har funksjonene mean() for gjennomsnitt, var() for varians og std() for standardavvik. Skriv ut resultatet, men bruk funksjonen round() for variansen og standardavviket for å runde ned til en desimal.

Algoritme:

1. Importer numpy som np
2. Lag listen koffein for datasettet
3. Lag en variabel snitt satt lik np.mean(koffein)
4. Lag en variabel varians satt lik np.var(koffein)
5. Lag en variabel standardavvik satt lik np.std(koffein)
6. Skriv ut resultatet, avrund varians og standardavvik ned til en desimal.

Programkode:

```
# Importer NumPy
import numpy as np

# Lag listen for koffeindatasettet
koffein = [245, 272, 252, 264, 261, 272, 255, 260, 268, 259]

# Lag variabler og knytt de mot NumPy-funksjonene
snitt = np.mean(koffein)
varians = np.var(koffein)
standardavvik = np.std(koffein)

# Skriv ut resultatene
print("Gjennomsnitt:", snitt, "mg/mL")
print("Varians:", round(varians, 1), "mg/mL")
print("Standardavvik:", round(standardavvik, 1), "mg/mL")
```

Programkode A.7.2: Kromatografi, Varians og standardavvik

Resultat:

```
Gjennomsnitt: 260.8 mg/mL
Varians: 67.8 mg/mL
Standardavvik: 8.2 mg/mL
```

Resultat A.7.2: Kromatografi, Varians of standardavvik

7.2 Grafisk framstilling av statistisk spredning:

Oppgave: Vi skal konstruere en standardkurve for magnesiumkonsentrasjonen i en vannprøve. Vi bruker en serie på 0.2, 0.3, 0.4, 0.5 og 0.6 µg/mL Mg^{2+} som vi analyserer tre ganger hver med et flammeatomabsorpsjonsspektrofotometer. Da har vi tre målinger for

absorpsjon per konsentrasjon. Plott disse målingene og usikkerheten med disse målingene ved bruk av NumPy.

Oppgaven og løsningen er hentet fra «Programmering og modellering for kjemikere» (Haraldsrud, 2021).

Løsning: Først må vi importere Matplotlib og NumPy. Deretter må vi skrive inn dataen inn i lister, først en for konsentrasjonen og en for absorbans. Lag må vi lage en liste som samler standardavviket og en liste som samler gjennomsnittet av absorpsjonsmålingene, og lage en løkke som regner ut dette ved bruk av NumPy sin std()- og mean()-funksjon. Deretter plottet vi ut figuren ved bruk av Matplotlib sin funksjon errorbar() som plotter dataen inn i en figur med usikkerhetsstolper.

Algoritme:

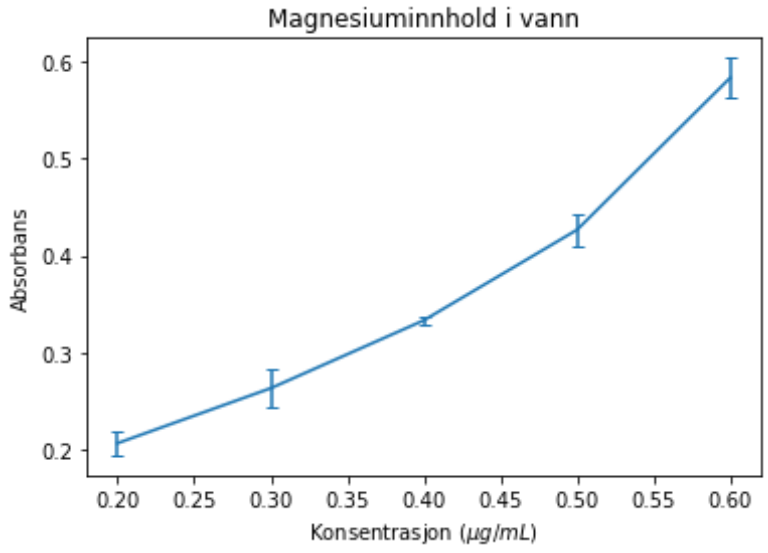
7. Importer matplotlib.pyplot som plt, og numpy som np
8. Lag en liste for konsentrasjonen og absorbans
9. Lag så en tom liste for standardavviket og gjennomsnittet
10. Skriv en for-løkke som utfører std() og mean() på hvert element i absorbanslisten
    a. Bruk append()-funksjonen for å legge resultatet inn i listen for standardavvik og gjennomsnittet
11. Plott ut figuren med bruk av plt.errorbar() funksjonen
    a. Legg til deskriptive funksjoner for å presentere funksjonen bedre
12. Vis figuren med bruk av plt.show()

Programkode:

```
# Importer Matplotlib og NumPy
import matplotlib.pyplot as plt
import numpy as np

# Lag liste for målignene
konsentrasjon = [0.2, 0.3, 0.4, 0.5, 0.6]
absorbans = [[0.21, 0.22, 0.19], [0.26, 0.29, 0.24], [0.33, 0.33, 0.34],
[0.41, 0.42, 0.45], [0.56, 0.61, 0.58]]

# Lag en tom liste for standardavviket og gjennomsnittet
standardavvik = []
snitt = []

# Lag en for-løkke som regner ut standardavviket og gjennsomsnittet for
målingene i absorbans
for element in absorbans:
    standardavvik.append(np.std(element))
    snitt.append(np.mean(element))

# Plott ut figuren med usikkerhetsstoler
plt.errorbar(konsentrasjon, snitt, yerr = standardavvik, capsize=3)
plt.errorbar
plt.title("Magnesiuminnhold i vann")
plt.xlabel("Konsentrasjon ($\mu g/mL$)")
plt.ylabel("Absorbans")
plt.show()
```

Programkode A.7.3: Grafisk fremstilling av statistisk spredning

Resultat:



Resultat A.7.3: Grafisk fremstilling av statistisk spredning

Vedlegg B

1. Innledning

Hensikten med dette dokumentet er å spesifisere grunnlaget til bachelor oppgaven og fungere som et utgangspunkt for oppgaven. Hensikten med oppgaven er å finne ut hvilke oppgaver fra kjemi i skolen som egner seg best til programmering for å øke dybdeforståelse, engasjement og kreativitet hos elever, og anbefalte fremgangsmåter for implementasjon av disse.

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Problem med	Å anvelde programmering i kjemi
berører	Lærere i kjemi i videregående skole
som resultatet av dette	Vil de ikke kunne bruke programmering i kjemi til å øke fagforståelsen i kjemi og nå målet for å øke elevenes digitale ferdigheter i disse fagene.
en vellykket løsning vil	gi lærere ressurser som gjør at de enklere kan bruke programmering i kjemi, selv med lite tidligere kompetanse i programmering.

2.2 Produktsammendrag

For	Lærere i kjemi i videregående skole
som	Har behov for hjelpemidler når det kommer til å bruke programmering i kjemi på videregående skole.
produktet navngitt	*"ProKje"*
som	Er en samling av oppgaver i kjemi hvor programmering og digitale verktøy er brukt for å løse oppgavene, samt løsningsforslag.

I motsetning til	Eksisterende oppgaver som ikke benytter programmering i noen stor grad
Har vårt produkt	Som mål å gi lærere enkle ressurser for å bruke programmering i kjemi slik at elevene blir mer engasjerte og forberedt på å bruke framtidige digitale verktøy.

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
NTNU-AIT	NTNU spiller en aktiv rolle i å tilby lærere etterutdanning i programmering slik at de skaffer kompetanse til å undervise i sine fag på en måte som tar i bruk programmering og digitale hjelpemidler. NTNU vil derfor ha evne til å kunne evaluere kvaliteten på resultatet av prosjektet og vil ha nytte av å kunne tilby lærere ressurser av denne typen.	*Evaluering*

3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Lærere	*Sluttbrukerne av produktet vil være lærere i kjemi på videregående skole.*	*Brukere*	*NTNU*

3.3 Brukermiljøet

Produktet er blir laget med tanke på å bli brukt av lærere i kjemi på videregående skole som et tillegg til undervisning, slik at lærere kan lettere knytte programmering opp til kjemi.

3.4 Sammendrag av brukernes behov

Behov	Prioritet	Påvikrer	Dagens løsning	Foreslått løsning
Algoritmer og programkode	Høy	Kjemifag	Tradisjonelle løsninger	Algoritmer og programkode med løsningsforslag

3.5 Alternativer til vårt produkt

Det eksisterer alternativer som passer til den berørte gruppen til forskjellig grad. Det finnes blant annet kurs i programmering som kan gi god nok kompetanse til å lage egne oppgaver i undervisning.

Det finnes også flere nettsteder som er laget for å skape grunnleggende kompetanse i programmering. Disse er mer egnet for å lære programmering i sin helhet, og er i liten grad egnet for å lære å tilpasse programmering inn i undervisning.

4. Produktoversikt

4.1 Produktets rolle i brukermiljøet

Rollen til produktet vil være å være et hjelpemiddel som lærere kan finne fram ved behov og vil derfor være mest praktisk å legge til et annet større system/en større database som lærere kan få tilgang til.

4.2 Forutsetninger og avhengigheter

Det forutses at det kan komme endringer som krever at prosjektets visjon må/kan endres. I disse tilfeller vil endringene bli diskutert med oppdragsgiver og vil ved godkjennelse bli beskrevet i en oppdatert variant av dette dokumentet.

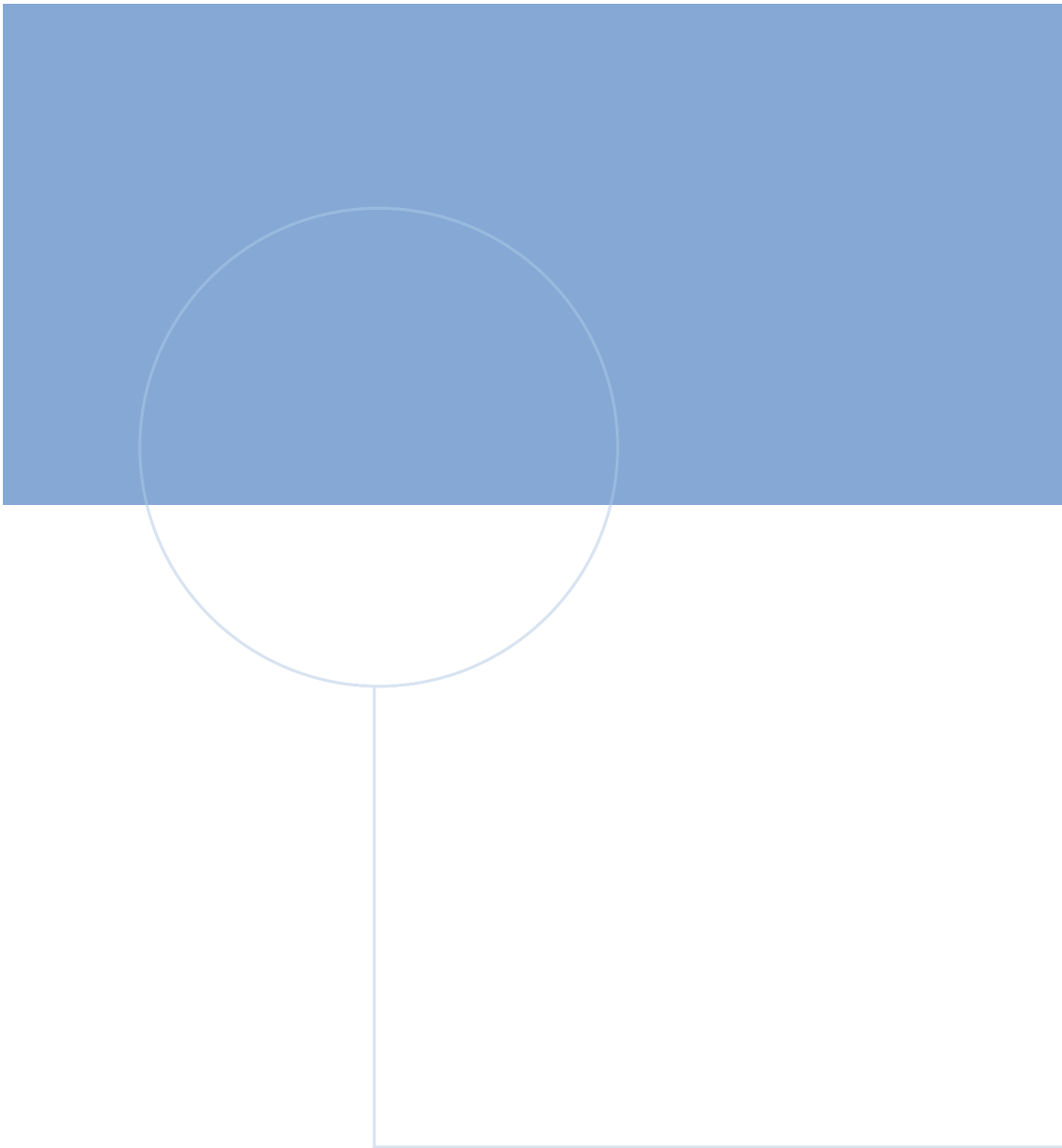
5. Produktets funksjonelle egenskaper

Sluttproduktet vil være et produkt av et forskningsarbeid, et dokument som skal oppfylle visse krav:

- Oppgavene må være beskrevet med fremgangsmåte og løsningsforslag
- Oppgavene må være relevante for fagstoffet
- Oppgavene burde gi et resultat (en output)
- Oppgavene burde bidra til å øke kreativitet og engasjement for elever.
- Oppgavene skal bidra til bedre dybdeforståelse.

6. Ikke-funksjonelle egenskaper og andre krav

Prosjektet skal leveres med en fullstendig rapport som skal begrunne og forklare produktet.



NTNU

Kunnskap for en bedre verden