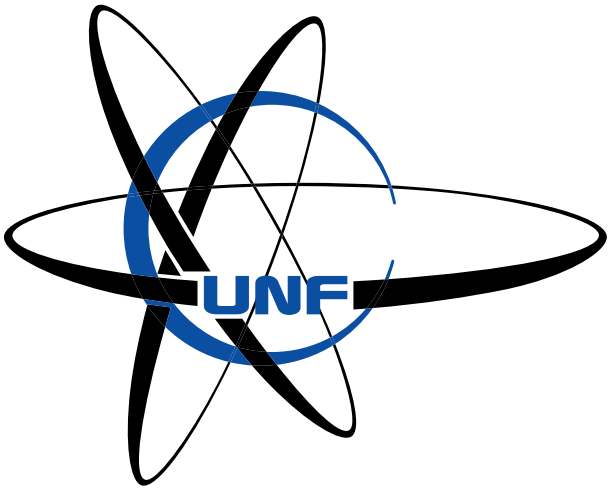


UNF Naturfagsweekend 2023

UNF København

Faglige:

Marie Stuhr Kaltoft	mark@unf.dk
Erik Søndergård Gimsing	esg@unf.dk
Rasmus Frigaard Lemvig	rle@unf.dk
Anders Bærentzen	nalb@unf.dk
Ask Kildelund Rosenkilde	akr@unf.dk
Robert Garbrecht Larsen	roe@unf.dk
Mette Thorup	mkth@unf.dk
Lin Bigom-Eriksen	libe@unf.dk
Knut Ibæk Topp Lindenhoff	knut@unf.dk
Sofie Krogsgaard	skr@unf.dk
Catarina Nampumee Maretti	cnm@unf.dk
Rasmus Peter Bjørn Whitehorn	rabw@unf.dk
Louie Ray Eistrup Juhl	loue@unf.dk
Jonas Bamse Andersen	joba@unf.dk
Ria Dole	riad@unf.dk
Lillian Sun Liang	lisl@unf.dk
Erik Scheel-Whitte	esw@unf.dk
Sana Asadighafari	sana@unf.dk

Ungdommens Naturvidenskabelige Forening

Kompendium til UNF Naturfagsweekend 2023

Kompendiet er skrevet af Marie Stuhr Kaltoft, Erik Søndergård Gimsing, Rasmus Frigaard Lemvig, Anders Bærentzen, Ask Kildelund Rosenkilde, Robert Garbrecht Larsen, Mette Thorup, Lin Bigom-Eriksen, Knut Ibæk Topp Lindenhoff, Sofie Krogsgaard, Catarina Nampumee Maretti, Rasmus Peter Bjørn Whitehorn, Louie Ray Eistrup Juhl, Jonas Bamse Andersen, Ria Dole, Lillian Sun Liang, Erik Scheel-Whitte og Sana Asadighafari. Kompendiet er trykt i marts 2023, og teksten er copyright © 2023 af UNF og forfatterne. Gengivelse med kildehenvisning tilladt.

Layout: Esben Skovhus Ditlefsen på forarbejde af Niels Jakob Søe Loft og Mick Althoff Kristensen.

Opsætning/TEXnisk ansvarlig: Morten Raahauge Bastholm

Indhold

Introduktion

Velkommen til det faglige kompendium! Kompendiet er bygget op af seks kapitler, som dækker de seks forløb, der er på campen. Hvert kapitel er udarbejdet af de faglige teams for hvert område. De samme teams står også for undervisningen på campen, hvor kompendiet benyttes som det primære undervisningsmateriale. Kompendiet er dog også velegnet til selvstudie. Alle os faglige ønsker jer god fornøjelse med campen!

Kapitel 1

Matematik

1.1 Mængdelære

Mængder

Mængdelære er læren om mængder, som er dens mest basale objekt. En mængde er lige præcis hvad man ville forvente:

Definition 1.1.1. En *mængde* er en samling af objekter, denne samling skrives op med tuborg-parenteser "$\{\cdots\}$" som i følgende eksempler:

$$A = \{1, 2, 3, 4\}$$
$$B = \{\text{Kridt},\ \text{Te},\ \text{Kaffe}\}$$
$$\mathbb{N} = \{1, 2, 3, \dots\}$$

Mængder kan også være uendelige, som f.eks. mængden $\mathbb{N}$. Her skal prikkerne forstås som at sige, at mønsteret fortsætter. En mængde kan ikke indeholde to kopier af det samme element, og rækkefølgen er underordnet,

$$\{1, 1, 2\} = \{1, 2\} = \{2, 1\}.$$

Derudover behøver en mængde ikke at indeholde nogen elementer. Mængden uden elementer kaldes den tomme mængde

$$\emptyset = \{\}.$$

Man skal altså lidt forestille sig en mængde som en slags abstrakt "pose", der kan ligge ting i. De ting kalder vi elementer.

Definition 1.1.2. Et *element* er et objekt, der er indholdt i en mængde. Vi skriver det med et $\in$ som i følgende eksempel:

$$\text{Kridt} \in \{\text{Kridt},\ \text{Te},\ \text{Kaffe}\}$$

Vi har nogle mængder, som er kendte nok til at have deres eget specielle bogstav (og som vi måske kommer til at bruge ved et uheld). I har dem derfor her:

Definition 1.1.3. Følgende mængder er uendelige talmængder

$$\mathbb{N} = \{1, 2, 3, \dots\}$$
$$\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$$
$$\mathbb{Q} = \{\text{Alle brøker}\}$$
$$\mathbb{R} = \{\text{Alle tal}\}$$

De sidste to er lidt snyd, men siger bare, at $\mathbb{Q}$ er mængden af alle brøker, og $\mathbb{R}$ er mængden af alle tal.

Eksempel 1.1.4. Mængden af matematik-undervisere på Naturfagsweekend er

$$A = \{\text{Erik},\ \text{Marie},\ \text{Rasmus}\}.$$

Mængden af emner på Naturfagsweekend er

$$F = \{\text{BIOLOGI},\ \text{DATALOGI},\ \text{FYSIK},\ \text{MEDICIN},\ \text{KEMI},\ \text{MATEMATIK}\}.$$

Mængden af lige heltal mellem 0 og 10 inklusiv er

$$B = \{0, 2, 4, 6, 8, 10\}.$$

∘

For at gøre mere komplicerede mængder lettere at skrive op, bruger man ofte følgende notation:

Definition 1.1.5. *Mængdebyggernotation* er en notation for at skrive mængder op på formen "$\{\cdots \mid \cdots\}$", hvor man blot beskriver, hvordan elementerne skal se ud, og hvilke regler der skal gælde for dem. For eksempel er

$$\{n \in \mathbb{N} \mid n \text{ er lige}\}$$

mængden af alle lige tal, og man læser ovenstående som "mængden af alle heltal n, hvorom der gælder, at n er lige". Alt før $\mid$ beskriver elementerne, $\mid$ står for "hvorom der gælder", og det, der står efter $\mid$, er nogle regler, som elementerne skal overholde.

Eksempel 1.1.6. Det er velkendt, at hvis vi lader F være mængden af emner på Naturfagsweekend, så er

$$\{f \in F \mid f \text{ er mega sjovt}\} = \{\text{MATEMATIK}\}$$

Et andet eksempel er

$$\{n \in \{1, 2, 3, 4\} \mid n \text{ er ulige}\}. = \{1, 3\}.$$

Et lidt mere komplekst eksempel kunne være den her måde at skrive de rationale tal $\mathbb{Q}$ (alle brøker) op på:

$$\left\{\frac{a}{b} \mid a \in \mathbb{Z},\ b \in \mathbb{N}\right\}.$$

∘

Definition 1.1.7. *En delmængde* er en mængde, hvorom der gælder, at alle dens elementer ligger i en anden mængde. Hvis A er en delmængde af B, gælder, at alle elementer i A ligger i B, og det skriver vi som

$$A \subseteq B.$$

Hvis A ikke er en delmængde af B, skriver vi $\not\subseteq$ i stedet for $\subseteq$.

Eksempel 1.1.8. Her er nogle sande udsagn:

$$\begin{aligned} \{1, 2\} &\subseteq \{1, 2, 3, 4\} \\ \emptyset &\subseteq \{1, 2, 3, 4\} \\ \{1, 2, 3, 4\} &\subseteq \{1, 2, 3, 4\} \\ \{1, 2\} &\not\subseteq \{2, 3, 4\} \end{aligned}$$

∘

Det er værd at nævne en variant af mængde-konceptet her, nemlig en tupel.

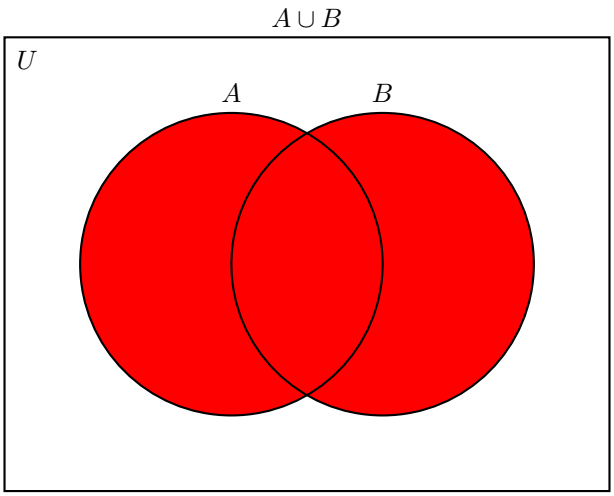
Definition 1.1.9. En *tupel* er en mængde, hvor rækkefølgen af elementerne er fast. Dette skrives med parenteser "$(\cdots)$", som med koordinater. Det vil altså sige at

$$(a, b) \neq (b, a)$$

Mængdeoperationer

Det første, man som matematiker ønsker sig efter at have fundet et nyt matematisk objekt, er måder at kombinere kendte eksempler til nye objekter af samme type. Vi vil derfor gennemgå nogle operationer, der tager to mængder og giver en ny mængde.

Definition 1.1.10. *Foreningsmængden* af to mængder A og B er mængden $A \cup B$, som indeholder både alle elementer fra A og alle fra B. Man kan tegne dette med et Venn-diagram:



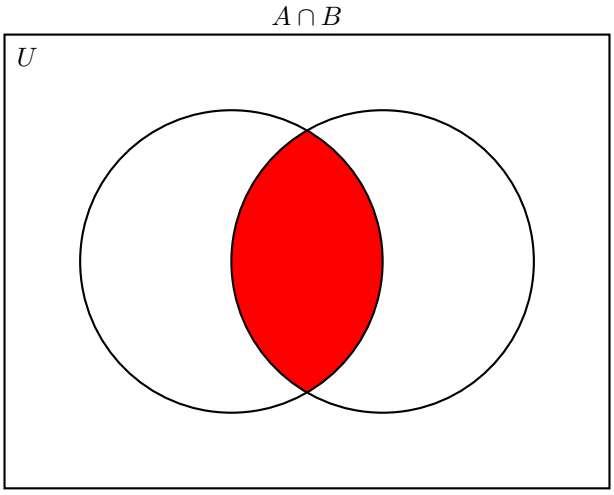
Arealet inde i cirklen A og B repræsenterer deres elementer, og det farvelagte område er deres forening.

Eksempel 1.1.11. En forening af to mængder findes ved at skrive en mængde med alle elementer fra begge ned, dog uden at gentage elementer:

$$\{1, 2, 3\} \cup \{2, 3, 4\} = \{1, 2, 3, 4\}$$

∘

Definition 1.1.12. *Fællesmængden* af to mængder A og B er mængden $A \cap B$, som indeholder de elementer, som ligger i både A og B. Man kan tegne dette med et Venn-diagram:



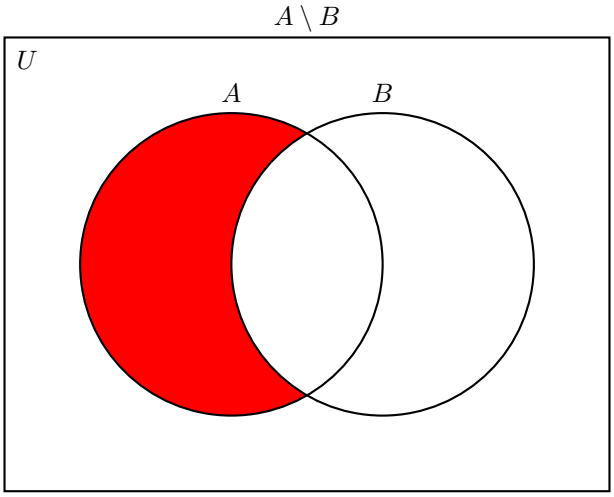
Arealet inde i cirklen A og B repræsenterer deres elementer, og det farvelagte område er deres fællesmængde. Man kalder også fællesmængder for snit.

Eksempel 1.1.13. En fællesmængde af to mængder findes ved at skrive en mængde med alle de elementer, de har tilfælles, ned:

$$\{1, 2, 3\} \cap \{2, 3, 4\} = \{2, 3\}$$

$\circ$

Definition 1.1.14. *Differensmængden* af to mængder A og B er en mængde $A \setminus B$, som indholder de elementer, som ligger i A, men ikke i B. Man kan tegne dette med et Venn-diagram:



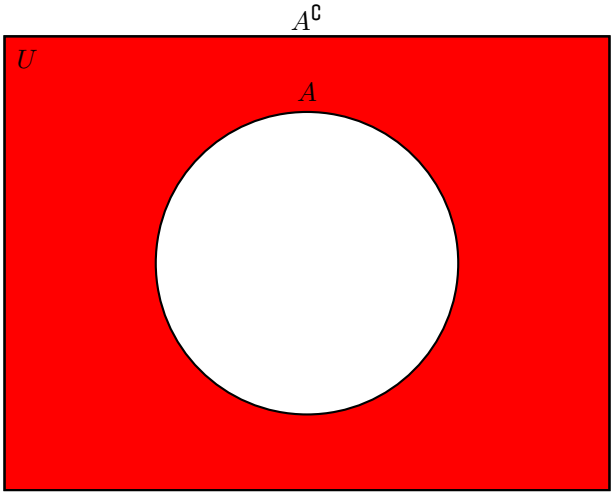
Arealet inde i cirklen A og B repræsenterer deres elementer, og det farvelagte område er deres differens.

Eksempel 1.1.15. En differens af to mængder findes ved at skrive en mængde med alle elementerne fra den første bortset fra dem i den anden:

$$\{1, 2, 3\} \setminus \{2, 3, 4\} = \{1\}$$

$\circ$

Definition 1.1.16. *Komplementet* $A^{\complement}$ af en mængde A i en større mængde U er alle elementer i U, som ikke ligger i A. Ofte kan man gætte ud fra konteksten hvilken mængde U, der er tale om. Man kan tegne dette med et Venn-diagram:



Arealet inde i cirklen A repræsenterer dens elementer, og det farvelagte område er dens komplement.

Eksempel 1.1.17. At beregne et komplement er meget ligesom at beregne differensmængden. Hvis vi har $A = \{2, 3, 4\}$ og $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, så har vi, at

$$A^{\complement} = \{1, 5, 6, 7, 8, 9, 10\}.$$

$\circ$

Opgaver

- **Opgave 1.1.1:**
Fortæl en ven, hvad svaret til følgende spørgsmål er:

  1) Hvad er en mængde?

  2) Hvad er et element?

  3) Hvad er en delmængde?

- **Opgave 1.1.2:**
Svar på følgende spørgsmål:

  1) Er du et element i mængden af deltagere på Naturfagsweekend?

  2) Hvad med Rasmus, er han et element i mængden af deltagere?

  3) Er mængden af deltagere på matematik-forløbet en delmængde af mængden af deltager på Naturfagsweekend?

●● **Opgave 1.1.3:**
Skriv følgende mængder ned:

  1) Mængden af lige tal mellem 10 og 20.

  2) Mængden af folk, der bor i dit hjem.

  3) Mængden af matematik-undervisere med bogstavet "a" i deres navn.

●●● **Opgave 1.1.4:**
Omskriv mængderne fra sidste opgave til mængdebygger-notation. Det ligegyldigt, hvordan du gør det, bare det beskriver den rigtige mængde:

  1) Opskriv mængden af lige tal mellem 10 og 20 med mængdebygger-notation.

  2) Opskriv mængden af folk der bor i dit hjem med mængdebygger-notation.

  3) Opskriv mængden af matematik-undervisere med bogstavet "a" i deres navn med mængdebygger-notation.

●● **Opgave 1.1.5:**
Afgør, om følgende udsagn er sande:

  1) $7 \in \mathbb{N}$

  2) $7 \in \mathbb{Z}$

  3) $7 \in \{2n | n \in \mathbb{Z}\}$

  4) $7 \subseteq \mathbb{N}$

  5) $\{7\} \subseteq \mathbb{N}$

  6) $\emptyset = \{\}$

  7) $\{\emptyset\} = \{\}$

- **Opgave 1.1.6:**
Lad A være mængden af deltagere på Naturfagsweekend og B mængden af arrangører på Naturfagsweekend. Bestem følgende:

  1) Hvad er $A \cup B$?

  2) Hvad er $A \cap B$?

  3) Hvad er $A \setminus B$?

  4) Hvad er $B \setminus A$?

- **Opgave 1.1.7:**
Lad A være mængden af dage, det regner i løbet af et år, og B mængden af dage, solen skinner på et år. Bestem følgende:

  1) Hvad er $A \cup B$?

2) Hvad er $A \cap B$?

3) Hvad er $A \setminus B$?

4) Hvad er $B \setminus A$?

●● **Opgave 1.1.8:**
Lad $A = \{a, b, c, d\}$ og $B = \{b, d, f\}$. Bestem følgende:

1) Hvad er $A \cup B$?

2) Hvad er $A \cap B$?

3) Hvad er $A \setminus B$?

4) Hvad er $B \setminus A$?

●● **Opgave 1.1.9:**
Lad A være mængden af lige tal, hvad er $A^{\complement}$ i $\mathbb{Z}$?

●●● **Opgave 1.1.10:**
Lad A være mængden af lige tal og B mængden af ulige tal. Bestem følgende:

1) Hvad er $A \cup B$?

2) Hvad er $A \cap B$?

3) Hvad er $A \setminus B$?

4) Hvad er $B \setminus A$?

1.2 Forsmag på ekstraemner

Sandsynlighedsteori

Hvad er sandsynlighed? Hvordan regner man med sandsynligheder, og hvordan bruges det? Vi skal i dette forløb se på, hvordan man matematisk kan beskrive tilfældighed. Undervejs ser vi på konkrete anvendelser af teorien, heriblandt i terningkast og andre spil. Til slut ser vi på en anvendelse i epidemier.

Induktion

"Sætning": Alle kaniner har samme farve.
"Bevis": Det er klart, at der er et endeligt antal kaniner i verden. Lad antallet af kaniner være N. Sætningen er altså bevist, hvis vi kan vise, at for alle $n \leq N$ er alle kaniner i en mængde med n kaniner samme farve.

I en mængde med $n = 1$ kanin er det klart, at alle kaniner har samme farve.

Antag nu, at alle mængder med m kaniner opfylder, at alle kaniner har samme farve. Betragt en mængde med $m + 1$ kaniner. Sæt en kanin til side, da har mængden med resten af kaninerne alle samme farve. Sæt kaninen tilbage og sæt en anden kanin til side. Igen har alle de resterende kaniner samme farve. Da vi sætter kaninen tilbage, må vi derfor have, at alle $m + 1$ kaniner har samme farve.

1.3 Sandsynlighedsteori - en anvendelse

Introduktion til sandsynlighed

I denne sektion skal vi formalisere (dvs. sætte matematikbegreber på) idéen bag sandsynligheder. Tænk f.eks. på en sekssidet terning. Kaster vi den én gang, får vi en af seks mulige udfald, nemlig 1, 2, 3, 4, 5 eller 6. Lad os gøre dette til en definition.

Definition 1.3.1. *Udfaldsrummet* for et eksperiment er en mængde. Elementerne i denne mængde kaldes *udfald*. En delmængde af udfaldsrummet kaldes en *begivenhed*.

Rent formelt er et udfaldsrum altså bare sandsynlighedsteoretikerens ord for en bestemt mængde. Vi vil som regel betegne udfaldsrummet med S.

Eksempel 1.3.2. Er vores eksperiment et kast med en sekssidet terning, er udfaldsrummet lig $S = \{1, 2, 3, 4, 5, 6\}$. Vi ser, at der med ét kast er 6 mulige udfald. 3 er et udfald, svarende til at slå 3 på terningen. Begivenheden $\{1, 4\}$ svarer til, at vi slår enten 1 eller 4. ∘

Eksempel 1.3.3. Kaster vi to terninger, er udfaldsrummet lig $S = \{(1,1), (1,2), ..., (1,6), (2,1), (2,2), ..., (6,6)\}$. Vi ser, at der er 36 udfald, da der til hvert udfald på første terning er 6 udfald på den anden terning. Begivenheden $\{(1,3), (2,2), (4,5)\}$ svarer til, at vi enten slår 1 og 3, 2 begge gange eller 4 og 5. ∘

Definition 1.3.4. Lad S være udfaldsrummet for et eksperiment. En sandsynlighed P er en funktion, der til enhver delmængde $A \subseteq S$ tildeler et tal $P(A)$ mellem 0 og 1. En sandsynlighed skal opfylde to regler:

1. $P(\emptyset) = 0$ og $P(S) = 1$.
2. Hvis A og B er begivenheder med $A \cap B = \emptyset$ er $P(A \cup B) = P(A) + P(B)$.

Det er vigtigt at forstå, hvorfor definitionen er valgt, som den er. Eksperimentet har altid et udfald. Begivenheden $\emptyset$, svarende til at intet finder sted, må derfor have sandsynlighed 0. S er begivenheden, at noget indtræffer, og derfor er $P(S) = 1$. Regel nummer to illustreres bedst med et eksempel.

Eksempel 1.3.5. Ser vi igen på terningekastet, ser vi, at

$$P(\{1\}) = P(\{2\}) = P(\{3\}) = P(\{4\}) = P(\{5\}) = P(\{6\}) = \frac{1}{6}.$$

Her antager vi, at terningen er fair. Vi kan bestemme sandsynligheden for alle andre begivenheder ud fra disse 6 sandsynligheder. Se f.eks. på begivenheden $\{1, 2, 3\}$, altså at vi slår 1, 2 eller 3. Vi kan skrive $\{1, 2, 3\} = \{1\} \cup \{2\} \cup \{3\}$, og de tre mængder har intet overlap, så vi får

$$P(\{1, 2, 3\}) = P(\{1\}) + P(\{2\}) + P(\{3\}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

hvilket vi også ville forvente. ∘

- **Opgave 1.3.1:**
  Vi kaster med en otte-sidet terning.
  1) Hvad er udfaldsrummet for eksperimentet?
  2) Hvad er sandsynligheden for at slå 1 eller 2?
  3) Hvad er sandsynligheden for at slå et ulige tal?

- **Opgave 1.3.2:**
  Du slår plat eller krone fire gange.
  1) Hvad er udfaldsrummet?

2) Hvad er sandsynligheden for at få kun plat?

3) Hvad er sandsynligheden for at få netop to krone?

4) Hvad er sandsynligheden for at få netop én plat?

For at gøre det nemmere for os selv at specificere et udfaldsrum med en tilhørende sandsynlighed, indfører vi følgende nyttige begreb.

Definition 1.3.6. Et udfaldsrum S med en sandsynlighed P skriver vi som (S, P), og vi kalder (S, P) et *sandsynlighedsrum*.

Eksempel 1.3.7. Vi slår med en sekssidet terning to gange. Hvad er sandsynligheden for at slå to ens? Ser vi på udfaldsrummet fra tidligere, kan vi se, at man kan slå to ens på 6 måder. Som begivenhed kan vi skrive $A = \{(1,1),(2,2),(3,3),(4,4),(5,5),(6,6)\}$. Da alle udfald er lige sandsynlige, bliver sandsynligheden

$$P(A) = \frac{6}{36} = \frac{1}{6}.$$

Vi kan fortolke det som, at vi i ca. 1 ud af 6 sådanne eksperimenter vil forvente at slå to ens. ∘

●● **Opgave 1.3.3:**
Antag, at vi har et sandsynlighedsrum (S, P) og en begivenhed A, der opfylder, at $P(A) = 1$. Er A nødvendigvis lig S? Bevis det eller find et modeksempel.

For at kunne lave mere interessant eksempler, er vi nødt til at udlede nogle resultater fra definitionen af sandsynlighed. Følgende resultat er ganske nyttigt i flere sammenhænge.

Sætning 1.3.8. *Lad (S, P) være et sandsynlighedsrum og $A, B \subseteq S$ to begivenheder. Da gælder følgende:*

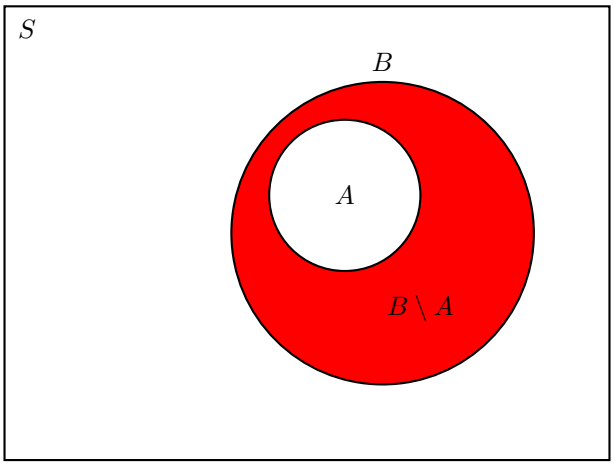
1. $P(A^{\complement}) = 1 - P(A)$.
2. *Hvis $A \subseteq B$ gælder* $P(B \setminus A) = P(B) - P(A)$.
3. *Hvis $A \subseteq B$ gælder* $P(A) \leq P(B)$.
4. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Bevis. 1. Bemærk, at $S = A \cup A^{\complement}$ og $A \cap A^{\complement} = \emptyset$ for alle begivenheder A. Regel 1 og 2 giver nu

$$1 = P(S) = P(A) + P(A^{\complement}).$$

Trækker vi $P(A)$ fra på begge sider af ligningen, står der $1 - P(A) = P(A^{\complement})$ som ønsket.

2. Situationen ser sådan ud i et Venn-diagram:



Vi ser, at vi her kan skrive

$$B = (B \setminus A) \cup A$$

og $(B \setminus A) \cap A = \emptyset$. Dermed giver regel 2:

$$P(B) = P(B \setminus A) + P(A)$$

og ved at trække $P(A)$ fra på begge sider fås

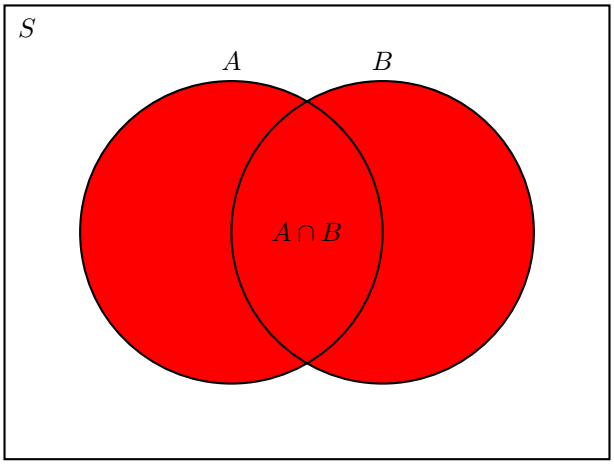
$$P(B \setminus A) = P(B) - P(A).$$

3. Ifølge forrige punkt kan vi her skrive:

$$P(B) = P(A) + P(B \setminus A) \geq P(A)$$

thi sandsynligheder er ikke-negative.

4. Igen tegner vi et Venn-diagram af situationen:



Vi får idéen til at lave følgende opskrivning:

$$A \cup B = A \cup (B \setminus (A \cap B)).$$

Ved at se på figuren, kan vi se, at A og $B \setminus (A \cap B)$ ikke overlapper. Da $A \cap B \subseteq B$ kan vi bruge regel 2 for sandsynligheder kombineret med forrige regel til at få

$$P(A \cup B) = P(A) + P(B \setminus (A \cap B)) = P(A) + P(B) - P(A \cap B).$$

■

Eksempel 1.3.9. Hvis man slår med en sekssidet terning to gange, hvad er sandsynligheden for at få to forskellige? Hvis A betegner begivenheden at slå to ens, så har vi tidligere beregnet $P(A) = 1/6$. Begivenheden at få to forskellige er netop $A^{\complement}$, så sandsynligheden er $P(A^{\complement}) = 1 - P(A) = 1 - 1/6 = 5/6$. ○

- **Opgave 1.3.4:**
  Vi kaster med to fire-sidede terninger.
  1) Hvad er udfaldsrummet for eksperimentet?
  2) Hvad er sandsynligheden for at slå to ens?
  3) Hvad er sandsynligheden for at slå to forskellige?

- **Opgave 1.3.5:**
  Antag, at vi har et sandsynlighedsrum (S, P) og en begivenhed A, der opfylder, at $P(A) = 1$. Vis at $P(A^{\complement}) = 0$.

●● **Opgave 1.3.6:**
Lad (S, P) være et sandsynlighedsrum og A, B to begivenheder med sandsynlighed nul, altså $P(A) = P(B) = 0$. Bevis, at $P(A \cup B) = 0$.

●● **Opgave 1.3.7:**
Du spiller i et lotteri, hvor du kan trække tallene fra 1 til 100. Alle tal har samme sandsynlighed for at blive trukket.

1) Hvad er sandsynligheden for at trække et tal i 3-tabellen?

2) Hvad er sandsynligheden for at trække et tal i 5-tabellen?

3) Hvad er sandsynligheden for at trække et tal, der ligger enten i 3- eller 5-tabellen? [Vink: Brug sætning 1.3.8 punkt 4]

●● **Opgave 1.3.8:**
Antag, at du slår med en 20-sidet terning 10 gange. Lad A være begivenheden, at du slår et tal i 4-tabellen og B begivenheden, at du slår et lige tal. Vis, at $P(A) \leq P(B)$.

Eksempel 1.3.10. Antag, at vi har 23 mennesker. Vi er interesseret i sandsynligheden for, at mindst to af dem har fødselsdag samme dag. Kald denne begivenhed A. Vi antager, at sandsynligheden for at blive født en bestemt dag er den samme for alle dage i året, og at folks fødselsdage er uafhængige af hinanden (vi vender tilbage til, hvad uafhængighed betyder, bare tænk på det som, at folks fødselsdage ikke er relaterede, specielt har vi ingen tvillinger). $A^{\complement}$ er begivenheden, at alle har fødselsdag forskellige dage. Første person har 365 muligheder. Den anden har da 364, den næste 363 osv. Da der er 365^{23} udfald i alt, bliver sandsynligheden

$$P(A^{\complement}) = \frac{365 \cdot 364 \cdots 342}{365^2 3} \approx 0,493$$

Sandsynligheden for, at mindst to har fødselsdag samme dag, er da omtrent $1 - 0.493 = 0.507$, dvs. lidt over 50 %. Dette kaldes for *fødselsdagsproblemet*, som lyder: "hvor mange mennesker skal være samlet for, at der er mindst 50% sandsynlighed for, at to i rummet har fødselsdag samme dag?". Af eksemplet fremgår det, at 23 mennesker er tilstrækkeligt. ∘

●●● **Opgave 1.3.9:**
Antag, at vi har et sandsynlighedsrum (S, P) med begivenheder A og B, der opfylder $P(A) = 4/5$ og $P(B) = 1/2$. Bevis, at $P(A \cap B) \geq 3/10$. [Vink: Brug sætning 1.3.8 punkt 4]

●●● **Opgave 1.3.10:**
For et sandsynlighedsrum (S, P) med to begivenheder A og B definerer vi den *symmetriske differens* $A\Delta B$ til at være $A\Delta B = (A \setminus B) \cup (B \setminus A)$.

1) Tegn $A\Delta B$ i et Venn-diagram.

2) Brug din tegning til at argumentere, hvorfor $P(A\Delta B) = P(A) + P(B) - 2P(A \cap B)$.

3) Vis, at $A \setminus B = A \setminus (A \cap B)$ og at $B \setminus A = B \setminus (A \cap B)$.

4) Brug sætning 1.3.8 punkt 2 til at bevise $P(A\Delta B) = P(A) + P(B) - 2P(A \cap B)$.

Betinget sandsynlighed

Definition 1.3.11. Lad (S, P) være et sandsynlighedsrum. For begivenheder $A, B \subseteq S$ med $P(B) > 0$ defineres den *betingede sandsynlighed for A givet B* som

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

Eksempel 1.3.12. Hvad er sandsynligheden for at slå 1 med en sekssidet terning givet, at vi har slået 1, 2 eller 3? Her er $A = \{1\}$ og $B = \{1, 2, 3\}$. Vi indsætter i formlen for betinget sandsynlighed og beregner:

$$P(A \mid B) = \frac{A \cap B}{P(B)} = \frac{P(\{1\})}{P(\{1,2,3\})} = \frac{1/6}{1/2} = \frac{1}{3}.$$

Altså er sandsynligheden 1/3 for at slå 1, hvis vi ved, at der er slået 1, 2 eller 3. Hvis vi vidste, at der var slået 4 eller 5, ville den betingede sandsynlighed være 0. Begge tilfælde stemmer godt overens med vores intuition. ∘

- **Opgave 1.3.11:**
Din ven kaster med en otte-sidet terning og fortæller dig, at han har slået et lige tal. Hvis kammeraten taler sandt, hvad er sandsynligheden for, at han har slået 2?

- **Opgave 1.3.12:**
Du har et almindeligt kortspil med 52 kort. Hvad er sandsynligheden for at trække en hjerter 2, hvis du ved, at du trækker et hjerter?

- **Opgave 1.3.13:**
Vis, at $P(A \mid A) = 1$ for enhver begivenhed A.

Eksempel 1.3.13. En familie har to børn. Hvad er sandsynligheden for, at begge er piger? Udfaldsrummet ser sådan ud:

$$S = \{(\text{dreng}, \text{dreng}), (\text{pige}, \text{dreng}), (\text{dreng}, \text{pige}), (\text{pige}, \text{pige})\}$$

Hvis vi antager, at alle udfald har samme sandsynlighed, da må sandsynligheden for, at begge er piger, være 1/4. Hvad hvis vi får at vide, at mindst én af de to børn er piger? Lad $A = \{(\text{pige}, \text{pige})\}$. Da er A begivenheden, at begge er piger. Begivenheden, at mindst én er en pige, er $B = \{(\text{pige}, \text{dreng}), (\text{dreng}, \text{pige}), (\text{pige}, \text{pige})\}$. Den betingede sandsynlighed er derfor

$$\begin{aligned} P(A \mid B) &= \frac{P(A \cap B)}{P(B)} = \frac{P(\{(\text{pige}, \text{pige})\})}{P(\{(\text{pige}, \text{dreng}), (\text{dreng}, \text{pige}), (\text{pige}, \text{pige})\})} \\ &= \frac{1/4}{3/4} = \frac{1}{3}. \end{aligned}$$

Det stemmer overens med vores intuition, at sandsynligheden er større end 1/4. Når vi ved, at mindst én af børnene er en pige, udelukker vi en situation, hvor ingen af børnene er piger. ∘

- **Opgave 1.3.14:**
En familie har to børn. Hvad er sandsynligheden for, at begge er piger, hvis vi ved, at den ældste er en pige? Sammenlign med eksempel 1.3.13. Er dit svar overraskende?

En central regel for betingede sandsynligheder er Bayes sætning/regel. Det er op til læseren at bevise sætningen ud fra definitionen af betinget sandsynlighed.

Sætning 1.3.14 (Bayes sætning). *For et sandsynlighedsrum (S, P) med begivenheder $A, B \subseteq S$, der opfylder $P(A) > 0$ og $P(B) > 0$, gælder reglen*

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}.$$

Bayes sætning har interessante konsekvenser, også i den virkelige verden. Et eksempel er den såkaldte *anklagers fejlslutning*, som den følgende opgave omhandler. Vi får et matematisk (og dog virkelighedsnært eksempel) at se i næste afsnit.

●●● **Opgave 1.3.15:**
Der bliver begået et bankrøveri, og røveren slipper væk med nogle penge. Senere samme dag stopper og arresterer politiet en mand, der har en pose med 10.000 kr. på sig. Politiets anklager hævder, at hvis manden er uskyldig, er det meget usandsynligt, at han ville gå rundt med 10.000 kr., og derfor mener anklageren, at manden må stå bag røveriet. Hvad er fejlen i anklagerens argument?

●●● **Opgave 1.3.16:**
Antag, at $P(A) = 1$. Vi vil vise, at $P(A \mid B) = 1$ for alle begivenheder B med $P(B) > 0$.

1) Vis, at $P(B \cap A^{\complement}) = 0$.

2) Vis, at $P(B) = P(B \cap A) + P(B \cap A^{\complement})$.

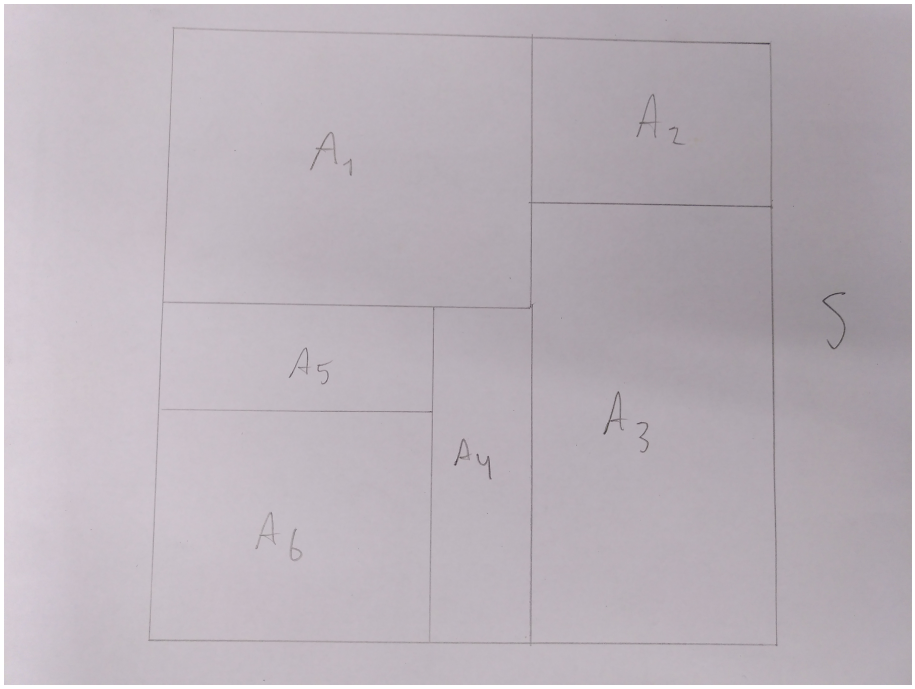
3) Vis, at $P(A \mid B) = 1$.

Loven om total sandsynlighed

Ligesom Bayes sætning er loven om total sandsynlighed et særdeles nyttigt værktøj til udregninger. Lad os starte med at få lidt terminologi på plads.

Definition 1.3.15. Lad S være et udfaldsrum. Lad $A_1, A_2, ..., A_n$ være delmængder af S. Da udgør $A_1, A_2, ..., A_n$ en *partition/opdeling* af S, hvis ingen af $A_i$erne overlapper (dvs. $A_i \cap A_j = \emptyset$ for alle $i \neq j$) og hvis foreningen af $A_1, A_n, ..., A_n$ er lig S.

En illustration af en opdeling er herunder:



Figur 1.1

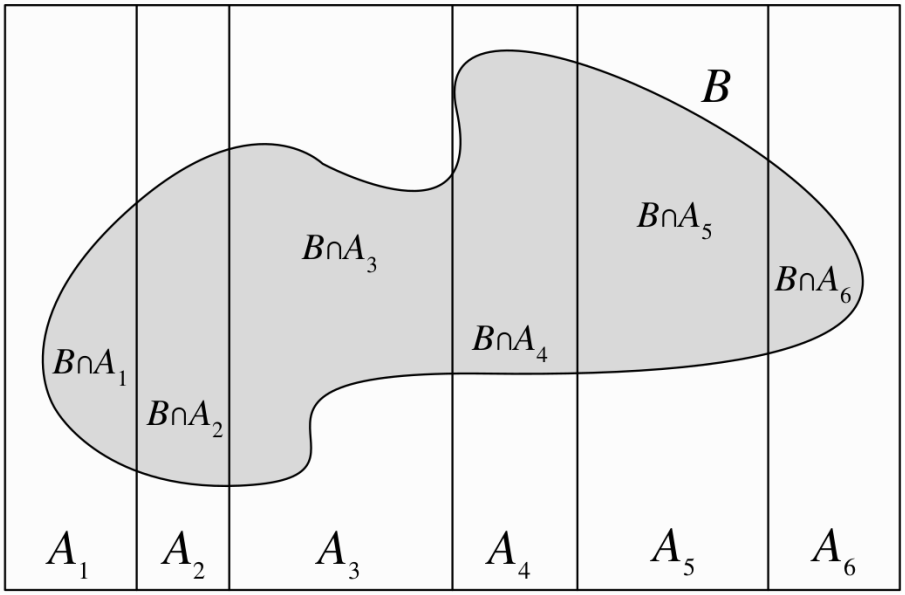
Sætning 1.3.16 (**Loven om total sandsynlighed**). *Lad (S, P) være et sandsynlighedsrum og $A_1, ..., A_n$ en opdeling af S, hvor $P(A_i) > 0$ for alle $i = 1, ..., n$. Da gælder for alle begivenheder B, at*

$$P(B) = P(B \mid A_1) \cdot P(A_1) + P(B \mid A_2) \cdot P(A_2) + \cdots + P(B \mid A_n) \cdot P(A_n).$$

Bevis. Ved at benytte, at $A_1, A_2, ..., A_n$ er en opdeling af S, fås

$$\begin{aligned} B &= B \cap S = B \cap (A_1 \cup A_2 \cup ... \cup A_n) \\ &= (B \cap A_1) \cup (B \cap A_2) \cup ... \cup (B \cap A_n) \end{aligned}$$

og vi ser, at ingen af mængderne $B \cap A_i$ overlapper med hinanden. Dette er illustreret på figuren herunder:



Figur 1.2

Vi bruger nu regel 2 for sandsynligheder og får

$$P(B) = P(B \cap A_1) + P(B \cap A_2) + \cdots + P(B \cap A_n).$$

Vi benytter nu et af matematikeres yndlingstricks. Vi ganger hvert led med 1 på en smart måde. Da ingen af $P(A_i)$ er nul, kan vi dividere og gange med $P(A_i)$ i hvert led i summen og bruge definitionen af betinget sandsynlighed til at få det ønskede:

$$\begin{aligned} P(B) &= \frac{P(B \cap A_1) \cdot P(A_1)}{P(A_1)} + \frac{P(B \cap A_2) \cdot P(A_2)}{P(A_2)} + \cdots + \frac{P(B \cap A_n) \cdot P(A_n)}{P(A_n)} \\ &= P(B \mid A_1) \cdot P(A_1) + P(B \mid A_2) \cdot P(A_2) + \cdots + P(B \mid A_n) \cdot P(A_n) \end{aligned}$$

■

I praksis har vi ofte brug for et specialtilfælde af loven om total sandsynlighed, nemlig hvor opdelingen har to mængder. Dette skriver vi ned som et korollar:

Korollar 1.3.17. *Lad (S, P) være et sandsynlighedsrum og $A \subseteq S$ en begivenhed med $P(A) > 0$. Da gælder for alle begivenheder B, at*

$$P(B) = P(B \mid A) \cdot P(A) + P(B \mid A^{\complement}) \cdot (1 - P(A))$$

Bevis. Fra loven om total sandsynlighed har vi, idet A og S A udgør en opdeling af S, at

$$P(B) = P(B \mid A) \cdot P(A) + P(B \mid A^{\complement}) \cdot P(A^{\complement}).$$

Ifølge sætning 1.3.8 1 er $P(A^{\complement}) = 1 - P(A)$, og resultatet følger nu ved at indsætte dette i ligningen ovenover. ■

Hvis den generelle sætning er svær at huske eller forstå, så er det til det her forløb helt tilstrækkeligt at huske ovenstående korollar. Lad os give et interessant eksempel på, hvordan Bayes regel og loven om total sandsynlighed kan benyttes i kombination.

Eksempel 1.3.18. Lad os antage, at en epidemi har ramt Danmark[1], og på nuværende tidspunkt er 1% af befolkningen smittet. Heldigvis har vi en test, som er 95% præcis. Det skal forstås på den måde, at hvis man er smittet, er der 95% sandsynlighed for,

[1] Ikke så urealistisk en antagelse!

at testen giver et positivt resultat. Tilsvarende, hvis man er rask, er der 95% sandsynlighed for, at testen giver et negativt resultat, dvs. 5% sandsynlighed for et positivt resultat i det tilfælde. Du går ned i det nærmeste testcenter og tager denne test, og den giver et positivt resultat. Hvad er sandsynligheden for, at du er smittet med sygdommen?

Lad os først give begivenhederne nogle navne. A er begivenheden, at du er smittet. B er begivenheden, at din test er positiv. Oplysningerne ovenover er da

$$P(A) = 0,01 \quad \text{og} \quad P(B \mid A) = 0,95.$$

Vi er interesseret i at beregne sandsynligheden $P(A \mid B)$, altså sandsynligheden for, at du er smittet, givet at din test er positiv. Lad os først benytte Bayes regel og få

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} = \frac{0,95 \cdot 0,01}{P(B)}.$$

For at komme videre skal vi have bestemt $P(B)$. Hertil benytter vi loven om total sandsynlighed med inddelingen $A, A^{\complement}$. Vi har da

$$\begin{aligned} P(B) &= P(B \mid A) \cdot P(A) + P(B \mid A^{\complement}) \cdot (1 - P(A)) \\ &= 0,95 \cdot 0,01 + 0,05 \cdot 0,99 = 0,0585. \end{aligned}$$

Indsætter vi i formlen fra før, får vi det endelige svar:

$$P(A \mid B) = \frac{0,95 \cdot 0,01}{0,0585} \approx 0,162.$$

Altså er sandsynligheden ca. 16% for, at du er smittet. Dette er virkelig overraskende for de fleste, når de ser det første gang, heriblandt læger. Dette er grunden til, at man tester flere gange for alvorlige sjældne sygdomme. Hvorfor er sandsynligheden så lav? Tænk på det sådan her. 99% af befolkningen er ikke smittet. Så hvis en stor andel, f.eks. 100.000 mennesker, tog en test, ville 1.000 af dem være smittede, men af de 99.000 raske ville 5%, 4950 mennesker, få en falsk positiv. Af de faktiske smittede ville blot 950 teste positiv. Så langt de fleste positive tests er faktisk falske positive! ○

●●● **Opgave 1.3.17:**
Vi har en pose med ti mønter. Ni af disse er almindelige, men én af dem vægtet og giver altid krone. Vi trækker en mønt fra posen tilfældigt. Vi kaster med mønten syv gange og får krone hver gang. Vi vil bestemme sandsynligheden for, at mønten vi har trukket, er vægtet. Lad A være begivenheden, at mønten giver krone syv gange. Lad B være begivenheden, at mønten er vægtet.

1) Beregn $P(B), P(A \mid B)$ og $P(A \mid B^{\complement})$.

2) Brug loven om total sandsynlighed til at beregne $P(A)$.

3) Beregn $P(B \mid A)$ med Bayes regel.

●●● **Opgave 1.3.18:**
En kvinde er gravid med tvillinger, der begge er drenge. En tredjedel af tvillinger er enæggede. Antag, at enæggede tvillinger har lige stor chance for at være piger eller drenge, og at alle fire muligheder har samme sandsynlighed for tveæggede tvillinger. Lad A være begivenheden, at man får énæggede tvillinger. Lad B være begivenheden, at man får to drenge.

1) Opskriv $P(A)$ og $P(B \mid A)$.

2) Udregn $P(B)$ med loven om total sandsynlighed.

3) Brug Bayes regel til at udregne $P(A \mid B)$, altså sandsynligheden for, at kvindens tvillinger er enæggede.

1.4 Induktion - en teoretisk fortsættelse

Eksempel 1.4.1 (Konstruktion af $\mathbb{N}$)**.** For at få ideen til "induktionsprincippet" vil vi starte med at "konstruere" (altså "bygge") de naturlige tal $\mathbb{N}$. Dette gør vi i to skridt:

(1) Definér det første naturlige tal til at være tallet 1.

(2) For hvert naturligt tal n definerer vi det næste naturlige tal til at være $n+1$.

Altså er 1 det første tal i $\mathbb{N}$. Ved hjælp af det andet skridt kan vi nu lade $n = 1$. Da er $n + 1 = 2$, så 2 er det næste tal i $\mathbb{N}$. Ved fortsat anvendelse af det andet skridt kan vi på denne måde konstruere alle de naturlige tal. ∘

Dette giver os ideen til følgende sætning.

Sætning 1.4.2 (Induktionsprincippet)**.** *Induktionsprincippet siger, at hvis et udsagn opfylder de to nedenstående punkter, da gælder udsagnet for alle tal i* $\mathbb{N}$*.*

(1) Det gælder for $n = 1$*.*

(2) Hvis det gælder for n*, så gælder det også for* $n + 1$*.*

Det første punkt kalder vi *induktionsstarten*, og det andet punkt kalder vi *induktionsskridtet*.

Vi starter med et par opgaver, hvor I kan afprøve induktionsprincippet og bedre forstå det.

• **Opgave 1.4.1:**
Brug induktionsprincippet til at vise, at lige tal nummer n er lig $2n$.
[Hint: Prøv først at regne det for $n = 1$, $n = 2$ og $n = 3$.]

• **Opgave 1.4.2:**
Vis, at ulige tal nummer n er lig $2n - 1$.
[Hint: Brug induktionsprincippet. Hvis du har svært ved andet skridt, så prøv at udregne det for nogle forskellige værdier af n.]

De følgende opgaver er en smule mere udfordrende. Disse er eksempler, hvor induktionsprincippet anvendes til at bevise nogle generelle udtryk.

•• **Opgave 1.4.3:**
Vis, at summen af de første n ulige tal er lig n^2, altså $1 + 3 + 5 + \cdots + (2n - 1) = n^2$.
[Hint: Husk, at ulige tal er på formen $2n - 1$.]

•• **Opgave 1.4.4:**
Vis, at en $(n + 2)$-kant har en vinkelsum på $n \cdot 180°$.
[Hint: Prøv først at regne nogle eksempler med små værdier af n.]

Definition 1.4.3 (Geometrisk sum)**.** En *geometrisk sum* er en sum på formen $x^0 + x^1 + \cdots + x^n$ for $x \neq 1$ og $n \in \mathbb{N}$.

•• **Opgave 1.4.5:**
Vis ved brug af induktion, at $x^0 + x^1 + \cdots + x^n = \frac{1-x^{n+1}}{1-x}$.
[Hint: Bemærk, at $x^0 = 1$ for alle tal x.]

••• **Opgave 1.4.6:**
Vis uden brug af induktion, at $x^0 + x^1 + \cdots + x^n = \frac{1-x^{n+1}}{1-x}$.
[Hint: Prøv at gange summen med $(1 - x)$.]

Kapitel 2

Fysik

2.1 Matematik

Ligesom i alle andre naturvidenskaber, spiller matematik en vigtig rolle i fysikken. Matematik er både det sprog vi bruger til at kommunikere fysiske sammenhænge, og samtidig et værktøj vi bruger til at løse problemer. I dette kapitel forudsætter vi derfor, at læseren er bekendt med algebraisk manipulation til løsning af simple ligninger. Hvis du ikke er helt sikker på hvad det betyder, så læs dette kapitel.

Algebra

Ligninger slipper man ikke uden om i fysikken, det kan være I har lært at løse en ligning før, men vi vil her give en kort forklaring. En fundamental regel er, at det du gør mod et element i din ligning, det gør du mod alle elementer i din ligning. Det kan forståes som, hvis du har ligeså mange lektier som din klassekammerat, kunne du opstille følgende ligning:

$$Dine\_lektier = Din\_vens\_lektier \tag{2.1}$$

Hvis i begge får 1 lektie mere for, så gælder ligningen for jeres mængde af lektier stadig, da I stadig har lige mange lektier for. Det kunne også være, at I begge klarede halvdelen af jeres lektier, så gælder ligningen stadig. Men hvis du klarer halvdelen af dine lektier, mens din kammerat ikke får lavet nogen, eller din klassekammerat får flere (ekstra) lektier for, så gælder ligningen ikke, da I så ikke længere har lige mange lektier for. I mange matematiske tilfælde beskriver vi *lektier*, eller hvad det nu er for en størrelse vi er interesseret i, med et x. Man vælger ofte x, da x kan betegne hvad som helst, f.x. lektier. Vi gengiver "*lektie ligningerne*" nedenunder, hvor x er mængden af lektier, som du startede med, og y er mængden af lektier, som din kammerat startede med. Hver side af lighedtegnet angiver, hvor mange lektier I hver især har nu.

I starter begge ud med den samme mængde lektier, men vi skriver dem her med symbolerne x og y:

$$x = y \tag{2.2}$$

I er hver især startet ud med x og y og I har begge fået en lektie mere, så nu har I følgende mængde lektier:

$$x + 1 = y + 1 \tag{2.3}$$

Er I hver især startet ud med x og y, og I har begge lavet halvdelen af jeres lektier, kan det skrives således:

$$\frac{x}{2} = \frac{y}{2} \tag{2.4}$$

Hvis vi indsætter og siger, at $x = 1$ i alle ligningerne, så skulle vi finde, at y har samme værdi i alle andre ligninger som i den første, da vi startede med at antage, at I havde fået den samme mængde lektier for:

$$1 = y \tag{2.5}$$

Her skriver vi, at y er 1, da vi indsatte $x = 1$ i $x = y$, derfor må y også være lig med 1 i de resterende ligninger.

Vi skulle gerne få samme resultat, når vi kigger på alle ligningerne som vi har gennemgået. Lad os derfor se på den anden ligning som vi gennemgik. I starter begge ud med x og y, og I har begge fået 1 lektie mere:

$$1 + 1 = y + 1 = 2 \tag{2.6}$$

Her får vi så en ny ligning $y + 1 = 2$, men hvis vi trækker 1 fra på begge sider, så må vi kunne finde ud af, hvad y er. Hvis vi formindsker begge sider med 1, så må de stadig være lige store, $20 = 20$ og $20 - 1 = 20 - 1 = 19$ har begge det samme stående på hver side af deres lighedstegn. Altså; 20 er ikke det samme som $20 - 1$, men $20 - 1$ er det samme som $20 - 1$. Trækker vi nu 1 fra på begge sider af ligningen:

$$y + 1 - 1 = 2 - 1 = 1 \tag{2.7}$$

Så får vi at $y = 1$, fordi $y+1-1 = y+0 = y$, da $+1$ og -1 går ud med hinanden og giver 0.

Vi ser nu på ligningen, hvor I begge har lavet halvdelen:

$$\frac{1}{2} = \frac{y}{2} \tag{2.8}$$

Her bliver vi nødt til at gange med 2 på begge sider, det må vi igen godt, fordi fordobler vi begge sider, så må begge sider stadig være lige store. For eksempel så har $20 = 20$ og $20 \cdot 2 = 20 \cdot 2 = 40$ begge det samme stående på hver side af deres lighedstegn, ligesom tidligere, hvor vi trak det samme fra på begge sider.

$$\frac{1}{2} \cdot 2 = \frac{y}{2} \cdot 2 = 1 = y \tag{2.9}$$

Så vi får altså igen, at $y = 1$, fordi $\frac{y}{2} \cdot 2 = y \cdot \frac{2}{2} = y \cdot 1$, da 2 halve giver 1 hel. Vi bemærker kort at alle regnerglerne også virker, hvis x og y ikke er lig med hinanden, som hvis du eller din ven gik i en klasse med ekstra lektier. Den første ligning skulle blot afspejle det.

Nu antager vi, at du starter ud med x lektier, og du har 1 lektie mere end din kammerat, der starter ud med y lektier. Dette kan udtrykkes i en ligning som denne:

$$x = y + 1 \tag{2.10}$$

Har vi så, at du får $x = 3$ lektier for, så må din kammerat få $y = 2$ lektier for, da $3 = 2+1$.

Forestil dig, at du får x lektier for, og du har en mindre søskende i en mindre klasse, hvor de får y lektier for, der er halvt så mange lektier, som du får:

$$x = y \cdot 2 \tag{2.11}$$

Nu kan vi prøve at sætte 2 ind som x og prøve at finde y:

$$2 = y \cdot 2 \tag{2.12}$$

Dette giver:

$$y \cdot 2 \cdot \frac{1}{2} = y \cdot \frac{2}{2} = 1 \tag{2.13}$$

Vi får altså, at din mindre søskende har 1 lektie for, hvis du har 2 og generelt bare dobbelt så mange som du har.

Konklusionen du kan tage fra det her er, at hvis du gerne vil isolere noget (det er hvad vi kalder det for når vi får noget til at stå alene, så for eksempel i udtrykket $x = y + 1$ er x isoleret), ligesom vi har gjort for at finde y, så skal du bare "anti"-gøre det, der påvirker den. I tabel 2.1 kan I se de forskellige regnetegn og deres modsætninger. Hvis du f.x. har et tal benævnt a, som du gerne vil fjerne fra den ene side i en ligning, f.x. i $x = y + a$, så skal du bare finde ud af, hvad den modsatte operation til + er, og her ville det være at trække fra, så $x - a = y + a - a = y$. Vi kan også prøve at indsætte $a = 1$ og se, at det passer $x - 1 = y + 1 - 1 = y$.

Tabel 2.1: Regneværktøjer og deres modsætning

Regneværktøj	Regnetegn	Modsætning	Anti-regnetegn
Lægge til	$a + b$	Trække fra	$a - b$
Gange	$a \cdot b$	Dividere	$\frac{a}{b}$
2. Potens	a^2	Kvadratrod	$\sqrt{a}$

Man skal bare huske, at gør du noget ved én ting, skal du gøre det ved alle ting, hvis du ligger 1 til på den ene side af lighedstegnet, så gør du også det på den anden side af lighedstegnet.

Hvis du fordobler et led, så gør du det på alle led, (her er et led bare tal ganget sammen og led på samme side af et lighedstegn er adskilt af + eller −, så $2 \cdot 3$ eller $x \cdot y$ er hver et led, men $2 \cdot 3 + x \cdot y \cdot z$ er to led og $1 \cdot 2 = 2 \cdot 3 \cdot 4 + x \cdot y$ er tre led, da det er tre forskellige ting, der hver for sig er ganget sammen, som lægges sammen. Dette er illustreret i følgende ligning:

$$\underbrace{1 \cdot 2}_{\text{1. led}} = \underbrace{2 \cdot 3 \cdot 4}_{\text{2. led}} + \underbrace{x \cdot y}_{\text{3. led}}$$

Se på det som, at du har 4 forskellige bunker, der har en eller anden samlet vægt, og hver bunke har sit materiale f.x. stål, pap, plast og træ. Den samlede vægt er altså:

$$\text{total} = \text{stål} + \text{pap} + \text{plast} + \text{træ}$$

Skal du fordoble den samlede vægt, så skal du fordoble vægten af dem alle sammen:

$$2 \cdot \text{total} = 2 \cdot \text{stål} + 2 \cdot \text{pap} + 2 \cdot \text{plast} + 2 \cdot \text{træ}$$

Vi kan ikke bare fordoble stål og sige, at den samlede vægt er fordoblet, da de hver især giver deres eget tillæg til den samlede vægt.

For at simplificere den formel, så vil vi bruge en parentes altså "()", det gøres således:

$$2 \cdot \text{total} = 2 \cdot (\text{stål} + \text{pap} + \text{plast} + \text{træ})$$

Parentesen siger at indholdet inden i den skal regnes først. I dette tilfælde kan man så enten gange det, der står udenfor parentesen ind, i dette tilfælde 2 tallet, på alle ledene indeni parentesen, så får vi:

$$2 \cdot (\text{stål} + \text{pap} + \text{plast} + \text{træ}) = 2 \cdot \text{stål} + 2 \cdot \text{pap} + 2 \cdot \text{plast} + 2 \cdot \text{træ}$$

ELLER man kan lægge stål sammen med de andre bunker først og så gange med 2. Dette er dog svært at illustrere med dette eksempel, da det ikke rigtig giver mening at lægge f.x. stål og træ sammen.

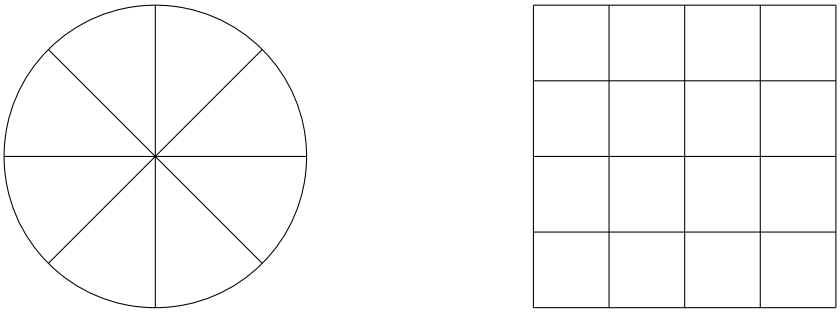
Set på samme eksempel men med penge, så om du får det dobbelte af 100 kr. og så får det dobbelte af 150 kr., eller om du får 100 kr. og 150 kr. og så får fordoblet, hvad du har lige har fået, så giver det samme resultat:

$$(100 + 150) \cdot 2 = 100 \cdot 2 + 150 \cdot 2 = 200 + 300 = 500$$
$$(100 + 150) \cdot 2 = (250) \cdot 2 = 500$$

Brug af parenteser er noget, som vi gør meget brug af i fysikken, fordi det gør vores formler enklere, f.x. med vores bunker ville vi ikke behøve at skrive gange 2 hele tiden, men kun én gang.

Regneregler for brøker og potenser

I dette afsnit vil der blive gennemgået forskellige regneregler, der ofte bliver brugt, når man løser forskellige ligninger i fysik. Der vil kort blive gennemgået, hvad brøker og potenser er, og hvordan man regner med dem. Til sidst i afsnittet vil regnereglerne være skrevet op på en generel form, selvom regnereglerne vil blive udledt ved brug af specifikke tal.



Figur 2.1: Repræsentation af brøker.

På figur 2.1 kan I se nogle figurer, der er delt op i mindre dele. Det er i princippet det, som vi gør med brøker. Vi har mindre dele af en hel enhed. For eksempel er et klassisk eksempel en halv eller $\frac{1}{2}$, men hvis man har to halve, altså $\frac{2}{2} = 1$, kan man konstater, at to halvdele giver en hel.

Dette virker ikke kun med halve, det gælder ligemeget hvor mange dele, som du vælger at inddele din enhed i. Har du har alle de mindre dele, så har du også den hele. Vores eksempel med to halve, gælder altså for alle tal, om det er halvfjerds eller 30 millioner, så halvfjerds halvfjerdsindstyvendedele er en hel, det samme skrevet med matematik er: $\frac{70}{70} = 1$.

Det er sådan, vi ofte arbejder med brøker, vi vil gerne have tingene giver 1, fordi så går de ud, eller rettere er ligegyldige, det betyder nemlig ikke så meget, hvis vi ganger med 1, fx. $1 \cdot 2 = 2$, vi kan altså bare skrive 2 i stedet for regnestykket. Der findes et tal, som vi ikke kan have dele af, det er 0, derfor er det påkrævet at det nederste i brøken ikke er nul

- **Opgave 2.1.1: Næ-nej, det må man ikke!** Prøv at tænke dig til, hvad man får, hvis man dividerer med 0.

Vi kan dog have en situation, hvor vi ganger brøker sammen, fx. en halv gange en halv eller halvdelen af en halv $\frac{1}{2} \cdot \frac{1}{2}$, halvdelen af en halvdel er en fjerdel, altså $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$, det får vi ved at *gange lige over*, altså $\frac{1}{2} \cdot \frac{1}{2} = \frac{1 \cdot 1}{2 \cdot 2} = \frac{1}{4}$. Det gælder selvfølgelig også for alle andre tal, bare vi husker vores regel med ikke at dividere med nul! Det modsatte kan også gøres, at *dividere* med brøker, hvilket kan være bøvlet nogle gange, men man gør i realiteten det samme. fx. $\frac{\frac{1}{2}}{\frac{1}{2}}$, her ved vi jo svaret er 1, da det er et tal divideret med sig selv. men i realiteten står der jo faktisk $\frac{1}{2} \cdot \frac{1}{\frac{1}{2}}$, hvilket vi kan regne med den forrige regel som $= \frac{1 \cdot 1}{2 \cdot \frac{1}{2}} = \frac{1}{\frac{2}{2}} = \frac{1}{1} = 1$ og minsandten så gav det 1.

Men vi kan også se det på en anden måde, hvor vi i stedet flytter rundt på den nederste del af brøken. $\frac{1}{2} \cdot \frac{1}{\frac{1}{2}}$, hvis vi halverer hvad *heleheden* er, så må vi have en dobbelt så stor del af helheden, altså hvis du nåede 1 ud af de 2 ting du skulle, har du $\frac{1}{2} = 0,5$, men hvis du kun skulle klare halvdelen altså 1, så $\frac{1}{\frac{2}{2}} = \frac{1}{1} = 1$, så har du nået det dobbelte af hvad du skulle opnå, ift. hvad du havde før. Det hænger altså sådan sammen, at hvis du dividerer den nederste del af brøken med noget, så kan du også bare

gange med det samme, i vores tilfælde $\frac{1}{2\cdot\frac{1}{2}} = 2\cdot\frac{1}{2} = \frac{2}{2} = 1$. Igen gælder dette for alle tal, så længe vi ikke dividerer med 0. Regneteknikken kan også huskes på, at man skal *gange omvendt*, ift. når vi ganger brøkerne *lige over*. Du ganger altså toppen på den ene med bunden på den anden og bunden af den ene med toppen af den anden. Dette gjorde vi her, ved at gange toppen 1 med bunden af det den bliver divideret med '$\frac{1}{2}$', altså 2, og bunden bliver ganget med 1. Skrevet på matematik-sprog:

$$\frac{1}{2\cdot\frac{1}{2}} = \frac{1\cdot 2}{2\cdot 1} = \frac{2}{2} = 1 \tag{2.14}$$

Så kan vi også have brøker, der lægges sammen, fx. to halve, som vi ved giver 1, det gør man matematisk således: $\frac{1}{2}+\frac{1}{2} = \frac{1+1}{2} = 1$, man lægger altså bare *delene* sammen. Det gælder dog kun hvis det er med ens bunde. fx:

$$\frac{1}{2}+\frac{1}{4} \neq \frac{1+1}{2}, \neq \frac{1+1}{4} \tag{2.15}$$

Her er vi nødt til at sørge for at vi har ens nævnere, det gør vi ved at forlænge de to brøker, fx $\frac{1}{2}\cdot\frac{2}{2} = \frac{1\cdot 2}{2\cdot 2} = \frac{2}{4}$ Vi får i vores eksempel i ligning ligning (2.15):

$$\frac{1}{2}+\frac{1}{4} = \frac{1\cdot 4}{2\cdot 4}+\frac{1\cdot 2}{4\cdot 2} = \frac{5}{8} \tag{2.16}$$

Nogle gange kan vi have at bunden er beskrevet ved en sum, fx. $\frac{1}{-1+1}$, den brøk må vi ikke have, da vi dividerer med 0 altså $\frac{1}{-1+1} = \frac{1}{0}$, så selvom der ikke direkte står 0, så kan det stadig være *ulovligt*, så hvis vi har en vilkårlig brøk $\frac{a}{b+c}$, så må det gælde at $b+c \neq 0$, fordi ellers er brøken *ulovlig*!

Nogle gange er det nyttigt at kunne gange det samme tal med sig selv flere gange. For eksempel kan man ønske sig at skulle gange 2 med sig selv 4 gange: $2\cdot 2\cdot 2\cdot 2 = 16$. Dette kan dog skrives på en nemmere måde ved at bruge potenser:

$$2^4 = \underbrace{2\cdot 2\cdot 2\cdot 2}_{4} = 16$$

På denne måde er det muligt at skrive nogle udtryk på en mere kompakt form:

$$2^4 = 16$$

Her kaldes tallet 2^4 for en potens, så når vi snakker om at opskrive en potens, så mener vi at opskrive et tal på den måde. I dette tilfælde kaldes tallet 2, der er farvet blåt, for *grundtallet* og tallet 4, der er farvet rødt, kaldes for *eksponenten*.

Der skal nu undersøges, hvad der sker, når forskellige udtryk med potenser kombineres, og i slutningen af dette afsnit vil regnereglerne være skrevet op på en generel måde.

Hvad sker der for eksempel, når hele den forrige ligning opløftes i tredje?

$$(2^4)^3 = \overbrace{\underbrace{(2\cdot 2\cdot 2\cdot 2)}_{4}\cdot\underbrace{(2\cdot 2\cdot 2\cdot 2)}_{4}\cdot\underbrace{(2\cdot 2\cdot 2\cdot 2)}_{4}}^{3\cdot 4=12} = 2^{12} = 2^{4\cdot 3}$$

Det kan således ses, at opløftes en potens med en eksponent, så ganges eksponenterne sammen.

Hvad sker der mon, når to potenser med samme grundtal men forskellige eksponenter divideres med hinanden?

$$\frac{2^5}{2^2} = \frac{\overbrace{(2\cdot 2\cdot 2\cdot 2\cdot 2)}^{5}}{\underbrace{(2\cdot 2)}_{2}} = \overbrace{(2\cdot 2\cdot 2)}^{3} = 2^3 = 2^{5-2}$$

Det kan således ses, at divideres to potenser med samme grundtal så fratrækkes eksponenten fra potensen i nævneren fra potensen i tælleren.

Er det derimod to potenser med forskelligt grundtal men samme eksponent, der divideres med hinanden gælder følgende:

$$\frac{2^3}{5^3} = \frac{\overbrace{(2\cdot 2\cdot 2)}^{3}}{\underbrace{(5\cdot 5\cdot 5)}_{3}} = \underbrace{\left(\frac{2}{5}\right)\cdot\left(\frac{2}{5}\right)\cdot\left(\frac{2}{5}\right)}_{3} = \left(\frac{2}{5}\right)^3$$

I dette tilfælde kan hele brøken sættes som grundtal og den fælles eksponent som den nye eksponent.

Er det nu to potenser med forskelligt grundtal og samme eksponent, der ganges med hinanden sker følgende:

$$2^3\cdot 5^3 = \underbrace{(2\cdot 2\cdot 2)}_{3}\cdot\underbrace{(5\cdot 5\cdot 5)}_{3} = \underbrace{(2\cdot 5)\cdot(2\cdot 5)\cdot(2\cdot 5)}_{3} = (2\cdot 5)^3$$

Produktet af de to grundtal bliver således det nye grundtal opløftet i den gamle eksponent.

Ganges to potenser med samme grundtal sker følgende:

$$2^3\cdot 2^2 = \underbrace{(2\cdot 2\cdot 2)}_{3}\cdot\underbrace{(2\cdot 2)}_{2} = \underbrace{(2\cdot 2\cdot 2\cdot 2\cdot 2)}_{5} = 2^5 = 2^{3+2}$$

Grundtallet forbliver således det samme, og den nye eksponent bliver summen af de to eksponenter.

De regneregler, der er blevet gennem gået i dette kapitel, står her til sidst på en mere generel måde [1], hvor bogstaverne a, b, c og d kan være forskellige tal, medmindre andet er angivet. For eksempel kan man i ligning (2.17) bruge $a = 3$, $b = 9$ og $c = -5$, hvilket giver $3\cdot 9/(-5) = 27/(-5)$.

Brøkregneregler:

$$a\cdot\frac{b}{c} = \frac{ab}{c}, \quad c\neq 0 \tag{2.17}$$

$$\frac{a}{b}\cdot\frac{c}{d} = \frac{ac}{bd}, \quad b\neq 0, d\neq 0 \tag{2.18}$$

$$\frac{\frac{a}{b}}{c} = \frac{a}{bc}, \quad b\neq 0, c\neq 0 \tag{2.19}$$

$$\frac{a}{\frac{b}{c}} = \frac{ac}{b}, \quad b\neq 0 \tag{2.20}$$

$$\frac{\frac{a}{b}}{\frac{c}{d}} = \frac{a}{b}\cdot\frac{d}{c} = \frac{ad}{bc}, \quad b\neq 0, c\neq 0 \tag{2.21}$$

$$\frac{ab}{ac} = \frac{b}{c}, \quad c\neq 0, a\neq 0 \tag{2.22}$$

Potensregneregler:

$$(a^b)^c = a^{b\cdot c}, \quad a>0 \tag{2.23}$$

$$a^{-1} = \frac{1}{a}, \quad a\neq 0 \tag{2.24}$$

$$\frac{a^m}{a^n} = a^{m-n}, \quad a\neq 0 \tag{2.25}$$

$$\frac{a^n}{b^n} = \left(\frac{a}{b}\right)^n, \quad b\neq 0 \tag{2.26}$$

$$a^n\cdot b^n = (a\cdot b)^n \tag{2.27}$$

$$a^m\cdot a^n = a^{m+n} \tag{2.28}$$

$$a^0 = 1, \quad a\neq 0 \tag{2.29}$$

2.2 Bevægelse i 1 dimension

Før vi kan forstå impuls og energi, er det en god ide at forstå, hvordan ting bevæger sig. I fysik skelner vi mellem bevægelse i 1, 2 og 3 dimensioner, så lad os først prøve at forstå begrebet *dimension*.

Dimensioner

Du har måske hørt, at vi bor i en tredimensional verden[1], men hvad vil det egentlig sige? Det betyder, at vi kan skelne mellem tre forskellige vinkelrette linjer i rummet; op/ned, højre/venstre og frem/tilbage.

Opgave 2.2.1: Den 4. Linje Prøv at tænke dig til en fjerde linje, der er vinkelret på de tre andre.

Du skulle gerne nå frem til, at opgave 2.2.1 ikke kan lade sig gøre. Det er derfor vi siger, at vi bor i en tredimensional verden[2]. I dette kapitel vil vi for nemhedens skyld holde os til at arbejde med én dimension. Det vil sige, at alting foregår på den samme linje, eller sagt på en anden måde, så er der kun to retninger man kan gå, nemlig frem og tilbage. Altså vil den rummelige verden vi arbejder med fra nu af, bestå af en enkelt linje.

Position, hastighed og fart

Du har måske en intuitiv forståelse for, hvordan bevægelse ser ud, så lad os nu sætte lidt matematik og nogle begreber på denne intuition. Her er det vigtigt at have tre begreber på plads. Det første er *position*. Dette er, hvor et givet objekt er henne i rummet. Når vi arbejder med 1 dimension, er positionen derfor angivet af bare et enkelt tal, altså hvor langt henne af linjen, man er. Generelt er position en vektor, mere om dette i det følgende.

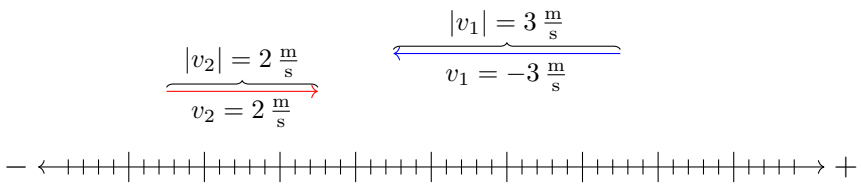
Det andet, vi skal kende til, er *hastighed*. Dette er, hvor hurtigt positionen af et objekt ændrer sig. For eksempel kunne et objekt have en hastighed på $5\,\frac{\mathrm{m}}{\mathrm{s}}$ mod nord. Grunden til, at jeg angav en retning, er, at hastighed er en vektor. Du behøver ikke vide særlig meget om vektorer for at forstå det følgende, bare tænkt på vektorer som tal med en retning eller en pil med en længde og en retning. Når vi arbejder i én dimension, er retningen angivet af fortegnet på hastigheden, for eksempel kunne minus betyde tilbage, og plus fremad.

Fart hænger sammen med hastighed, og det er en hyppig fejl, at man bytter om på dem. Fart er *længden* af hastighedsvektoren, da hastigheden jo er givet ved en vektor, og modsat hastigheden er farten ikke en vektor. Altså har fart ingen retning, kun en størrelse, mens hastighed har både størrelse og retning. Længder kan kun være positive, da det for eksempel ikke giver mening at sige, at en pind er $-5\,\mathrm{cm}$ lang, og derfor er farten af et objekt altid en positiv størrelse. Det betyder, at i det endimensionelle tilfælde, ville både et objekt med en hastighed på $-5\,\frac{\mathrm{m}}{\mathrm{s}}$ og $5\,\frac{\mathrm{m}}{\mathrm{s}}$ have en fart på $5\,\frac{\mathrm{m}}{\mathrm{s}}$. Med farten kan vi altså sige om nogen løber eller går, men med hastigheden kan vi sige noget om, hvorvidt de løber eller går væk fra os eller tættere på os.

Dette er illustreret på figur 2.2, hvor der er tegnet en rød og en blå pil, der symboliserer to forskellige objekter, der bevæger sig med to forskellige hastigheder:

[1] Med det mener vi 3 rummelige dimensioner. Derudover har vi en tidslig dimension, men den skiller sig lidt ud fra de andre. Hvis du vil vide mere om dette, så find en god bog om speciel relativitetsteori

[2] Mere formelt kan man sige at der er tre *lineært uafhængige* retninger



Figur 2.2: $|v|$ angiver længden, farten, af pilen, og v angiver, hastigheden, retningen af pilen. Farten er altid positiv, selvom hastigheden godt kan være negativ.

Acceleration

Det kan være, at du har hørt om acceleration, f.x. i forhold til biler. Der hører man måske om en bil, der kan accelerer fra "*0 til 100 på 10 sekunder*". Det der bliver sagt her er, at bilen har ændret sin hastighed fra 0 $\frac{\mathrm{km}}{\mathrm{t}}$ til 100 $\frac{\mathrm{km}}{\mathrm{t}}$ over en periode på 10 s. Når vi beskriver acceleration, så beskriver vi altså ændringen på hastigheden over, hvor lang tid det tog at ændre hastigheden, f.x. en stigning på 100 $\frac{\mathrm{km}}{\mathrm{t}}$ over 10 s.

Dette minder dig måske om det ovenstående afsnit, hvor hastigheden var ændringen af positionen over tid. Acceleration er altså ændringen af ændringen af position over tid[3]. Vi kunne blive ved med at kigge på ændringer af ændringer af ændringer. Men det er sjældent at vi gider at gå dybere end acceleration. Ligesom hastigheder er accelerationer også vektorer, du kan jo ikke blive hurtigere i ingen retning, så står du bare stille.

For at beregne acceleration, så skal du egentligt bare gøre matematisk, som vi lige beskrev det. Acceleration er en ændring i hastighed over den tid, som det tog. En *ændring* i hastighed betyder, at vi finder ud af, hvor meget større den hastighed man slutter med er, end den man starter med. Man trækker altså de to hastigheder fra hinanden, f.x er ændringen fra 1 til 4 lig med 3. Det finder vi ved at trække 1 fra 4. I tilfælde hvor starthastigheden er nul, så trækker du så nul fra sluthastigheden, hvilket ikke ændrer noget. Når der står noget over noget andet, betyder det matematisk, at *noget* er divideret med *noget andet*. Dette skriver vi ofte som brøker, som vi har beskrevet i afsnit 2.1, der starter på side 22.

Man finder altså accelerationen a ved at trække starthastigheden v_{start} fra sluthastigheden v_{slut}, altså sluthastigheden minus starthastigheden, og dividere det med hvor lang tid det tog $t_{slut} - t_{start}$, det giver os følgende formel:

$$a = \frac{v_{slut} - v_{start}}{t_{slut} - t_{start}} \tag{2.30}$$

Vi vil gøre den lidt mere generel:

$$a = \frac{v_{slut} - v_{start}}{\Delta t} \tag{2.31}$$

Her har vi beskrevet tiden det tog, som Δt, det gør vi, fordi i matematikkens verden betyder Δ[4], også kaldet *delta*, ændring. Fordi nogle gange er vi bare givet ændringen i tid, altså hvad starttid trukket fra sluttid var. Men i realiteten står der det samme, det er bare en mere generel, simpelere og pænere måde at skrive det på.

Opgave 2.2.2: Fra 0 $\frac{\mathrm{m}}{\mathrm{s}}$ til 20 $\frac{\mathrm{m}}{\mathrm{s}}$ på 10 sekunder En cyklist begynder at cykle. Efter 5 sekunder har han nået en hastighed på 20 $\frac{\mathrm{m}}{\mathrm{s}}$.

1) Hvor hurtigt har han accelereret? (Hint: brug ligning (2.31))

[3] I gymnasiet lærer man noget, der hedder *differentialregning* og *integralregning*. Dette kan bruges til at beskrive ændringer i et system, og det kan således bruges til med matematik at beskrive hastighed som en ændring af position over tid, og ligeledes for acceleration.

[4] Dette symbol er den store udgave Δ af det græske bogstav delta. Det lille delta ser sådan her ud δ.

Kraft

Nu har vi beskrevet, hvad position, hastighed og acceleration er. Nu skal vi forklare det, som Isaac Newton arbejdede med, *kraft*. Kraft er forholdet mellem inertien, altså vægten, af en ting ganget med dens acceleration, det er faktisk det som Newtons anden lov siger:

$$F = m \cdot a \tag{2.32}$$

f.x. har vi tyngdekraften, som altid skubber os lodret ned.

Som I dog nok har lagt mærke til, så bliver vi ikke trukket ned igennem jorden. Dette er, fordi jorden skubber tilbage med en ligeså stor modsatrettet kraft, (den kraft kaldes normalkraften), som gør, at kræfternes reelle påvirkning på os er 0, vi har altså en ligning som giver nul:

$$F_{tyngde} + F_{normal} = 0 \tag{2.33}$$

Det er meget herligt at den giver nul, da den både gør, så vi ikke falder igennem jorden, og så vi videre kan udlede formler for impuls. Vi kommer ikke til at arbejde så meget mere med kræfter i dette kompendium, vi bruger dem kun til at udlede de love, vi skal bruge, og det gør vi nede i afsnit 2.3.

2.3 Impuls

Impuls er et begreb, der beskriver hastigheden af noget sammen med dets inerti. Inerti er et begreb for, hvor meget en ting vil modstride acceleration, altså ændring i sin hastighed. Det kan forståes som hastigheden af en ting, og hvor svær den er at stoppe. I vores tilfælde er inerti ensbetydende med masse eller vægt. Det kan være, at det umiddelbart giver mening, hvis vi forestiller os en bowlingkugle, den er svære at flytte end et papirsfly, da bowlingkuglen er tungere og har dermed højere en inerti end papirsflyet. Vægt er i vores verden det, der gør ting svære at flytte, f.x. en bowlingkugle ift. et papirsfly. Når der er bevægelse af ting med vægt eller rettere inerti[5], så kan det beskrives med impuls, dette gøres især, hvis der er to ting med inerti som rammer hinanden, f.x. en bowlingkugle, der rammer bowlingkegler, så kan vi se, hvordan hastigheden og impulsen bliver overført fra en tung ting til en mindre tung ting og omvendt.

Nu skal vi vise jer udledningen af formlen for impuls. Det kan godt være, at matematikken ser tung ud, men vi skal nok forklare det trin for trin, og hvis du er blevet ekspert i algebraen vi har lært i afsnit 2.1, så vil du ikke have nogen problemer her. Vi starter med at kigge på Newtons 2. lov fra ligning (2.32), hvor vi vil forbinde den med ligning (2.31), der var formlen for acceleration:

$$F = m \cdot a = m \cdot \frac{v_{slut} - v_{start}}{\Delta t} \tag{2.34}$$

Vi fortalte om, hvordan to kræfter kan udligne hinanden, ligesom tyngdekraften og normalkraften i ligning (2.33). Vi skal her se på et tilfælde, hvor to bolde rammer hinanden således, at de påvirker hinanden med den samme kraft, så den totale påvirkning giver nul:

$$F_1 + F_2 = 0 \tag{2.35}$$

Det vil vi sammensætte med den formel, ligning (2.34), vi lige sammensatte for, hvad en kraft er:

$$m_1 \cdot \frac{v_{1_{slut}} - v_{1_{start}}}{\Delta t} + m_2 \cdot \frac{v_{2_{slut}} - v_{2_{start}}}{\Delta t} = 0 \tag{2.36}$$

Nu kan vi så gange ændringen af tiden væk, da det er den samme tid for begge bolde:

$$m_1 \cdot \frac{v_{1_{slut}} - v_{1_{start}}}{\Delta t} \cdot \Delta t + m_2 \cdot \frac{v_{2_{slut}} - v_{2_{start}}}{\Delta t} \cdot \Delta t = 0 \cdot \Delta t \tag{2.37}$$

[5]Det er en mærkværdig ting, at inerti og masse (ved hastigheder langt under lysets) er det samme i vores verden, prøv at forestille dig hvis de ikke var.

Så vi får:

$$m_1 \cdot (v_{1_{slut}} - v_{1_{start}}) + m_2 \cdot (v_{2_{slut}} - v_{2_{start}}) = 0 \tag{2.38}$$

Så kan vi gange ind i parentesen:

$$m_1 \cdot v_{1_{slut}} - m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{slut}} - m_2 \cdot v_{2_{start}} = 0 \tag{2.39}$$

I kan måske se, hvad vi skal nu? Lægger vi alt det, der har et minus foran sig, til på begge sidder, så kan vi vise, at udtrykkende med sluthastighederne er lige med udtrykkene med starthastigheder:

$$m_1 \cdot v_{1_{slut}} + m_2 \cdot v_{2_{slut}} = 0 + m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{start}} \tag{2.40}$$

Det lufter allerede lidt af, at der er noget som er bevaret før og efter stødet, det er selvfølgelig impuls, som vi nu bruger til at simplificere udtrykket. Vi skriver formlen for impuls som følgende:

$$p = m \cdot v \tag{2.41}$$

Så kan vi altså skrive følgende:

$$p_{1_{start}} + p_{2_{start}} = p_{1_{slut}} + p_{2_{slut}} \tag{2.42}$$

Her kan vi altså se, hvordan impulsen i starten lagt sammen, giver impulsen i slutningen lagt sammen. Det vil vi beskrive nærmere i afsnit 2.5, hvor vi taler om bevarede størrelser, som inkluderer bevarelse af impuls.

Opgave 2.3.1: Bowlingkuglens impuls En bowlingkugle med vægten 5 kg bevæger sig med 10 $\frac{\text{m}}{\text{s}}$

1) Find impulsen af bowlingkuglen (Hint: kig på ligning (2.41))

2) Hvilken enhed har impuls? (Hint: prøv at ignorere tallene og bare kig på, hvad du ganger sammen)

2.4 Energi

Energi er et begreb, I nok har hørt om, det er en størrelse der kan have mange former, fopr eksempel har både en bil der kører, din mobil og dig forskellige former for energi. Når ting bevæger sig, så har det energi, hvilket er grunden til, at man kan omdanne bevægelse til elektrisk energi, f.x. ved en hamster i et hjul. Formlen for energien af noget, der bevæger sig, den *kinetiske energi*, er:

$$E_{kin} = \frac{1}{2} \cdot m \cdot v^2 \tag{2.43}$$

I de fleste tilfælde kan formlen for, hvad energien af en ting, som gør noget, er, skrives som halvdelen af noget der har med tingen at gøre, ganget med hvor meget den gør det, i anden[6]. Her er det halvdelen af, noget der har med tingen at gøre (vægten), ganget med, hvor hurtigt den bevæger sig (hastigheden) i anden.

Vi kan se, at hastigheden er løftet til anden potens, så derfor betyder det ikke noget for energien, om den bevæger sig væk eller mod vores målepunkt. Det ville heller ikke give mening, hvis noget kunne have anti-energi, hvis det bare gik væk fra os, nej, den har brug for energi for at bevæge sig, den giver ikke energi for at bevæge sig.

Opgave 2.4.1: En flyvende hjerne En hjerne med vægten på 2 kg flyver gennem luften med en hastighed på 2 $\frac{\text{m}}{\text{s}}$

1) Hvor høj kinetisk energi har hjernen?

[6]Dette er ikke en reel regel, men bare en huskeregel for, hvad energiformlen ofte er. For eksempel minder formlen for rotationel energi meget om denne. Faktisk er rotationel energi en form for kinetisk energi.

2.5 Bevarede størrelser

I afsnit 2.3 om impuls og i introduktion til kapitlet, har vi brugt begrebet *bevarede størrelser*, som betyder størrelser, som ikke ændrer sig[7]. I har måske hørt, at energi er konstant, det betyder, at vi hverken kan ødelægge eller skabe ny energi. Det siger dog ikke noget om, at vi ikke kan omdanne en type af energi til en anden type, så længe den samlede mængde energi holdes konstant.

Dette er sandt i hvad vi kalder et *lukket system*, som er et system, hvor der ikke er noget, der påvirker det ude fra. Et lukket system er f.x. en bold, der ligger på jorden, den har ingen kinetisk energi og ligger stille. Det system er ikke længere lukket, hvis der kommer en kraft, som en fodboldspiller, der sparker den. Dog kunne vi godt have haft fodboldspillerens spark med i vores udregning af energien. Derfor kan det måske give mening, hvordan hele universet er et lukket system, hvis vi tager alting med i vores udregninger. På den måde er der ikke nogle ydre kræfter, og der bliver ikke skabt energi, men I kan nok godt forestille jer, at det er lidt svært at holde styr på *hele* universet, vi estimerer derfor, at f.x. fodbolden er i et meget lukket system.

Bevaret energi

En bevaret størrelse, som vi kommer til at arbejde med, er bevaret energi. Som vi lige har nævnt, kan man ikke ødelægge og skabe energi, derfor er der altid så meget energi i et lukket system ved slut, som der var ved start. Energien kan ikke skabes eller ødelægges, men den kan godt *tabes*[8], ved f.x. at blive til varme, I har måske prøvet at gnide jeres hænder mod hinanden for varme, der omdanner du bevægelses-energi til varme-energi. Når energi bliver omdannet til varme, så mister vi altså noget af bevægelsesenergien, det er derfor, at ting ikke kan holde hastigheder uden, der bliver tilsat mere energi, medmindre de er i specielle situationer som f.x. ude i rummet.

Hvis vi ved, at der ikke er noget energi, der bliver *tabt* som varme, så kan vi beskrive bevægelsesenergien i et system som følgende:

$$E_{kin_1} = E_{kin_2} \tag{2.44}$$

Ligningen siger, at den kinetiske energi i situation 1 er den samme som den i situation 2. Denne ligning kan vi sætte sammen med formlen for kinetisk energi, så vi får at:

$$\frac{1}{2} \cdot m_1 \cdot v_1^2 = \frac{1}{2} \cdot m_2 \cdot v_2^2 \tag{2.45}$$

Her siger vi at halvdelen af inertien, eller vægten, ganget med hastigheden i anden i situation 1 er det samme som halvdelen af inertien, eller vægten, ganget med hastigheden i anden i situation 2.

Hvis vi går ud fra, at tingene ikke ændrer vægt undervejs, så $m_1 = m_2$, kan vi beskrive dem begge med samme bogstav m, så kan vi se, hvordan der står det samme på begge sider, bortset fra v_1 og v_2, derfor ved vi, at $v_1 = v_2$, ud fra vores lignings regneregler:

$$\frac{1}{2} \cdot m \cdot v_1^2 = \frac{1}{2} \cdot m \cdot v_2^2 \tag{2.46}$$

Vi kan dividere med $\frac{1}{2} \cdot m$ på begge sidder:

$$\frac{\frac{1}{2} \cdot m}{\frac{1}{2} \cdot m} \cdot v_1^2 = \frac{\frac{1}{2} \cdot m}{\frac{1}{2} \cdot m} \cdot v_2^2 \tag{2.47}$$

[7] Der findes en tæt sammenhæng mellem symmetrier og bevarede størrelser, så at for hver gang en størrelse er bevaret, findes der en symmetri som er bundet sammen med den.

[8] Når man mener, at energi *tabes*, så mener man bare, at det er blevet omdannet til en mindre brugbar energiform.

Og så får vi:

$$1 \cdot v_1^2 = 1 \cdot v_2^2 \tag{2.48}$$
$$v_1^2 = v_2^2 \tag{2.49}$$
$$\sqrt{v_1^2} = \sqrt{v_2^2} \tag{2.50}$$
$$v_1 = v_2 \tag{2.51}$$

Vi får altså, at hastigheden i situation 1 er lig med hastigheden i situation 2, hvis der ikke tabes energi eller ændres inerti eller vægt, undervejs. Når vi arbejder med energibevarelsen, så vil vi dog arbejde med flere ting, der er i bevægelse, de udregninger kan ses i afsnit 2.6 om *stød*.

Bevaret impuls

En anden bevaret størrelse, som vi kommer til at arbejde med, er bevaret impuls i forhold til sammenstød. Det er simpelthen bare, at den mængde impuls, der var før et sammenstød, er den samme, der er efter. For eksempel to bolde, hvor den ene ligger stille og den anden har en hastighed, hvis de rammer hinanden, så deler hastigheden sig ud over dem begge. Hvad deres hastigheder slutter af med at være, det kommer så an på deres inerti, altså vægt, og hvilken form for stød det var, det vil vi dække nærmere i afsnit 2.6 om *stød*.

Igen så har vi behov for et isoleret system før dette gælder, igen hvis en fodboldspiller kommer ind og sparker til en af boldene, før vi har kunne finde deres slutimpuls, så kan der være mistet noget impuls eller skabt noget, impulsen vil altså ikke være bevaret, hvis der er noget der kommer ind og ændre på situationen.

Hvis der ikke har været nogle fodboldspillere, der har sparket til vores forsøg, det vil sige, at systemet har været isoleret, imens vi udførte vores forsøg, så kan vi opstille følgende formel, da impulsen er bevaret:

$$p_1 = p_2 \tag{2.52}$$

Ligesom vores formel for bevaret energi, så siger vi, at impulsen i situation 1 er den samme i situation 2. Nu kan vi, ligesom da vi opsatte formlen for bevaret energi, sætte formlen for impuls ind:

$$m_1 \cdot v_1 = m_2 \cdot v_2 \tag{2.53}$$

Hvis vi går ud fra, at der ikke bliver tabt eller tilføjet nogen inerti, eller vægt, så kan vi igen beskrive $m_1 = m_2 = m$, så vi kan lave følgende formel, som vi kan udlede fra:

$$m \cdot v_1 = m \cdot v_2 \tag{2.54}$$

Så kan vi igen dividere med inertien, eller vægten, på begge sider:

$$\frac{m}{m} \cdot v_1 = \frac{m}{m} \cdot v_2 \tag{2.55}$$
$$v_1 = v_2 \tag{2.56}$$

Fra begge vores udledninger kan vi se, at har et objekt en hastighed og ingenting sparker eller på anden vis påvirker det, så ændrer det ikke hastighed. I den virkelige verden vil der dog umiddelbart altid være lidt friktion og luftmodstand osv., men det ser vi bort fra, da vi alligevel kan få nogle nyttige resultater, og det gør beregninger meget nemmere. I fysik vælger vi ofte at tænke på verden som perfekt, for at gøre det muligt at regne på.

Men hvad nu hvis der ikke bare er én ting, der bevæger sig? Det er det spændende ved at regne med impuls, og det vi vil kigge på i afsnit 2.6 om *Stød*. Når vi snakker om "stød" er dette ikke elektrisk stød, men stød som i sammenstød.

2.6 Stød

Stød er det, der fremkommer, når flere bevægende ting rammer hinanden, og for at holde det simpelt, så vil vi holde os til to bevægende ting, der rammer hinanden. Princpperne, der gælder for to bevægende objekter, er de samme, der gælder for tre, fire, firehundredeogtyve osv. bevægende objekter, men det er nemmere at holde styr på jo færre objekter, som man regner med.

Flere bevægende ting

Når vi har to (eller flere) bevægende ting, så må de jo have en energi og en impuls, men rammer de hinanden, så er der jo noget, der "sparker" og ændrer på de forskellige hastigheder i systemet. Det er sandt, men det betyder ikke, at systemet mister impuls eller energi, medmindre energien går tabt i varme. De ligninger, vi skrev op under afsnit 2.5 om bevarede størrelser, gælder faktisk for hele systemet. Det gør de, da vi kan lægge energier sammen og impulser sammen. Det er det, vi vil vise i de næste afsnit.

Energien af flere bevægende ting

Det kan måske give mening, at har vi to bolde, der bevæger sig med en energi på 10 J, så må et system med de to bolde i sig have en samlet kinetisk energi på 20 J, da skulle noget blive ramt af begge bolde, så ville det blive ramt af den samlede energi, derfor kan vi lægge et systems energier sammen. Vi opstiller derfor følgende formel for den samlede energi i et system:

$$E_{kin_{total}} = E_{kin_1} + E_{kin_2} + \ldots + E_{kin_n} \tag{2.57}$$

Her har vi brugt notationen n, det bogstavet betyder, er egentligt bare det antal ting vi har, hvis det er tyve bolde, så er $n = 20$, hvilket betyder, vi ligger energi 1 sammen med 2, sammen med 3, sammen med 4 osv. indtil vi har lagt det sammen med den kinetiske energi af den 20. og sidste ting. I vores opgaver arbejder vi ikke med mere end 2, men vi ville bare give jer den generelle formel. Denne formel kan vi så sætte sammen med formlen, vi havde for den bevarede energi i afsnit 2.5 for at få følgende:

$$E_{kin_{total\,start}} = E_{kin_{total\,slut}} \tag{2.58}$$

Vi ville ikke bruge 1 og 2 til at beskrive situationerne her, for at formindske forvirring senere. I denne ligning behøver energi 1 og 2 i den totale kinetiske energi i startsituationen ikke være den samme som i den slutsituationen. Lægges de sammen, så skal de bare give det samme resultat. Faktisk kan energien godt bytte, som f.x. hvis du skyder en bold ind i en anden bold, der har samme masse, så stopper din bold måske, men den anden har præcis samme energi som din havde lige før den ramte.

Vi beskriver energien i et system med to bevægende ting, der rammer hinanden, via følgende formel:

$$E_{kin_{1\,start}} + E_{kin_{2\,start}} = E_{kin_{1\,slut}} + E_{kin_{2\,slut}} \tag{2.59}$$

Her siger vi at, hvis vi lægger den kinetiske energi af ting 1 og ting 2 sammen i startsituationen, så får vi det samme som, hvis vi lægger den kinetiske energi af ting 1 og ting 2 sammen i slutsituationen. I kan måske se, Hvis vi kender 3 af energierne, så kan vi finde den sidste via algebraen vi lærte i afsnit 2.1 om matematik.

Opgave 2.6.1: Energien af et bold stød En bold, der bevæger sig med energien 10 J, er på vej mod en bold, som ikke bevæger sig, og den har dermed en kinetisk energi på 0 J.

1) Hvad er den totale kinetiske energi i systemet med de to bolde? (Hint: prøv ligning (2.57))

2) Efter boldene har ramt, så har den første bold nu en kinetisk energi på 5 J, hvad er den kinetiske energi af den anden bold efter den blev ramt? (Hint: prøv at bruge ligning (2.59), og isoler den anden bolds kinetiske energi)

Impulsen af flere bevægende ting

Ligesom med energien, når vi har to (eller flere) objekter, der bevæger sig, så kan vi lægge deres impuls sammen, men her skal man huske, at modsat energien har impuls en retning. Det betyder, at vi kan have en negativ impuls. Dette kan gøre at når vi ligger impulsen af to objekter sammen, kan vi få nul. Det kan være at det ikke givet intuitiv mening for dig, at impulsen er 0 i et system med bevægende ting, men husk at det vi taler om er den *totale* impuls. Prøv at forestille dig to bolde med lige stor impuls rammer en kasse, de har præcis impuls nok til at give kassen en fart på 1 meter i sekundet i den retning de bevæger sig, men da de kommer fra hver deres side, så skubber den ene bold kassen til højre, mens den anden skubber kassen til venstre, så kassen ender med at have en hastighed på 0. Vi opstiller altså følgende formel, som ligner den for energi:

$$p_{total} = p_1 + p_2 + \ldots + p_n \tag{2.60}$$

Igen, har vi brugt den mere generelle formel med n og igen så bruger vi ikke fremover n som højere end 2, derfor er det bare impuls 1 lagt sammen med impuls 2. I alle sammenstød, så er der impulsbevarelse (hvis vi går ud fra at de ikke mister inerti), så vi kan altså benytte den totale impuls sammen med formlen for impulsbevarelsen fra afsnit 2.5 **Bevarede størrelser**:

$$p_{total_{start}} = p_{total_{slut}} \tag{2.61}$$

Den kan vi så udvide igen, ligesom vi gjorde med energien i ligning (2.59):

$$p_{1_{start}} + p_{2_{start}} = p_{1_{slut}} + p_{2_{slut}} \tag{2.62}$$

Her kan vi altså se hvordan, hvis vi kender tre impulser, så kan vi finde den sidste, ved hjælp af algebra.

Opgave 2.6.2: Impulsen af et bold-stød En bold bevæger sig med impulsen $10\,\mathrm{kg}\frac{\mathrm{m}}{\mathrm{s}}$ er på vej mod en bold som ikke bevæger sig og dermed har en kinetisk energi på $0\,\mathrm{kg}\frac{\mathrm{m}}{\mathrm{s}}$

1) Hvad er den totale impuls i systemet med de to bolde? (Hint: brug ligning (2.61))

2) Efter boldene har ramt, så har den første bold nu en kinetisk energi på $5\,\mathrm{kg}\frac{\mathrm{m}}{\mathrm{s}}$. Hvad er den kinetiske energi af den anden bold efter den blev ramt? (Hint: du kender de to startimpulser og en slutimpuls, kan du isolere den sidste i ligning (2.62))

Stødtyper

Når vi arbejder med impuls, arbejder vi med to ideelle former for stød, det elastiske og komplet uelastiske stød. Forskellen mellem dem er, som du nok kunne gætte, at den ene er elastisk, og den anden slet ikke er. Det, at et stød er elastisk, betyder, der ikke går noget energi tabt i at deformere tingene, der rammer hinanden. Et eksempel på elastiske stød sker f.x. med hoppebolde og elastikker, de får ikke nogle permanente buler, som kræver energi at skabe, som f.x. biler gør, når de støder ind i hinanden.

Elastiske stød

Når vi har at gøre med et fuldstændigt elastiske stød, så kan vi regne med, at ingen energi går tabt i at lave buler eller deformation. Vi kan altså regne med at energien er konstant. Vi kan dermed opstille to formler for sammenstødet, ved at bruge både impuls- og energibevarelsen:

$$m_1 \cdot v_{1start} + m_2 \cdot v_{2start} = m_1 \cdot v_{1slut} + m_2 \cdot v_{2slut} \tag{2.63}$$

og

$$E_{kin_{1\,start}} + E_{kin_{2\,start}} = E_{kin_{1\,slut}} + E_{kin_{2\,slut}} \tag{2.64}$$

Ved kombination af de to ligninger, så kan vi, gennem en længere udledning, komme frem til følgende formel:

$$v_{1start} + v_{2start} = v_{1slut} + v_{2slut} \tag{2.65}$$

Kender vi tre hastigheder, så kan vi finde den sidste. Dette kan være de to starthastigheder og en sluthastighed eller omvendt.

- **Opgave 2.6.3: Hoppeboldstød** To hoppebolde støder ind i hinanden. Til start, så har den ene hastigheden $2\,\frac{\mathrm{m}}{\mathrm{s}}$, og den anden har hastigheden $-1\,\frac{\mathrm{m}}{\mathrm{s}}$, da den ruller imod den første. Til slut så har den anden hoppebold hastigheden $2\,\frac{\mathrm{m}}{\mathrm{s}}$.

  1) Hvad vil den første hoppebolds sluthastighed være? (Hint: brug ligning (2.65)

  2) Hvad er der sket med de to boldes hastigheder?

Komplet uelastiske stød

I fuldstændigt ikke elastiske stød, så ender de objekter, der rammer hinanden, med at bevæge sig i samme retning med samme hastighed, så de hænger altså sammen, efter de har ramt hinanden.[9] I fuldstændigt ikke elastiske stød, så er det faktum, der kan fremkomme buler o.l., der gør, at vi kan miste energi i sammenstødet, hvilket gør, at vi ikke kan regne med, at energien er konstant i sammenstødet. Vi udleder derfor en anden formel, hvor vi har at gøre med et komplet uelastisk stød, så vi kan opstille følgende variant af ligning (2.40):

$$m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{start}} = m_1 \cdot v_{slut} + m_2 \cdot v_{slut} \tag{2.66}$$

Vi simplificerer den, ved at rykke m_1 og m_2 ind i en parentes, da de begge er ganget med v_{slut}:

$$m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{start}} = (m_1 + m_2) \cdot v_{slut} \tag{2.67}$$

Så kan vi isolere sluthastigheden, den bruger vi som en generel formel for fuldstændigt ikke elastiske stød:

$$\frac{m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{start}}}{(m_1 + m_2)} = \frac{(m_1 + m_2)}{(m_1 + m_2)} \cdot v_{slut} \tag{2.68}$$

Som giver os formlen:

$$v_{slut} = \frac{m_1 \cdot v_{1_{start}} + m_2 \cdot v_{2_{start}}}{m_1 + m_2} \tag{2.69}$$

Her har vi fire variable, så finder vi tre af dem, så kan vi via algebra lave en formel for at finde den sidste.

- **Opgave 2.6.4: Fælleshastighed af to kugler** To kugler bevæger sig mod hinanden og kolliderer ikke-elastisk, den ene har en hastighed på $2\,\frac{\mathrm{m}}{\mathrm{s}}$ før sammenstødet, og den anden bevæger sig halvt så hurtigt imod den første før støddet og har derfor hastigheden $-1\,\frac{\mathrm{m}}{\mathrm{s}}$. De vejer begge to 1 kg.

  1) Hvad vil deres fælles sluthastighed være? (Hint: brug ligning (2.69))

Til slut bør nævnes, at man i den virkelige verden også oplever stød, som hverken er elastiske eller komplet uelastiske. Disse stød kalder man for uelastiske stød (ikke at forveksle med de *komplet* uelastiske stød!)- I disse stød bliver en del af energien tabt i form af derformering af objekterne, men objekterne hænger ikke sammen bagefter. For at regne på disse, bliver man nødt til at kende mere end bare starthastighederne og masserne.

[9] Det kunne være to biler efter en trafikulykke

2.7 Opgaver

Ud for hver opgave er en antal prikker fra 0 til 3. Flere prikker betyder sværere opgave, vurderet af arrangørerne. Du vil måske opleve at det ikke altid stemmer med din egen oplevelse af hvad der er svært, da alle har svært ved forskellige ting.

Opgave 2.7.1: Hawkeye

0

Hawkeye affyrer en pil med en inerti på 100 g og en fart på 30 m/s

1) Hvad er pilens kinetiske energi?

2) Hvor langt væk når den fra Hawkeye på 5 sekunder?

Opgave 2.7.2: Batmobilen I

0

Batmobilen kører med en hastighed på $-120\ \frac{\text{km}}{\text{t}}$, og har en inerti på 1 ton

1) Hvor stor er batmobilens impuls?

2) Kører batmobilen i negativ eller i positiv retning?

3) Hvad er batmobilens kinetiske energi?

Opgave 2.7.3: Krudt og kugler I

1

En superskurk affyrer en kanon direkte mod et tog, der kommer kørende. Toget har en hastighed på -100 km/t og en inerti på 27 tons, projektilet fra kanonen har en inerti på 2 kg og en hastighed på 600 m/s. Kanonkuglen rammer toget, og de to hænger sammen bagefter.

1) Hvad er hastigheden af toget og kuglen bagefter?

2) Antag nu i stedet et elastisk stød, hvad bliver sluthastighederne så?

Opgave 2.7.4: Batmobilen II

1

En røver er ved at slippe væk, efter at have stjålet en kuffert med penge. Røveren kører i en flugtbil med hastigheden $70\ \frac{\text{km}}{\text{t}}$. Batman forfølger røveren i sin batmobil, og indhenter røveren på 7 s. Røveren har et forspring på 120 m.

1) Hvor hurtigt kører batmobilen?

Opgave 2.7.5: Hulk

1

Bruce Banner springer ud af et tog, og har en hastighed på 4 m/s. Bruce vejer 70 kg. Midt i luften transformerer han til Hulk. Efter transformationen har han en en hastighed på 1 m/s. Antag impulsbevarelse.

1) Hvor meget vejer Hulk?

2) Er den kinetiske energi bevaret?

3) Hvor stor en kinetisk energi har Bruce inden han transformerer?

4) Hvor stor er hans kinetiske energi efter han har transformeret?

5) Er hulk varmere eller koldere efter transformationen?

Opgave 2.7.6: Krudt og kugler II

2

Denne opgave fortsætter opgave 2.7.3.

1) Hvor mange kugler skal skurken affyre for at toget stopper helt? (Antag komplet uelastiske stød)

2) Løs ovenstående opgave med elastiske stød. Er det flere eller færre kugler? Er det som du forventede?

Opgave 2.7.7: Supermans knytnæve
2

En Bankrøver kommer løbende med en fart på $10\,\frac{\text{km}}{\text{t}}$ mod Superman. Superman slår ham, så han flyver tilbage. Efter kollisionen mellem bankrøveren og Supermans knytnæve, står næven helt stille, mens bankrøveren flyver bagud med en fart på $10\,\frac{\text{m}}{\text{s}}$. Bankrøveren har en inerti på 87 kg, mens Supermans knytnæve har en inerti på 200 g

1) Bestem hastigheden af Supermans knytnæve inden den rammer bankrøveren.

Opgave 2.7.8: Iron Mans raketter
3

Iron Man svæver i luften ved hjælp af sine raketter. Han påvirkes af tyngdekraften med en kraft på 1800 N.

1) Hvor stor en opadrettet kraft leverer raketterne?

Nu Begynder Iron Man at accelerere opad. På 10 sekunder opnår han en hastighed på 10 m/s.

2) Hvad er hans acceleraton?

3) Hvor stor en kraft skal raketterne levere, for at han opnår denne acceleration?

Opgave 2.7.9: Superman og toget
3

Et tog kommer kørende med en hastighed på $120\,\frac{\text{km}}{\text{t}}$. Toget har en inerti på 160 tons.

1) Bestem togets impuls

Et stykke $x = 100\,\text{m}$ længere henne af sporet er en bro hen over en afgrund styrtet sammen. Superman kommer flyvende og vil forsøge at stoppe toget inden det kører i afgrunden. Superman er 200 m bag togets forende, når toget er afstanden x fra afgrunden. Superman flyver med en konstant hastighed $u = 180\,\text{m/s}$.

2) Hvor langt tid tager det superman at nå frem til forenden af toget?

3) Hvor langt er toget fra afgrunden, når Superman når frem til forenden af toget?

Superman bremser nu toget ved at skubbe på forenden. Toget stopper lige nøjagtigt ved kanten af afgrunden. Det kan antages, at toget bremses med en konstant acceleration.

4) Hvor stor er accelerationen på toget under opbremsningen?

Opgave 2.7.10: Superman og symboler
3

Superman har inertien m_s, og svæver i luften uden at bevæge sig. Han griber nu en bold med en inerti på m_b, der bevæger sig med en hastighed på v_b.

1) Hvad er Supermans hastighed bagefter?

2) Hvad er energien af bolden og superman efter han har grebet bolden?

3) Hvad er energien af systemet før bolden blev grebet?

4) Hvor meget energi er gået tabt i form af varme?

2.8 Forsøg: Luftpudevogne

Introduktion

I dette forsøg skal vi se på, om vi kan bekræfte, at der findes impulsbevarelse, som vi har gennemgået i dette forløb. Det vil sige, at vi skal se, om vi kan eftervise følgende:

$$p_{\text{før}} = p_{\text{efter}} \tag{2.70}$$

$$m_{\text{før}} v_{\text{før}} = m_{\text{efter}} v_{\text{efter}} \tag{2.71}$$

$$p_{\text{før}} = m_{\text{før}} v_{\text{før}} \quad \text{og} \quad p_{\text{efter}} = m_{\text{efter}} v_{\text{efter}} \tag{2.72}$$

Vi vil i dette forsøg tage udgangspunkt i to luftpudevogne, som vi vil støde ind i hinanden. Ved at se på vognenes inerti og deres start- og sluthastigheder, kan vi sammenligne og se, om der er impulsbevarelse, som beskrevet i afsnit 2.5.

Forsøgsopstilling

Der er forskellige måder, som I kan prøve at eftervise impulsbevarelse, men vi tænker, at I kan lave en opstilling i stil med opstillingen i figur 2.3, hvis I ikke selv har nogle forslag.

Figur 2.3: Forslag til en mulig forsøgsopstilling til forsøget.

Materialer

Til forsøget har I følgende materialer til rådighed:

- 2 stativer med stænger, muffer og fødder
- 2 fotosensorer
- 2 luftpudevogne med finner og lodder
- 1 luftpudebane
- 1 blæser
- 1 LabQuest enhed

- 1 computer til målinger
- vægt

Fremgangsmåde

Her er beskrevet, hvordan I kan bruge det udstyr, som er givet til, og har I andre ideer til, hvordan tingene kan bruges, kan I spørge underviserne og høre, hvad de synes.

- For at opsætte luftpudevogne skal I sætte en skinne op på et bord og fastgøre en af blæserne til enden af skinnen. Hvis I så tænder for blæseren burde I kunne placere en af luftpudevognene på skinnen og køre den frem og tilbage uden den store friktion mellem vognen og skinnen. (Hvorfor er dette en god ting i forhold til forsøget?)
- Vognene har forskellige ting, der kan sættes på dem, så I kan eksperimentere med forskellige ting på vognene.
- Stativerne sættes sammen med stativmufferne, som er metal objekterne med to huller, så I kan indsætte to metalstænger, der så vil stå vinkelrette på hinanden. Ved brug af stativerne kan I for eksempel sætte fotosensorerne i jeres opstilling. (Hvorfor er dette en god ting i forhold til forsøget?)
- Fotosensorerne kan bruges til at detektorer, hvornår noget blokerer for lysstrålen mellem de to ben i fotosensorerne. (Hvordan kan dette mon bruges?)

Data

I bestemmer selv, hvilke værdier, som I vil prøve at måle på, og tabel 2.2 er et forslag til, hvad I kan beregne og nedskrive. I kan således sagtens skrive nogle andre ting ned.

Forsøg	Masse 1 m_1	Masse 2 m_2	Hastighed 1 v_1	Hastighed 2 v_2	Impuls 1 p_1	Impuls 2 p_2
1: Før						
1: Efter						
2: Før						
2: Efter						
3: Før						
3: Efter						
4: Før						
4: Efter						
5: Før						
5: Efter						
6: Før						
6: Efter						
7: Før						
7: Efter						
8: Før						
8: Efter						
9: Før						
9: Efter						

Tabel 2.2: Forslag til tabel til nedskrivning af værdier.

Kapitel 3

Kemi

3.1 Introduktion

Velkommen til kemiforløbet på UNF's naturfagsweekend! I folkeskolen er de to fag fysik og kemi blandet sammen i ét fag Fysik/Kemi, men på f.eks. gymnasiet eller universitet er der kæmpe forskel på de to fag! Kemi handler om at forstå den måde universet atomare byggesten reagerer med hinanden, for at dét stof som hele verdens fysiske materiale består af. Mere lavpraktisk, så er det den del af naturfaget hvor man kan hælde to væsker sammen, hvorefter der sker en farveændring, en eksplosion, eller noget andet spændende!

I dette års undervisning/kompendium skal I lære de basale kemiske færdigheder, der gør jer I stand til at regne på den kemi, der skal i glasset nemlig *mængdeberegning*, og derefter to grene inden for kemien: *Uorganisk kemi* og *termodynamik*.

Hvad er kemi for os?

Mette

Det, jeg finder interessant ved kemi, er, hvordan det kan forklare mange hverdags situationer. Man kan måske se, at der er sket noget som, at metal ruster og celler formerer sig. Bag begge fænomener er det kemiske reaktioner der finder sted. At finde ud af hvad der er sket, og hvordan den kan påvirkes er for mig det virkelig spændende. Stoffers kemiske egenskaber kan forklare hvorfor, stoffet reagerer, som de gør. For mig kan kemi give en forklaring af, hvordan ting hænger sammen og bruge kemi til at forbedre processer.

Knut

Kemi for mig er læren om verdens komponenter og hvordan, alt er bygget op. I kemi arbejder man på at kunne syntetisere nye stoffer og kunne detektere dem, dette gør man med et hav af analysemetoder. Kemi er derfor et naturvidenskabeligt fag, hvor al den viden, man har, kommer fra forsøg og eksperimenter. Det fantastiske ved kemi er, at det er med til gøre verden til et bedre sted og mange af de store problemers løsninger er baseret på viden fra kemi.

Ved hjælp af kemien kan man opskalere reaktioner og udvikle nye kemiske stoffer eller kunne producere stoffer, man tidligere fik fra naturen. Kemi hænger derfor meget sammen med industrien og har været en "katalysator" for industrialiseringen i de sidste 150 år. Blandingen af at finde ud af, hvordan verden er bygget op, og samtidig være en vigtig del af industrien er det, jeg synes, der gør kemi megafedt.

Lin

Kemi er læren om reaktioner, og hvordan én ting kan ændre sig og blive til en helt anden ting. På den måde kan kemien beskrive alle mulige seje ændringer vi ser i hverdagen; som hvordan fyrværkeriet på himlen nytårsaften fungerer, eller hvordan en tændstik og

en tændstikæske lige pludselig kan lave ild! For mange år siden var en kemiker mest beskæftiget med at fremstille uægte guld, mens nutidige kemikere kan lave nye lægemidler eller designe store tanke når der skal produceres en masse håndsprit. Det er dét jeg synes er fedt ved kemien: den kan beskrive alle de weird ting der sker omkring dig, og kan samtidig kan kemi hjælpe en masse mennesker - og det er bare super fedt.

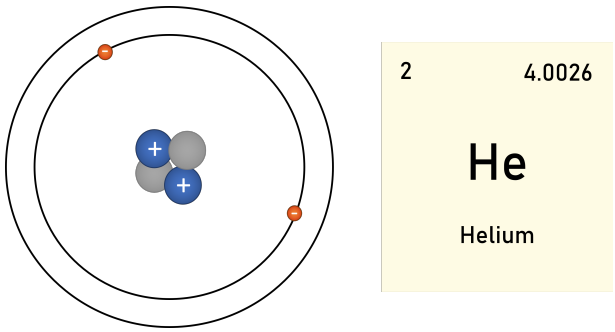
3.2 Introduktion til atomer og molekyler

Atomet

Et atom er den mindste partikel der kan eksistere alene. Et atom består af en kerne, som er positiv ladet. Kernen består af 2 typer forskellige partikler: protoner og neutroner. Protonerene og neutroner er meget små og har en meget lille masse på $1.67 \cdot 10^{-27} kg$. Neutronen har en smule højre masse end protonen. Protoner er positive ladet og en ladning på $1.6 \cdot 10^{-19} C$, som er den ladning man kalder for elementarladningen. Det vil sige at protoner har en meget høj ladningen i forhold til dens masse. Elektroner er negative ladet og er endnu mindre end protoner, og har en masse på kun $9.11 \cdot 10^{-31} kg$, men har samme størrelsesorden af ladning som en proton.

Bohrs atommodel

En god model af atomet er Bohrs atom model, som er en simplificering af virkeligheden. I virklhedden bevæger elektroner ikke baner, men det er en god approksimation til mange formål i kemien. Bohrs atom model består af en kerne hvor protoner og neutroner befinder sig i. På billedet nedefor (figur 3.1) er en atommodel af grundstoffet helium tegnet. Helium er grundstof nr. 2 og har derfor to protoner (blå og positivt ladet). Desuden er der i heliums kerne to neutroner. Kernen er omgivet af "elektronbaner", hvor elektroner befinder sig i. Disse kaldes typisk for elektronskaller. Disse tegnes som sorte cirkler. I den inderste bane kan der være 2 elektroner, i den næste kan der være 8, derefter 18. Helium har kun 2 elektroner, og har derfor fyldt første bane. Siden at kernen er positiv ladet og elektroner er negativt ladet, så vil elektronerne blive tiltrukket af kernen. Det kræver derfor energi at trække elektron væk fra kernen. Jo længere væk en elektron er fra kernen, jo høre potentielt enerig har den. Det vil sige at elektroner bliver fyldt op i den inderste elektronbane først, og derefter i den næst-inderste og så videre.



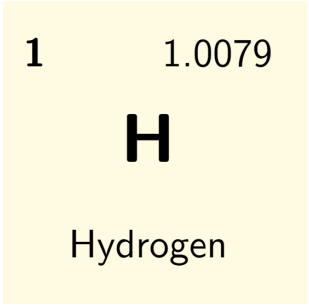
Figur 3.1: Atommodel af grundstoffet helium ved siden af helium som det står i det periodiske system. Helium har to protoner (blå), to neutroner (grå) og to elektroner (røde).

Grundstoffer og det periodiske system

Et grundstof er et atom med et bestemt antal protoner, hvor atomerne har lige mange elektroner og protoner når de ikke har en ladning. Hvis et atom mister eller får protoner bliver det til et nyt grundstof. Det sker i radioaktive processorer, men det er uden for

kemiens verden. I kemi arbejder man kun med integrationen imellem elektroner, det vil sige mængden af grundstoffer er bevaret før og efter reaktion.

Det periodiske system er en oversigt over alle de kendte grundstoffer, man kender til. Hvert grundstof har et nummer, nummeret repræsenter det antal af protoner som grundstoffet har. Omme bagerst i dette kompendiet er der et printet periodisk system. Det første grundstof i det periodiske system er hydrogen, den har grundstof nummer 1 og består af et proton en elektron. Nogle hydrogen atomer har også en neutron. Antallet af neutroner der er i kernen kalder man for isotoper. For hydrogen findes der isotoper med 0, 1 eller 2 neutroner, som også kaldes for deuterium og trirtum. Dog er næsten alt hydrogen uden en neutron. På figur 3.2 kan du se et zoom af det periodisk system på hydrogen. Øverst til venstre er grundstof nummeret, til højre for det er den gennemsnitlige masse for hydrogen. Tallet er i units, hvor 1 unit er massen af en proton. Grunden til at massen af hydrogen er en smule højre end 1 er fordi der findes andre isotoper af hydrogen. Derudover har hvert grundstof en eller to bogstaver, som er en form for forkorte for grundstoffet. Dette kalder man et grundstofsymbol, som f.eks. bruges til at skrive sammensætninger af molekyler på en kortere form.



Figur 3.2: Hydrogen i det periodiske system. Tallet øverst til venstre betyder det er grundstof nr. 1, og tallet øverst til højre er atomet masse målt i enheden "u".

Det periodiske system er bygget op efter hvor mange elektroner der i yderste skal og hvor mange skaller grundstoffet har. De grundstoffer der er øverste har deres yderste elektron i første skal, grundstoffer der har yderste elektroner i 2. skal, er også i anden række. Hver søjle i det periodiske system står for gruppe nummer,hvor dem der har tal øverste og går fra 1 til 8 er hovedgrupper, de andre er undergrupper.

Grundstoffer i hovedgruppe 1 har en elektron en i den yderste skal og grundstoffer i hovedgruppe 8 har 8 elektroner i den yderste skal.

Kemiske bindinger

Atomer eksisterer meget sjældent kun for sig selv, men indgår i stedet i forbindelser med andre atomer. Det sker i gennem en kemisk binding. Den følgende forklaring på kemiske bindinger er en forsimpling af virkeligheden. Bindinger dannes fordi atomer er mest stabile hvis deres yderste elektronbaner er fyldt op med elektroner. Dette kaldes også for oktetreglen, som siger at atomer skal have 8 elektroner i deres yderste skal for at være stabile. Dette er dog med undtagelse for H, He, Li og Be. Her er skal der være 2 elektroner i den yderste skal. Atomerne kan opnå dette ved at lave en kemisk binding. Der findes overordnet 2 forsklige typer af binding kovalente og ionske. Ioniske bindinger er hvor at elektroner bliver afgivet eller optaget af atomerne, fx kan Cl i vand optage en elektron fra Na og blive til Na^+ og Cl^-. Når Cl bliver til Cl^- er der 8 elektroner i den yderste skal, og oktetreglen er opfyldt, Na atomet af giver den yderste elektron og har 8 i den nu yderste skal og bliver til Na^+. Man kalder et atom der har flere eller færre elektroner end antallet af protoner for ioner.

Salte

Et salt er et neutralt ladet stof, som består af ioner. Det vil sige der er både er positive ioner og negative ioner, som udgør saltet. Et eksempel på et salt er køkkensalt NaCl, som består kationen (positiv ion) Na^+ og anionen (negativ ion) Cl^-.

Kovalente bindinger

En kovalnt binding sker når et atom indgår bindinger med et andet atom og der deles elektroner med et andet atom. Det fungere ved at begge atomer ser elektroner som deres eget. Dette gør fx at to hydrogen atomer kan danne en binding, og derved opfylde dubletregelen og blive stabil ved at danne H_2.

Molekyler

Et molekyle er sammensætningen af flere forskellige atomer, der er bundet sammen med kovalente bindinger. Man kan udtrykke molekyler ud fra en sumformel. Et eksempel på det er methan, som skrives som CH_4. Denne sumformel viser at molekylet består af et carbon atom og 4 hydrogen atomer, på grund af der er et nedsænket 4 tal efter. Når der kun er et af grundstoffet i molekylelet skriver man dog ikke et 1 tal, da det er implicit at der er én.

3.3 Reaktionsligninger og mængdeberegning

Introduktionen til reaktionsligninger

I kemien snakker man oftest om *reaktioner* mellem molekyler. En kemisk reaktion er, når molekyler reagerer med hinanden og danner nye molekyler. Et eksempel på en reaktion er forbrændingen af sukker, se reaktion 1

$$C_6H_{12}O_6 + 6\,O_2 \longrightarrow 6\,CO_2 + 6\,H_2O \quad \{1\}$$

Dette kaldes for et reaktionsskema eller en reaktionsligning. Det fortæller, at når der brændes ét $C_6H_{12}O_6$-molekyle (sukker), så bliver der brugt 6 O_2 (oxygen) molekyler , og der er blevet dannet 6 CO_2 (carbondioxid) og 6 H_2O (vand).

Afstemning af reaktionsligninger

For at kunne regne på, hvad ændringen er af molekyler, er det vigtigt at reaktionsskemaet er *afstemt*. Dette betyder, er der er lige meget på begge sider, i forhold til masse, ladning og i forhold til mængden af de forskellige grundstoffer. Vi kigger på reaktionsligning (2):

$$N_2 + O_2 \longrightarrow NO_2 \quad \{2\}$$

Hvis reaktionen er afstemt, er der lige mange nitrogenatomer (N) og oxygenatomer (O) på venstre side af pilen, som på højre side af pilen. Lige nu er der 2 nitrogenatomer på venstre, og kun ét på højre, det fikser vi:

$$N_2 + O_2 \longrightarrow 2\,NO_2 \quad \{3\}$$

Men nu er der ikke nok oxygenatomer på venstre side, det fikser vi også:

$$N_2 + 2\,O_2 \longrightarrow 2\,NO_2 \quad \{4\}$$

Nu er der lige mange af N på højre og venstre side, og det samme med O. Så er reaktionsligningen afstemt!

Opgaver i afstemning af reaktionsligninger

- **Opgave 3.3.1:**
Afstem følgende reaktion (sørg for der er lige mange af hvert atom på hver side):

$$N_2 + H_2 \longrightarrow NH_3 \quad \{5\}$$

- **Opgave 3.3.2:**
Afstem følgende reaktion:

$$ZnS + O_2 \longrightarrow ZnO + SO_2 \quad \{6\}$$

Introduktionen til mængdeberegning

For at kunne regne på de reaktioner, man vil udføre, f.eks. hvor meget der skal tilføjes af et bestemt stof, er der brug for mængdeberegning. Mængdebregning er viden om, hvordan man "tæller" det antal molekyler, man arbejder med. Hvis man regner i per molekyler arbejder vil man komme til at arbejde med meget store tal, fx i en liter vand er der 34563375600000000000000000 eller eller $3,45 \cdot 10^{25}$ vandmolekyler. For at gøre det meget nemmere at regne med, har man defineret enheden mol.

Stofmængde og mol

Stofmængde er, hvor mange molekyler man har af et bestemt stof. Til at beskrive antallet af molekyler bruger man enheden mol. Hvis man har et mol stof, f.eks. et mol af O_2 (ilt), har man $6,022 \cdot 10^{23}$ atomer. Man angiver altid stofmængder i enheden mol, og ikke i antal atomer, fordi at det gør det meget nemmere at regne med.

Masse

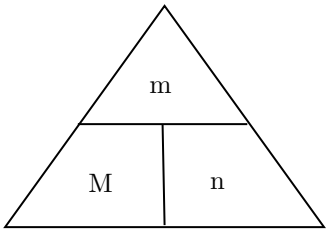
Masse er udtryk for, hvor meget materiale man har. I de fleste tilfælde og i hverdagen er massen og vægten det samme. Massen kommer fra de partikler, som atomet består af, det vil sige neutroner, protoner og elektroner. Hver af disse partikler har en bestemt masse. Masse måler man i kg, som står for kilogram. Her er gram enheden for massen, og kilo er et præfiks, der betyder 1000 gange. Det vil sige at der går 1000 gram på et kilogram. Det er samme princip inde for længde, som f.eks. med at på en kilometer går der 1000 meter.

Molarmasse

For at kunne udregne stofmængden af et bestemt stof har man brug for at kende molarmassen. Molarmassen fortæller, hvor mange gram stof der skal til, før man har et mol af det. F.eks. vejer 1 mol jern (Fe) 55,845 g. Man finder molarmassen ved at bruge det periodiske system. Da det, der giver massen kommer fra de partikler, som atomet består af, kan man finde massen af et atom ved at lægge massen af alle protonerne, neutronerne og elektronerne sammen.

Eksempel på beregning af carbons molarmasse

Carbon er nr. 6 i det periodiske system, det vil sige at det indholder 6 protoner og 6 elektroner. Afhængigt af isotopen har carbon 6, 7 eller 8 neutroner, langt de fleste carbonatomer har 6 neutroner. Det vil sige, at et carbonatom generelt har 6 protoner, 6 neutroner og 6 elektroner. En proton har en masse på $1{,}66 \cdot 10^{-27}$ kg. Massen af en mol protoner er på præsis 1 g. Dette kan man se ved at gange massen af et proton med Avogadros tal (antallet af molekyler i et mol stof), som kan ses i ligning (3.1)



$$M_{proton} = m_{proton} \cdot N_A = 1{,}66 \cdot 10^{-27}\,\mathrm{kg} \cdot 6{,}022 \cdot 10^{23}\,\mathrm{mol}^{-1} = 1\,\frac{\mathrm{g}}{\mathrm{mol}} \tag{3.1}$$

Protoer og neutroner masse næsten er ens og elektroner masse er meget lille i forhold til er masse af et mol carbon atomer 12g. Molarmassen af et bestem af atom er massen af 1 mol af atomet. Det vil sige at molarmassen af carbon er 12 $\frac{\mathrm{g}}{\mathrm{mol}}$. Molarmassen af alle grundstoffer står i det periodiske tabel. Man kan finde molarmassen af et molekyler ved at ligge molarmssen hvor hvert atom i molekyllet sammen.

Eksempel på beregning af vands molarmasse
Vand (H_2O) består af 2 hydrogen atomer og ét oxygen atom. Molarmassen af hydrogen er 1 $\frac{\mathrm{g}}{\mathrm{mol}}$ og molarmassen af oxygen er 16 $\frac{\mathrm{g}}{\mathrm{mol}}$. Der tages summen af 2 gange molarmassen af hydrogen og en gang af molarmassen af oxygen, som ses i ligning (3.2):

$$M_{\mathrm{H_2O}} = 2 \cdot M_H + 1 \cdot M_O = 2 \cdot 1\,\frac{\mathrm{g}}{\mathrm{mol}} + 1 \cdot 16\,\frac{\mathrm{g}}{\mathrm{mol}} = 18\,\frac{\mathrm{g}}{\mathrm{mol}} \tag{3.2}$$

Sammenhæng mellem stofmængde, molarmasse og masse

I afsnittet før er der blevet introduceret tre grundværdier inde for mængdeberegning:

- Stofmængde: hvor mange molekyler der er, symbol n
- Masse: vægten af stoffet, symbol m
- Molarmassen: massen af 1 mol af et vilkårligt stof, symbol M

Når to af disse værdier kendes, kan man finde den sidste. Sammenhængen mellem dem ses på den følgende skitse.

Ligning 3.3 viser sammenhængen mellem stofmængde, molarmasse og masse

$$n = \frac{m}{M} = \text{stofmængde} = \frac{\text{masse}}{\text{molarmasse}} \tag{3.3}$$

For at kunne finde én af de tre, skal man kende de to andre. Hvis man f.eks. kender vægten en en klump jern, og ved at jerns molarmasse er 55,845 g/mol kan man finde stofmængden. Der vises et eksempel på, hvordan man finder molarmassen ud fra, at man kender massen og stofmængden.

Eksempel på bestemmelse af molarmasse for en ukendt blanding gas
Man vil undersøge en gas for, om det er methan (CH_4) eller ethan (C_2H_6), hvor man har kunne køle gassen ned og målt massen af den til at være 57 g, og ved hjælp af en trykmåling er det fundet, at der er 3,4 mol gas. Der findes først molarmassen af methan og ethan.
Methan består af ét carbonatom og 4 hydrogenatomer og har en molarmasse på

$$M_{\mathrm{CH_4}} = 12\,\mathrm{g/mol} + 4 \cdot 1\,\mathrm{g/mol} = 16\,\mathrm{g/mol}$$

og ethan består af 2 carbonatomer og 6 hydrogenatomer og en molarmasse på:

$$M_{C_2H_6} = 12\,\text{g/mol} \cdot 2 + 6 \cdot 1\,\text{g/mol} = 30\,\text{g/mol}$$

Nu findes molarmassen af den undersøgte gas, og derefter bliver den sammenlignet med molarmassen af ethan og methan. Der bruges ligning (3.3) til at isolere for molarmassen:

$$M = \frac{n}{n} \cdot M = \frac{m}{M} \cdot M \cdot \frac{1}{n} = \frac{m}{n} \tag{3.4}$$

De kendte værdier indsættes i dette:

$$M_{gas} = \frac{m}{n} = \frac{57\,\text{g}}{3{,}4\,\text{mol}} = 16{,}7\,\frac{\text{g}}{\text{mol}} \tag{3.5}$$

Det vil sige, at den ukendte gas består primært af methan, da 16,7 er tættere på 16 end 30.

Opgaver i mængdeberegning

- **Opgave 3.3.3:**
  Bestem molarmassen af H_2O (vand) med hjælp fra et periodisk system.

- **Opgave 3.3.4:**
  Bestem molarmassen af $CuFeS_2$ (kobberpyrit) med hjælp fra et periodisk system.

- **Opgave 3.3.5:**
  Et stykke jern vejer 5 g (dvs. $m = 5g$), hvad er stofmængden af jern? (n =?). Brug molarmassen af jern $M = 55,8\frac{g}{mol}$.

- **Opgave 3.3.6:**
  Der haves 2 g CaO, hvad er stofmængden af CaO?

- **Opgave 3.3.7:**
  Der haves 7 mol Hg, hvad vejer denne væske?

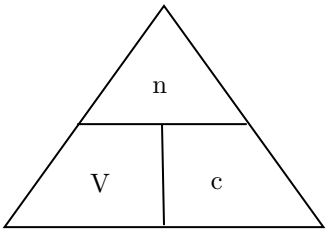
- **Opgave 3.3.8:**
  Lin har fundet en gråt metal på jorden uden for Sukkertoppen gymnasium. Hun har vejet det og konstaterer at klumpen vejer 10,0 g, og med sine seje kemividen ved hun at stofmængden er 0,178 mol. Hvilket metal er der tale om?

Koncentrationer

Når stofmængder er begrænset til et rumfang, kan det betegnes som koncentration. Hvis man f.eks. har en beholder med 1 liter vand og 1 mol af et stof opløst i vandet, er der således en koncentration på 1 mol pr. liter, og den samme koncentration kan opnås hvis man har 2 mol i 2 liter vand. Der er således lige mange molekyler pr. plads. Koncentration kan regnes med følgende formel

$$c = \frac{n}{V} \tag{3.6}$$

Hvor n er stofmængden, V er rumfanget og c er koncentrationen som er målt i enheden molær (M), som betyder mol pr. liter. Sammenhængen mellem n, V og c kan ses på følgende skitse



Figur 3.3: Regnetrekant over koncentrationer

Fordelen ved koncentrationer er, at hvis koncentrationen er kendt, vil man kunne bestemme stofmængden ud fra et afmålt rumfang.

Eksempel på beregning af koncentrationen af salt i en saltopløsning

Hvis 2 gram NaCl (bordsalt) opløses i 2 L vand, hvad vil koncentrationen af NaCl så være?

Først beregnes stofmængden. Molarmassen af NaCl er 58,5 $\frac{\text{g}}{\text{mol}}$

$$n = \frac{m}{M} = \frac{2\,\text{g}}{58{,}5\,\frac{\text{g}}{\text{mol}}} = 0{,}0342\,\text{mol}$$

Koncentrationen beregnes ud fra formlen $c = \frac{n}{V}$

$$c = \frac{0{,}0342\,\text{mol}}{2\,\text{L}} = 0{,}0171\,\text{M}$$

Koncentrationen af NaCl i saltopløsningen er 0,0171 M (som betyder mol per. liter).

Opgaver i koncentrationer

- **Opgave 3.3.9:**
  10 mol NaCl (bordsalt) er blevet hældt i et POKAL-Ikea glas på 0,35 L. Hvad er koncentrationen af salt i vandet?

●●● **Opgave 3.3.10:**
Ifølge pålidelige kilder kan et menneske drikke saltvand (NaCl i vand) med 180 $\frac{mg}{L}$. Kan man drikke glasset med saltvand fra forrige opgave?

●●● **Opgave 3.3.11:**
Der er vejet 5g $KMnO_4$ af i en målekolbe, og fyldt demineraliseret vand op til 100 mL. Hvad er koncentrationen af $KMnO_4$ i opløsningen?

3.4 Reaktionshastigheder

Introduktion

En hastighed af en reaktion er hvor mange gange reaktion forløber over en given tid. Det vil sige at hvis for et eksempel reaktion 7 sker 5 gange på et sekund er reaktionshastighed 5 reaktioner per sekund. Reaktionshastigheden er derved ikke hvor hurtige selve reaktion er for en reaktion af 3 H_2 og en N_2 der daner 2 NH_3.

$$\text{N}_2 + 3\,\text{H}_2 \longrightarrow 2\,\text{NH}_3 \qquad \{7\}$$

Det tidligere afsnit handlede om mændgeberegning, og der regner man på mange molekyler, det vil sige man tæller i antal mol i stedet for antal molekyler. Det gør man også, når man ser på reaktionshastigheder.

Reaktioner har forskelige hastigheder

Når man ser på en reaktion som reaktion 8, der beskriver dannelsen af NO_2, som er meget giftigt selv i små mol-procenter som 0,01%. Hvordan kan det være vi så ikke er konstant ved at død når der er 20% oxygen og ca. 78% nitrogen i luften? Dette er fordi reaktion for løber meget meget langsomt og at det derfor ville tage tusinder af år før det vil blive et problem i en lukket beholder.

$$N_2 + 2\,O_2 \longrightarrow 2\,NO_2 \qquad \{8\}$$

Når man måler på reaktionshastigheder gør man det oftest ved at se på hvordan koncentration af et given stof ændrer sig over tid. Reaktionshastigheden er derfor oftest i enheden 1 $\frac{\text{mol}}{\text{s}\cdot\text{m}^3}$. Reaktionshastigheden kalder man også for reaktionsraten og man bruger udtrykket r_A, som er hvor hurtigt molekyle "A" bliver dannet.

Hvad påvirker reaktionshastigheden

Koncentration

For at en reaktion kan ske, skal reaktanterne (molekyler på venstre side af reaktionspilen) mødes og ramme hinanden, det vil sige generelt, at jo større sandsynligheden er for at molekylerne mødes, jo oftere sker der en reaktion og reaktionshastigheden bliver derfor hurtigere.
Koncentration er en størrelse der tæller hvor mange molekyler der er per volumen. Det vil sige, hvis koncentration af 2 reaktanter er stor, vil de 2 molekyler mødes oftere, derfor bliver reaktionshastigheden højre når koncentration stiger.

Det betyder altså: en større koncentration af reaktanter giver en større reaktionshastighed.

Overfladeareal

Hvis en reaktion foregår på et fast materiale, har overfladearealet af det materiale en stor betydning. Meget ofte er sådan nogle reaktioner enten en forbrændingsreaktion (reaktion 9) eller opløsning af et salt (reaktion 10).

$$C(s) + O_2(g) \longrightarrow CO_2(g) \qquad \{9\}$$

$$NaCl(s) \longrightarrow Na^+(aq) + Cl^-(aq) \qquad \{10\}$$

I en forbrændingsreaktion skal et oxygen molekyle reagere med noget af det carbon der er på overfladen. Jo mere overflade der er, jo flere steder kan reaktion ske. Når man brænder kul, sender man det først gemen en mølle der maler kullet ned til en masse små partikler, dette gør at det samlet overflade areal bliver meget større.
Læseren kan overveje hvorfor overfladearealet blive større. **Hint:** se på overfladeareal og volumen af en kugle og se hvad der sker når radius bliver mindre.
Når man opløser salte sørger man også for at have det så fint som muligt. Dette er også for at øge overfladearealet. Det er derfor fint køkken slat hurtigter bliver opløst end groftsalt.

Det betyder altså: et større overfladeareal giver en større reaktionshastighed.

Temperaturen

Temperaturen af et system (blandgas, opløsning), fortæller noget om hvor meget energi der er i systemet. I gas er temperaturen et udtryk for hvor meget bevægelsesenergi molekylerene har. I en varm gas bevæger molekylerne sig meget hurtigt rundt, og i en kold gas bevæger de sig meget langsomt. Men hvad har temperaturen med reaktionshastigheden at gøre? For at en reaktion sker skal molekylerene mødes, men de skal faktisk

også have nok energi for at reaktionen sker. Når temperaturen stiger, stiger molekylernes gennemsnitsenergi og sandsynligheden for at reaktionen sker når molekylerne støder sammen stiger derfor også. En højere temperatur gør også at molekyler støder lidt oftere sammen, men det er en lille effekt i forhold til det andet.

Det betyder altså: en højere temperatur giver en større reaktionshastighed.

Reaktionsrate

Reaktionshastigheden for omdannelsen af et molekyle i en reaktion, hvor et molekyle A og B reagerer til at blive til et molekyle C kan skrives op som i ligning (3.7). Her er k det, man kalder for hastighedskonstanten, som afhænger af temperaturen og er forskellig for alle reaktioner. I ligningen er k ganget med koncentrationen af hver reaktant.

$$-r_A = k \cdot c_A \cdot c_B \tag{3.7}$$

Måling af reaktionshastigheder

Der findes mange måder at måle reaktionshastigheder på. Man kan eksempelvis tage en prøve ud fra en reaktionskolbe og lave en kemisk analyse for at se hvor meget, der er blevet omdannet. Ved at tage prøver ud på forskellige tidspunkt imens reaktionen kører, kan man finde ud af, hvor hurtigt reaktion forløber. Man kan bruge noget der hedder spektroskopi til at følge med i hvad der bliver dannet, imens reaktionen sker. Man kan for nogle reaktioner se tydeligt at reaktionen skifter farve undervejs. Hvor "farvet" en opløsning er, afhænger af koncentration af molekyler, der kan absorbere lys. Læseren kommer til at bruge to metoder for at måle reaktionshastighed: 1) ved at måle strømstyrken undervejs, og 2) ved at se på udviklingen af gas.

3.5 Forsøg 1: Forsøg med ballon

Introduktion

I dette forsøg vil der blive undsøgt hvordan, man kan måle reaktionshastigheden kvalitativt. Den reaktion, vi vil se nærmere på, er blandingen af syre med natriumhydrogencarbonat (bagepulver). Natriumhydrogencarbonat er det, man kalder for en sammensat ion. Den består af Na^+,H^+ og ${CO_3}^{2-}$. Hvis man opløser bagepulveret i vand og derefter tilføjer mere syre (H^+), bliver der dannet bobler og frigivet gas til omgivelserene. Dette er fordi, der bliver dannet CO_2. Reaktion kan ses i ligning 11.

$$2\,H^+(aq) + {CO_3}^{2-}(aq) \longrightarrow CO_2(g) + H_2O(l) \qquad \{11\}$$

Hver gang, at reaktionen forløber én gang, bliver der dannet ét gasmolekyle. Det vil sige at jo hurtigere at reaktionshastigheden er, jo hurtigere bliver der udviklet gas.

Formål

At undersøge konceptet af reaktionshastighed og hvilke faktorer, der har en effekt.

Sikkerhed

I laboratoriet skal der altid bruges sikkerhedsbriller, langt hår skal være sat op, og man skal have lukket kittel på. Når man forlader laboratoriet, skal man altid vaske hænder, før man går ud. Hvis man spilder kemikalie på hænderne, skal man skylle med rigeligt vand. Før øvelsen starter, bliver der gennemgået sikkerhedsregler dybere. Kemikalieaffald skal som tommel-fingerregel opsamles i kemikaliedunke. I denne øvelse arbejdes der dog kun med bagpulver og edikke. I en del af forsøget skal I bruge en varmeplade, her er det meget

vigtigt, at I passer på, at I ikke brænder jer! I kommer til at bruge eddikesyre, som er ætsende og er meget farligt at få i øjnene. Det er derfor ekstra vigtigt, at I beholder jeres briller på hele tiden, når I er inde i laboratoriet.

Materialer

- Kemikalier
  - Natriumhydrogencarbonat ($NaHCO_3$)
  - 2 M Eddikesyre (CH_3COOH)
- Glasudstyr
  - 9 stk. 150 ml koniske kolber
  - 10 ml måleglas
  - 100 ml måleglas
  - 2 stk. 250 ml bægerglas
- Generelt udstyr
  - 10 balloner
  - Varmeplade
  - Isbad

Figur 3.4: Ballonforsøget med 2g natriumhydrogencarbonat og med 3 forskellige koncentrationer af eddikesyre.

Metode

Del 1

I den første del bliver der testet hvordan reaktionshastigheden ændrer sig, når koncentrationen ændrer sig. Tag 3 stk. 150 ml koniske kolber med 2g $NaHCO_3$ i hver og tilføj henholdsvis 10 ml, 20 ml og 40 ml vand i hver sin kolbe. Dette gør I ved at fylde demineraliseret vand op i et 250 ml bægerglas og derefter overføre vandet til det 100 ml måleglas, og afmåle den mængde vand i skal bruge. Ryst kolberne forsigtigt for at $NaHCO_3$ bliver opløst i vandet.

Tænk jer om: Hvad regner I med, at der kommer til at ske, når der bliver tilsat eddikesyre, og hvilken af kolberne, tror I, der hurtigst får fyldt ballonen op? Når det er gjort, skal I så i gang med at foretage målingerne.

Tag én konisk kolbe ad gangen. Et gruppemedlem har en ballon klar og udtrukket til at tage ned over den koniske kolbe. Et andet gruppemedlem har fyldt 10 ml 2 M eddikesyre op i et måleglas og er klar til at overføre til den koniske kolbe.
Når det ene gruppemedlem har tilføjet eddikesyre til opløsningen, sætter det andet medlem ballonen på kolben og starter et stopur. I stopper uret, når ballonen stopper med at vokse. Dette gentager I for de 2 andre kolber.
Når I har gjort det for de 3 kobler, gentager I forsøget.

Del 2

I første del af forsøget undersøgte I, hvordan koncentration har en effekt på reaktionshastighed. Nu skal I undersøge, hvordan temperaturen har en effekt på reaktionshastighed.

Ligesom i første del af forsøget skal I tage 3 stk. 150 ml koniske kolber med 2g $NaHCO_3$. Her skal i tilføje 20 ml vand til hver af dem. Igen rystes kolberne forsigtigt, så alt $NaHCO_3$ opløses. Hent et isbad og en varmeplade. Sæt én af kolberne ned i isbadet og én af kolberne på varmepladen, og indstil varmepladen til at være 50^oC.
Start med at udføre forsøget på kolben, der ikke er opvarmet eller er nedkølet.
Det udføres på samme måde, som hvor I undersøgte for koncentrationens indflydelse på reaktionshastigheden. Dernæst tager I kolben på varmepladen forsigtigt af varmepladen med et par varmehandsker, og så udfører I forsøget. Til sidst udfører I forsøget på kolben i isbadet, hvor I tager kolben op af isbadet og udfører forsøget.

Resultatbehandling

Brug tabel 3.1 til at notere målinger fra forsøgene, ved tempraure skal i blot skriv om det er varmt, stuetempratur eller koldt.

Tabel 3.1: Tabel til notering af tiden det tager af udvide ballonen og forholdene for hver kolbe.

Forsøg 1	Kolbe 1	Kolbe 2	Kolbe 3
Temperatur			
Tilsat vand (ml)			
Koncentration($\frac{mol}{L}$)			
Tid (s)			
Forsøg 2	Kolbe 1	Kolbe 2	Kolbe 3
Temperatur			
Tilsat vand (ml)			
Koncentration($\frac{mol}{L}$)			
Tid (s)			
Forsøg 3	Kolbe 1	Kolbe 2	Kolbe 3
Temperatur			
Tilsat vand (ml)			
Koncentration($\frac{mol}{L}$)			
Tid (s)			

3.6 Forsøg 2: Måling af reaktionshastigheder med elektrokemi

I dette forsøg vil der blive brugt elektromkmei til at bestemme reaktionshastigehden.

Teori

Den reaktion som der bliver undersøgt er reaktion imellem zink og kobber.

$$\mathrm{Zn(s) + Cu^{2+}(aq) \longrightarrow Cu(s) + Zn^{2+}} \quad \{12\}$$

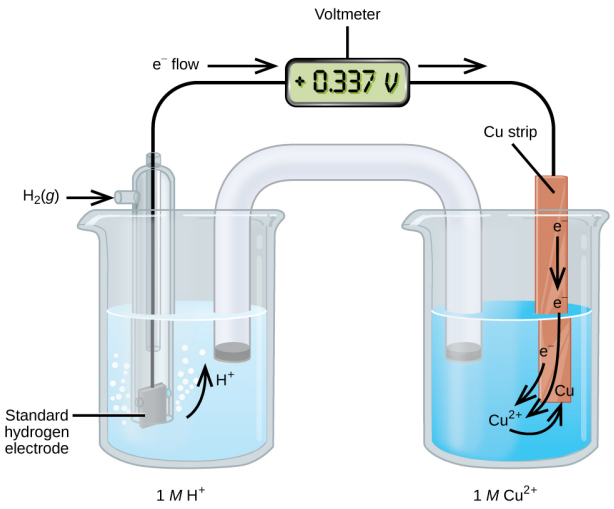
Man kan også dele reaktionen op i det der hedder **katode-reaktionen** og **anode-reaktionen**. Katodereaktionen er den reaktion hvor man har en positiv ion der mister sin ladning og anode-reaktionen er der hvor et metal bliver til en positiv ion. Når man deler reaktionen op på den måde, kalder man det for halvcellereaktion. Hvilket er fordi at man forestiller sig at reaktionen sker i to "dele" i en elektrokemisk celle.

$$\mathrm{Zn(s) \longrightarrow Zn^{2+}(aq) + 2\,e^-} \quad \{13\}$$

$$\mathrm{Cu^{2+}(aq) + 2\,e^- \longrightarrow Cu(s)} \quad \{14\}$$

Reaktion 13 er anode-reaktionen og 14 er katode-reaktionen. Hvis man ligger de to reaktion sammen får man reaktion 12.

I halvcellereaktionerne kan man se at der indgår elektroner. Det betyder at hvis man fx tilføjer elektroner til en zink plade dyppet i en opløsning, kan man få zink atomer til at gå i opløsning. Problemet er bare hvordan man får tilføjet elektroner! Men det gør man simplethen bare ved at sende strøm igennem cellen. Man kan meget simplistiske tænkte på elektrisk strøm som en flod af elektroner der går igennem en ledning.



(a) En skitse af en elektrokemisk celle.

(b) En opstilling af en elektrokemisk celle, her med kobber og aluminium.

Figur 3.5: Billeder af elektrokemiske celler.

En elektrokemisk celle kan man sætte strøm igennem, ved at man forbinder hver halvcelle med en ledning der kan transportere elektronerne imellem de 2 celler, som vist i figur 3.5. Hvis man forbinder et multimeter til kredsløbet, kan man måle både spændingen (volt), som siger noget om ligevægten af reaktionen, men man kan også måle strømstrykken (ampere). Strømstyrken (I) er defineret som hvor mange ladninger der strømmer igennem ledningen per sekund. Ud fra halvcellereaktionerne kan man aflæse at der for reaktionen 12, skal overføres 2 elektroner hver gang at en reaktion sker. Derfor

kan man ud fra strømstyrken måle hvor hurtigt reaktionen sker. Dette kan udregnes ved at bruge definition på strømstyrke, i ligning (3.8), hvor at I er strømstrykken, Q er ladningen og t er tiden.

$$I = \frac{Q}{t} \tag{3.8}$$

Ved at udnytte at der for hvert reaktion skal bruges 2 elektroner kan man opskrive hvor meget der bliver dannet af Cu per sekund, som vil blive beskrevet som ν og som svare til rekaktionshastigheden.

$$I_{CuZn} = \frac{Q_{CuZn}}{t} = \frac{n_{Cu,dannet}}{2t} F = \nu \frac{F}{2} \tag{3.9}$$

$$\nu = 2\frac{I_{CuZn}}{F} \tag{3.10}$$

I ligning (3.10) er der beskrevet hvordan man kan regne reaktionshastigheden ud fra strømstyrken. Her er F Faradays konstant som er den samlede ladning af en mol elektroner og har en værdi på $96\,485\ \frac{\mathrm{C}}{\mathrm{mol}}$.

Formål

At undersøge konceptet af reaktionshastighed via elektrokemi kvantitativt, ved at undersøge reaktion imellem kobber og zink.

Sikkerhed

I laboratoriet skal der altid bruges sikkerhedsbriller, langt hår skal være sat op, og man skal have lukket kittel på. Når man forlader laboratoriet, skal man altid vaske hænder, før man går ud. Hvis man spilder kemikalie på hænderne, skal man skylle med rigeligt vand. Før øvelsen starter, bliver der gennemgået sikkerhedsregler dybere. Kemikalieaffald skal som tommel-fingerregel opsamle i kemikaliedunke. Det er ekstra vigtigt, at I beholder jeres briller på hele tiden, når I er inde i laboratoriet.

Materialer

- Kemikalier
  - Zink plade (Zn)
  - Kobber plade (Cu)
  - 2 M kobbersulfat $CuSO_4$
  - 2 M zinksulfat $ZnSO_4$
- Glasudstyr
  - 2 stk. 150 ml bægerglas
- Generelt udstyr
  - Multimeter
  - Varmeplade
  - 2 krokodillenæb
  - 2 ledninger
  - Saltbro
  - Stativ

Metode

Overfør 100-150 ml 2 M $CuSO_4$ til et 150 ml bægerglas, og overfør 100-150 ml 2 M $ZnSO_4$ til et andet 150 ml bægerglas. Hent en saltbro, som er fremstillet i forvejen, og forbind væsken i de to bægerglas med denne. Dette skal gøres **forsigtigt**. Sæt et krodillenæb på en kobberplade og forbind den med en leding til multimeteret og tag derefter en ledning fra multimeteret og forbind det til et krokodillenæb der sidder på en zinkplade.

Sæt multimeteret til at måle strømstyrken i den der måler μA og forsøget kan nu startes. Sænk stille og roligt zink pladen ned i $ZnSO_4$ opløsningen og kobber pladen stille roligt ned i $CuSO_4$ opløsningen. Læg mærke til om strømstyrken ændrer sig imens at I sænker pladerne ned i opløsningerne.
Når pladerne er sat ned i opløsningen, lader i dem stå i noget tid hvor I holder øje med om strømstyrken ændrer sig. Imens det sker, kan i skifte til at måle spændingen (måles i volt) og notere hvad den er.

Temperatur effekt på den elektrokemisk celle

Forsøget gentages nu hvor at opstilling blive sat på en varmeblok, hvor i sætter temperaturen til at være på 50 ^oC. Vent til at opløsning er noget op til at opløsningen er 50 ^oC varm, og foretag de samme målinger som i forsøget før.

Resultatbehandling

Tabel 3.2: Tabel til notering af strømstrykken og spændningen.

	Zn og Cu plader kun lidt døbbet	Zn og Cu plader havlt nedsunket	Fludnedsunket	Efter 10 min
Spænding(V)				
Strømstyrke(mA)				
Spænding(V) på varmepladen				
Strømstyrke(mA) på varmepladen				

3.7 Titrering af Fe(II) med $KMnO_4$

Indledning

I kemien er det tit nødvendigt at bestemme stofmængden eller koncentrationen i en ukendt prøve. En af de analysemetoder der bruges er en titrering. I denne titrering skal i bestemme koncentrationen af Fe(II) i en opløsning. Dette kan f.eks. være relevant hvis man skal bestemme mængden af Fe(II) i den jernkatalysator der benyttes industrielt når man skal producere ammoniak. Ammoniak bruges både til gødning og til bomber.

Figur 3.6: Billede af $KMnO_4$ salt

I titreringen benyttes der $KMnO_4$ (se 3.6) som *reagens* og som *indikator*. Det er $KMnO_4$ der reagerer med Fe(II)'en, og også den der sørger for at opløsningen skifter farve når al Fe(II)'en i den ukendte opløsning er reageret med. Dette tidspunkt kalder man ækvivalenspunktet - det punkt hvor man har tilsat lige så meget $KMnO_4$ som der er Fe(II) i opløsningen. Stoffet er også indikator, da det vil føre til at opløsningen skifter farve ved ækvivalenspunktet.

Følgende reaktion sker i kolben når der tilsættes $KMnO_4$:

$$5\,Fe^{2+} + MnO_4^- + 8\,H^+ \longrightarrow Mn^{2+} + 5\,Fe^{3+} + 4\,H_2O \qquad \{15\}$$

MnO_4^- er en stærkt farvet ion (pink), men hvis der stadig er Fe^{2+} tilbage i opløsningen reagerer ionerne og bliver til de farveløse Mn^{2+}. Her er det vigtigt at huske at for hver 5 Fe(II)-ioner reagerer kun 1 MnO_4^-. Dvs. ved ækvivalenspunktet hvor der er **en blivende pink farve** gælder ligning 3.11:

$$n(Fe^{2+}) = 5 \cdot n(MnO_4^-) \qquad (3.11)$$

Denne skal i bruge til at beregne stofmængden i den ukendte opløsning.

Formål

Formålet med øvelsen er at betsemme koncentrationen af Fe(II)-ioner i den ukendte opløsning.

Forsøgsvejledning

Til titreringen vil der blive brugt $KMnO_4$ som titrand, hvor reaktionen 15 vil ske. Når der er tilsat en ækvivalent mængde MnO_4^--ioner til den ukendte opløsning vil der være en blivende pink farve.

Sikkerhed

I laboratoriet skal der **altid** bruges sikkerhedsbriller, langt hår skal være sat op, og man skal have **lukket** kittel på. Når man forlader laboratoriet, skal man **altid** vaske hænder før man går ud. Hvis man spilder kemikalie på hænderne skal man skylle med rigeligt vand. Før øvelsen starter, bliver der gennemgået sikkerhedsregler dybere.

Kemikalieaffald skal som tommel-fingerregel opsamles i kemikaliedunke. I denne øvelse laves der dog blot Fe^{3+}, Mn^{2+} og vand. Disse kan alle hældes ud i vasken. Er der derimod stadig MnO_4^--ioner tilbage, må det **ikke** hældes ud, da permanganationer forurener havmiljøet. Det vil sige at **hvis man har titrand i overskud må det ikke komme ud i vasken**.

Materialer

I listen kan der ses alle de materialer der skal bruges til forsøget:

- Kemikalier og opløsninger
  - 0,02 M $KMnO_4$-opløsning
  - 25 mL ukendt $FeSO_4$-opløsning
- Glasudstyr
  - 25 ml burette
  - To 50 ml bægerglas
  - 50 ml konisk kolbe
  - 25 mL glaspipette
- Resterende udstyr
  - A-fodsstativ
  - Buretteholder
  - Pipettebold
  - Engangspipetter
  - Tragt

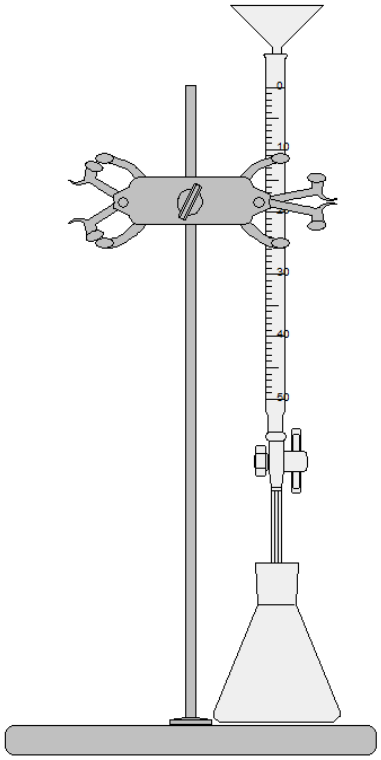
Metode

Sæt buretten fast til stativet ved at bruge buretteholderen, sørg for at buretten er placeret lodret og at hanen er lukket. Tragten sættes i toppen af buretten og kolben stilles under. Se 3.11 for billede af opstilling. Når du hælder titrand ($KMnO_4$) op skal der stilles et tomt bægerglas under buretten.

Overfør ved hjælp af en fuldpipette 25 ml af den ukendte opløsning (prøven) af $FeSO_4$ til en 50 ml konisk kolbe. Overfør ca. 25 ml af 0,02 M $KMnO_4$ til et 50 ml bægerglas. Tag begge kemikalier tilbage til opstillingen af forsøget.

Kontrollér at hanen på buretten er lukket og derefter fyld buretten op med 0,02 M $KMnO_4$. Placér et 50 ml bægerglas under buretten, åbn for buretten og lad et par milliliter løbe igennem, luk derefter for buretten. Dette er for at at fjerne luftboblen i bunden af buretten. Aflæs så startvolumen på buretten og skriv den ned.

Placér den koniske kolbe med prøven under buretten. Start titreringen ved at åbne stille og roligt for buretten. Der titreres optimalt ved at swirle den koniske kolbe en gang imellem så væsken blandes helt. Fortsæt indtil, at der opnås et permanent farveskifte til pink, hvor ækvivalenspunktet da er nået. Når ækvivalenspunktet er nået, noteres den nuværende volumen af buretten. **Det er det brugte volumen 0,02 M $KMnO_4$ der skal bruges til at beregne koncentrationen af den ukendte $FeSO_4$-opløsning**.



Figur 3.7: Figur af titreringspostilling med tragt, burette i stativ, og kolbe.

Resultatbehandling

Notér først det tilsatte volumen af NaOH i tabellen og beregn derefter koncentrationen af NaOH.

Tabel 3.3: Tabel til notering af tilsat volumen og beregning af koncentration ved titrering.

	Titrering 1	Titrering 2	Titrering 3
$V(KMnO_4)_{start}$			
$V(KMnO_4)_{slut}$			
$V(KMnO_4)_{tilsat}$			
$n(KMnO_4)$			
$n(Fe^{2+})$			
$c(Fe^{2+})$			

Kapitel 4

Biologi

4.1 Evolution/Økologi

Evolution betyder forandring, og er som bekendt også studiet af livets udvikling. Evolution bygger på samspillet mellem miljøet og jordens organismer. Rygraden i evolutionstudiet er naturlig selektion, hvorfra den verdensberømte sætning "survival of the fittest" er baseret på. Studiet, evolution, er i stor samspil med økologi, da økologi undersøger organismers interaktion mellem hinanden og miljøet, og hvordan disse interaktioner kan lede til naturlig selektion, og derved resultere i forskellig geno-/og fænotyper. Ved økologi studeres forholdet mellem miljøet og dets organismer, både i organisk og inorganisk form, samt deres direkte interaktioner med og mellem hinanden. For at overleve, bliver en organisme uanset størrelse og tilpasning nødt til at uddrage ressourcer fra miljøet uden selv at ende som føde for andre organismer, så denne kan videregive sit arvemateriale til den næste genneration. Organismer agere i forskellige økosystemer som de er tilpasset til.

Økosystemer indeles i to interagerende områder. Det levende (biotiske), alle levende organismer, dyr og mikrober, og det ikke levende (abiotiske), med dette forstås økosystemest kemiske og fysiske komponenter.

Når evolution drives af klimaændringer, er der tale om, at de abiotiske faktore ændrer de biotiske, men det kan også gå den anden vej, de biotiske faktore kan ændre forholdet af de abiotiske. Eksempelvis kan træerne i en skov ændre på lysforholdet, fugtigheden, samt mængde og bevarelse af næringsstoffer. Derved er disse i et konstant komplekst sammenspil. Den ene kan derfor ikke tages i betragtning uden den anden. De mange grupper af organismer er bestående af mange forskellige arter, der hver især er tilpasset et specifikt økosystem. Grupper af samme individer i et økosystem kaldes en population, som på positiv eller negativ vis, interagerer med populationer af andre arter. Alle populationer i et økosystem kaldes samlet et fællessamfund

Figur 4.1

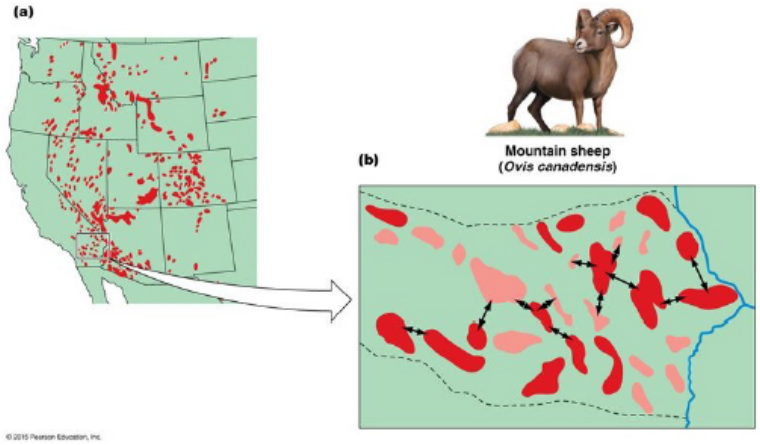
De Fysiske Miljøer

I et økosystem interageres der på mange forskellige niveauer. Disse niveauer kan indeles i et hierarki bestemt efter størrelse

Herfra kan man se, at alle de interag erende niveauer befinder sig inden for biosfæren. Dette er således det største økologiske niveau, der kan observeres på. Hvilket niveau man kigger på, afhænger af hvad der ønskes undersøgt, f.eks. er det en undersøgelse af

samspillet mellem arter i et specifikt økosystem eller er det samspillet mellem flere arter i forskellige økosystemer, der undersøges.

Individet og Populationer



Figur 4.2

Helt generelt siges det, at det er individet, der driver populationen. I form af at det er den enkelte med de bedst tilpassede gener, der afgør populationens udvikling. Men hvad der oftest overses, er at det i høj grad også er populationen der driver individet. Dette sker oftest i form af trends eller sexuel adfærd, som " lokker" individet til at agere på en sådan måde, der kan påvirke populationen i en positiv eller negativ retning. Derfor siges der, at det er individet der er kernen når det kommer til studiet af økologi, da det er individet, der agere, og respondere til miljøet og andre organismer. Det er død og fødsel af individer, der driver en populations dynamikkerne. Individet kan derfor anvendes i forhold til at undersøge årsagen til udviklingen, samt hvornår populationsudviklingen havde sin begyndelse. Men når populations udvikling og naturlig selektion opfattes som helhed siger man, at genetisk variation sker i niveau med populationen. I det tidsmæssige forhold har genetisk variation lille betydning, grundet individets korte levetid. Individets genetiske variation fra andre kan derfor ikke tages i betragtning i forhold til drivkraften af en population, da denne ændring sker nu og her. Det er derimod vurdering af hvordan den genetiske variation er organiseret i en population, der skal betragtes for at danne et overordnet forståelse af tilpasning gennem naturlig selektion.

En dyreart er meget sjældent udgjort af en enkelt population, og er som regel udgjort af flere større eller mindre populationer, spredt over landarealer. Disse forskellige grupper af populationer kaldes enkeltvis sub-populationer. Eksempel: Danmarks bestand af svale består a to subpopulationer, hvor den ene befinder sig i Gladsaxe kommune og den anden i Odense. For at sub-populationer kan tilhøre en bestemt dyre population, skal sub-populationen kunne dele genetisk arvemateriale med andre sub-populationer, og derved lede en genetiske drift i den overordnede population. Grundet eksistensen af sub-populationer kan genetisk variation finde sted i form af et hierarkisk niveau. Det kan nemlig finde sted inden i sub-populationerne og mellem sub-populationerne.

Naturlig Selektion

Naturlig selektion er en grundsten i evolutionslæren, men også i biologien generelt. Helt simpelt betyder det, at fra naturens side vil dårligt tilpasset individer blive fravalgt over tid, hvilket føre til en overordnet udvikling indenfor populationen/arten/gruppen. Teorien blev grundlagt af Chales Darwin og Alfred Russel Wallace omkring 1859. Darwin er den meste kendte af de to for sit værk "Origin of the species" hvori han udforsker bl.a. en gruppe finker på Galapagos. Det viser sig, at fugle fra en slægt havde træk så

forskellige at de kunne anses for seperate arter, selvom de boede indenfor et meget lille geografisk område. Det samme viste sig at være tilfældet med landskildpadder i Galapagos. Wallace fandt samme mønster, men i sommerfugle. Begge to nåede frem til teorien om, at den naturlige variation blandt individer over tid vil betyde, at de variationer som giver en fordel vil blive beholdt i en population, mens andre ville forsvinde. Dette kan føre til nye arter. Den måske hyppigste måde dette sker på er hvis en population af individer bliver splittet i to, og at disse to udsættes for forskellige miljøer. I tilfældet med finkerne, så var fugle fra samme art kommet til øerne, og havde så dannet seperate populationer. Hvad der så formentlig er sket er at en af populationerne har udviklet sig at spise nødder og har store næb til at knuse skaller, andre til at spise insekter og har smallere næb til at nå ind i bark og revner. Over tid er disse forskelle blevet så markante, at der er tale om forskellige arter.

Når man snakker naturlig selektion er det vigtigt at huske to ting;

- Det handler om tilpasningsdygtighed til sit miljø, ikke overlegenhed indenfor en bestemt enhed
- Det er ikke altid til at forudsige hvad en gavnlig mutation er

I det første tilfælde, hvis vi forestiller os to planter. Den ene er markant bedre til at optage vand end den anden. Dette er en fordel i områder med tørke, men hvis der er rigeligt nedbør, så vil den energi der bliver brugt på at unødigt opbevare vand betyde at planten vil gro langsommere, og derved formere sig langsommere end den anden plante. Over tid vil den plante som ikke optagere vand helt så godt nok klare sig bedre, fordi den har ramt det rette niveau af vandoptag. For at se på den anden, tænk på påfuglen. Den har en enorm farverig fjerdragt, hvilket gør den nem at spotte for rovdyr. Dette er en ulempe. Til gengæld giver en stor og flot fjerdragt hannen større mulighed for at tiltrække en mage. Tiltrækningen af en mage vejer her tungere end at kunne skjule sig. EN årsag til dette er, at for at en mutation kan blive spredt i en population, kræver det, at det individ der oprindeligt fik den, kan finde en mage og reproducere. Man kan sige, at der er en konstant balancegang mellem hvilke træk der er vigtigst.

For sjov, forestil jer de dansene fuglene fra paradisøerner. På disse øer er der en mængde fuglearter, hvor hannerne er kendt ikke bare for deres prangende fjerdragt, men også på at bruge uanede mængder af tid på at danse. Dette inkludere at fjerne blade fra deres scene, jage andre hanner væk fra særligt gode dansesteder og bruge timer på at øve sig. Alt dette er for at tiltrække en mage. Kan du tænke dig til en årsag til at disse fugle kan dedikere så meget energi og variation til denne ene ting.

4.2 Genetik

Genetik er, som navnet antyder, studiet af gener. Alle organismer opbevarer deres gener i form af det meget kendte molekyle DNA, hvis fulde navn er deoxyribosesyre. Disse nedarves fra forældre tilfældigt. Begge forældre har 23 kromosompar og dermed 46 i alt. Et tilfældigt kromosom fra hvert par gives videre, så barnet ender med at have de samme par, men altså med kromosomer fra begge forældre. Der er lige stor sandsynlighed for hvilket af de to kromosomer i forælderens par, der gives videre.

Gener er interessante, da rigtigt mange af menneskers egenskaber kan findes i generne. Generne styrer eksempelvis højde og øjenfarve, men også rigtig mange sygedomme kan findes i generne.

DNA og Kromosomer

DNA molekyler er, relativt set, enormt lange: hele 5 cm per styk. Dette virker ikke som så meget, men til sammenligning er de 0,0000002 cm tykke. Da der skal være ca. 2,3 meter DNA i hver celle i kroppen, så skal det naturligtvis rulles op. Helt simpelt foregår dette ved at rulle det op i en struktur som kaldes et kromosom.

Kromosompar og Alleler

Det forholder sig sådan at (næsten) alle kromosomer kommer i par, hvor der er gener for de samme ting på hver del af a parret. Det vil sige, at der i langt de fleste tilfælde vil være to versioner af samme gen, og disse kaldes alleler.

Et allel er altså en version af et gen, og man har som udgangspunkt to alleler for hvert gen med mindre man har XY-kønskromosomerne (det vi typisk kalder en dreng), for her har X og Y kromosomet ikke kun de samme gener.

Genotype og Fænotype

En organismes, for eksempel et menneskes, genotype for et gen betegnes ved de (typisk) to alleler, som den har for det.

Fænotypen er derimod det fænomen, der forekommer på baggrund af genet.

Stamtræer

Da gener gives videre fra forælder til børn, så kan man tegne et træ, der viser hvordan nogle individer (børnene) har arvet deres gener fra andre (forældrene). Et helt basalt stamtræ kan ikke bruges til så meget, men holder man øje med et bestemt karaktertræk, en fænotype, i en stor familie, så er det ofte muligt at regne ud hvilke gener, genotype, der har forårsaget dette.

Mendels Ærteplanter

Faderen af genetik, Gregor Mendel, beskrev allerede i midten af 18-hundredetallet den første genetiske nedarvning.

Her var der tale om ærteplanter, som kunne være hvide, lyserøde eller noget midt i mellem. Her opdagede han efter et par forsøg, at man kunne have en ærteplante med to alleler for hvid og en med to alleler for rød, og at ærterne fra disse ville frembringe nye planter med farven midt i mellem.

Dette sker, da de nye planter har arvet et enkelt hvidt allel og et enkelt rødt allel fra sine "forældre". Dog bliver det straks mere kompliceret, hvis disse nye "blandede ærteplanter" formerer sig.

Scenarie 1: Forælder 1 giver sit "hvide kromosom", og det samme gør forælder 2. Da har "barnet" 2 hvide alleler, og vil dermed blive hvidt. Da der er 50 procent sandsynlighed for at disse to ting sker, så er der i alt 25 procent sandsynlighed for dette scenarie.

Scenarie 2: Samme som scenarie 1, her er det blot det "lyserøde kromosom".

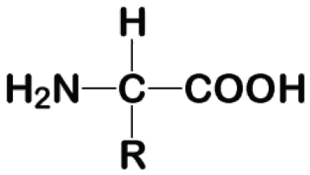
Scenarie 3: Forælder 1 giver sit hvide, forælder 2 giver sit lyserøde *eller* forælder 1 giver sit lyserøde, forælder 2 giver sit hvide. Der er 25 procent sandsynlighed for de to scenarier hver især, og lagt sammen giver det dermed 50 procent sandsynlighed.

4.3 Enzymer

Enzymer, kort fortalt, er en gruppe proteiner som fungere som et klippe-klister system. De ses oftest i reaktioner hvor de skaber eller bryder bindinger mellem andre molekyler. Ser man på det kemisk, så klassificerer man enzymer som katalysatorer. En katalysator indgår i en reaktion ligesom reaktanter og produkter, men ændres *ikke* i løbet af reaktionen. Deres funktion er derimod og få reaktionen til at forløbe lettere. For eksempel, hvis man forstiller sig en reaktion hvor reaktant A bliver nedbrudt til produkt B og C. Naturligt vil reaktionen måske kun kunne ske ved omkring 70 °C og tage et par år, men ved brug af enzymer kan reaktionen foregå ved 30 °C og i løbet af få minutter. Det er denne egenskab som gør enzymer livsnødvendige.

Struktur og Funktion

Som med alt andet biologisk, så er skruktur afgørende for funktion. Enzymer er proteiner og er bygget op af aminosyrer. Deres struktur er bestemt udfra den kæde af aminosyrer som de består af. Der findes 20 aminosyrer som alle er bygget op over samme grundformel



Figur 4.3: Generel struktur af aminosyre

Det afgørende for strukturen er den variable R-gruppe. R-gruppen kan være alt fra et enkelt hydrogen til en ring. Alt efter hvad R-gruppen indeholder klassificere man aminosyren som stor eller lille, polar eller u-polar, positiv eller negativ.

Det er interaktionerne mellem disse R-grupper som bestemmer et proteins form i 3D, f.eks. vil positive og negative R-grupper bevæger sig mod hinanden, hydrofobe R-grupper vil samle sig for at undgå hydrofile R-grupper, cysteiner kan danne svovlbroer mellem hinanden og lignende. Alt dette vil få kæden af aminosyrer til at folde sig sammen til en "klump".

Foldningen af et enzym, eller proteiner i det hele taget, foregår enten i cytoplasmaet eller i organnellet ER. En af hovedgrundende til, at det foldede enzym ikke straks begynder at klippe og klistre i alt hvad det er i nærheden af, skyldes at enzymer er uhyre specifikke. Dette betyder, at de kun kan arbejde på bestemt(e) substrater. Specificiteten kan forklares med Lås-og-Nøgle modellen.

Lås-og-Nøgle Modellen

Lås-og-Nøgle modellen forklarer enzymers specificitet ved sat betragte enzymer og deres substrat som et lås og nøgle par. Den del af enzymet som udfører katalysen kaldes et aktive sæde. Denne del ligger ofte i en fordybning i enzymet og kun molekyler som passer ned i fordybningen, ligesom en nøgle i en lås.

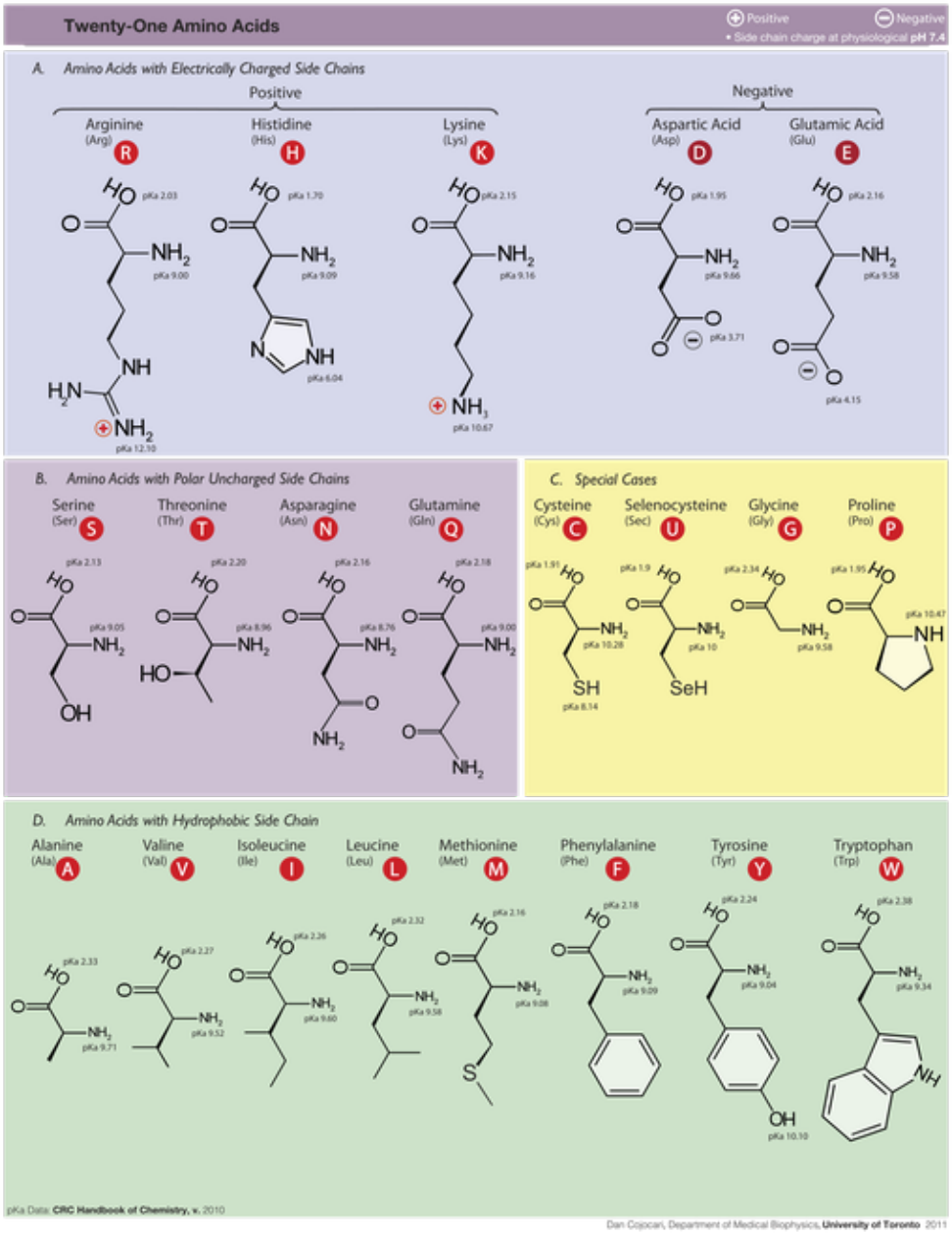
Induced-Fit Modellen

Teorien om det induceret fit/pasning, er en nyere viderebygning på Lås-og-Nøgle modellen. Teorien går ud på at enzymer er en lille smule plastiske, dvs. de kan ændre form en lille smule. Effekten af dette er, at når en et substrat binder til enzymet, så vil enzymet kunne ændre form en lille smule hvorefter substratet vil passe til det aktive sæde. Forskellen på denne teori og Lås-og-Nøgle modellen er, at ved Induceret Fit så kan et enzym have flere substrater, så længe de ligner hinanden. For eksempel kan nogle proteaser nedbryde forskellige proteiner, hvorimod laktase kun katalysere nedbrydningen af laktose (mælkesukker)

Klassificering

Enzymer klassificeres efter hvad de gør, og navngives som regel efter deres substrat. Her er tilføjet en tabel med nogle af dem man hyppigst støder på.

Som sidenote, enzymer ender som regel på -ase. Af de overstående er hydrolaser og transferaser overklasser. F.eks. alle de klasser som er listet under transferase er transferaser. Disse navngives som regel substrat transferase. Dette kan enten være transferase eller den specifikke type af transferase.



Figur 4.4: R-grupper og deres funktioner

Figur 4.5: Lås-og-Nøgle modellen

Kinetik

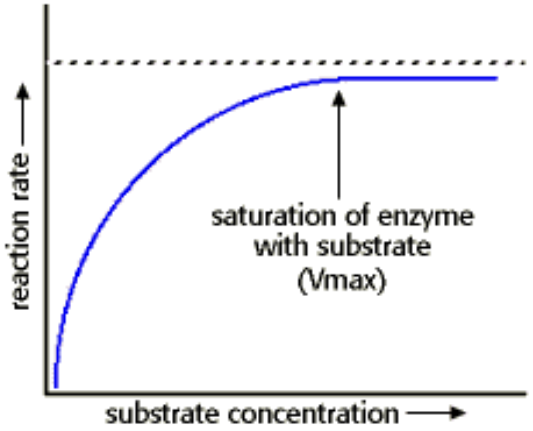
Kemisk set er kinetik relateret til reaktioner og særligt reaktionshastigheder. I forhold til enzymer, så bruger vi kinetik til at beskrive reaktionen mellem et enzym og dets substrat, samt de forhold som påvirker reaktionen.

Tabel 4.1: Tabel over enzymklasser og deres funktion

Navn	**Function**
Protease	Klipper peptidbindinger i stykker (proteiner)
Lipase	Nedbryder fedtstoffer
Amylase	Nedbryder stivelse til sukker
Polymerase	Danner lange kæder af nukleotider
Hydrolase	Bruger vand til bryde kemiske bond
Transferase	Overfører et molekyle/en gruppe til et andet
Kinase	Tilføjer en fosfatgruppe
Fosfatase	Fjerner en fosfat gruppe
Acetylase	Tilføjer en acetylgruppe
De-acetylase	Fjerner en acetylgruppe
Metylase	Tilfører en metylgruppe
De-metylase	Fjerner en metylgruppe

Koncentration af Enzym og Substrat

Et af de vigtigste forhold som vil påvirke en reaktion er koncentrationen af de forskellige elementer. Generelt set, finder en enzym katalyseret reaktion sted i en opløsning med frie elementer. Det betyder, at både enzymer og deres substrater diffundere rundt indenfor et afgrænset område. For at en reaktion imellem den kan finde sted, så skal de støde ind i hinanden. Deraf følger at jeg større koncentrationen af enzym og substrat er, desto oftere vil de mødes og en reaktion finde sted. Dette beskrives af følgende graf;

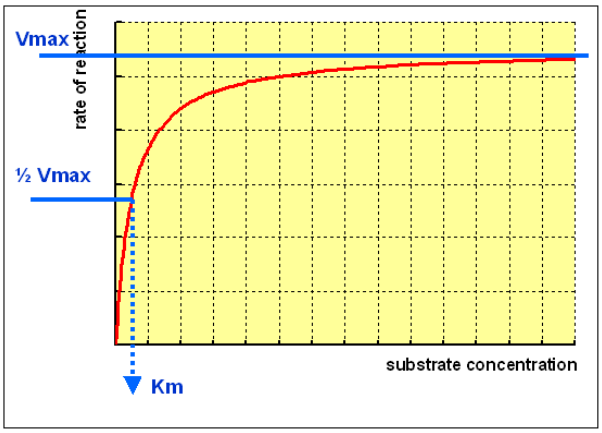


Figur 4.6: Forhold mellem reaktionshastighed og Substratkoncentration

Fig 4.6 er nok den mest kendte af enzymkinetik graferne. Den beskriver at ved en låst enzym koncentration så vil mængden af produkt produceret per tidsenhed stige med substrat koncentrationen. Først er stigningen lineær, så aftagende og til har man nået Vmax, den maksimale hastighed / maksimale mængde produkt, der kan produceres ved den givne enzym koncentration. Dette kaldes saturation. Matematisk er dette en logaritmisk funktion, hvor Vmax er asymptoten, så længe substrat koncentrationen er den eneste variable faktor vil V_max aldrig blive oversteget.

V_max og K_m er to af de vigtigste begreber indenfor enzymkinetik. V_max er den hurtigste reaktionsrate der kan opnås ved en given enzym koncentration, K_m defineres som mængden af substrat der skal til for at opnå v = 1/2 V_max, ved en låst enzym koncentration. De kan bestemmes eksperimentelt ved at lave en substratsmætningskurve;

V_max og K_m bruges til mere komplekse kinetiske beregninger, men lavpraktisk kan de bruges til at sammenligne enzymer med lignende substrater eller undersøge hvordan forskellige forhold påvirker et enzym.

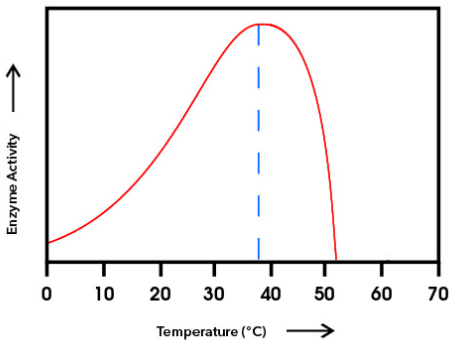


Figur 4.7: Experimentiel bestemmelse af Vmax og K_m

Temperatur og pH

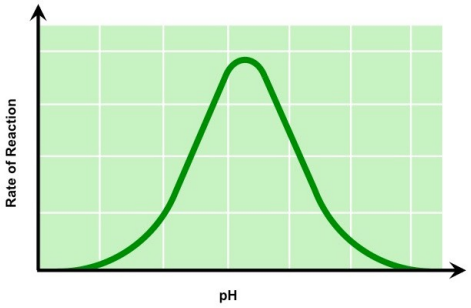
To andre faktorer som påvirker en reaktion er reaktions- temperatur/pH. Enzymer, som alle proteiner er dynamiske. De sitre lidt afhængigt af deres omgivelser. Høje temperature får de individuelle aminosyre til at bevæge sig hurtigere og da det ikke altid er i samme retning kan det føre til at den overordnede struktur går tabt. Ved lavere temperature bevæger aminosyrene sig langsommere og det tager længere tid for enzym og subtrat at mødes og gennemgå den konformatielle ændring som leder til en reaktion. pH har en lignende effekt. pH afhænger af mængden af frie protoner og aminosyrer vil enten afgive eller optage protoner afhængigt af deres struktur og pH'en. Afhængigt af hvordan proteinet er opbygget, tab eller forøgning af mængden af protoner vil ændre bindingerne mellem aminosyrerne og den overordnede struktur vil falde fra hinanden.

Hvis alle andre forhold er konstante, og kun temperatur eller pH ændres så vil en reaktionshastigheden følge en "klokke-kurve".



Figur 4.8: Temperaturoptimumskurve

Modsat substratkoncentration, så har enzymer et temperatur-/og pH-optimum. Optimaet svarer til toppen af "klokke-kurven" og afhænger af enzymet. Udover pæne grafer, så har disse optima en vigtig rolle at spille i forhold til brug og kontrol af enzymer. Vores fordøjelsessystem er et glimrende eksempel. Amylase er et enzym som nedbryder kulhydrater og som findes i vores spyt. Dets pH optimum er omkring 7. Amylase er aktivt fra vi begynder at tygge til maden når mavesækken. Som del af fordøjelsesprocessen producere mavesækken mavesyre, dybest set saltsyre pH 2. Dette er alt for lavt for amylase, som nedbrydes og ikke længere er aktivt. Pepsin som er en general endopeptidase har tilgængæld et pH optimum på 2 og bliver nu aktivt. Mavesækken er beskyttet er lag af slim som forhindrer pepsin (og saltsyren) i at angribe selve mavesækken. Når maden



Figur 4.9: pH-optimumskurve

passere videre til 12-finger tarmen følger slimen ikke med, men pepsinen gør og ville potenielt begynde at nedbryde alle de proteiner vi selv producere. Dette er forhindret, da maden samtidig med at den bliver sendt til 12-finger tarmen bliver tilført hydrogencarbonat som optager fire protoner. Dette leder til en pH omkring 7, hvilket er alt for højt for pepsin som nedbrydes.

Øvelser

Dette er en opdigtet kæde af aminosyrere, HHAGKTGAGCGFALYCINPDME, brug 4.4 til at forudsige strukturen

Er enzymstruktur nedarvet, begrund dit svar?

Er proteaser, amylaser og lipaser hydrolaser?

Udfra det du ved, beskriv funktionen af:

- histone acetylase
- RNA polymerase
- MAP kinase
- MAP kinase kinase

Nogle af enzymerne nævnt i tabel 4.1 kommer i par, f.eks. kinase fosfatase. Kan du tænke dig til en årsag til, at netop disse enzymer og ikke f.eks. lipaser altid har en partner med modsat effekt?

Tænk tilbage op graf 4.6, hvordan tror en lignende graf vil se ud, hvis substrat koncentrationen var låst og det var mængden af enzym som ændrede sig?

Se på 4.8, forklar hvorfor grafen ser ude som den går;

- Før den stiplede linje
- Ved den stiplede linje
- Efter den stiplede linje

Sammenlign Fig 4.8 og Fig 4.9, kan du forklare hvorfor graferne ser forskellige ud på de to sider af optimaet.

Tegn 3 grafer med samme V_max, men forskellige K_m. Udfra dine grafer, ville du forvente at et enzym med en høj K_m værdi binder sit substrat bedre eller værre end et enzym med en lav K_m

4.4 Ordliste

ER: Endoplasmatisk reticulum, et organel i cellen som behandler proteiner og klargør den til udskillelse fra cellen.

Svovlbro: en kovalent binding mellem to svovl atomer, bruges til at danne og fastholde en bestemt struktur i et protein

Hydrofil: vandelskende,

Hydrofob: vandhadende,

Kapitel 5

Datalogi

5.1 Introduktion

Datalogi er et ord, som I måske ikke har hørt ofte. Dog har datalogien og dets væsen en større indflydelse på vores samfund end nogensinde før. Datalogi er egentlig blot den danske version af ordet computer science. Datalogi beskæftiger sig med den teoretiske forståelse af computeren fra hardware til software, og alt derimellem. Det er den digitale logik, 1'ere og 0'ere og hvordan de kan opereres på med logiske operationer, men også hvordan man sorterer tal i en liste. I dette afsnit kommer vi til at lære indledende programmering i Python, som er den praktiske del af datalogi.

Hvem er vi og hvad betyder datalogi for os?

Bamse

Det første jeg programmerede var Lego Mindstorms. Dernæst kom jeg igang med en tutorial til Gamemaker som Troldspejlet havde lavet, så jeg fik lavet nogle små computerspil. Derfra fik jeg langsomt prøvet mere og mere programmering, så nu har jeg en kandidat i datalogi. I øjeblikket arbejder jeg dog kun lidt med programmering. Jeg har et deltidsjob i et Escape Room, hvor jeg programmerer nogle Raspberry Pi's og computere til at vise videoer, tage imod indtastninger af forskellige koder og åbne for elektroniske låse. Det er meget sjovt, men datalogimæssigt er det ikke det der interesserer mig mest.

De dele af datalogi jeg synes er mest spændende er i den teoretiske og matematiske side. Grafteori, algoritmer og kompleksitetsanalyse er det jeg har skrevet speciale om og som jeg synes er sjovest at arbejde med. Kort sagt kan jeg godt lide at udtænke løsningen på et problem, men har ikke et stort behov for faktisk at kode det. I øvrigt elsker jeg at lære fra mig. Det er helt fantastisk at se når folk lærer noget nyt, og dejligt at kunne hjælpe dem på vej.

Hvis man har det sjovt mens man programmerer, f.eks. ved at lave et fjollet spil, så plejer det at resultere i at man får programmeret mere, fordi man er motiveret til bruge tiden på at lære det som er nødvendigt. Så mit råd er at prøve at finde et sjovt projekt, som kan hjælpe en med at holde sig igang.

Louie

Siden helt lille har jeg været fascineret af matematik og dens underliggende betydning for vores verden. Datalogi var noget jeg mødte for første gang i gymnasiet, hvor valgte at tage programmeringslinjen. Programmering var et fag jeg meget hurtigt blev glad for, fordi jeg fandt ud af at det ikke bare et værktøj, som kunne benyttes til at skabe hjemmesider, men også gav viden, som kunne bruges til løse udfordrende problemer igennem diverse algoritmer. I dag studerer jeg fysik, og bruger ofte datalogi til studiet, blandt andet til simulation af planeters baner samt avancerede former for databehandling.

Mit gode råd for en person som gerne vil lære at programmere er at skrive men masse programmer, enten som små projekter eller løsninger til programmeringsproblemer.

Ria

Mit første rigtige møde med datalogi og programmering var faktisk på mit studie, der ikke direkte handler om nogen af delene. Jeg studerer på kandidaten i matematik på KU, hvor statistik fylder en del i min linje. Derfra lærte jeg at kode i R og har sidenhen også kodet en smule i Python.

Jeg vidste aldrig, at der ville være så meget kodning i matematiske fag, men jeg er rigtig glad for overraskelsen, for det har lært mig en masse om, hvor sejt og brugbart datalogi faktisk er. Særligt kan jeg godt lide programmeringsopgaver der handler om simulation og hvad der ellers indeholder sandsynlighedsregning.

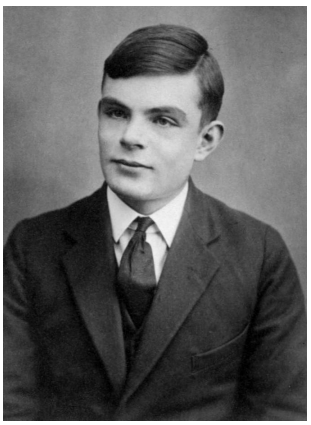
Der findes utroligt mange grene af datalogien og endnu flere anvendelser. Hvis man godt kan lide naturvidenskab kan man stortset altid finde et projekt, der både involverer ens interesser og også lærer en at kode.

Hvad er datalogi og programmering

Dette afsnit er en kort introduktion til hvad programmering er, og kan anbefales for alle at læse, før de kaster sig ud i det. For at forstå hvad programmering er, skal vi først forstå hvordan en computer fungerer.

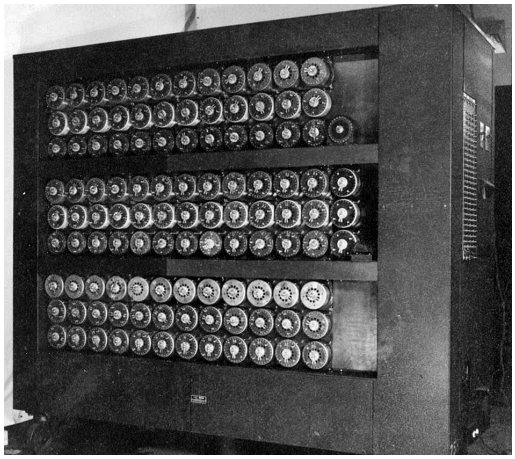
Historien af computeren

Digitale computere som vi bruger i dag har sin oprindelse tilbage fra anden verdenskrig, hvor de allierede var desperate for at kunne knække det såkaldte "Enigma-koden", som tyskerene brugt til at kommunikere med hinanden. Mange har måske hørt om Alan Turing, som hyldes for at være grundlæggeren for moderne datalogiske principper.



Figur 5.1: Alan Turing da han var 16

Turing var netop den der forsøgte at designe således en maskine under anden verdenskrig, og det lykkedes for ham at opfinde en maskine der kunne knække tyskernes Enigma, hvilket havde stor betydning for at de allierede kunne vinde krigen.



Figur 5.2: Maskinen "Bombe", som Turing designede for at knække Enigma

Bombe var ikke en computer i nutidens forstand da den havde meget begrænset mulighed for at beregne alt andet end at knække Enigma-koder, mere specifikt kunne den ikke udvides til en "Turing-maskine". En Turing-maskine er en abstrakt maskine der teoretisk ville kunne udføre enhver slags beregning som man kan finde på. Turing-maskiner eksisterer ikke i virkeligheden, og er blot en koncept, men de har inspireret udviklingen af moderne computere. Nutidens computere kan udvides til en Turing-machine i den forstand at hvis de havde adgang til uendelige meget hukommelse/lagerplads, så ville de være ækvivalent med en Turing-maskine. Det samme kan man ikke sige om Bombe, da den i meget begrænset omfang var kun målrettet til at kunne knække Enigma-koder. I dag med den processorkraft som vi har i 2022, kan vi faktisk simulere Bombe på vores egne bærbare computere!

Hvad er en computer?

Computere virker ved, at vi mennesker fortæller den hvad den skal gøre, altså hvilke slags beregninger den skal køre for at nå frem til et resultat. Det er at *manipulere/udføre beregninger på* ***information***, som reelt set udgør formålet ved en computer. Det er også derfor at ordet IT benyttes, da det er en forkortelse for informations-teknologi. Information kan i nemmeste forstand, tænkes som et stykke viden som har en betydning for hvordan man *handler* i verden. Fx kunne jeg sige til dig "Det regner virkeligt meget udenfor". Det kan være nyttigt for dig at vide det, hvis du skal planlægge hvorvidt du bør tage regnjakken på. Din handling er så det at tage regnjakken på, som afhænger af informationen om hvorvidt det regner. Hvis en computer skulle kodes for at kunne fortælle dig hvad du skal gøre når det regner, så kunne det fx være:

```
regn = True
if regn:
    print("Tag regnjakken på")
else:
    print("Det regner ikke, du behøver ikke en regnjakke")
```

Dette er en meget simpel *algoritme*, og de fleste kan nok regne ud, at de bør tage regnjakken på hvis det regner. Men når informationen man er givet bliver mere og mere kompleks bliver man nødt til at regne mere for at tage beslutninger, og det er her træder computere ind i billedet. I den virkelige verden har man brug for computere og algoritmer til at fortælle hvad der præcist skal gøres under hvilke omstændigheder. Det kan fx. være at man i en fabrik har en computer, der styrer en robot, som hælder noget kemikalie i et glas for at frembringe en kemisk reaktion. Her kan det være meget vigtigt at man holder øje med, hvor meget kemikalie der er i glasset, hvad man har hældt i glasset, samt hvor lang tid er der gået. Disse størrelser udgør den konkrete information som en computer

kan regne på for at finde ud af helt præcist hvor meget den skal hælde i for at glasset ikke eksploderer.

Hvad er programmering?

Programmering er så måden man udtrykker til computeren hvordan den skal behandle den information som den modtager. I programmering skriver man i et *programmeringssprog*, som er et sprog som computeren kan forstå for at vide helt præcist hvad den skal gøre. For at kunne programmere computeren til at gøre noget, skal man kunne snakke dens sprog. Vi vil her komme til at beskæftige os med programmeringssproget Python, som er et meget populært og nemt sprog at lære, og er nok er det allermest brugte programmeringssprog i hele verden!

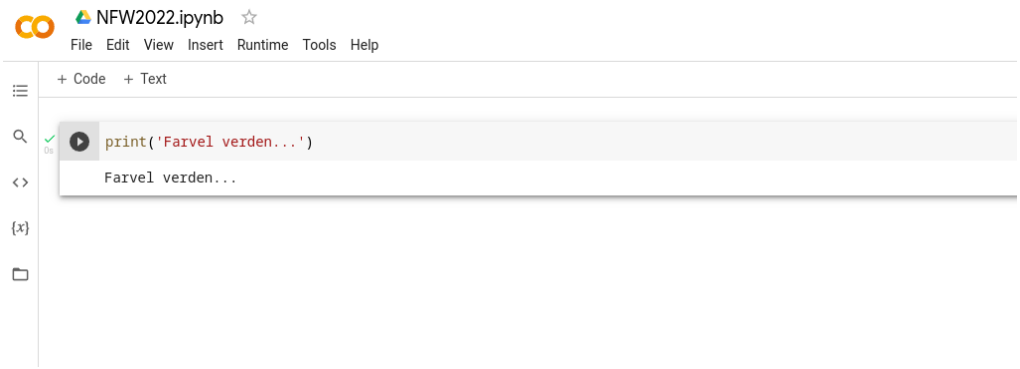
Når man programmerer, oversætter computeren fra det programmeringssprog man benytter videre til maskinkode, som er de specifikke *instruktioner* computeren tænker at den skal udføre. Det er ligesom, at man fx. kan snakke med en tysker hvis man snakker engelsk, men man tænker selv i dansk, og tyskeren tænker i tysk, dog har I et fælles sprog, engelsk, som I begge kan forstå. Med computeren bliver det klart, at det fælles sprog som I begge kan tale er Python. Men mere korrekt skal man forstå, at dette er mere eller mindre en en-vejs kommunikation, computeren gør det du fortæller den at den skal gøre, og den rapporterer så tilbage resultatet af arbejdet. I regnjakke-eksempelet ovenfor bad vi computeren om at skrive: "Tag regnjakken på" hvis regn variablen var sand, ellers ville den skrive "Det regner ikke, du behøver ikke en regnjakke".

Tabel 5.1: Relevante termer og deres betydning

Ord	Definition
Information	Noget viden der gør at du handler på en bestemt måde
Computer	En maskine der manipulerer på den information som vi mennesker giver den
Turing-maskine	En hypotetisk maskine der er i stand til at beregne alt hvad vi kan finde på, alle moderne computere kan udvides til Turing-maskiner
Programmering	Disciplinen i at kommunikere med computeren - at fortælle den hvad den skal gøre
Programmeringssprog/kode	Det man snakker for at kommunikere med computeren
Maskinkode	Instruktioner/ord som computeren forstår
Python	Et programmeringssprog udviklet i 1991 af Guido van Rossum
Python-interpreter	Et program som oversætter Python kode til maskinkode
Google-Colab	Online værktøj udviklet af Google for at køre kode på Google's computere

Konsollen og print funktionen

Du er nu klar til at skrive dit første stykke Python-kode. Vi benytter Google-Colab's Jupyter Notebooks for at gøre dette. Det er en online Python-editor hvor man kan skrive Python kode, og derefter eksekvere koden.



Figur 5.3: Google-Colab, hvor vi printer en streng

I billedet oven skriver vi koden `print('Farvel verden...')`, hvorefter resultatet, som er teksten, "Farvel verden...", ville blive "printet" til konsollen linjen under. Konsollen er der hvor Python udskriver resultatet for dens beregninger. For at printe noget til konsollen skal man bruge `print` funktionen, som er en indbygget i Python. Teksten, som man printer, kaldes for en `string`, som bare er Pythons ord for et stykke tekst. Teksten i en `string` skal man altid skrive i mellem apostrofer, eller gåseøjne, `''`, `""`. Man kan bruge `print` på følgende måder:

```
# Prøv at skrive disse i Google-Colab
print('Normal print med \', man skal skrive \' for at Python
    ↪  ikke tror at det er slutningen på ens streng ')
print("Normal print med \", man kan også bruge gåseøjne frem
    ↪  for apostrof")

# Hvis man skriver , kan man print noget ekstra, fx et tal
    ↪ lige efter ens string
print('Min alder er', 19)

# Man kan også bruge +, her skal man dog først konvertere et
    ↪  tal til en string
print('Jeg er '+str(19)+' år gammel')
```

Oven har vi nogle gange skrevet #, hvilket Python tolker som en *kommentar*. Kommentarer har ingen påvirkning på ens kode, og er der kun for at hjælpe programmører med at huske hvad ens kode gør. Man har også mulighed for at få Python til at tage imod bruger-input, dvs. at man kan få Python til at køre noget kode baseret på data man får fra brugeren. Forsøg fx at køre følgende

```
navn = input('Hvad er dit navn? ')
print('Hej '+navn+', jeg glad for at møde dig!')

# Man kan også skrive, men så laver Python selv mellemrum
print('Hej', navn, ', jeg glad for at møde dig!')
```

Man får så lov til at skrive ens navn i konsollen, derefter vil Python printe ens navn og at den er glad for at møde dig.

Opgaver til konsollen og print funktionen

●● **Opgave 5.1.1:**
Hvorfor virker følgende program ikke, kan du fikse det? (Se evt. næste kapitel)

```
tal = input('Giv mig et tal ')
print('Dit tal +5 er', tal+5)
```

5.2 Typer og variabler

Datatyper

Hvad er en datatype? En datatype er en klassificering af hvilken type data man har at gøre med. Det kan godt lyde lidt som en cyklisk definition, så for at forstå det bedre kigges der på nogle eksempler. Der er 5 basale datatyper i Python, de er:

1. `int`, som repræsenterer heltal, fx, `-1, 2, 44, 42, 69`
2. `float`, som er decimaltal, fx, `-1.12, 3.1415, 1.4423, 2/3`
3. `boolean`, som kan antage to sandhedsværdier, `True` for sandt, og `False` for falsk
4. `str`, som er et stykke tekst, fx, `"Se mor! Jeg koder i Python!", "Jeg er 14 ↪ år gammel"`
5. `NoneType`, ingenting, hvis variablen er `None`.

Disse datatyper er dem som vi vil primært beskæftige os med. Typisk har man brug for at konvertere fra en datatype til en anden. Fx kan Python ikke tolke `print('9'+10)` da man forsøger at plusse en `str` med en `int`, skal den printe `'910'` eller `19`? Det ved python simpelthen ikke. Derfor opstiller man i programmeringssprog klare konventioner for hvordan operationer skal udføres. For at kunne gøre ovenstående, skal man derfor enten konvertere tallet 9 fra `str` til `int` eller omvendt for 10. Det kan man gøre ved at skrive, `print(int('9')+10)` eller `print('9'+str(10))`. Man kan konvertere alt fra en bestemt datatype til en anden, hvis man bare skriver `t(_)` hvor `t` kan være alt mellem `int, str, bool, float` og `_` er det tal/streng man vil gerne konvertere.

Variabler

Variabler bruger man til at gemme en værdi, som kan antage de datatyper beskrevet ovenfor:

```
fem = 5
to = 2

syv = fem + to
print(syv)
>>> 7
```

For at danne en variable, skriver man først navnet på sin variabel, som i eksemplet foroven var `fem` eller `syv`, variabel navnene kunne ligeså godt være kaldt hhv. `tallet_fem` eller `antal_dage_i_ugen`. Derefter skriver man et lighedstegn `=` følgende med værdien man gerne vil definere ens variabel med. Her kan man se flere eksempler

```
mit_yndlingstal = 420
tallet_pi = 3.14159265359
det_er_sandt = True
vores_navne = "Ria, Bamse og Louie"

# Det her er en kommentar, som ikke ville blive kørt i koden
# Man kan lave matematik med variabler
nyt_tal = mit_yndlingstal + tallet_pi
print(nyt_tal)
>>> 423.14159265359
```

Du har måske bemærket, at vi ikke skriver med mellemrum, tal, " eller ", når vi definerer variabler. Det er noget, som Python klager over, hvis man gør (bare prøv at gøre det hvis ikke I stoler på os!). Der findes visse begrænsninger for variablernes navne, alle variabler skal opfylde følgende

1. Den kan enten starte med underscore, `_` eller et bogstav. Man kan ikke starte en variabel med et tal!
2. Variabler må kun indeholde bogstaver, tal eller underscore, `_`.
3. Variabler er "case-sensitive", dvs. `TALLET_fem` og `tallet_fem` er to forskellige variabler.

Man kan fx se et eksempel her

```
# Lovlige navne på variabler:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"

# Ulovlige variable navne, ALDRIG skriv disse:
2myvar = "John"
my-var = "John"
my var = "John"
```

Matematiske operationer

Variabler er en meget nyttige at bruge når man skal lave regnestykker, som kræver at man kan referer til noget data som man har gemt før, fx i matematik kan man lave andengradsligninger, hvor der indgår variablerne a, b, c og x i ligningen $ax^2 + bx + c = 0$. I Python kan man udregne matematiske ligninger og funktioner ved at definere variabler, og så derefter lave matematiske operationer på dem:

```
a = 5
b = 3
c = -9

# Vi udregner andengradsligningen her (det hedder rent
    ↪ faktisk et andengradspolynomium...) for x lig 4

x = 4
resultat = a*x**2 + b*x + c

print(resultat)
>>> 83
```

På linje 8 skrev vi koden `a*x**2 + b*x + c`, som repræsenterer den matematiske formel $ax^2 + bx + c = 5 \cdot 4^2 + 3 \cdot 4 - 9 = 83$. Der er visse operationer som man kan lave, her kan I se en liste af dem

```
# Tal
3  # => 3

# Matematik som man forventer det
1 + 1   # => 2
8 - 1   # => 7
10 * 2  # => 20
35 / 5  # => 7.0

# Når man dividerer 2 tal får man altid en float
```

```
10 / 3  # => 3.3333333333333335

# Modulo operation, division med rest
7 % 3   # => 1, fordi 2 gange 3 = 6, som er 1 væk fra 7

# Potensløftning (x**y, x opløftede i det y'te potens)
2**3  # => 2*2*2=8

# Parentes for hvilken operationer man regner først
1 + 3 * 2    # => 7
(1 + 3) * 2  # => 8
```

Man kan lave ret avancerede matematiske ligninger, fx idealgasligningen (noget I kommer til at lære i gymnasiet), $PV = nRT$. Her kan man isolere $n = \frac{PV}{RT}$ og udregne n givet at man har defineret variablerne P, V, R, T

```
P = 197 # Atmosfærisk tryk
V = 22.4 # Liter (vi definerer en float her)
T = 28 + 273 # Kelvin
R = 8.21 * 10**(-2) # konstanten R, videnskabelige notation

n = (P*V)/(R*T)
print('n har vi udregnet til at være', n, 'mol')

# Hvis man ikke havde variabler, så skulle man skrive noget
   ↪ rod i denne stil
n = (197*22.4)/((8.21 * 10**(-2))*(28 + 273))
# Som er uoverskueligt da man ikke ved hvad tallene betyder
```

5.3 Sammenligninger og udtryk

Noget af det mest fundamentale i programmering er at man skal kunne sammenligne to værdier med hinanden for at afgøre hvorvidt noget kode skal køre eller ej. I matematik, kan man fx opstille *udtryk* såsom:

$$\begin{aligned} &\text{sandt} : 5 < 10 \\ &\text{falskt} : 1 = 2 \\ &\text{sandt} : 6 \cdot 5^2 + 4 \cdot 5 + 9 > 0 \end{aligned}$$

Hver af disse linjer har en *sandhedsværdi* som er hvorvidt udsagnet er sandt eller falsk. Det er sandt at 5 er mindre end 10, derimod er det falsk at 1 er lig 2. Bemærk, at ethvert af disse udtryk, består netop af 3 dele, 2 værdier og et tegn der sammenligner disse to værdier. I det første har vi værdierne "5" og "10", som bliver placeret på hver sin side omkring mindre-end tegnet "<". I det allersidste udtryk foroven, har vi andengradspolynomiet $6 \cdot 5^2 + 4 \cdot 5 + 9$ som antager værdien 179 hvis man udregner den, og som er selvfølgelig større end 0.

I Python findes der logiske *udtryk*, som har en af to værdier. `True` for sand, eller `False` for falsk. Fx kunne nogle af disse udtryk se ud på følgende måder

```
print(1 == 2)
>>> False

print(5*10-10 < 40)
>>> False

print('Ingen stavefejl her' != 'Ingen stavefejl hr')
```

```
>>> True

tid = 800
print(tid < 900)
>>> True
```

Vi kalder de tegn man bruger til at sammenligne to værdier med hinanden for sammenligningsoperatorer, eller comparison operators på engelsk. Her er en liste af alle de vigtige sammenligningsoperatorer.

Tabel 5.2: Tabel af sammenligningsoperatorer

Operator	Tilsvarende i matematik	Eksempel
`==`	$=$	`5 == 4` er `False`
`!=`	$\neq$	`5 != 4` er `True`
`>=`	$\geq$	`5 >= 4` er `True`
`<=`	$\leq$	`5 <= 4` er `False`
`>`	$>$	`5 > 4` er `True`
`<`	$<$	`5 < 4` er `False`

Som man så i eksemplet foroven, så kan man fx. godt sammenligne `strings` med hinanden, hvilket kunne være meget nyttigt hvis man fx skulle teste om hvis en bruger har tastet noget korrekt. Desuden sammenligner man for det meste af tiden, en variabel med en anden værdi, som også kunne være en variabel. Dette gør, at vi kan opstille betingelser for hvornår noget kode skal køre, alt afhængigt af hvilken værdi variablen har (se næste kapitel).

Vi skal nu til at snakke om boolske operationer, som er operationer der sammenligner og negerer *udtryk*, hvor vi førhen havde sammenligningsoperationer, som sammenlignede *værdier*, og gav et *udtryk*, bruger vi boolske operationer på selve udtrykkene. De tre mest basale boolske operationer er, `not`, `and` og `or`. Operationen `not` negerer den boolske værdi, hvilket ville sige at et `False` bliver til et `True`, og omvendt bliver `True` til `False`. Operationen `or`, sammenligner to udtryk, og hvis blot en af dem er `True`, så returnere den `True`. Kun hvis begge er falske returneres `False`. Sidst, men ikke mindst, så er der `and`. `and` sammenligner ligesom `or`, to værdier. `and` returnerer kun `True`, hvis begge værdierne af sande, ellers returneres `False`. For at give et eksempel så betragte følgende. Lad A og B være vilkårlige udtryk, de kunne fx være `5*5 == 25` eller `tid < 1000`, hvor `tid` er en variabel. Disse udtryk har så en sandhedsværdi, og vi kan opstille en tabel hvor vi bruger de boolske operationer, `and` og `or` på disse udtryk:

Tabel 5.3: Tabel for boolske operationer

A	B	`or`	`and`
`True`	`True`	`True`	`True`
`False`	`True`	`True`	`False`
`True`	`False`	`True`	`False`
`False`	`False`	`False`	`False`

Vi kan nu fx skrive mere komplicerede udtryk såsom disse:

```
tid = 900 # kl 9.
penge = 1000 # kr.

print(tid < 1000 and penge == 800)
>>> False
```

```
tid_er_lig_penge = tid == penge # man kan få variabler til
    ↪ at gemme resultatet af et udtryk
print(tid + 100 == penge or tid_er_lig_penge)
>>> True

print(not (tid + 100 == penge))
>>> False

print(tid + 100 != penge) # det samme som oven
>>> False

print(not (tid == penge and penge-100 < 100))
>>> True
```

Opgaver til datatyper og udtryk

- **Opgave 5.3.1:**
  Udregn følgende i Python og derefter print resultatet.

  1. $939577132 \cdot (2124 - 65)$
  2. $129^3 \cdot 2124 - \frac{4}{3}$
  3. 420^{69}
  4. 5^{5^5}

- **Opgave 5.3.2:**
  Tænk over resultatet (og om det kan evalueres og evaluer bagefter i Python.

  1. `(not False and True)or False`
  2. `not not not not False`

- **Opgave 5.3.3:**
  For vilkårlige udtryk `A` og B, er det korrekt at

  1. `not (A and B) == (not A or not B)`
  2. `not (A or B) == (not A and not B)`

  Altid evaluerer til `True`?
  Svar: ja, denne lov kaldes for De-Morgans lov, overvej hvorfor.

5.4 Control sequences, if, else, for, while

Vi skal nu kigge på control sequences, som er måden man styrer hvilken kode, der skal køre under hvilke betingelser. Det er vigtigt at man mestrer brugen af disse når man skriver sine programmer.

if-else udtryk

Vi starter med at forstå, hvad et if-else udtryk er. Koden for dette udtryk er struktureret på følgende måde:

```
mit_yndlingstal = 42
if mit_yndlingstal > 50:
    print("Mit yndlingstal er større end 50")
elif 40 < mit_yndlingstal < 50:
    print("Mit yndlingstal er mellem 40 og 50")
else:
    print("Mit yndlingstal må være mindre eller lig 40")
```

Ordet "`if`" er "hvis" på dansk. Derudover er "`elif`" blot en forkortelse af "`else-if`", hvilket betyder "ellers-hvis" på dansk. Til sidst har vi "`else`", som betyder "ellers". Det er ikke særligt svært at gennemskue hvad der sker i koden. Der bliver først defineret en variabel, nemlig `mit_yndlingstal`, der sættes til `42`. I det første `if` skriver vi en *betingelse*, som reelt kan være hvilket som helst udtryk. Betingelsen i dette tilfælde er `mit_yndlingstal > 50` og hvis den er opfyldt, dvs. hvis variablen som hedder `mit_yndlingstal` er rent faktisk større end 50, så vil strengen "Mit yndlingstal er større end 50" blive printet i konsollen. Det første if-udtryk kan læses meget direkte som: "Hvis mit yndlingstal er større end halvtreds, så kør koden nedenunder, der printer en sætning i konsollen".

Bemærk her at der er 4 mellemrum (en tab) under linjen efter vores `if, elif, else`. Det er nødvendigt for Python at vide hvilket stykke kode den skal køre, hvis betingelsen er opfyldt. Fx er disse ikke lovlige if-else udtryk:

```
mit_yndlingstal = 42

# Python ville sige at der er en fejl, prøv at køre hvis du
    ↪ ikke tror på os!
if mit_yndlingstal > 50:
    print("Mit yndlingstal er større end 50")
elif 40 < mit_yndlingstal < 50:
    print("Mit yndlingstal er mellem 40 og 50")
else:
    print("Mit yndlingstal må være mindre eller lig 40")
```

I dette tilfælde er `mit_yndlingstal` dog ikke 50, det er 42, så her kommer `elif` ind i billedet. Et "else-if" bliver kørt hvis ens første "if" ikke var opfyldt. Så vil Python kigge *sekventielt* på den næste betingelse, der skal evalueres for at køre noget, hvilket i dette tilfælde står i `elif` udtrykket. Man kan have vilkårlige mange `elif` udtryk og det gør så man kan kontrollere meget præcist, hvad der skal køres under hvilke omstændigheder. Til allersidst har vi `else`, som køres hvis alt andet fejler, dvs. hvis alle ens `if, elif` udtryk er falske. Her behøver man ikke at skrive en betingelse.

Opgaver til if-else udtryk

- **Opgave 5.4.1:**
  Hvilket af følgende er korrekt brug af if-else?

```
# 1
else:
    print("Hej med dig")

# 2 Overvej hvad der ville blive printet i den følgende
min_yndlingsfarve = "lilla"
if min_yndlingsfarve != "lilla":
```

```
    print("Min yndlingsfarve var ikke lilla!")

# 3
louies_tal = 420
bamses_tal = 40
if louies_tal < 500:
    if bamses_tal < 500:
        print("Både Louie og Bamse har et tal under 500")
else:
    print("Louies tal er ihvertfald større end 500, vi ved
        ↪ ikke om Bamse")

# 4 Overvej om hvis if-udtrykket (kig bort fra else) i 3 er
    ↪ det samme som
if louies_tal + bamses_tal < 1000:
    print("Både Louie og Bamse har et tal under 500")

# og/eller
if louies_tal < 500 and bamses_tal < 500:
    print("Både Louie og Bamse har et tal under 500")
```

5.5 For og while loops

Vi skal nu se på for og while loops (løkker på dansk), som er måden man kan få Python til at køre det samme kode flere gange. Et for-loop antager følgende form:

```
# 1 Printer alle tal fra 0 til 9, 10 er ikke inklusive
for tal in range(0, 10):
    print("Hej tal:", tal)

# 2 Man bruger for loops til at iterere over lister, som i l
    ↪ ærer i næste kapitel
# Koden neden er samme som 1
nul_til_ni = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for tal in nul_til_ni:
    print("Hej tal:", tal)

# 3 Man kan have lister af alt muligt, her bruger vi strings
louie_og_bamse = ["Louie", "Bamse"]
for unfer in louie_og_bamse:
    print("Hej", unfer)
```

Hvis man får Python til at køre koden foroven, så vil den printe strengen: "Hej tal: _", hvor _ løber fra 0 til 9. Når man skriver et for-loop, *itererer* man over *elementer*, som fx kunne være tal eller eller elementer i en liste. Hvis man vil gerne tælle fra et bestemt heltal a til et tal b, kan man bruge Python's indbygget funktion "`range`", til at skrive `range(a, b)` som vil *genererer* alle tal fra a til b hvor b ikke er inkluderet. Det er helt afgørende når man skriver for-loops, at man bruger operatoren "`in`", som udtrykker til Python, at man gerne vil iterere over elementer der *ligger i* en liste eller `range`. I det første eksempel oven, kan vi læse for-loopet som "for alle tal mellem 0 og 10, print "Hej tal: _" til konsollen". Vi bevæger os videre til while-loops, som er en anden måde man kan skrive løkker på:

```
tal = 0
while tal < 10:
```

```
    print("Hej tal:", tal)
    tal += 1
```

Med while-loops er der tale om en betingelse, som skal være opfyldt for at while-loop'et må køre. I eksemplet foroven, har vi et while-loop som betinger sig på, at `tal` skal være mindre end 10, i hvilket tilfælde den så printer tallet, altså fuldstændigt tilsvarende til det tidligere for-loop. I selve while-loop'et har vi koden `tal += 1` som svarer til `tal = tal + 1`, som I måske kan regne ud hvad det betyder.

Et for-loop kører på en anden slags betingelse end et while-loop. Den væsentligste forskel er, at for-loop'et kører så længe, at der er elementer til rådighed. Et while-loop, derimod, kører på at while-udtrykket er opfyldt. I dette tilfælde er det altså ikke længere end liste eller "`range`", der bestemmer hvor længe loopet kører, men snarere et boolesk udtryk.

Opgaver til for og while loops

●●● **Opgave 5.5.1:**
Overvej hvad der ville ske med koden nedenunder, prøv evt. at skrive dem i Python!

```
# 1
for tal in range(7, 5):
   print(tal)

# 2
tal = 0
for tal < 10:
    print(tal)

# 3 Tænk grundigt over følgende!
tal = 8
while tal < 10:
    print(tal)

# 4
unf = ["Louie", "Bamse", "Ria"]
for unfer in unf:
    print("Vi har fat i", unfer)
    if unfer == "Louie" or unfer == "Bamse":
        print("Og han er datalog!")
        if unfer == "Louie":
            print("Okay, han er faktisk fysiker, men det er
               ↪ hemmeligt, shh!")
        elif unfer == "Bamse":
            print("Og han kan lave mange seje tricks!")
    else:
        print("Hmm, gad vide om vi har fat i den rigtige
           ↪ person? Mon ikke det er en matematiker?")
```

5.6 Datastrukturer: Lister og dictionaries

I denne sektion skal vi kigge på nogle simple datastrukturer: lister og dictionaries.

Lister

Så hvad er en liste? En liste er blot en samling af elementer, den kan være vilkårligt lang og man kan tilføje og fjerne ting fra listen. Man bruger lister til at organisere ens data,

fx hvis man har data der skal læses i en bestemt rækkefølge for at det giver mening. Her kan vi se hvordan vi definerer en liste med nogle elementer.

```
primtallene = [2, 3, 5, 7, 11, 13] # En liste med nogle
    ↪ primtal
ugens_dage = ["Monday", "Tuesday", "Wednesday", "Thursday",
    ↪ "Friday", "Saturday", "Sunday"] # En liste med ugens
    ↪ dage
forskellige = ["Monday", 1, "Tuesday", 2] # Man kan have
    ↪ forskellige datatyper i en liste
# Man behøver ikke at have noget i listen når man definerer
    ↪ den
# Det kan være at man senere vil gerne tilføje til den.
tom_liste = []
```

Så hvad kan man gøre med lister? Man kan indicere, indsætte, fjerne og meget andet. Nedenfor kan nogle eksempler ses. Her er det vigtigt, at nævne at lister er indicerede fra 0 af, hvilket betyder at man tilgår det første element ved at referere til den 0 position i listen.

```
# Indicere (tilgå et element)
print(primtallene[0])
>>> 2

# Udskifte (Udskifte et element)
ugens_dage[1] = "Tirsdag" # Husk vi starter ved element 0, s
    ↪ å er Tuesdag nr. 1 i listen
print(ugens_dage[1])
>>> Tirsdag

# Appende (Indsætte et element i slutningen af listen)
tom_liste.append("Hej")
tom_liste.append("med")
tom_liste.append("dig")
print(tom_liste)
>>> ['Hej', 'med', 'dig']

# Sæt to lister sammen
hej1 = ["Hej", "med", "dig"]
hej2 = [",", "hvordan", "går", "det?"]
hej3 = hej1+hej2
print(hej1+hej2)
>>> ['Hej', 'med', 'dig', ',', 'hvordan', 'går', 'det?']

# Slette
søndag = ugens_dage.pop() # Fjerner allersidste element og
    ↪ returnere den, som er "Sunday"
print(søndag)
>>> Sunday
mandag = ugens_dage.pop(0) # Fjerner det første element
print(ugens_dage)
>>> ['Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday
    ↪ ']

# Dette må du ikke gøre!
print(primtallene[10]) # Dette element eksisterer ikke (out
    ↪ of bounds)
```

```
primtallene[10] = 15 # Du kan igen ikke tilgå a[10]
```

Vi kan gøre mange ting med lister, hvis vi fx vil gerne udregne kvadratet af de første 10 naturlige tal, kan vi bruge en for loop sammen med en liste

```
kvadraterne = []
for tal in range(10):
    kvadraterne.append(tal ** 2)
print(kvadraterne)
>>> [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Opgaver til lister

- **Opgave 5.6.1:**
  Lav en liste med kvadraterne af tallene $1, 2, 3, ..., 100$ og print resultatet.

●●● **Opgave 5.6.2:**
Lav en liste hvor du bruger **en** for-loop til at finde differensen for hvert element i `kvadraterne` med hvert element i `kubik_tallene` elementvis (træk kvaradratet fra kubik-tallet). Print derefter resultatet.

```
    # denne måde at definere lister på hedder list-
      ↪ comprehension, de nysgerrige kan slå det op på
      ↪ Google :)
    kvadraterne = [i ** 2 for i in range(100)]
    kubik_tallene = [i ** 3 for i in range(100)]
    # Skriv din kode her
```

●●●● **Opgave 5.6.3:**
Konstrurere listen `[[[[[[[[[[’Hej’]]]]]]]]]]` ved at bruge en for loop. Start fra listen `[’Hej’]` og forsøg at lave flere ’lag’ med en for loop. Overveje derefter hvordan du ville gøre det for vilkårlige mange lag.

Dictionaries

Dictionaries kan man tænke lidt som en telefonbog, i gamle dage før internettet eksisterede brugte man telefonbøger til at slå op telefonnummrene på fx tømre eller eletrikere. Hvis man kender navnet på det selsskab man ledte efter, kunne man se at telefonbogen var organiseret i alfabetisk rækkefølge, og derefter slå deres nummer op. En dictionary på dansk kaldes for en ordbog, som også giver lidt et intuitivt billede af hvad de kan.

```
telefon_bogen = {
    "Bamse": 56723822, # det her er ikke et rigtigt nummer
    "Louie": 66733234, # det her er ikke et rigtigt nummer
    "Politiet": 112 # det her tilgengæld er rigtigt
}
```

Man definerer en dictionary ved at skrive `{}`. Dictionaries har følgende struktur

```
min_dictionary = {
    key: value,
    key2: value2,
    key3: value3, # der skal tilføjes komma efter hver entry
    ... # man kan have vilkårlige mange entries (indgange)
}
```

Hvor `key` er typisk `int`, `str` og `value` kan være hvad som helst (`key` skal være en hashable type, slå det op på Google hvad det betyder). Vi kan så slå ting op i vores dictionary, fx hvis vi vil gerne have Louies nummer, kan vi skrive

```
louies_nr = telefon_bogen["Louie"]
print(louies_nr)
>>> 66733234
```

Vi kan tilføje en ny værdi til denne dictionary ved at skrive

```
telefon_bogen["Dig"] = 6213812
print(telefon_bogen)
>>> {'Bamse': 56723822, 'Louie': 66733234, 'Politiet': 112,
    ↪ 'Dig': 6213812}
```

Vi kan have mere avancerede dictionaries, fx kan vi have dictionaries i dictionaries

```
folk_fra_unf = {
    "Bamse": {"tlf": 56723822, "hobbyer": ["programmering",
       ↪ "minecraft"], "alder": 19},
    "Louie": {"tlf": 66733234, "hobbyer": ["kompetitiv
       ↪ programmering", "unf", "fysik"], "alder": 20},
    "Dig": {"tlf": 6213812, "hobbyer": ["datalogi", "unf"],
       ↪ "alder": 420},
}
```

Vi kan så tilgå nogle af disse værdier ved at skrive:

```
print("Bamses telefon nr. er", folk_fra_unf["Bamse"]["tlf"])
>>> Bamses telefon nr. er 56723822
print("Hans hobbyer er", folk_fra_unf["Bamse"]["hobbyer"])
>>> Hans hobbyer er ['programmering', 'minecraft']
```

Vi kan også loope i gennem en dictionary, her looper vi med to variabler `navn` og `info` på `folk_fra_unf.items()` som giver os alle de key, value par for hver entry

```
# Prøv at køre koden i Colab!
for navn, info in folk_fra_unf.items():
    print("Vi har fat i", navn)
    print("Her er deres info", info)
```

Opgaver til dictionaries

●● **Opgave 5.6.4:**
Forsøg at printe informationerne i for-loop'en på en lidt pænere måde, gerne fx printe hver hobby for sig i stedet for at printe hele listen. Hint: Lav endnu en loop i for-loop'en for at printe listen af hobbyer

●●●● **Opgave 5.6.5:**
Lav et program der simulerer dictionaries vha. lister. Man skal kunne tilgå entries, appende, og fjerne elementer. Lav det hele uden at bruge Pythons dictionaries. Hint: tænk på en liste af lister med 2 elementer, kan man bruge dem til at konstrurere en dictionary?

Funktioner

Funktioner er nok en af de allervigtigste begreber inden for programmering, det kan være svært at forstå hvad en funktion er i starten, men den er et helt kritisk værktøj i programmørens værkstøjskasse som enhver programmør bør mestre brugen af. Det er nemmest i første omgang at tænke funktionen som en maskine, der gør et andet. Man *kalder* funktionen, betyder at man tænder for maskinen. Vi har hidtil støt på `print`, som er en funktion der tager en string som *input* og printer den til konsollen. De fleste funktioner tager i mod en eller flere *input*s, og *returnerer* en *output*. Lad os definerer en simpel funktion

```
def kvadradtet_af(x):
    kvadrat = x ** 2
    return kvadrat
```

Funktionen oven kvadrere et tal, i dette tilfælde tager den i mod en input, et tal `x` og sætter den i anden, hvorefter værdien bliver tildelt til variablen `kvadrat`, som vi til allersidst retunerer. Vi kalder funktionen ved at skrive

```
ni = kvadradtet_af(3)
fire_treds = kvadradtet_af(8)
print(ni, fire_treds)
>>> 9 64
```

I koden oven finder vi kvadratet af tallene 3 og 8. Tallene erstatter `x` i definitionen af vores funktion, og derefter bliver `x ** 2` udregnet og så returneret. Vi kan også definere funktioner med flere input variabler, navnene på input variablerne er vilkårligt valgt

```
def sum_af_to_tal(x, y):
    summen = x + y
    return summen

# Man kan også skrive
def sum_af_to_tal(tal1, tal2):
    summen = tal1 + tal2
    return summen
```

Vi behøver egentlig ikke returnere noget fra funktionen, fx kan vi godt have at funktionen blot printer en string, i så fald er en `return` unøvendigt

```
def hils(unfer):
    print("Hej med dig", unfer, "godt at møde dig!")

hils("Louie")
>>> Hej med dig Louie godt at møde dig!
```

Typisk bruger man funktioner til at mindske den mængde af kode man skal skrive, hvis man fx har noget kode som gentages under forskellige omstændigheder, så giver det ikke mening at man skriver det samme kode flere gange hvis man allerede skrevet koden en gang. Programmering handler om at man ikke skal skrive mere end nødvendigt, grunden til at man fx har for loops er således at man ikke ender med at have kode såsom dette...

```
# Ser grimt ud...
print(kvadratet_af(0))
print(kvadratet_af(1))
print(kvadratet_af(2))
print(kvadratet_af(3))
print(kvadratet_af(4))
print(kvadratet_af(5))
```

```
# Lækker og kort
for i in range(6)
    print(kvadratet_af(i))
```

For at demonstrere hvorfor det er smart med funktioner, så kan det være at man bliver bedt om at anvende funktioner på nogle elementer i en liste på en bestemt en rækkefølge. Det kan være at man vil kvadrere alle tallene i en liste, derefter gange 5 og trække 2 fra. Men hvad hvis man vil også prøve først at trække 2 fra, derefter kvadrere og gange med 5? Vi kan definere disse funktioner

```
def træk_2_fra(x):
    return x-2 # Man behøver ikke at definere en variabel
       ↪ for det man returnere

def gange_med_5(x):
    return x*5

def kvadrere(x):
    return x**2

def vores_funktion(liste_af_tal, liste_af_funktioner):
    resultat_listen = []
    for tal in liste_af_tal:
        resultat = tal
        for funktion in liste_af_funktioner:
            if funktion == 'kvadrere':
                resultat = kvadrere(resultat)
            elif funktion == 'gange5':
                resultat = gange_med_5(resultat)
            elif funktion == 'træk2':
                resultat = træk_2_fra(resultat)
        resultat_listen.append(resultat)
    return resultat_listen
vores_funktion([1, 2, 3, 4, 5], ["kvadrere", "gange5", "træ
   ↪ k2"])
>>> [3, 18, 43, 78, 123]

vores_funktion([1, 2, 3, 4, 5], ["træk2", "kvadrere", "
   ↪ gange5"])
>>> [5, 0, 5, 20, 45]

vores_funktion([1, 2, 3, 4, 5], ["kvadrere", "gange5", "træ
   ↪ k2", "træk2"])
>>> [1, 16, 41, 76, 121]

vores_funktion([1, 2, 3, 4, 5], ["kvadrere", "gange5", "træ
   ↪ k2", "træk2", "træk2"])
>>> [-1, 14, 39, 74, 119]

vores_funktion([1, 2, 3, 4, 5], ["gange5", "gange5"])
>>> [25, 50, 75, 100, 125]
```

Der er flere sjove ting man kan lave med funktioner, fx kan man lave en funktion der returnerer en funktion, eller rekursive funktioner som I kommer til at se i den næste kapitel.

Opgaver til funktioner

●●● **Opgave 5.6.6:**
I Python kan man lave en liste af funktioner, fx i koden tidligere kunne man have defineret listen `liste_af_funktioner = [kvadrere, gange_med_5, træk_2_fra]`, man kan så kvadrere fx ved at skrive `liste_af_funktioner[0](3)`. Brug en liste af funktioner til at erstatte if-udtrykkene i `vores_funktion`.

●●●● **Opgave 5.6.7:**
Lav en funktion der returnerer det inderste element givet ved funktionen

```
def wrap(tal, gange):
    wrapped = [tal]
    for i in range(gange):
        wrapped = [wrapped]
    return wrapped

# fx kan vi kalde
hvad = wrap(5, 100)
print(hvad)
>>> [[...[[[5]]]...]] # ... repræsenterer mange []
    ↪ parenteser
```

Hint: Man kan tjekke om hvis noget er en liste ved at skrive `type(x)== list`, brug denne information samt en while-loop

●●●●●● **Opgave 5.6.8:**
En **matematisk** funktion $f : X \to Y$ er defineret som en relation mellem to mængder X og Y. Helt formelt kan man definere en matematisk funktion, f, som en mængde af ordnede par, hvorved der gælder det følgende udsagn:

$$\forall x \in X \forall y \in Y \forall z \in Y : (x, y) \in f \land (x, z) \in f \implies y = z.$$

Dvs. for ethvert input x til funktionen, hvis det gælder at $f(x) = y$ og samtidigt $f(x) = z$ så må $y = z$.
Givet at du nu har modtaget en liste af ordnede par i Python (et ordnet par kan tænkes som en liste med 2 elementer): `[[1,2], [4, 5], [2, 3], [6, 3]]`, så skal du skrive en funktion i Python der returnerer enten en `True` eller `False` afhængigt af hvorvidt listen kan udgør definitionen for en funktion. Antag at et ordnet par ikke kan optræde flere gange i listen. Hint 1: start med at skrive `def er_en_funktion(f):` hvor `f` er den liste du får. Hint 2: Prøv at slå op Pythons indbygget funktioner `set()` og `len()` og overvej hvordan du kan bruge disse til at bekræfte hvorvidt `f` er en funktion.

5.7 Algoritmer og biblioteker

I dette kapitel vil vi beskæftige os med algoritmer. Hvad er en algoritme, er det første spørgsmål man bør stille sig selv, når man møder ordet. Forsimplet, så er en algoritme, en sekvens af operationer, der udføres på et input, og derefter terminerer. Læg her mærk til det sidste ord, terminerer. Det er meget vigtigt, at algoritmen terminerer, ellers kan vi aldrig vide om den rent faktisk virker. En algoritme kunne være, en funktion der tager et vilkårligt antal tal og lægger dem sammen, også kaldet en sum. Vi vil i de næste delkapitler kigge på nogle simple algoritmer, som kan bruges til en lang række problemer.

Recursion

Vi starter ud med at diskutere rekursion, som er en bestemt type af algoritme. Rekursion er når en værdi i en mængde afhænger af en eller flere værdier i selvsamme mængde. Man kan for eksempel definere de naturlige tal rekursivt:

$$x_n = x_{n-1} + 1$$

Hvor n går fra $1, 2, ...$ til uendeligt, notationen x_n betyder "det n'te x'er", og vi kalder n for indekset. Vi mangler dog en betingelse for at færdiggøre den rekursive definition. Prøv at tænk over hvad den skal være, inden du læser videre.
For at den rekursive definition er veldefineret, skal man have en startværdi. I tilfældet for værdien af et naturligt tal, vil det være

$$x_1 = 1$$

I Python kan man skrive ovenstående, som et stykke kode.

```
def vaerdi_naturlige_tal(n):
    if n == 1:
        return 1
    else:
        return 1+vaerdi_naturlige_tal(n-1) #Kig tilbage på
            ↪ rekursionsligningen og overbevis dig selv om,
            ↪ at det samme står her i koden.
```

Det her if statement indeholder det vi kalder en *basecase*. En basecase eller startværdi, er noget man ved om nogle af de første værdier i følgen.

●●● **Opgave 5.7.1:**
I stedet for at finde det n'ete naturlige tal (hvilket jo var ret åbenlyst), så vil vi nu forsøge at finde summen af de naturlige af de naturlige tal fra 1 til og med n. Hvis nu n er 5, vil summen være 5+4+3+2+1. Startværdien er her $sum_1 = 1$. Det rekursive udtryk er at $sum_n = sum_{n-1} + n$. Prøv nu at implementer en funktion i python der kan regne ovenstående rekursionsligning.

●●● **Opgave 5.7.2:**
I denne opgave skal du arbejde med fakultetsfunktionen. Fakultet, som skrives $n!$, hvor n er et helt tal større end nul er beskrevet på følgende vis. $n! = n \cdot (n-1)!$ og startværdien er $0! = 1$. Du har fået givet startværdien/basecasen samt det rekursive udtryk. Implementer nu en funktion i Python der kan regne n! ud.

●●●● **Opgave 5.7.3:**
Denne opgave beskæftiger sig med potenser. Vi vil gerne vide, hvad 3^n er, hvor $n \geq 0$. Du må her ikke bruge din viden om at Python bare kan udregne resultatet for dig.

1. Prøv at tænke over hvad en basecase eller startværdi kunne være. Hint: hvad er 3^0?
2. Du har nu fundet frem til hvad basecasen er. Kan du skrive et udtryk op for x_n, som afhænger af x_{n-1}
3. Du har nu skrevet et udtryk op. Implementer det i Python.
4. Ekstra opgave: Generaliser din funktion, så du kan regne x^n

●●●●● **Opgave 5.7.4:**
Fibonacci tallene er en sekvens af tal, som er spændende indenfor naturvidenskab, da forholdet mellem to sideliggende værdier konvergerer mod den gyldne ratio. Den gyldne ratio ses ofte i naturen. Fibonacci sekvensen er defineret, som

$$fib_n = fib_{n-1} + fib_{n-2}$$

Hvor at $fib_1 = 0$ og $fib_1 = 1$ Du skal nu finde det n'te fibonaccital med en funktion i python.

Sortering

Nu går vi videre til en af de vigtigste ting inden for at arbejde med datastrukturer, nemlig sortering. At sortere en mængde, data er essentielt. Vi kan let finde medianen i et sorteret datasæt, max og min værdier samt meget andet. I dette afsnit vil vi kigge på en sorteringsalgoritme nemlig insertion sort. Måden hvorpå insertion sort fungerer er:

- Itererer fra det andet element til det sidste element i listen
- Hver gang du er ved et element sammenligner du det forrige element med det nuværende.
- Hvis det forhenværende element er større en det nuværende, så byt rundt på elementet og det nuværende. Nu vil dit nuværende element være på det $i - 1$ elements plads. Fortsæt dette skridt indtil at elementet er større en elementet forinden eller at elementet er blevet placeret på den aller første plads.

●● **Opgave 5.7.5:**
Givet et array med 3 elementer. Find på hvad de tre elementers værdier kan være for at du skal lave et maksimalt antal sammenligner og iterationer for at sortere de tre elementer.

●●●●● **Opgave 5.7.6:**
Kan du sige noget mere generelt om hvor mange sammenligninger og iterationer man skal lave i det værste tilfælde. Hvad med i det bedste tilfælde?

●●●●●●●●●● **Opgave 5.7.7:**
Denne opgave er til de ekstra nysgerrige. Prøv selv at implementere insertion sort som skrevet foroven. Vink: brug et for loop til at iterere over elementerne og under hver iteration brug et while loop for at lave sammenligningerne og rokaderne af elementerne.

Fremadrettet må I gerne bruge Pythons `sort` funktion.

●●●● **Opgave 5.7.8:**
Find medianen af datasættet `[-1, 5, -5, -9, 1, -17, 3, 10, -3, 100, 73, 64, 902, 123, 653]` Husk: medianen er givet som det midterste element hvis der er et ulige antal elementer og ved et lige antal elementer er gennemsnittet af de to midterste elementers værdier.

●●●●● **Opgave 5.7.9:**
Bamse har N tallerkner, og han vil gerne stable dem hurtigst muligt. Bamse vil gerne stable sine tallerkner ved først at stable de to tallerkner, som er tættest på hinanden og derefter fortsætte denne process. Du skal i dette tilfælde kun hjælpe Bamse med at finde de første to tallerkner, som han skal lægge ovenpå hinanden. Du er givet en liste af punkter og skal ud fra denne liste bestemme afstanden de to tallerkner som ligger tættest på hinanden. `[173.47, 1028.42, 479.5, 127.82, 802, 386.5, -102, 2034.2, 17.3, 640]`. Opgaven kan løses på flere forskellige måder.

Two pointers

En metode som kan benyttes til at løse rigtig mange problemer effektivt er two pointer metoden. I two pointer metoden holder man konstant styr på to af sine elementer og deres værdier og bruger disse oplysninger smart.

Forestil jer at vi står på i begyndelsen af en endelig gang, hvor der er lokaler på venstre side. Vi ved ikke hvor mange lokaler der i alt, men vi har fået af vide af vores klasselærer, at vi skal finde det midterste lokale. Vi kunne godt bare tælle alle lokalerne, men vi kan faktisk gøre noget andet som er super smart. Vi er to personer, og vi lægger den strategi, at hver gang jeg passerer et lokale, så skal du passere to lokaler. Når du så når det sidste lokale så gangen råber du tilbage til mig "Der er nu ikke flere lokaler. Du har fundet det midterste". Kan du se hvorfor dette er sandt?

Ideen er her, at det midterste lokale nødvendigvis antallet er lokaler divideret med to. Fordi du går to lokaler frem ad gangen, og jeg kun et, vil jeg have gået halvt så langt som dig, når du har nået enden af gangen. Derfor er jeg altså ved det midterste lokale.

Vi vil nu opstille et eksempel, denne gang med kode, hvor two pointers teknikken bruges. Vi er givet et sorteret array af tal, og vi har fået at vide, at to af tallene lagt sammen er lig x. Da tallene er sorterede ved vi at den mindste sum må være det første element lagt sammen med det andet element, og at den største sum må være når vi lægger den sidste værdi sammen med den næstsidste. Vi kan derfor starte med at kigge på summen af det første og det sidste element. Hvis denne sum er mindre end x, så vil vi gerne have en større sum. Dette kan vi nu kun gøre ved at tage og element det andet element sammen med det sidste. Hvis nu summen er for stor, så skal vi nu gøre summen mindre. Dette kan gøres ved at vælge det næst sidste element og lægge det sammen med det andet element. Denne process fortsættes indtil, at man finder en sum som er lig x.

```
arr = [1, 3, 18, 23, 103, 527, 700, 1903, 12039]
a = 0
b = len(arr)-1 #Lister er jo nul indekserede
x = 550
while a <= b:
    summen = arr[a]+arr[b]
    if summen == x:
        break
    elif arr[a]+arr[b] > x:
        b -= 1
    else:
        a += 1

print(f"De to tal der lagt sammen gav {x} var {arr[a]} og {
    ↪ arr[b]}")
```

●●●●●● **Opgave 5.7.10:**
Find den mindste sum af to tal i en liste, som er større end x. Vink: prøv at tag inspiration fra den kode du ser foroven.

●●●●●●●● **Opgave 5.7.11:**
En delsum er defineret som at man tager elementerne fra et index til et andet index og lægger disse værdier sammen. Givet en liste af tal find den maksimale delsum.

●●●●●●●● **Opgave 5.7.12:**
En delsum er defineret som at man tager elementerne fra et index til et andet index og lægger disse værdier sammen. Givet en liste af tal find den maksimale delsum.

●●●●●●●●●● **Opgave 5.7.13:**
Roter elementerne i et array k gange. Med rotation menes, at man rykker elementerne en til højre, og hvis elementet er det sidste element rykkes det til start.

Biblioteker

I Python kan man importere kode som andre har skrevet for at gøre ens liv lettere, man kalder disse kode for "biblioteker". Der findes masser af Python biblioteker hvis man googler på internettet, lad os kigge på de biblioteker der hedder `math` og `random`. Du kan importere et bibliotek ved at skrive:

```
import random
```

Her vil random biblioteket blive importeret, og den har en række funktioner som vi kan kalde. Google evt. "random library python" for at se hvilke funktioner den har. Som et eksempel kan vi genererer et tilfældet decimaltal mellem 0 og 1 ved at skrive:

```
import random

tilfældigt_tal = random.random()
print(tilfældigt_tal)
>>> 0.32935699857938416
```

Med `math` biblioteket kan vi kalde matematiske funktioner

```
import math

print(math.pi) # Konstanten pi
>>> 3.141592653589793

print(math.exp(1)) # Konstanten e
>>> 2.718281828459045

print(math.factorial(6)) # Beregner 6!
>>> 720
```

Når vi importerer et bibliotek skriver vi som regel `import bibliotek`, derefter for at kalde bibliotekets funktioner kan vi skrive `bibliotek.funktion()`. Vi kan også importere specifikke funktioner fra biblioteket ved at skrive `from bibliotek import funktion`, så behøver vi ikke at kalde funktionen ved at skrive `bibliotek.` foran. Fx kan vi så skrive:

```
from random import random

print(random())
>>> 0.6595022595617526
```

5.8 Er datalogi en naturvidenskab?

Datalogi i sig selv kan ikke helt kaldes for en naturvidenskab, datalogiens ontologiske status bygger på matematikkens, og især indenfor klassisk datalogi (vi kigger bort fra kvantecomputere), så opnås ny viden i datalogi ikke gennem den naturvidenskabelige metode, men i stedet gennem den aksiomatisk-deduktive metode. Resultater vedrørende algoritmer og køretider i datalogi bliver bevist vha. matematiske metoder, og er fundamentalt anderledes end fx fysik, kemi eller biologi, hvor man bl.a. anvender induktive

metoder ved at indsamle mange observationer. Tilgengæld, anvendes den naturvidenskabelige metode i et meget specifikt område indefor programmering, og det kan argumenteres, at det er en af reneste anvendelser af den naturvidenskabelige metode, eller i hvertfald den deduktive metode.

Den naturvidenskabelige debugging

Debugging, eller fejlfinding, er det vigtigste evne for alle programmører at mestre. Det er et kendt ordsprog blandt programmører at kun 20% af den tid man bruger i udvikling, skriver man rent faktisk kode, de resterende 80% af tiden er brugt på at fejlfinde. Processen i at fejlfinde, er en ren naturvidenskabelig proces. Det handler om at man vil gerne identificere en *årsag* til et *symptom* som man *observerer* i ens program, hvilket kan gøres vh.a. den deduktive metode. Som programmør opstiller man så en række *hypoteser* som man tester af, i håbet om at finde hvad symptomet egentlig skyldes.

Identificering af symptomet

Symptomet, eller fejlen, er noget der ikke stemmer overens med vores forventning om hvordan vores program bør have eksekveret. Symptomet skal præciseres i sprog før man er i stand til at anvende den naturvidenskabelige metode, fx er følgende eksempler på en god identificering af symptomet vs dårlige identificeringer. Her er der taget udgangspunkt i at man fx har en funktion der ikke returnerer det rigtige output:

1. Dårlig identificering, denne er en hypotese, ikke en identificering: "Min funktion returnerer ikke den rigtige output fordi jeg ikke kørte den med de rigtige parametre"
2. Dårlig identificering, vagt udsagn: "Programmet laver ikke det den skal"
3. God identificering: "Funktionen returnerer en `None` value, hvor den skulle have returneret en liste med 5 tal.

Man skal præcisere hvad man mener når man siger at ens program ikke virker, på hvilken måde burde det have virket? Kom med tekniske detaljer på hvad der skulle have returneres og hvor henne. En identificering af symptomet kan tænkes som det man læser direkte fra skærmen, uden at man har gjort nogen tanker omkring hvad det kan skyldes.

Hypotese-dannelse

En god identificering af symptomet gør typisk at man meget hurtigt kan danne sig nogle hypoteser som man vil forsøge at teste af. Man skal desuden være meget opmærksom på de antagelser som man har gjort sig for at have nået til denne fase. Det er vigtigt, at man ikke allerede i starten da man skulle identificere symptomet, laver for mange antagelser, for så vil man kunne risikere at bruge lang tid på at jagte efter en falsk hypotese, som man ikke ville have gjort hvis man har identificeret symptomet klart og tydeligt. Fx kunne et eksempel være at man tror at man har kaldt en specifik funktion før man kørte ens kode, det er en antagelse som kunne være forkert, og i lyset af at man har antaget noget forkert, kommer ens hypoteser heller ikke til at være baseret i virkeligheden.

Selve hypotese-dannelsen kommer så efter man har identificeret symptomet. Vi kan kalde vores symptom for p som er blot et udsagn eller påstand, fx fra tidligere eksempel:

$$p : \text{Funktionen returnerer en } \texttt{None} \text{ value}$$

Derefter er vores arbejde at forsøge om at finde årsagen q hvor det gælder at $q \implies p$, som læses "q medfører p". Et simpelt eksempel på q kunne være:

$$q : \texttt{return} \text{ statement var blevet kommenteret ud ud}$$

Som kan verificeres hvis man læser det sted i koden hvor man har ens `return` statement. Men det er ikke nok at verificere q, vi skal også vide at den q som vi har fundet, rent

faktisk forårsager vores symptom. Her benytter man en kendt slutningsregel, *modus ponens*, som siger at vi skal vide at q eksisterer, og samtidigt at $q \implies p$ vi kan sige at q er den rigtige årsag. Det kan være en af årsagerne, der kan også eksistere flere andre årsager som potentielt kan resultere i det symptom vi observerer, altså kan der være tale om en række årsager, $q_1, q_2, ..., q_n$, der hver især medfører p. Vores job er så at udelukke alle de årsager som ikke findes, dvs. vi skal konstatere at de ikke er tilstede i vores program, her kan være smart at benytte *kontraponering* som kommer fra *bevis ved kontraposition*. Typisk har flere af disse mulige årsager $q_1, q_2, ..., q_n$ også visse "side-effects", $s_1, s_2, ..., s_k$, som er andre konsekvenser der kan ske udover det specifikke symptom som vi observerer. Vi ved **på forhånd** (typisk gennem erfaring) at disse side-effects sker hvis q'erne sker. Disse side-effects kan være noget som vi meget nemt kan bekræfte eller afkræfte ved at observerer programmets opførelse. Vi vil så være sikker på at en årsag q_1 medfører en række side-effects: s_1, s_2, s_3, så kan vi blot forkaste en af disse (kommer an på hvad der er nemmest) for at modbevise q_1 findes, denne form for kontraponering af en implikation ser således ud i logisk form:

$$q \implies p \equiv \neg p \implies \neg q$$

Hvor symbolet $\neg$ betyder "ikke" eller "negation", og $\equiv$ betyder "ækvivalens", som i at de to slutningsformer er ækvivalente. Et konkret eksempel kunne være at q stod for "Det regner" og p var "Jorden er våd", implikationen $q \implies p$ har så den tolkning: "Hvis det regner, så bliver jorden våd". Ved kontraponering kan vi så slutte at $\neg p \implies \neg q$, hvilket betyder "Hvis jorden ikke er våd, så regner det ikke". Kontraponering er et af de stærkeste værktøjer man har når man skal fejlfinde.

Konkret debugging eksempel

Lad os tage udgangspunkt i det tidligere eksempel hvor man havde en funktion der returnerede et `None` frem for en liste af 5 tal. Vi har umiddelbart identificeret symptomet direkte ved at sige helt konkret på det kode-mæssigt niveau. Vi har nu muligheden for at opstille flere forskellige hypoteser for hvad er årsagerne til dette symptom, de hypotetiske årsager kunne fx være:

1. q_1: `return` udtrykket var blevet udkommenteret
2. q_2: Der var en stavefejl da vi kaldte funktionen med dens navn, altså var der en anden funktion der hed det samme
3. q_3: Der var sket en kvantemekanisk fluktuation i computerens hukommelse som slettede listen
4. q_4: Vi havde Chrome-browseren åbent samtidigt da vi kørte koden

Det ses at q_3 en sjov hypotetisk årsag, som ikke nødvendigvis er forkert, da den vil sagtens gør at listen ville blive slettet, men hypotesen er altså svært at bekræfte eller afkræfte hvilket gør at den ikke nødvendigvis er en særlig god hypotese. Man kunne så sige, at sandsynligheden for at q_3 sker er ekstrem lille, måske hvis man derfor kørte funktionen en anden gang og den så lykkedes, så ville man have noget grundlag for at acceptere hypotesen, men det er stadigt meget tvivlsomt om hvis man ville det.

Vi ser at q_4 er nemt at teste for, dog allerede på forhånd ser vi at den er en lidt dum hypotese da det slet ikke klart er hvordan ens Chrome-browser vil kunne påvirke den Python kode som vi kører. Der skal være en meget mystisk forklaring bag hypotesen for at gøre q_4 rent faktisk medfører vores p. Hypotesen kan nemt afkræftes hvis vi slukker Chrome-browseren og kører koden igen (eller kan vi det? Se en af opgaverne), og man burde egentlig ikke have dannet denne hypotese til at begynde med.

Til sidst ser man at q_1 og q_2 er eksempler på nogle mere fornuftige hypoteser, der er nemt at tjekke og som ville samtidigt kunne medføre vores p, hvis de rent faktisk var sande. Men lad os antage nu at vi har tjekket q_1 og q_2 ved at læse i koden, og vi ser at de ikke er tilstede, hvad kan vi så gøre? Vi bliver så nødt til at opstille andre forklaringer, fx

kunne man spørger om hvis koden bliver rent faktisk gemt, altså kunne man så forsøge at modbevise at ens kode er gemt, så ville ens hypotese være

$$q : \text{Koden er gemt}$$

Vi ser at en umiddelbart konsekvens/side-effect kunne være

$$s : \text{Alle } \texttt{print} \text{ udtryk vi skriver ville blive printet}$$

Så kunne en taktik for at modbevise q være at vi skrev nogle `print` udtryk i funktionen, hvis vi så observerer at de *ikke* bliver printet til konsollen, så ville vi kunne ved kontraponering konkludere at koden ikke blev gemt (eller i hvertfald ikke gemt ordentlig). Denne slutning kan vi lave fordi vi ved på forhånd at $p \implies s$, dermed betyder det at $\neg s \implies \neg p$. Vi vil måske så se at vi ikke har trykket på Ctrl-S for at gemme, og vi vil have god grund til at acceptere det som årsagen til vores symptom hvis vi ser, efter vi har gemt koden, at den kører som vi havde forventede.

Perspektivering af debugging mod andre naturvidenskab

Det kan derfor argumenteres at fejlfinding i programmering er nok den reneste anvendelse af den deduktive metode i naturvidenskabelige sammenhæng. Debugging er et fantastisk eksempel på det deduktive metode, fordi software er uafhængigt af tid, betyder det at hvis man kørte noget kode i dag, ville det ikke køre anderledes i morgen hvis ikke man har ændret i koden eller opdateret ens system. Denne tids-uafhængighed giver os muligheden for at eksperimentere med ændring til en variabel ad gangen, og gør at vi kan være ret sikker på at vi har fundet fejlen. I modsætning til andre naturvidenskab, kan det nogle gange være svært at sige hvad har forårsaget en begivenhed, da man ikke kan gå tilbage i tiden. Nogle gange har man kun en enkel observation til rådighed, hvilket gør at man kun må anvende *abduktion* eller slutning til bedste forklaring, som ikke er særligt tilfredsstillende.

Nu har vi kun nævnt den deduktive metode, men man benytter faktisk også induktion og abduktion i debugging processen. Abduktion, eller slutning til bedste forklaring, bruges når vi genererer vores hypoteser. Hypoteserne er valgt efter hvad der giver bedst mening, og her vil vi altid gerne identificere de mest *sandsynlige* årsager først. Induktion benytter vi når vi generaliserer fra de observationer vi har, det kan være meget nyttigt at identificere symptomet. Fx kunne vi godt observere en mærkelig opførelse blandt de funktioner der kalder en bestemt funktion X, så kan vi generalisere ved at sige at vores p er "Alle funktioner der kalder X har en mærkelig opførelse". Men vi skal passe på når vi benytter disse induktive argumenter, vi husker at vores symptom p er selv et udsagn, som godt kan være forkert. Det er derfor kritisk at vi er helt sikker på at vores identificering af symptomet er korrekt. For at teste at hvorvidt symptomet er sandt, kan vi betragte de side-effects som symptomet ville medføre, og se hvis de ikke er tilstede, og derefter benytte kontraponering for at modbevise at vores symptom var korrekt identificeret.

Tænke-opgaver til debugging og NV-metoden

●● **Opgave 5.8.1:**
Hvordan kan vi være sikker på at $q \implies s$, hvis vi tager udgangspunkt i det konkrete eksempel fra tidligere. Kan vi altid være sikker på at en årsag har den konsekvens som den har? Overvej hvad der ligger grund for denne antagelse.

●● **Opgave 5.8.2:**
Kan vi slutte $p \implies q$ fra $q \implies p$? Overvej hvis q hed "Det regner" og p hed "Jorden er våd". Konkluder hvorfor vi ikke nødvendigvis kan slutte årsagen direkte efter at have observeret en side-effect.

●●● **Opgave 5.8.3:**
Kontraponering gør os i stand til at identificere hvilke årsager **ikke** kan være rigtige, overvej om hvis der findes fordele ved at benytte kontraponering, frem for at forsøge at

pege på en årsag direkte. Overvej det i sammenhæng med at et symptom eller side-effect kan være resulteret på grund af flere *mulige* årsager.

●●●● **Opgave 5.8.4:**
Med udgangspunkt i det konkrete debugging eksempel fra før, hvis vi observerer at koden ikke kører når vi har Chrome-browseren åbent, og observerer samtidigt at den heller ikke kører når vi lukker for browseren, er der så grundlag for at konkludere at årsagen ikke skyldes Chrome-browseren? Tænk q: "Chrome-browseren er åbent", hvis p sker under både q og $\neg q$, er p så uafhængigt af q eller $\neg q$?

●●●●● **Opgave 5.8.5:**
Hvis vi efter at have trykket på Ctrl-S for at gemme, finder ud af at vores kode rent faktisk kørte, har vi så nok grundlag til at sige at årsagen var at vi ikke gemte koden? Prøv at google "underdetermination" og læs Wikipedia artiklen, overvej hvordan begrebet kan være relevant i denne sammenhæng. Giv derefter et eksempel på en anden årsag (som gerne må være absurd), der potentielt kunne have i stedet gjort at vores kode virkede.

●●●●●●● **Opgave 5.8.6:**
Hvordan skal vi vælge de hypoteser vi har lyst til at teste? Kan du tænke på nogle regler/tips for hvad der definerer gode hypoteser? Tænk: hvor mange hypoteser kan man egentlig lave?

5.9 Projekter

Nu har I lært hvordan Python fungerer, så kunne det være sjovt at anvende jeres programmeringsviden i nogle projekter som vi har lavet, her kan I vælge mellem to forskellige projekter: Casino projekt og Approksimation af π projekt. Begge projekter handler om simulation. I casino projektet vil du lære, hvordan man kan bruge Python til at simulere et spil på et casino. I approksimation af pi vil vi benytte viden om Pythagoras, cirkler og andet, til at approksimere pi.

5.10 Casino simulation

Vi vil se på, hvordan det ville gå, hvis du tog på casino og spillede plat eller krone. Fra et datalogisk perspektiv er målet at langsomt bygge nogle funktioner i Python, der i sidste ende vil fortælle dig, hvordan din tur på casinoet gik.

Plat eller krone

I sportsgrene med to hold ser vi tit, at retten til at starte med bolden afgøres med et møntkast. Mønten er et godt værktøj, da den har en 50/50 sandsynlighed for at vise plat eller krone og derfor er fair. Det betyder, at hvert hold ville starte med bolden cirka lige mange gange hvis de spillede flere gange.
For at kunne oversætte den idé til casinoer, er vi først nødt til at lære en smule sandsynlighedsregning. I matematikken bruger man såkaldte Bernoullifordelinger til at beskrive møntkast og det skrives $bern(p)$, hvor p er sandsynligheden for at den lander på plat, f.eks. 50%. En fair mønt vil altid have $p = 50\% = 0.5$.

Lad os nu forestille os, at vi tog på casino, hvor vi vinder 1 kr. hvis mønten viser plat og mister 1 kr. hvis den viser krone. Det er meget almindeligt, at casinoer ikke bruger fair mønter. For at de kan tjene penge på deres spil, er de nødt til at sikre, at du taber flere gange end du vinder. Med andre ord skal mønten vise krone flere gange end den viser plat, hvilket casinoet sikrer ved at ændre på sandsynligheden p. For nu kan vi antage, at $p = 46\% = 0.46$. Hvis vi spillede plat eller krone 100 gange, ville *vi* vinde 46

gange og *casinoet* ville vinde 54 gange, præcis som de vil have. Det er altså ikke længere en 50/50, men snarere end 46/54.

- **Opgave 5.10.1:**
Hvad betyder det i ord, at en mønt er $bern(0.4)$?

Møntkast i Python

Lad os nu forsøge at simulere et enkelt møntkast i Python. Vi vil bruge pakken `scipy` til at hente Bernoulli variablen, der vil fungere som den vægtede mønt, og den grundlæggende kode vil se ud på følgende måde.

```
from scipy.stats import bernoulli
udfald = bernoulli.rvs(0.46)
print(udfald)
```

Variablen `udfald` vil enten tage værdien 1 eller 0, hvor 1 svarer til plat og 0 svarer til krone. Vi skriver værdien 0.46, da vi har valgt, at sandsynligheden for at vinde 1 kr. skal være 46%.
Hvis vi er interesserede i at teste udfaldet for forskellige sandsynligheder, kan vi definere en funktion i Python. Denne funktion skal tage en sandsynlighed som input og lave et møntkast som output.

```
def coin(sandsynlighed):
    udfald = bernoulli.rvs(sandsynlighed)
    print(udfald)

#vi kører nu funktionen med en sandsynlighed der ligger
   ↪ mellem 0 og 1
coin(0.55)
```

```
x = 1
while x < 10:
    #gør noget andet
    x = x + 1
```

Med en funktion kan vi altså hurtigt simulere et møntkast for enhver sandsynlighed, hvor eksemplet ovenover bruger $p = 0.55 = 55\%$.

- **Opgave 5.10.2:**
Definér funktionen `coin` i Python. Funktionen returnerer tallet 0 eller 1, hvorimod vi gerne vil vide, om mønten viser plat (værdi 1) eller krone (værdi 0). Implementér en løsning på dette problem i din `coin` funktion ved hjælp af et `ifelse` statement og `print` funktionen.

Penge på spil

Lad os nu se på, hvordan vi kan bruge ovenstående til at undersøge, hvilken effekt sandsynligheden ville have på et spil plat eller krone i et casino.
Vi kan tilføje en variabel i funktionen `coin`, der fortæller, hvor mange penge du har. Funktionen skal nu være defineret ved:

```
def coin(sandsynlighed, penge):
    #inde i funktionen står indtil videre resten af din
       ↪ kode fra opgave 1.11.2
```

I spillet har vi antaget, at vi vinder eller taber 1 kr. ad gangen. Vi vil gerne implementere denne konsekvens i `ifelse`-statementet fra opgaven ovenover. Idéen er følgende:

```
penge = penge + 1 #hvis vi slår plat
#hvad skal der stå hvis vi slår krone?
```

Når denne idé bliver tilføjet i `coin`, vil funktionen både fortælle os om vi slog plat eller krone og justere pengebeløbet. Du har tidligere lært, hvordan man får Python til at printe tekst og tal samlet i konsollen, hvilket vil være brugbart at implementere i funktionen.

●● **Opgave 5.10.3:**
Vi bygger videre på `coin` fra opgave 1.11.2, men nu med den ekstra variabel *penge*. I dit `ifelse`-statement skal du tilføje, at pengebeløbet bliver justeret efter møntkastet. Derudover skal din `print`-funktion nu ikke blot fortælle, om vi slog plat eller krone, men også hvor mange penge vi nu har.

● **Opgave 5.10.4:**
Prøv at køre funktionen for sandsynligheden $p = 0.46 = 46\%$ og et antal penge under 50. Kører din funktion for evigt, eller stopper den fordi du mister alle dine penge? Hvad nu hvis du kører den flere gange; får du nogensinde flere og flere penge, eller går du i 0 hver gang? Hvad kunne være en forklaring på dit resultat?

●●● **Opgave 5.10.5:**
For bedre at kunne følge udviklingen i spillet, kører vi nu flere møntkast ad gangen. Vi vil gerne tilføje en tredje variabel kaldet *antal*, der fortæller hvor mange gange vi spiller. Funktionen `coin` skal starte på følgende måde:

```
def coin(sandsynlighed, penge, antal):
x = 1
while x < antal:
    #her står din kode fra opagve 1.11.3
    x = x + 1 #dette skal være den sidste linje i din
       ↪ kode
```

Brug ovenstående samt din kode fra opgave 1.11.3 til at implementere `coin` som beskrevet ovenover.

Husets fordel

Vi kan beregne, hvor mange penge vi forventer at tjene på spillet. I spillet beskrevet i hele sektionen, nemlig $p = 0.46 = 46\%$ samt et tab og en gevinst på ± 1, gøres det på følgende måde:

$$profit = 1 \cdot \frac{46}{100} - 1 \cdot \frac{54}{100} = -0.08 = -8\%$$

Vi forventer at miste 8% af de penge, vi startede med. Hvis vi starter med 100 kr., vil vi tage hjem med 92 kr. I casino-begreber siger vi, at *husets fordel* er 8%. På engelsk kaldes denne for the house edge. Mere generelt vil formlen se således ud:

$$profit = gevinst \cdot p - tab \cdot (1 - p).$$

For at få tallet i procent, kan vi gange med 100. Hvis profitten bliver negativ, vil spillet for det meste gå skidt, og hvis profitten er positiv, vil vi forvente, at det går godt. De -8% forklarer, hvorfor vi går i 0 hver gang i opgave. 1.11.4; i gennemsnit vil vi altid tabe.

●● **Opgave 5.10.6:**
Implementér en funktion der kan beregne husets fordel på følgende måde:

```
def fordel(gevinst, tab, p):
    #her skal din kode stå
```

●●●●● **Opgave 5.10.7:**
Vi gemmer nu følgende variable efter at have defineret de to funktioner `coin` og `fordel`:

```
loss = 1
profit = 1
money = 40
probability = 0.46
plays = 1000
edge = fordel(profit, loss, probability) #husets fordel
   ↪ med gevinst og tab lig 1 og sandsynlighed p = 46%
```

Som spiller vil du måske gerne vide, hvad husets fordel er inden du siger ja til at tage på casino og spille plat eller krone.
I denne opgave skal du bruge `ifelse`-statements, `print`-funktionen og `input`-funktionen til at skrive kode, der fortæller spilleren hvad husets fordel er, og spørger om de vil fortsætte. Fremgangsmetoden er følgende:
Hvis *edge*-variablen er positiv (dvs. spilleren har fordelen), antager vi at spilleren gerne vil fortsætte.
Hvis *edge*-variablen er negativ (dvs. at huset har fordelen), skal koden spørge spilleren, om de vil fortsætte på trods af de dårlige odds.
Der er nu to muligheder. Hvis spilleren takker nej, skal koden stoppe, hvilket du kan gøre ved at skrive `exit()` i den relevante del af koden. Hvis de takker ja, skal koden blot fortsætte.

Plot over udvikling

Nu er vi næsten i mål med simulationen af turen i casinoet, men vi mangler en fornuftig måde at fortolke resultaterne på. Det kan være svært at danne sig et overblik over turen, når udfaldet fra hvert møntkast bliver printet i sin egen linje. I stedet vil vi hellere plotte hele udviklingen i vores pengebeløb, så vi får én graf, der fortæller os alt om spillet.

Til dette formål bruger vi pakken `matplotlib` og du skal skrive:

```
import matplotlib.pyplot as grafplot
```

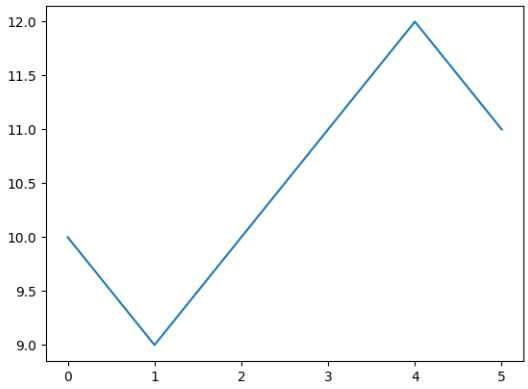
For at kunne plotte udviklingen, er vi nødt til at vide, hvor mange penge vi har tilbage efter hvert enkelt møntkast, så f.eks. 10 kr., 9 kr., 10 kr., 11 kr., 12 kr., 11 kr. og så videre. Vi laver en liste kaldet *udvikling*, der kan gemme alle disse værdier. Derudover bruger vi `append`-funktionen til at tilføje det nye pengebeløb efter hvert møntkast. Et eksempel på dette kunne være:

```
a = [10, 9, 10, 11, 12]
print(a)
a.append(11)
print(a)

>>> [10, 9, 10, 11, 12] #første print
>>> [10, 9, 10, 11, 12, 11] #anden print
```

Efter `while`-loopet i `coin`-funktionen plottes værdierne i *udviklingen*-listen ved hjælp af `grafplot`. Hvis vi fortsætter med eksemplet ovenover, vil plottet blive lavet på følgende måde:

```
    grafplot.plot(a) #her gemmer vi plottet
    grafplot.show() #her får vi Python til at vise os
       ↪ plottet
```



Figur 5.4: Plot over tallene i listen a

●●●●● **Opgave 5.10.8:**
Tilføj kode til din `coin`-funktion, der kan plotte udviklingen i pengene. Du kan følge nedenstående kode:

```
def coin(sandsynlighed, penge, antal):
udvikling = [penge] #tilføjelse: her laves listen og den
    ↪ første værdi er det beløb vi starter med
x = 1
while x < antal:
    #her står din kode fra opgave 1.11.5
    udvikling.append(penge) #tilføjelse: gemmer det nye
        ↪ beløb
    x = x + 1 #dette er fra opgave 1.11.5 og skal
        ↪ stadigvæk være den sidste linje i dit while-
        ↪ loop
grafplot.plot(udvikling) #tilføjelse: her plottes væ
    ↪ rdierne
```

●● **Opgave 5.10.9:**
Nu kan du køre hele koden! Husk at have gemt variablene fra opgave 1.11.7. Tilføj følgende i bunden af din kode og kør hele dokumentet:

```
coin(probability, money, plays)
grafplot.show()
```

Du må også gerne ændre variablene og eksperimentere lidt. Spillet er dit!

● **Opgave 5.10.10:**
Hvis du gerne vil simlere spillet mange gange og få flere grafer i samme plot, kan du skrive følgende i stedet for linjen fra opgave 1.11.9:

```
y = 1
while y < 100:
    coin(probability, money, plays)
    y = y + 1

grafplot.show()
```

Her har vi bare valgt at spille 100 gange, men du kan vælge det tal du har lyst til.

Projekt i at approksimere π

Konstanten π dukker op overalt, og ikke mindst kan vi ved brug af numeriske metoder, forsøge at tilnærme os konstanten. For dem der ikke ved hvad π er, så tænk på runde objekter I har set i virkeligheden. Hvis man tager et rundt objekt, måler dets omkreds og dividerer med dets diameter, så ville man se at uanset hvilket objekt man tager, er forholdet cirka lig $3.14 \approx \pi$. Vi kan så forestille os at hvis vi havde det perfekte runde objekt der havde perfekt lige omkreds, så kan vi tilnærme os π meget præcist med en meget præcis lineal.

I dette projekt tager vi en anderledes tilgang til at approksimere π, vi vil benytte os metoden Monte-Carlo, som handler om at man vha. tilfældighed (tilfældige tal), kan lave nogle rimelig præcise approksimationer. I bedes om at gå ind på `https://bit.ly/3HqDyZE`, som er en Google Colab notebook hvor vi har nogle opgaver klar til jer.

5.11 Konklusion og videre emner

I har nu været igennem en del af datalogiens verden, en meget meget lille del. Der er meget meget mere at lære, og man kan gå i mange forskellige retninger. Så dette afsnit er dedikeret til forskellige retninger og tilhørende ressourcer, så kan I selv kigge dem i gennem.

Googling og StackOverflow

Google, google og google. Det kan ikke understreges nok hvor vigtigt det er at kunne google som programmør, google alt det som I ikke forstå. Der er så mange detaljer vi har undladt at skrive om i dette afsnit for Datalogi, simpelthen fordi det ikke kan lade sig gøres at dække alt omkring programmering, det er derfor vigtigt at I selv finder frem til de kilder ved at google. En fejl som mange typisk laver er at google på dansk, det er svært at finde brugbare resultater hvis man googler på dansk, oversæt dit spørgsmål til engelsk i stedet når du forsøger at finde noget, for ellers bliver du skuffet. Typisk kan man finde svar til ens programmeringsspørgsmål på en hjemmeside der hedder StackOverflow, det er et forum hvor man kan stille spørgsmål til (typisk) mere erfarende programmører (google det :)).

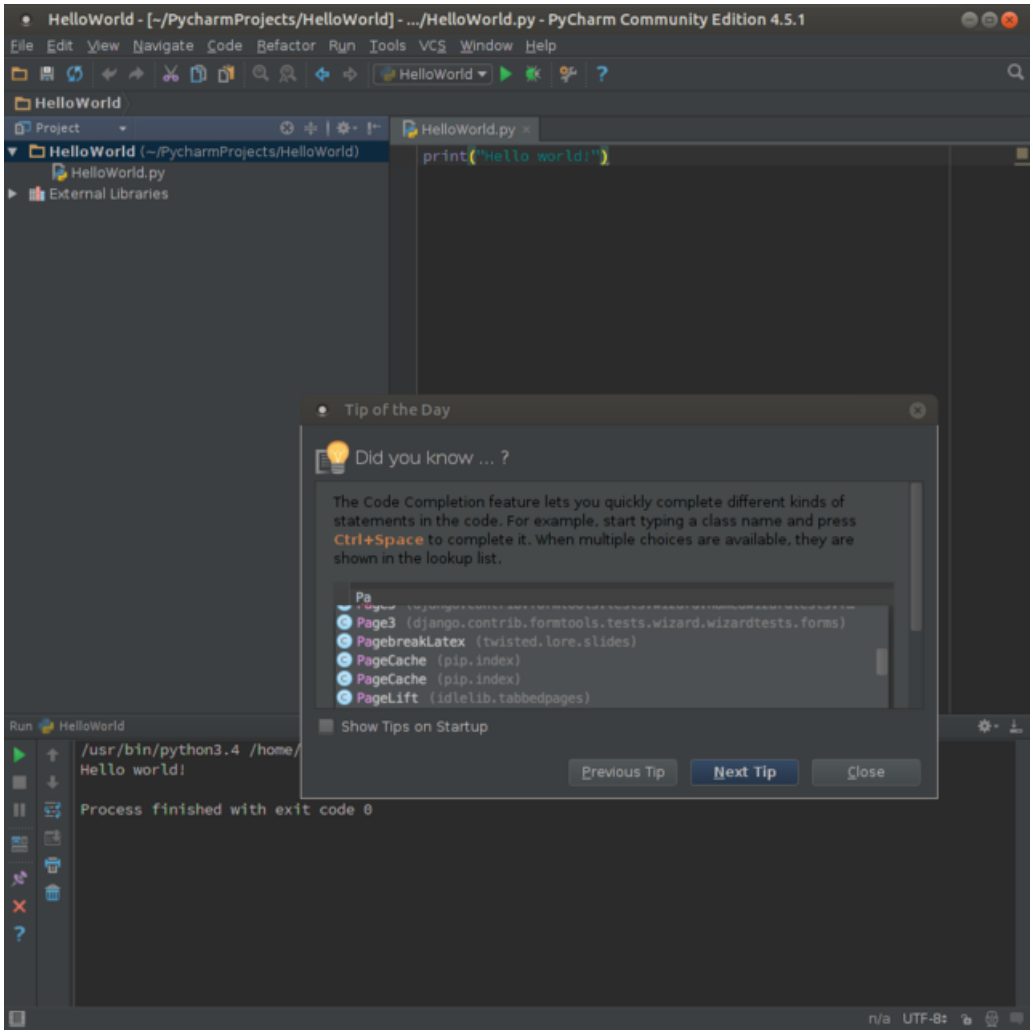
Editors og IDEs

Hvis man vil gerne lave større Python projekter, eller bare kode projekter generelt, så kan det være en god ide at sætte en kode editor op. En kode editor eller IDE som står for Integrated Development Environment, kommer med adskillige features der gør det nemmere for programmører at spotte fejl i deres kode. Vi anbefaler bl.a. PyCharm og Visual Studio, som er nok de mest brugte IDEs.

Andre programmeringssprog

Der findes mange andre programmeringssprog i verden, I kan selv google jer frem til hvilke de er. Her er dog en liste af de mest populære programmeringssprog og deres beskrivelser:

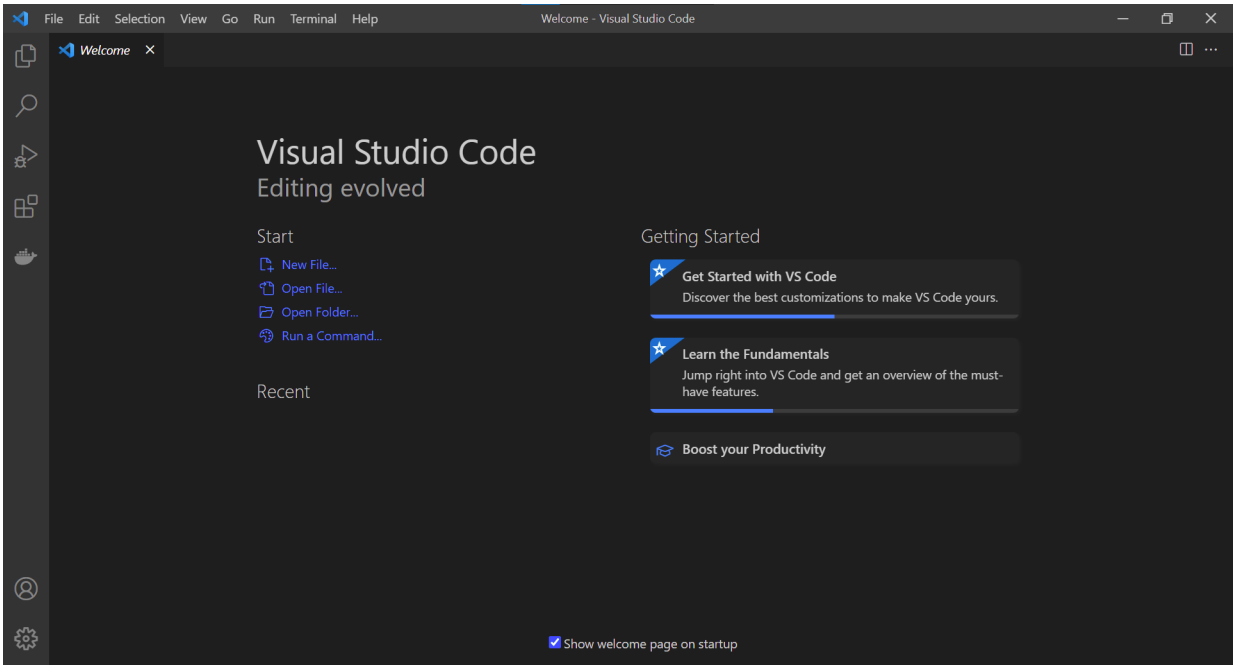
- **Java**: Java er nok en af de mest kendte programmeringssprog, og var helt sikkert den mest brugte i starten af den 20 århundrede. Java programmering følger noget som man kalder for OOP (Object Oriented Programming) paradigmet, hvilket er en bestemt måde at tænke kode på. Det kan være utroligt nyttigt at forstå forskellene mellem programmeringsparadigmerne (google dem!). OOP paradigmet haft en kæmpe indflydelse på alt vi ser på internettet i dag. Mange nye startups i dag er gået væk fra at lave OOP til at lave funktionel programmering, og det er fx noget som JavaScript er kendt for. Java er typisk brugt i store coporate enterprise firmaer, dvs. banker, store administrative bureauer.



Figur 5.5: PyCharm CE (Screenshot er taget fra Wikimedia Commons, public domain)

- **JavaScript**: Har næsten intet at gøre med Java, bare lige heads up. JavaScript er det der kører på ens browser når man loader en hjemmeside, altså bliver den brugt til at lave ret meget frontend-programmering som er programmering rettet mod bruger-vendt interfaces (google hvad det betyder). JavaScript bruges typisk i de meget populære frameworks som er Vue, React Native, Express.js (på node.js) (google!). Man siger at JavaScript er funktionel programmering, fordi man arbejder ikke så meget med at manipulere states på objekter, i modsætning til OOP programmering.

- **C og C++**: C er nok den allerførste moderne programmeringssprog, og C++ baserer sig på C som kommer med nogen flere features. Man skriver C eller C++ hvis man virkelig vil gerne optimere ens kode, og lad den køre hurtigt. Et program skrevet i Python, kan være op til 45 gange langsommere end det samme program skrevet i C (google "Python vs C speed"). Man kan så spørge, hvorfor skriver alle ikke bare C eller C++? Det er et godt spørgsmål, og svaret ligger i at hvis man selv forsøger at lære C eller C++, så indser man hvor svært det er at forstå noget C kode, fremfor at bare køre Python på en bedre computer. Ej, det er lidt overdrevet, det reelle svar er noget i den stil, der er ikke så mange biblioteker/værktøjer til C eller C++ i modsætning til Python. Python er meget mere nemmere at lære og meget mere overskueligt at læse. Man bruger dog C i mange steder hvor det kræver det kører hurtigt. Fx er de fleste styresystemer skrevet i C eller C++, og



Figur 5.6: Visual Studio Code (Screenshot er taget fra Wikimedia Commons, public domain)

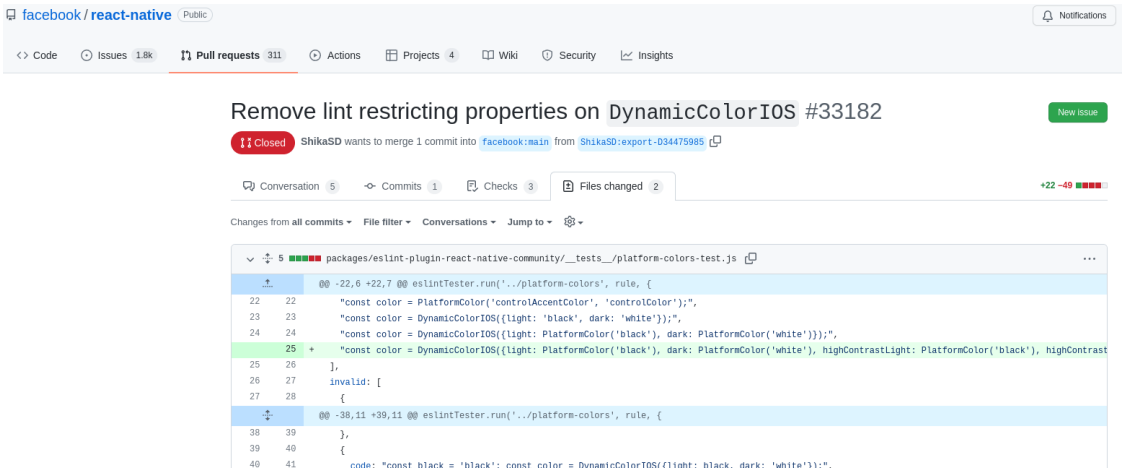
Python er faktisk selv skrevet i C!

- **PHP**: Lad vær med at bruge det her, bare kør med JavaScript og Express.js i stedet...

- **Assembly**: Hvis man vil helt ned til metallet, og arbejde med at flytte bits rundt i en computerens hukommelse, samt fortælle direkte til CPU'en hvad den skal gøre, så skriver man i Assembly language. Assembly er i sig selv ikke er et programmering sprog, men i stedet en kategori af programmeringssprog.

Programmering i virkeligheden

Programmering i den virkelige verden består af mange forskellige områder, en af de vigtigste aspekter inden for software-udvikling er "software deployment". Software er ikke færdig før den kan køres, bare fordi man har skrevet noget kode skal man også sikre sig at koden kører ordentlig på de maskiner og servere man har til rådighed. Typisk når man skriver noget kode, så skal den i gennem en længere proces hvor flere udviklere skal reviewe og godkende ens kode. Man benytter bl.a. værktøjet GitHub, som det man kalder for "version-control" eller "versionsstyring" på dansk. GitHub er udviklet for at sikre at den kode man skriver ikke indeholder bugs og fejl, dette gøres gennem "pull-requests", hvor man beder andre udviklere om at se på ens kode før den bliver skubbet til kodebasen. Derefter skal den kode man har skrevet deployes på forskellige servere, hvis man fx skriver backend kode. Man benytter sig at forskellige cloud-providers, som kunne bl.a. være Google Cloud, Amazon Web Services (AWS) eller Microsoft Azure. Udtrykket "cloud-computing" refererer til brugen af virtuelle computerressourcer, som ikke ligger lokalt hos den udvikler eller firma der arbejder på at udvikle software. Man har typisk kæmpe datacentre andre steder i Europa, hvor al den kode man har skrevet bliver uploadet og kørt. Processen for udvikling og deployment kaldes også for "DevOps". "Dev" udtrykket kommer fra ordet "Development", som betyder udvikling, og "Ops" står for "IT Operations", som er de processer vedrørende deployment af selve koden og maintenance.



Figur 5.7: En GitHub pull request

Figur 5.8: Nogle server racks, formentlig i et datacenter

Datalogiens videnskabsteori

Vi har introduceret jer til den aspekt vedrørende programmering og udvikling inden for datalogiens verden, men datalogi er meget bredere end blot programmering. Programmering er blot en af de praktiske anvendelser af datalogiens teori, men Datalogi har også en teoretisk fundament som kan være interessant at studere for sig selv. Her beskæftiger man bl.a. med spørgsmål om fx Turing-completeness, som er en egenskab af programmeringssprog, tidskompleksitet, som er hvor lang tid det tager at køre en bestemt algoritme, rekursionsteori, kvantecomputerer, Kolmogorovs kompleksitet, og mange flere! Et godt sted at starte på at lære disse emner er gennem Wikipedia, man kan starte med at søge "Computer Science Wikipedia" i Google, og læse hvad man selv finder interessant!

Kompetitiv programmering

Kompetitiv programmering kan beskrives som datalogiens sportsgren. Konceptet går ud på, at man får givet en opgave som man skal løse ved sin datalogiske viden. Ofte skal

man få nogle ideer, så man kan restrukturere problemet til noget man ved hvordan man kan løse. Et aspekt af kompetitiv programmering, som gør det svært, er at man ofte skal komme på de bedste løsninger til problemet. Med bedste løsninger, menes der dem der på noget input kan give et svar, hurtigt. Årsagen til dette er at der er en tidsbegræsning på hvor lang tid det må tage koden, at give output ved et givent input. Kompetitiv programmering er ikke bare en god måde at udvide sin datalogiske forståelse, men også et middel til at udvikle sine problemløsningsevner. Der findes mange hjemmesider med problemer, og vi kan varmt anbefale følgende.

- `https://cses.fi/`
- `https://codeforces.com/`
- `https://open.kattis.com/`
- `https://leetcode.com/`
- `https://onlinejudge.org/`

Det anbefales desuden at man starter med problemer som har lav rating og derefter bevæger op. Man kan komme ret langt med betinget udførsel, loops, dictionaries, arrays osv. Derefter har man brug for at lære teori. Det første link har tilknyttet en god bog til at lære basal teori.

Kapitel 6

Medicin

6.1 Hvad er medicin?

Hjertet er et af de vigtigste organer, vi har i kroppen. Det sidder i brystet, godt gemt og beskyttet af muskler og knogler. Hjertet er motoren, der får hele kroppen til at køre. Dets funktion er at pumpe alt det blod vi har, rundt i kroppen, så vi får ilt og energi helt ned til tæerne. Og det gør hjertet, uanset om vi tænker over det eller ej. Hvor hurtigt og hvor hårdt hjertet slår, afhænger dog af, hvad vi laver. Det er blod, som hjertet arbejder så hårdt for at pumpe rundt. Blod indeholder alt den energi og ilt som vi har brug for. Når man bare ser på blod med det blotte øje, er det bare en varm, rød, væske. Men hvis vi kigger nærmere, kan vi se at blod indeholder mange forskellige celler, som alle bidrager til kroppens funktion. Det var hjertet og blod - helt kort, lad os nu grave lidt dybere ned i de forskellige aspekter.

Overordnet overblik over kroppen:

Hjerne

Vores hjerne - sammen med rygmarven, som sidder i vores rygrad i forlængelse af hjernen - er vores centralnervesystem. Det er her de komplekse interaktioner mellem nerveceller, neuroner, giver ophav til vores bevidsthed, vores tanker, vores forståelse af sanseindtryk, vores personlighed. Alt hvad vi vælger at gøre har ophav i hjernen - hver eneste ord, hvert skridt. Der går forbindelser fra overalt rundt om i kroppen til hjernen, og fra hjernen rundt i hele kroppen.

Knogler

Vores skelet består af knogler - 206 knogler. Og de knogler har mange vigtige funktioner, nogle mere åbenlyse end andre. For det første giver vores knogler os stabilitet. Knogler er faste og giver noget som vores muskler kan hæfte sig til. Knogler beskytter også vores mere skrøbelige dele af kroppen: rygmarven beskyttes af rygraden, hjernen beskyttes af kraniet, vores lunger og hjerte beskyttes af ribben. Knoglerne er dog ikke kun som mursten blokke vi bruger til at bygge os selv, de har også meget aktive funktioner. Knoglerne har netop et hårdt, fast ydre, men indeni finder man knoglemarven. Her starter processen der laver vores blodceller - både røde som hvide. Vores knogler består hovedsageligt at to ting: calcium og fosfat. De to stoffer skal man have en balance af i kroppen, og knoglerne fungerer som et depot for dem. Hvis man får meget calcium vil noget af det bliver lagret i knoglerne, mens hvis man har for lidt calcium i resten af kroppen vil noget blive frigivet fra knoglerne.

6.2 Hjertets anatomi

Hjertet er en muskelpumpe der er placeret i brystkassen (thorax) mellem lungerne, hjertet er skråtstillet og ligger ikke helt centralt men er lidt til venstre for kroppens midterlinje, hvilket gøre at den venstre lunge er lidt mindre end den højre. I oprejst stilling ligger hjertet bag de nederste to-tredjedel af os sternum (brystbenet) og er beskyttet af ribbenene (costae). Hjertet hviler på den muskulære membran, kaldet diaphragma, som adskiller brysthulen fra bughulen. Lige under mellemgulvet, i den øvre del af bughulen, ligger Leveren, højre side af leveren hviler op ad hjertet. Samlet set ligger hjertet tæt på lungerne og leveren, mens det er langt fra milt, pancreas, tarmene, nyrerne og blæren. Hjertet er omgivet af en membran kaldet pericardium, som hjælper med at beskytte hjertet og holder det på plads i brysthulen. Som mange andre organer vokser hjertet sig med tiden, men fysisk træning og visse sygdomme (fx forhøjet blodtryk) kan også øge størrelsen af hjertet. Hjertet kan inddeles i to forkamre kaldes for atrier og to hovedkamre kaldes ventrikler. Højre atrium og ventrikel arbejder sammen for at pumpe iltfattigt blod til lungerne, mens venstre atrium og ventrikel pumper iltet blod ud til resten af kroppen. Hjertet har også fire hjerteklapper, som åbner og lukker for at styre blodstrømmen gennem hjertet. Hjerteklapperne fungerer som ventiler, der sikrer, at blodet flyder i den rigtige retning gennem hjertet. De fire hjerteklapper er: mitralklappen, aortaklappen, tricuspidalklappen og pulmonalklappen.

1. Mitralklappen: Denne klappen er placeret mellem venstre atrium (VA) og venstre ventrikel (VV). Når ventriklen trækker sig sammen, lukker mitralklappen for at forhindre blod i at strømme tilbage til atriumet. Når ventriklen slapper af, åbner mitralklappen, så blod kan strømme fra atriumet til ventriklen.

2. Aortaklappen: Denne klappen er placeret mellem venstre ventrikel og aorta, kroppens største arterie. Når ventriklen trækker sig sammen, åbner aortaklappen, så blod kan strømme ud i aorta og videre ud i kroppen. Når ventriklen slapper af, lukker aortaklappen for at forhindre blod i at strømme tilbage til ventriklen.

3. Tricuspidalklappen: Denne klappen er placeret mellem HA og HV. Når ventriklen trækker sig sammen, lukker tricuspidalklappen for at forhindre blod i at strømme tilbage til atriumet. Når ventriklen slapper af, åbner tricuspidalklappen, så blod kan strømme fra atriumet til ventriklen.

4. Pulmonalklappen: Denne klappen er placeret mellem højre ventrikel og lungearterien. Pulmonalklappen åbner for at tillade blod at strømme ud i lungekredsløbet, når ventriklen trækker sig sammen, og den lukker for at forhindre blod i at strømme tilbage til ventriklen, når den slapper af.

Blodet transporteres til og fra hjertet vha. kar. Hjertets kar er vigtige for at sikre en effektiv transport af blodet gennem kroppen. Der er to hovedtyper af kar: arterier og vener. Arterier fører blodet væk fra hjertet og ud i kroppens væv, mens vener fører blodet tilbage til hjertet. De vigtigste kar i hjertet omfatter:

1. Vena cava superior og inferior: Vena cava superior (øvre hulvene) og inferior (nedre hulvene) er to store vener, der fører iltfattigt blod fra kroppens øvre og nedre dele til højre atrium.

2. A. pulmonalis: A. pulmonalis (lungearterien) fører iltfattigt blod fra højre ventrikel til lungerne, hvor det optager ilt og afgiver kuldioxid.

3. Vv. pulmonales: Vv. pulmonales (lungevenerne) fører iltet blod fra lungerne til venstre atrium.

4. Aorta: Aorta (legemspulsåren) er den største arterie i kroppen og fører ilt- og næringsrigt blod fra venstre ventrikel til resten af kroppen.

Det system, som transporterer blodet rundt i kroppen kaldes for Hjertets kredsløb. Det består af to separate kredsløb, det systemiske kredsløb og det pulmonale kredsløb.

- Det systemiske kredsløb er ansvarligt for at transportere ilt- og næringsrigt blod fra venstre ventrikel gennem aorta og ud til kroppens celler. Når blodet har afgivet ilt og næringsstoffer til cellerne, bliver det iltfattige blod transporteret tilbage gennem venerne og ind i højre atrium.

- Det pulmonale kredsløb er ansvarligt for at transportere iltfattigt blod fra højre ventrikel til lungerne, hvor det optager ilt og afgiver kuldioxid. Det iltede blod føres derefter tilbage til venstre atrium gennem lungearterierne og lungevennerne.

Begge kredsløb er drevet af hjertets kontraktioner, som skaber et tryk, der driver blodet gennem kredsløbssystemet. Hjertets fire kamre og de fire hjerteklapper spiller dermed en afgørende rolle i at sikre, at blodet bevæger sig i den rigtige retning gennem kredsløbet. Helt specifik kan man inddelle blodets vej gennem hjertet og kroppen i disse trin:

1. I det systemiske kredsløb kommer det ilt- og næringsrige blod fra lungerne ind i venstre forkammer gennem vena pulmonalis(lungevenerne).

2. Venstre forkammer trækker sig sammen, og blodet strømmer gennem mitralklappen (bicuspidalklappen) ned i venstre ventrikel.

3. Venstre ventrikel trækker sig sammen, og blodet pumpes ud gennem aortaklappen og ind i aorta, som fører blodet ud i det systemiske kredsløb.

4. Blodet føres rundt i kroppen gennem arterierne og arteriolerne, hvor det afgiver ilt og næringsstoffer til vævene.

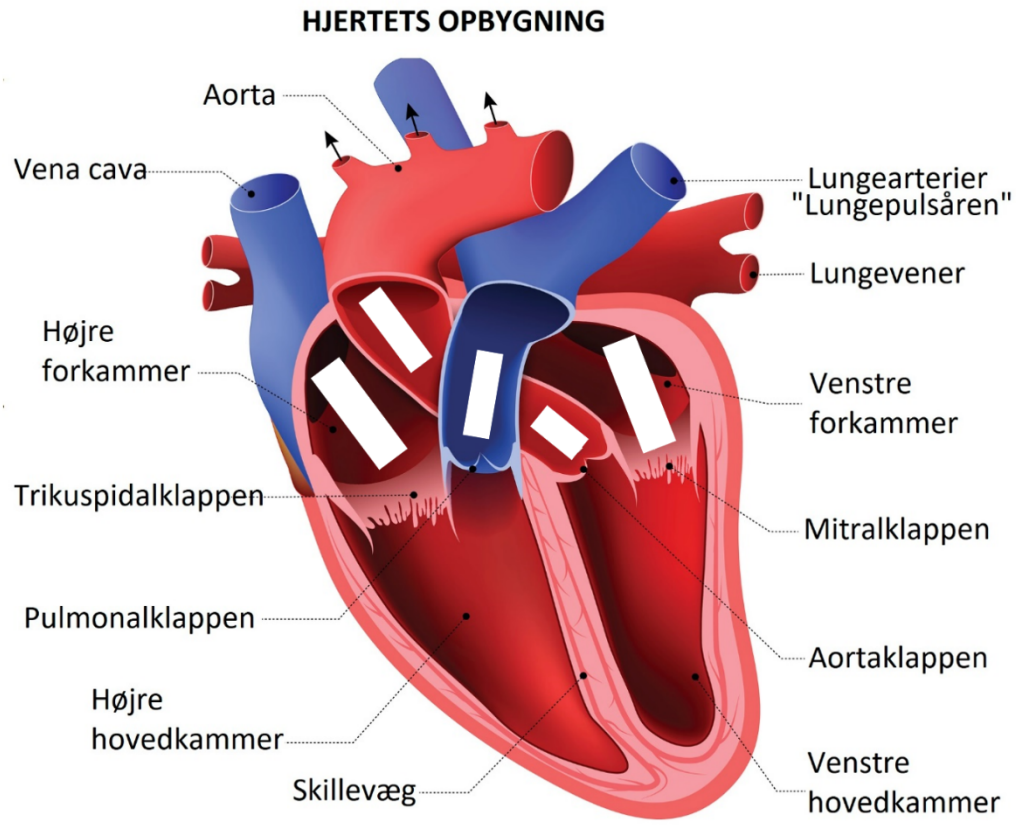
5. I det pulmonale kredsløb kommer det iltfattige blod fra kroppens væv tilbage til hjertet gennem de store vener, og det løber ind i højre forkammer.

6. Højre forkammer trækker sig sammen, og blodet strømmer gennem trikuspidalklappen ned i højre ventrikel.

7. Højre ventrikel trækker sig sammen, og blodet pumpes ud gennem pulmonalklappen og ind i lungearterierne, som fører blodet til lungerne.

8. I lungerne afgiver blodet kuldioxid og optager ilt gennem kapillærerne, og det iltede blod løber tilbage til hjertet via lungevenerne og strømmer ind i venstre forkammer for at starte kredsløbet igen.

Det er vigtigt at bemærke, at hjerteklapperne i både det systemiske og pulmonale kredsløb sørger for, at blodet kun bevæger sig i én retning gennem hjertet og kroppen, og at hjertemuskulaturen i ventriklerne trækker sig sammen rytmisk og koordineret for at pumpe blodet effektivt ud i kredsløbet.



Figur 6.1: På billedet kan man se de nævnte kar og klapper. Prøv at se, om I kan skrive blodets retning ind i de hvide bokse.

6.3 Hjertets fysiologi

Fysiologi er et område af medicin, der handler om funktion. Hjertefysiologi handler derfor om hjertets pumpefunktion. Hjertets fornemmeste opgave er at sørge for, at der er ilt og næring til hele kroppen. Hjertet modtager afiltet blod fra kroppen. Blodet skal først ud til lungerne, hvor det optager en masse ilt. Derefter skal det tilbage til hjertet. Og til sidst bliver blodet sendt ud i resten af kroppen. For at hjertet skal kunne pumpe så meget blod ud i kroppen er det vigtigt at pumpefunktionen er godt koordineret, så der kommer én samlet kontraktion. Mange faktorer spiller ind på, hvor hurtigt og hvor hårdt hjertet pumper. Denne regulering er vigtig for at man kan lave de ting, som kræver lidt mere af kroppen end den er vant til.

Hjertets pumpefunktion

Vi ved nu, hvilken vej blodet løber gennem hjertet, og dets rute gennem lungerne og ud til kroppen. Men hvordan kommer blodet rundt? Hjertets funktion er at pumpe, men hvad betyder det egentlig? Helt kort betyder det, at hjertet først har en afslappet fase, hvor det fyldes med blod og dernæst har en fase, hvor det trækker sig sammen og skubber blodet ud til enten lunger eller resten af kroppen.

Den fase, hvor hjertet fyldes med blod, kaldes diastolen. Dia- betyder åben og -stole betyder at sende. Højre atrium fyldes af afiltet blod, der kommer fra hele kroppen. I diastolen åbnes tricuspidalklappen og højre atrium trækker sig sammen. Det betyder, at blodet fra højre atrium løber ned i højre ventrikel. Samtidigt løber blod fra lungevenerne ind i venstre atrium, som fyldes med iltet blod fra lungerne. I diastolen åbnes mitralklappen og venstre atrium trækker sig sammen, ligesom på højre side. Dermed løber blodet fra venstre atrium ind i venstre ventrikel. I diastolen bliver klapperne mellem atrier og

ventrikler åbnet og blodet fra atrierne bliver sendt til ventriklerne. Når ventriklerne er fyldt med blod kan systolen starte. Sys- betyder sammen (som i sammentrækning). Højre ventrikel er nu fyldt med blod, der kom fra højre atrium. Under systolen lukkes tricuspidalklappen, pulmonalklappen åbner og højre ventrikel trækker sig sammen. Dermed bliver blodet sendt gennem pulmonalarterien til lungerne. Venstre ventrikel er samtidigt også fyldt med blod, der kom fra venstre atrium. Under systolen lukkes mitralklappen, aortaklappen åbnes og venstre ventrikel trækker sig sammen. Herfra bliver blodet sendt ud gennem aorta til resten af kroppen. Denne cyklus mellem diastole og systole kører hele tiden. Man kan høre klapperne der åbner og lukker, når man lytter på hjertet med et stetoskop.

Blodtryk

Når blodet pumpes ud af hjertet, bliver det presset ud i arterierne med en vis kraft. Blodet i hele kredsløbet bliver hele tiden trykket fremad af hjertet. Dette tryk kaldes for blodtrykket. Et blodtryk består af et systolisk blodtryk og et diastolisk blodtryk. Det systoliske blodtryk er det tryk som opstår i blodkarrene under systolen, hvor ventriklerne trækker sig sammen. Det er på det her tidspunkt, at trykket er højest, og man taler derfor om at det systoliske tryk, er det største tryk, som hjertet kan lave. Når hjertet slapper af i diastolen, kan man måle det diastoliske blodtryk. Det diastoliske tryk er altså det laveste blodtryk, som man kan måle i blodkarrene.

Blodtryk måles i en lidt mærkelig enhed, som hedder mmHg ("millimeter kviksølv"). I gamle dage, da man målte blodtryk, brugte man en lidt kompleks målemetode. For at måle blodtrykket brugte man en cylindrisk beholder. I den skabte man et vakuum ved hjælp af kviksølv. Jo højere et blodtryk, des mere ville overfladen på kviksølvet stige. I dag bruger man ikke længere kviksølv til at måle blodtryk. Man bruger i stedet et sphygmomanometer. Sphygmomanometeret består af en oppustelig manchet og en lille computer, der kan vise blodtrykket. Manchetten pustes op og luften tømmes langsomt ud af den igen. På den måde kan maskinen måle både det systoliske og diastoliske blodtryk. Man siger at et normalt systolisk blodtryk er omkring 120, mens det diastoliske skal ligge omkring 80. Det skriver man på den her måde: 120/80 ("120 over 80"). Det betyder at hjertet i systolen laver et tryk der er stort nok til at få kviksølvsøjlen til at stige med 120 mm. Under diastolen stiger søjlen med 80 mm.

Regulering af blodtrykket

Det er vigtigt at kunne kontrollere sit blodtryk. Det er ikke noget man kan gøre bevidst, men der er mange faktorer der spiller ind i regulering af blodtrykket. Man kan beregne blodtrykket ved ligningen: $\text{MAP} = \text{CO} \cdot \text{TPR}$.

CO: Cardiac output (minutvolumen)

Hjertets minutvolumen er den mængde blod, som hjertet kan pumpe rundt på 1 minut. En normal CO ligger omkring 5 L/minut. Det svarer nogenlunde til at alt blodet i ens krop kommer igennem hjertet på 1 minut! CO kan ligesom blodtrykket variere og afhænger meget af, hvor meget ilt og næring kroppen skal bruge. Jo mere ilt der er behov for, des større skal CO være, så man kan tilbyde kroppen den ilt, der skal bruges.

CO regnes ved ligningen: $\text{CO} = \text{HR} \cdot \text{SV}$. HR (hjerterytme) er hvor hurtigt hjertet slår. Det er også det man kan mærke som puls. Hjertet er meget god til at regulere puls. Pulsen kan stige og falde meget hurtigt, alt afhængigt af, hvor stort et behov der er for ilt i kroppen. SV (slagvolumen) er hvor meget blod der bliver pumpet ud af hjertet i ét slag. Man kan også se det som, hvor hårdt hjertet slår. Størrelsen på SV kan ikke ændre sig lige så meget og lige så hurtigt som pulsen. Men med længere tids træning kan SV også blive større.

Lige så snart man går i gang med en aktivitet der kræver mere ilt, eksempelvis sport begynder hjertet altså at slå hurtigere og hårdere. Det gør, at der kan pumpes mere blod ud i kroppen, som kan bruges af musklerne, som bruger mere ilt, når de arbejder.

TPR: Total perifer resistance

Når blodet kommer fra hjertet ud i blodkarrene, begynder det lidt efter lidt at løbe langsommere. Det skyldes at der er modstand i blodkarrene. Når blodkarrene har en lille diameter, er det sværere for blodet at løbe igennem, og der er stor modstand. Har blodkarrene til gengæld en stor diameter, er det lettere for blodet at løbe igennem, og der er lille modstand. Generelt bliver modstanden større, jo længere væk man kommer fra hjertet, da blodkarrene bliver mindre og mindre. Men et blodkar har også muligheden for at ændre lidt på sin diameter. Blodkar har kontrahere (trække sig sammen) og dilatere (slappe af), hvilket gør deres diameter mindre og større. Når en bestemt muskel bliver brugt, vil blodkarrene der løber til den muskel dilatere, så der kan komme mere blod ud til den muskel. Bliver en muskel ikke brugt vil blodkarrene til den muskel kontrahere, og der kommer dermed mindre blod ud til denne muskel. Hvis man eksempelvis løber en tur vil blodkarrene i ens ben generelt være dilaterede, mens blodkarrene til ens tarme er mere kontraherede, da man jo ikke bruger sine tarme til at løbe.

Hjertets elektriske ledning

For at kunne lave en god kontraktion, der kan pumpe så meget blod som muligt ud af ventriklerne under systolen er det vigtigt at hele hjertet trækker sig sammen på samme tid. Det gør hjertet helt uden at vi skal tænke over det. I venstre atrium sidder en lille gruppe af særlige hjertemuskelceller, som vi kalder for sinusknuden. Fra sinusknuden udgår en elektrisk impuls i en regelmæssig rytme, som dikterer hvor hurtigt hjertet skal slå. Impulsen spreder sig først til begge atrier, som trækker sig sammen først i diastolen. Der er kun ét sted i hjertet, hvor den elektriske impuls kan løbe fra atrierne til ventriklerne. Det sted kaldes for AV-knuden. Når AV-knuden modtager impulsen har den en vigtig funktion. Her skal impulsen nemlig forsinkes, bare et lille øjeblik. På den måde får hjertet lov til at lave diastolen. Herefter sendes impulsen ned gennem ventrikelseptum og ud til ydervæggene af ventriklerne. Først når alle dele af begge ventrikler har modtaget signalet, trækker ventriklerne sig sammen og der bliver lavet en kraftig uddrivning af blodet fra ventriklerne: Systolen.

- Iltning (samspil med lunger) Ligger mellem lungerne og arbejder tæt sammen med dem.

6.4 Blodets komponenter

Vores blod består af mange forskellige komponenter. Disse komponenter har hver deres rolle som er enormt vigtige for kroppen. Her gennemgås de røde blodceller, de hvide blodceller, blodpladerne, koagulationsfaktorer og det der er tilbage, serum.

Serum

Blodets serum, som er størstedelen af blodet på ca. 55%, indeholder alle de dele af blodet som ikke er celler eller blodstørknende komponenter. Dette inkluderer blandt andet hormoner, som skal rejse gennem blodet og fungere som signalstof fra ét organ til andet væv. Der findes derudover adskillige proteiner, elektrolytter, antistoffer samt mere.

Blodplader

Blodplader, som også hedder trombocytter, er fragmenter fra meget store celler der hedder megakaryocytter. Trombocytter flyder frit rundt overalt i vores blod. Trombocytter er helt essentielle for blodets evne til at størkne. Trombocytter klistrer til hinanden og til der hvor såret er opstået og stopper dermed blødningen. Mangel på raske trombocytter kan derfor lede til en tilstand hvor sår bliver ved med eller tager meget længere tid før de stopper med at bløde. Trombocytterne udgør kun en meget lille del af blodets komponenter.

Koagulationsfaktorer

koagulationsfaktorer er proteiner, som er meget mindre end celler eller trombocytter, der findes i blodet og gennem en kompleks række af reaktioner med trombocytter og i tilstedeværelsen af et sår leder til koagulation og dermed stoppes en blødning. Koagulation kræver både blodplader såvel som koagulationsfaktorer og andre stoffer ikke nævnt her.

Hvide blodceller

Hvide blodceller hedder leukocytter og er ansvarlige for vores immunsystem, der bekæmper sygdomme. Leukocytterne er kun en lille brøkdel af blodet og udgør kun ca. 1% af blodets komponenter. Vi har både leukocytter der reagerer hurtigt og uden at tilpasse sig sygdommen, og så har vi mere specifikke leukocytter der tager længere tid at reagere på sygdommen, men kan tilpasse sig - også kaldt adaptere sig - til sygdommen og lave en "hukommelse", så vi er bedre beskyttet mod den sygdom i fremtiden. Der er flere forskellige leukocytter både blandt dem der reagerer hurtigt og dem der reagerer langsomt med forskellige funktioner og roller. Blandt de adaptive leukocytter er der dem der kan lave antistoffer. Antistoffer er små proteiner som har meget specifikke mål de angriber, og de mål kaldes antigener. Alle mulige dele af vores krop, eller en fremmed mikroorganisme, kan være et antigen for et specifikt antistof, og der kan laves millionvis af forskellige antistoffer.

Røde blodceller

Erythrocytter, som er navnet for de røde blodceller, udgør omkring 45% af blodets komponenter. Det er dem der giver blodet dets røde farve og de er ansvarlige for at transportere ilt rundt omkring i blodet. Uden ilt kan kroppen ikke overleve, da ilt er nødvendigt for at danne energi i kroppen. Kroppen kan kun producere energi, især til hjernen, i meget kort tid uden ilt. Erythrocytter har ikke nogen cellekerne og en sjov form som en disk der buer indad i midten. Erythrocytter bærer proteiner på deres overflade som andre blodkomponenter kan interagere med. Men de proteiner er forskellige i forskellige mennesker og er bestemt af vores DNA og hvad vi har nedarvet fra vores forældre. Disse proteiner på overfladen af erythrocytter kan blive angrebet af vores leukocytter hvis de ses som "fremmede" af de leukocytter. Men på den modsatte side har vores leukocytter tolerance for vores egne erythrocytters overfladeproteiner, så vores egne erythrocytter ikke bliver angrebet af vores egne leukocytter. Når erythrocytternes overfladeproteiner bliver angrebet af leukocytter sker det ved at leukocytterne laver antistoffer der angriber overfladeproteinerne. Dermed fungerer disse overfladeproteiner som antigener for antistofferne. Et menneske laver aldrig antistoffer mod deres egne erythrocytters antigener, men laver tilgængæld antistoffer mod alle andre slags erythrocyt antigener. Det giver ophav til det vi kalder blodtyper. Disse antigener kaldes meget simpelt A og B, og man kan have den ene, den anden, begge, eller ingen af dem. Hvis man har både antigen A og B er ens blodtype AB. Hvis man kun har antigen A har man blodtype A, og samme for antigen B og blodtype B. Hvis man hverken har antigen A eller B er ens blodtype 0. Det er kombinationen af ens gener der bestemmer ens blodtype. Man nedarver netop en kopi fra ens far og en kopi fra ens mor. Det at man har to versioner af et gen hedder at man har to "alleler". En allel kan være "dominant" eller "recessiv". En dominant allel vil altid komme til udtryk, mens en recessiv allel kun kommer til udtryk hvis der ikke er nogen dominante alleler. A og B er begge dominante alleler, mens 0 er recessiv. Man kan derfor kun have blodtype 0 hvis begge ens alleler er 0. Når man arver blodtype gener fra ens forældre vil det kun være ét af deres to alleler, og det er tilfældigt hvilket det er.

6.5 Sygdomslære

Nu har vi kigget på, hvordan hjertet og blodet ser ud, når det hele fungerer. En stor del af medicin og lægevidenskab handler til gengæld om den syge krop, og hvordan man

behandler sygdomme. Så hvad kan gå galt? Vi kigger lidt nærmere på, hvad der sker med hjertet og kredsløbet ved en af de store folkesygdomme: Hypertension.

Hypertension

Hvad er hypertension?

Hos nogle mennesker fungerer den fine regulering af blodtrykket ikke helt som den skal. Det kan føre til at man får for højt blodtryk. For højt blodtryk hedder hypertension på lægesprog. Hyper- betyder for meget og -tension betyder tryk, altså for meget tryk. Når man har hypertension, arbejder hele kredsløbet lidt hårdere end det egentlig har lyst til.

Et normalt blodtryk defineres som systolisk tryk under 135 og et diastolisk tryk under 85. Grænserne er dog fleksible, og det afhænger af, hvilke sygdomme man ellers har. I udgangspunktet snakker man dog om en øvre grænse på 135/85. Hvis man har endnu mere tryk på systemet kan man have meget højt blodtryk. Meget højt blodtryk er et blodtryk på 175/105 eller derover.

Mekanismer og årsager bag hypertension

Når man ved, hvilke faktorer der spiller ind på blodtrykket, kan man også finde ud af, hvor i ligningen reguleringen slår fejl. Fejlen kan ligge i hjertets slagvolumen (som afhænger af hjertets rytme og det samlede blodvolumen) eller den totale perifere modstand. Lad os kigge på de forskellige dele af blodtryksreguleringen.

Hjerterytme

Hjertets rytme kan være hurtig eller langsom. Jo hurtigere den er, des højere vil blodtrykket være. Hjertets rytme afhænger i høj grad af, hvad vi laver. Hvis kroppen er i fysisk aktivitet, vil pulsen stige, og dermed vil blodtrykket også stige. Når man er i hvile, vil hjertet slå langsommere og blodtrykket falder. Det er et normalt respons fra kroppen. Hvis man i sin hverdag dyrker meget motion træner man sine muskler. Jo mere man øver sig i at løfte tunge ting, des tungere ting vil man kunne løfte på lang sigt. Det samme gør sig gældende for hjertet, da hjertet også er en muskel. Når man er fysisk aktiv træner man dermed også sit hjerte. Når man træner sit hjerte, vil muskelcellerne i hjertet blive bedre til at kontrahere, og selve kontraktionen af hjertet bliver derfor kraftigere. Hjertet bliver bedre til at pumpe, og kan nu uddrive mere blod pr. slag det laver. Vi skal stadig bruge samme mængde blod i kroppen. Derfor kræver det altså færre hjerteslag at pumpe samme mængde blod, da hvert slag bevæger mere blod. Noget andet der sker i kroppen, når man træner, er at man bliver bedre til at udnytte den ilt og den næring, som blodet bringer rundt i kroppen. Når man øver sig på noget og gør det ofte, bliver man bedre til at udføre handlingen. Når man dyrker motion bliver kroppen altså bedre til at trække ilt ud af lungerne. Cellerne bliver bedre til at komme af med deres affaldsstoffer. Kroppen danner endda nye kapillærer, så der kan komme mere blod ud til de muskler som bliver brugt meget. Alt det her betyder, at kroppen bliver meget bedre til at bruge blodet effektivt. Den samme mængde blod kan derfor give mere energi, hvis man er meget fysisk aktiv. Derfor kan hjertet slå lidt langsommere, når man er i god form, fordi kroppen bliver bedre til at udnytte blodet.

Så kroppen har mange måder, hvorpå den kan gøre sig selv bedre til både at pumpe blodet rundt i kroppen og at udnytte det blod, der bliver pumpet rundt. Derfor har man en lavere puls, når man er regelmæssigt fysisk aktiv. Og dermed vil ens blodtryk være lavere, når man dyrker motion i sin hverdag. Hvis man så ikke dyrker motion, vil ens puls altså være højere, og man har større risiko for, at ens blodtryk også er højere.

Blodvolumen

En anden faktor der påvirker blodtrykket, er mængde af blod i kredsløbet. Jo mere blod der er, des større tryk kommer der på systemet, og dermed stiger blodtrykket.

Kroppen regulerer selv hvor meget blod der er i karbanen, og blodvolumen afhænger af flere forskellige faktorer. Blod består som tidligere beskrevet af mange komponenter, som alle skal være i balance, når det hele fungerer som det skal. Det betyder, at hvis der er flere celler, proteiner eller elektrolytter, skal der også være mere væske. Vi kan ikke selv kontrollere, hvor mange celler der er i blodet. Dette kunne være man var forkølet, hvilket vil give flere hvide blodlegemer i blodet. Vi kan heller ikke kontrollere hvor mange proteiner der er i blodet. Proteiner dannes mange steder i kroppen. Jo flere proteiner i blodet, des mere væske skal der også være. Dog kan proteiner have forskellig virkning på kroppen. Både bugspytkirtel, nyrer og hjerne kan producere proteiner som er signalstoffer, der sætter processer i kroppen i gang, som respons på ting der sker omkring os. Hvis der er et for lille blodvolumen, som kommer gennem nyrerne, vil de sende signal ud om, at der skal ske en stigning i blodtrykket. Dette sker bl.a. ved, at man bliver mere tørstig og at man skal tisse mindre. På den måde bliver mere væske holdt tilbage i kroppen, sådan at man kan få blodtrykket til at stige. Omvendt, hvis der kommer for meget blod gennem nyrerne vil kroppen få signal om, at der skal udskilles mere væske. Dette sker ved, at man får besked af hjernen om at tisse noget mere. Den del af blodet, som vi i højere grad selv kan styre, er elektrolytterne, særligt natrium er vigtig i blodtrykskontrol. En stor del af det natrium vi har i kroppen, kommer fra den salt vi spiser. Jo mere salt man indtager, des mere natrium i blodet. Når der kommer mere natrium i blodet vil der nødvendigvis også komme mere væske. Ens blodvolumen bliver altså større, når man spiser meget salt. Ens blodvolumen bliver også større, når man drikker meget vand. Vand bliver fordelt på en særlig måde i kroppen, sådan at der er en bestemt andel af vandet i blodbanen. Hvis man drikker meget vand, kommer der derfor mere vand ud i blodkarrene, og derfor vil blodtrykket stige, hvis man drikker meget vand.

Total perifer modstand

Den sidste faktor som påvirker blodtrykket, er den totale perifere modstand. Normalt er blodkar meget elastiske, og kan dermed dilateres og kontraheres i den grad, som kroppen har brug for, for at holde blodtrykket under kontrol. Det kan dog ske, at blodkarrene bliver stiverer, så deres elasticitet falder. De kan derfor ikke dilatere og kontrahere som de skal. Det er især et problem ift. dilatation, da en øget diameter på blodkarrene kan bidrage til at sænke blodtrykket. Årsagen til at blodkars elasticitet falder er hyppigst fordi der bliver aflejret fedt i karvæggen. Det kaldes arterosklerose. Artero- betyder kar og -sklerose betyder at noget bliver hårdere i strukturen. Det der sker, er derfor at karrene bliver mere faste i strukturen, og dermed kan de ikke udvide sig lige så meget, som de plejer. Mængden af arterosklerose afhænger af flere faktorer, men i høj grad af hvor meget fedt man spiser. Jo mere fedt man spiser, des hårdere skal kroppen arbejde, for at opbevare det de rigtige steder. Kan kroppen ikke følge med, kan fedtet blive lagt i forkerte steder i kroppen. Blodkarrene er et af de steder, der ikke er ment til at opbevare fedt. Derfor bliver blodkarrenes funktion dårligere, når der kommer fedt ind i deres væg. Når blodkarrene ikke kan dilatere som de skal, får man derfor større risiko for at få hypertension.

Opsamling på årsager til hypertension

Selvom der er mange ting man selv kan gøre, for at ændre på sit blodtryk, er det faktisk sjældent man kan udpege én bestemt årsag til ens hypertension. Hos 90-95% kan man ikke sige med sikkerhed, hvorfor de har hypertension. Dog ved man at hypertension hos langt de fleste skyldes en blanding af arv og livsstil.

Er hypertension farligt?

Det er ikke sundt for kroppen at have højt blodtryk i lang tid. Når trykket er højt hele tiden, er der stort pres på kredsløbet, og der kan ske skader på forskellige organer.

Først og fremmest skal hjertet arbejde meget hårdere for at pumpe blodet rundt i kroppen. Som tidligere nævnt bliver hjertet større og stærkere, når man er fysisk aktiv, og dermed træner sit hjerte. Men forskellen på at have højt blodtryk, når man dyrker motion og at have hypertension er, at hjertet får lov at hvile ved et lavere blodtryk, når man ikke længere er fysisk aktiv. Når man har hypertension, er trykket er højt hele tiden, og hjertet har ikke tid til at hvile. Hvileperioden er vigtig, for det er her hjertet bygges op systematisk til at være stærkere og at kunne pumpe kraftigere. Når hjertet ikke kan hvile og få lov at opbygge mere muskel på en ordentlig måde, bliver musklerne bare opbygget lidt tilfældigt. Hjertet har brug for mere styrke til at pumpe det høje blodtryk, men fordi der ikke er tid til at opbygge muskler under hvile, bliver muskelfibrene lavet lidt hulter til bulter. Normalt er muskelfibrene i hjertet lagt i samme retning, sådan at de alle kan samarbejde om at lave en god kontraktion. Ved hypertension ser man at hjertemuskulaturen bliver opbygget i alle mulige forskellige retninger. Så selvom hjertet rent fysisk bliver større, bliver dets funktion ikke bedre. Faktisk bliver funktionen dårligere. Ved hypertension kan man altså ende med et hjerte, der er for stort og som virker dårligere.

Udover hjertet kan mange andre organer også tage skade. Det er især organer med mange kapillærer. Da kapillærer er de mindste blodkar vi har i kroppen kan de ikke håndtere et stort tryk. Normalt er trykket meget lavt i kapillærerne, men når man har hypertension, kan trykket i kapillærerne ende med at blive meget højere end normalt. Det gør at kapillærerne kan gå i stykker, og at man kan bløde i de organer med mange kapillærer. Dette kunne være i hjernen, øjnene eller nyrerne.

Der kan ske mange ting i kroppen, hvis man har hypertension, og det kan blive rigtigt farligt. Derfor er det vigtigt at opdage og behandle hypertension, så man kan undgå at få skade på sin krop.

Behandling af hypertension

Lad os kigge på, hvordan man behandler hypertension. Der er flere måder, man kan gøre det på. Vigtigst er det at vide, at behandling af de fleste sygdomme er baseret på et trappeprincip. Det betyder at man starter med behandling, som er nem at bruge og med få ting der kan gå galt, men som måske ikke virker så godt. Hvis det ikke kan gøre folk raske, går man et trin op af trappen og man lægger mere behandling oveni. Den behandling man lægger oveni, kan være lidt sværere at bruge, have lidt flere ting der kan gå galt, men som måske også virker bedre. Når man behandler sygdomme, er det vigtigt at finde balancen med, hvor store risici man skal tage ift. hvor god en virkning kan være.

Hvem skal behandles?

Nogle mennesker har flere risikofaktorer, som gør, at de skal have mere behandling end andre. Derfor skal lægen sammen med patienten finde ud af, hvilken behandling patienten skal have. Der er mange forskellige ting man skal tænke over, når man skal behandle en patient med hypertension. Her er nogle af de ting, som kan give en større risiko for at få skader på kroppen, når man har hypertension:

- Man er en mand
- Man har en høj alder (55+ for mænd og 65+ for kvinder)
- Man har et familiemedlem som har hypertension
- Man ryger
- Man er overvægtig

Dem med flere risikofaktorer skal ofte behandles mere end dem med få risikofaktorer. Hvor meget man skal behandles med, afhænger også af, hvor højt blodtrykket er. Figur 6.2 nedenfor viser, hvordan hvilken behandling man skal give.

	Grad af blodtryksforhøjelse (mmHg)			
Risikofaktorer Asymptomatisk organskade eller sygdom	Høj normal SBT 125-134 eller DBT 80-84	Grad 1 HT SBT 135-154 eller DBT 85-94	Grad 2 HT SBT 155-174 eller DBT 95-104	Grad 3 HT SBT ≥175 eller DBT ≥105
Ingen	• Ingen BT intervention	• Livsstilsændringer • Evt. tillæg BT-medicin efter 6-12 mdr med målet <135/85	• Livsstilsændringer • Senere tillæg BT-medicin med målet <135/85	• Livsstilsændringer • Omgående BT-medicin med målet <135/85
1-2 risikofaktorer	• Livsstilsændringer • Ingen BT-medicin	• Livsstilsændringer. • Senere tillæg BT-medicin efter 3-6 mdr med målet <135/85	• Livsstilsændringer • Senere tillæg BT-medicin med målet <135/85	• Livsstilsændringer • Omgående BT-medicin med målet <135/85
≥3 risikofaktorer	• Livsstilsændringer • Ingen BT-medicin	• Livsstilsændringer • Senere tillæg BT-medicin med målet <135/85	• Livsstilsændringer • BT-medicin med målet <130/80 ***	• Livsstilsændringer • Omgående BT-medicin med målet <130/80 ***

Figur 6.2: Behandlingsalgoritme til hypertension. HT: Hypertension. BT: Blodtryk. SBT: Systolisk blodtryk. DBT: Diastolisk blodtryk.

Livsstilsændringer

Det første man gør, når man vil behandle hypertension, er at kigge på livsstil. Der er flere af risikofaktorerne, som man kan fjerne ved at få en sundere livsstil. Man kan stoppe med at ryge, ændre på det man spiser og drikker (mindre fedt, salt og alkohol), og man kan dyrke mere motion. At ændre på livsstil er ikke farligt, og der er ikke så mange ting, som kan gå galt. Derfor er livsstilsændringer det første trin på behandlingstrappen. Selvom livsstil er det første trin på trappen, er det ikke altid let at ændre på sin livsstil. Derfor er det vigtigt at den plan man lægger som læge er i samarbejde med patienten, så man er sikker på, at patienten får alt den hjælp de behøver.

At ændre på livsstil kan være nok for nogle, mens andre skal have medicin for at kontrollere deres hypertension.

Medicin

Der er mange forskellige slags medicin man kan give patienter med hypertension. De modarbejder alle de forskellige mekanismer, som giver hypertension.

Man kan give medicin som gør, at hjertet slår langsommere og mindre kraftigt. Man kan også give medicin, som gør at man tisser mere, hvilket gør at blodvolumen bliver mindre. Til sidst kan man give medicin som forhindrer at der kommer mere fedt ind i karvæggen, hvilket gør, at blodkarrene ikke bliver endnu stivere end de allerede er. Som regel starter man med en slags medicin. Hvis en slags medicin ikke er nok til at sænke blodtrykket kan man lægge flere oveni.

●●● **Opgave 6.5.1:**
Hvor mange risikofaktorer har patienten? Hvilken grad af hypertension har patienten? Og hvordan vil du behandle patienten?

1) Karl er en mand på 67 med et blodtryk på 150/82. Han løber 2 gange om ugen og drikker en flaske vin hver fredag.

2) Johan er en mand på 75 med et blodtryk på 146/90. Han ryger en pakke cigaretter og spiser en pose chips om dagen. Han kan godt lide at ligge på sofaen.

3) Hanne er en kvinde på 50 med et blodtryk på 178/110. Hun drikker ikke alkohol, ryger ikke og går i fitnesscenter 4 gange om ugen.

For mere information om hypertension refererer vi til [2] og [3].

Bibliografi

[1] Lars Pedersen. *Matematik 112. Førstehjælp til formler*. dansk. 3. udg. PRAXIS - Nyt Teknisk Forlag, 2014. ISBN: 978-87-571-2662-4.

[2] Mayo Clinic Staff. *High blood pressure dangers: Hypertension's effects on your body*. 2022. URL: https://www.mayoclinic.org/diseases-conditions/high-blood-pressure/in-depth/high-blood-pressure/art-20045868.

[3] Annemie Stege Bojer m.fl. *Arteriel Hypertension*. 2022. URL: https://nbv.cardio.dk/hypertension.

Velkommen til Sukkertoppen Gymnasium

Sukkertoppen Gymnasium er et stort, veldrevet gymnasium med fokus på naturvidenskab, teknologi og design.
Vores nøglebegreber er høj faglighed, stor Rummelighed og et aktivt og alsidigt studiemiljø med en levende kultur både fagligt og socialt.

Du arbejder med både teori og praksis og får selvfølgelig mulighed for at slippe din indre forskerspire eller opfinder løs i vores topmoderne laboratorier og værksteder.

Hvad enten du drømmer om at deltage i kemi-olympiade eller arrangere weekendlange LAN-parties, er Sukkertoppen stedet for dig.

På Sukkertoppen Gymnasium bliver du en del af et stærkt fagligt miljø, hvor du som elev udfordres til at blive så dygtig som muligt. Det kommer blandt andet til udtryk, når vi hvert år har elever med til DM i teknologi, DM i science, biologi OL, kemi OL, datalogi OL, Georg Mohr og meget mere.

Vores 1100 elever er ambitiøse, målrettede og trives i et godt studiemiljø, hvor der er klare retningslinjer og et godt sammenhold. Vi har et aktivt elevråd og en god dialog mellem eleverne og ledelsen. Det er med til at skabe den gode stemning, du vil opleve på Sukkertoppen.

Det sociale liv er en vigtig del af gymnasiet.
På Sukkertoppen har vi et godt socialt miljø med fester, brætspilscafeer, filmklub, idrætsdage, musikarrangementer, fredagscafeer og meget andet.

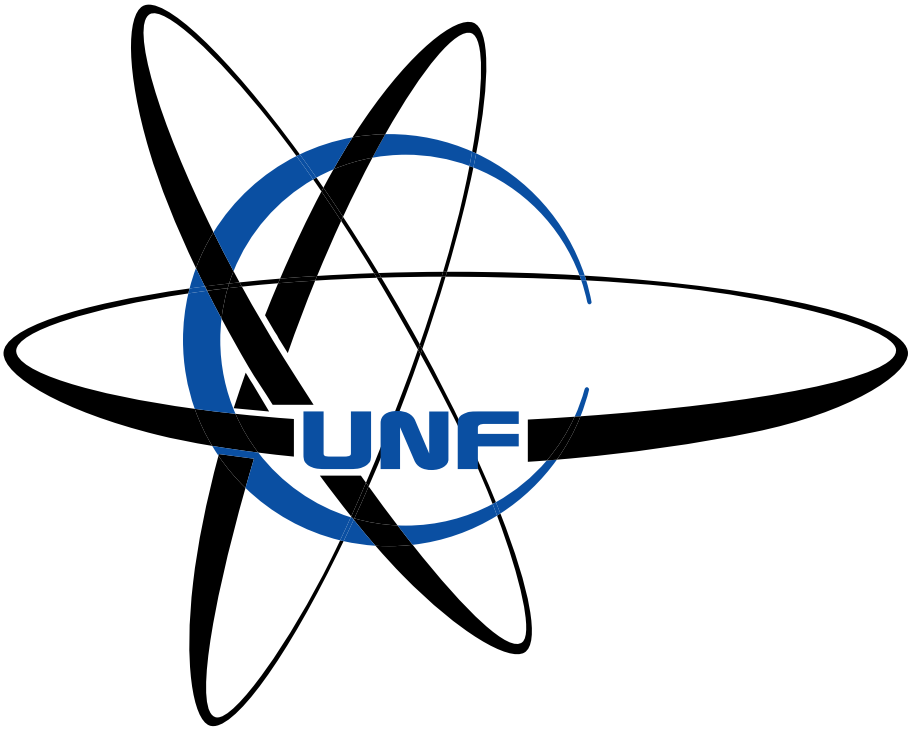
Sukkertoppen Gymnasium er stolt af at huse UNF Naturfagsweekend 2023

Dansk Ungdoms Fællesråd - DUF

DUFs lokalforeningspulje støtter de lokale foreningers arbejde for børn og unge i Danmark og Sydslesvig. Puljen kan søges af lokalforeninger og lokale grupper i DUFs medlemsorganisationer og observatørorganisationer.

DUF

DANSK
UNGDOMS
FÆLLESRÅD



Ungdommens Naturvidenskabelige Forening

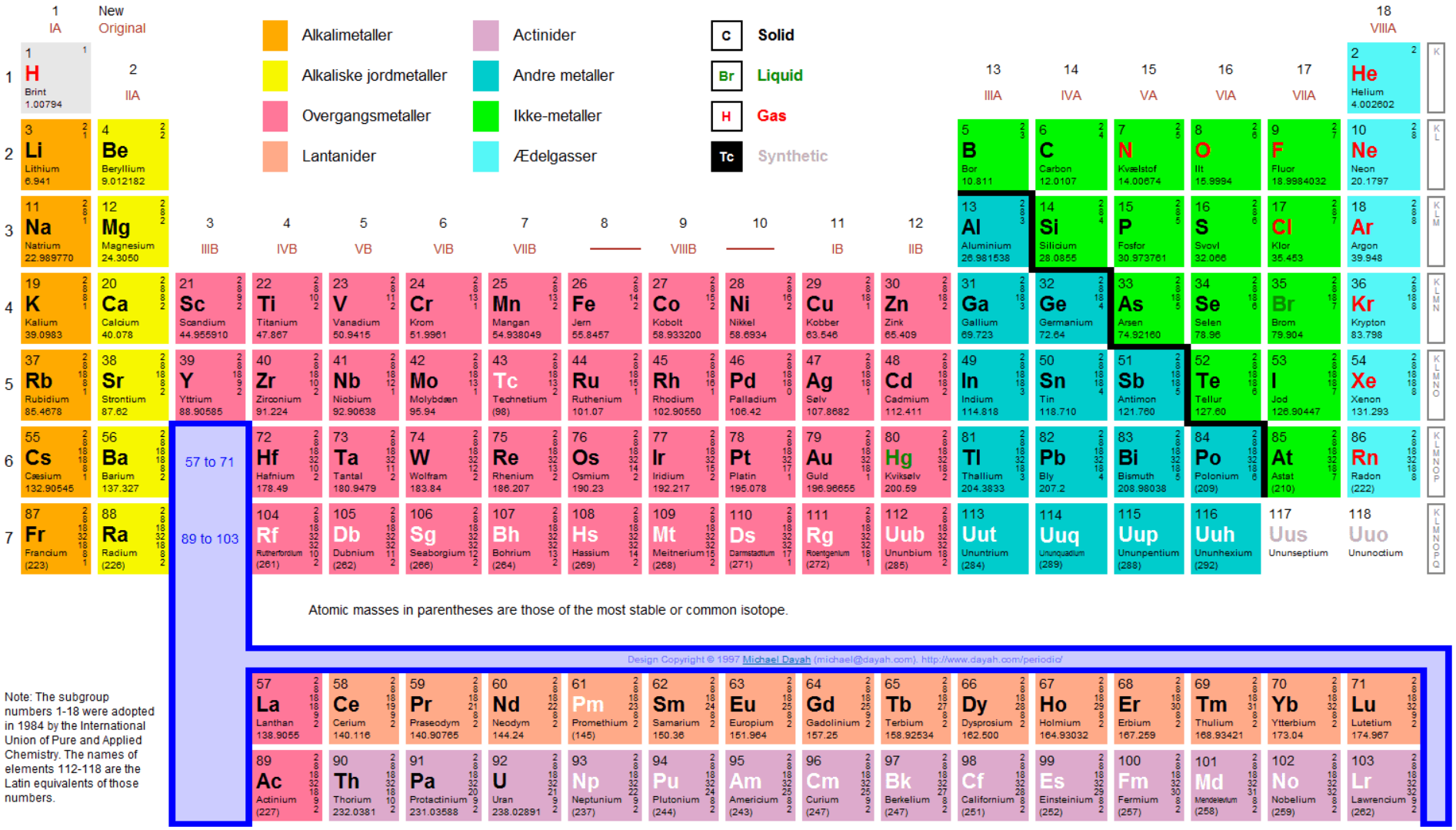
Ungdommens Naturvidenskabelige Forening (UNF) har til formål at fremme kendskabet til og interessen for naturvidenskab og teknologi fortrinsvis blandt unge.
Formålet realiseres ved at arrangere foredrag, studiebesøg, workshops, offentlige events og ikke mindst ScienceCamps! Målgruppen er især elever ved ungdomsuddannelserne, men i de seneste år er der også kommet fokus på folkeskolens ældste klasser. Mens især ScienceCamps altid har en veldefineret målgruppe, er de øvrige aktiviteter normalt åbne for alle.

På landsplan har vi over 15.000 deltagere, der hvert år har glæde af mere end 170 spændende arrangementer.
Det hele organiseres af over 300 frivillige unge, primært universitetsstuderende og gymnasieelever, som brænder for at dele deres glæde ved naturvidenskaben. De frivillige i foreningen planlægger og afholder aktiviteterne, og de, der har lyst, er også med til selv at undervise og formidle naturvidenskaben. Det frivillige arbejde i UNF er kompetencegivende i flere sammenhænge, og samtidigt giver det et godt netværk på tværs af hele landet.

Foreningens rødder strækker sig helt tilbage til H. C. Ørsted, som i 1824 stiftede SNU - Selskabet for Naturlærens Udbredelse. UNF så dagens lys i 1944, da en gruppe gymnasielærere fra SNU ønskede at fokusere udelukkende på unge. Stiftelsen af landsforeningen, UNF Danmark, i 2002 samlede de fire lokalforeninger i Aalborg, København, Odense og Aarhsu og muliggjorde blandt andet afholdelsen af ScienceCamps og andre landsdækkende arrangementer.

I april 2005 blev Hendes Kongelige Højhed Kronprinsesse Mary protektor for UNF. Vi er meget beærede og taknemmelige for den anerkendelse, der ligger i vores protektorat. Det er UNF's håb, at det kongelige protektorat kan hjælpe os i vores virke for at fremme de naturvidenskabelige og tekniske discipliner i Danmark.

Grundstoffernes Periodiske System



Note: The subgroup numbers 1-18 were adopted in 1984 by the International Union of Pure and Applied Chemistry. The names of elements 112-118 are the Latin equivalents of those numbers.