

DNA-Analyse

I dette dokumentet skal vi se litt nærmere på DNA, og hvordan vi kan bruke programering som et verktøy til å analysere og jobbe med DNA-sekvenser. Vi kommer bare til å jobbe med mindre sekvenser som eksempler, men dette er gode eksempler på et viktig emne innenfor bioinformatikk, hvor man gjør analyser av hele det mennesklige *genomet*.

Notebooken er delt i to hovedbiter. I den første biten går vi igjennom og repeterer korte biter av hvordan DNA fungerer og er bygget opp. I tillegg skriver vi korte kodesnutter som utfører små analyseoppgaver av DNA-sekvenser.

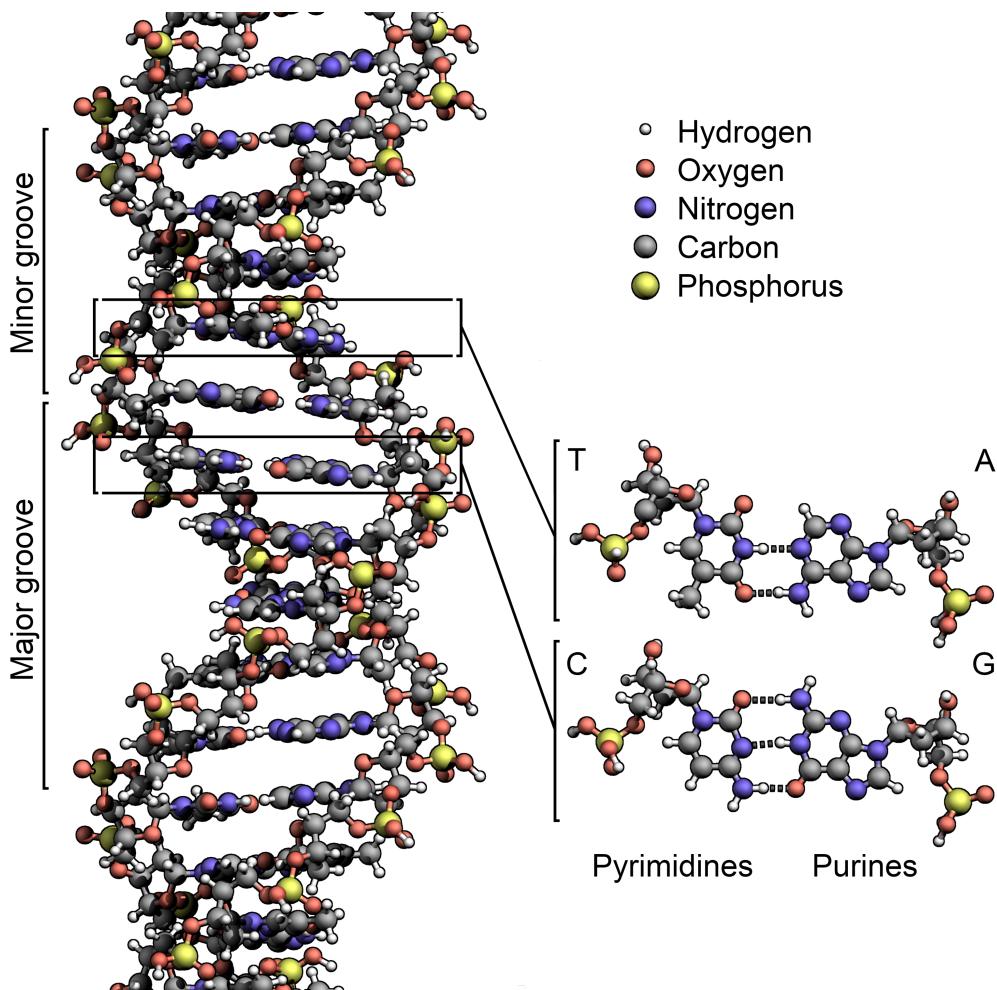
I andre halvdel av denne notebooken går vi igjennom et bestemt case, som er basert på en forskningsartikkel fra 2008. I dette cases skal vi bruke et par av verktøyene vi har laget for å se på mutasjoner i genet for proteinet *insulin*, som kan føre til diabetes.

Intro: Kort repetisjon om DNA

Deoksyribonukleinsyre (DNA) er et makromolekyl som utgjør arvematerialet til levende organismer. Molekylet er bygget opp som en lang dobbeltrådet spiralstruktur, som vist i bildet under. Mellom de to trådene ligger det *nukleotider*, DNA-ets byggestener. Disse nukleotidene ligger i jevn avstand mellom de to trådene, som trinnene i en stige. Hver nukleotid er bygget opp av ulike baser koblet til en sukker og en fosfatgruppe. Det er fire ulike baser som kan inngå i et nukleotid:

- Adenin (A)
- Tyamin (T)
- Guanin (G)
- Cytosin (C) basepar i jevn avstand, som trinnene i en stige. Det er disse baseparene som er selve informasjonen lagret i DNA-molekylet.

Disse forekommer alltid i komplementære par av A-T og G-C. Om det for eksempel er en adenin base langs den ene tråden, så vil det være en parvis tyamin på nabotråden. De to trådene i DNA inneholder altså eksakte kopier av den samme informasjonen. Dette er viktig for DNA-maskineret som vedlikeholder og kopierer DNA.



Som nevnt er informasjonen i DNA lagret i rekkefølgen av nukleotider, eller baser, og man kan derfor se på DNA som en *sekvens* av bokstavene A/T/G/C. Man trenger kun å spesifisere basene langs en av DNA-ets tråder, ettersom at de to er kopier av hverandre. Et eksempel på en DNA-sekvens vil derfor være som følger

In []:

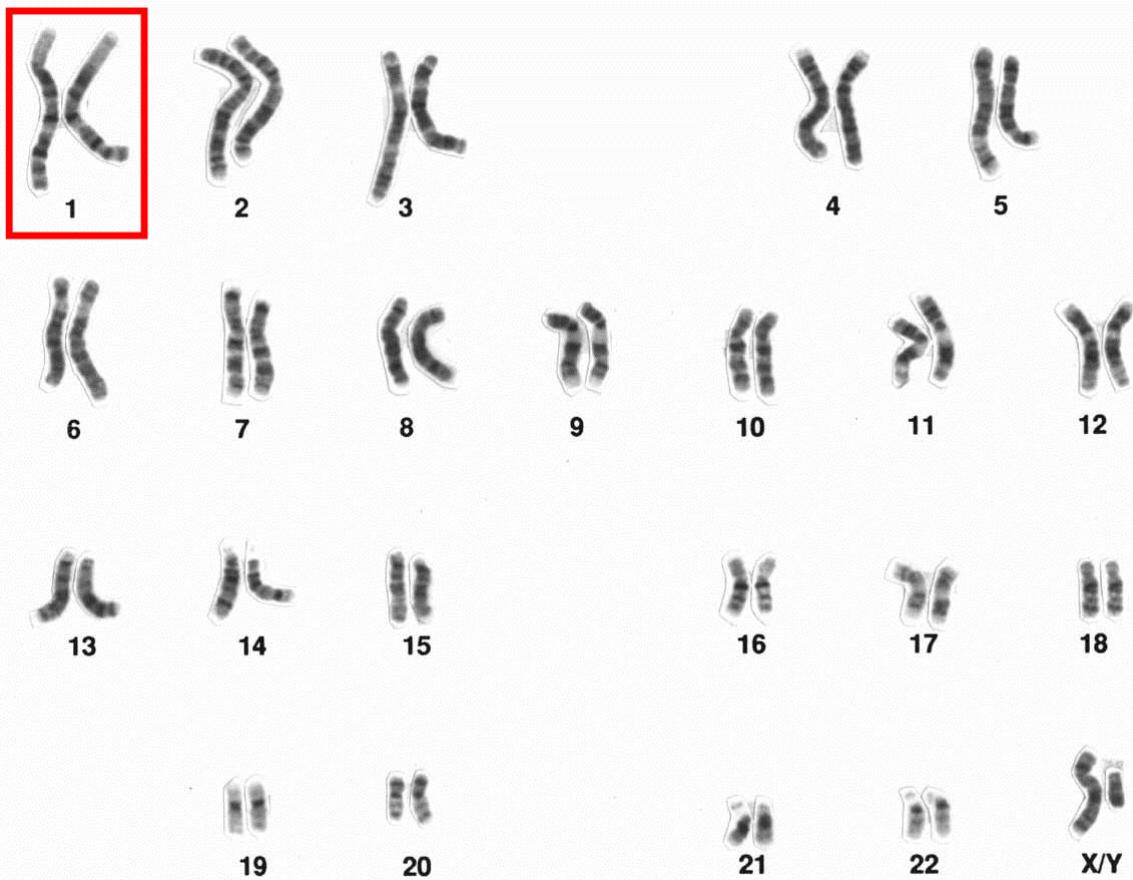
```
1 sekvens = "ACTGATTGACCGCGAGTAG"
```

Denne informasjonen har vi skrevet inn som en *tekststreng* i Python. Når vi gjør det kan vi bruke programmering til å manipulere sekvensen. Dette skal vi se på snart.

Det mennesklige genomet

Mennesker har stort sett det samme arvematerialer, med kun små forskjeller. Studier viser at omtrent 0.1% av det totale DNA-et skiller seg fra person til person. Stort sett deler vi altså det samme arvematerialet, og som helhet kaller vi dette for det *mennesklige genomet*.

Menneskets arvemateriale er fordelt over 23 ulike kromosomer. Vi er også diploide, som betyr at vi i hver celle har to versjoner av hvert kromosom, som godt kan være litt forskjellige.



Menneskets 23 kromosompar sett igjennom et mikroskop. Kilde: [Wikimedia Commons](https://en.wikipedia.org/wiki/Chromosome_1#/media/File:Human_male_karyotype_high_resolution_Chromosome_1.png) (https://en.wikipedia.org/wiki/Chromosome_1#/media/File:Human_male_karyotype_high_resolution_Chromosome_1.png) (Public Domain)

Det mennesklige genomet består av litt i overkant av 3.2 milliarder basepar, så siden vi er diploide har vi hver celle altså omrent 6.4 milliarder basepar.

La oss gjøre en kjapp beregning av hvor mye informasjon som er lagret i det mennesklige genom. Ettersom at kromosonparene inneholder omrent den samme informasjonen trenger vi bare å ta hensyn til 3.2 milliarder basepar. Men hvordan regner man på "hvor mye informasjon" noe inneholder? Sett på en annen måte er dette det samme som å spørre hvor mye lagringsplass man trenger for å lagre hele det mennesklige genomet på en datamaskin.

På en datamaskin lagres all informasjon i totalsystemet, altså lagres all informasjon som en lang rekke med 0 og 1. Hvert siffer kalles 1 *bit*. Hvor mange bits trenger vi for å lagre en DNA-sekvens? For hver base er det fire muligheter: A, T, C og G. Hvor mange bits trenger vi for å entydig representere disse? Fra kombinatorikk vet vi at dette blir 2 bits, fordi $2^2 = 4$. For eksempel kan vi si at vi bruker følgende "oversettelse":

- A - 00
- T - 01
- G - 10
- C - 11

Altså vil hvert basepar trenge 2 bits. Ettersom at det mennesklige genomet består av 3.2 milliarder basepar har vi da Vi trenger altså

$$3.2 \cdot 10^9 \text{ basepar} \cdot 2 \frac{\text{bits}}{\text{basepar}} = 6.4 \cdot 10^9 \text{ bits.}$$

Men hvor mye er egentlig 6.4 milliarder bits? De fleste har nok et bedre forhold til *bytes*, og en byte er 8 bits. Så da får vi at dette svarer til

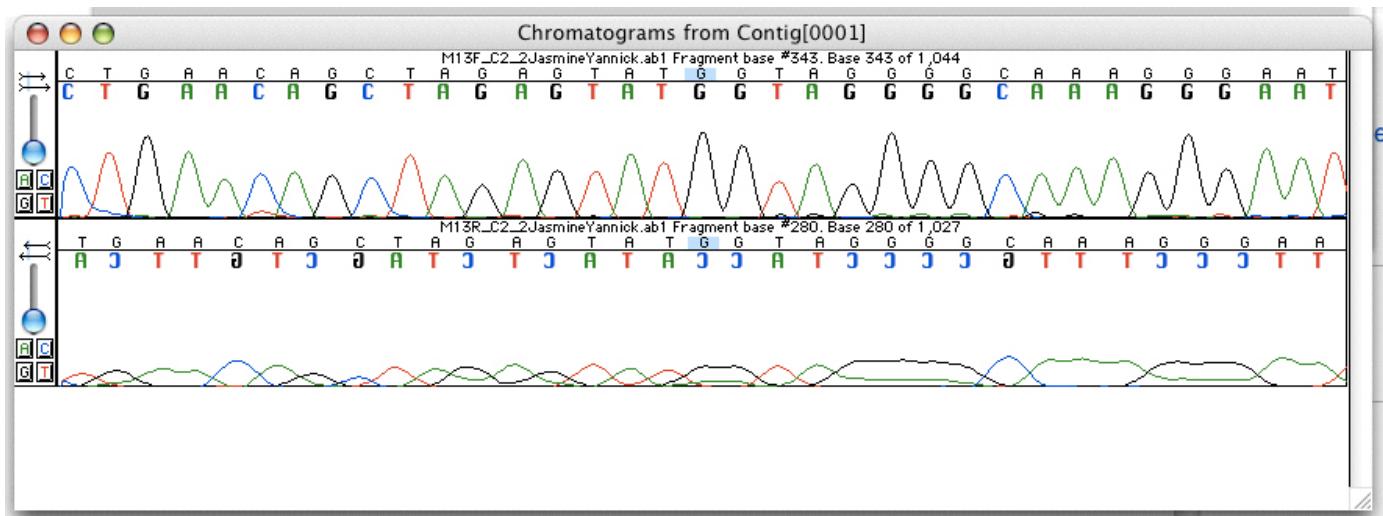
$$\frac{6.4 \cdot 10^9 \text{ bits}}{8 \frac{\text{bits}}{\text{byte}}} = 0.8 \cdot 10^9 \text{ bytes} = 0.8 \text{ gigabyte.}$$

Altså er hele det mennesklige genomet omtrent 0.8 gigabyte. Dette er ganske mye, men samtidig overraskende lite, det mindre enn en liten stor app eller en film lagret i HD-kvalitet.

DNA-Sekvensiering

Om vi ønsker å utforske informasjonen som er lagret i arvemateriale må vi på en eller annen måte kunne lese ut sekvensen av basepar i DNA-et. Denne prosessen kalles *sekvensiering*. De første DNA-sekvensene ble funnet på 1970-tallet, men stor-skala DNA-sekvensiering er kun blitt nylig i moderne tid. I 1990 startet [Human Genome Project](https://en.wikipedia.org/wiki/Human_Genome_Project) (https://en.wikipedia.org/wiki/Human_Genome_Project), et storskala vitenskapelig prosjekt for å kartlegge hele det mennesklige genomet. Prosjektet fullførte oppgaven sin i 2003.

Vi skal ikke si så mye om hvordan sekvensiering er teknologisk mulig, det er et interessant tema, men det får vi ta en annen gang. For nå holder vi oss til å påpeke at rask sekvensiering av DNA er et teknologisk gjennombrudd som har ført til store nye datamengder og et nytt felt innen biologien: bioinformatikken. Dette feltet handler om å kombinere eksperimentelle metoder og teknologi for å hente inn store mengder informasjon, blant annet fra DNA, og å bruke verktøy fra matematikk og informatikk for å analysere og tolke denne informasjonen.



Eksempel på eksperimentell sekvensiering av DNA. Kilde: [Wikimedia Commons](https://en.wikipedia.org/wiki/Electropherogram#/media/File:Chromatogram.jpg) (<https://en.wikipedia.org/wiki/Electropherogram#/media/File:Chromatogram.jpg>) (Public Domain).



Moderne sekvensieringsmaskiner kan sekvensiere utrolige mengder DNA på kort tid.[Kilde: Wikimedia Commons](https://en.wikipedia.org/wiki/DNA_sequencer#/media/File:DNA-Sequencers_from_Flickr_57080968.jpg) (https://en.wikipedia.org/wiki/DNA_sequencer#/media/File:DNA-Sequencers_from_Flickr_57080968.jpg) (CC BY 2.0 (<https://creativecommons.org/licenses/by/2.0/>))

Jobbe med DNA-sekvenser i Python

Nå har vi kort snakket litt om DNA, men la oss snu over til å begynne å analysere og manipulere DNA-sekvenser ved hjelp av programmeringsspråket Python. Vi starter med å jobbe med helt oppdiktete sekvenser, men vi skal til slutt se på et eksempel med ekte DNA-sekvenser knyttet til diabetes.

Vi går igjennom et par ulike oppgaver. Om du ønsker å se oppgavene på en annen måte, eller å lære enda mer om bioinformatikk kan du finne flere oppgaver på rosalind.info (<http://rosalind.info/problems/list-view/>).

Telle baser

Den første oppgaven vi skal løse er å teller hvor mange ganger de fire ulike basene forekommer i en gitt DNA-streng. Om vi blir gitt en DNA-streng som en tekststreng i Python, så kan vi løkke over den ved hjelp av en for-løkke:

In []:

```
1 dna_sekvens = "ACTGATTGAAG"
2
3 for base in dna_sekvens:
4     print(base)
```

Inne i løkka kan vi nå gjøre mer spennende ting enn å bare skrive ut basen. Vi lager nå en funksjon som for en vilkårlig DNA sekvens løkker over den og teller de ulike basene:

In []:

```
1 def tell_baser(DNA_sekvens):
2     """Gitt en DNA sekvens, tell antallet av de 4 ulike basene."""
3     A = 0
4     C = 0
5     G = 0
6     T = 0
7
8     for base in DNA_sekvens:
9         if base == 'A':
10             A += 1
11         elif base == 'C':
12             C += 1
13         elif base == 'G':
14             G += 1
15         elif base == 'T':
16             T += 1
17         else:
18             print("Ugyldig base funnet i DNA-sekvens")
19
20     N = len(DNA_sekvens)
21
22     print(f"Baser totalt: {N}")
23     print(f"-----")
24     print(f"Antall (A)denin: {A:2} ({A/N:.1%})")
25     print(f"Antall (C)ytosin: {C:2} ({C/N:.1%})")
26     print(f"Antall (G)uanin: {G:2} ({G/N:.1%})")
27     print(f"Antall (T)ymin: {T:2} ({T/N:.1%})")
28     print(f"-----")
```

In []:

```
1 DNA_sekvens = "GCGAGGTTAATAATTGGCGTGCCTGAAGGTTAAAGCCC"
2 tell_baser(DNA_sekvens)
```

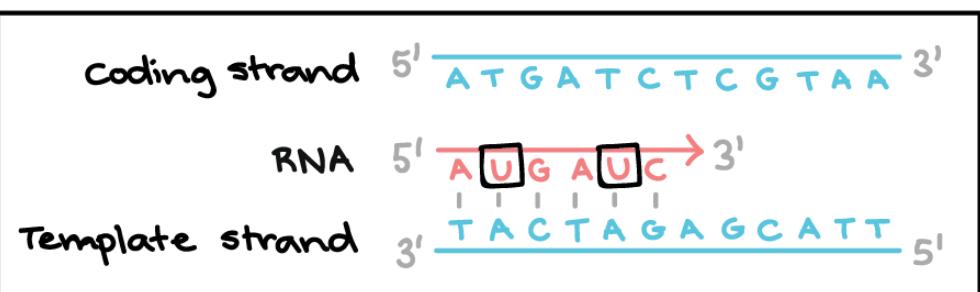
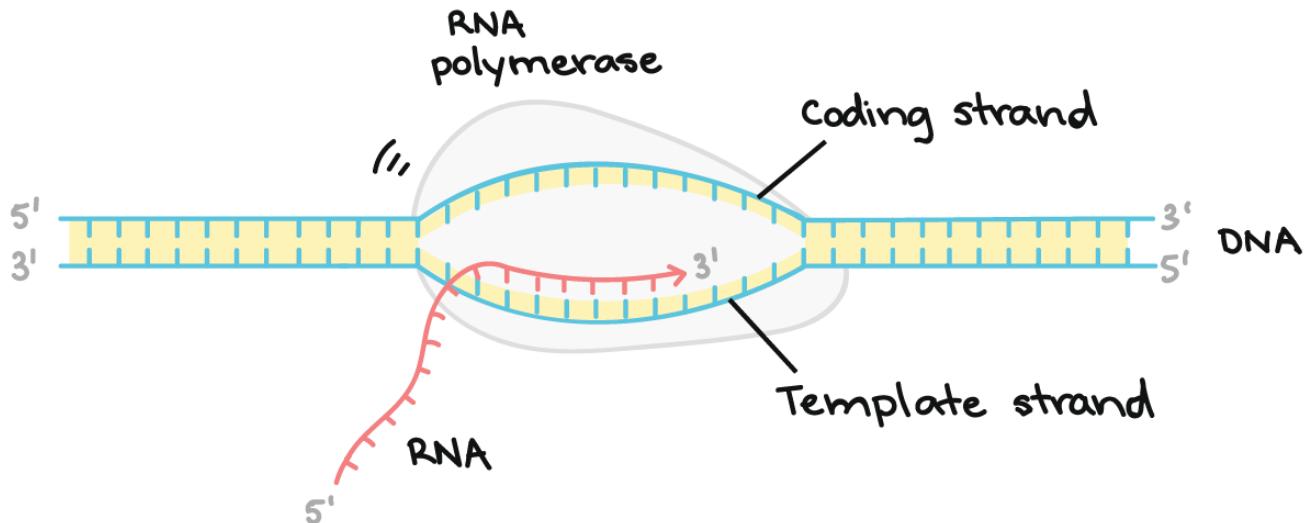
Oppgave 1a) Telle baser

- Kjør koden over
- Diskuter koden med sidemannen og forklar hva de ulike bitene av koden gjør.

Komplementere en DNA-sekvens

DNA er dobbeltrådet, der informasjonen langs den ene tråden er en *komplementær* kopi av informasjonen langs den andre tråden. At det er en *komplementær* kopi betyr at vi ikke finner de eksakt samme basene, men de "motsatte" basene, det vil si at hver A langs en tråd har en T i den andre tråden, og vica versa. Det samme gjelder G og C.

De to trådene i DNA kalles gjerne for kodestrand og malstrand, eller på engelsk: coding strand/template strand. Kodestranden er den som koder for faktiske gener. Når DNA skal kopieres over til mRNA for fraktes ut av cellekjernen er det derimot maltråden som brukes. Situasjonen oppsummeres av bildet under:



Bildet er tatt fra [Khan Academy](https://www.khanacademy.org/science/biology/gene-expression-central-dogma/transcription-of-dna-into-rna/a/overview-of-transcription) (<https://www.khanacademy.org/science/biology/gene-expression-central-dogma/transcription-of-dna-into-rna/a/overview-of-transcription>).

Oppgave 1b) Komplementere en DNA-sekvens

Du skal nå lage en funksjon som gitt en DNA-sekvens, finner den komplementære sekvensen. Men andre ord, returner en ny streng der følgende endringer er gjort:

- A → T
- T → A
- G → C
- C → G

For å få til dette, fyll inn koden under. For hver base må du bruke en if-test (som i eksempelet over) og legge til en ny bokstav i den komplementære strengen, her kan du bruke `+ =` for eksempel.

In []:

```

1 def komplementer_DNA(sekvens):
2     """Gitt en DNA sekvens, finn den komplementær sekvensen"""
3     komplement = ""
4     for base in sekvens:
5         ...
6
7     return komplement

```

Test koden din på strengen under:

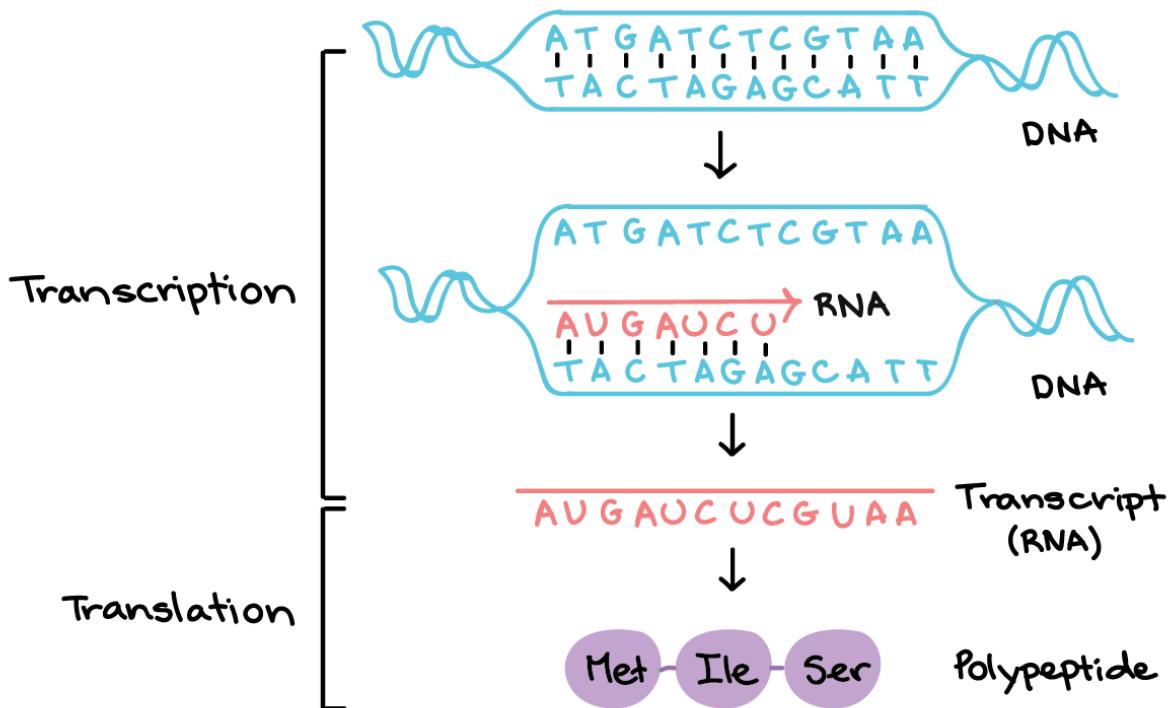
In []:

```
1 kodetråd = "GTTTAATAATTGGCGTGCCTGAAGGTTAAAGCCC"  
2 maltråd = komplementer_DNA(kodetråd)  
3  
4 print(kodetråd)  
5 print(maltråd)
```

Fra DNA til protein: Transkripsjon og Translasjon

Arvematerialer i DNA koder for ulike proteiner. Den delen av DNA som koder for ett gitt protein kalles et *gen*. Vi skal nå se nærmere på prosessen på hvordan vi går fra et gen, altså en DNA-sekvens, til et protein. Denne prosessen deles gjerne i to:

1. Først kopieres den relevante biten av DNA-et over til en bit med mRNA som kan transportes ut av cellekjernen og til ribosomene. Dette kalles gjerne for *transkripsjon*.
2. Deretter oversettes mRNA-sekvensen til et polypeptid ved at RNA-sekvensen oversettes til aminosyrer som settes sammen som perler på en snor. Dette kalles gjerne for *translasjon*.



Bildet er tatt fra Khan Academy (<https://www.khanacademy.org/science/biology/gene-expression-central-dogma/transcription-of-dna-into-rna/a/overview-of-transcription>).

Vi skal nå skrive kode som utfører disse to stegene.

Transkripsjon

Første steg handler om å lage en mRNA-kopi av en gitt DNA-sekvens. mRNA ligner veldig på DNA, men er enkeltrådet, istedenfor dobbeltrådet. I tillegg har den også basen uracil, istedenfor tyamin.

Det er enzymet RNA polymerase som har som oppgave å kopiere over informasjon i DNA til RNA. Den vil gjøre dette fra DNAets maltråd, derav "mal".

Oppgave 1c) Transkripsjon

Lag en funksjon som finner mRNA-sekvensen til et gen, gitt DNA-sekvensen langs maltråden. Hint: Denne funksjonen vil ligne ganske mye på svaret fra (1b).

In []:

```
1 def transkripsjon(DNA_sekvens):
2     """Gitt en DNA sekvens fra en malstrand, finn mRNA-sekvens gjennom transkri-
3     mRNA = ""
4     for base in DNA_sekvens:
5         ...
6     return mRNA
```

Når man har sekvensert DNA er det vanlig å oppgi en DNA-sekvens slik den ser ut langs kodestranden. Forklar hvilke steg du må gjøre om du skal oversette en slik sekvens til mRNA med funksjonene du har skrevet så langt.

Translasjon

Translasjon av mRNA til et polypeptid skjer i ribosomene. Her vil tre og tre basepar i mRNA-et utgjøre et *kodon*, som skoder for en gitt aminosyre. Ett tRNA-molekyl vil gjenkjenne dette kodonet og gjøre jobben med å koble riktig aminosyre på polypeptidet.

Ettersom at det er 4 ulike baser, og 3 baser utgjør et koden, finnes det $4^3 = 64$, ulike kodon. Derimot består proteiner som menneskroppen lager kun av 20 ulike aminosyrer. Det er altså ulike kombinasjoner av basepar som koder for samme aminosyre. Ettersom at det bare er 20 aminosyrer er det standardisert å beskrive hver aminosyre med en gitt bokstav.

Vi ønsker nå å oversette mRNA-sekvensen vår til en sekvens av aminosyrer. Da må vi først finne en kodontabell, som viser oss hva de ulike kodonene svarer til. Det kan du for eksempel slå opp på wikipedia:

- https://en.wikipedia.org/wiki/Genetic_code#RNA_codon_table
(https://en.wikipedia.org/wiki/Genetic_code#RNA_codon_table)

Alternativt har Rosalind.info en som er lettere å kopiere her:

- <http://rosalind.info/glossary/rna-codon-table/> (<http://rosalind.info/glossary/rna-codon-table/>)

Vi har tatt jobben med å kopiere over hele kodontabellen til Pythonkode her:

In []:

```
1 kodontabell = {
2     "UUU": "F", "UUC": "F", "UUA": "L", "UUG": "L",
3     "UCU": "S", "UCC": "S", "UCA": "S", "UCG": "S",
4     "UAU": "Y", "UAC": "Y", "UAA": "Stop", "UAG": "Stop",
5     "UGU": "C", "UGC": "C", "UGA": "Stop", "UGG": "W",
6     "CUU": "L", "CUC": "L", "CUA": "L", "CUG": "L",
7     "CCU": "P", "CCC": "P", "CCA": "P", "CCG": "P",
8     "CAU": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
9     "CGU": "R", "CGC": "R", "CGA": "R", "CGG": "R",
10    "AUU": "I", "AUC": "I", "AUA": "I", "AUG": "M",
11    "ACU": "T", "ACC": "T", "ACA": "T", "ACG": "T",
12    "AAU": "N", "AAC": "N", "AAA": "K", "AAG": "K",
13    "AGU": "S", "AGC": "S", "AGA": "R", "AGG": "R",
14    "GUU": "V", "GUC": "V", "GUA": "V", "GUG": "V",
15    "GCU": "A", "GCC": "A", "GCA": "A", "GCG": "A",
16    "GAU": "D", "GAC": "D", "GAA": "E", "GAG": "E",
17    "GGU": "G", "GGC": "G", "GGA": "G", "GGG": "G", }
```

Dette er et spesielt Python-objekt som kalles en *dictionary* (ordbok). Den fungerer ved at vi har par som hører sammen, slik at vi kan skrive for eksempel: `kodontabell['AAG']`, og da får vi ut aminosyren K.

Oppgave 1d) Translasjon av mRNA

Koden under løkker igjennom mRNA tråden og slår sammen hver tredje base til et koden. Fyll inn koden så vi bygger opp en sekvens av aminosyrer

In []:

```
1 def translasjon(mRNA):
2     """Oversett en mRNA sekvens til en sekvens av aminosyrer"""
3     polypeptid = ""
4
5     for k in range(0, len(mRNA), 3):
6         kodon = mRNA[k:k+3]
7         polypeptid += ...
8
9     return polypeptid
```

Oppgave 1e)

- Forklar hvordan vi nå kan utføre hele prosessen fra en DNA-sekvens oppgitt fra kodestranden, til et polypeptid av aminosyrer
- Test med DNA-sekvensen ATCGGTACGGCATTG i kodecellen under. Dette skal bli polypeptidet: IGTAL .

In []:

```
1 kodetråd = "ATCGGTACGGCATTG"
2 maltråd = ...
3 mRNA = ...
4 polypeptid = ...
5 print(polypeptid)
```

Sammenligne sekvenser og identifisere Mutasjoner

Så langt har vi sett kort på hvordan vi går fra en DNA-sekvens til et polypeptid. La oss nå se hvordan vi kan sammenligne ulike DNA-sekvenser for å finne forskjeller. Slike forskjeller kommer fra mutasjoner, som er punktendringer i DNA-sekvensen.

Finne antall forskjeller

Det først vi ser på er å sammenligne to sekvenser og finne *antall* forskjeller eller mutasjoner. På fagspråket kalles dette for *Hamming-avstanden* mellom to sekvenser. Hamming avstanden er rett og slett antall karakterer som er forskjellig mellom to strenger.

Oppgave 1f) Hamming-avstanden

Gitt to sekvenser, `t` og `s`. Lag en funksjon som returnerer antall karakterer som er ulike. Dette kan altså enten være antall baser som er forskjellig, eller antall aminosyrer. For å løkke over to baser samtidig kan vi bruke funksjonen `zip`:

In []:

```
1 def hamming(t, s):
2     """Finn antall ulikheter mellom to sekvenser"""
3     ...
4
5     for base1, base2 in zip(t, s):
6         if base1 != base2:
7             ...
8
9     return ...
```

Sjekk funksjonen din ved å sammenligne sekvensene: ATCGC og ACTGC for hånd og med kode:

In []:

```
1 antall_ulikheter = hamming('ATCGC', 'ACTGC')
2 print(f"De to strengene har {antall_ulikheter} ulikheter")
```

Oppgave 1g) Identifisere mutasjoner

Om vi sammenligner to sekvenser og finner ut av de er ulike, så kan det ofte være vanskelig å finne nøyaktig hvor der ulike. La oss derfor også skrive en funksjon som gjør dette lettere.

Det finnes mange måter vi kan gjøre dette på, men la oss gå for en enkel løsning. La oss lage en ny streng om innholdet i de to input-strengene er lik, så legger vi inn en - . Om innholder er forskjellig legger vi inn en + . Vi kan da skrive ut denne nye strengen under de to andre, så kan vi enkelt se hvor i strengen de er ulike.

For eksempel kan vi gjøre som følger:

```
t = "ACGTGCACATGAGATCTGATTAGACAGGAGACCAGATATACAGATATATGATACAGATACGATA"  
s = "ACGTGCACATGAGATCTGATTAGACAGCAGACCAGATATACAGATATATGATACAGATACGATA"
```

```
print(t)  
print(s)  
print(finn_forskjeller(t, s))
```

og få ut:

```
ACGTGCACATGAGATCTGATTAGACAGGAGACCAGATATACAGATATATGATACAGATACGATA  
ACGTGCACATGAGATCTGATTAGACAGCAGACCAGATATACAGATATATGATACAGATACGATA  
-----+-----
```

Fyll inn koden under så funksjonen virker som dette. Test at koden fungerer som den skal

In []:

```
1 def finn_forskjeller(s, t):  
2     """Gitt to strenger, s og t, finn forskjellene mellom dem."""  
3     ...
```

In []:

```
1 t = "ACGTGCACATGAGATCTGATTAGACAGGAGACCAGATATACAGATATATGATACAGATACGATA"  
2 s = "ACGTGCACATGAGATCTGATTAGACAGCAGACCAGATATACAGATATATGATACAGATACGATA"  
3  
4 print(t)  
5 print(s)  
6 print(finn_forskjeller(t, s))
```

2) Case: Gener, Insulin og Diabetes

Vi skal nå ta for oss et case som handler om gener, mutasjoner og diabetes. Her kan vi se hvordan noen av de verktøyene vi laget over kan hjelpe oss å forstå en genetisk studie som ble gjennomført i 2008. Opplegget er inspirert av <http://education.expasy.org/bioinformatique/Diabetes.html> (<http://education.expasy.org/bioinformatique/Diabetes.html>), hvor mer informasjon også kan finnes.

Diabetes er sykdom som fører til abnormalt høye nivåer av blodsukker som et resultat av at kroppen ikke klarer å produsere, eller benytte seg av, insulin. Diabetes som sykdom kan være forrsaket av mange ulike faktorer, og som oftest tenker man at diabetes er delvis genetisk, og delvis miljøpåvirket. Merk spesielt at diabetes er en type sykdom, og forskjellige mennesker kan ha veldig forskjellige former for diabetes.

I denne caseoppgaven skal vi se på en sjeldnere form for diabetes som kalles *monogen diabetes*. Navnet *monogen* refererer til at dette formen for diabetes er forårsaket av punktmutasjoner til ett enkelt gen (mono=én/ett). I monogenisk diabetes deler man gjerne videre opp i to grupper, de som blir diagnostisert med diabetes som nyfødte (nyføtdiabetes), og de som utvikler det senere i livet, gjerne iløpet av ungdomsårene (MODY diabetes).

Monogen diabetes er som nevnt forårsaket på grunn av mutasjoner til ett enkelt gen, men for forskjellige mennesker kan det være mutasjoner til forskjellige gener. Ettersom at diabetes er en sykdom forårsaket av unormalt insulin, skulle man kanskje tro at genet for insulin var det vanligste genet i monogen diabetes, men slik er det ikke. Ifølge diabetesforbundet er det til nå 13 kjente ulike former for MODY, som kommer av mutasjoner til 5 ulike gener:

- CSK-genet
- HNF1A-genet
- HNF1B-genet
- HNF4A-genet
- INS-genet

Her er det sistnevnte genet, INS, genet for insulin. Derimot er monogen MODY diabetes ofte forårsaket av mutasjonen til de to første genene

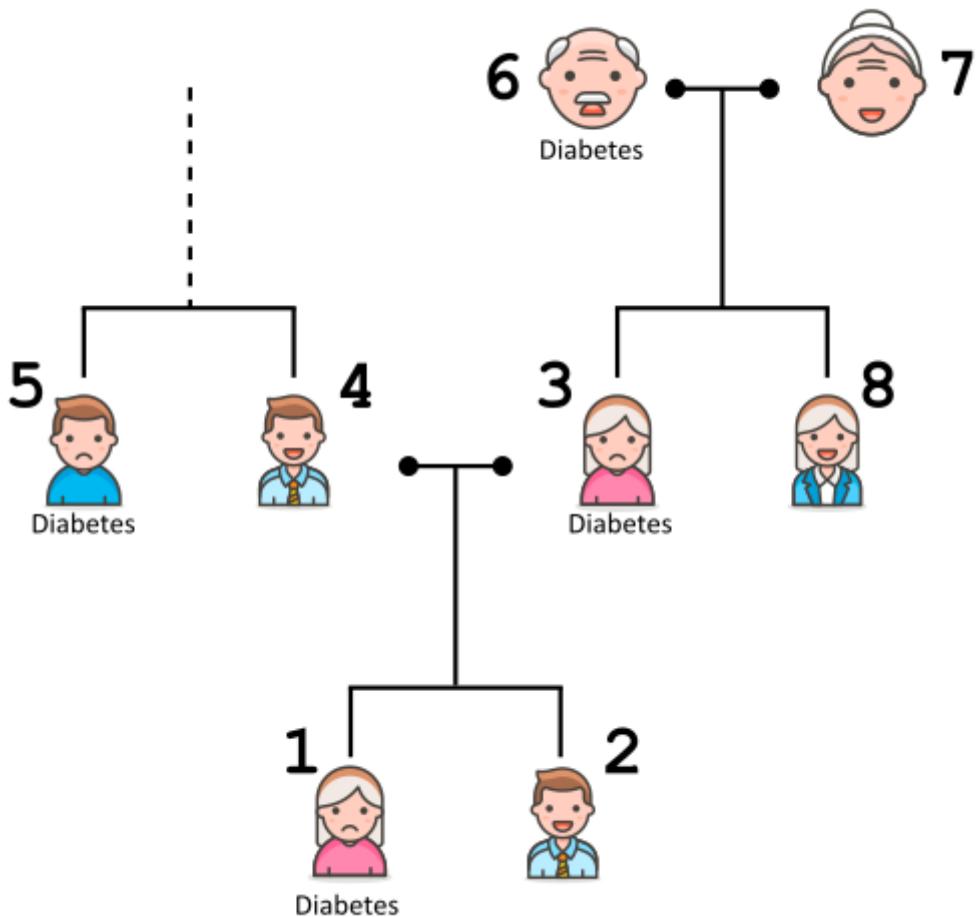
Vi skal nå se på en forskningsstudie fra 2008. På dette tidspunktet var det kjente tilfeller av nyføtdiabetes grunnet mutasjoner i INS-genet, men ikke tilfeller av MODY. Forskningsgruppen som gjennomførte studien hadde en hypotese om at det også måtte finnes tilfeller av MODY som var grunnet INS-mutasjoner, og forsket derfor på dette.

I studien gikk forskerne DNA for INS-genet fra en rekke pasienter med MODY for å finne mulige mutasjoner. Blant annet jobbe de seg igjennom en norsk database for diabetes blant barn. De klarte til slutt å finne to ulike former for slike mutasjoner. Vi skal nå se på ett av disse to tilfellene.

Vi har lagt ved den originale forskningsartikkelen sammen med opplegget, men merk at dette er en fagartikkel, og er derfor ikke så veldig lett å lese eller forstå seg på. Vi legger den ved så man kan få et forhold til hvordan forskningsartikler ser ut.

Familien

I det ene tilfellet hvor de oppdaget en ny form for monogenisk diabetes har vi DNA-sekvensiering av INS-genet fra 8 ulike personer i samme slekt. De 8 personene og deres plass i slektstreet ser ut som følger:



Av de 8 personene har 4 blitt diagnostisert med diabetes. Person 5 og 6 har diabetes type 2, som de ble diagnostisert med ved henholdsvis ved 50 og 40 års alder. Person 1 og 3 ble derimot diagnostisert i ungdomsårene, ved hhv alder av 10 og 13 år. Vi skal nå se nærmere på DNA-et for INS-genet til disse menneskene og se hva vi kan finne.

Villtypen

Ettersom at mennesker deler 99.9% av DNA-et sitt har de aller fleste en identisk kopi av INS-genet. Denne "vanlige" allele forekomsten kalles gjerne for *villtypen* til INS-genet ("wildtype" på engelsk). Men noen har altså et INS-gen der det forekommer ulikheter fra villtypen.

Vi kommer her ikke til å se på *hele* INS-genet, fordi det er ganske stort, men vi velger oss en liten bit av den, nemlig sekvensen:

In []:

```
1 villtype = "GTGTGCGGGGAACGAGGCTTCTTACACACCCAAGACCCGCCGGGAGGCAGAGGAC"
```

Oppgave 2a) Hvilket kromosom ligger INS-genet på?

Før vi begynner å se på mutasjonene våres, kan vi jo finne ut litt mer om INS-genet, for eksempel hvilket kromosom det ligger på. For å gjøre dette kan vi søke etter villtype sekvensen i det mennesklige Genomet. Dette finnes det flere verktøy for å gjøre, for eksempel denne nettsiden:

- <http://genome.ucsc.edu/cgi-bin/hgBlat> (<http://genome.ucsc.edu/cgi-bin/hgBlat>)

Gå inn på siden, lim inn INS-gen sekvensen vi har gitt og gjør et søk. Hvilket kromosom ligger INS-genet på?

DNA-sekvensene

Mennesker er diploide, som betyr at vi har do kopier av hele arvematerialet. Enhver person vil altså bære to ulike kopier av INS-genet. For hver av de 8 personene har vi da to *alleler* av INS-genet. Under gir vi de totalt 16 variantene av den samme biten av INS-genet. Merk at INS-genet er større en den biten vi gir her, men du kan anta at alle de andre bitene av genet er likt som villtypen for alle 8. Vi har altså plukket ut den viktige biten av genet for denne studien.

In []:

```

1 person1_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCTGCCGGAGGCAGAGGAC"
2 person1_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
3
4 person2_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
5 person2_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
6
7 person3_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCTGCCGGAGGCAGAGGAC"
8 person3_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
9
10 person4_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
11 person4_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
12
13 person5_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
14 person5_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
15
16 person6_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
17 person6_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
18
19 person7_allele1 = "GTGTGCGGAGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
20 person7_allele2 = "GTGTGCGGAGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
21
22 person8_allele1 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"
23 person8_allele2 = "GTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGAGGCAGAGGAC"

```

Oppgave 2b) Finne punktmutasjoner

Bruk funksjonen du skrev for å finne *Hamming*-avstanden mellom to sekvenser til å sjekke hvor langt unna villtypen de 16 allellene er. (Hint: Bruk Ctrl+C og Ctrl+V flitting så du slipper å skrive så utrolig mye.)

In []:

```

1 print("Antall mutasjoner:")
2 print("====")
3 print(f"Person 1, allel 1: {hamming(...)}")
4 print(f"Person 1, allel 2: {hamming(...)}")
5 print(...)

```

Oppgave 2c) Kartlegge mutasjonene

- Hvilke alleler skiller seg ut fra villtypen?
- Hvor mange mutasjoner er det som skiller disse allelene fra villtypen?
- Klarer du å finne ut akkurat hvor mutasjonen er? Er alle allele samme mutasjon?
- Om man har to like alleler av et gen kalles dette for en homozygot, har man to ulike alleler er det heterozygot. Klassifiser de ulike mutasjonene som homo/heterozygote.

Oppgave 2d) Finne konsekvensene av mutasjonen

En punktmutasjon betyr at en gitt base i DNA-sekvensen er byttet ut med en annen, men hva vil dette bety i praksis?

- Kjør villtypen og de muterte allelene du fant igjennom prosessen av transkripsjon og translasjon du gjorde istad og skriv ut polypeptidet du får ut av hver variant.
- (OBS: Du trenger bare å sjekke hver *unike mutasjon* du fant istad, du trenger ikke kjøre igjennom alle 16 allelene, da de fleste viste seg å være identiske!)

In []:

```
1 print("Villtype:")
2 kodetråd = ...
3 maltråd = ...
4 mRNA = ...
5 polypeptid = ...
6 print(polypeptid)
7
8 print("Person X:")
9 kodetråd = ...
10 maltråd = ...
11 mRNA = ...
12 polypeptid = ...
13 print(polypeptid)
```

- Hvilke aminosyrer har blitt byttet ut?
- Forklar hvordan en allele kan inneholde punktmutasjoner, uten at proteinet det koder for ikke er endret.
- Kan du forklare hvem av de 8 personene i familien som lider av monogenisk diabetes grunnet mutasjon til INS-genet?

Oppsummerende kommentarer og Løsning

(Under oppsummerer vi funnene i forskningsrapporten, dette avslører "løsningen" til oppgaven. Vent med å lese dette til du vil sjekke svaret ditt, eller til du har gjort et ordentlig forsøk men sitter fast.)

Ved å analysere DNA-et til de 8 menneskene finner vi at personene som ble diagnostisert med diabetes i ungdomsårene (Person 1 og 3) har en mutasjon på INS-genet som gir en endring av en aminosyre fra arginin (R) til cystein (C). Forskerne som laget denne rapporten så nærmere og fant at dette var grunnen til at de to personene utviklet diabetes. Denne formen for monogenisk diabetes blir referret til som R55C, fordi den 55 aminosyren i INS-genet er byttet fra R til C.

Ettersom at foreldrene til Person 3 ikke har den samme mutasjonen i noen av sine alleler kan vi også konkludere med at denne mutasjonen har oppstått i Person 3, som så har gitt den videre til sin datter (Person 1), men ikke til sin sønn (Person 2).

Vi ser også at person 7 har mutasjoner i INS-genet i begge sine alleler. Derimot påvirker ikke dette faktisk insulin-proteinet til denne personen, fordi som vi ser koder genet for nøyaktig de samme aminosyrene. Dette er fordi det er såpass mange overlappende kodon, så punktmutasjonen fra G til A i DNA-et fører fortsatt ikke til noen endring i aminosyrene. Av denne grunnen er det slik at de fleste mutasjoner faktisk ikke fører til noen endringer i praksis.