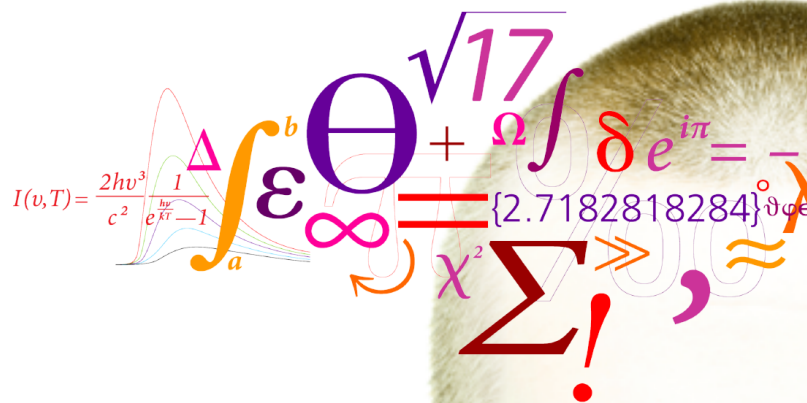


# Introduction to programming and data processing

Project Catalogue for DTU Course 02631–34, 02691–94

Mikkel N. Schmidt, Vedrana Andersen, Martin S. Andersen,  
Jens Starke, Nikolay K. Dimitrov.

2017



## Indhold

<b>Krav til projektarbejdet</b>	<b>i</b>
Introduktion . . . . .	i
Program-specifikationer . . . . .	i
Evaluering . . . . .	ii
Aflevering . . . . .	ii
 <b>1 Øvelsesprojekt</b>	 <b>1</b>
<b>1A Bakterie-dataanalyse</b>	<b>3</b>
Introduktion . . . . .	3
Data load funktion . . . . .	3
Data statistik funktion . . . . .	4
Data plot funktion . . . . .	4
Hoved-script . . . . .	5
 <b>2 Eksamensprojekt</b>	 <b>7</b>
<b>2B Program til karaktergivning af studerende</b>	<b>9</b>
Introduktion . . . . .	9
Karakterafrundingsfunktion . . . . .	9
Endelig karakter-funktion . . . . .	9
Karakter plot-funktion . . . . .	10
Hoved-script . . . . .	10
<b>2C Nedbøjning af bjælke</b>	<b>13</b>
Introduktion . . . . .	13
Funktion til beregning af bjælke nedbøjning . . . . .	13
Superpositionsfunktion . . . . .	13
Bjælke plot funktion . . . . .	14
Hoved-script . . . . .	14
<b>2D Lindenmayer-systemer</b>	<b>17</b>
Introduktion . . . . .	17
Lindenmayer Iteration . . . . .	18
Oversættelse til skildpadde grafik kommandoer . . . . .	18
Plot funktion for skildpadde grafik . . . . .	19
Hoved-script . . . . .	19
<b>2E Vind-data-analyse</b>	<b>21</b>
Introduktion . . . . .	21
Data load funktion . . . . .	21
Data statistik funktion . . . . .	22
Data plot funktion . . . . .	22
Hoved-script . . . . .	23
<b>2F Analyse af el-forbrug i husholdning</b>	<b>25</b>
Introduktion . . . . .	25
Data load funktion . . . . .	25
Data aggregeringsfunktion . . . . .	26
Statistik-funktion . . . . .	26
Hoved-script . . . . .	27

# Krav til projektarbejdet

## Introduktion

I dette projektarbejde skal du udvikle, teste og dokumentere et program. Den overordnede ide bag projekterne er at teste din forståelse af de programmerings-værktøjer og -metoder som du har lært gennem kurset, og at gøre dette på en sjov og berigende måde. Det er dit ansvar at demonstrere at du opfylder læringsmålene (se DTU Kursusbasen). Din kode vil blive evalueret efter de kriterier der er beskrevet nedenfor, og du skal aflevere dit projekt i overensstemmelse med de beskrevne kriterier inden den deadline der fremgår af CampusNet (Assignments). Afleveringer der modtages efter deadline vil ikke blive accepteret.

## Program-specifikationer

I projektbeskrivelserne vil specifikationerne for programmet blive beskrevet, dvs. de påkrævede funktioner og hoved-script. Bemærk at specifikationerne ikke er udtømmende beskrevet, og at en del af opgaven er at identificere, løse, implementere og dokumentere dele af programmet. Dette er valgt, fordi vi ønsker at se jer træffe jeres egne valg omkring håndteringen af de problemer i møder. Det er fuldt ud tilladt, at I træffer jeres egne valg når noget ikke er beskrevet i specifikationerne. Bemærk dog at jeres valg skal dokumenteres og jeres argumenter for at træffe disse valg vil blive evalueret.

## Overordnede krav

- Alle funktioner skal implementeres ifølge det givne interface og må ikke implementeres som scripts. Det er tilladt at benytte under-funktioner og indbyggede funktioner inden i en funktion. Du må også gerne udvikle "hjælpe"-funktioner som kaldes inde fra en funktion, hvis du mener det giver en mere elegant eller praktisk løsning.
- Hoved-scriptet skal implementeres ifølge de givne specifikationer. Det er ikke tilladt at implementere hoved-scriptet som en funktion.
- Det er ikke tilladt at kalde et script inde fra et andet script eller fra en funktion. Du må kun implementere ét script, nemlig hoved-scriptet.
- Du må gerne benytte indbyggede funktioner, og du opfordres til at undersøge om der findes indbyggede funktioner som du kan bruge i din egen kode.
- Alle funktioner skal indeholde en passende hjælpe-tekst som beskriver formålet med funktionen, input og output argumenter, hvordan funktionen benyttes samt anden information der er nødvendig for at forstå og benytte funktionen. Det anbefales at benytte de hjælpe-tekster som findes for de indbyggede funktioner til inspiration.
- Overdreven brug af fejlhåndtering inde i en funktion kan gøre funktionen unødvendigt langsom, særligt hvis funktionen kaldes ofte. Derfor er det ofte den bedste praksis at antage at alle input argumenter til en funktion er gyldige (og angive i funktionens hjælpe-tekst hvilke krav der er til argumenterne.) Bemærk at dette ikke gælder for interaktive bruger-input. Interaktive bruger-input skal fejl-checkes i forbindelse med at de indlæses.

## Dine egne udvidelser

- Du opfordres til at udvide funktionaliteten af de specificerede funktioner, hvis det lykkes at færdiggøre de specificerede krav til funktioner og script inden deadline. På denne måde får du mere programmeringserfaring, og du kan yderligere illustrerer omfanget af din programmeringsforståelse. Bemærk: Vælger du at udvide funktionerne, er det tilladt at udvide de specificerede interfaces, givet at det stadig er muligt at kalde funktionerne som beskrevet.

---

## Evaluering

Vi benytter de følgende kriterier i bedømmelsen af dit arbejde.

- Opfyldelse af læringsmålene, som specificeret i kursusbasen (<http://www.kurser.dtu.dk>) koblet til pensum.
- Helheden og kvaliteten af jeres program-implementation evalueret i forhold til projektkravene, læringsmålene samt “best practice” programming, som beskrevet i modul-noterne.

Mere specifikt vil vi undersøge følgende (ikke udtømmende) liste af kriterier:

### A) Specifikationer

- Er al specificerede funktionalitet implementeret?
- Følger de påkrævede funktioner det specificerede format?
- Er alle specifikationer i øvrigt overholdt?

### B) Korrekthed

- Indlæses data korrekt?
- Gør alle funktioner det de skal?
- Er beregninger mv. korrekt udført?

### C) Struktur og valg af løsninger

- Er der for lang / uoverskuelig kode?
- Er der over-komplicerede løsninger som burde simplificeres?
- Er løkker og vektoriseret beregning anvendt fornuftigt?
- Anvendes passende funktioner fra eksisterende programbiblioteker?

### D) Visualisering og grafiske plots

- Viser plots det de skal?
- Er der fornuftige akse-betegnelser, legends, overskifter mv.?
- Er plots overskuelige og lette at aflæse?

### E) Kommentarer og dokumentation

- Er interfacet til alle funktioner dokumenteret fyldestgørende?
- Er der en passende mængde kommentarer i koden?
- Er kommentarerne gode og brugbare til at skabe overblik over koden?

### F) Brugervenlighed og fejlhåndtering

- Er det let og logisk for en bruger at anvende?
- Er brugeren informeret om hvordan programmet anvendes?
- Håndteres forkert brugerinput på en fornuftig måde?

## Aflevering

Før deadline (se CampusNet Assignments) skal du aflevere alle implementerede kode-filer (hoved-script og funktioner) samt andre nødvendige filer, separat eller i en pakket zip-fil som en “gruppeaflevering”. Husk at teste at script og funktioner kører fejlfrit inden aflevering, da en del af vores vurderingsgrundlag er kørsel og test af programmet.

# Projekt 1

## Øvelsesprojekt



# Bakterie-dataanalyse

## Introduktion

I dette projekt skal du udvikle et computerprogram til håndtering af data relateret til vækstrater for forskellige bakterier ved forskellige temperaturer. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

## Data load funktion

### Interface

```
def dataLoad(filename):  
    # Insert your code here  
    return data
```

### Input argumenter

**filename:** En tekst-streng som indeholder filnavnet på en datafil.

### Output argumenter

**data:** En  $N \times 3$  matrix.

### Bruger-input

Nej.

### Skærm-output

Ja (fejl-beskeder, se specifikationerne nedenfor.)

### Beskrivelse

Funktionen skal indlæse data fra datafilen **filename**. Hver linje i datafilen indeholder de følgende felter:

**Temperature**, **Growth rate**, og **Bacteria**.

Felterne indeholder numeriske værdier og er separeret af et mellemrum. Det følgende illustrerer et eksempel på et uddrag af en sådan datafil:

```
25  0.109  1  
20  0.096  2  
15  0.517  3  
35  1.086  4  
40  0.934  2  
35  0.109  1  
⋮
```

Feltet **Bacteria** indeholder en numerisk kode som svarer til et af de følgende bakterie-navne:

Bacteria	Navn
1	Salmonella enterica
2	Bacillus cereus
3	Listeria
4	Brochothrix thermosphacta

Data fra filen skal gemmes i en  $N \times 3$  matrix kaldet **data**, hvor  $N$  er antallet af gyldige rækker i datafilen.

### Håndtering af data-fejl

Der kan være en eller flere fejlbehæftede (ugyldige) linjer i datafilen, hvilket funktionen skal håndtere. Hvis data load funktionen detekterer en fejlbehæftet linje i datafilen skal den springe denne linje over, skrive en fejlbesked på skærmen og fortsætte til den næste linje. Funktionen skal kun returnere data svarende til de gyldige linjer. Fejlbeskeden skal beskrive i hvilken linje fejlen var og hvad fejlen bestod i. Funktionen skal checke følgende:

- **Temperature** skal være et tal mellem 10 og 60.
- **Growth rate** skal være et positivt tal.
- **Bacteria** skal være en af de fire beskrevet i tabellen ovenfor.

## Data statistik funktion

Interface	<pre>def dataStatistics(data, statistic):     # Insert your code here     return result</pre>
Input argumenter	<b>data</b> : En $N \times 3$ matrix med søjlerne Temperature, Growth rate, og Bacteria. <b>statistic</b> : En tekst-streng som specificerer den statistik som skal beregnes.
Output argumenter	<b>result</b> : En skalar indeholdende den beregnede statistik.
Bruger-input	Nej.
Skærm-output	Nej.
Beskrivelse	<p>Denne funktion skal beregne en af et antal mulige statistikker baseret på data. En "statistik" skal her forstås som et enkelt tal som beskriver et aspekt ved data, som for eksempel en middelværdi (et gennemsnit). Hvilken statistik der skal beregnes afhænger af værdien af tekst-strengen <b>statistic</b>. Den følgende tabel viser de forskellige mulige værdier af <b>statistic</b> og en beskrivelse af beregningen af de tilsvarende statistikker.</p>

<b>statistic</b>	Beskrivelse
'Mean Temperature'	Middelværdi (gennemsnit) af Temperature.
'Mean Growth rate'	Middelværdi (gennemsnit) af Growth rate.
'Std Temperature'	Standard-afvigelse af Temperature.
'Std Growth rate'	Standard-afvigelse af Growth rate.
'Rows'	Det totale antal rækker i data.
'Mean Cold Growth rate'	Middelværdi (gennemsnit) af Growth rate hvor Temperature er mindre end 20 grader.
'Mean Hot Growth rate'	Middelværdi (gennemsnit) af Growth rate hvor Temperature er større end 50 grader.

Du opfordres til at benytte indbyggede funktioner til at beregne disse statistikker hvor det er muligt.

## Data plot funktion

Interface	<pre>def dataPlot(data):     # Insert your code here</pre>
Input argumenter	<b>data</b> : En $N \times 3$ matrix med søjlerne Temperature, Growth rate, og Bacteria.
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja (diagrammer, se specifikationer nedenfor.)
Beskrivelse	<p>Denne funktion skal generere og vise to diagrammer:</p> <ol style="list-style-type: none"> <li>1. "Number of bacteria": Et søjlediagram af antallet af hver af de forskellige typer Bacteria i data.</li> <li>2. "Growth rate by temperature": Et diagram med Temperature på x-aksen og Growth rate på y-aksen. X-aksen skal gå fra 10 til 60 grader og y-aksen skal starte ved 0. Plottet skal have ét koordinatsystem med fire grafer, en for hver type Bacteria. De forskellige grafer skal kunne skelnes fra hinanden for eksempel ved brug af forskellige farver, markører eller linje-typer.</li> </ol>

Diagrammerne skal indeholde passende titel, beskrivende akse-betegnelser og en signaturforklaring hvor det er hensigtsmæssigt. Du må gerne præsentere diagrammerne i separate figur-vinduer eller som under-plot i et enkelt figur-vindue.



## Hoved-script

Interface	Skal implementeres som et script.
Input argumenter	Nej.
Output argumenter	Nej.
Bruger-input	Ja (se specifikationer nedenfor.)
Skærm-output	Ja (se specifikationer nedenfor.)
Beskrivelse	<p>Brugeren af data-analyse programmet interagerer med programmet gennem hoved-scriptet. Når brugeren kører hoved-scriptet skal han/hun som minimum have følgende muligheder:</p> <ol style="list-style-type: none"> <li>1. Indlæs data.</li> <li>2. Filtrer data.</li> <li>3. Vis statistik.</li> <li>4. Generer diagrammer.</li> <li>5. Afslut.</li> </ol> <p>Det skal være muligt for brugeren at udføre disse handlinger (se specifikationer nedenfor) i vilkårlig rækkefølge så længe han/hun ønsker det, indtil han/hun vælger at afslutte programmet. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et cool script.</p>
Fejlhåndtering	<p>Du skal teste at alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at indtaste et nyt gyldigt input.</p> <p>Det må ikke være muligt at vise statistikker eller generere diagrammer før data er indlæst. Hvis brugeren forsøger at gøre dette skal han/hun have passende feedback der forklarer at dette ikke er muligt.</p>
1. Indlæs data	<p>Hvis brugeren vælger at indlæse data, skal du bede brugeren om at indtaste et filnavn på en data-fil. Husk at checke at filnavnet er gyldigt. Indlæs data ved hjælp af <code>dataLoad</code> funktionen, som vil vise information om eventuelle fejlbehæftede linjer i datafilen.</p>
2. Filter data	<p>Hvis brugeren vælger at filtrere data, skal han/hun have mulighed for at specificere en eller flere betingelser som skal være opfyldt for at en data-række inkluderes i beregningen af statistikker og generering af diagrammer. Som minimum skal brugeren kunne specificere to typer af filtre:</p> <ol style="list-style-type: none"> <li>1. Et filter for typen af Bacteria, for eksempel <code>Bacteria = Listeria</code>.</li> <li>2. Et interval-filter for Growth rate, for eksempel <math>0.5 \leq \text{Growth rate} \leq 1</math>.</li> </ol> <p>Når et sådant filter er specificeret, skal statistikker og diagrammer genereres på baggrund af den delmængde af data-rækker hvor filterets kriterier er opfyldt. Brugeren skal også have mulighed for at deaktivere filterbetingelserne, og når han/hun gør dette skal efterfølgende statistikker og diagrammer igen genereres på baggrund af alle data-rækker. Hvis et filter er aktivt skal information om filteret hele tiden være synligt i programmets brugerinterface.</p>

- 3. Vis statistik      Hvis brugeren vælger at vise statistikker, skal du spørge brugeren hvilken statistik der skal vises. Brug funktionen `dataStatistics` til at beregne den ønskede statistik og vis den på skærmen sammen med en beskrivelse af statistikken. Hvis et filter er aktivt, skal statistikken beregnes udelukkende baseret på de data-rækker det opfylder filterets betingelser.
  
- 4. Generer diagrammer      Hvis brugeren vælger at generere diagrammer, skal funktionen `dataPlot` kaldes for at vise plottene. Hvis et filter er aktivt, skal plottene genereres udelukkende på baseret på de data-rækker der opfylder filterets betingelser.
  
- 5. Afslut      Hvis brugeren vælger at afslutte programmet, skal hoved-scriptet stoppe.
  
- Data fil      Du opfordres til at lave din egen test-datafil for at kontrollere at dit program fungerer korrekt. Det er vigtigt at du også sikrer og dokumenterer at programmet virker korrekt i tilfælde af fejlbehæftede linjer i datafilen som beskrevet i afsnit .

**Projekt 2**

**Eksamensprojekt**



# Program til karaktergivning af studerende

## Introduktion

I dette projekt skal du udvikle, teste og dokumentere et program til behandling af karakterer til studerende. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

## Karakterafrundingsfunktion

Interface	<pre>def roundGrade(grades):     # Insert your code here     return gradesRounded</pre>
Input argumenter	<b>grades</b> : En vektor (hvert element er et tal mellem $-3$ og $12$ ).
Output argumenter	<b>gradesRounded</b> : En vektor (hvert element er et tal på 7-trinsskalaen).
Bruger-input	Nej.
Skærm-output	Nej.
Beskrivelse	Funktionen skal afrunde hvert element i vektoren <b>grades</b> og returnere den nærmeste karakter på 7-trinsskalaen:

7-trinsskalaen: Karakterer	12	10	7	4	02	00	$-3$
----------------------------	----	----	---	---	----	----	------

For eksempel, hvis funktionen får vektoren  $[8.2, -0.5]$  som input, skal den returnere de afrundede karakterer  $[7, 0]$ , som er de nærmeste tal på karakterskalaen.

## Endelig karakter-funktion

Interface	<pre>def computeFinalGrades(grades):     # Insert your code here     return gradesFinal</pre>
Input argumenter	<b>grades</b> : En $N \times M$ matrix som indeholder karakterer på 7-trinsskalaen givet til $N$ studerende for $M$ opgaver.
Output argumenter	<b>gradesFinal</b> : En vektor med længde $N$ som indeholder den endelige karakter for hver af de $N$ studerende.
Bruger-input	Nej.
Skærm-output	Nej.
Beskrivelse	For hver studerende skal den endelige karakter beregnes på følgende måde:

1. Hvis der kun er en opgave ( $M = 1$ ) er den endelige karakter lig karakteren for den opgave.
2. Hvis der er to eller flere opgaver ( $M > 1$ ) skal den laveste karakter ikke medregnes. Den endelige karakter beregnes som gennemsnittet af de  $M-1$  højeste karakterer, afrundet til nærmeste karakter på skalaen (vha. funktionen **roundGrade**).
3. Uanset ovenstående skal den endelige karakter altid være  $-3$ , hvis den studerende har fået  $-3$  i en eller flere af opgaverne.

## Karakter plot-funktion

Interface	<pre>def gradesPlot(grades):     # Insert your code here</pre>
Input argumenter	<b>grades</b> : En $N \times M$ matrix som indeholder karakterer på 7-trinsskalaen givet til $N$ studerende for $M$ opgaver.
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja (diagrammer, se specifikationer nedenfor.)
Beskrivelse	<p>Denne funktion skal generere og vise to diagrammer:</p> <ol style="list-style-type: none"> <li>1. <b>"Final grades"</b>: Et søjlediagram af antallet af studerende som har fået hver af de mulige endelige karakterer på 7-trinsskalaen (beregnet med funktionen <code>computeFinalGrades</code>).</li> <li>2. <b>"Grades per assignment"</b>: Et plot med opgaverne på x-aksen og karaktererne på y-aksen. X-aksen skal vise alle opgaver 1 til <math>M</math> og y-aksen skal vise alle karakterer fra <math>-3</math> til <math>12</math>. Plottet skal indeholde: <ol style="list-style-type: none"> <li>1. Hver af de afgivne karakterer markeret med en prik. Du skal lægge et lille tilfældigt tal (mellem <math>-0.1</math> og <math>0.1</math>) til x- og y-koordinatet for hver prik, for at kunne skelne de forskellige prikker, som ellers ville ligge oven i hinanden når mere end en studerende har fået samme karakter i den samme opgave.</li> <li>2. Den gennemsnitlige karakter for hver opgave vist som en linje.</li> </ol> </li> </ol>

Diagrammerne skal indeholde passende titel, beskrivende akse-betegnelser og en signaturforklaring hvor det er hensigtsmæssigt. Du må gerne præsentere diagrammerne i separate figur-vinduer eller som under-plot i et enkelt figur-vindue.

## Hoved-script

Interface	Skal implementeres som et script.
Input argumenter	Nej.
Output argumenter	Nej.
Bruger-input	Ja (se specifikationer nedenfor.)
Skærm-output	Ja (se specifikationer nedenfor.)
Beskrivelse	<p>Når brugeren kører hoved-scriptet skal han/hun bedes om at indtaste navnet på en komma-separeret (CSV) datafil som indeholder karakter givet til et antal studerende for et antal opgaver (se beskrivelse af filformat nedenfor). Husk at checke at filnavnet er gyldigt. Efter at have indlæst datafilen, skal du vise information om det indlæste data: Som minimum skal antallet af studerende og antallet af opgaver vises.</p> <p>Derefter skal brugeren som minimum have følgende muligheder:</p> <ol style="list-style-type: none"> <li>1. Indlæs ny data.</li> <li>2. Check for datafejl.</li> <li>3. Generer diagrammer.</li> <li>4. Vis karakterliste.</li> <li>5. Afslut.</li> </ol>

Det skal være muligt for brugeren at udføre disse handlinger (se specifikationer nedenfor) i vilkårlig rækkefølge så længe han/hun ønsker det, indtil brugeren vælger at afslutte programmet. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et cool script.

#### Filformat

Den første række i CSV-filen indeholder en overskrift for hver søjle. Hver af de følgende linjer indeholder et studie-id, navn samt et antal karakterer for en studerende. Et eksempel på en datafil med fire studerende og tre opgaver er givet nedenfor:

```
StudentID,Name,Assignment1,Assignment2,Assignment3
s123456,Michael Andersen,7,7,4
s123789,Bettina Petersen,12,10,10
s123468,Thomas Nielsen,-3,7,2
s123579,Marie Hansen,10,12,12
```

Husk at dit program skal fungere for et vilkårligt antal studerende og et vilkårligt antal opgaver. Du opfordres til at lave din egen test-datafil for at kontrollere at dit program fungerer korrekt.

#### Fejlhåndtering

Du skal teste at alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at indtaste et nyt gyldigt input.

##### 1. Indlæs ny data

Hvis brugeren vælger at indlæse ny data, skal brugeren bedes om at indtaste et gyldigt filnavn på en data-fil, og data skal indlæses på samme måde som i starten af scriptet.

##### 2. Check for datafejl

Hvis brugeren vælger at checke for datafejl, skal du vise en rapport over fejl (hvis der er nogen) i den indlæste datafil. Dit program skal som minimum kunne detektere og vise information om følgende mulige fejl:

1. Hvis to studerende i datasættet har samme studie-id.
2. Hvis en karakter i datasættet ikke er en af de mulige karakterer på 7-trinsskalaen.

##### 3. Generer diagrammer

Hvis brugeren vælger at generere diagrammer, skal funktionen `gradesPlot` kaldes for at vise plottene.

##### 4. Vis karakterliste

Hvis brugeren vælger at vise karakterlisten, skal du vise karaktererne for hver opgave samt den endelige karakter for alle de studerende i alfabetisk orden efter deres navn. Den viste liste skal formateres på en måde så den er let at aflæse.

##### 5. Afslut

Hvis brugeren vælger at afslutte programmet, skal hoved-scriptet stoppe.





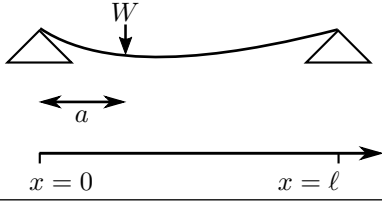
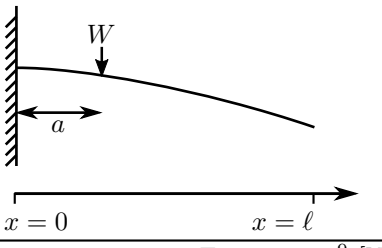
## Nedbøjning af bjælke

### Introduktion

I dette projekt skal du udvikle, teste og dokumentere et program til analyse af et fysisk system hvor en bjælke nedbøjes af en belastning. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

### Funktion til beregning af bjælke nedbøjning

Interface	<pre>def beamDeflection(positions, beamLength, loadPosition, loadForce,                     beamSupport):     # Insert your code here     return deflection</pre>
Input argumenter	<p><b>positions:</b> Positioner [m] hvor bjælkens nedbøjning skal beregnes (vektor), nedenfor benævnt <math>x</math>.</p> <p><b>beamLength:</b> Bjælkens længde [m] (skalar), nedenfor benævnt <math>\ell</math>.</p> <p><b>loadPosition:</b> Belastningens position [m] (skalar), nedenfor benævnt <math>a</math>.</p> <p><b>loadForce:</b> Belastningens kraft [N] (skalar), nedenfor benævnt <math>W</math>.</p> <p><b>beamSupport:</b> Bjælkens understøtning (streng), lig <b>both</b> eller <b>cantilever</b>.</p>
Output argumenter	<p><b>deflection:</b> Nedbøjning [m] (vektor af samme længde som input <b>positions</b>), nedenfor benævnt <math>y</math>.</p>
Bruger-input	Nej.
Skærm-output	Nej
Beskrivelse	Funktionen skal beregne nedbøjningen af en bjælke med den givne længde med en belastning med den givne kraft placeret på den givne position, ifølge formlerne nedenfor. Bjælken kan enten være understøttet i begge ender eller kun i den ene ende (cantilever).

beamSupport	Skitse	Formel for nedbøjning
both		$y = \begin{cases} \frac{W(\ell - a)x}{6EI\ell}(\ell^2 - x^2 - (\ell - a)^2), & x < a \\ \frac{Wa(\ell - x)}{6EI\ell}(\ell^2 - (\ell - x)^2 - a^2), & x \geq a \end{cases}$
cantilever		$y = \begin{cases} \frac{Wx^2}{6EI}(3a - x), & x < a \\ \frac{Wa^2}{6EI}(3x - a), & x \geq a \end{cases}$

$E = 200 \cdot 10^9 \text{ [N/m}^2\text{]}, \quad I = 0.001 \text{ [m}^4\text{]}.$

### Superpositionsfunktion

Interface	<pre>def beamSuperposition(positions, beamLength, loadPositions, loadForces,                         beamSupport):     # Insert your code here     return deflection</pre>
-----------	--

Input argumenter	<b>positions:</b> Positioner [m] hvor bjælkens nedbøjning skal beregnes (vektor). <b>beamLength:</b> Bjælkens længde [m] (skalar). <b>loadPositions:</b> Belastningernes positioner [m] (vektor). <b>loadForces:</b> Belastningernes kræfter [N] (vektor). <b>beamSupport:</b> Bjælkens understøtning (streng), lig <b>both</b> eller <b>cantilever</b> .
Output argumenter	<b>deflection:</b> Nedbøjning [m] (vektor af samme længde som input <b>positions</b> ).
Bruger-input	Nej.
Skærm-output	Nej
Beskrivelse	<p>Funktionen skal beregne nedbøjningen af en bjælke med den givne længde og understøttelse med <i>multiple belastninger</i> med de givne kræfter placeret på de givne positioner. Ifølge superpositionsprincippet skal du, for at beregne nedbøjningen under multiple belastninger, beregne nedbøjningen for hver af belastningerne for sig og summere nedbøjningerne.</p> <p>Hvis der ikke er nogen belastninger (hvis vektorerne <b>loadPositions</b> og <b>loadForces</b> er tomme) skal funktionen returnere en nul-vektor med samme længde som <b>positions</b>.</p>

## Bjælke plot funktion

Interface	<pre>def beamPlot(beamLength, loadPositions, loadForces, beamSupport):     # Insert your code here</pre>
Input argumenter	<b>beamLength:</b> Bjælkens længde [m] (skalar). <b>loadPositions:</b> Belastningernes positioner [m] (vektor). <b>loadForces:</b> Belastningernes kræfter [N] (vektor). <b>beamSupport:</b> Bjælkens understøtning (streng), lig <b>both</b> eller <b>cantilever</b> .
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja (diagrammer, se specifikationer nedenfor.)
Beskrivelse	<p>Denne funktion skal generere og vise et diagram med titlen: "Beam deflection"</p> <p>Plottet skal vise bjælkens nedbøjning som funktion af x-koordinatet for hele bjælken (dvs. et plot i stil med de to skitser i sektion , dog med multiple belastninger). Fra plottet skal det være let at se eller aflæse længden af bjælken, placeringerne og kræfterne af belastningerne, den maksimale nedbøjning, samt typen af bjælkens understøttelse.</p> <p>Diagrammet skal indeholde en titel, beskrivende akse-betegnelser og en signaturforklaring hvis det er hensigtsmæssigt.</p>

## Hoved-script

Interface	Skal implementeres som et script.
Input argumenter	Nej.
Output argumenter	Nej.
Bruger-input	Ja (se specifikationer nedenfor.)
Skærm-output	Ja (se specifikationer nedenfor.)
Beskrivelse	<p>Dette program giver brugeren mulighed for at konfigurere en bjælke med multiple belastninger, gemme/indlæse en konfigureret bjælke, samt vise et plot af bjælkens nedbøjning. Brugeren af programmet interagerer med programmet gennem hoved-scriptet. Når brugeren kører hoved-scriptet skal han/hun som minimum have følgende muligheder:</p>

1. Konfigurer bjælke.
2. Konfigurer belastninger.
3. Gem bjælke og belastninger.
4. Indlæs bjælke og belastninger.
5. Generer diagram.
6. Afslut.

Det skal være muligt for brugeren at udføre disse handlinger (se specifikationer nedenfor) i vilkårlig rækkefølge så længe han/hun ønsker det, indtil han/hun vælger at afslutte programmet. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et cool script.

Fejlhåndtering	Du skal teste at alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at indtaste et nyt gyldigt input. Det må ikke være muligt at have belastninger placeret uden for bjælken eller at have negativ bjækelængde eller negative belastningskræfter.
Initialisering	Når programmet starter skal bjælkens længde sættes til 10 [m], understøttelsen sættes til <code>both</code> og der skal ikke være nogen belastninger.
1. Konfigurer bjælke	Hvis brugeren vælger at konfigurere bjælken, skal han/hun bedes om at indtaste bjælkens længde og understøttelsens type.
2. Konfigurer belastninger	Hvis brugeren vælger at konfigurere belastninger, skal han/hun have mulighed for at se de nuværende belastninger (hvis der er nogen), tilføje en belastning (position samt kraft), eller fjerne en belastning.
3. Gem bjælke og belastninger	Hvis brugeren vælger at gemme, skal han/hun bedes om at indtaste et filnavn. Den nuværende bjælke og belastninger (bjælkens længde, understøttelsens type, belastningernes positioner og kræfter) skal gemmes til filen. Du skal selv beslutte hvilket fil-format du vil benytte.
4. Indlæs bjælke og belastninger	Hvis brugeren vælger at indlæse en bjælke og belastninger, skal han/hun bedes om at indtaste et filnavn, hvorefter bjælken og belastningerne indlæses fra filen. Programmet skal kunne indlæse filer gemt i menupunktet ovenfor.
4. Generer diagram	Hvis brugeren vælger at generere et diagram, skal funktionen <code>beamPlot</code> kaldes for at vise plottet.
5. Afslut	Hvis brugeren vælger at afslutte programmet, skal hoved-scriptet stoppe.



## Lindenmayer-systemer

### Introduktion

I dette projekt skal du udvikle, teste og dokumentere et program til beregning og visualisering af Lindenmeyer-systemer. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

**Lindenmayer-systemer** Et Lindenmeyer-system defineres iterativt og de består af: a) et symbolsk alfabet der kan bruges til at skabe strenge, b) en start streng der bruges til at initialisere den iterative konstruktion og c) erstatnings regler der specificere hvorledes hvert symbol i strengen skal erstattes af en ny streng af symboler. Disse Lindenmayer systemer blev oprindeligt brugt til at beskrive plante cellers adfærd og udviklingen af planters vækst processer. I denne opgave skal I arbejde med to systemer kaldet Koch kurven og Sierpinski trekanten.

**Koch Kurven** Koch kurven kan genereres med a) et symbolsk alfabet bestående af S, L, og R, b) en initialiserende streng 'S' og c) de følgende erstatnings regler

$$\begin{aligned} S &\rightarrow \text{SLSRSL} \\ L &\rightarrow L \\ R &\rightarrow R \end{aligned}$$

Den initialiserende streng er S. Efter første iteration opnåes strengen SLSRSL. Efter anden iteration opnåes strengen SLSRSLSLSRSLSRSLSLSRSL. De første tre iterationer er visualiseret vha. skildpaddegrafik (se nedenfor) i Fig. 2D.1.

**Sierpinski Trekanten** Sierpinski trekanten kan genereres vha. a) et alfabet bestående af symbolerne A, B, L og R, b) en start streng A og c) de følgende erstatnings regler

$$\begin{aligned} A &\rightarrow \text{BRARB} \\ B &\rightarrow \text{ALBLA} \\ L &\rightarrow L \\ R &\rightarrow R \end{aligned}$$

Den initialiserende streng er A. Første iteration giver strengen BRARB. Efter anden iteration opnåes ALBLARBRARBALBLA. Initialisationen og resultatet efter iteration 4 og 8 er visualiseret vha. skildpaddegrafik (se nedenfor) i Fig. 2D.2.

**Visualisering med skildpadde-grafik** Sekvensen af symboler i strengen visualiseres ved at oversætte hvert symbol vha. skildpadde grafik. S, A, og B fortolkes som et linje segment fra en given position (koordinatsystemets origo vil blive brugt til dette formål) langs en given retning — her bruger vi den kanoniske basis vector  $(1, 0)^T$ . L fortolkes som en venstre drejning med vinklen  $\frac{1}{3}\pi$  og R fortolkes som en højre-drejning med vinklen  $-\frac{2}{3}\pi$  (Koch) eller  $-\frac{1}{3}\pi$  (Sierpinski). Længden af linje-segmentet skales med en faktor  $\frac{1}{3}$  (Koch) eller  $\frac{1}{2}$  (Sierpinski) efter hver iteration.

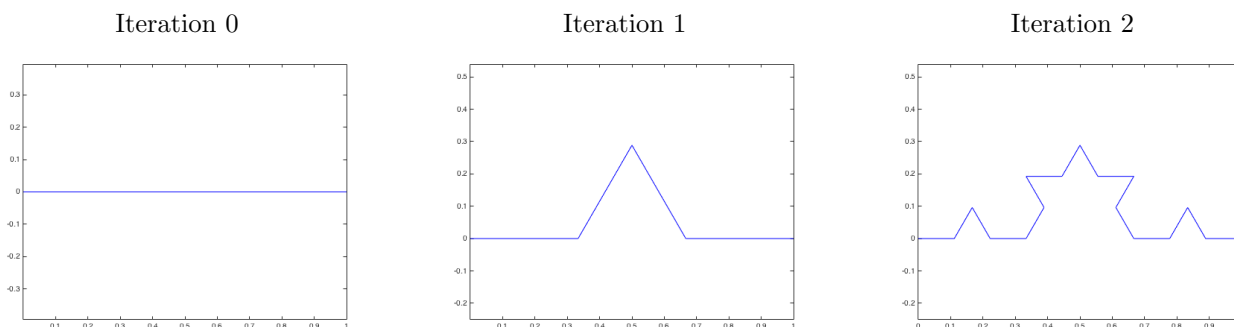


Figure 2D.1: Grafisk repræsentation af de første iterationer i en Koch kurve.

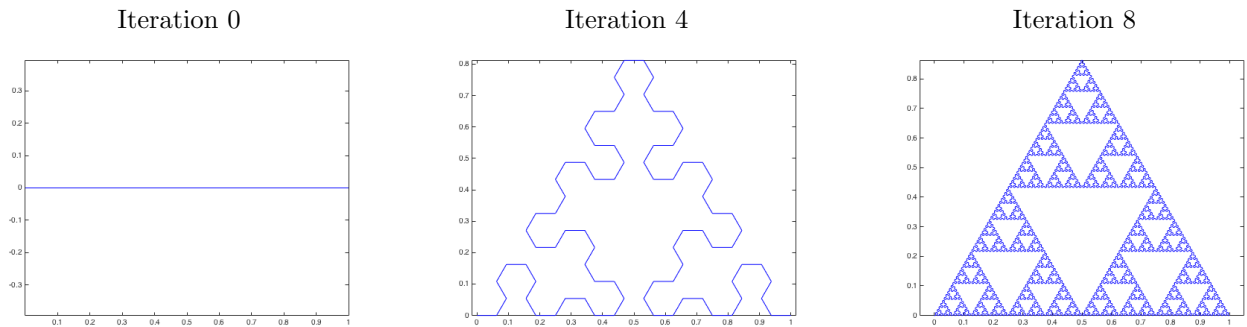


Figure 2D.2: Grafisk repræsentation af initialisationen samt fjerde og ottende iteration af Sierpinski-trekanten.

## Lindenmayer Iteration

### Interface

```
def LindIter(System, N):
    # Insert your code here
    return LindenmayerString
```

### Input argumenter

**System:** En streng der indeholder en specifikation for Lindenmayer systemet der skal undersøges. Inputtet kan indeholde en af værdierne Koch eller Sierpinski.  
**N:** Antallet af iterationer der skal udføres.

### Output argumenter

**LindenmayerString:** En string der indeholder resultatet det valgte Lindenmayer system efter **N** iterationer.

### Bruger-input

Nej.

### Skærm-output

Nej.

### Beskrivelse

Funktionen skal udregne N iterationer for systemet specificeret af **System** ved brug af erstatnings reglerne for enten Koch kurven and Sierpinski trekanten.  
 Outputtet for iterations funktionen skal gives i strengen **LindenmayerString**.

## Oversættelse til skildpadde grafik kommandoer

### Interface

```
def turtleGraph(LindenmayerString):
    # Insert your code here
    return turtleCommands
```

### Input argumenter

**LindenmayerString:** En streng af symboler der repræsenterer systemets stadie efter Lindemayer iterationen.

### Output argumenter

**turtleCommands:** En række vektor der indeholder skildpadde grafik kommandoer der består af skiftevis længde og vinkel specifikationer  $[l_1, \phi_1, l_2, \phi_2, \dots]$ .

### Bruger-input

Nej.

### Skærm-output

Nej.

### Beskrivelse

Denne funktion oversætter symbolerne til en sekvens af skildpadde grafik kommandoer. Outputtet består af en numerisk række vektor, der skiftevis angiver længden af et linje segment der skal tegnes og en vinkel der bestemmer orienteringen i forhold til det tidligere linje segments retning.

Din function skal benytte erstatnings reglerne der svarer til det valgte Lindenmayer system. Dette kan enten udledes af direkte af inputtet eller ved at tilføje et ekstra input argument der specificere systemets type. Du kan frit vælge hvorledes dette problem skal løses og kreativitet vil blive værdsat.

## Plot funktion for skildpadde grafik

Interface	<pre>def turtlePlot(turtleCommands):     # Insert your code here</pre>
Input argumenter	<b>turtleCommands:</b> En række vektor der består af skiftevis længde og vinkel specifikationer $[l_1, \phi_1, l_2, \phi_2, \dots]$ .
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja (diagram, se specifikationer nedenfor.)
Beskrivelse	<p>Plot funktionen skal oversætte <b>turtleCommands</b> vektoren til en grafisk repræsentation. Dette kan gøres ved at forbinde (x, y) koordinaterne af linje segmenternes hjørner. Funktionen skal finde disse koordinater ved at følge input vektoren. Et nyt par koordinater tilføjes for hvert linje segment. Linje segmentet har længde <math>l_i</math> og skal tegnes med vinkel <math>\phi_i</math> i forhold til det tidligere linje segment (som specificeret af input vektoren, <b>turtleCommands</b>). Start punktet er <math>\vec{x}_0 = (0, 0)^T</math> og start retningen er givet ved enhedsvektoren <math>\vec{d}_0 = (1, 0)^T</math>. En positiv vinkel svarer til at en venstre drejning og en negativ svarer til en højre drejning. Udregningen af vektoren der peger mod punkt <math>\vec{x}_{i+1}</math> ved at bruge <math>\vec{x}_i</math> og den tidligere tegne retning <math>\vec{d}_i</math> med,</p>

$$\vec{d}_{i+1} = \begin{pmatrix} \cos(\phi_{i+1}) & -\sin(\phi_{i+1}) \\ \sin(\phi_{i+1}) & \cos(\phi_{i+1}) \end{pmatrix} \cdot \vec{d}_i \quad (2D.1)$$

$$\vec{x}_{i+1} = \vec{x}_i + l_{i+1} \cdot \vec{d}_{i+1} \quad (2D.2)$$

Resultaterne kan checkes ved sammeligning med Fig. 2D.1 og 2D.2. Diagrammerne skal indeholde en passende titel, beskrivende akse-betegnelser og en signaturforklaring hvor det er hensigtsmæssigt. Du må gerne vise diagrammerne i separate figur-vinduer eller som under-plot i et enkelt figur-vindue.

## Hoved-script

Interface	Skal implementeres som et script.
Input argumenter	Nej.
Output argumenter	Nej.
Bruger-input	Ja (se specifikationer nedenfor.)
Skærm-output	Ja (se specifikationer nedenfor.)
Beskrivelse	<p>Brugeren interagerer med programmet gennem hoved-scriptet. Når brugeren kører hoved-scriptet skal han/hun som minimum have følgende muligheder:</p> <ol style="list-style-type: none"> <li>1. Vælg en type af Lindenmayer system og et antal iterationer.</li> <li>2. Generer diagrammer.</li> <li>3. Afslut.</li> </ol>

Brugeren skal kunne udføre enhvert rimeligt antal af Lindenmayer iterationer for det valgte Lindenmayer system og efterfølgende grafiske repræsentation af den resulterende streng. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et superduper fedt script.

### Fejlhåndtering

Du skal teste hvorvidt alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at intaste et nyt gyldigt input. Dette inkluderer også urimelige valgt af iterations antal (fx. pga. eksekveringstid).

1. **Vælg et Lindenmayer System** Når brugeren skal vælge et Lindemayer system, skal han/hun bedes om en af følgende muligheder.

Input Option	Navn
1	Koch curve
2	Sierpinski triangle

2. **Generer diagrammer**

Hvis brugeren vælger at generere diagrammer, så kald funktionen `turtlePlot` for at vise et diagram svarende til det valgte Lindenmayer system.

3. **Afslut**

Hvis brugeren vælger at afslutte programmet, skal hoved-scriptet stoppe.

### Mulige udvidelser

En mulig udvidelse (ikke påkrævet) kunne være at du tillader brugeren at definere et nyt Lindenmayer system, udregne iterationer og visualisere dette. Andre muligheder kunne være at udregne visse egenskaber af Lindenmayer systemerne såsom kurve længde afhængigt af iterations antal.



# Vind-data-analyse

## Introduktion

I dette projekt skal du udvikle et computerprogram til håndtering af data relateret til vindenergi. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

## Data load funktion

### Interface

```
def dataLoad(filename, Nx, Ny, Nz):  
    # Insert your code here  
    return data
```

### Input argumenter

**filename:** En tekst-streng som indeholder filnavnet på en datafil.  
**Nx, Ny, Nz:** Størrelsen på første, anden og tredje dimension (x-, y- og z-koordinat) af det 3-dimensionelle output array.

### Output argumenter

**data:** Et 3-dimensionelt array med størrelse  $N_x \times N_y \times N_z$ .

### Bruger-input

Nej.

### Skærm-output

Ja (fejl-beskeder, se specifikationerne nedenfor.)

### Beskrivelse

Funktionen skal indlæse data fra den binære fil **filename**. Bemærk at filen er en binær fil og ikke en tekst-fil. Datafilen indeholder  $N_z \times N_y \times N_x$  tal (floating point single precision). Sekvensen af tallene svarer til indexene  $z$ ,  $y$  og  $x$  som alle stiger sekventielt fra 1 til  $N_z$ ,  $N_y$  and  $N_x$  respektivt. Det hurtigst varierende index er  $z$ , efterfulgt af  $y$ , og det langsomst varierende index er  $x$ . Det vil sige at de første  $N_z$  tal fra sekvensen svarer til index  $z =$  gående fra 1 til  $N_z$ ,  $y = 1$ , og  $x = 1$ . På baggrund af denne rækkefølge skal funktionen konvertere data til et tre-dimensionel array med dimensionerne  $N_z \times N_y \times N_x$ .

### Håndtering af data-fejl

Der kan være uoverensstemmelse mellem antallet af data punkter i input-filen og dimensionerne af det specificered array. Hvis antallet af datapunkter ikke svarer overens med det forventede antal, skal funktionen udskrive en passende informativ fejlmeddelelse.

## Data statistik funktion

Interface	<pre>def dataStatistics(data, statistic, Yref, Zref, DeltaX):     # Insert your code here     return result</pre>
Input argumenter	<p><b>data:</b> Et <math>N_z \times N_y \times N_x</math> array som indeholder vindhastigheder.</p> <p><b>statistic:</b> En tekst-streng som specificerer den statistik som skal beregnes (<b>Mean</b>, <b>Variance</b> eller <b>Cross correlation</b>.)</p> <p><b>Yref, Zref:</b> Reference y- og z-koordinat til kryds-korrelationen. Kun nødvendig når <b>statistic</b> er <b>Cross correlation</b>. Kaldet <math>y_{\text{ref}}</math> og <math>z_{\text{ref}}</math> nedenfor.</p> <p><b>DeltaX:</b> Separation i x-koordinat for hvilken kryds-korrelationen skal evalueres. Kun nødvendig når <b>statistic</b> er <b>Cross correlation</b>. Kaldet <math>\Delta x</math> nedenfor.</p>
Output argumenter	<p><b>result:</b> Et to-dimensionelt array med størrelse <math>(N_y \times N_z)</math> indeholdende de beregnede statistikker for hvert punkt i y-z planet.</p>
Bruger-input	Nej.
Skærm-output	Nej.
Beskrivelse	<p>Denne funktion skal beregne en af tre mulige statistikker baseret på data. Med “statistik” menes her en <math>N_y \times N_z</math> matrix af tal beregnet henover x-dimensionen for hvert af y- og z-koordinaterne. Det følgende beskriver de forskellige statistikker og hvordan de skal beregnes.</p>

**Mean** Middelværdien for hvert punkt i y-z planet, beregnet over x-dimensioen.

$$M_{y,z} = \frac{1}{N_x} \sum_{x=1}^{N_x} D_{x,y,z} \quad (2E.1)$$

**Variance** Variansen for hvert punkt i y-z planet, beregnet over x-dimensionen.

$$V_{y,z} = \frac{1}{N_x} \sum_{x=1}^{N_x} (D_{x,y,z} - M_{y,z})^2 \quad (2E.2)$$

**Cross-correlation** Kryds-korrelationen ved lag  $\Delta x$  mellem hver tidsserie i y-z planet og reference-tidsserien ved  $y_{\text{ref}}, z_{\text{ref}}$ .

$$C_{y,z} = \frac{1}{N_x - \Delta x} \sum_{x=1}^{N_x - \Delta x} D_{x,y,z} D_{x+\Delta x, y_{\text{ref}}, z_{\text{ref}}} \quad (2E.3)$$

Du opfordres til at benytte indbyggede funktioner til at beregne disse statistikker hvis det er muligt.

## Data plot funktion

Interface	<pre>def dataPlot(data, statistic):     # Insert your code here</pre>
Input argumenter	<p><b>data:</b> Et <math>N_y \times N_z</math> array indeholdende beregnede statistikker.</p> <p><b>statistic:</b> En streng der beskriver den type statistik som skal plottes (<b>Mean</b>, <b>Variance</b> eller <b>Cross correlation</b>.)</p>
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja (diagrammer, se specifikationer nedenfor.)

Beskrivelse	<p>Denne funktion skal lave et plot af en statistik som er beregnet af funktionen <code>dataStatistics</code>. Funktionen skal lave et 2-dimensionelt plot hvor værdierne af arrayet for y- og z-koordinaterne kan ses, fx. et contour eller surface plot.</p> <p>Diagrammet skal indeholde passende titel, beskrivende akse-betegnelser og en signaturforklaring hvor det er hensigtsmæssigt.</p>
Hoved-script	
Interface	Skal implementeres som et script.
Input argumenter	Nej.
Output argumenter	Nej.
Bruger-input	Ja (se specifikationer nedenfor.)
Skærm-output	Ja (se specifikationer nedenfor.)
Beskrivelse	<p>Brugeren af data-analyse programmet interagerer med programmet gennem hoved-scriptet. Når brugeren kører hoved-scriptet skal han/hun som minimum have følgende muligheder:</p> <ol style="list-style-type: none"> <li>1. Indlæs data.</li> <li>2. Vis statistik.</li> <li>3. Generer diagrammer.</li> <li>4. Afslut.</li> </ol> <p>Det skal være muligt for brugeren at udføre disse handlinger (se specifikationer nedenfor) i vilkårlig rækkefølge så længe han/hun ønsker det, indtil han/hun vælger at afslutte programmet. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et cool script.</p>
Fejlhåndtering	<p>Du skal teste at alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at indtaste et nyt gyldigt input.</p> <p>Det må ikke være muligt at vise statistikker eller generere diagrammer før data er indlæst. Hvis brugeren forsøger at gøre dette skal han/hun have passende feedback der forklarer at dette ikke er muligt.</p>
1. Indlæs data	Hvis brugeren vælger at indlæse data, skal du bede brugeren om at indtaste et filnavn på en data-fil. Husk at checke at filnavnet er gyldigt. Bed brugeren indtaste dimensionerne i data, og indlæs data ved hjælp af <code>dataLoad</code> funktionen.
2. Vis statistik	Hvis brugeren vælger at vise statistikker, skal du spørge brugeren hvilken statistik der skal vises, samt for hvilke koordinater statistikken ønskes vist for. Brug funktionen <code>dataStatistics</code> til at beregne den ønskede statistik og vis den på skærmen sammen med en beskrivelse af statistikken. Brugeren skal som minimum kunne få vist middelværdi og varians for et givent y- og z-koordinat.
4. Generer diagrammer	Hvis brugeren vælger at generere diagrammer, skal du spørge brugeren hvilket plot der ønskes. Herefter kaldes funktionen <code>dataPlot</code> for at vise plottet.
5. Afslut	Hvis brugeren vælger at afslutte programmet, skal hoved-scriptet stoppe.
Data fil	En stor data-fil med rigtige vindmålinger er til rådighed og kan bruges til at teste dit program. Du opfordres også til at lave din egen test-datafil for at kontrollere at dit program fungerer korrekt.



# Analyse af el-forbrug i husholdning

## Introduktion

I dette projekt skal du udvikle, teste og dokumentere et program til analyse af el-forbrug i en husholdning. Du skal implementere de funktioner og det hoved-script som er beskrevet i den følgende afsnit.

## Data load funktion

Interface `def load_measurements(filename, fmode):`  
# Insert your code here  
`return (tvec, data)`

Input argumenter `filename`: En tekst-streng som indeholder filnavnet på en datafil.  
`fmode`: En tekst-streng som specificerer, hvordan ødelagte målinger skal håndteres.

Output argumenter `tvec`: En  $N \times 6$  matrix, hvor hver række er en tidsvektor.  
`data`: En  $N \times 4$  matrix, hvor hver række er et sæt målinger.

Bruger-input Nej.

Skærm-output Advarsler (se beskrivelsen nedenfor).

Beskrivelse Funktionen skal indlæse data fra datafilen `filename`. Hver linje i datafilen indeholder de følgende felter:

`year, month, day, hour, minute, second, zone1, zone2, zone3, zone4.`

Felterne indeholder numeriske værdier og er separeret af et komma. Her er et eksempel på en linje fra en sådan datafil:

`2008,1,5,15,8,0,12.0,0.0,18.0,35.4`

Dette svarer til en måling foretaget/påbegyndt den 5. januar 2008 kl. 15:08:00, og de fire målinger er 12.0, 0.0, 18.0 og 35.4. Målingerne er i Watt-hour og svarer til el-forbruget over en periode på ét minut i en husholdning, som er inddelt i fire zoner. Zone 1 er køkkenet (hovedsageligt opvaskemaskine, oven og mikrobølgeoven), zone 2 er vaskerummet (vaskemaskine, tørretumbler, køleskab og lys), zone 3 består af en elektrisk vandvarmer og en air-conditioner og zone 4 består af alt andet elektrisk udstyr, som ikke er i zone 1-3.

Rækkerne i datafilen er i kronologisk rækkefølge, men enkelte rækker kan indeholde indeholde fejlagtige/ødelagte målinger. Disse målinger har værdien `-1`. Som eksempel svarer følgende række til en fejlagtig måling i zone 2:

`2006,2,13,9,15,0,10.0,-1,20.0,32.8`

Inputtet `fmode` specificerer, hvordan fejlagtige/ødelagte målinger skal håndteres:

fmode	Beskrivelse
'forward fill'	erstat med seneste måling uden fejl
'backward fill'	erstat med næste måling uden fejl
'drop'	fjern fejlagtige målinger

Hvis `fmode` har værdien `'forward fill'` og den første række indeholder en måling med fejl eller hvis `fmode` har værdien `'backward fill'` og den sidste række indeholder en måling med fejl, så fjern alle rækker med fejlagtige målinger og udskriv en advarsel på skærmen.

Data i filen skal gemmes i en  $N \times 4$  matrix kaldet `data`, hvor søjle  $i$  svarer til målinger i zone  $i$ , og hvor  $N$  enten er antallet af rækker i filen (hvis `fmode` har værdien `'forward fill'` eller `'backward fill'`) eller antallet af målinger uden fejl (hvis `fmode` har værdien `'drop'`). Tidsvektorerne skal gemmes i en  $N \times 6$  matrix kaldet `tvec`, hvor søjle 1-6 svarer til hhv. `year`, `month`, `day`, `hour`, `minute` og `second`.

## Data aggregeringsfunktion

Interface	<pre>def aggregate_measurements(tvec, data, period):     # Insert your code here     return (tvec_a, data_a)</pre>
Input argumenter	<p><code>tvec</code>: En <math>N \times 6</math> matrix, hvor hver række er en tidsvektor.</p> <p><code>data</code>: En <math>N \times 4</math> matrix, hvor hver række er et sæt målinger.</p> <p><code>period</code>: En tekst-streng som specificerer, hvordan målinger aggregeres.</p>
Output argumenter	<p><code>tvec_a</code>: En <math>M \times 6</math> matrix, hvor hver række er en tidsvektor.</p> <p><code>data_a</code>: En <math>M \times 4</math> matrix med aggregerede målinger.</p>
Bruger-input	Nej.
Skærm-output	Nej.
Beskrivelse	<p>Funktionens input er en matrix med tidsvektorer <code>tvec</code>, en matrix med målinger <code>data</code> og en tekst-streng <code>period</code>, som specificerer, hvordan målingerne skal aggregeres<sup>1</sup>. Inputtet <code>period</code> kan have følgende værdier:</p>

period	Beskrivelse
'hour'	forbrug pr. time
'day'	dagligt forbrug
'month'	månedligt forbrug
'hour of the day'	gennemsnitligt forbrug efter tid på dagen (pr. time)

Funktionens output `data_a` og `tvec_a` indeholder de aggregerede målinger og de tilhørende tidsvektorer. Hvis `period` har værdien `'hour'`/`'day'`/`'month'`, så kombineres (summeres) alle målinger inden for samme time/dag/måned, og  $M$  er antallet af timer/dage/måneder i løbet af hvilke, der er foretaget én eller flere målinger. Rækkerne af `tvec_a` identificerer starten af de  $M$  måleperioder. Hvis `period` har værdien `'hour of the day'`, så er  $M$  lig 24 svarende til de 24 tidsintervaller 00:00-01:00, 01:00-02:00, ..., 23:00-00:00, den aggregerede data matrix `data_a` indeholder det gennemsnitlige forbrug (over alle dage) i hvert af disse tidsintervaller, og `tvec_a` identificerer starten på tidsintervallerne.

## Statistik-funktion

Interface	<pre>def print_statistics(tvec, data):     # Insert your code here     return</pre>
Input argumenter	<p><code>tvec</code>: En <math>N \times 6</math> matrix, hvor hver række er en tidsvektor.</p> <p><code>data</code>: En <math>N \times 4</math> matrix, hvor hver række er et sæt målinger.</p>
Output argumenter	Nej.
Bruger-input	Nej.
Skærm-output	Ja, en tabel med statistikker (se beskrivelsen nedenfor).

<sup>1</sup>At aggregere betyder at "samle sammen" eller at "danne summen af en mængde ens enheder."

## Beskrivelse

Funktionen udskriver en tabel i følgende form

Zone	Minimum	1. kvartil	2. kvartil	3. kvartil	Maximum
1					
2					
3					
4					
Alle					

med følgende statistikker/percentiler: Minimum, 1. kvartil, 2. kvartil (median), 3. kvartil og maximum.

Du opfordres til at benytte indbyggede funktioner til at beregne disse statistikker.

## Hoved-script

### Header

Skal implementeres som et script.

### Bruger-input

Ja (se specifikationer nedenfor.)

### Skærm-output

Ja (se specifikationer nedenfor.)

### Beskrivelse

Brugeren af programmet interagerer med programmet gennem hoved-scriptet. Når brugeren kører hoved-scriptet skal han/hun som minimum have følgende muligheder:

1. Indlæs data.
2. Aggregér data.
3. Vis statistik.
4. Visualisér el-forbrug.
5. Afslut.

Det skal være muligt for brugeren at udføre disse handlinger (se specifikationer nedenfor) i vilkårlig rækkefølge indtil han/hun vælger at afslutte programmet. Detaljerne vedrørende hvordan hoved-scriptet skal implementeres skal du selv bestemme. Det er et krav at programmet er interaktivt, og du skal tilstræbe at gøre programmet brugervenligt ved at give fornuftige valgmuligheder. Tænk på hvad du ville forvente hvis du skulle bruge sådan et script, og på hvad du synes kunne være sjovt. Leg med implementationen og lav et cool script.

### Fejlhåndtering

Du skal teste at alle input givet af brugeren er gyldige. Hvis brugeren giver et ugyldigt input, skal du give informativ feedback til brugeren og gøre det muligt at indtaste et nyt gyldigt input.

Det må ikke være muligt at aggregere data, at vise statistikker eller generere diagrammer før data er indlæst. Hvis brugeren forsøger at gøre dette skal han/hun have passende feedback, der forklarer, at dette ikke er muligt.

### 1. Indlæs data

Hvis brugeren vælger at indlæse data, skal du bede brugeren om at indtaste et filnavn på en data-fil. Husk at checke at filnavnet er gyldigt. Spørg bruger om, hvordan han/hun ønsker at håndtere fejlagtige/ødelagte målinger og indlæs data ved hjælp af `load_measurements` funktionen. Som minimum skal brugeren kunne vælge at håndtere fejlagtige/ødelagte målinger på følgende måder:

1. Fill forward (erstat fejlagtige/ødelagt måling med seneste måling uden fejl)
2. Fill backward (erstat fejlagtige/ødelagt måling med næste måling unde fejl)
3. Fjern fejlagtige/ødelagte målinger

### 2. Aggregér data

Hvis brugeren vælger at aggregere data, skal han/hun have mulighed for at specificere hvordan data skal aggregeres. Som minimum skal brugeren have følgende muligheder:

1. Forbrug pr. minut (ingen aggregering)
2. Forbrug pr. time
3. Forbrug pr. dag
4. Forbrug pr. måned
5. Forbrug efter tid på dagen (timegennemsnit)

De aggregerede data skal bruges efterfølgende, når brugeren vælger af beregne statistikker eller at visualisere data.

3. Vis statistik
- Hvis brugeren vælger at vise statistikker, så brug funktionen `print_statistics` til at beregne og udskrive en tabel med percentiler på skærmen. Output til skærmen skal også inkludere en beskrivelse af tidsskala (forbrug pr. minut, time, dag, etc.) samt måleenhed (f.eks. Watt-hour eller Kilowatt-hour).
4. Visualisér
- Hvis brugeren vælger at visualisere el-forbruget, spørg brugeren om han/hun ønsker at plotte forbruget for hver af de fire zoner eller det samlede forbrug (alle zoner). Vis et plot med passende akser (tid på x-aksen og forbrug på y-aksen), labeler og en titel. Brug et søjlediagram hvis de aggregerede data består af færre end 25 målinger.
5. Afslut
- Stop hoved-scriptet hvis brugeren vælger at afslutte programmet.
- Data fil
- En stor datafil med målinger fra 2008 er til rådighed og kan bruges i forbindelse med tests. Du opfordres endvidere til at lave din egne test-datafiler for at kontrollere at dit program fungerer korrekt.