



TDT4110 Informasjonsteknologi grunnkurs:

Tema: Filer

Utgave 3: Kap. 6

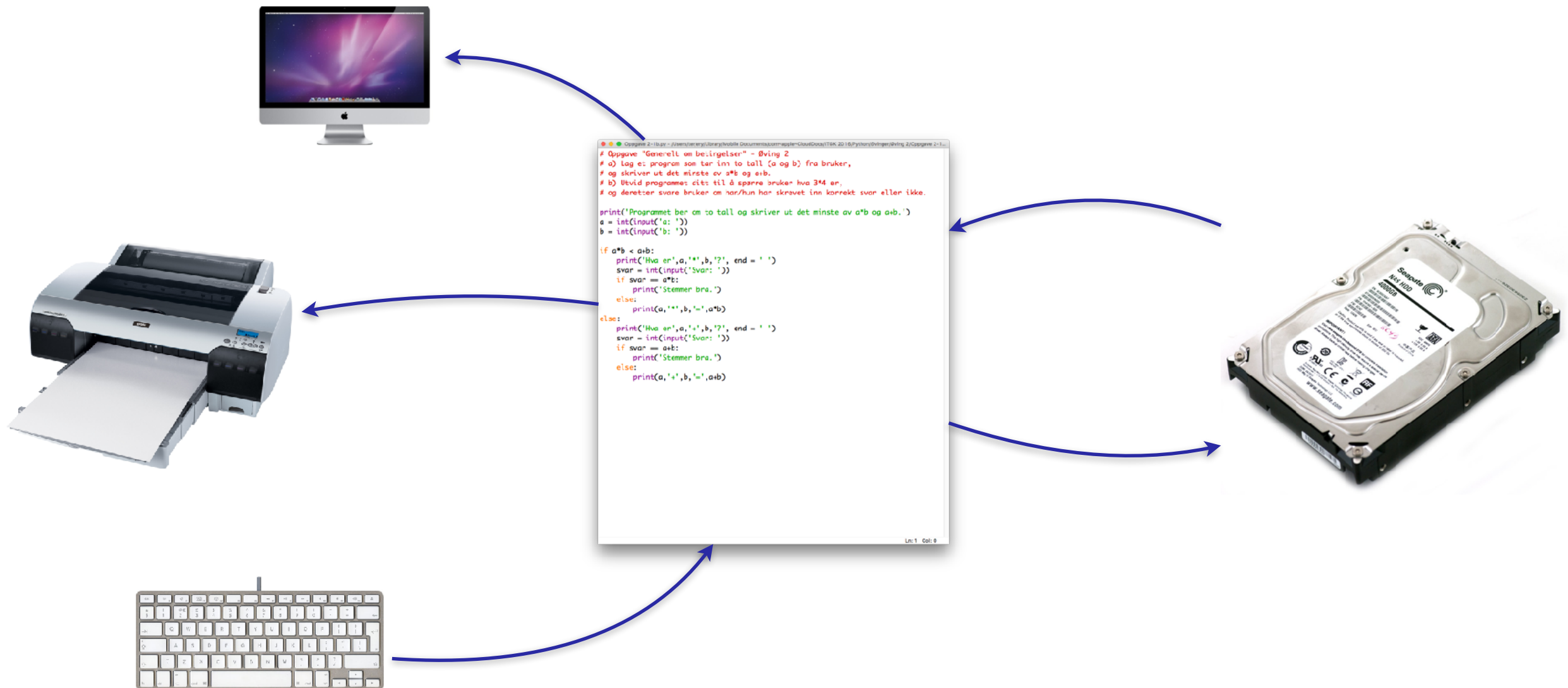
Terje Rydland - IDI/NTNU

Læringsmål og pensum

- Mål
 - Lære bruk av inn- og ut-operasjoner i Python
 - Lære å kunne bruke lesing og skriving til fil
- Pensum
 - Starting out with Python, Chapter 6 "Files and Exceptions"
 - Kap 9 - serializing av objekter

Inn- og utoperasjoner


- I programmering brukes begrepet inn/ut-operasjoner, I/O eller IO når programmet leser eller skriver innholdet av variabler (data) til omverdenen.
- Omverdenen kan være skjermen (ut), tastaturet (inn), skriver (ut), eller til filer (inn og ut)



Inn- og utoperasjoner

- Bruk av tastatur og skjerm er ok for små datamengder, men upraktisk for store datamengder:
 - Eks. værdata, modeller av skip eller konstruksjon, osv. som kan bestå av milliarder av bytes (gigabytes) eller mer.
 - Eks. data fra seismiske undersøkelser kan være i størrelsesorden petabytes (en million milliarder bytes).

```
1001,$GPGGA,165443.06,3802.159312,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*49
1002,$GPGGA,165444.27,3802.160934,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*49
1003,$GPGGA,165445.29,3802.162556,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4C
1004,$GPGGA,165446.86,3802.164178,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*44
1005,$GPGGA,165446.85,3802.165801,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*41
1006,$GPGGA,165448.16,3802.169045,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*41
1007,$GPGGA,165449.11,3802.170668,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*46
1008,$GPGGA,165450.34,3802.172296,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*48
1009,$GPGGA,165451.39,3802.173912,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*44
1010,$GPGGA,165452.87,3802.175535,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4D
1011,$GPGGA,165501.27,3802.191758,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*43
1012,$GPGGA,165502.48,3802.193386,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4A
1013,$GPGGA,165503.51,3802.195003,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4D
1014,$GPGGA,165504.51,3802.196625,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4B
1015,$GPGGA,165505.27,3802.198247,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*45
1016,$GPGGA,165506.11,3802.199876,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*4C
1017,$GPGGA,165507.48,3802.201492,N,12300,W,2,4,4.69,23,M,-28,M,3.65,0000*43
```



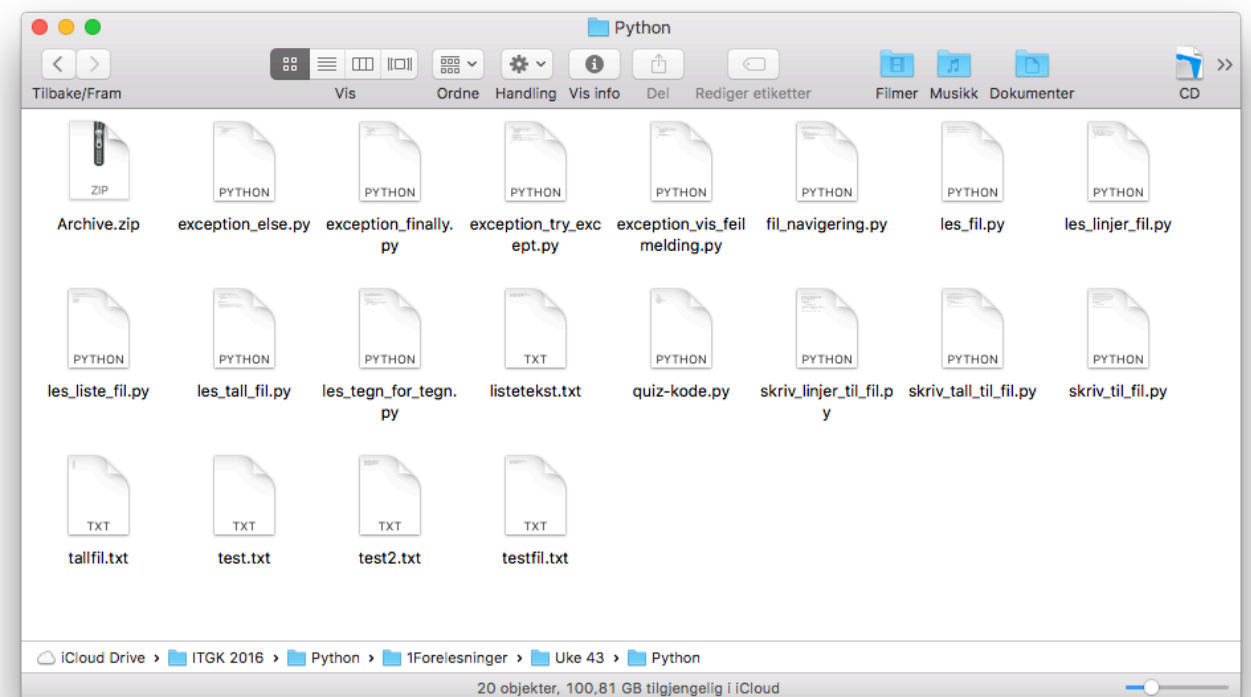
```
# Oppgave "Generelt om belegg" - Øving 2
# a) lag et program som tar inn to tall {a og b} fra bruker,
# og skriver ut det minste av a*b og a+b.
# b) Bevid programmet ditt til å spørre bruker hva 3+4 er.
# og deretter svare bruker om når/hun har skrevet inn korrekt svar eller ikke.

print('Programmet ber om to tall og skriver ut det minste av a*b og a+b.')
a = int(input('a: '))
b = int(input('b: '))

if a*b < a+b:
    print('Hva er',a,'+',b,'?', end = ' ')
    svar = int(input('Svar: '))
    if svar == a*b:
        print('Stemmer bra.')
    else:
        print(a,'+',b,'=',a+b)
else:
    print('Hva er',a,'+',b,'?', end = ' ')
    svar = int(input('Svar: '))
    if svar == a+b:
        print('Stemmer bra.')
    else:
        print(a,'+',b,'=',a+b)
```

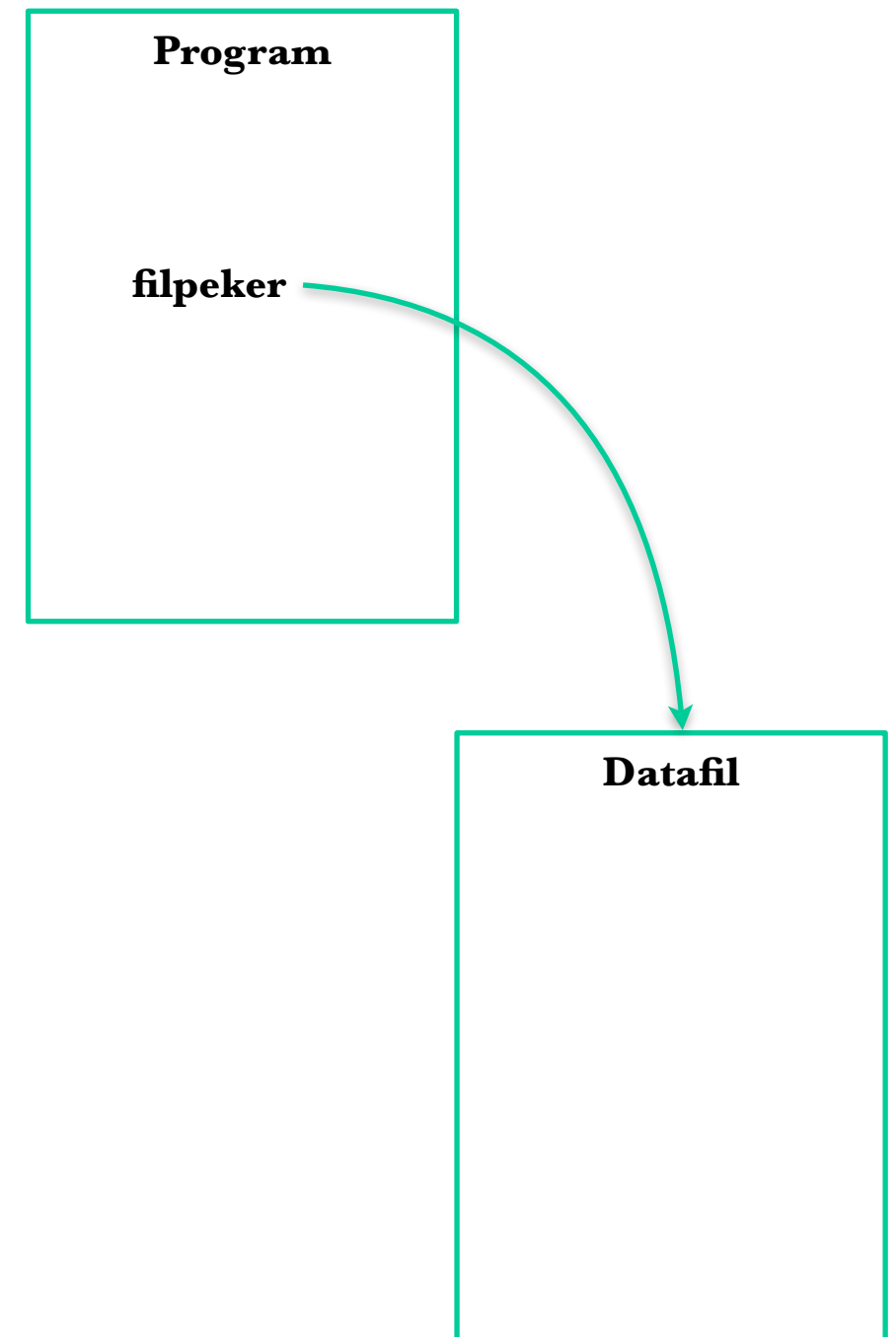
Inn- og utoperasjoner (2)

- Ved kun bruk av I/O-operasjoner mot tastatur og skjerm, vil all data forsvinne etter at vi har avsluttet Python.
- Å lagre filer som inneholder data til sekundærminne (for eksempel harddisk)
 - huske data mellom hver gang man kjører/avslutter programmer.
 - Vi har allerede lagret programmer i Python-filer som vi kan gjenbruke gang etter gang.
- Tema:
 - gjøre det samme med dataene som brukes i programmer (innholdet i variabler).



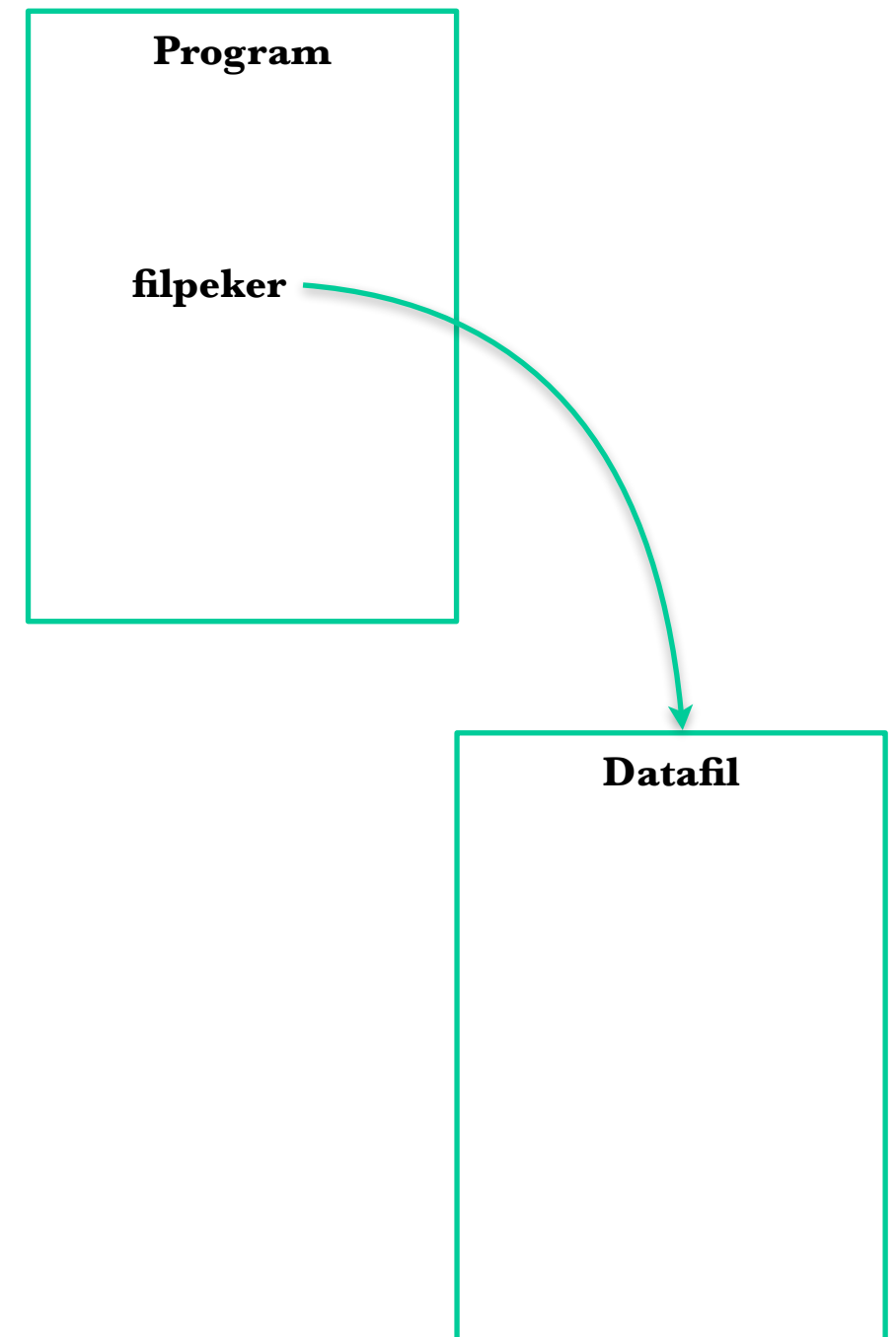
Hva er en fil i Python?

- I et Python-program, blir ei fil representert som en verdi av typen file.
 - **Verdien til en fil representerer ikke innholdet i fila, men en referanse eller portal til dataene.**
- En fil som er lagret på en harddisk kan inneholde mer data enn du kan ha i minnet på en gang.
 - Må ha referanse til fila og kan navigere deg igjennom en fil for å skrive eller hente data.
 - Man henter ofte in bare deler av ei fil og lagrer dette i variabler.



2 typer filer

- Tekstfiler
 - Mange datatyper kan lagres som tekst.
 - Tekster, tall etc...
 - Kan da åpne filen i en teksteditor og redigere innholdet
- Binærfiler
 - Men hva med dictionaries, 2D-lister etc...
 - Hvordan ta vare på strukturen i dataene?
 - Enklest med binærfiler
 - Kan da ikke se dataene fra filen, men må lese filen med et program.

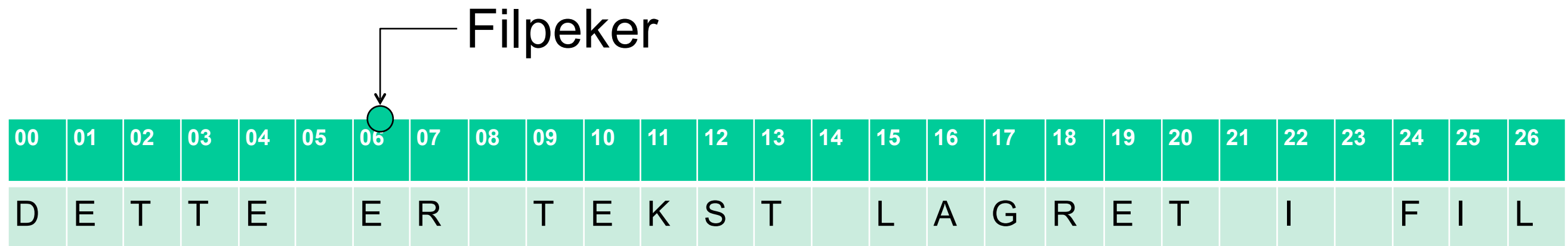


Kapittel 6

Tekstfiler

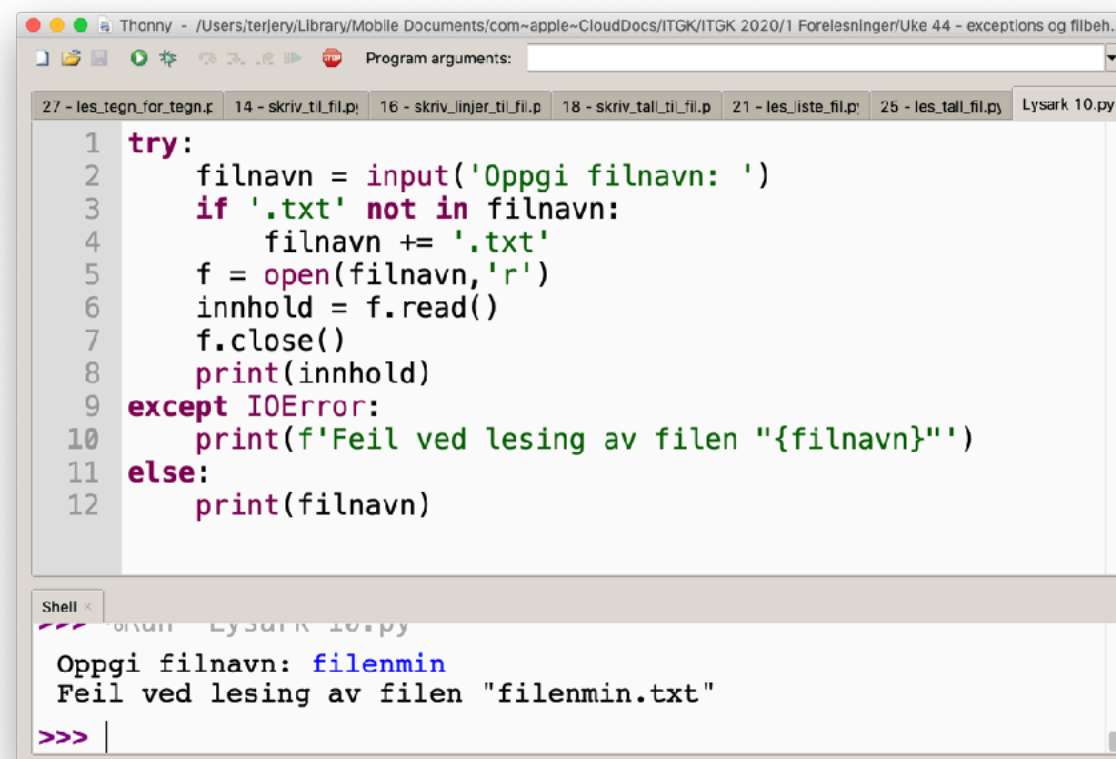
Hva er en fil i Python? (2)

- I en fil lagres data etter hverandre (sekvensielt).
- Etter hvert som man skriver data til en fil, vil en filpeker holde orden på hvor langt man har kommet i fila.
- Filpekeren kan også flyttes ved kommandoer.



Proessen for filoperasjoner i Python

- Filoperasjoner i Python gjøres i tre steg:
 1. **Fila åpnes:** Etablerer en link mellom filvariabelen og informasjonen lagret på disken.
 - Filreferansen som peker til den fysiske fila på disk blir lagret i en variabel.
 - Alle operasjoner mot fila må bruke denne variabelen som filreferanse.
 2. **Verdier leses fra og skrives til fila** ved hjelp av filreferansen:
 - Lesing: Data lagret i fil leses inn og lagres i variabler.
 - Skrivning: Data lagret i variabler skrives som data lagret i en fil.
 3. **Fila stenges.**
 - Etter at fila er stengt, kan man ikke lese eller skrive til fila.



```
1 try:
2     filnavn = input('Oppgi filnavn: ')
3     if '.txt' not in filnavn:
4         filnavn += '.txt'
5     f = open(filnavn, 'r')
6     innhold = f.read()
7     f.close()
8     print(innhold)
9 except IOError:
10    print(f'Feil ved lesing av filen "{filnavn}"')
11 else:
12    print(filnavn)
```

```
Oppgi filnavn: filenmin
Feil ved lesing av filen "filenmin.txt"
```

Pass på at det er en tekstfil

- Det er ikke alltid at tekstredigeringsprogrammene lagrer ting som ren tekst

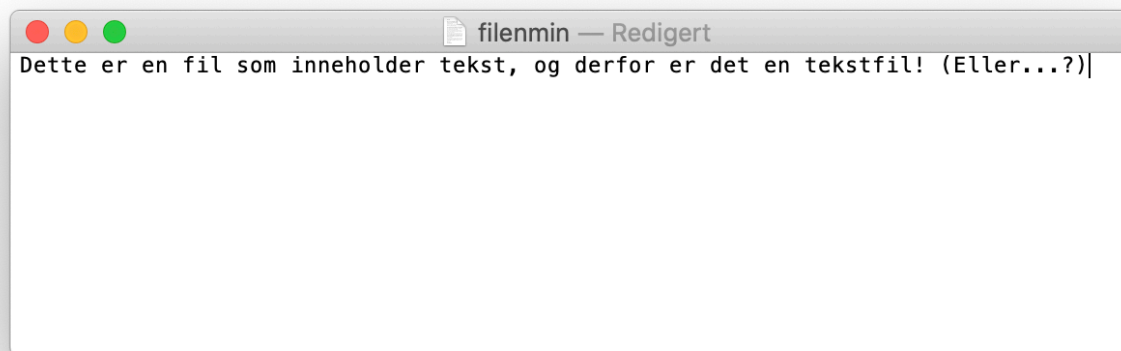


TextEdit



txt

Pass på at det er en tekstfil



Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på lysark/Lysark 12.py @ 10 : 51

Program arguments:

27 - les_tegn_for_tegn.py × 14 - skriv_til_fil.py × 16 - skriv_linjer_til_fil.py × 18 - skriv_tall_til_fil.py × 21 - les_liste_fil.py × 25 - les_tall_fil.py × Lysark 10.py × Lysark 12.py ×

```
1 try:
2     filnavn = input('Oppgi filnavn: ')
3     if '.txt' not in filnavn:
4         filnavn += '.txt'
5     f = open(filnavn, 'r')
6     innhold = f.read()
7     f.close()
8     print(innhold)
9 except IOError:
10    print(f'Feil ved lesing av filen "{filnavn}")
```

Shell ×

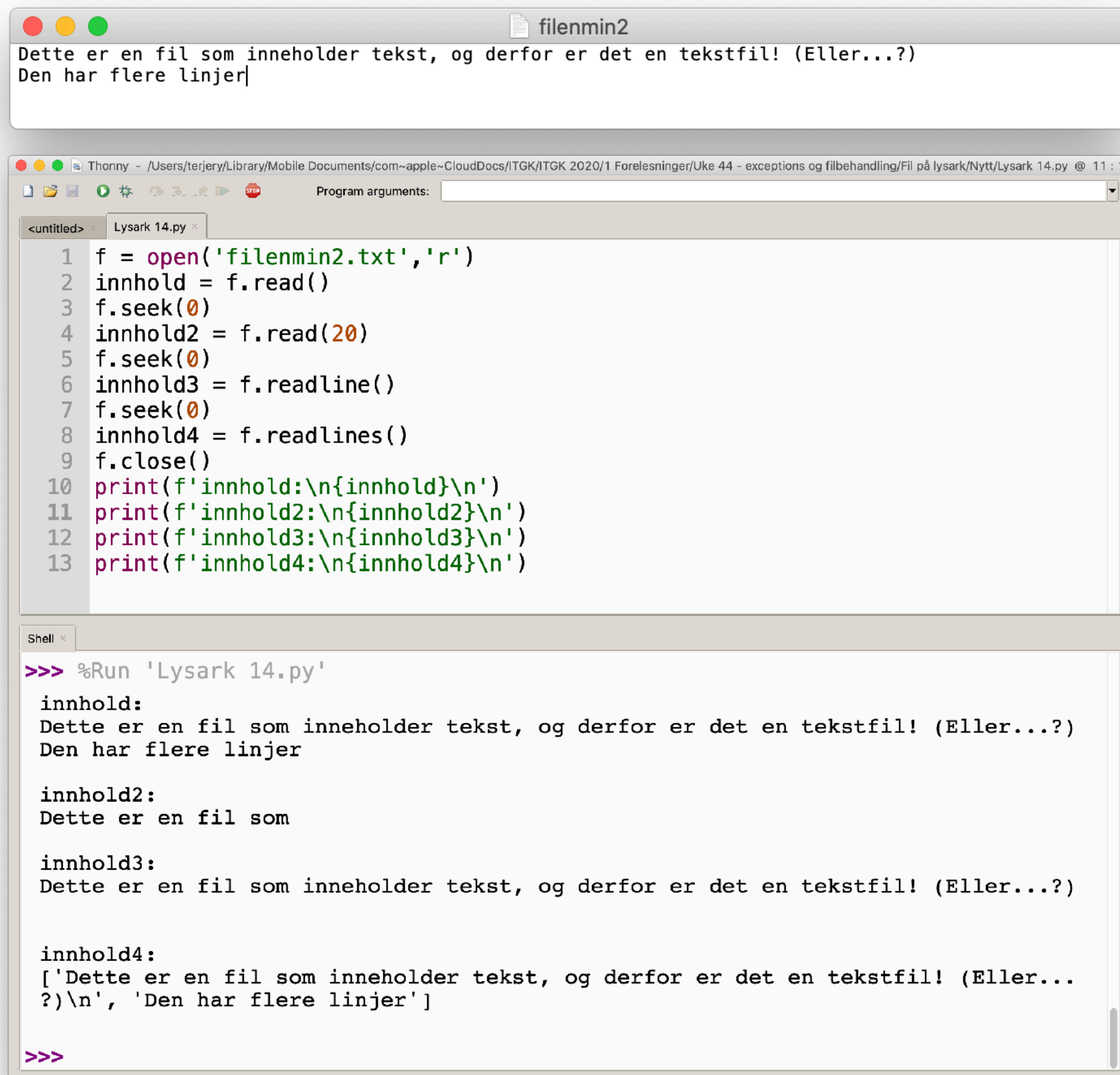
```
>>> %Run 'Lysark 12.py'
Oppgi filnavn: filenmin
Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...?)
>>> |
```

Håndtering av tekstfiler i Python

Filkommandoer	Forklaring
<code>f = open('filnavn')</code>	Åpner ei fil, returnerer filreferanse
<code>f = open('filnavn', 'tilgang')</code>	Åpner ei fil, med spesifisert tilgang. F.eks. 'w' åpner ei fil for skriving (se neste side)
<code>f.read()</code>	Returnerer hele innholdet av fila
<code>f.read(n)</code>	Returnerer n tegn av innholdet
<code>f.readline()</code>	Returnerer neste linje (før \n)
<code>f.readlines()</code>	Returnerer hele innholdet av fila som ei liste
<code>f.write(s)</code>	Skriver strengen s til fil
<code>f.writelines(liste)</code>	Skriver innholdet av liste av strenger til fil
<code>f.seek(offset, fra_hvor)</code>	Forflytter filpekeren (index) i fila
<code>f.tell()</code>	Returnerer posisjon til filpekeren i fila
<code>f.close()</code>	Stenger fila

f representerer variabelen som tar vare på filpekeren

Eksempler



The screenshot shows a Thonny IDE window with a file named 'filenmin2' open. The file contains the text: 'Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...?)' and 'Den har flere linjer'. Below the file editor, the Python script 'Lysark 14.py' is shown. The script opens the file, reads its content in four different ways (f.read(), f.read(20), f.readline(), f.readlines()), and prints each result. The output of the script is shown in the Shell window, demonstrating the effect of f.seek(0) on the file pointer.

```
filenmin2
Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...?)
Den har flere linjer

Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på lysark/Nytt/Lysark 14.py @ 11 : 1
Program arguments:

<untitled> x Lysark 14.py x
1 f = open('filenmin2.txt','r')
2 innhold = f.read()
3 f.seek(0)
4 innhold2 = f.read(20)
5 f.seek(0)
6 innhold3 = f.readline()
7 f.seek(0)
8 innhold4 = f.readlines()
9 f.close()
10 print(f'innhold:\n{innhold}\n')
11 print(f'innhold2:\n{innhold2}\n')
12 print(f'innhold3:\n{innhold3}\n')
13 print(f'innhold4:\n{innhold4}\n')

Shell x
>>> %Run 'Lysark 14.py'
innhold:
Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...?)
Den har flere linjer

innhold2:
Dette er en fil som

innhold3:
Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...?)

innhold4:
['Dette er en fil som inneholder tekst, og derfor er det en tekstfil! (Eller...
?)\n', 'Den har flere linjer']

>>>
```

`f.seek(0)`
flytter filpekeren
fremst i fila

Åpning av filer

- For å lagre data til fil eller hente data fra fil, må man først åpne fila ved hjelp av open:

```
variabel = open('filnavn' , 'tilgangstype')
```

- Forklaring:
 - variabel: Får en referanse som peker til fila med angitt filnavn
 - filnavn: Angir et stinavn og filnavn til fila som skal åpnes
 - tilgangstype: Angir en kode for typen filoverførings som skal gjøres
- Eks:

```
f = open('datafil.txt','r') # Fil for lesing
```

```
f = open('datafil.txt','w') # Fil for skriving
```

```
f = open('datafil.txt','a') # Fil for oppdatering
```


Tilgangstyper for open

- Vi har følgende tilgangstyper for **open**:

Streng	Filoperasjon
'r'	Åpne en fil for <i>lesing</i>
'w'	Åpne en fil for <i>skrivning</i> og fjern evt. gammelt innhold . Lag ny fil hvis den ikke finnes.
'a'	Åpne en fil for å <i>legge til</i> nye data på slutten (logging). Lag ny fil hvis den ikke finnes.
'r+'	Åpne en fil som finnes fra før for <i>lesing og skrivning</i>
'w+'	Åpne en fil for <i>lesing og skrivning</i> og fjern evt. gammelt innhold . Lag ny fil hvis den ikke finnes.
'a+'	Åpne en fil for å <i>lese og legge til</i> nye data på slutten (logging). Lag ny fil hvis den ikke finnes.

```
f = open('datafil.txt', 'r') # Fil for lesing
```

```
f = open('datafil.txt', 'w') # Fil for skrivning
```

```
f = open('datafil.txt', 'a') # Fil for oppdatering
```


Stenging av filer

- Når et program åpner en fil for lesing, vil operativsystemet vite at et program leser fila.
- Når et program åpner en fil for skriving (endring), låser operativsystemet fila slik at ingen andre får lov til å endre på fila samtidig.
- Etter programmet er ferdig med å lese fra og skrive til fila, må fila lukkes for å si ifra at nå er den fritt vilt:

```
filvariabel.close() # Stenger fila
```

open() er en funksjon (skal tilordne en filreferanse til en ny filvariabel)

```
f = open(filnavn,'r')
```

close() er en metode (anvendes på eksisterende filvariabel)

```
f.close()
```

Lese tall fra tekstfil

```
tallfil
2
4
65
12
99
86
```

Med bruk av `f.readlines()`, blir "nonprinting characters" som `\n` med.

```
Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - ex...
Program arguments:

<untitled> Lysark 14.py Lysark 19.py Lysark 19-2.py
1 f = open('tallfil.txt', 'r')
2 innhold = f.readlines()
3 f.close()
4 print(f'innhold:\n{innhold}\n')
5 # Konverter til tall
6 tall = []
7 for i in innhold:
8     tall.append(int(i.strip()))
9 print(tall)
10 # Eller:
11 for i in range(len(innhold)):
12     innhold[i] = int(innhold[i].strip())
13 print(innhold)

Shell x
>>> %Run 'Lysark 19.py'

innhold:
['2\n', '4\n', '65\n', '12\n', '99\n', '86']

[2, 4, 65, 12, 99, 86]
[2, 4, 65, 12, 99, 86]
>>> |
```

Med bruk av `f.readline()`, blir "nonprinting characters" som `\n` **ikke** med.

```
Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesn...
Program arguments:

<untitled> Lysark 14.py Lysark 19.py Lysark 19-2.py
1 f = open('tallfil.txt', 'r')
2 innhold = []
3 linje = f.readline()
4 while linje:
5     innhold.append(int(linje))
6     linje = f.readline()
7 f.close()
8 print(f'innhold:\n{innhold}\n')
9

Shell x
>>> %Run 'Lysark 19-2.py'

innhold:
[2, 4, 65, 12, 99, 86]
>>>
```

Skrive strenger til fil

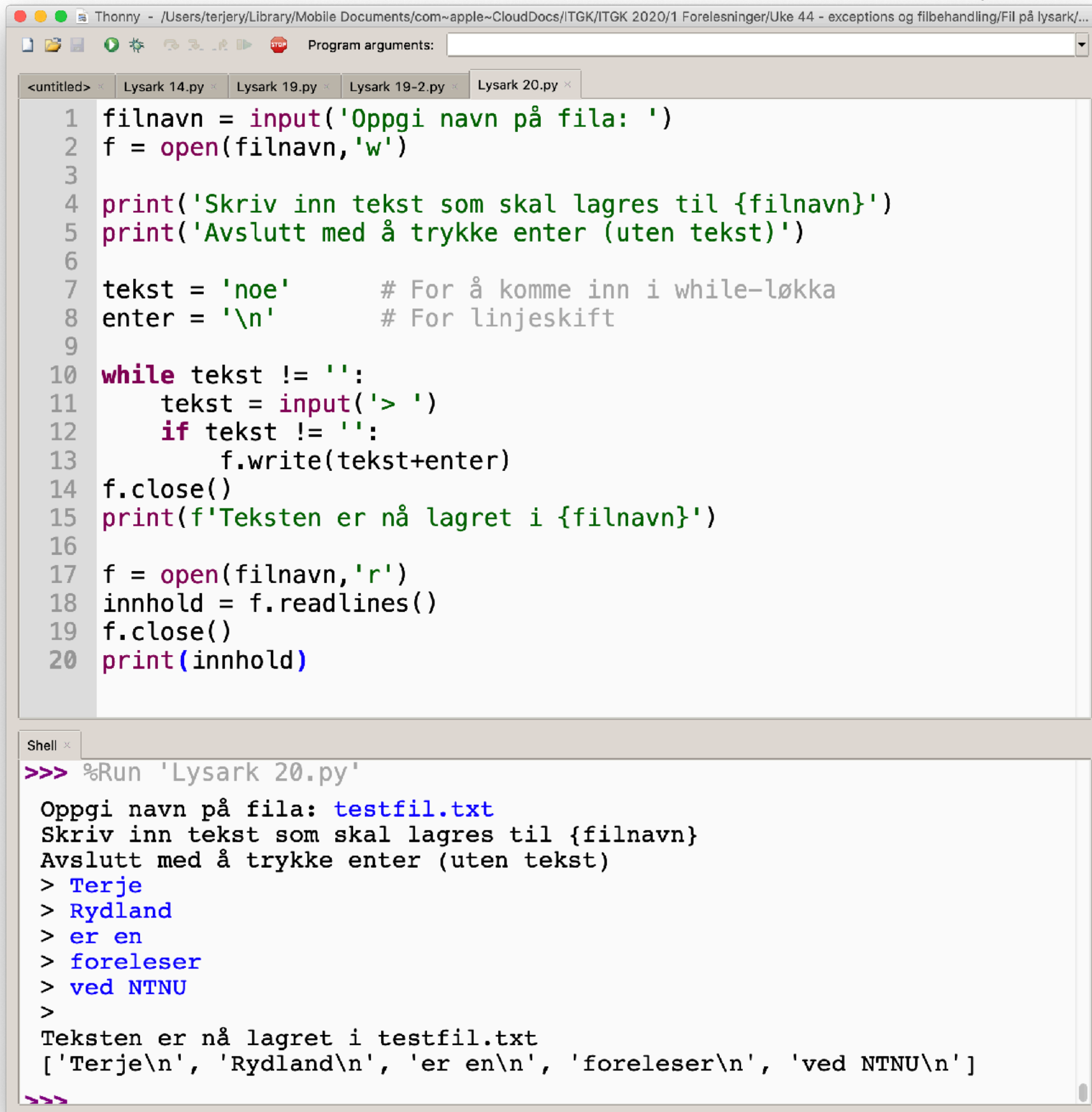
- For å skrive data til fil i Python brukes følgende:

`f.write(s)` # Skriver strengen s til fil med ref f

`f.writelines(liste)` # Skriver en liste av strenger til fil

- Vi ser på et program der brukeren kan skrive inn navnet på fila hvor tekst brukeren skriver inn blir lagret.
- Brukeren avslutter skriving med linjeskift (uten tekst)
- Vi prøver...

skriv_til_fil.py



The screenshot shows a Thonny IDE window with the following components:

- File Explorer:** Shows the file path `/Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på lysark/...`.
- Program arguments:** An empty text box.
- Code Editor:** Contains the Python script `skriv_til_fil.py` with line numbers 1 to 20. The script prompts the user for a filename, writes text to it, and then reads the content back.
- Shell:** Shows the execution of `Lysark 20.py`. It displays the prompts and user input, followed by the printed output of the file's contents.

```
1 filnavn = input('Oppgi navn på fila: ')
2 f = open(filnavn, 'w')
3
4 print('Skriv inn tekst som skal lagres til {filnavn}')
5 print('Avslutt med å trykke enter (uten tekst)')
6
7 tekst = 'noe'          # For å komme inn i while-løkke
8 enter = '\n'          # For linjeskift
9
10 while tekst != '':
11     tekst = input('> ')
12     if tekst != '':
13         f.write(tekst+enter)
14 f.close()
15 print(f'Teksten er nå lagret i {filnavn}')
16
17 f = open(filnavn, 'r')
18 innhold = f.readlines()
19 f.close()
20 print(innhold)
```

Shell

```
>>> %Run 'Lysark 20.py'
Oppgi navn på fila: testfil.txt
Skriv inn tekst som skal lagres til {filnavn}
Avslutt med å trykke enter (uten tekst)
> Terje
> Rydland
> er en
> foreleser
> ved NTNU
>
Teksten er nå lagret i testfil.txt
['Terje\n', 'Rydland\n', 'er en\n', 'foreleser\n', 'ved NTNU\n']
>>>
```



The screenshot shows a text file named `testfil.txt` with the following content:

```
Terje
Rydland
er en
foreleser
ved NTNU
```

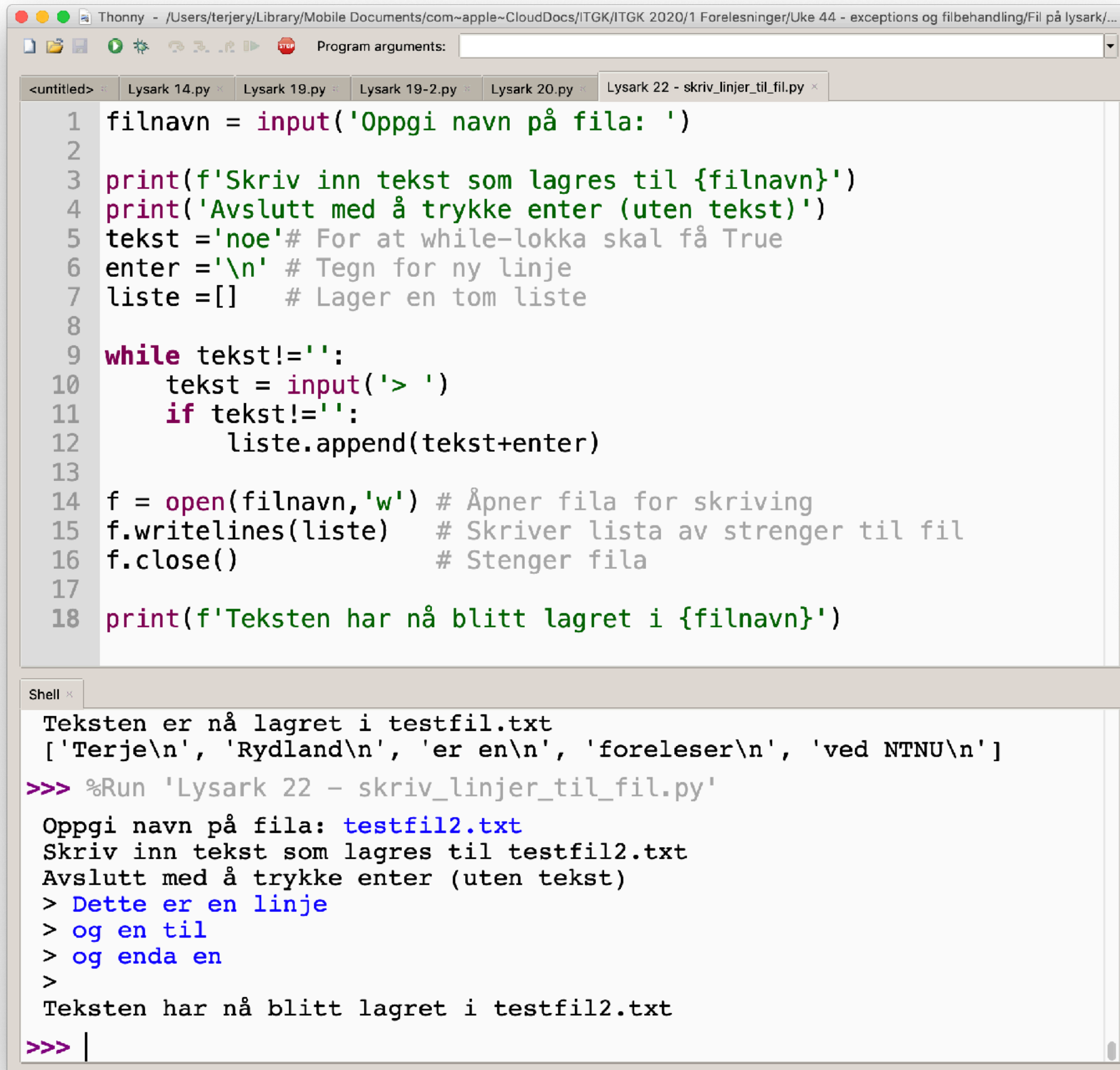
Skrive liste av strenger til fil

- En liste av strenger kan skrives til fil ved hjelp av:

```
filvariabel.writelines(liste)
```

- Vi lager en variant av skriv_til_fil.py der vi bruker liste. Må da:
 - Opprette en tom liste
 - append tekst-strengen brukeren skriver inn til lista
 - Skrive lista til fil
- Vi prøver...

skriv_linjer_til_fil.py



The screenshot shows a Thonny IDE window with a Python script named `skriv_linjer_til_fil.py` and a terminal window below it.

Python Script:

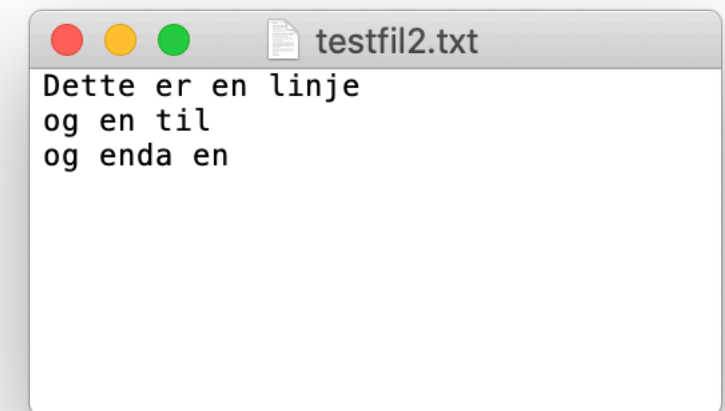
```

1 filnavn = input('Oppgi navn på fila: ')
2
3 print(f'Skriv inn tekst som lagres til {filnavn}')
4 print('Avslutt med å trykke enter (uten tekst)')
5 tekst = 'noe' # For at while-lokka skal få True
6 enter = '\n' # Tegn for ny linje
7 liste = [] # Lager en tom liste
8
9 while tekst != '':
10     tekst = input('> ')
11     if tekst != '':
12         liste.append(tekst+enter)
13
14 f = open(filnavn, 'w') # Åpner fila for skriving
15 f.writelines(liste) # Skriver lista av strenger til fil
16 f.close() # Stenger fila
17
18 print(f'Teksten har nå blitt lagret i {filnavn}')
```

Shell Output:

```

Teksten er nå lagret i testfil.txt
['Terje\n', 'Rydland\n', 'er en\n', 'foreleser\n', 'ved NTNU\n']
>>> %Run 'Lysark 22 - skriv_linjer_til_fil.py'
Oppgi navn på fila: testfil2.txt
Skriv inn tekst som lagres til testfil2.txt
Avslutt med å trykke enter (uten tekst)
> Dette er en linje
> og en til
> og enda en
>
Teksten har nå blitt lagret i testfil2.txt
>>> |
```



The screenshot shows a text file named `testfil2.txt` with the following content:

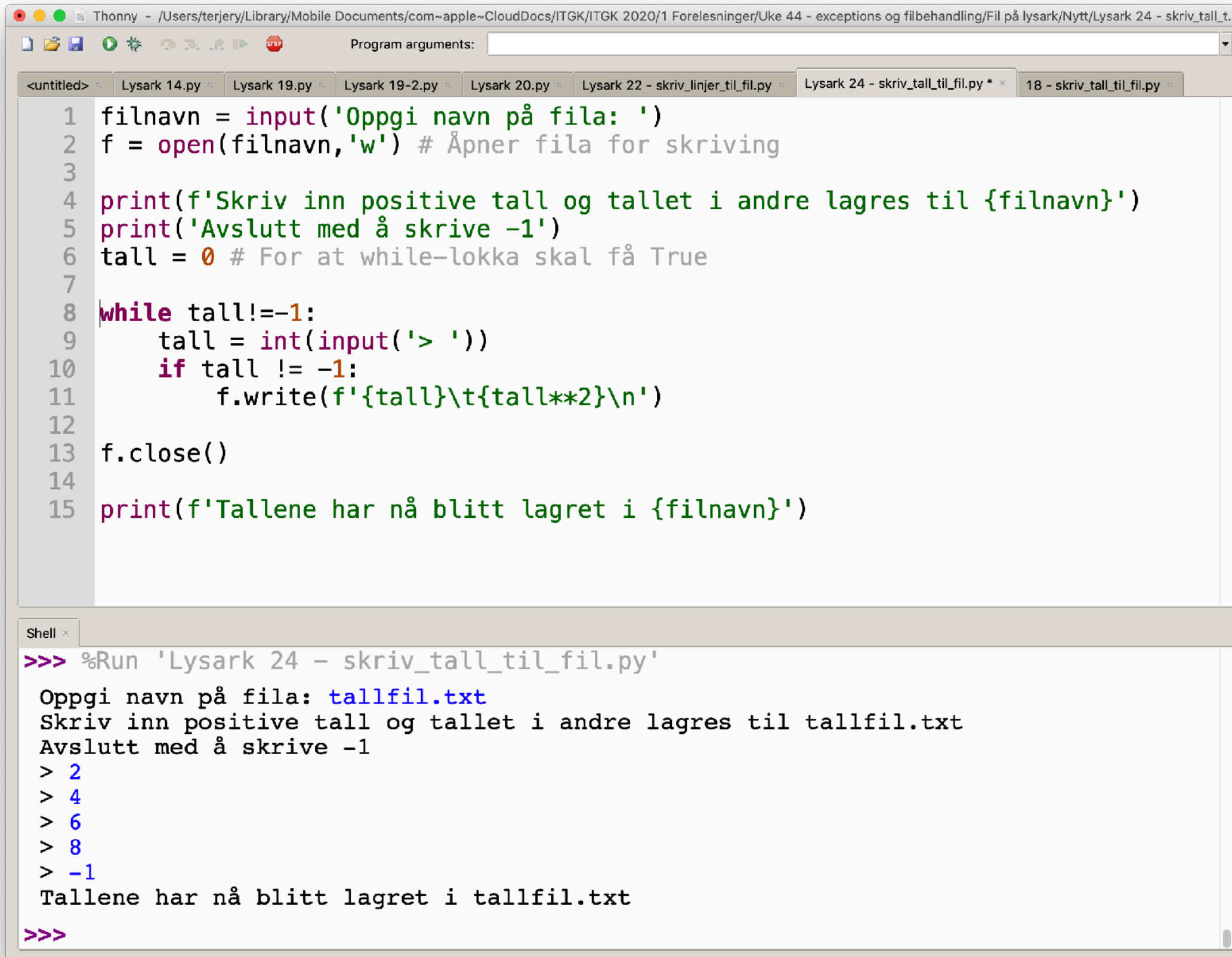
```

Dette er en linje
og en til
og enda en
```


Skrive annen data til fil

- Merk at det er **kun strenger som skrives til fil**.
- For å skrive annen data en strenger, så må man konvertere dette til streng:
 - Kan bruke `str(variabel)`
 - Best å bruke formatted text
- Vi ser på et eksempel for å lagre masse tall en bruker skriver inn til en fil med filnavn spesifisert av brukeren.
- Vi ser på et eksempel...

Skriv_tall_til_fil.py



The screenshot shows the Thonny Python IDE interface. The top window displays the script `skriv_tall_til_fil.py` with the following code:

```
1 filnavn = input('Oppgi navn på fila: ')\n2 f = open(filnavn, 'w') # Åpner fila for skriving\n3\n4 print(f'Skriv inn positive tall og tallet i andre lagres til {filnavn}')\n5 print('Avslutt med å skrive -1')\n6 tall = 0 # For at while-lokka skal få True\n7\n8 while tall != -1:\n9     tall = int(input('> '))\n10     if tall != -1:\n11         f.write(f'{tall}\\t{tall**2}\\n')\n12\n13 f.close()\n14\n15 print(f'Tallene har nå blitt lagret i {filnavn}')
```

The bottom window shows the shell output:

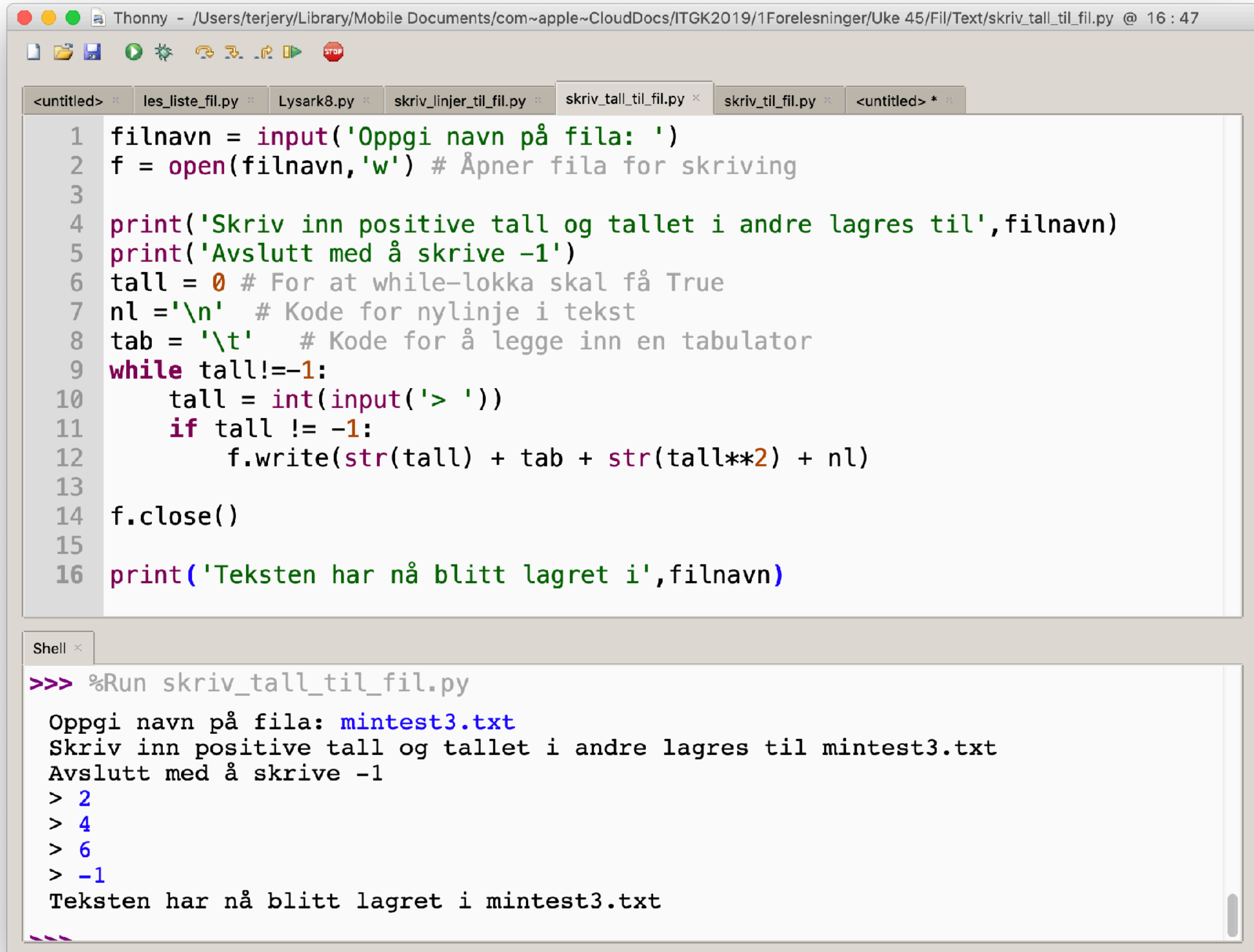
```
>>> %Run 'Lysark 24 - skriv_tall_til_fil.py'\nOppgi navn på fila: tallfil.txt\nSkriv inn positive tall og tallet i andre lagres til tallfil.txt\nAvslutt med å skrive -1\n> 2\n> 4\n> 6\n> 8\n> -1\nTallene har nå blitt lagret i tallfil.txt\n>>>
```



A small window titled `tallfil.txt` showing the contents of the file:

2	4
4	16
6	36
8	64

Skriv_tall_til_fil.py



The screenshot shows the Thonny Python IDE interface. The top window displays the script `skriv_tall_til_fil.py` with the following code:

```
1 filnavn = input('Oppgi navn på fila: ')
2 f = open(filnavn, 'w') # Åpner fila for skriving
3
4 print('Skriv inn positive tall og tallet i andre lagres til', filnavn)
5 print('Avslutt med å skrive -1')
6 tall = 0 # For at while-lokka skal få True
7 nl = '\n' # Kode for nylinje i tekst
8 tab = '\t' # Kode for å legge inn en tabulator
9 while tall != -1:
10     tall = int(input('> '))
11     if tall != -1:
12         f.write(str(tall) + tab + str(tall**2) + nl)
13
14 f.close()
15
16 print('Teksten har nå blitt lagret i', filnavn)
```

The bottom window, titled "Shell", shows the execution of the script:

```
>>> %Run skriv_tall_til_fil.py
Oppgi navn på fila: mintest3.txt
Skriv inn positive tall og tallet i andre lagres til mintest3.txt
Avslutt med å skrive -1
> 2
> 4
> 6
> -1
Teksten har nå blitt lagret i mintest3.txt
^^^
```

Lese strenger fra fil

- For å lese strenger fra fil, benyttes:

```
streng = filvariabel.read() # returnerer hele innholdet
```

```
streng = filvariabel.readline() # returnerer ei linje
```
- `read()` kan benyttes hvis fila ikke inneholder for store mengder data, men bør unngås for store filer.
- `readline()` gjør det mulig å gå igjennom fila linje for linje, men krever at fila er oppdelt med linjeskift (`\n`).
 - Kan bruke while-løkke for å sjekke om vi har kommet til enden.
- Vi ser på to eksempler:

Lese strenger fra fil

```

Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på lysark/Nytt/L...
Program arguments:
<untitled> Lysark 14.py Lysark 19.py Lysark 19-2.py Lysark 20.py Lysark 22 - skriv_linjer_til_fil.py Lysark 24 - skriv_tall_til_fil.py Lysark 27-1 les_fil.py
1 filnavn = input('Oppgi navn på fila: ')
2 try:
3     f = open(filnavn, 'r') # Åpner fila for lesing
4     innhold = f.read()    # Leser inn hele driten på en gang
5     f.close()
6     print(innhold)
7 except IOError:
8     print('Finner ikke fila.')

Shell x
>>> %Run 'Lysark 27-1 les_fil.py'
Oppgi navn på fila: tallfil.txt
2      4
4      16
6      36
8      64
>>>

```

tallfil.txt

2	4
4	16
6	36
8	64

```

Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på l...
Program arguments:
<untitled> Lysark 14. Lysark 19. Lysark 19-2. Lysark 20. Lysark 22 - skriv_linjer_ti Lysark 24 - skriv_tall_ti Lysark 27-1 les_f Lysark 27-2 les_linjer
1 filnavn = input('Oppgi navn på fila: ')
2 teller = 1 # Skal telle antall linjer
3 try:
4     f = open(filnavn, 'r') # Åpner fila for lesing
5     linje = f.readline()  # Leser første linje
6     while linje:
7         linje = linje.strip() # Fjern linjeskift
8         print(f'{teller}\t{linje}')
9         linje = f.readline()
10        teller += 1
11    f.close()
12 except IOError:
13    print('Finner ikke fila.')

Shell x
>>> %Run 'Lysark 27-2 les_linjer_fil.py'
Oppgi navn på fila: tallfil.txt
1      2      4
2      4      16
3      6      36
4      8      64
>>>

```

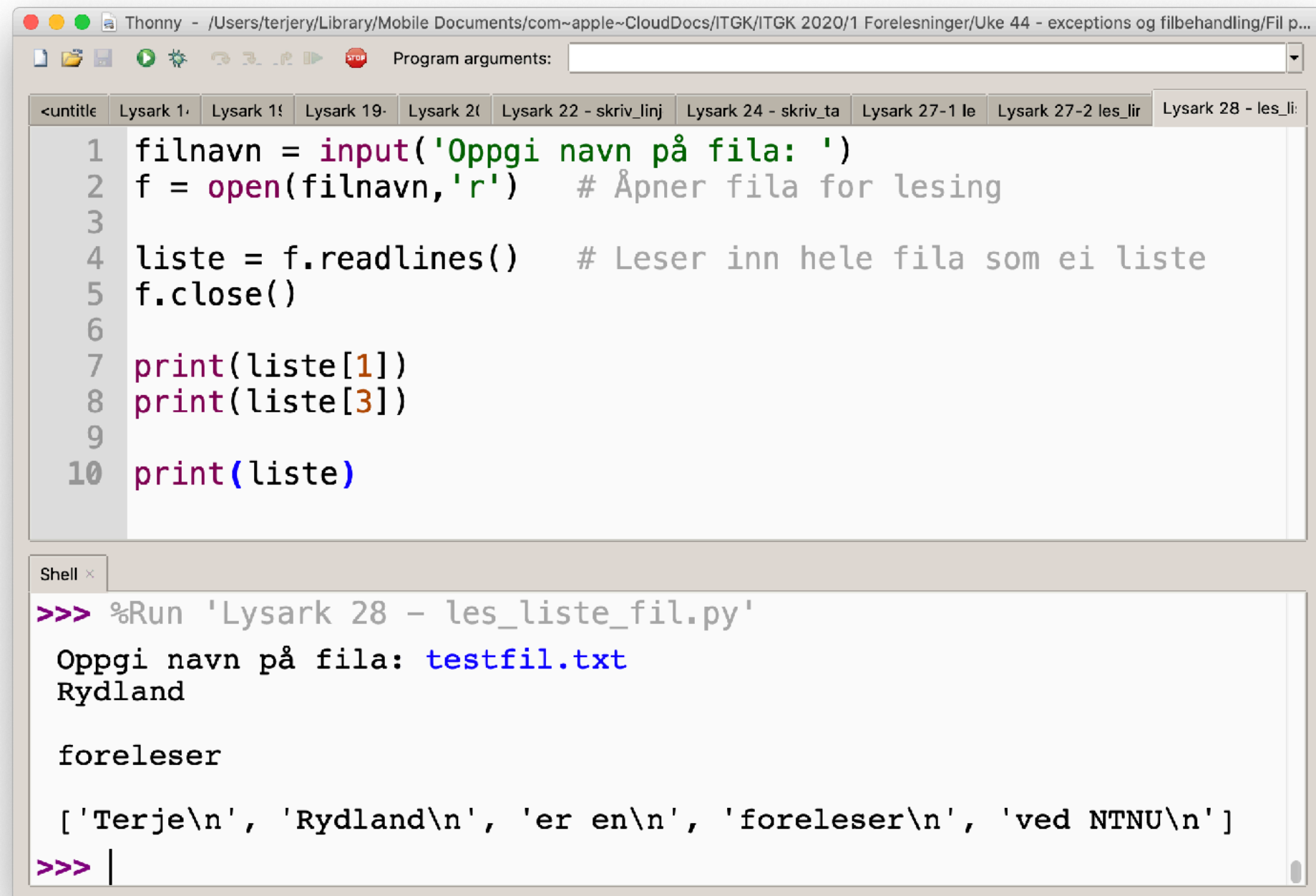
Lese fil som liste av strenger

- Vi kan også få returnert innholdet av ei tekstfil som ei liste av strenger:

```
liste = filvariabel.readlines() # returnerer liste
```

- `readlines()` kan være veldig praktisk hvis man skal utføre listeoperasjoner på innholdet i fila, for eksempel highscore liste for dataspill.

- Legg merke til at "non-printing" tegn som `\n` etc er med



The screenshot shows the Thonny Python IDE interface. The top toolbar includes icons for file operations and a 'Program arguments' field. The editor displays a Python script with the following code:

```
1 filnavn = input('Oppgi navn på fila: ')
2 f = open(filnavn, 'r') # Åpner fila for lesing
3
4 liste = f.readlines() # Leser inn hele fila som ei liste
5 f.close()
6
7 print(liste[1])
8 print(liste[3])
9
10 print(liste)
```

The bottom panel shows the Shell output for running 'Lysark 28 - les_liste_fil.py':

```
>>> %Run 'Lysark 28 - les_liste_fil.py'
Oppgi navn på fila: testfil.txt
Rydland

foreleser

['Terje\n', 'Rydland\n', 'er en\n', 'foreleser\n', 'ved NTNU\n']
>>> |
```

Å bruke Pythons **for**-løkke til å lese linjer

- Python tillater å skrive en **for**-løkke som automatisk leser linjer fra fil og slutter for-løkka når den når slutten av fila:
 - Format:

```
for line in file_object:  
    kode..
```
 - Løkka går igjennom (itererer) fila linje for linje

Oppgave: les tall fra fil



- Skriv Python-koden for å gjøre følgende:
 - Spør bruker om filnavn
 - Åpne fila for lesing
 - Les inn fra fil, linje for linje ved hjelp av for-løkke
 - Konverter fra streng til tall
 - Skriv ut tallet lest inn fra fil opphøyd i tredje

Oppgave: les tall fra fil



```
Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 4...
Program arguments:

<unt Lysar Lysar Lysark Lysar Lysark 22 - sk Lysark 24 - s Lysark 27 Lysark 27-2 Lysark 28 - Lysark 31

1 filnavn = input('Oppgi filnavn: ')
2 try:
3     f = open(filnavn, 'r')
4     for linje in f:
5         tall = int(linje)
6         print(tall**3)
7     f.close()
8 except IOError:
9     print(f'Fila {filnavn} finnes ikke.')

Shell x
>>> %Run 'Lysark 31 les_tall_fil.py'
Oppgi filnavn: tallfil2.txt
8
27
64
125
216
343
512
>>> |
```

Lese fra ei fil, tegn for tegn

- Det er mulig å spesifisere at vi skal lese ett gitt antall tegn i gangen fra en fil. Dette gjøres ved:

```
tegn = filvariabel.read(n) # n er antall tegn
```
- Dette gjøre det mulig å for eksempel søke etter et spesielt tegn inne i fila.
- Vi ser på et eksempel der brukeren kan skrive inn et tegn som det skal søkes etter i ei fil med filnavn som brukeren også skriver inn:

`les_tegn_for_tegn.py`

les_tegn_for_tegn.py

testfil2.txt

Dette er en linje
og en til
og enda en

```
Thonny - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil på lysark/Nytt/Lysark 33 les_tegn_for_t...
Program arguments:

Lysark 33 les_tegn_for_tegn.py x
1 filnavn = input('Oppgi filnavn: ')
2 try:
3     f = open(filnavn, 'r')
4     søker = input('Oppgi tegn det søkes etter: ')
5     posisjon = 0
6     tegn = f.read(1) # Leser et tegn fra fila
7     posListe = []
8     while tegn:
9         if tegn == søker:
10             posListe.append(posisjon)
11             tegn = f.read(1) # Leser neste tegn
12             posisjon += 1
13     f.close()
14     if len(posListe) != 0:
15         print(f'Tegnet {søker} ble funnet i følgende posisjoner {posListe}')
16     else:
17         print(f'Tegnet {søker} ble ikke funnet i fila "{filnavn}")
18 except IOError:
19     print(f'Fila {filnavn} finnes ikke.')

Shell x
Oppgi filnavn: testfil2.txt
Oppgi tegn det søkes etter: e
Tegnet e ble funnet i følgende posisjoner [1, 4, 6, 9, 16, 21, 31, 36]
>>> |
```

Oppsummering

- Filhåndteringsprosess:
 - Åpner en fil med en gitt aksess
 - Leser fra fil/skriver til, evt. forflytter filpeker
 - Lukker fil
- Vi kan jobbe med flere filer samtidig:
 - Filvariabelen med referanse til fila bestemmer hvilken fil vi jobber med.

Kapittel 9.3

Serialisering av objekter

Problem

- Hvordan lagre slike datastrukturer på fil
 - Vi kan ikke lagre dem som tekst, da mister vi sammenhengen mellom nøkkel og verdi
- Løsning
 - Lagre dem som binærfiler - en strøm av bytes
- Ulempe
 - Vi må vite hvilken datastruktur som representeres når filen leses
- Fordel
 - Slipper å strippe ut uønskede tegn

Serialisering av objekter

Kap 9.3

- Serialisering av objekter er prosessen å **konvertere et objekt til en strøm av bytes** som kan lagres til fil, som senere kan lastes inn igjen.
 - F.eks. for lagring av **dictionary** eller **set**
 - Da beholdes datastrukturen når data hentes inn
- Python har biblioteket **pickle** som gjør det mulig å lagre og laste dictionary til/fra disk.

- Fordeler med binærfiler (vs. tekstfil)
 - Får dataene direkte inn i ønsket datastruktur – Slipper konvertering til / fra strenger
 - Vanligvis mindre plasskrevende
- Ulemper vs. tekstfil
 - Fila er ikke lesbar for mennesker
 - Applikasjonsavhengig
 - må ha spesifikk kjennskap til filformatet for å kunne bruke fila

Lagre dictionary til disk

- For å lagre en dictionary til disk kan **biblioteket pickle** brukes sammen med **metoden dump**.
- Må åpne en fil som binærfil og ikke som tekst, ettersom det er mer enn tekst som lagres (struktur):

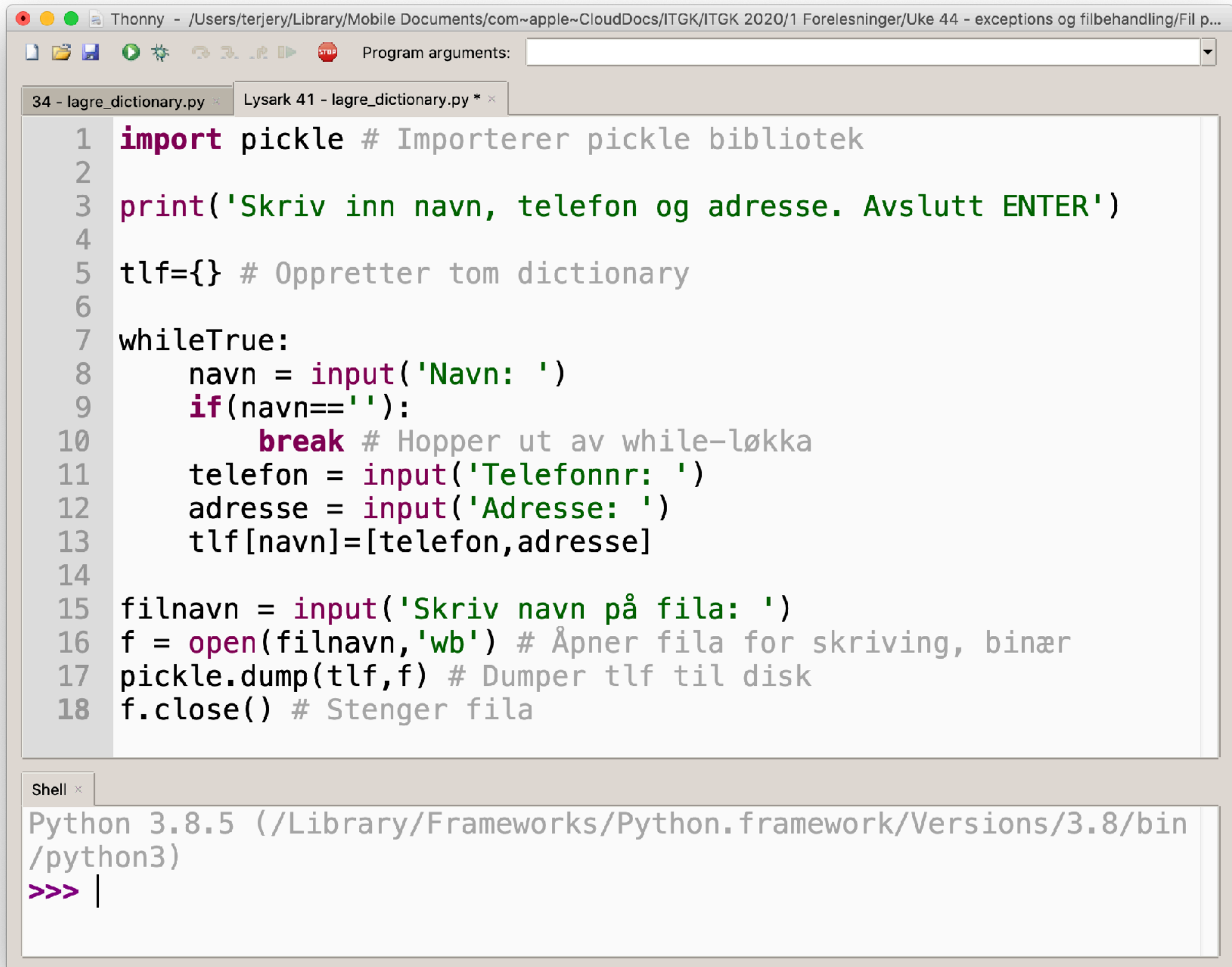
```
db={'Jo':[10,'Skogata 3'],'Per':[20,'Height 2']}  
import pickle                                # Importerer modulen  
f = open('database.dat','wb')                # wb = write binary  
pickle.dump(db,f)                            # Dumper db til disk  
f.close()                                    # Stenger fila
```

lagre_dictionary.py

- For å lagre en dictionary til disk kan biblioteket pickle brukes sammen med metoden dump.
- Må åpne en fil som binærfil og ikke som tekst, ettersom det er mer enn tekst som lagres (struktur):

```
db={'Jo':[10,'Skogata 3'],'Per':[20,'Height 2']}  
import pickle # Importerer modulen  
f = open('database.dat','wb') # b=binary  
pickle.dump(db,f) # Dumper db til disk  
f.close() # Stenger fila
```


lagre_dictionary.py



The screenshot shows a Thonny Python IDE window. The title bar indicates the file path: `/Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK/ITGK 2020/1 Forelesninger/Uke 44 - exceptions og filbehandling/Fil p...`. The editor has two tabs: `34 - lagre_dictionary.py` and `Lysark 41 - lagre_dictionary.py`. The code in the editor is as follows:

```
1 import pickle # Importerer pickle bibliotek
2
3 print('Skriv inn navn, telefon og adresse. Avslutt ENTER')
4
5 tlf={} # Oppretter tom dictionary
6
7 while True:
8     navn = input('Navn: ')
9     if (navn==''):
10         break # Hopper ut av while-løkke
11     telefon = input('Telefonnr: ')
12     adresse = input('Adresse: ')
13     tlf[navn]=[telefon,adresse]
14
15 filnavn = input('Skriv navn på fila: ')
16 f = open(filnavn,'wb') # Åpner fila for skrijving, binær
17 pickle.dump(tlf,f) # Dumper tlf til disk
18 f.close() # Stenger fila
```

Below the editor is a `Shell` tab. It shows the Python version `Python 3.8.5 (/Library/Frameworks/Python.framework/Versions/3.8/bin/python3)` and the prompt `>>> |`.

Laste inn dictionary fra disk

- Laste inn dictionary fra disk gjøres ved hjelp av load metoden i pickle-biblioteket.
- Husk at åpne fila som binærfil og ikke tekst.

```
import pickle  
  
f = open('datafil.dat', 'rb')      # r=read, b=binary  
  
data = pickle.load(f)              # Laster inn dict fra disk  
  
f.close()
```



The screenshot shows a Thonny IDE window with a file named 'Lysark 42 - laste_dictionary.py'. The code in the editor is as follows:

```
1 import pickle # Importerer pickle bibliotek  
2  
3 filnavn = input("Skriv inn navn på datafil: ")  
4  
5 f = open(filnavn, "rb") # Åpner for lesing og binærfil  
6 database = pickle.load(f) # Laste inn dictionary fra disk  
7 f.close() # Stenger fila  
8  
9 for item in database:  
10     print(item, ":", database[item])
```

Below the editor is a 'Shell' window showing the Python 3.8.5 prompt 'Python 3.8.5 (/Library/Frameworks/Python.framework/Versions/3.8/bin/python3)' and the prompt '>>>>'.