NTNU

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Filer og unntak (exceptions)
Utgave 3: Kap. 6

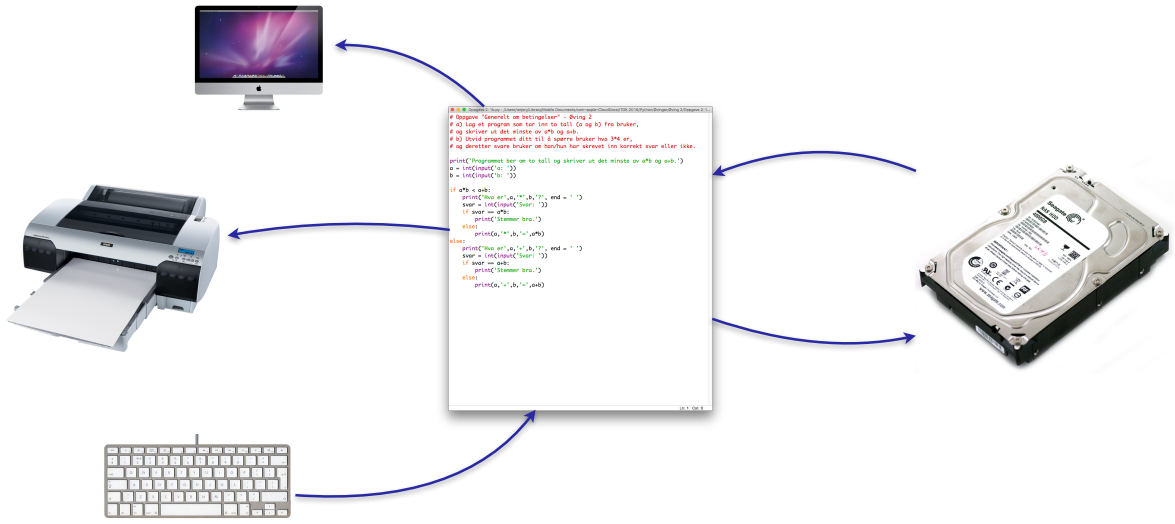
Terje Rydland - IDI/NTNU

Læringsmål og pensum

- Mål
  - Lære bruk av inn- og ut-operasjoner i Python
  - Lære å kunne bruke lesing og skriving til fil
  - Lære å bruke unntak (Exceptions)
- Pensum
  - Starting out with Python,Chapter 6 ”Files and Exceptions”

Inn- og utoperasjoner

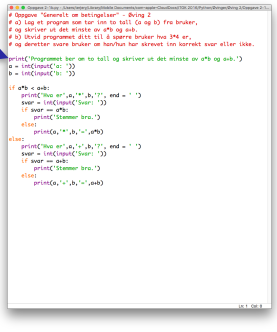
- I programmering brukes begrepet inn/ut-operasjoner, I/O eller IO når programmet leser eller skriver innholdet av variabler (data) til omverdenen.
  - Omverdenen kan være skjermen (ut), tastaturet (inn), skriver (ut), eller til filer (inn og ut)



Inn- og utoperasjoner

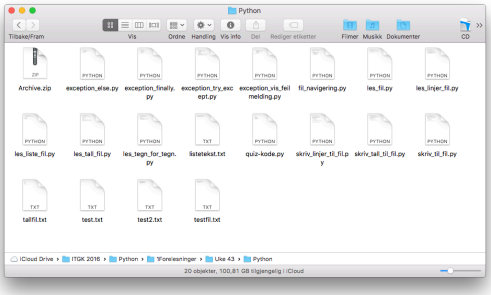
- Bruk av tastatur og skjerm er ok for små datamengder, men upraktisk for store datamengder:
  - Eks. værdata, modeller av skip eller konstruksjon, osv. som kan bestå av milliarder av bytes (gigabytes) eller mer.
  - Eks. data fra seismiske undersøkelser kan være i størrelsesorden pentabytes (en million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



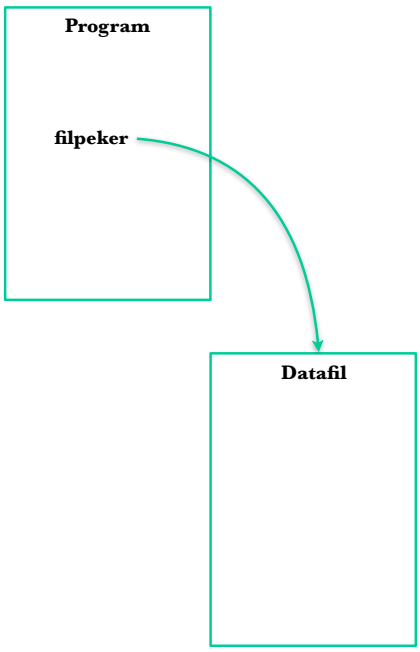
Inn- og utoperasjoner (2)

- 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)
  - gjør det mulig å 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).



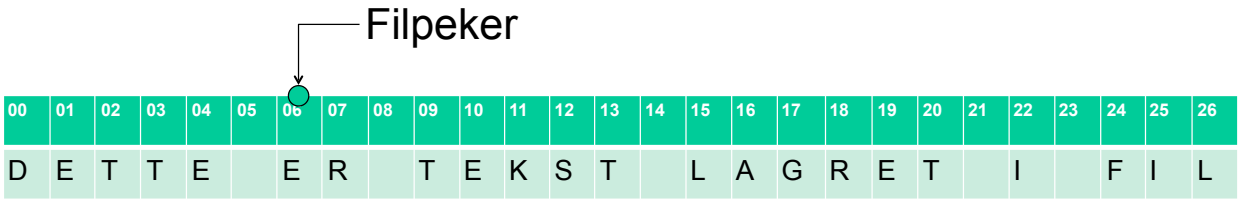
Hva er en fil i Python?

- 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.



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.



Prosessen 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.
     - Skriving: 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.

les_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Foreles...

```
filnavn = input("Oppgi navn på fila: ")
f = open(filnavn,"r") # Åpner fila for lesing

innhold = f.read() # Leser inn hele dritten på en gang
f.close() # Stenger fila

print(innhold) # Skriver ut innholdet i variabelen
```

Ln: 9 Col: 0

Filhåndtering i Python

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

f representerer variabelen som tar vare på filpekeren

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
```

Tilgangstyper for fopen

- Vi har følgende tilgangstyper for `open`:

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


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

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

skriv_til_fil.py

*skriv_til_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/P...

```
filnavn = input('Oppgi navn på fila: ')
f = open(filnavn,'w') # Åpner fil for skriving

print('Skriv inn tekst som lagres til',filnavn)
print('Avslutt med å trykke enter (uten tekst)')

tekst ='noe' # For å komme inn i while-lokka første gang
enter ='\n'  # For linjeskift (newline)

while tekst!='':
    tekst = input('> ')
    if(tekst!=''):
        f.write(tekst+enter)

f.close()

print('Teksten har nå blitt lagret i',filnavn)
```

Ln: 17 Col: 46

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

skriv_linjer_til_fil.py

skriv_linjer_til_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/...

```
filnavn = input('Oppgi navn på fila: ')

print('Skriv inn tekst som lagres til',filnavn)
print('Avslutt med å trykke enter (uten tekst)')
tekst ='noe'# For at while-lokka skal få True
enter ='\n' # Tegn for ny linje
liste =[]   # Lager en tom liste

while(tekst!=''):
    tekst = input('> ')
    if(tekst!=''):
        liste.append(tekst+enter)


f = open(filnavn,'w') # Åpner fila for skriving
f.writelines(liste)   # Skriver lista av strenger til fil
f.close()             # Stenger fila


print('Teksten har nå blitt lagret i',filnavn)
```

Ln: 21 Col: 0

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)
- 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

Skriv_tall_til_fil.py

*skriv_tall_til_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesninger/Uk...

```
filnavn = input('Oppgi navn på fila: ')
f = open(filnavn,'w') # Åpner fila for skriving

print('Skriv inn positive tall og tallet i andre lagres til',filnavn)
print('Avslutt med å skrive -1')
tall = 0 # For at while-lokka skal få True
enter ='\n'  # Kode for nylinje i tekst
tab = '\t'   # Kode for å legge inn en tabulator

while(tall!=-1):
    tall = int(input('> '))
    if (tall!=-1):
        f.write(str(tall) + tab + str(tall**2) + enter)

f.close()

print('Teksten har nå blitt lagret i',filnavn)
```

Ln: 16 Col: 0

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:

les_fil.py les_linjer_fil.py

Lese strenger fra fil

les_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/Uke...

```
filnavn = input("Oppgi navn på fila: ")
f = open(filnavn,"r") # Åpner fila for lesing

innhold = f.read() # Leser inn hele dritten på en gang
f.close() # Stenger fila

print(innhold) # Skriver ut innholdet i variabelen
```

Ln: 2 Col: 29

*les_linjer_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesninger/...

```
filnavn = input('Oppgi navn på fila: ')
f = open(filnavn,'r')           # Åpner fila for lesing

teller = 1                      # Skal telle linjer i fil

linje = f.readline()            #Leser ei linje fra fil
while linje :                   # Så lenge linje har innhold (streng)
    ren_linje=linje.strip()     # Fjerner linjeskift
    print(teller,ren_linje)
    linje = f.readline()        # Leser en ny linje
    teller +=1                  # Samme som teller = teller + 1

f.close()
```

Ln: 13 Col: 9

Lese som liste av strenger fra fil

- 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.
- Vi ser på et eksempel:

les_liste_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1For...

```
filnavn = input('Oppgi navn på fila: ')
f = open(filnavn,'r')   # Åpner fila for lesing

liste = f.readlines()   # Leser inn hele fila som ei liste
f.close()

print(liste[1])
print(liste[3])
print(liste)
liste.sort()
print(liste)
```

Ln: 1 Col: 30

les_liste_fil.py

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

les_tall_fil.py

Oppgave: les tall fra fil

*les_tall_fil.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesni...

```
filnavn = input('Oppgi navn på fila: ')
f = open(filnavn,'r')   # Åpner fila for lesing

for linje in f:
    tall = int(linje)   # Oversetter streng til heltall
    print(tall**3)
f.close()


#Alternativ:
#linje = f.readline()
#
#while(linje):
#    tall = int(linje) # Oversetter streng til flyttall
#    linje = f.readline() # Leser ny linje fra fil
#    print (tall**3) # Skriver ut tallet opphoyd i tredje
#f.close()
```

Ln: 17 Col: 10

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

*les_tegn_for_tegn.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016...

```
filnavn = input('Oppgi navn på fila: ')
soker = input('Oppgi tegn som det søkes etter: ')
f = open(filnavn,'r')       # Åpner for lesing

posisjon = 0

tegn = f.read(1) # Leser ett tegn fra fila
while tegn:
    if tegn == soker:
        print('Jeg fant',soker,'på posisjon',posisjon)
    tegn = f.read(1)
    posisjon += 1     # Samme som posisjon = posisjon +1
f.close()           # Stenger fila
```

Ln: 13 Col: 34

Exception / Unntak

- En **exception** er en feil som oppstår under kjøring som får programmet til å stoppe opp.
- Typiske feil som gir **exception** er:
  - Prøver å gjøre om tekststrenger til tall med strenger uten tall
  - Divisjon på 0
  - Prøver å åpne filer som ikke eksisterer
- En måte å unngå dette er å sjekke bruker-input.
- I Python kan du også bruke **`try/exception`** uttrykk for å unngå at programmet stopper opp under slike feil.

Exception: try – except uttrykk

- "Usikker" kode skrives inne i et `try:` uttrykk
  - Tester ut om denne koden kjører uten problemer
- I tillegg må vi legge til kode som fanger opp eventuelle feil

  `except ExceptionName:`

```
try:                  # En feil i try-blokka, trigger except
     uttrykk
     uttrykk
     …
except ExceptionName: # Hopper hit hvis feil i try
     uttrykk
     uttrykk
```

Exception – ExceptionName

- Ulike typer Exceptions har ulike navn.
- Vi kan fange opp disse ved å lage en exception i kode.
- Typiske ExceptionName er:
  - ValueError: Typisk feil i datatype (streng når det skal være tall)
  - ZeroDivisionError: Prøver å dividere med 0
  - IOError: Feil med filbehandling
  - Exception: Alle mulige feil (generell)
- Ser på et eksempel på bruk av try – except:

exception_try_except.py

exception_try_except.py

*exception_try_except.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesni...

```
def main():
    try:
        tall = int(input('Skriv inn et tall: '))
        print('500 delt på',tall,'er',500/tall)
        filnavn=input('Skriv inn et filnavn: ')
        f = open(filnavn)
        tekst=f.read()
        print(tekst)

    except ValueError:
        print('FEIL: Må skrive inn et tall, ikke tekststreng!!!')
    except ZeroDivisionError:
        print('FEIL: Kan ikke skrive tallet 0!!!')
    except IOError:
        print('FEIL: Det oppsto en feil ved lesing av fila',filnavn)
    except:
        print('FEIL: Programmet feilet av uviss grunn!')

main()
```

Ln: 5 Col: 39

Exception – vis innebygd feilmelding

- Det er mulig å fange opp feilmeldingen som Python gir ved en Exception ved bruk av følgende kode:

```
try:
        uttrykk…
except Exception as variabel:
        print(variabel)
```

- Uttrykket ***as variabel***, fanger opp feilen og lagrer feilmeldingen i en variabel som opprettes.
- Vi ser på et eksempel:

exception_vis_feilmelding.py

Exception_vis_feilmelding.py

*exception_vis_feilmelding.py - /Users/terjery/Library/Mobile Documents/com~apple~Clou...

```
def main():
    try:
        tall = int(input('Skriv inn et tall: '))
        print('500 delt på',tall,'er',500/tall)
        filnavn=input('Skriv inn et fil navn: ')
        f = open(filnavn)
        tekst=f.read()
        print(tekst)

    except Exception as feil:
        print('Feilmelding:',feil)

main()
```

Ln: 13 Col: 8

Exception – else og finally

- Et `try – except` uttrykk kan også bestå av `else` og `finally`:
- `else` blir utført hvis ingen exceptions ble trigget.
- `finally` blir utført til slutt uansett om exceptions ble trigget eller ikke

```
try:
      uttrykk…
except ExceptionName:
   uttrykk…
else:
   uttrykk…
finally:
   uttrykk…
```

exception_finally.py

exception_finally.py

*exception_finally.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesninge...

```
def main():
    try:
        filnavn=input('Hva heter fila? ')
        f = open(filnavn)
        tekst=f.read()
    except Exception as err:
        print(err)
    else:
        print(tekst)
    finally:
        print('Dette skriver jeg ut uansett om feil eller ikke!!!')

main()
```

Ln: 13 Col: 6

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.

Exceptions / Unntak

- Exception: feil som skjer når et program kjører
  - Som regel fører det fører det til at programmet stopper (kræsjer)
- Exception handling: Håndtere "exceptions" ved å gi brukeren fornuftig tilbakemelding uten at programmet stopper helt opp.
- Benytter:

```
try:      # Prøv om koden lar seg kjøre
except    # Fanger opp hvis koden i try feiler
except Exception as variable: # fanger feilmelding
else:     # Kjøres hvis det ikke blir exception
finally: # Kjøres uansett til slutt
```