NTNU

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Unntak (exceptions) (Kap 6)
Dictionaries (Kap. 9)

Terje Rydland - IDI/NTNU

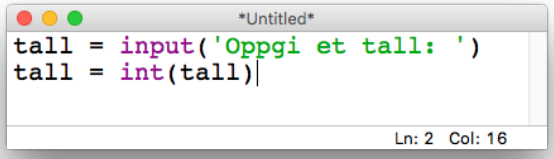
Læringsmål og pensum

- Mål
  - Lære å bruke unntak (Exceptions)
  - Dictionaries
- Pensum
  - Starting out with Python,Chapter 6 ”Files and Exceptions”

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

Denne enkle koden vil få programmet til å krasje hvis man skriver inn et flyttall eller en tekst på tastaturet fordi vi da vil prøve å konvertere en tekst eller et flyttall til heltall.



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

Praktisk ved feilsjekking på indata

try2-3.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Py...

```
while True:
    try:
        verdi = int(input('Oppgi et heltall: '))
        verdi *= 5
        break
    except ValueError:
        print('Du må skrive et heltall. Prøv igjen!')
print(verdi)
```

Ln: 9 Col: 0

try2-2.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Fore...

```
def innData():
    riktig,verdi = False,0
    try:
        verdi = int(input('Oppgi et heltall: '))
        verdi *= 5
        riktig = True
    except ValueError:
        print('Du må skrive et heltall. Prøv igjen!')
    return riktig, verdi


riktig = False
while not riktig:
    riktig,verdi = innData()
print(verdi)
```

Ln: 11 Col: 0

Hvis man vil skille mellom forskjellige feil

try3.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1Forelesninger/Uke 47/Torsdag/try3.py (3.5.2)

```
def innData():
    riktig,verdi = False,0
    try:
        verdi = input('Oppgi et heltall [1-4]: ')
        verdi = int(verdi)  # gir ValueError hvis verdi ikke kan konverteres til int
        if verdi in [1,2,3,4]:
            riktig = True
        else:
            print('Tallet må være mellom 1 og 4.')
    except ValueError:
        try:
            x = float(verdi)    # gir ValueError hvis verdi ikke er et tall
            print('Du må skrive et heltall, ikke et desimaltall.')
        except ValueError:
            print('Du må skrive et tall, ikke en tekst.')
    return riktig, verdi


riktig = False
while not riktig:
    riktig,verdi = innData()
```

Ln: 21 Col: 28

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
```


Datastruktur: *Dictionaries* Kap 9.1

- Dictionary er et objekt som lagrer en samling av data.
- Minner litt om lister men har klare forskjeller:
  - Defineres ved å bruke krøllparanteser { } (ALT+SHIFT 8/9 på Mac)
  - Kan bruke **hva som helst som nøkkel** (indeks) (ikke bare tall som i lister):
    - Tekststrenger, Heltall (men trenger ikke å være i rekkefølge), Flyttall, Sannhetsverdier (True eller False), En kombinasjon av de ovenfor

```
A = {}     # Tom dictionary
A['Kari'] = 92925492 # Oppretter et element
tlf={'Jo':73540000,'Per':92542312,'Else':54239212}
print(tlf['Per']) # Skriver ut verdien 92542312
```

- {nøkkel$_1$: verdi, nøkkel$_2$: verdi,…,nøkkel$_n$: verdi}

Forskjeller dictionary vs. 2D-tabeller

- Lister og 2D-tabeller bruker tall (0 ->) som indekser, en dictionary kan bruke hva som helst som indeks (nøkkel)
  - enklere å søke i en dictionary enn en 2D-tabell
  - enklere å slette et innslag i en dictionary
  - enklere å oppdatere elementer i en dictionary

Eksempel - Telefondatabase

- 3 mulige datastrukturer
  - Liste der navn har indekser med partall, nummer med oddetall
    - ['Per' , 23456789 , 'Kari' , 43217654]
  - 2D liste der navn og nummer er en liste i lista
    - [['Per' , 23456789] , ['Kari' , 43217654]]
  - Dictionary der navnet utgjør nøkkelen og nummeret er verdien
    - {'Per' : 23456789 , 'Kari' : 43217654}

Eksempel - Telefondatabase

- Legge inn data
  - Skrive inn data og bruke 2 append-metoder
  - 2D liste
    - Skrive inn data og bruke 1 append-metode
  - Dictionary
    - Tilordne en dataverdi

Dictionary

```
def legg_inn(katalog):
    navn = input('Navn: ')
    tlf = input ('Telefonnr: ')
    katalog[navn] = tlf
    return katalog
```

Liste

```
def legg_inn(katalog):
    navn = input('Navn: ')
    tlf = input ('Telefonnr: ')
    katalog.append(navn)
    katalog.append(tlf)
    return katalog
```

2DListe

```
def legg_inn(katalog):
    navn = input('Navn: ')
    tlf = input ('Telefonnr: ')
    katalog.append([navn,tlf])
    return katalog
```


Eksempel - Telefondatabase

- Søk etter person og skriv ut data
  - Liste
    - if navn in liste
  - 2D liste
    - for i in range(len(2Dliste)):
      - if navn in 2Dliste[i] : funnet = True
  - Dictionary
    - if navn in dict.keys()

Dictionary

```
def finn(katalog):
    navn = input('Navn: ')
    if navn in katalog.keys():
        print(katalog[navn])
    else:
        print('Ukjent navn.')
```

Liste

```
def finn(katalog):
    navn = input('Navn: ')
    if navn in katalog:
        print(katalog[katalog.index(navn)+1])
    else:
        print('Ukjent navn.')
```

2DListe

```
def finn(katalog):
    navn = input('Navn: ')
    funnet = False
    for i in range(len(katalog)):
        if katalog[i][0] == navn:
            funnet = True
            break
    if funnet:
        print(katalog[i])
    else:
        print('Ukjent navn.')
    return katalog
```

Eksempel - Telefondatabase

- Redigere data/legge inn ny person
  - Liste
    - finne navnet i lista og notere index
    - oppdatere index+1
  - 2D liste
    - finne index på liste som inneholder navn
    - oppdatere 2Dliste[index][1]
  - Dictionary
    - dict[navn] = ny verdi

Dictionary

```
def rediger(katalog):
    navn = input('Navn: ')
    tlf = input('Oppgi nytt telefonnummer: ')
    katalog[navn] = tlf
    return katalog
```

Liste

```
def rediger(katalog):
    navn = input('Navn: ')
    tlf = input('Oppgi nytt telefonnummer: ')
    if navn in katalog:
        ind = katalog.index(navn)
        katalog[ind + 1] = tlf
    else:
        katalog.append(navn)
        katalog.append(tlf)
    return katalog
```

2DListe

```
def rediger(katalog):
    funnet = False
    navn = input('Navn: ')
    tlf = input('Oppgi nytt telefonnummer: ')
    for i in range(len(katalog)):
        if navn in katalog[i]:
            katalog[i][1] = tlf
            funnet = True
            break
    if not funnet:
        katalog.append([navn,tlf])
    return katalog
```


Eksempel - Telefondatabase

- Slette en person
  - Liste
    - Finne indeks i lista der personen finnes
    - Slette denne og påfølgende indeks
  - 2D liste
    - finne index på liste som inneholder navn
    - slette denne indeksen
  - Dictionary
    - Sjekke om navnet finnes i keys og bruke del

Dictionary

```
def slett(katalog):
    navn = input('Navn på den du vil slette: ')
    if navn in katalog.keys():
        del katalog[navn]
    else:
        print('Ukjent navn.')
    return katalog
```

Liste

```
def slett(katalog):
    navn = input('Navn på den du vil slette: ')
    if navn in katalog:
        ind = katalog.index(navn)
        katalog[ind:ind+2] = []
    else:
        print('Ukjent navn.')
    return katalog
```

2DListe

```
def slett(katalog):
    navn = input('Navn på den du vil slette: ')
    funnet = False
    for i in range(len(katalog)):
        if katalog[i][0] == navn:
            funnet = True
            break
    if funnet:
        katalog[i] = []
        katalog.remove([])
    else:
        print('Ukjent navn.')
    return katalog
```

Operasjoner for *dictionaries*

Operasjon	Forklaring
len(d)	Antall elementer i d
d[k]	Verdi til element i d med nøkkel k
d[k] = v	Sett element k til verdi v
del d[k]	Slett element k i d
d.clear()	Fjern alle elementer i d
d.copy()	Lag kopi av d

Operasjon	Forklaring
d.items()	Returnerer liste av (nøkkel,verdi) par
d.keys()	Returnerer liste av nøkler i d
d.values()	Returnerer liste av verdier i d
d.get(k)	Samme som d[k]
d.get(k,v)	Returnerer d[k] hvis k er gyldig, ellers v

`dictionary_metoder.py`