Tallkonvertering

September 11, 2020

```
<div class="navbar-header">
  <a class="navbar-brand" href="_Oving1.ipynb">Øving 1</a>
</div>
<ul class="nav navbar-nav">
    <li><a href="Intro%20til%20jupyter.ipynb">Intro til Jupyter</a></li>
  <li ><a href="Jeg%20elsker%20ITGK!.ipynb">Jeg elsker ITGK!</a></li>
<li ><a href="Kalkulasjoner.ipynb">Kalkulasjoner</a></li>
<li><a href="Input%20og%20variable.ipynb">Input og variable</a></li>
<li class="active"><a href="Tallkonvertering.ipynb">Tallkonvertering</a></li>
<li><a href="Peppes%20Pizza.ipynb">Peppes Pizza</a></li>
<li><a href="Geometri.ipynb">Geometri</a></li>
<li><a href="Vitenskapelig%20notasjon.ipynb">Vitenskapelig notasjon</a></li>
<li><a href="Tetraeder.ipynb">Tetraeder</a></li>
<li><a href="Bakekurs.ipynb">Bakekurs</a></li>
<li ><a href="James%20Bond%20and%20Operation%20round().ipynb">James Bond and Operation Round</a
</ul>
```

1 Tallkonvertering

Læringsmål:

- Konvertering mellom datatyper med standardfunksjoner som int() og float()
- Forstå definisjonen av heltall og flyttall

Starting Out with Python:

- Kap. 2.6
- Kap. 2.8

1.1 Tutorial del 1: Konvertering mellom datatyper

Vi kan ha ulike typer data, som tekststrenger (f.eks. `"Python"`), heltall (f.eks. `42`), flyttall (f.eks. `9.80`) og sannhetsverdier (`True`, `False`). Ofte kommer vi i situasjoner hvor vi har data av en viss type, men vi trenger samme data bare med en annen type. Da må vi konvertere dataene. Noen vanlige konverteringsfunksjoner:

`int()` - konverterer til heltall. - `int('423')` gir 423 (dvs. tekststrengen blir konvertert til et tall). Virker kun hvis tekststrengen faktisk inneholder et heltall. - `int(5.69)` gir 5 (dvs. for flyttall blir desimaldelen fjernet)

`float()` - konverterer til flyttall - `float('5.69')` gir 5.69 (tekststreng konvertert til tall) - `float('5')` gir 5.0, dvs. float() virker på tekststrenger enten de inneholder flyttall eller heltall (men ikke på strenger som er noe annet enn tall) - `float(5)` gir 5.0

`str()` - konverterer til tekststreng - `str(42)` gir '42' - `str(5.69)` gir '5.69' Koden under feiler fordi vi har glemt å konvertere. Kjør den og se hva som skjer.

[ ]:
```
alder = input("Hvor gammel er du?")
alder_mor = input("Hvor gammel er din mor?")
sum_alder = alder + alder_mor
print("Gratulerer, til sammen er dere", sum_alder, "år!")
svar = input("Hva syns dere om å være ", sum_alder, "? ")
```

Et eksempel på kjøring kan være som følger:

```
Hvor gammel er du? 13
Hvor gammel er din mor? 37
Gratulerer, til sammen er dere 1337 år!
Traceback (most recent call last):
  File "/Users/guttorm/Documents/tut2.py", line 5, in <module>
    svar = input("Hva syns dere om å være ", sum_alder, "? ")
TypeError: input expected at most 1 arguments, got 3
>>>
```

Den første feilen viser seg i linjen "Gratulerer…" Summen skulle ha blitt 50 år. Men vi har de to alderne fortsatt bare lagret som tekststrenger. Da betyr `+` å hekte sammen strengene, ikke å gjøre noen addisjon. Altså får vi `'13' + '37'` som blir `'1337'` heller enn `13 + 37` som blir `50`. Her måtte vi ha konvertert fra tekst til tall før vi gjorde addisjonen.

Den andre feilen oppstår i input-setninga. I `print()` er det lov å liste opp mange argumenter, både tekst og tall, med komma mellom. I en `input()` er det bare lov å ha ett argument, som må være en tekststreng (ledeteksten). Som det sies i feilmeldinga: "input expected at most 1 arguments, got 3". For å klare oss med bare ett argument - men samtidig få stilt det spørsmålet vi ønsker ("Hva syns dere om å være 50?" hvis summen var 50) - må vi plusse sammen delene til én tekststreng heller enn å ha delene skilt med komma. `input("Hva syns dere om å være" + sum_alder + "? ")` vil imidlertid ikke funke hvis sum_alder er et tall, for å det går ikke å plusse sammen tekst og tall. To tekster kan derimot godt plusses sammen (som vi nettopp så med `'1337'`). Hvis vi konverterer `sum_alder` til tekst, vil det funke. Fikset kode vist under:

[ ]:
```
alder = int(input("Hvor gammel er du?"))
alder_mor = int(input("Hvor gammel er din mor?"))
sum_alder = alder + alder_mor
print("Gratulerer, til sammen er dere", sum_alder, "år!")
svar = input("Hva syns dere om å være " + str(sum_alder) + "? ")
```

Altså: bruker `int()` i linje 1 og 2, dette gjør at vi får heltall i variablene `alder` og `alder_mor` så vi blir i stand til å regne med dem. Bruker deretter `str()` i linje 5 så denne opplysningen kan settes sammen med annen tekst og brukes i ledeteksten til en ny `input()`. Dette eksemplet viser dermed både et tilfelle hvor vi har tekst men trenger tall, og ett hvor vi har et tall men trenger tekst. Hvis det er aktuelt at brukeren skriver inn desimaltall på alder (f.eks. `13.5`) vil imidlertid koden over

ikke funke. Da måtte vi ha brukt funksjonen `float()` der vi nå har brukt `int()`.

1.2 a)

Be brukeren om å skrive inn 3 flyttall. Konverter først strengene til flyttall med `float()`, bruk så `int()` til å konvertere dem til heltall. Be også brukeren om å skrive inn et heltall og bruk `float()` til å konvertere tallet til et desimaltall.

Eksempel på utskrift fra programmet hvis du får til det til å funke:

```
Skriv inn et flyttall: 4.232443
Skriv inn enda et flyttall: 3.24324
Skriv inn et siste flyttall: 1.22342454
Konvertert til heltall blir det: 4 3 1
Skriv inn et heltall: 13
Konvertert til flyttall blir det: 13.0
```

Skriv koden din i blokka under:

[3]:
```
f1 = float(input('Skriv inn et flyttall: '))
f2 = float(input('Skriv inn enda et flyttall: '))
f3 = float(input('Skriv inn et siste flyttall: '))
print('Konvertert til heltall blir det:', int(f1), int(f2), int(f3))
Hel = int(input('Skriv inn et heltall: '))
print('Konvertert til flyttall blir det:', float(Hel))
```

```
Skriv inn et flyttall: 5.5
Skriv inn enda et flyttall: 6.8
Skriv inn et siste flyttall: 9.8
Konvertert til heltall blir det: 5 6 9
Skriv inn et heltall: 85
Konvertert til flyttall blir det: 85.0
```

Hint Hver gang du skal be brukeren om noe benyttes `input()`. Husk å ha det i en `int()` dersom det er et heltall du ber om og `float()` dersom det er et flyttall. En tekststreng som inneholder et flyttall, f.eks. `"4.325"`, kan ikke konverteres direkte til heltall. Dvs., `int("4.325")` vil gi feilmelding. Hvis du får en slik streng og egentlig ønsker int, må du først konvertere fra streng til float, deretter fra float til int.

1.3 b)

Skriv et program som ber brukeren oppgi navn og alder, samt hvor gammel vedkommende var da han/hun begynte å programmere. Et eksempel på kjøring er vist nedenfor, hvor *Martin, 21*, *19* og *Ja* bakerst i linjene 1, 2, 3 og 5 skrives av brukeren mens resten skrives av maskinen. Lag programmet slik at det tilpasser seg dataene som skrives inn. (Særlig for denne deloppgaven lønner det seg å se på tutorial lenger oppe hvis du ikke får den til)

Eksempel på kjøring:

```
Skriv ditt navn: Martin
Hei, Martin, hvor gammel er du? 21
```

```
Hvor gammel var du da du begynte å programmere? 19
Da har du programmert i 2 år.
Syns du de 2 årene har vært givende? Ja
Takk for svar, Martin!
```

Skriv koden din i blokka under.

[13]:
```
navn = input('Skriv ditt navn: ')
alder = input('Hei, ' + navn + ', hvor gammel er du? ')
alderK = input('Hvor gammel var du da du begynte å programmere? ')
tidK = int(alder) - int(alderK)
print('Da har du programmert i', tidK, 'år.')
ubetydelig = input('Syns du de, ' + str(tidK) +' årene har vært givende? ')
print('Takk for svar,', navn)
```

```
Skriv ditt navn: s
Hei, s, hvor gammel er du? 8
Hvor gammel var du da du begynte å programmere? 5
Da har du programmert i 3 år.
Syns du de,3årene har vært givende?5
Takk for svar, s
```

1.4 Tutorial del 2: int() vs. round()

Ofte har man flyttall, men trenger heltall, f.eks. hvis man skal bruke innebygde Python-funksjoner som krever heltall som argument, eller skal bruke tallet som indeks til en streng eller liste (som vi vil se senere i pensum). Flyttall kan konverteres til heltall med funksjoner som `int()` eller `round()`. Kodeblokka under viser litt forskjell på hvordan disse virker.

[ ]:
```
print("int() bare kutter desimalene, uansett hvor stor eller liten desimaldelen␣
 ↪er:")
print("int(2.25) er", int(2.25))
print("int(2.5) er", int(2.5))
print("int(2.99) er", int(2.99))
print("round() runder av til nærmeste heltall, f.eks.")
print("round(2.25) er", round(2.25))
print("round(2.51) er", round(2.51))
print("Hva hvis tallet er midt mellom to heltall?")
print("round(2.5) er", round(2.5))
print("round(3.5) er", round(3.5))
print("round() bruker en IEEE standard som velger partallet for␣
 ↪midt-imellom-situasjoner.")
print("Mens int() alltid gir heltall kan round() brukes for antall desimaler:")
print("round(2.5488, 1) blir", round(2.5488, 1))
print("round(2.5488, 3) blir", round(2.5488, 3))
print("Med negativt antall desimaler kan vi få round() til å runde større enn␣
 ↪heltall:")
print("round(12345.67, -3) blir", round(12345.67, -3)
```

Som du ser i eksemplet, blir 2.5 rundet av til 2 mens 3.5 blir rundet til 4. Dette kan virke litt uvant, i dagliglivet er man mest kjent med såkalt "kjøpmannsavrunding", hvor det alltid rundes opp hvis man er midt mellom (dvs., 2.5 skulle i så fall ha blitt rundet til 3). Konsekvent runding oppover når man er midt mellom har imidlertid en uheldig side, nemlig at man pådrar seg en systematisk feil hvis man har mange data som avrundes. Tenk f.eks. temperaturmålinger for lange perioder, hvor man deretter skal regne ut et snitt for hele perioden. Hvis alle temperaturer som er midt når det gjelder siste brukte siffer, rundes opp, vil snittet for perioden alltid bli litt for høyt. Hvis man i stedet går i partallsretning i alle slike midt mellom situasjoner, vil man runde opp cirka halvparten av gangene og ned cirka halvparten av gangene og dermed unngå slike systematiske feil. Men for kjøpmannen er systematisk runding oppover selvsagt bedre med tanke på å få inn mest mulig penger.

1.5 c)

Be brukeren om skrive inn et flyttall med minst 5 siffer både før og etter punktum. Ta vare på flyttallet i en variabel, konverter det deretter på diverse måter som antydet i eksempelutskriften under. Til slutt skal heltallet du fikk fra `int()` konverteres tilbake til et flyttall.

Kjøring av kode:

```
Vennligst gi inn et flyttall med minst 5 siffer både før og etter .
Hva er tallet ditt? 123456.724353
Konvertert til heltall med int(): 123456
Avrundet til nærmeste heltall: 123457
Avrundet til to desimaler: 123456.72
Avrundet til fire desimaler: 123456.7244
Avrundet til nærmeste tusen: 123000
Heltall fra int() konvertert tilbake til flyttall: 123456.0
```

Skriv koden din i blokka under.

[14]:
```
siffer = 5
print('Venligst gi inn et flyttall med minst', siffer, 'siffer både før og␣
 ↪etter .')
tall = float(input('Hva er tallet ditt? '))
tallint = int(tall)
print('Konvertert til heltall med int():', tallint)
print('Avrundet til nærmeste heltall:', round(tall))
print('Avrundet til to desimaler:', round(tall, 2))
print('Avrundet til fire desimaler:', round(tall, 4))
print('Avrundet til nærmeste tusen:', int(round(tall, -3)))
print('Heltall fra int() konvertert tilbake til flyttall:', float(tallint))
```

```
Venligst gi inn et flyttall med minst 5 siffer både før og etter .
Hva er tallet ditt? 123456.7891234
Konvertert til heltall med int(): 123456
Avrundet til nærmeste heltall: 123457
Avrundet til to desimaler: 123456.79
Avrundet til fire desimaler: 123456.7891
```

```
Avrundet til nærmeste tusen: 123000.0
Heltall fra int() konvertert tilbake til flyttall:, 123456.0
```

1.6 d)

Ta først inn et flyttall fra brukeren og spør deretter brukeren om hvor mange desimaler i det avrundede tallet som er ønskelig, og bruk `round()` med dette tallet.

Eksempel på kjøring:

```
Skriv et flyttall: 3.14159263
Hvor mange desimaler er ønskelig? 2
Tallet du skrev er  3.14159263 og med  2  desimaler blir det til  3.14
```

Skriv koden din i blokka under.

[17]:
```
ftall = float(input('Skriv inn et flyttall: '))
dtall = int(input('Hvor mange desimaler er ønskelig?'))
print('Tallet du skrev er', ftall, 'og med', dtall, 'desimaler blir det til',␣
 ↪round(ftall, dtall))
```

```
Skriv inn et flyttall: 3.14159
Hvor mange desimaler er ønskelig?2
Tallet du skrev er 3.14159 og med 2 desimaler blir det til 3.14
```