Input og variable

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 class="active"><a href="Input%20og%20variable.ipynb">Input og variable</a></li>
<li><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 Input og variable

Læringsmål:

- Lage programmer der brukeren gir inn tekst med input()
- Enkel bruk av variable
- Korrekt navngivning av variable

Starting Out with Python:

- Kap. 2.2
- Kap. 2.5-2.6

1.1 Tutorial del 1: funksjonen input()

Hvorfor trenger vi input? I mange programmer er det viktig å kunne la brukeren gi input. I en nettbutikk må kunden kunne velge produkt, oppgi adresse det skal sendes til, betalingsmåte, osv.

Input kan gis på mange måter, f.eks. via berøringsskjermer, mus, eller med stemme. Her skal du lære den måten som er lettest å programmere, nemlig input av tekst fra tastatur. Dette gjøres i Python ved hjelp av funksjonen `input()`. I likhet med `print()` er dette en funksjon i Pythons standardbibliotek. Kjør koden under, så ser du noen viktige forskjeller på print() og input()

print() og input()

```
[ ]: print("print() skriver tekst ut på skjerm og skifter linje")
     input("input() venter at du skriver noe og slår Enter: ")
     print("print() kan ha", "mange tekster", "skilt med komma")
     input("input() tillater kun en tekst! OK? ")
```

Den ene teksten `input()` kan inneholde, er den såkalte **ledeteksten** som skal forklare brukeren hva slags input som forventes. Ledeteksten bør være presis så brukeren vet hva slags opplysning det er spurt om.
F.eks. er "Oppgi vekt i kg:" en bedre ledetekst enn bare "Oppgi vekt:", som igjen er mye bedre enn "Skriv et tall:" eller "Gi input:". Blankt tegn bakerst i ledeteksten er ofte en fordel, ellers kommer brukerens input kloss i ledeteksten.
I det lille eksemplet over brukte vi ikke resultatet fra `input()`, men vanligvis ber vi brukeren om data fordi dataen trengs til noe. For noe litt mer nyttig, anta at vi ønsker å spørre brukeren om navn - og deretter benytte dette navnet i en påfølgende print-setning, så vi får en liten dialog mellom bruker og maskin:

```
Hei, hva heter du? Nina
Nina - det var et fint navn.
>>>
```

Her spør maskinen "Hei, hva heter du?", brukeren svarer "Nina" og maskinen skriver "Nina - det var et fint navn."
Denne lille dialogen kan oppnås med følgende kodelinje:

Enkel dialog med bruker:

```
[ ]: print(input("Hei, hva heter du? "), "- det var et fint navn.")
```

Denne koden funker fordi parenteser alltid utføres innenfra og ut. Print-setningen kan ikke kjøres før man vet hva som skal printes, Derfor må input-setningen kjøres først, den gir som resultat det navnet brukeren skriver inn (f.eks. Nina).
Dette navnet skrives deretter ut sammen med den påfølgende teksten "- det var et fint navn.")

1.2 a) Simpel bruk av input direkte i print()-setning

Ta utgangspunkt i tutorial-koden nedenfor (og kjør den gjerne en gang for å se hvordan den virker; hvis du ikke forstår hvordan, les først tutorial):

```
[3]: print(input("Navn? "), " - kult navn!")
     print(input('favorittfag? '), ' - interessant!')
```

```
Navn? ole
ole  - kult navn!
favorittfag?eina
eina  - interessant!
```

Endre tekststrengene i denne kodelinjen så dialogen med brukeren i stedet blir som vist nedenfor. Maskinen skal alltid gjenta det navnet brukeren skrev, Nina er bare et eksempel.

```
Navn? Nina
Nina - kult navn!
```

Legg så til en ny, lignende kodelinje som spør om brukerens favorittfag. Ved ferdig program skal dialogen se ut som nedenfor.
Utskriften til programmet skal selvsagt tilpasse seg det brukeren skriver inn som navn og favorittfag.

```
Navn? Per
Per - kult navn!
favorittfag? Ex.phil.
Ex.phil. - interessant!
```

1.3 Tutorial del 2: variable - grunnleggende intro

Hvorfor trenger vi variable? Poenget med variable er å **huske data underveis** i utførelsen av et program.

Variable er derfor et sentralt konsept i programmering, ikke bare i Python men uansett hva slags språk man programmerer i.

Uten variable støter vi fort på en rekke problemer fordi programmet vårt ikke kan huske noe, f.eks. at

- vi må be brukeren gi inn opplysninger på nytt som brukeren har gitt tidligere
- vi må regne ut på nytt data vi allerede har regnet ut tidligere

Dette sløser tid og strøm og vil i mange tilfeller gjøre programmet fullstendig ubrukelig.

I det lille eksempelprogrammet i tutorial del 1 klarte vi oss uten noen variabel, fordi navnet vi innhentet fra bruker kun ble benyttet én gang, og dette skjedde umiddelbart etter at det var tastet inn.

```
print(input("Hei, hva heter du? "), "- det var et fint navn.")
```

```
Hei, hva heter du? Nina
Nina - det var et fint navn.
>>>>
```

Men ofte skal samme data brukes flere ganger, og etter at vi har gjort andre ting i mellomtiden. Da må data huskes i variable. Anta at vi ønsker en bare litt mer avansert dialog med brukeren:

```
Hei, hva heter du? Nina
Nina - det var et fint navn.
Lykke til med ITGK, Nina!
>>>>
```

Her vil vi bruke det innleste navnet i to påfølgende print-setninger. Hvis vi prøver samme triks som tidligere med å sette input-setning direkte i print-setning, får vi koden:

```
print(input("Hei, hva heter du? "), "- det var et fint navn.")
print("Lykke til med ITGK,", input("Hei, hva heter du?"))
```

Kjøring viser hva som er dumt med denne koden, nemlig at spørsmålet "Hei, hva heter du?" kommer to ganger.

```
Hei, hva heter du? Nina
Nina - det var et fint navn.
Hei, hva heter du? Nina
Lykke til med ITGK, Nina
```

Ikke noe katastrofalt problem her, men tenk deg et program hvor samme opplysning skal brukes 100 ganger eller mer i en kritisk arbeidsoppgave som haster.

Kan vi løse det på en bedre måte? JA - med en variabel for å huske navnet. Koden blir da

[ ]:
```
navn = input("Hei, hva heter du?" )
print(navn, "- det var et fint navn.")
print("Lykke til med ITGK,", navn)
```

Dette programmet kan forklares som følger:

- linje 1, til høyre for `=` : bruker `input()` for å spørre hva brukeren heter
- linje 1, til venstre for `=`: oppretter en variabel som heter `navn`.
- linje 1, tegnet `=`. Dette er **tilordningsoperatoren**. Betyr at verdien av uttrykket på høyre side, resultatet av `input()`, blir husket i variabelen kalt `navn`. Hvis brukeren skriver Nina, vil variabelen `navn` da inneholde strengen `'Nina'`
- linje 2, variabelen `navn` brukes fremst i print-setningen. Merk at variabelnavnet **ikke** skal ha fnutter rundt seg. Med fnuttter ville vi i stedet ha skrevet "navn - det var et fint navn". Ordet navn som står bakerst i setningen "det var et fint navn." er ikke variabelen, her er ordet navn bare del av en tekststreng.
- linje 3, variabelen `navn` brukes bakerst i print-setningen. Igjen uten fnutter; det er ikke ordet navn vi ønsker å skrive, men den tekststrengen som variabelen `navn` inneholder (f.eks. Nina)

Ved hjelp av variabelen som her ble kalt navn, unngår vi å stille samme spørsmål to ganger. Vi spør bare én gang, i starten av programmet, og husker da opplysningen brukeren gir oss ved å putte den inn i en variabel.

Videre i programmet kan vi benytte denne variabelen hver gang vi trenger navnet - enten det som her var bare to ganger, eller om det hadde vært flere.

1.4 b) Huske input i variable

Kjør koden under for å se hvordan den virker. Som du vil se, plager den brukeren med å gjenta begge spørsmålene to ganger.

Forbedre koden ved å introdusere en variabel for navn og en annen variabel for favorittfag, slik at brukeren får hvert av spørsmålene bare en gang (dvs. spørres kun en gang om navn og kun en gang om favorittfag).

Hvis du er i tvil om hvordan du skal angripe problemet, se lignende eksempel i tutorial like over.

[5]:
```
navn = input('Navn? ')
print("Hei,", navn)
favorittfag = input("Favorittfag? ")
print(favorittfag, "- interessant!")
print("Ha en fin dag,", navn)
print("- og lykke til med", favorittfag)
```

```
Navn? a
Hei, a
Favorittfag? b
b - interessant!
Ha en fin dag, a
- og lykke til med b
```

Hvis du får til å bruke de to variablene som tenkt, skal kjøringen av det forbedrede programmet se slik ut (men også funke om brukeren skriver inn noe annet enn Ada på spørsmålet Navn? og noe annet enn ITGK på Favorittfag?)

```
Navn? Ada
Hei, Ada
Favorittfag? ITGK
ITGK - interessant!
Ha en fin dag, Ada
- og lykke til med ITGK
```

1.5 Tutorial del 3 - bruk av variable i beregninger

Variable brukes ikke bare i sammenheng med `input()`, men i alle mulige slags program. I matematiske beregninger skal resultatet av en beregning ofte brukes videre i nye beregninger. Da må disse tallene huskes i variable.
Koden under viser samme eksempel gjort på to måter, nemlig utregning av areal for en sirkel, samt volum for en sylinder som har denne sirkelen som grunnflate. Versjon 1 er gjort uten variable, mens Versjon 2 bruker variable.

Sirkel og sylinder

[ ]:
```python
import math

# VERSJON 1, uten variable
print("Areal av sirkelen:", math.pi * 5.4**2)
print("Volum av sylinderen:", math.pi * 5.4**2 * 7.9)

print()

# VERSJON 2, med variable
r = 5.4 # radius for en sirkel
A_sirkel = math.pi * r**2
print("Areal av sirkelen:", A_sirkel)
h = 7.9 # høyde sylinder hvor sirkelen er grunnflate
V_syl = A_sirkel * h
print("Volum av sylinderen:", V_syl)
```

Hvis du kjører koden, vil du se at begge gir samme resultat. Hva er da forskjellen?

- Versjon 2 er vesentlig lenger (6 kodelinjer, mot bare 2) fordi det brukes ekstra linjer på variable. Lenger kode er en mulig ulempe. MEN:

- Formlene i Versjon 2 er lettere å forstå fordi det er intuitive navn som `r`, `h`, `A_sirkel` heller enn bare tall direkte.
- Koden i V2 er mer fleksibel for å kjapt endre verdier. Hvis radius skal byttes fra 5.4 til 6.2 må dette tallet bare endres ett sted i V2, mens flere i V1.
- Versjon 1 utfører **5 operasjoner** av type `*` og `**`, mens Versjon 2 bare utfører ***3***. Dette fordi Versjon 2 husker arealet i A_sirkel og deretter kan bruke dette, mens Versjon 1 må regne ut `math.pi * 5.4**2` på nytt. **Med færre multiplikasjoner vil VERSJON 2 spare både strøm og tid i forhold til VERSJON 1, dvs. koden utfører mindre jobb og går raskere selv om det er flere kodelinjer.**

1.6 c) Bruke variable i beregninger

Nedenfor står et program hvor vi regner ut omkrets og areal for en sirkel etter de velkjente formlene O=2 r og A = r2. Bortsett fra de innebygde konstantene `math.pi` og `math.tau` (=2) bruker vi ingen variable. Dette gjør at når vi skal regne ut arealet av en sylinder hvor sirkelen er grunnflate, må vi gjøre om igjen flere beregninger som vi allerede har gjort tidligere.

Arealet av sylinderen med høyde h vil være `mOmkrets_sirkel * h +  2 * Areal_sirkel`, hvor det første leddet er arealet av sylinderveggen og det siste leddet er topp- og bunnlokket.

Oppgave: Endre koden ved å tilordne og deretter bruke variable for radiusen, høyden, sirkelens omkrets og areal, slik at programmet unngår å gjøre på nytt beregninger som allerede er gjort før.

[9]:
```
import math
r = 5.4
h = 7.9
areal = math.pi * r**2
om = math.tau * r
print("Har en sirkel med radius", r, "som er grunnflate i en sylinder med␣
 ↪høyde", h)
print("Omkrets av sirkelen:", om)  #tau er det samme som 2 pi
print("Areal av sirkelen:", areal)
print("Areal av sylinderen:", om * h + 2 * areal)
```

```
Har en sirkel med radius 5.4 som er grunnflate i en sylinder med høyde 7.9
Omkrets av sirkelen: 33.929200658769766
Areal av sirkelen: 91.60884177867838
Areal av sylinderen: 451.25836876163794
```

Resultatet av kjøring av koden skal være uendret, dvs utskrift skal bli som vist nedenfor (men hvis du vil, kan du gjerne i tillegg avrunde svarene til én desimal).

```
Har en sirkel med radius 5.4 som er grunnflate i en sylinder med høyde 7.9
Omkrets av sirkelen: 33.929200658769766
Areal av sirkelen: 91.60884177867838
Areal av sylinderen: 451.25836876163794
```

1.7 Tutorial del 4: Navngiving av variable

En variabel er et navn som representerer en verdi som lagres i datamaskinens minne. Den vanligste måten å opprette en variabel på er ved en tilordningssetning:

`variable = expression`

I dette tilfellet er variable navnet til variabelen, mens expression er verdien. Noen regler for slike tilordningssetninger:

- variabelen som opprettes skal alltid stå på venstre side av uttrykket, og venstre side skal kun inneholde denne variabelen, ikke noe annet
- høyde side kan alt fra en enkelt verdi (f.eks. et tall) eller en enkelt variabel, til mer sammensatte uttrykk som må beregnes. Hvis høyre side inneholder variable, må dette være variable som allerede er opprettet tidligere i koden.
- variabelnavnet må tilfredsstille følgende regler:
- ord som er reserverte ord i Python, f.eks. `if`, `def`, eller som er navn på standardfunksjoner som `print`, `min`, `max`, … bør unngås som varibelnavn
- variabelnavn må begynne med en bokstav eller tegnet _ (understrek)
- kan ellers inneholde bokstaver, tall og understrek, dvs. kan f.eks. ikke inneholde blanke tegn.
- Python skiller mellom små og store bokstaver, så `Areal` og `areal` vil være to ulike variable.

Det anbefales å lage variabelnavn som er intuitivt forståelige, f.eks. er `areal` et bedre navn enn `x` på en variabel som inneholder et areal. Sammensatte variabelnavn skrives typisk som pukkelord (eng.: camelCase) eller med understrek for å vise hvor ett ord slutter og det neste begynner, f.eks. `startTime`, `pricePerLiter` eller `start_time`, `price_per_liter`, siden direkte sammensetning uten noe som helst skille vil gi lange variabelnavn som blir vanskelige å lese.

Kodeblokka under viser eksempler på variable som funker og ikke funker:

[ ]:
```
# Eksempel på tilordningssetninger som funker
pokemon_name = "Tyranitar"
MaxCP = 3670
antall = 3
antall = antall + 1      # høyre side regnes ut som 3+1, så 4 blir ny verdi i␣
↪variabelen antall
resists_fighting = False
level42 = "to be done"   # tall er OK i variabelnavn unntatt helt fremst

# Eksempel på tilordninger som IKKE funker
1 = antall               # variabelen må stå på venstre side
antall + 1 = antall      # og v.s. kan KUN inneholde et variabelnavn, ikke et␣
↪større uttrykk
10kamp = "gøy"           # variabel kan ikke begynne med tall, kun bokstav eller␣
↪_
antall = 3               # denne er OK, men se neste linje
antall = Antall + 1      # Python skiller mellom store og små bokstaver, Antall␣
↪vil være en annen
                         # variabel og gir NameError her fordi den ikke er␣
↪opprettet i en tidligere setning
```

```
happy hour = 20          # navn kan ikke inneholde mellomrom, burde vært␣
 ↪happy_hour eller happyHour
alkohol% = 4.5           # % kan ikke brukes i variabelnavn (betyr modulo).␣
 ↪Samme gjelder andre spesialtegn,
                         # hold deg til vanlige bokstaver og tall
```

1.8 d) Variabelnavn

Prøv å kjør koden under. Som du vil se, funker den ikke pga. diverse feil med variabelnavn og tilordningssetninger. Fiks feilene så programmet kjører som det skal.

[10]:
```
navn = "Per"
idealAlder = 42
kundensAlder = 37
differanse = idealAlder - kundensAlder
print(navn, "er", differanse, "år unna idealalderen")
```

```
Per er 5 år unna idealalderen
```

1.9 e) Variabel-program

Lag et program i kodeblokka under som først lagrer navnet ditt i en variabel og alderen din i en annen variabel, for så å printe det ut med `print()`-funksjonen. Her trenger du altså ikke å bruke `input()`-funksjonen!

Eksempel på kjøring:

```
Jeg heter Bob Bernt, og er 46 år.
```

Skriv koden din i blokka under.

[14]:
```
navn = "Ole"
alder = 54
print("Jeg heter", str(navn) + ", og er ", alder, " år.")
```

```
Jeg heter Ole, og er  54  år.
```