Variabler

...

Hva er en variabel?

- Et variabelnavn som peker på en verdi
- Vi assosierer et navn med en verdi vi har lyst til å lagre
- Husk at alle variabelnavn må være i ett ord!

Eksempler

- Siden alle variabelnavn må være ett ord bruker vi i python understrek _ mellom ordene. Verdiene som er stringer kan være flere ord, “Karl” kunne godt vært “Karl Gustav”.

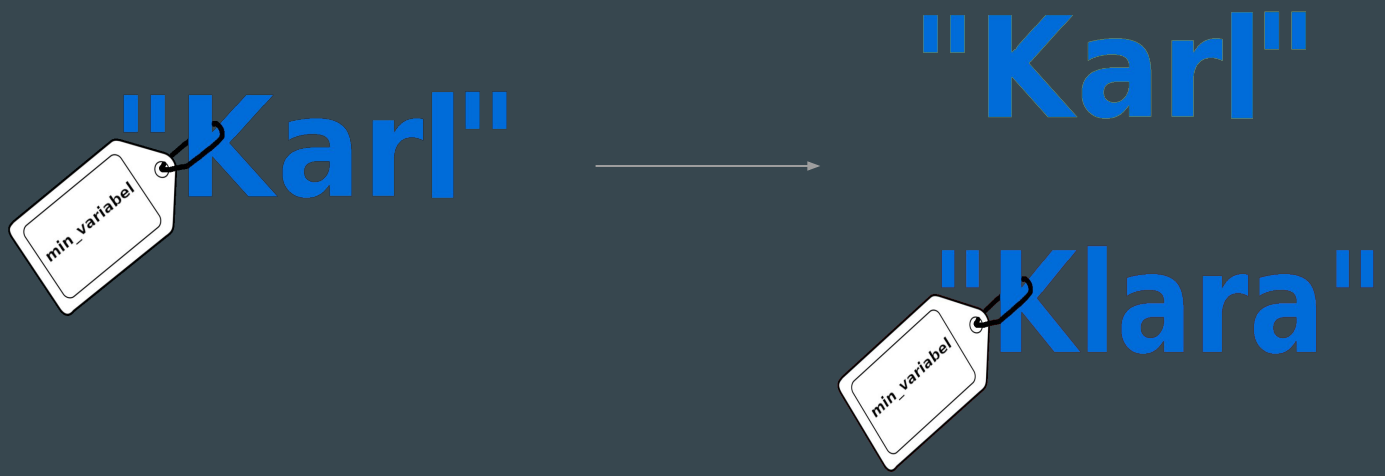

```
min_variabel = "Karl"
```

```
mitt_tall = 10
```

Gi en variabel ny verdi

Hvis vi endrer verdien til variabelen vår flyttes merkelappen til den nye verdien.

```
min_variabel = "Karl"
min_variabel = "Klara"
```



Typer

...

Hvilke typer har vi?

Hva slags type er verdiene våre?

- string (str): `"Dette er en string"`
- heltall (integer/int): `5 + 5 = 10`
- float: `5.2 + 5.5 = 10.7`
- boolean: `True`/`False`

String

```
min_variabel = "Her, mellom fnuttene, kan man skrive en tekst"
ny_variabel = "Teksten kan ha tall: 1, 4.5, det er fremdeles en string"
```

Alle tall kan gjøres om til stringer slik med funksjonen str():

```
str(5)   >>"5"
str(5.5)    >>"5.5"
```

Heltall (integer/int)

Heltall er tall uten desimal. 5.0, 5.5 og 5.14 er ikke heltall, 5 er et heltall.

Man kan gjøre om en string av et heltall til et heltall med funksjonen int():

```
int("5")   >>5
int("365")   >>365
```

Float

Float, eller flyttall på norsk, er tall med desimal. F.eks. 5.0, 5.5 og 5.14

Man kan gjøre om et flyttall i en string til flyttall:

```
float("5.5")  >>5.5
float("6.5")  >>6.5
```

Addere verdier

Man kan addere tall:

```
5 + 5      >>10
5.5 + 5      >>10.5
```

Og man kan addere stringer (det heter konkatenering):

```
"Hei " + "på deg " + "!"      >>“Hei på deg!”
```

Men man kan ikke addere dem sammen:

```
"Hei" + 5      >>ERROR
```

Boolean

Se for deg en lysbryter, enten er den av/`False` eller så er den på/`True`.

Så denne typen har bare to verdier: `False` og `True` (husk stor forbokstav)

Boolske uttrykk

...

Hva er et boolsk uttrykk?

Et boolsk uttrykk evaluerer til enten `True` eller `False`

Alle disse evaluerer til en boolean:

```
"a" == "a"   >>True
1 != 2       >>True
1 == "1"     >>False
1 < 1        >>False
1 <= 1       >>True
```

“!=” leses “er ikke lik”

“<” leses “er mindre enn”

“<=” leses “er mindre enn eller lik”

Not

Dersom et uttrykk er usant, vil not() av uttrykket være sant, og omvendt.

```
not(False) > True

not(True) > False

not(5 == 5) > False

not(5 > 1) > False

not(5 > 20) > True

not(5 == 1) > True
```

Diskuter hva disse blir:

```
not("a" == "a")   >>

not(1 != 2)       >>

not(1 == "1")     >>

not(1 < 1)        >>

not(1 <= 1)       >>
```

Sette sammen boolske uttrykk

Vi kan sette sammen flere boolske uttrykk, og vi skal lære to måter å gjøre det på:

1) And
2) Or

And

Begge uttrykkene må være sanne, både a og b!

```
False and False > False

False and True > False

True and True > True

(5 == 1) and (5 > 20)  > False

(5 == 1) and (5 > 1)  > False

(5 == 5) and (5 > 1)  > True
```

Or

Minst ett av uttrykkene må være sanne, enten a, eller b, eller begge!

```
False or False > False

False or True > True

True or True > True

(5 == 1) or (5 > 20)  > False

(5 == 1) or (5 > 1)  > True

(5 == 5) or (5 > 1)  > True
```

lf-tester

...

If-tester sjekker boolske uttrykk

Husk at alle boolske uttrykk evaluerer til enten `True` eller `False`! If testene sjekke boolske uttrykk og utfører koden hvis det boolske uttrykket er `True`.

```
if(True):
   #Do this
elif(True):
   #Do this
else:
   #Do this

#Man kan også ha en if-sjekk uten en
elif-sjekk eller else-sjekk:

if(True):
   #Do this
```

Eksempler

```
if(5 < 1):
   print("Hei på deg!")
else:
   print("Hade!")

# “Hade!” printes alltid
```

```
if(5 == 1):
   print("Hei på deg!")
else:
   print("Hade!")

# “Hei på deg!” printes alltid
```