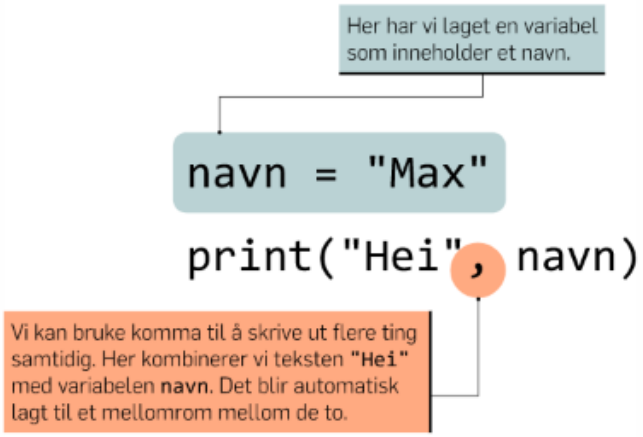
Variabler

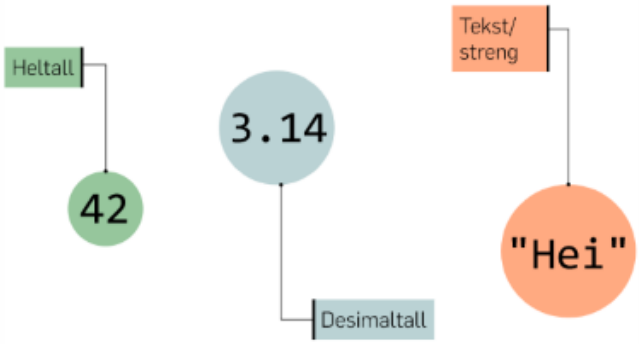
Varibler utgjør en viktig del av alle programmer. La oss først se på et eksempel.



Datatyper

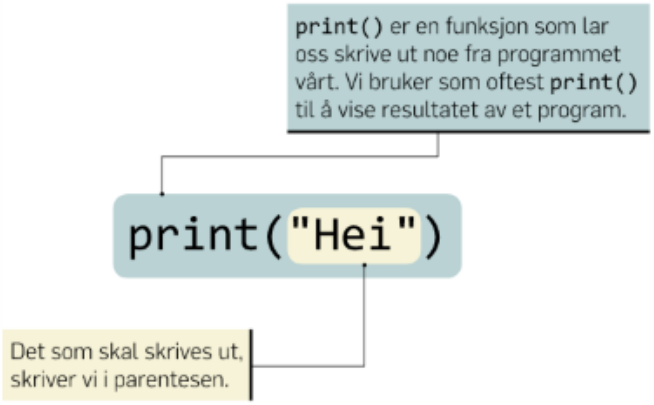
Vi har allerede nevnt at Python skiller mellom tekst og tall. Vi sier at tekst og tall er to ulike *datatyper*. For å angi en tekst bruker vi anførselstegn, mens tallene angir vi uten anførselstegn.

Det finnes to ulike typer tall: heltall og flyttall (desimaltall). For å angi et flyttall bruker vi punktum som desimaltegn (ikke komma).



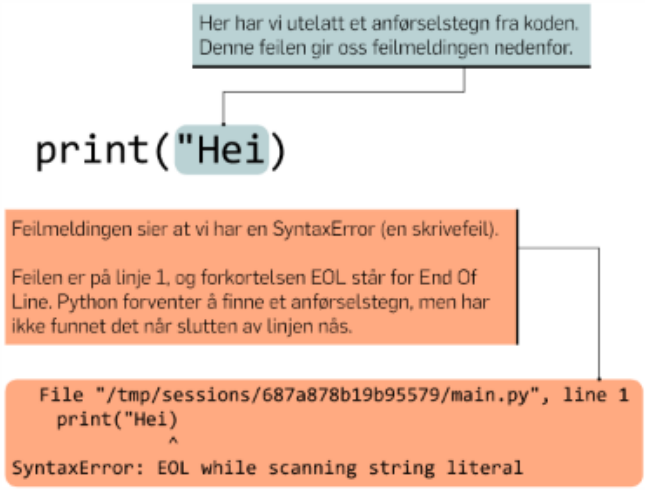
print()

La oss starte med det aller enkleste programmet vi kan lage: et program som «sier hei». Prøv ut koden nedenfor ved å trykke på «Run».



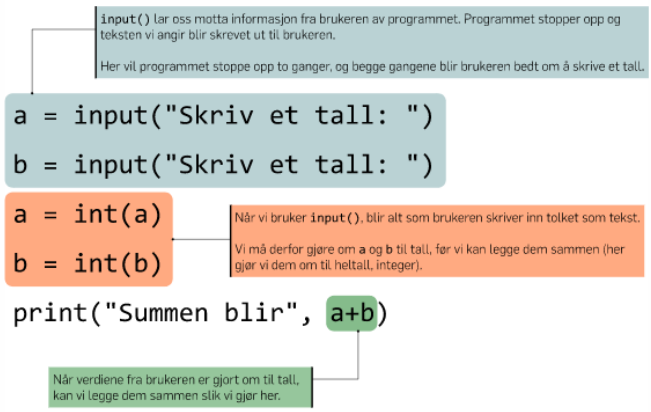
Feilmeldinger

Når vi programmerer, er det viktig å lese og tolke feilmeldinger. De ser kanskje litt skremmende ut, men ofte inneholder de verdifull informasjon om feilen.



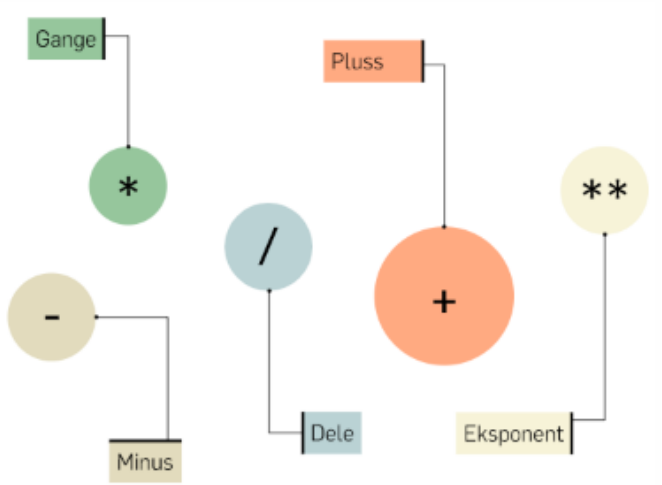
Informasjon fra bruker (input)

Ofte vil det være nyttig å la brukeren av et program skrive inn sine egne verdier. Vi kan se på et enkelt eksempel der brukeren skriver inn navnet sitt, og programmet skriver en tilpasset hilsen.



Regneoperasjoner

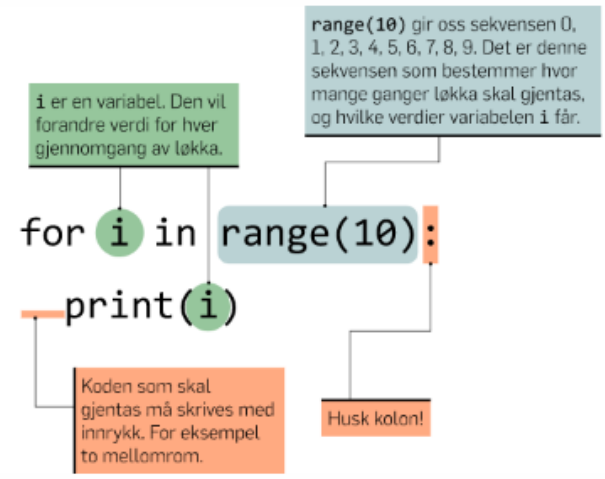
Vi har allerede sett på hvordan vi kan legge sammen tall med Python. I tabellen nedenfor finner du en oversikt over regneoperatorene vi kan bruke i Python.



`+`	Pluss	`a + b`
`-`	Minus	`a - b`
`*`	Gange	`a * b`
`/`	Dele	`a / b`
`**`	Eksponent	`a**b`
`%`	Modulus (rest ved divisjon)	`17 % 5` (gir 2)
`//`	Heltallsdivisjon	`17 // 5` (gir 3)

for-løkker

Vi bruker en `for`-løkke når vi eller programmet vårt vet hvor mange gjentakelser vi ønsker. I koden nedenfor kan du se en `for`-løkke som skriver ut tallene 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9.

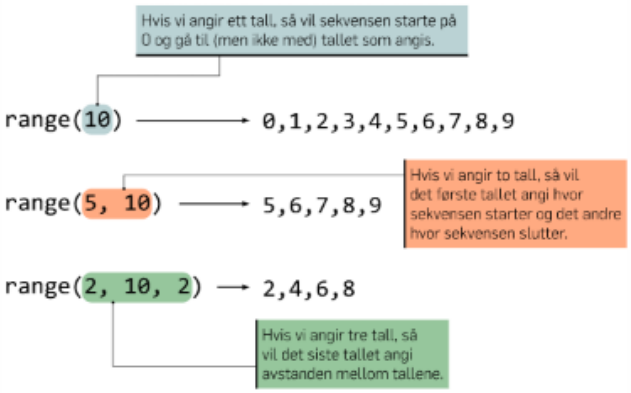



```
for x in range(10):
  print(x)

for x in range(5,10):
  print(x)

for x in range(2,21,2):
  print(x)
```

range()

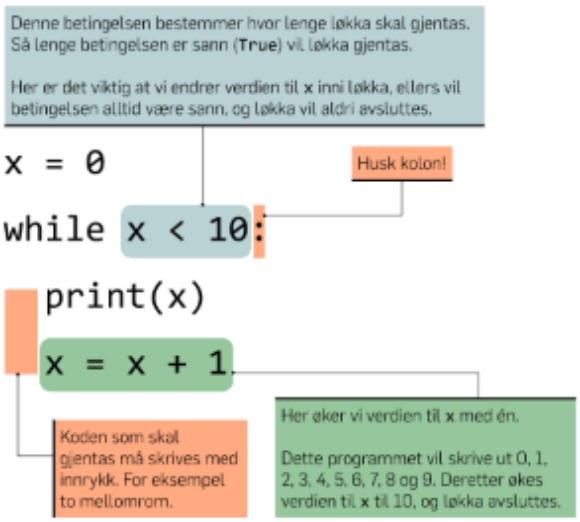
Vi kan bruke `range()` på tre ulike måter:

- Med bare ett tall i parentesen, angir vi tallet sekvensen skal gå opp til (men ikke inkludere). Sekvensen vil da begynne på 0 og øke med én av gangen.
- Med to tall i parentesen, så vil det første tallet angi tallet vi skal begynne på, og det andre tallet er tallet sekvensen skal gå opp til (men ikke inkludere).
- Med tre tall i parentesen, så får vi samme resultat som med to tall, bortsett fra at det tredje tallet angir avstanden mellom hvert tall (altså hvor mye vi skal øke med for hvert tall).



while-løkker

Ordet «while» kan oversettes til «imens» eller «så lenge». En `while`-løkke utfører en bestemt oppgave *så lenge* en gitt betingelse er sann. La oss se på et enkelt eksempel.

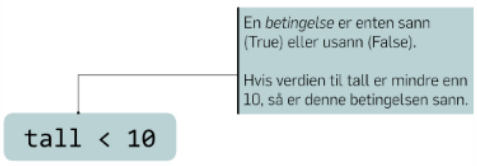



```
x = 0

while x < 10:
  print(x)
  x = x + 1
```

Betingelser

For å programmere med valg, må vi først se på *betingelser*. Det er betingelsene som styrer valgene.

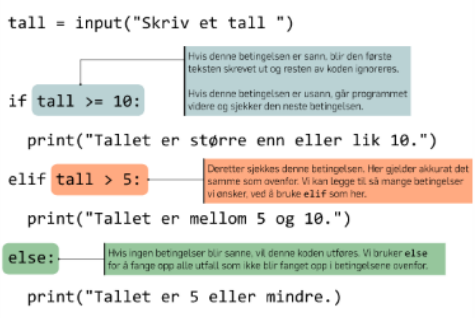
I programmering er en betingelse noe som enten er sant (`True`) eller usant (`False`). Nedenfor kan du se noen eksempler på betingelser og hvilke resultater de gir. Du kjenner kanskje igjen noen fra `while`-løkkene vi så på tidligere?



Valg (if-setninger)

Med betingelser på plass, kan vi se på *valg*. Sammen med løkker utgjør valg kjernen av de fleste programmer. En løkke lar oss gjenta noe, mens et valg lar oss bruke betingelser for å kontrollere hvilken kode som utføres. Vi bruker det vi kaller en `if`-setning for å programmere valg.

Vi starter med å se på et enkelt eksempel:




```
tall = input("Skriv et tall ")
tall = int(tall)

if tall > 10:
  print("Tallet er større enn 10.")
```

```
# input, lagrer som tekst
navn = input("Hva heter du?")

# Skriver ut navn
print("Hei " + navn)

# input, lagrer i variabler av typen int
a = int(input("Tast inn tall a: "))
b = int(input("Tast inn tall b: "))

# test om hvilket tall som er størst
if a > b:
    print("a er større enn b")
elif a == b:
    print("a og b er like")
else:
    print("a er mindre enn b")
```

Valg og løkker

Når vi kombinerer valg og løkker, kan vi begynne å lage ganske avanserte programmer. Prøv ut koden nedenfor. Hva tror du den gjør? Husk at modulus-operatoren (`%`) gir *resten* ved deling.

```
for x in range(100):
    if x**2 > 20:
        print(x)
```

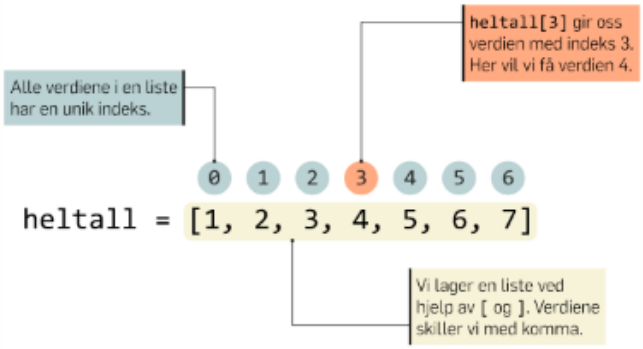
Vi kan plassere if-setninger inni løkker, og vi kan plassere løkker inni if-setninger. Det gir mange muligheter.

Her skriver vi ut alle de tallene som opphøyd i annen er større enn 20.

Operator	Beskrivelse	Eksempel
`==`	Lik	`4 == 4` gir `True`
`!=`	Ulik	`4 != 4` gir `False`
`<`	Mindre enn	`4 < 8` gir `True`
`>`	Større enn	`4 > 8` gir `False`
`<=`	Mindre enn eller lik	`5 <= 5` gir `True`
`>=`	Større enn eller lik	`6 >= 7` gir `False`
`and`	Flere betingelser må være sanne samtidig	`a < b and b < c`
`or`	Det holder at én av betingelsene er sann	`a < b or b > c`

Lister

En liste minner om en variabel, men i stedet for å lagre én verdi, kan vi lagre et stort antall verdier i en liste. Hver av verdiene får en fast posisjon (indeks), som vi kan bruke for å hente dem ut eller redigere dem. Legg merke til at posisjonene/indeksene begynner på null.

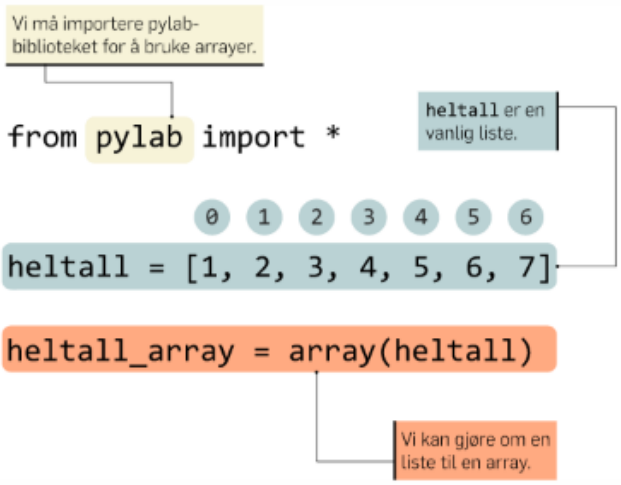



```
desimaltall = [2.5, 2.718, 3.14, 4.2, 9.3]

print(desimaltall[0])
print(desimaltall[4])
```

Arrayer

Lister har sin styrke når vi ønsker å fylle dem opp ved å bruke `append()` slik vi gjorde på forrige side. Vi skal se på et godt eksempel på neste side, der vi fyller opp en liste med tilfeldige verdier («terningkast»).

Hvis vi derimot ønsker oss en liste som er ferdig utfylt, spesielt hvis vi ønsker desimaltall, så er det enklere å bruke en array. En array er på mange måter som en liste, men den gir oss flere muligheter når vi ønsker å jobbe med matematiske problemstillinger.

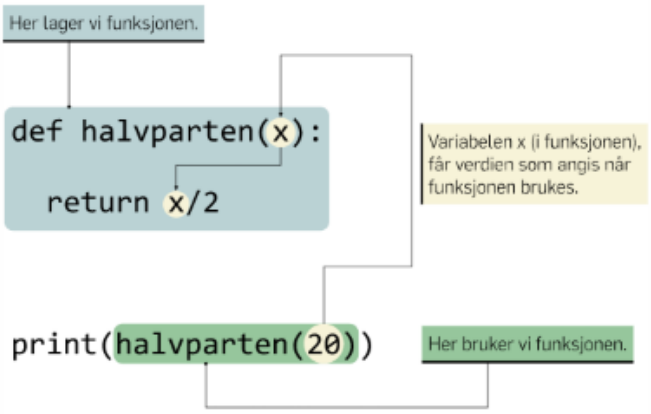



```
from pylab import * # henter inn biblioteket pylab

heltall = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] # lager en vanlige liste
print(heltall)

heltall_array = array(heltall) # gjør om listen til en array
print(heltall_array)
```

Funksjoner

Vi skal først se på noen generelle funksjoner. Kort fortalt bruker vi funksjoner i programmering til å forenkle kode. Vi kan for eksempel «lagre» en algoritme som vi bruker mange ganger i en funksjon. Et eksempel på det er `sqrt()`, en funksjon i pylab som regner ut kvadratroten av et tall.

I figuren til høyre og koden nedenfor kan du se en enkel funksjon. Kan du se hvordan koden fungerer?




```
def halvparten(x):
  return x/2

print(halvparten(20))
```

Matematiske funksjoner

Vi har sett at funksjoner i Python kan brukes til mye forskjellig, men vi kan også bruke dem på en måte som minner om funksjonene vi bruker i matematikken. Figuren til høyre viser for eksempel funksjonen $f(x) = 2x + 1$ som en Python-funksjon. Vi har gitt den navnet `f` for at den skal minne om den opprinnelige funksjonen, men den kan fint ha et lengre navn om vi ønsker det.