0.1 Introduksjon til Python

```python
print("Hello world!")
```

Output

```
Hello world!
```

Python deler talll inn i tre typer:

`int`	reelle heltall
`float`	relle tall
`complex`	komplekse tall

Vi skal i denne boka konsentrere oss om `int` og `float`. Tallypene definerer vi ved å ekskludere eller inkludere punktum:

```python
a = 3 # a er av typen int
e = -5 # e er av typen int
b = 2.8 # b er av typen float
c = 2. # c=2.0, og er av typen float
d = .7 # d=0.7, og er av typen float
f = -0.01 # f er av typen float
```

```python
a = 5
b = 2

print("a+b = ", a+b);
print("a-b = ", a-b);
print ("a*b = " ,a*b);
print ("a/b = " ,a/b);
print("a**b = " ,a**b); # potens med grunntall a og
                         # eksponent b
print ("a//b = ", a//b); # 5/2 rundet ned til nærmeste
                          # heltall
print ("a%b = ", a%b); # resten til a//b
```

Output

```
a+b = 7
a-b = 3
a*b = 10
a/b = 2.5
a**b = 25
a//b = 2
a%b = 1
```

Lister

Lister kan vi bruke for å samle objekter. Objektene som er i listen kalles elementene til listen.

```
strings = ["98", "99", "100"]
floats = [1.7, 1.2]
ints = [96, 97, 98, 99, 100]
mixed = [1.7, 96, "100"]
empty = []
```

Elementene i lister er indekserte. Første objekt har indeks 0, andre objekt har indeks 1 og så videre:

```
strings = ["98", "99", "100"]
floats = [1.7, 1.2]
ints = [96, 97, 98, 99, 100]
mixed = [1.7, 96, "100"]
empty = []
```

Output

```
96
99
98
```

Med den innebygde funksjonen `append()` kan vi legge til et objekt i enden av listen. Dette er en innebygd funksjon[1], som vi skriver i enden av navnet på listen, med et punktum foran.

```
min_liste = []
print(min_liste)

min_liste.append(3)
print(min_liste)

min_liste.append(7)
print(min_liste)
```

Output

```
[]
[3]
[3, 7]
```

[1]Kort fortalt betyr det at det bare er noen typer objekter som kan bruke denne funksjonen.

Med funksjonen `pop()` kan vi hente ut et objekt fra listen

```
min_liste = [6, 10, 15, 19]

a = min_liste.pop() # a = det siste elementet i listen
print("a =",a)
print("min_liste =",min_liste)

a = min_liste.pop(1) # a = elementet med indeks 1
print("a =",a)
print("min_liste =",min_liste)
```

Output

```
a = 19
min_liste = [6, 10, 15]
a = 10
min_liste = [6, 15]
```

Forklar for deg selv

Hva er forskjellen på å skive `a = min_liste[1]` og å skrive `a = min_liste.pop(1)`?

`for`-looper

For objekter som inneholder flere elementer, kan vi bruke `for`-looper til å utføre handlinger for hvert element. Handlingene må vi skrive med et innrykk etter `for`-uttrykket:

```python
min_liste = [5, 10, 15]

for number in min_liste:
  print(number)
  print(number*10)
  print("\n") # lager et blankt mellomrom

```

Output

```
5
50

10
100

15
150
```

Språkboksen

Å gå gjennom hvert element i (for eksempel) en liste kalles å iterere over listen.

`print(x)`	Skriver x til terminal.
`range(a)`	Lager en følge som starter på 0, og øker med 1 fram til $a-1$ er nådd. Følgen kan indekseres som en liste.
`for x in a`	Itererer over hvert element i a. a kan være en liste, et array eller en følge laget av `range`

Obs!

Bruk aldri innebygde funksjoner som navn på variabler.

0.2 Newtons metode

Gitt en funskjon $f(x)$ og likningen

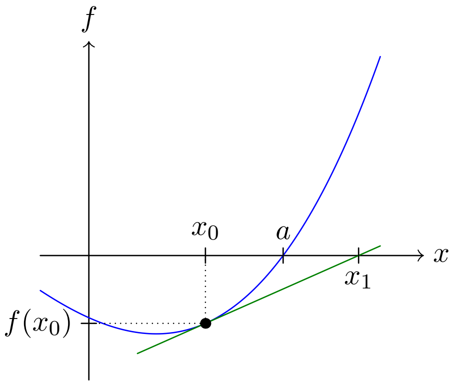
$$f = 0$$

hvor $f(a) = 0$. Ved Newtons metode gjør vi denne antakelsen for å en tilnærming a:

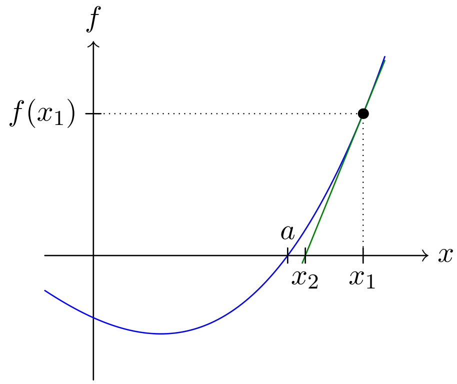
> La x_1 være skjæringspunktet mellom x-aksen og tangenten til f i x_0. Vi antar da at $|x_1 - a| < |x_0 - a|$. Sagt med ord antar vi at x_1 gir en bedre tilnærming for a enn det x_0 gjør.

Siden x_1 er skjæringspunktet mellom x-aksen og tangenten til f i x_0, har vi at[1]

$$\begin{aligned} f'(x_0)(x_1 - x_0) + f(x_0) &= 0 \\ f'(x_0)x_1 &= f'(x_0)x_0 - f(x_0) \\ x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$



(a)



(b)

La x_2 være skjæringspunktet mellom x-aksen og tangenten til f i x_1. Ved å gjenta prosedyren vi brukte for å finne x_1, kan vi finne x_2, som vi antar er en enda bedre tilnærming for a enn x_1. Prosedyren kan vi gjenta fram til vi har funnet en x-verdi som gir en tilstrekkelig[2] tilnærming til a.

---

[1] Se oppgave??

[2] Hva som er en *tilstrekkelig tilnærming* er det opp til oss selv å bestemme.

Regel 0.1 Newtons metode

Gitt en funskjon $f(x)$ og likningen

$$f = 0$$

hvor $f(a) = 0$. Gitt x-verdiene x_n og x_{n+1} for $n \in \mathbb{N}$. Ved å bruke formelen

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

antas det at x_{n+1} gir en bedre tilnærming for a enn x_n.

Språkboksen

Newtons metode kalles også Newton-Rhapsos metode.

0.3 Trapesmetoden

Gitt en funksjone $f(x)$. Integralet $\int_a^b f\,dx$ kan vi tilnærme ved å

1. Dele intervallet $[a, b]$ inn i mindre intervall. Disse kaller vi delintervall.
2. Finne en tilnærmet verdi for integralet av f på hvert delintervall.
3. Summere verdiene fra punkt 2.

I *figur 1a* har vi 3 like store delintervaller. Hvis vi setter $a = x_0$ og $\Delta x = \frac{b-a}{3}$, betyr dette at

$$x_1 = x_0 + \Delta x \qquad x_2 = x_0 + 2\Delta x \qquad x_3 = x_3 + 3\Delta x = b$$

En tilnæret verdi for $\int_a^{x_1} f\,dx$ får vi ved å finne arealet til trapeset med hjørner (husk at $x_0 = a$)

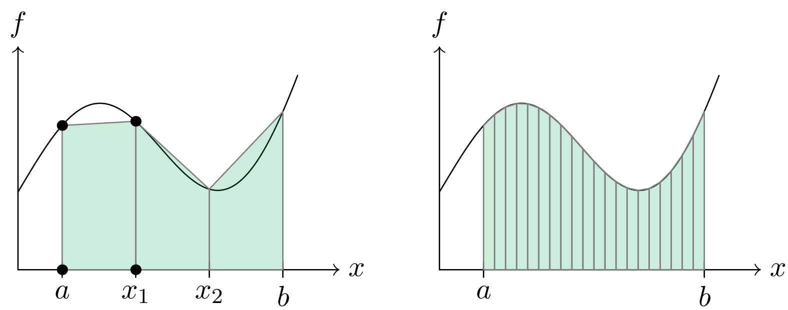
$$(x_0, 0) \qquad (x_1, 0) \qquad (x_1, f(x_1)) \qquad (x_0, f(a))$$

Dette arealet er gitt ved uttrykket

$$\frac{1}{2}(x_1 - x_0)\,[f(x_0) + f(x_1)] = \frac{\Delta x}{2}\,[f(x_0) - f(x_1)]$$

Ved å tilnærme integralet for hvert delintervall på denne måten, kan vi skrive

$$\int_a^b f\,dx \approx \frac{\Delta x}{2} \sum_{i=0}^{2} [f(x_i) + f(x_{i+1})]$$



(a) Tilnærming med 3 delintervaller.

(b) Tilnærming med 20 delintervaller

Figur 1

Regel 0.2 Trapesmetoden

Gitt en integrerbar funksjon f. En tilnærmet verdi for $\int_a^b f\,dx$ er da gitt som

$$\int_a^b f\,dx \approx \frac{\Delta x}{2} \sum_{i=0}^{n} [f(x_i) + f(x_{i+1})] \tag{1}$$

hvor

$$n \in \hat{\mathbb{N}}$$

$$a = x_0$$

$$b = x_n$$

$$\Delta x = \frac{b-a}{n+1}$$

$$x_{n+1} = x_n + i\Delta x$$

Merk

Slik *regel 0.2* er formulert, vil $[a, b]$ være delt inn i $n+1$ delintervaller.