

Mer om funksjoner (kap. 4)

Joakim Sundnes^{1,2}

Simula Research Laboratory¹

Universitetet i Oslo, Institutt for informatikk²

Sep 3, 2026

Plan for 3. september

- Live-programmering, funksjoner:
 - Oppg 3.23, 3.28, 3.25
- Mer om funksjoner
 - Returverdier (forts fra mandag)
 - Funksjoner som argumenter til funksjoner
 - Nyttige triks:
 - Lambda-funksjoner
 - doc-strenger
 - Rekursjon

Repetisjon fra mandag - funksjoner

Funksjonsdefinisjon:

```
def min_func(x, a, b, c):  
    (...) #setninger  
    return (...)
```

Funksjonskall:

```
z = min_func(0.5, 1, 1, 1)
```

Funksjoner kan returnere flere verdier

Vi ønsker å beregne både $y(t)$ og $y'(t) = v_0 - gt$:

```
def yfunc(t, v0):  
    g = 9.81  
    y = v0*t - 0.5*g*t**2  
    dydt = v0 - g*t  
    return y, dydt
```

kall:

```
position, velocity = yfunc(0.6, 3)
```

- Objektene som skal returneres skilles med komma (egentlig returneres en *tuppel*).
- Variabler som skal tilordnes skilles med komma

Funksjoner trenger ikke ha en return-setning

```
def somefunc(obj):  
    print(obj)  
  
return_value = somefunc(3.4)
```

Her får variabelen `return_value` verdien `None` fordi funksjoen ikke har en return-setning. Da setter Python automatisk inn `return None`.

Eksempel på en funksjon uten returverdi

Skriv ut verdier til skjermen:

```
def ball_table():  
    v0 = 5; g = 9.81  
    a, b = 0, 2 * v0 / g  
  
    n = 5  
    h = (b - a) / n  
  
    for i in range(n + 1):  
        t = a + h * i  
        y = v0 * t - 0.5 * g * t**2  
        print(f"{t:5.5f} {y:5.5f}")  
  
ball_table()
```

- Ingen grunn til å returnere noe her - formålet er å skrive ut.
- Funksjoner som ikke returnerer noe kalles gjerne *prosedyrer*. I Python-programmering er ikke dette skillet veldig viktig.

Python-funksjoner som argumenter til Python-funksjoner

- Argumenter til funksjoner kan være alle slags Python-objekter, også andre funksjoner
- Ofte veldig nyttig i matematikk, for å lage funksjoner som kan løse likninger, derivere, integrere o.l.:
 - numerisk integrasjon: $\int_a^b f(x)dx$
 - numerisk derivasjon: $f'(x)$
 - numerisk løsning av likninger: $f(x) = 0$
- Alle tre tilfeller trenger f implementert som en Python-funksjon $f(x)$

Eksempel: beregning av den deriverte

Vi kan estimere den deriverte til en funksjon numerisk:

$$f'(x) \approx \frac{f(x - h) - f(x)}{h}$$

Oppgave: Skriv en Python-funksjon som tar en funksjon som argument, og bruker denne formelen til å regne ut og returnere den deriverte.

Løsning på eksempel fra forrige slide

Definisjon av funksjonen:

```
def diff(f, x, h=1e-6):  
    r = (f(x+h) - f(x)) / h  
    return r
```

Bruk:

```
def g(t):  
    return t**(-6)  
  
dgdt = diff(g, 1.0) #derivert av g ved t = 1.0
```

Lambda-funksjoner for kompakte inline-funksjonsdefinisjoner

```
def f(x):  
    return x**2 - 1
```

En *lambda-funksjon* definerer den samme funksjonen på én linje:

```
f = lambda x: x**2 - 1
```

Generelt er

```
en_func = lambda a1, a2, ...: et_uttrykk
```

ekvivalent med

```
def en_func(a1, a2, ...):  
    return et_uttrykk
```

Lambda-funksjoner kan brukes direkte som argumenter i funksjonskall:

```
verdi = en_annen_func(lambda x, y, z: x+y+3*z, 4)
```

Eksempel på bruk av en lambda-funksjon for å spare skrivning

Standard funksjonsdefinisjon:

```
def g(t):  
    return t**(-6)  
  
dgdt = diff(g, 2)  
print(dgdt)
```

Mer kompakt kode med lambda:

```
dgdt = diff(lambda t: t**(-6), 2)  
print(dgdt)
```

Doc-strenger brukes til å dokumentere bruken av en funksjon

Viktig Python-konvensjon:

Dokumenter formålet med en funksjon, dens argumenter og dens returverdier i en *doc-streng* - en (trippel-anførselstegn-merket) streng skrevet rett etter funksjonsdefinisjonen.

```
def C2F(C):  
    """Konverterer Celsius-grader (C) til Fahrenheit."""  
    return (9.0/5)*C + 32  
  
def line(x0, y0, x1, y1):  
    """  
    Beregner koeffisientene a og b i det matematiske  
    uttrykket for en rett linje  $y = a*x + b$  som går  
    gjennom to punkter (x0, y0) og (x1, y1).  
  
    x0, y0: et punkt på linjen (floats).  
    x1, y1: et annet punkt på linjen (floats).  
    return: a, b (floats) for linjen (y=a*x+b).  
    """  
    a = (y1 - y0)/(x1 - x0)  
    b = y0 - a*x0  
    return a, b
```

Rekursjon - funksjoner kan kalle på seg selv

- En funksjon kan kalle på seg selv. Dette kalles en rekursiv funksjon, og er nyttig for mange oppgaver.
- Eksempel: regn ut fakultet til et tall n

$$n! = n \cdot (n - 1) \cdot \dots \cdot 1$$

- Utregningen kan gjøres med en løkke eller med rekursjon.
- Oppgave 3.25 i "A primer on ..."