Henning Bueie

Programmering i **matematikk** og **statistikk**

For økonomifag



FAGBOKFORLAGET

Programmering i matematikk og statistikk

For økonomifag

Henning Bueie

Programmering i matematikk og statistikk

For økonomifag

FAGBOKFORLAGET

Copyright © 2025 by
Vigmostad & Bjørke AS
All Rights Reserved

ISBN: 978-82-450-5957-1

1. utgave, 1. versjon 2025
E-utgaven er basert på 1. trykte utgave, 1. opplag: ISBN 978-82-450-5325-8

Elektronisk tilrettelegging: John Grieg: Bergen
Forsidedesign: Endre Barstad

Spørsmål om denne boken kan rettes til:
Fagbokforlaget
Kanalveien 51
5068 Bergen
Tlf.: 55 38 88 00
e-post: fagbokforlaget@fagbokforlaget.no
www.fagbokforlaget.no

Materialet er vernet etter åndsverkloven.
Uten uttrykkelig samtykke er eksemplarfremstilling
bare tillatt når det er hjemlet i lov eller avtale med Kopinor.

Vigmostad & Bjørke AS er Miljøfyrtårn-sertifisert,
og bøkene er produsert i miljøsertifiserte trykkerier.

Miljøfyrtårn®
SERTIFISERT VIRKSOMHET

Forord

Arbeidet med denne boka har vært givende og lærerikt. Til tider har jeg virkelig kost meg. Håpet mitt er at boka du nå sitter og leser, skal være et bidrag til å gjøre matematikk og statistikk enda mer tilgjengelig for deg. Om jeg har lykkes, må nesten du som leser vurdere selv. Husk uansett på at hvis du sitter og drar deg i håret fordi du strever med en vanskelig oppgave, så har jeg kun hatt gode intensjoner.

Først og fremst vil jeg takke studentene mine ved bachelorstudiet i økonomi og administrasjon på Lillehammer. De holder ut år etter år og har testet ut en god del av oppgavene og eksemplene. Videre vil jeg takke forskningsgruppen i Business analytics ved Universitetet i Innlandet for nyttige og konstruktive tilbakemeldinger i skriveprosessen.

Takk til redaktør Jannicke Bærheim, som hadde tro på prosjektet og har gitt verdifulle tilbakemeldinger underveis, og takk til Terje Idland, som har vært konsulent for boka.

Til slutt vil jeg takke Kornelia, Viktoria, Magnus, Johannes og Elisabeth for at dere lar meg holde på med en av hobbyene mine. Det går jo med noen timer på disse bokprosjektene.

Alt godt.
Henning
Lillehammer, april 2025

Innhold

Introduksjon 9

KAPITTEL 1
Programmering i matematikk og statistikk for økonomifag 11
1.1 Programmeringsspråket Python og utviklingsmiljøet Spyder 12
1.2 Variabler og datatyper 14
1.3 Regneoperatorer, formler og regnerekkefølger 15
1.4 Algebra med *sympy* 17
1.5 Ligningsløsning med *sympy* 19

KAPITTEL 2
Kontroll- og datastrukturer 22
2.1 *if - elif - else* 23
2.2 Andregradsligningen 24
2.3 *for*-løkker og lister 26
2.4 *while*-løkker 28

KAPITTEL 3
Funksjoner og derivasjon 31
3.1 Plotting av grafer 32
3.2 Regresjon med *polyfit* 38
3.3 Derivasjon med *sympy* 39
3.4 Derivasjon med utgangspunkt i definisjonen av den deriverte 41

KAPITTEL 4
Logaritmer og eksponentialfunksjoner 44
4.1 Tallet *e* og kontinuerlig forrentning 45
4.2 Eksponential- og logaritmefunksjoner 46
4.3 Regresjon med *curve_fit* 49
4.4 Logaritme- og eksponentialfunksjoner med *sympy* 51

KAPITTEL 5
Følger, rekker og integralregning 54
5.1 Aritmetiske følger og rekker 55
5.2 Geometriske følger og rekker 56
5.3 Finansmatematikk og formatering av utskrift 58
5.4 Ubestemte og bestemte integraler med *sympy* 66
5.5 Rekker vs. bestemte integraler 69

KAPITTEL 6
Flervariabelanalyse 71
6.1 Funksjoner av to variabler 72
6.2 Nivålinjer 76
6.3 Partiell derivasjon med *sympy* 80
6.4 Klassifisering av stasjonære punkter 82
6.5 Optimering under bibetingelse 84

KAPITTEL 7
Deskriptiv statistikk 88
7.1 Sentral- og spredningsmål 89
7.2 Innlasting av data med *pandas* via Excel 92
7.3 Diagrammer 94
7.4 Kovarians og korrelasjon 98
7.5 Standardavvik, kovarians og korrelasjon via *for*-løkker 101

KAPITTEL 8
Simulering og sannsynlighetsfordelinger 103
8.1 Tilfeldig trekning av hele tall og de store talls lov 104
8.2 Utvalgsstandardavvik vs. populasjonsstandardavvik 107
8.3 Trekning av tilfeldige flyttall og simulering av binomiske modeller 110
8.4 Normalfordelte trekkesituasjoner 113
8.5 Sentralgrenseteoremet 116
8.6 Hypotesetesting – binomisk modell 118
8.7 Hypotesetesting – normalfordelt modell 121
8.8 Korrelasjon vs R-kvadrat med simulering 124

Løsningsforslag 127
Litteraturliste 188
Stikkordregister 189

Introduksjon

Programmering har i de siste årene blitt en obligatorisk del av opplæringen i matematikk i grunn- og videregående skole. I fagområdet økonomi har programmering alltid hatt mange bruksområder, og flere kommer stadig til. I denne boka skal du lære programmering på matematikkens og statistikkens premisser. Eller sagt på en annen måte, så skal du jobbe med matematikk og statistikk ved hjelp av programmering. Det vil si at de eksemplene og oppgavene du får prøve deg på her, først og fremst tar utgangspunkt i matematikk- og statistikkursene som du møter i økonomiutdanningen du tar. Boka inneholder derfor i liten grad ren gjennomgang av matematikk eller statistikk, da boka først og fremst skal fungere som en verktøybok og læringsstøtte i disse kursene. Dette er ikke tenkt som en bok du sitter og leser i som i en tradisjonell bok, men det er en bok du henter eksempler fra, legger inn på datamaskinen, kjører koden og ser hva som skjer. Deretter justerer du gjerne litt på koden med utgangspunkt i noen oppgaver fra boka eller andre steder, eller kanskje det er noe du har lyst til å finne ut mer om selv. I denne prosessen vil du som leser av boka forhåpentligvis få en grundigere forståelse for matematikk og statistikk, i tillegg til å få en grunnleggende innføring i programmering.

Du skal kunne jobbe med denne boka uten noen forkunnskaper i programmering, utover det du har møtt i grunn- og videregående opplæring. Boka er i hovedsak tenkt brukt i forbindelse med undervisning i grunnleggende matematikk og statistikk for økonomer, men den har en stikkordliste bakerst, slik at den også vil kunne fungere godt som et lite oppslagsverk.

Håpet er i alle tilfeller at boka vil gjøre det enklere for deg som student å lære matematikk og statistikk. Programmeringen du lærer, er uansett noe du nok vil få stor glede av senere.

KAPITTEL 1

Programmering i matematikk og statistikk for økonomifag

Programmering har vært en nødvendig ferdighet i moderne økonomi i mange tiår. Vi bruker programmering til å automatisere og optimalisere prosesser, analysere store datamengder og simulere økonomiske modeller. Programmering i økonomi kan brukes til å lage algoritmer for handel på finansmarkeder, modellere økonomisk vekst, analysere konsumentatferd, lage kundebehandlingsapplikasjoner og mye mer. Programmering er også sentralt i felt som økonometri og maskinlæring, som bruker store datasett for å forstå økonomiske fenomener. Med økende mengder tilgjengelig data og voksende kompleksitet i økonomiske systemer er programmering en stadig viktigere ferdighet for økonomer.

De første datamaskinene ble bygd utelukkende for å gjøre beregninger. For å kunne utføre disse beregningene var en avhengig av å gi instruksjoner til datamaskinen om hva som skulle regnes ut, og på hvilken måte det skulles regnes ut. Slik sett har programmering og matematikk alltid vært nært knyttet sammen.

I det første kapittelet fokuserer vi på hva Python er, utviklingsmiljøet Spyder og hvordan vi kan bruke Python til grunnleggende matematiske beregninger. De økonomiske anvendelsene kommer i senere kapitler. Funksjonene og bibliotek-navnene du møter i kapittel 1, er samlet i denne tabellen.

Nytt i kapittel 1	
print()	skriver til skjerm
input()	leser fra skjerm til variabel
int()	heltallformaterer
float()	desimaltallformaterer
str()	tekstformaterer
type()	oppgir datatype
symbols, factor, expand, simplify	funksjoner fra *sympy*-biblioteket som brukes til behandling av uttrykk
symbols, Eq, solve	funksjoner fra *sympy*-biblioteket som brukes til ligningsløsning

1.1 Programmeringsspråket Python og utviklingsmiljøet Spyder

Programmeringsspråket Python har navnet sitt fra BBC-TV-serien «Monty Python and the flying circus», som var populært tidlig på 70-tallet. Utvikleren av Python ønsket et enkelt og litt spennende navn på programmeringsspråket.

Python er et populært programmeringsspråk som er i bruk både til opplæring og utvikling. Det er mange årsaker til dette. Språket er kjent for god lesbarhet, samtidig som at måten koden struktureres på er med på å definere hvordan den skal tolkes. Språket kjøres sekvensielt. Det betyr at koden tolkes linje for linje, noe som også er med på å gjøre språket lett å forstå. Dette gjør at nybegynnere får god veiledning i hvordan koden må bygges opp for at maskinen skal kunne tolke den. Python har et stort og aktivt utviklersamfunn, og det finnes derfor mange ferdigutviklede ressurser for både læring og problemløsning. Du kan finne store biblioteker av ferdig utviklet programkode som ligger åpent tilgjengelig, og som er i stadig utvikling. Disse bibliotekene dekker blant annet dataanalyse, maskinlæring, webutvikling, statistikk og matematikk.

I denne boka skal vi bruke utviklingsmiljøet Spyder til å kjøre kode vi lager i programmeringsspråket Python[1]. Spyder er en åpen kildekode primært designet for vitenskapelig programmering. Brukergrensesnittet inkluderer en editor med

1 Det finnes en rekke utviklingsmiljøer en kan kjøre Python-kode i.

syntaksframheving, avansert feilsøking og variabelutforsking. Spyder støtter også interaktiv testing og korrigering av biter av kode, og gir brukere muligheten til å utforske og redigere variabler direkte. I tillegg tilbyr Spyder integrasjon med mange populære vitenskapelige biblioteker, som NumPy, SciPy og Matplotlib. Det er et godt utviklingsmiljø for dem som jobber med dataanalyse, maskinlæring og vitenskapelig forskning.

Spyder laster du ned her:
https://www.spyder-ide.org/

Når du har lastet ned Spyder, kan du forsøke å lage ditt første lille program. Skriv inn koden som du ser i figur 1.1.

```
1 #mitt første program
2
3 print('Hallo verden')
```

Figur 1.1 og eksempel 1.1. Mitt første program.

```
In [1]: runfile('C:/programmering/eks 1.1 -
mitt forste program.py', wdir='C:/
programmering')
Hallo verden
```

Figur 1.2. Hallo verden.

Resultatet av kjøringen kommer opp i konsollvinduet som du ser i figur 1.2.

Vi har nå fått på plass verktøyet Spyder slik at vi er klare for å lære å programmere.

Oppgave 1.1

Skriv inn koden, og finn ut hva koden gjør.

```
1 #en hilsen
2
3 navn = input('Hva heter du?')
4 print('Hei', navn,'.','Håper du synes programmering er gøy.')
```

1.2 Variabler og datatyper

I matematikk dukker ordet variabel opp i forbindelse med algebra. Vi regner med variabler og lager oss uttrykk som vi kan bruke videre i beregninger. I programmering bruker vi variabler til å lagre innhold av et eller annet slag.

Det kan for eksempel være et heltall, et desimaltall, tekst eller annet.

I eksempel 1.2 har vi laget en kort tekst som forteller hva programmet handler om (linje 1). Kodeleseren skal ikke tolke denne linjen, derfor markerer vi den med # foran teksten. Videre tilordner vi heltallet 3 til variabel a (linje 3), desimaltallet 2.4 til variabel b (linje 4) og teksten 'litt tekst' til variabel c (linje 5). Merk at punktum brukes som desimalskilletegn, og at tekst må settes i apostrof `' '`[2].

I kodelinje 7, 8 og 9 skrives innholdet i de tre variablene til skjerm i motsatt rekkefølge.

Eksempel 1.2

```
1 #tilordning til variabler
2
3 a = 3
4 b = 2.4
5 c ='litt tekst'
6
7 print(c)
8 print(b)
9 print(a)
```

I matematikk snakker vi gjerne om naturlige tall, hele tall, rasjonale tall eller reelle tall. I programmering kategoriserer vi tallene vi skal jobbe med, i to grupper. Enten er det et heltall (engelsk: *integer*) eller det er et desimaltall (flyt-tall engelsk: *float*). I matematikk og statistikk vil det nok være mest relevant å bruke variablene til å lagre tall av forskjellig slag, men det er av og til behov for å lagre tekst også. Når vi lagrer tekst i en variabel i Python, lagres tekst i variabelen med en egen datatype som kalles *string*. *Integer*, *float* og *string* er eksempel på de tre mest brukte datatypene i programmering.

2 Det er også mulig å bruke hermetegn `" "`.

Oppgave 1.2

Juster koden i eksempel 1.2 ved å skrive `a = float(3)` i kodelinje 3 og `b = int(2.4)` i kodelinje 4 . Kjør programmet nå.
Hvilken datatype har variabel *a* og variabel *b* nå?

Du kan kontrollere dette med å for eksempel skrive

```
print(type(a))
```

i kodelinje 10 og kjøre programmet.

Oppgave 1.3

Eulers tall $e = 2{,}71828182845\ldots$ dukker opp i mange sammenhenger i økonomifag.
Lag et program der du lagrer tallet i en variabel (ta med 10 desimaler).
Skriv innholdet i variabelen til skjerm med *print()*-funksjonen.

1.3 Regneoperatorer, formler og regnerekkefølger

I matematikk har vi forskjellige regneoperatorer, og vi finner igjen de samme i Python. I tabell 1.1 har vi listet opp de mest aktuelle foreløpig.

Tabell 1.1.

Pluss	+
Minus	-
Gange	*
Dele	/
Opphøyd i	**
Kvadratrot	**(1/2)
Gir rest ved heltallsdivisjon	%

Vi bruker disse regneoperatorene til å sette sammen variablene til uttrykk og gjøre beregninger.

Vi ser på et eksempel der vi skal regne ut verdien av uttrykket $\frac{(a+(b+c)a)}{2a}$ og får oppgitt at $a = 3$, $b = 6$ og $c = 7$.

Eksempel 1.3

```
1 #regneoperatorer
2
3 a = 3
4 b = 6
5 c = 7
6
7 resultat = ((a+(b + c)*a)/(2*a))
8 print(resultat)
```

```
7.0
```

Vi ser at kjøring av programmet gir tallet 7 som resultat.

Oppgave 1.4

Lag et program som regner ut verdien av uttrykket

$$-(a+b)(a-2b)+3(a+b)(a+b)$$

når du får vite at $a = 2$ og $b = -3$.

Oppgave 1.5

Lag et program som regner ut verdien av uttrykket

$$\frac{\frac{2^{-7}}{x}\cdot\left(x^2\right)^5}{\left(\frac{1}{4}x\right)^2\cdot\frac{1}{8}}$$

når du får vite at $x = 3$.

Oppgave 1.6

Lag et program som tar utgangspunkt i uttrykket 2^n. Velg en passende verdi for n slik at programmet skriver et tall med 10 siffer til skjerm.

1.4 Algebra med *sympy*

Til nå har vi fokusert på å regne kun med verdier. Python har egne funksjonsbibliotek som gjør at det er mulig å regne vanlig algebra. Et slikt bibliotek er *sympy*. Når du skal bruke funksjoner fra de ulike bibliotekene, må de importeres. I linje 3 i eksempel 1.4 importerer vi biblioteket *sympy*. Foreløpig er vi interessert i den delen som kan regne med bokstaver (*symbols*), og den delen som kan forenkle uttrykk (*simplify*).

I eksempel 1.4 ser vi hvordan vi kan forenkle uttrykket $-(a-b)(a-2b)+3(a+b)^2$ med Python.

Eksempel 1.4

```
1  #algebra med sympy
2
3  from sympy import symbols, simplify
4
5  # Definer symbolene
6  a, b = symbols('a b')
7
8  # Definerer uttrykket
9  uttrykk = -(a-b)*(a-2*b) + 3*(a+b)**2
10
11 # Forenkle uttrykket
12 forenklet_uttrykk = simplify(uttrykk)
13
14 print(forenklet_uttrykk)
```

Vi importerer de nødvendige funksjonene (linje 3) og forteller Python hvilke bokstaver som skal behandles som symboler (linje 6). I linje 9 legger vi inn uttrykket, og selve forenklingen blir da foretatt i linje 12. Ved kjøring av programmet skrives da resultatet $2a^2+9ab+b^2$ til skjerm.

Av og til når vi regner algebra, blir vi sittende med uttrykk som vi ønsker å faktorisere.

Sympy-biblioteket kan brukes til å faktorisere uttrykk. Da må vi importere *factor*-funksjonen fra *sympy*-biblioteket (linje 3 i eksempel 1.5).

Eksempel 1.5

```
1  #faktorisering og utviding med sympy
2
3  from sympy import symbols, factor, expand
4
5  # Definer symbolene
6  x = symbols('x')
7
8  # Definer uttrykket
9  uttrykk = x**3 - 3*x**2 + 2*x
10
11 # Faktoriser uttrykket
12 faktorisert_uttrykk = factor(uttrykk)
13 print('Faktorisert uttrykk:',faktorisert_uttrykk)
14
15 #regner ut parentesen
16 utvidet_uttrykk = expand(faktorisert_uttrykk)
17 print('Utvidet uttrykk:',utvidet_uttrykk)
```

I linje 12 bruker vi funksjonen *factor* til å faktorisere uttrykket.

Resultatet av faktoriseringen blir $x(x-2)(x-1)$.

Merk at vi kan bruke Python til å kode motsatt vei også, altså for å multiplisere ut parentesene. I kodelinje 16 utvider vi uttrykket slik at vi kommer tilbake til utgangspunktet.

Oppgave 1.7

Lag et program som regner ut $(y+2z)(x+2z)$ og skriver resultatet til skjerm. Utvid programmet slik at $2xy+4xz+y^2+2yz$ legges til resultatet. Faktoriser resultatet nå, og skriv det til skjerm.

Oppgave 1.8

Lag et program som regner ut

$$64z^4+160z^3+132z^2+40z+4-\big((4z+1)(2+2z)\big)^2$$

1.5 Ligningsløsning med *sympy*

Vi støter ofte på ligninger av forskjellig slag i matematikk. Biblioteket *sympy* gjør det mulig å jobbe med ligninger og ligningsløsning. Vi skal nå se på noen eksempler der vi går gjennom hvordan vi kan løse en ligning eller et ligningssett med en eller flere ukjente og andre- eller tredjegradsligninger.

I eksempel 1.6 løser vi ligningen

$$2x - 3 = 7.$$

Fra *sympy*-biblioteket må vi hente funksjonene *symbols*, *Eq* og *solve* (linje 3).

I kodelinje 6 forteller vi Python hvilken bokstav vi vil bruke som ukjent. Ligningen legges inn i eq(linje 9). I kodelinje 12 løses ligningen, og resultatet skrives til skjerm i kodelinje 14.

Eksempel 1.6

```
1  #ligning med en ukjent
2
3  from sympy import symbols, Eq, solve
4
5  # Definer symbolet
6  x = symbols('x')
7
8  # Definer ligningen
9  eq = Eq(2*x - 3, 7)
10
11 # Løs ligningen
12 losning = solve(eq, x)
13
14 print(losning)
```

Løsningen på ligningen skrives i klammeform: [5]. Det er fordi den skrives på listeform. En liste er en datastruktur som gjør det mulig å lagre flere verdier samlet i en liste.

Det er ikke store endringen vi trenger å gjøre på koden for å få den til å løse andregrads- eller tredjegradsligninger. Den eneste endringen er hvilket uttrykk vi legger inn i kodelinje 9.

I eksempel 1.7 løser vi andregradsligningen $x^2 + 2x - 3 = 0$.