FINN AAKRE HAUGEN
OG MARIUS LYSAKER

HELT PYTHON

FAGBOKFORLAGET

FINN AAKRE HAUGEN
OG MARIUS LYSAKER

HELT
PYTHON

FAGBOKFORLAGET

Copyright © 2026 by
Vigmostad & Bjørke AS
All Rights Reserved

1. utgave 2026 / 1. versjon 2026

ISBN: 978-82-450-6008-9
E-utgaven er basert på 1. trykte utgave, 1. opplag: ISBN: 978-82-450-6007-2

Elektronisk tilrettelegging: John Grieg, Bergen
Forsidedesign ved forlaget: 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.

Miljøfyrtårn®
SERTIFISERT VIRKSOMHET

Innhold

1 Innledning **11**
- 1.1 Hvorfor lære å programmere? . . . 11
  - 1.1.1 Anvendelser . . . 11
  - 1.1.2 Undervisning . . . 12
  - 1.1.3 Bruk av kunstig intelligens i læringsprosessen . . . 13
- 1.2 Program med inndata og utdata . . . 14
- 1.3 Om Python . . . 15
  - 1.3.1 Python – kort fortalt . . . 15
  - 1.3.2 Hvorfor satse på Python? . . . 16
  - 1.3.3 Når er Python ikke aktuelt? . . . 16
  - 1.3.4 Hvem temmer Python? . . . 16
- 1.4 Skrivemåter i boken . . . 16
- 1.5 Om bokens struktur . . . 18
- 1.6 Lære mer? . . . 18
  - 1.6.1 Nettsteder . . . 18
  - 1.6.2 Nettbaserte kurs . . . 18
  - 1.6.3 Bøker . . . 19
- 1.7 Oppgaver til kapittel 1 . . . 19
- 1.8 Løsninger til kapittel 1 . . . 19

2 Programmeringsmiljøer **21**
- 2.1 Innledning . . . 21
- 2.2 Anaconda . . . 21
- 2.3 Spyder . . . 23
  - 2.3.1 Hvordan starte Spyder . . . 23
  - 2.3.2 Spyders brukergrensesnitt . . . 23
  - 2.3.3 Programmering i konsollen . . . 25
    - 2.3.3.1 Programmering kun med tall i konsollen . . . 25
    - 2.3.3.2 Programmering med variabler i konsollen . . . 27
  - 2.3.4 Programmering med skript . . . 28
  - 2.3.5 Innstilling av Spyder . . . 33
- 2.4 Hjelp . . . 33
  - 2.4.1 Hjelp om innebygde Python-funksjoner . . . 33
  - 2.4.2 Hjelp til å finne syntaksfeil i din programkode . . . 34
  - 2.4.3 Hjelp om funksjonsfeil . . . 34

2.4.4 Hjelp til å finne kode som du kan skrive i ditt program 35
2.5 Jupyter Notebook 35
2.5.1 Hvordan starte Jupyter Notebook 36
2.5.2 Hvordan lage og kjøre notebook-er 36
2.5.3 Noen nyttige tastaturkombinasjoner i Jypyter Notebook 38
2.5.4 Hvordan lagre notebook-er 39
2.5.5 Formatering av notebook-er 39
2.6 Skybasert programmeringsmiljø 40
2.7 Import og bruk av funksjonspakker og -moduler 41
2.7.1 Pakker og pakkebehandling med conda eller pip 41
2.7.1.1 Pakker 41
2.7.1.2 Pakkebehandling 42
2.7.2 Innebygde funksjoner i Python (standardpakken) 42
2.7.3 Import av pakker i Anaconda-distribusjonen 43
2.7.3.1 Opplisting av pakker som er installert på PC-en 43
2.7.3.2 Hvordan importere pakker som er installert på PC-en . 44
2.7.3.3 Hvordan importere moduler som inngår i pakker . . . 45
2.7.4 Installering og import av pakker som ikke følger med Anaconda-distribusjonen 46
2.8 Oppgaver til kapittel 2 46
2.9 Løsninger til kapittel 2 48

3 Variabler og datatyper **53**
3.1 Innledning 53
3.2 Variabler 53
3.2.1 Hva er en variabel? 53
3.2.2 Hvorfor bruke variabler når vi kunne brukt verdier direkte? . . . 54
3.2.3 Hvordan velge variabelnavn 55
3.3 Litt om funksjoner 56
3.4 Tall og grunnleggende matematiske operasjoner 58
3.4.1 Talltyper 58
3.4.2 Hvordan formatere tall i print-funksjonen 59
3.4.3 Matematiske operatorer 61
3.5 Tekststrenger (strings) 63
3.6 Fra tall til tekst og fra tekst til tall 64
3.7 Boolske verdier 66
3.8 Lister (lists) 67
3.8.1 Hva er lister? 67
3.8.2 Operasjoner på lister 68
3.8.2.1 Lese listeelementer 68
3.8.2.2 Hvordan oppdatere listeelementer med nye verdier . . 70
3.8.2.3 Utvide lister med nye elementer 71
3.8.2.4 Fjerne listeelementer 72
3.8.2.5 Listemanipulering med + og * 73
3.9 Tupler 73
3.10 Dictionaries 75
3.11 Arrayer 76

3.12 Oppgaver til kapittel 3 . . . 76
3.13 Løsninger til kapittel 3 . . . 80

4 Numpy-arrayer **93**
4.1 Innledning . . . 93
4.2 Hvordan konvertere lister til arrayer, og motsatt . . . 94
4.2.1 Konvertering fra liste til array . . . 94
4.2.2 Konvertering fra array til liste . . . 94
4.3 Lage arrayer med spesiell utforming . . . 95
4.3.1 Arrayer med like elementverdier . . . 95
4.3.2 Array med fast inkrement mellom elementene . . . 96
4.4 Flerdimensjonale eller n-dimensjonale arrayer . . . 98
4.5 Operasjoner på arrayer . . . 99
4.5.1 Innledning . . . 99
4.5.2 Størrelsen av en array . . . 99
4.5.3 Lese elementverdier i en array . . . 100
4.5.4 Oppdatere elementer i en array . . . 101
4.5.5 Utvide arrayer med nye elementer . . . 102
4.5.6 Fjerne elementer fra arrayer . . . 102
4.5.7 Finne maksimum og minimum i arrayer . . . 103
4.6 Beregninger med arrayer . . . 103
4.6.1 Addisjon med skalar og multiplikasjon med skalar . . . 103
4.6.2 Hvordan lage radvektorer og kolonnevektorer og matriser . . . 104
4.6.3 Vektor- og matrisemultiplikasjoner . . . 107
4.6.4 Matrisefunksjoner for lineæralgebra . . . 110
4.6.5 Vektoriserte beregninger . . . 111
4.7 Oppgaver til kapittel 4 . . . 114
4.8 Løsninger til kapittel 4 . . . 118

5 Plotting **129**
5.1 Innledning . . . 129
5.2 Kurveplott . . . 130
5.2.1 Et super-enkelt kurveplott . . . 130
5.2.2 Et skikkelig plott . . . 131
5.2.3 Flere enn ett figurvindu? . . . 134
5.2.4 Vise plott i Plots-fanen i Help-vinduet eller i eksternt vindu? . . . 134
5.2.4.1 Plottfigurer vist i Plots-fanen i Help-vinduet . . . 134
5.2.4.2 Plottfigur vist i et eksternt vindu . . . 135
5.2.5 Hvordan lagre figur på fil automatisk . . . 136
5.2.6 Subplott . . . 136
5.2.7 Hvordan sette størrelsen av figurvinduet . . . 137
5.3 Stolpediagram . . . 138
5.4 Kakediagram . . . 140
5.5 Histogram . . . 141
5.6 Objektorientert plottekoding . . . 142
5.7 Oppgaver til kapittel 5 . . . 143
5.8 Løsninger til kapittel 5 . . . 145

6 Funksjoner **151**
6.1 Innledning . . . 151
6.2 Programmering av funksjoner . . . 152
6.2.1 Hvordan definere funksjonen . . . 152
6.2.2 Hvordan returnere flere enn én verdi . . . 154
6.2.3 Posisjonsargumenter og keyword-argumenter . . . 155
6.2.4 Keyword-argumenter i ferdiglagde funksjoner i Python . . . 157
6.2.5 Dokumentasjonstekst (docstring) . . . 157
6.3 Navnerom (namespace) . . . 160
6.4 Programmering av moduler . . . 161
6.5 Oppgaver til kapittel 6 . . . 164
6.6 Løsninger til kapittel 6 . . . 167

7 Objektorientert programmering **177**
7.1 Innledning . . . 177
7.2 Hvorfor objektorientert programmering? . . . 177
7.3 Attributter og metoder . . . 178
7.4 Hvordan programmere med OOP . . . 179
7.5 Plassering av klassedefinisjonen i en modul . . . 182
7.6 Oppgaver til kapittel 7 . . . 183
7.7 Løsninger til kapittel 7 . . . 184

8 Testing av kode **189**
8.1 Innledning . . . 189
8.2 Hvordan teste for funksjonsfeil? . . . 189
8.3 Hvordan kjøre kun en del av programmet? . . . 192
8.4 Oppgaver til kapittel 8 . . . 192
8.5 Løsninger til kapittel 8 . . . 196

9 If-else **199**
9.1 Innledning . . . 199
9.2 Sammenlikningsoperatorer . . . 199
9.3 Logiske funksjoner . . . 200
9.4 if-else . . . 201
9.5 Et liv uten else . . . 203
9.6 elif . . . 204
9.7 Oppgaver til kapittel 9 . . . 205
9.8 Løsninger til kapittel 9 . . . 207

10 For- og while-løkker **213**
10.1 Innledning . . . 213
10.2 For-løkker . . . 214
10.2.1 Innledning . . . 214
10.2.2 Direkte iterasjon . . . 214
10.2.3 Indeksbasert iterasjon . . . 216
10.2.4 Nøstede løkker . . . 218
10.3 While-løkker med ulike fortsett-betingelser . . . 221

10.4 Refleksjon rundt for-løkker og while-løkker . . . 224
10.5 En anvendelse av løkker: Brute force-optimering . . . 225
10.5.1 Innledning . . . 225
10.5.2 Formulering av optimeringsproblemer . . . 225
10.5.3 Brute force-optimering . . . 227
10.5.4 Modelltilpassing med brute force-optimering . . . 231
10.6 Oppgaver til kapittel 10 . . . 240
10.7 Løsninger til kapittel 10 . . . 245

11 Lese og skrive fildata **265**
11.1 Innledning . . . 265
11.2 Filformater . . . 265
11.2.1 Tekstbaserte datafiler . . . 265
11.2.2 Binærdatafiler . . . 267
11.3 Lese fildata med loadtxt-funksjonen . . . 268
11.4 Skrive data til fil med savetxt-funksjonen . . . 269
11.5 Lese og skrive fildata med open-funksjonen . . . 270
11.5.1 Innledning . . . 270
11.5.2 Lese fildata med open-funksjonen . . . 271
11.5.3 Skrive fildata med open-funksjonen . . . 272
11.6 Excel-filer i Python . . . 273
11.6.1 Innledning . . . 273
11.6.2 Lese data fra Excel-filer . . . 273
11.6.3 Skrive data til Excel-filer . . . 275
11.7 Oppgaver til kapittel 11 . . . 276
11.8 Løsninger til kapittel 11 . . . 282

12 Søking etter tegnsekvenser med regex-pakken **297**
12.1 Introduksjon . . . 297
12.2 Søke i tekst etter gitte mønstre . . . 297
12.3 Mer anvansert søk – bruk av grupper . . . 301
12.4 Oppgaver til kapittel 12 . . . 303
12.5 Løsninger til kapittel 12 . . . 304

13 Blanda drops (oppgavesamling) **309**
13.1 Oppgaver . . . 309
13.2 Løsninger . . . 319

14 Hurtigreferanse **331**

Referanser **335**

Register **337**

Forord

Vi har skrevet denne boken for alle som vil lære å programmere i Python. Vi tror at boken vil være nyttig for studenter og ansatte ved universiteter, høgskoler og fagskoler, elever og lærere i videregående skoler, ansatte i bedrifter som vil lære programmering fra grunnen av, eller som allerede driver programmering i et annet programmeringsspråk og som vil lære seg et nytt språk.

Litt om vår bakgrunn:

- Finn: Jeg har hovedstilling som dosent ved Universitetet i Sørøst-Norge (USN), campus Porsgrunn, og bistilling som dosent ved OsloMet. Min utdanning er sivilingeniør i elektro/teknisk kybernetikk fra tidligere Norges tekniske høgskole (NTH) i 1986 og doktorgrad (ph.d.) fra tidligere Høgskolen i Telemark. Jeg har lang erfaring fra undervisning ved flere universiteter, høgskoler og fagskoler og fra kursvirksomhet for industrien. Jeg har brukt programmering til problemløsing innen teknisk kybernetikk siden 80-tallet.

- Marius: Jeg er professor i matematikk med simulering og modellering som hovedområder ved USN, campus Porsgrunn. Siden 2010 har jeg undervist i matematikk, algoritmisk tenkning og programmering i Python ved ingeniørstudiet. I perioden 2004–2012 jobbet jeg med matematiske modeller og utvikling av algoritmer ved Simula Research Laboratory i Oslo. Jeg har gjennom flere år vært partner i senteret for fremragende undervisning – Center for Computing in Science Education, Universitetet i Oslo. Jeg har en doktorgrad (ph.d.) i anvendt matematikk fra Universitet i Bergen (2004).

Programmering i Python inngår i emnene (fagene) vi underviser ved USN, og vi bruker Python i forskningsprosjekter. Vi har samarbeidet om mange kurs i Python-programmering de siste årene; kurs for realfagslærere i videregående skoler og kurs rettet mot industriansatte på oppdrag fra Direktoratet for høyre utdanning og kompetanse (HK-dir).

Vi kan med hånden på hjertet si at programmering beriker vår undervisning og forskning innen kybernetikk og matematikk. Med programmering får vi tatt i bruk teoretiske metoder til å løse teoretiske og praktiske problemer på en effektiv måte. Det fascinerer oss. Og vi har stor glede av å prøve å gi studenter noe av den samme opplevelsen. Uten disse positive opplevelsene som vi har av programmering, hadde det nok ikke blitt noen bok.

Python er blitt et av de mest populære programmeringsspråkene i verden. Det skyldes at Python er brukervennlig, kraftig og fritt tilgjengelig!

Python er et engelskspråklig programmeringsspråk i den forstand at terminologien i programmeringsmiljøene og i dokumentasjonen er på engelsk. Etter vår mening er det fornuftig å bruke den opprinnelige engelske terminologien og ikke finne opp norsk Python-terminologi, med mindre det faller naturlig. En alternativ «skyggeterminologi» kan skape misforståelser. Ellers bruker vi selvsagt norske ord og uttrykk der det er naturlig.

Et undervisningsopplegg for læring av Python-programmering må selvsagt inkludere øvingsoppgaver. Vi har lagt inn oppgaver med detaljerte løsninger i slutten av hvert kapittel.

Boken inneholder et stort antall komplette programmer, i eksempler og i løsningsforslag til oppgaver. Lenker til programmene er angitt i boken. Programmene er også tilgjengelige via bokens hjemmeside på https://techteach.no/python.

Vi har valgt å bruke samme fonttype og fontstørrelse på både vanlig tekst og pythonske (Python-spesifikke) navn og begreper i tekstdelene av boken. Det framgår av konteksten hva som er norsk og hva som er pythonsk. I kodesnutter og kodelistinger har vi imidlertid brukt typewriter-font.

Vi har tidligere utgitt boken *Python for realfag* på Fagbokforlaget – utgave 1, 2 og 3. *Helt Python* er en kraftig omarbeidet versjon: Vi har omarbeidet/forbedret flere av kapitlene, lagt inn et nytt kapittel om søking etter tegnsekvenser i tekst og redusert bokens omfang betydelig ved å *ikke* inkludere de tidligere omfattende kapitlene om ulike anvendelser.

Takk til alle som har bidratt med kommentarer og innspill til boken! En spesiell takk for prof. Maths Halstensen ved USN for hans innspill. Takk også til alle de positive folkene i Fagbokforlaget, spesielt fagredaktør Jannicke Bærheim.

Vi håper boken er lærerik og lett å lese. Dersom du har kommentarer til boken, kan du gjerne sende dem til oss i e-post.

Med hilsen

Finn Aakre Haugen
finn.haugen@usn.no

Marius Lysaker
marius.lysaker@usn.no

Kapittel 1

Innledning

1.1 Hvorfor lære å programmere?

Det kan være gode grunner for å lære å programmere, enten du er student, elev, lærer, forsker, systemutvikler, ingeniør, kunstner o.a. Nedenfor gir vi noen stikkord med anvendelser og undervisning som perspektiv.

1.1.1 Anvendelser

Her er noen anvendelsesområder av programmering:

- Informasjonsdeling
- Kommunikasjon
- Underholdning
- Datalagring og databehandling
- Tekstanalyse
- Spill
- Løsing av matematiske problemer
- Simulering, inkl. prediksjon
- Optimering
- Matematisk modellering
- Monitorering
- Automatisk styring
- Signalbehandling, f.eks. støyfiltrering

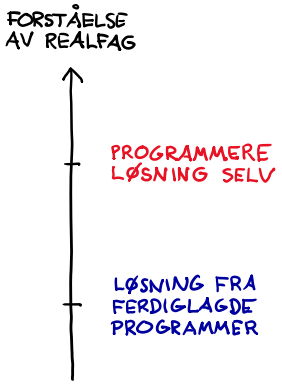
- Eksperimentering med praktisk utstyr
- m.m.

Vi kan også gjerne stille spørsmålet om slike og andre anvendelser av programmering er nyttige for enkeltmennesker og samfunnet. Hva mener du?

1.1.2 Undervisning

Vi (forfattere) er fagfolk innen teknologi og bruker programmering til å løse faglige problemer. Vi er også undervisere og slår gjerne et slag for programmering som verktøy for å fremme læring. Slik vi ser det, støtter programmering sentrale undervisningsprinsipper, nemlig motivering, aktivisering, konkretisering, individualisering og samarbeid[1]:

- ***Motivering***: Programmering motiverer for teori i matematikk og realfag fordi elever og studenter ser at programmering er nyttig for å anvende og konkretisere teori. Det er også motiverende å se at programmering er et svært viktig verktøy for å løse praktiske problemstillinger innen ulike fagfelt, som dataanalyse, matematisk modellering, simulering, automatisering m.m. med anvendelser innen teknologi og industri o.a.
- ***Aktivisering***: Gjennom programmering vil elever og studenter arbeide aktivt med teorien, og forståelse og læring av teorien utvikles. Programmering gir også økt mulighet til eksperimentering. En kan ha mye nytte av ferdiglagde programmer, men det er vår erfaring at det er mer lærerikt å programmere løsninger selv enn å bruke ferdiglagde programmer, som illustrert i figur 1.1.

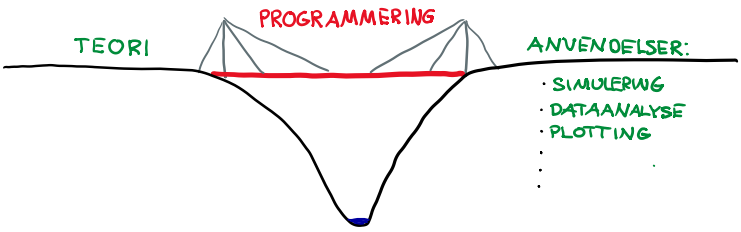


Figur 1.1: Gjennom programmering vil elever og studenter arbeide *aktivt* med teorien, og forståelse og læring utvikles.

- ***Konkretisering***: Programmering er en bro mellom teori og anvendelser, se figur 1.2. Med programmering kan teori og anvendelser knyttes sammen gjennom arbeid med

[1] = MAKIS

konkrete problemstillinger. Dette utvikler forståelse og læring av teorien, og en kan også se i hvilken grad teorien duger i praksis.



Figur 1.2: Programmering er en bro mellom teori og anvendelser.

- ***Individualisering***: Programmering gir gode muligheter til individuell tilpasning av undervisningen, siden en kan arbeide eller prøve seg fram i eget tempo.
- ***Samarbeid***: Programmering gir også gode muligheter for samarbeid ved at elever eller studenter diskuterer løsninger og/eller bidrar i felles programmeringsoppgaver.

1.1.3 Bruk av kunstig intelligens i læringsprosessen

Kunstig intelligens-verktøy (KI-verktøy) som Copilot og ChatGPT (språkmodeller) har de senere årene blitt tilgjengelige for oss alle. Disse verktøyene fungerer som digitale medhjelpere og kan gi støtte gjennom hele læringsprosessen – uansett tidspunkt på døgnet. Med tilgang til hjelp 24 timer i døgnet, 7 dager i uken trenger du ikke lenger vente på en lærer, medstudent eller forumrespons for å komme videre.

Dette endrer dynamikken i læring: du kan nå bestemme om du skal stå fast på en oppgave, eller ikke. Ved å stille spørsmål, be om forklaringer, få forslag til kode eller få hjelp til å forstå feilmeldinger kan du komme videre på egen hånd. Kort fortalt – KI-verktøy kan være en enorm ressurs i læringsprosessen.

Vi tror at KI-verktøy kan løse alle oppgavene i denne boken på en god måte. Men målet for læring av programmering er ikke bare er å finne fram til kode som fungerer – målet er i stedet å kunne utvikle kode selv, hvilket krever forståelse av prosessen som leder fram til en god løsning. Skal du lære programmering, må du derfor øve deg i å forstå selve problemstillingen i en oppgave, ha kunnskaper om kodetekniske verktøy (variabler, datatyper, konsepter osv.), teste løsninger, feile og justere koden. Når du bruker KI-verktøy, bør du derfor ikke bare kopiere løsningen, men heller bruke verktøyene som en samtalepartner: spør hvorfor koden fungerer, og hva alternativene er.

Et godt tips er å bruke disse verktøyene til å få hint og veiledning, heller enn ferdige løsninger. For eksempel kan du be om en forklaring på en algoritme, eller få hjelp til å forstå hvordan en bestemt datastruktur fungerer. Du kan også bruke dem til å få forslag til hvordan du kan forbedre din egen kode, eller til å forstå hvorfor noe ikke fungerer som forventet.

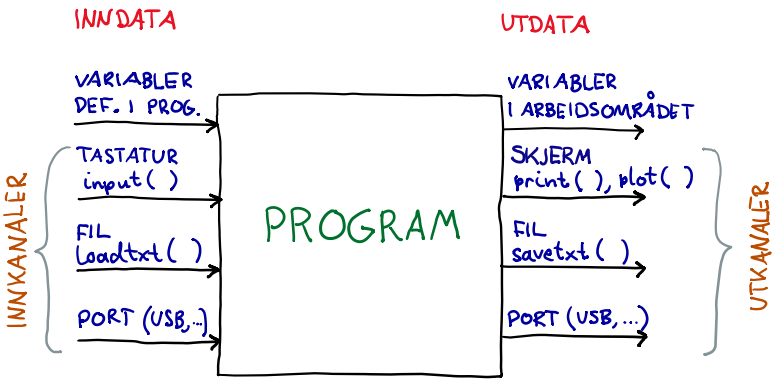
KI-verktøy brukt på denne måten blir ikke bare verktøy for å produsere kode – men også verktøy for å utvikle forståelse. Og det er nettopp denne forståelsen – sammen med tekniske ferdigheter i koding – som er kjernen i solid programmeringskompetanse.

1.2 Program med inndata og utdata

Vi beskriver her programmering og data med et overordnet perspektiv.

Et (Python-)program er en abstrakt «ting». Du kan ikke ta på et program, og derfor kalles gjerne programvare for bløtvare, eller software på godt norsk. Når datamaskinens mikroprosessor utfører instruksjonene som er uttrykt i programkoden, *kjører* (engelsk: run) programmet. Det er når det kjører at det gjør noe, og det kan da betraktes som en «prosess». Dette kapittelet skal utvikle en forståelse av de fundamentale aspektene ved denne prosessen, altså det kjørende programmet.

Et Python-program som kjører, produserer utdata (engelsk: output) fra inndata (engelsk: input). Programmet har et arbeidsområde (pythonsk: workspace) som fungerer som et slags arbeidsbord for programmet. Figur 1.3 illustrerer programmet med inndata og utdata. Typiske Python-funksjoner som brukes for å håndtere dataene er angitt, f.eks. loadtxt-funksjonen for å lese data fra fil. Du møter disse og mange andre funksjoner utover i boken.



Figur 1.3: Et program med inndata og utdata.

Her følger en beskrivelse av figur 1.3:

- **Arbeidsområdet** (pythonsk: workspace) inneholder alle variablene med sine verdier. Arbeidsområdets innhold beholdes etter at programmet er stoppet, og eksisterer fremdeles når det samme eller et annet program startes. Arbeidsområdet slettes automatisk når du går ut av programmeringsverktøyet (f.eks. Spyder).

- **Inndata** (engelsk: input) er typisk tall- eller tekstdata som programmet bruker i sin databehandling. Inndatene eksisterer i arbeidsområdet. Inndatene kan ha flere kilder:

  - *Variabler* med verdier definert i selve programmet, f.eks. array-variablene mnd og temp i programmet vist i figur 5.2. Dette er nok den vanligste inndatakilden til programmer.

  - *Tastaturet* der vi har skrevet tekst eller tall. Dataene kan leses inn til arbeidsområdet med Pythons input-funksjon, som er nærmere beskrevet i kap. 3.6.

  - *Fil* med tekst- eller talldata. Dataene kan leses inn fra filen til arbeidsområdet med den generelle open-funksjonen. Hvis dataene er talldata i form av «tekst» som vi kan lese med det blotte øye, er loadtxt-funksjonen i numpy-pakken mer hensiktsmessig enn open-funksjonen. Bruk av loadtxt-funksjonen beskrives i kap. 11.3.

  - *Port*, dvs. en kommunikasjonsport på datamaskinen. En typisk anvendelse er kontinuerlig innlesing av måledata fra måleelementer (sensorer) som genererer spenningsverdier som, etter at de er omdannet til digitale verdier, leses inn via en USB-port.

- **Utdata** (engelsk: output) er typisk tall- eller tekstdata som programmet genererer når det kjører. Utdataene eksisterer i arbeidsområdet. Utdataene kan ha flere mottakere:

  - *Arbeidsområdet selv.* Variabler som får sine verdier som resultat av programkjøringen, er tilgjengelige for bruk i programkoden som kjører.

  - *Konsollen* e.l. i det aktuelle programmeringsverktøyet, f.eks. konsollen i Spyder, se nede til høyre i figur 5.2. print-funksjonen, som vi møter gjennom hele boken, brukes til å skrive tall- eller tekstverdier til konsollen. plot-funksjonen i numpy-pakken kan brukes til å plotte data i figurer i konsollen (ev. i eksterne figurer).

  - *Fil.* Data i arbeidsområdet kan skrives til fil vha. den generelle open-funksjonen. Hvis dataene er talldata i form av «tekst», er savetxt-funksjonen i numpy-pakken mer hensiktsmessig enn open-funksjonen. Bruk av savetxt-funksjonen beskrives i kap. 11.4.

  - *Port.* En typisk anvendelse er kontinuerlig skriving av styresignaler som vårt Python-program har beregnet, til såkalte pådragsorganer som motorer, pumper, ventiler, varmeelementer, lamper, brytere osv. via en USB-port.

1.3 Om Python

1.3.1 Python – kort fortalt

Python er et programmeringsspråk utviklet av Guido van Rossum. Python ble lansert i 1991 og utvikles kontinuerlig. Gjennom denne boken vil du lære grunnleggende Python-programmering.

1.3.2 Hvorfor satse på Python?

Det fins mange alternative programmeringsspråk. Hvorfor satse på Python? Noen grunner:

- Python er gratis.
- Python kan kjøres på ulike plattformer: Windows, Mac og Linux.
- Python er et kraftig språk for analyse av tekstdata og talldata og er på nivå med MATLAB som numerisk verktøy.
- Python får (egentlig: tvinger) programmereren til å lage programkode med god visuell struktur.
- Python er populært og utbredt.
- Svært mye materiell er fritt tilgjengelig på nettet.

1.3.3 Når er Python ikke aktuelt?

Python er et tekstbasert programmeringsspråk. Python er ikke aktuelt hvis du vil programmere grafisk. Med grafisk programmering mener vi her at funksjonsblokker sammenkoples via signallinjer i et digitalt «tegneark», dvs. at en blokks utgang koples til en annen blokks inngang rent grafisk. LabVIEW og SIMULINK er eksempler på grafiske programmeringsspråk. I disse språkene kan du imidlertid inkludere tekstbasert kode. For eksempel kan du inkludere C-, Python- eller MATLAB-kode i LabVIEW.

1.3.4 Hvem temmer Python?

Hvem sørger for at Python er tam og til å stole på? Det gjør The Python Software Foundation (PSF). PSF holder i trådene når det gjelder utvikling og utgivelse av Python. PSF har hjemmeside på http://python.org.

Figur 1.4 viser PSFs hjemmeside.

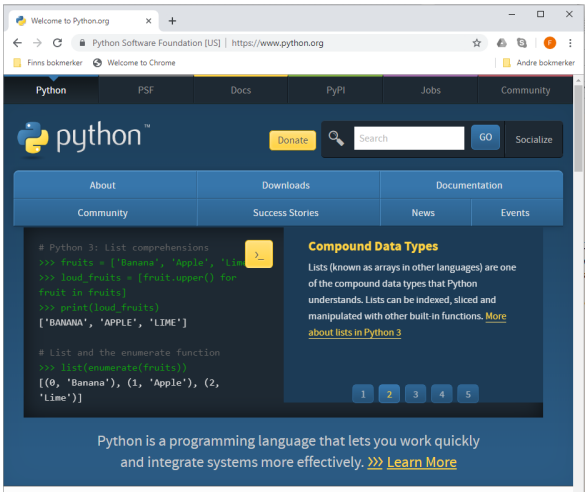
Fra PSFs hjemmeside: «The Python Software Foundation is the charitable organization behind the Python programming language.» På PSFs hjemmeside fins installeringsfiler for Python, dokumentasjon, veiledninger (tutorials), oversikt over Python-pakker m.m.

Du kan installere Python fra en installeringsfil på PSFs hjemmeside, men de fleste velger i stedet å installere Python gjennom en såkalt Python-distribusjon, som er en samling av selve Python-kompilatoren («Python-maskinen» som kjører Python-kode) og mange slags verktøy til Python-programmering. Den mest populære distribusjonen er Anaconda, som blir nærmere beskrevet i kap. 2.2.

1.4 Skrivemåter i boken

Her er informasjon om skrivemåter i boken:

- **Desimalkomma og desimalpunktum:** I norsk tekst skriver vi desimalkomma, f.eks. 3,14. I Pythonkode MÅ vi bruke desimalpunktum, f.eks. 3.14. I boken bruker vi



Figur 1.4: Hjemmesiden for Python Software Foundation (PSF) på http://python.org.

som hovedregel desimalkomma i teksten, men desimalpunktum der det er nødvendig for å unngå misforståelser.

- **Norsk/engelsk i programeksempler:** Iht. de såkalte PEP 8-reglene-reglene for god kode-stil skal kommentarer som hovedregel skrives på engelsk. (PEP er forkortelse for Python Enhancement Proposal. PEP8: Style Guide for Python Code, se https://www.python.org/dev/peps/pep-0008/.) Ja, det er klart at vi må skrive engelsk dersom vi antar at programmene skal kunne leses og brukes av folk som ikke behersker norsk. Men hvis vi er sikre på at kun folk som forstår norsk skal lese og bruke programmene, mener vi at norsk er ok. Vi har ikke vært konsekvente på norsk eller engelsk i programeksemplene i boken. Du vil altså finne at noen av programmene er «engelske», mens andre er «norske».

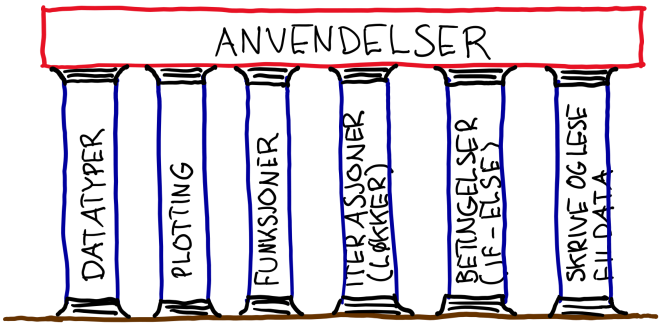
- **Norske tegn:** Vi har valgt å unngå særnorske tegn i både programkode og kommentarer innbakt i programkoden. Noen programmeringsmiljøer for Python og dokumentverktøy der programkoden skal vises, kan ikke vise disse tegnene korrekt. Det gjelder f.eks. nettlesere. Vi har derfor valgt erstatte de særnorske tegnene slik:

  - ae for æ
  - oe for ø
  - aa for å

- «**Skriv ut**»: Vi mener da at du skal bruke Pythons print-funksjon til å presentere verdien på skjermen, og ikke at du skal kople PC-en til en skriver og skrive ut verdien på papir. «Skriv ut» betyr da altså «presenter».

1.5 Om bokens struktur

Figur 1.5 illustrerer bokens struktur. Boken skal gi fundamentet for at du skal kunne programmere sammensatte applikasjoner av Python-programmering, som kan være numeriske applikasjoner som søking etter tegnsekvenser i tekstdata, programmering av simulatorer for fysiske systemer, løsing av matematiske optimeringsproblemer (≪hvordan finne beste løsning≫), løsing av likninger, tilpassing av matematiske modeller til data, statistisk analyse, beregning av sannsynligheter m.m. Vi gir noen eksempler på hvordan du kan utvikle slike sammensatte applikasjoner.



Figur 1.5: Illustrasjon av bokens struktur: Boken gir fundamentene som trengs for å kunne programmere sammensatte applikasjoner.

1.6 Lære mer?

Nedenfor er noen tips om hvor du kan lære mer om Python-programmering enn fra denne boken. Du kan sikkert finne mange andre ressurser med et nettsøk. Du kan også lære mye fra å stille gode spørsmål til kunstig intelligens-verktøy.

1.6.1 Nettsteder

- https://www.python.org/
- https://www.datacamp.com/
- https://www.w3schools.com/python/
- https://realpython.com/

1.6.2 Nettbaserte kurs

- http://coursera.com
- http://udemy.com

1.6.3 Bøker

Ta en titt på referanselisten bak i boken.

1.7 Oppgaver til kapittel 1

Oppgave 1.1 *Innkanal og utkanal*

Anta at du har laget et program som leser temperaturmålinger som er lagret som tall i en fil, og plotter målingene i et diagram som vises på PC-skjermen. Hva er innkanalen, og hva er utkanalen?

1.8 Løsninger til kapittel 1

Løsning til oppgave 1.1

Innkanalen: Datafilen. Utkanalen: Skjermen som viser diagrammet.