

Henrik Hillestad Løvold



Objektorientert programmering med PYTHON



FAGBOKFORLAGET

Objektorientert programmering med Python

Objektorientert programmering med Python

Henrik Hillestad Løvold



FAGBOKFORLAGET

Copyright © 2024 by
Vigmostad & Bjørke AS
All Rights Reserved
1. utgave 2024 / 1. versjon 2024
ISBN: 978-82-450-5761-4

E-utgaven er basert på 1. trykte utgave, 1. opplag:
ISBN: 978-82-450-5156-8

Elektronisk tilrettelegging: John Grieg, Bergen
Forsidedesign: Endre Barstad

Forfatteren har mottatt støtte fra Det faglitterære fond.

Boken er utgitt med støtte fra Kunnskapsdepartementet
ved Lærebokordningen for høyere utdanning.

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 eksemplarframstilling
bare tillatt når det er hjemlet i lov eller avtale med Kopinor.

Vigmostad & Bjørke AS er Miljøfyrtårn-sertifisert.



Jeg begynte så smått å programmere da jeg gikk på ungdomsskolen, den gang i Notepad på Windows 98. Min venn Hans Petter og jeg skrev nettsider i HTML 3.0, og selv om min nettside *kalkunweb.com* ble relativt ålreit, forstod jeg aldri hvordan han fikk til table-tag'en.

Mitt første møte med programmering i en faglig kontekst var, utrolig nok, i Flash/ActionScript 3.0. Hele andreåret mitt på videregående skrev vi merksnodige skript som utrolig nok fikk figurer til å bevege seg på skjermen. Det var også dette året jeg forstod at det het “while-løkker” og ikke “whælløkker”, som min lærer ofte sa.

I 2012 startet jeg mitt informatikk-studium ved Universitetet i Oslo. Studiet bød på mange faglige utfordringer, og etter både oppturer og nedturer (jeg ser på deg, *Metoder i grafisk databehandling og diskret geometri*) gikk jeg ut med en mastergrad i 2017.

Det som fenget meg mest under studiet var ikke bare studiet selv, men også mitt arbeid som gruppelærer i kursene om objektorientert programmering. Å undervise sammen med dyktige medstudenter slik som Astrid Helene Holter og Malin Aandahl ga meg motivasjon og inspirasjon.

Med litt flaks, mye hjelp fra universitetslektor Siri Moe Jensen og Arbeidsmiljøloven § 14-9 endte jeg til slutt opp med en fast stilling ved UiO. I 2021 ble jeg tilbudt en stilling ved UiT i Tromsø som jeg valgte å tiltre, mens jeg beholdt kontakten med UiO gjennom en bistilling.

I dag er jeg takknemlig for å få lov til å undervise informatikk og programmering til både bachelor- og masterstudenter. Å se nye studenter komme inn i auditoriet for første gang, og fem år senere se de samme studentene forlate universitetet med en mastergrad som jeg har bidratt til, gjør meg både glad og litt vemodig.

Å skrive en bok er et stort arbeid, og en spesiell takk må rettes til min kjære Frida E. Aasen for at jeg har fått tid til arbeidet. Uten hennes hjelp og støtte ville ikke dette prosjektet blitt noe av.

En stor takk må også rettes til Norsk faglitterær forfatter- og oversetterforening, som har gitt støtte til å skrive denne boken.

Jeg håper du liker denne boken, at den kan vekke nysgjerrighet og gi motivasjon til å lære.

Om denne boken

I denne boken har jeg tatt for meg de grunnleggende konseptene i objektorientert programmering, og bruker Python som programmeringsspråk for å vise hvordan du kan bruke objektorientert programmering i praksis. Du trenger ikke ha programmeringserfaring fra før, for boken starter med en innføring i grunnleggende programmering. Hvis du kan dette fra før, og er mer interessert i å fordype deg i den objektorienterte delen, så er det bare å hoppe over den første delen.

Objektorientert programmering er et programmeringsparadigme som ble utviklet på 60-tallet, og som har sitt utspring fra Universitetet i Oslo. Professorene Ole-Johan Dahl og Kristen Nygaard ønsket å ta programmeringen nærmere de konseptene vi kjenner fra den virkelige verden. De introduserte begrepene klasser og objekter for å kunne modellere programmer på en naturlig og oversiktlig måte. I dag er objektorientering et av de mest brukte paradigmene i programmering som benyttes både i forskning, undervisning og næringsliv.

Python er et språk som er mye benyttet i undervisning, ettersom uttryksformen i språket er relativt nært naturlige språk som vi benytter i det daglige. Språket ble først utviklet på slutten av 80-tallet, og er nå i versjon 3. Python er bygget opp som et utelukkende objektorientert språk, og egner seg derfor godt for dem som ønsker et lettfattelig språk som dekker de objektorienterte prinsippene godt. Utenom undervisning benyttes det mye i forskning på grunn av sin gode støtte for matematiske og statistiske applikasjoner, men også i næringslivet i alt fra web-programmering til komplekse systemer.

Denne boken dekker pensum i de grunnleggende emnene i objektorientert programmering som typisk gis i løpet av det første året ved universiteter og høyskoler. Den er spesielt tilpasset undervisningen slik den gis i Norge, og har et lettfattelig språk og korte kapitler. Hvert kapittel oppsummeres i eksempeloppgaver som det kan være nyttig å gjøre for å bekrefte at man har forstått det man har lest.

Takk!

Henrik Hillestad Løvold, Tromsø, juni 2024

Til Ella

Innhold

I Introduksjon

1	Hva er programmering?	15
1.1	Programmering og algoritmer	15
1.2	Programmeringsspråk	16
1.2.1	Python	16
2	Objektorientert programmering	19
2.1	Hullkort og assembly-språk	19
2.2	Moderne språk	20
2.3	Simula 67, objektorientert programmering	20
3	Programmeringsverktøy	23
3.1	Python	23
3.1.1	Nedlasting og installasjon av Python 3	23
3.2	Editor	24
3.3	Å kjøre et Python-program	25
3.3.1	Åpne terminalen	25
3.3.2	Navigering i terminalen	25
3.3.3	Kjøring av Python-programmer i terminalen	27

II Grunnleggende programmering

4	Utsagn, variabler og datatyper	33
4.1	Utsagn	33
4.1.1	Rekkefølge	33
4.2	Variabler	34

4.2.1	Deklarasjon og tilordning	34
4.2.2	Endre verdi på variabel	35
4.3	Datatyper	35
4.3.1	Tekst	35
4.3.2	Heltall og desimaltall	36
4.3.3	Sannhetsverdi	37
4.4	Oppgaver	37
5	Beslutninger, output og input	39
5.1	Output og input	39
5.1.1	Typecasting	39
5.2	Beslutninger	40
5.2.1	Logiske operatører	41
5.2.2	if-setninger	41
5.2.3	elif og else	42
5.3	Oppgaver	43
6	Uttrykk og feilmeldinger	45
6.1	Evaluering av uttrykk	45
6.1.1	Tilordning	45
6.1.2	Kall til funksjoner	46
6.2	Feil og feilmeldinger	47
6.2.1	Syntaks- og kjøretidsfeil	47
6.2.2	Logiske feil	48
6.2.3	Å lese en feilmelding	49
6.3	Oppgaver	50
7	Prosedyrer, funksjoner og skop	53
7.1	Prosedyrer	53
7.1.1	Prosedyre uten parametere	53
7.1.2	Prosedyrer med parametere	54
7.2	Funksjoner	56
7.3	Variabler og skop	57
7.4	Oppgaver	59
8	Samlinger	61
8.1	Lister	61
8.1.1	Opprette lister, legge til og fjerne elementer	62
8.1.2	Indeksering	63
8.2	Ordbøker	64

8.2.1	Opprette ordbøker, legge til og fjerne elementer . . .	64
8.2.2	Indeksering	66
8.3	Nøstede datastrukturer	67
8.3.1	Nøstede lister	67
8.3.2	Ordbøker med lister som verdi	68
8.3.3	Andre nøstede datastrukturer	69
8.4	Oppgaver	70
9	Løkker	73
9.1	While-løkker	73
9.2	For-løkker	75
9.3	Løkker og samlinger	77
9.4	Oppgaver	80
10	Arbeid med filer	83
10.1	Lese fra fil	84
10.2	Skrive til fil	87
10.3	Oppgaver	88
11	Fange feil med try og catch	91
11.1	Try og catch	91
11.1.1	Spesifisere feiltype	92
11.2	Skriving med try, except og finally	93
11.3	Oppgaver	94
 III Objektorientert programmering 		
12	Objekter	101
12.1	Objekter tilbyr tjenester	101
12.2	Objekter er separate fra hverandre	102
12.3	Oppgaver	102
13	Klasser	105
13.1	Definere egne datatyper	105
13.2	Implementasjon og grensesnitt	107
13.2.1	Implementasjon	107
13.2.2	Grensesnitt	108
13.3	Begrense muligheter for feil	108
13.4	Interaksjon med objekter av egne klasser	109
13.5	Oppgaver	110

14 Referanser	113
14.1 Hva er en referanse?	113
14.2 Variabler refererer til samme objekt	115
14.3 Referanser til lister	117
14.4 Skop	118
14.4.1 Skop i klasser	120
14.4.2 To variabler med samme navn	120
14.5 Referanser og pekere	122
14.6 Oppgaver	122
15 Objektorientert design	125
15.1 Kohesjon og kobling	125
15.2 Aggregering	127
15.3 Spesialmetoder i Python	129
15.3.1 Utskrift av objekter	130
15.3.2 Sammenlikning av objekter	131
15.4 Iteratorer	132
15.5 Oppgaver	134
16 Modellering og dokumentasjon	137
16.1 Tekstlig dokumentasjon	137
16.1.1 Kommentarer	137
16.1.2 PyDoc	139
16.2 Diagrammer og visuell modellering	143
16.2.1 UML klassediagram	143
16.3 Oppgaver	147
17 Et større objektorientert system	151
17.1 Krav til systemet, og hvordan å lese dem	151
17.1.1 Kravspesifikasjon	152
17.1.2 Hvordan leser vi kravene?	153
17.2 Klassen Rute	153
17.2.1 Implementasjon av klassen	155
17.3 Klassen Stoppested	156
17.3.1 Implementasjon av klassen	157
17.4 Klassen Kollektivsystem	159
17.4.1 Ordreløkke	159
17.4.2 Metoder for valg	161
17.5 Noen avsluttende ord om systemet	164

IV Avanserte objektorienterte prinsipper

18 Arv	171
18.1 Superklasser og subklasser	171
18.1.1 Flere klasser arver fra samme superklasse	173
18.2 Polymorfi	175
18.3 Multippel arv	177
18.3.1 Diamantproblemet	177
18.3.2 En liten advarsel	179
18.4 Operator overloading	179
18.4.1 Likhhet mellom objekter	180
18.4.2 Større og mindre	181
18.4.3 Regneoperatorer	183
18.5 Modellering av arv	184
18.6 Oppgaver	185
19 Avanserte designmønstre	189
19.1 Arv og komposisjon	189
19.1.1 Arv	189
19.1.2 Komposisjon	191
19.1.3 Arv versus komposisjon	193
19.2 Dekoratorer	194
19.2.1 Funksjoner som returverdi	196
19.3 Strategimønster	197
19.3.1 Sorteringsstrategier	198
19.4 Oppgaver	201
20 Egne datastrukturer med klasser og objekter	203
20.1 Lenket liste	203
20.1.1 Oppbygging av en lenket liste	204
20.1.2 Fjerne objekter	205
20.1.3 Andre listeoperasjoner	207
20.2 Rekursiv iterasjon	207
20.2.1 Rekursjon med objekter	208
20.3 Andre datastrukturer	208
20.4 Oppgaver	209
21 Testing av objektorienterte programmer	211
21.1 Enhetstesting	211
21.1.1 Å skrive enhetstester	212

21.1.2 Assert	214
21.1.3 Modulen unittest	215
21.2 Andre tester	217
21.3 Oppgaver	217
22 Utviklingsmetoder	221
22.1 Top-down eller bottom-up	221
22.1.1 Top-down	222
22.1.2 Bottom-up	223
22.2 Metodikk	224
22.2.1 Fossefallsmetoden	224
22.2.2 Smidig utvikling	225
22.3 Parprogrammering	226
Register	229

Del I

Introduksjon

Kapittel 1

Hva er programmering?

Programmering er en metode for å løse et problem. Den som programmerer må definere fremgangsmåten så presist at selv en datamaskin kan forstå hvordan problemet skal løses.

I dette kapittelet skal vi ta for oss begrepene programmering, algoritmer, programmeringsspråk og Python.

1.1 Programmering og algoritmer

For å forstå hva programmering er, behøver du i grunn ikke ha kunnskap om datamaskiner i det hele tatt. Alle programmerer vi til stadighet, uten at vi engang er klar over det. Et program er nemlig bare et sett med veldefinerte instruksjoner, satt sammen til en fremgangsmåte.

Er du glad i å bake, følger du kanskje oppskrifter for å lage ulikt bakeverk. En slik oppskrift inneholder informasjon om hvilke ingredienser som skal brukes, og i hvilken rekkefølge steg skal utføres for å få best mulig resultat. Ingredienser og fremgangsmåter vil være ulikt for hver type av bakeverk, men det er viktig at oppskriften er skrevet slik at vi kan forstå den og følge den.

Nettopp viktigheten av det å være presis, blir tydelig når vi skal programmere en datamaskin til å utføre en oppgave. Om det mangler et komma i en bakeoppskrift, eller kanskje “gjær” er skrevet feil, er vi i stand til å forstå hva som er ment likevel. En datamaskin er dum, og ikke i stand til å gjøre antakelser i det hele tatt. Om vi skal definere en

fremgangsmåte for å få datamaskinen til å summere tall, og ved slump glemmer å avslutte en parentes, vil ikke datamaskinen gjette seg til hva vi *antakelig* mente å gjøre, men heller gi oss beskjed om at dette ikke er noe den kan forstå.

Fremgangsmåter er altså helt sentralt innen programmering. Å programmere innebærer å gå fra et utgangspunkt til et ønsket resultat, gjennom et bestemt sett av steg. En informatiker vil kalle en slik fremgangsmåte for en *algoritme*. Dette begrepet vil du få god kjennskap til i løpet av de kommende kapitler i denne boken.

1.2 Programmeringsspråk

Vi mennesker snakker med hverandre på menneskespråk. Enten det er norsk, engelsk eller tysk, bruker vi ord som bærer mening, og setter dem sammen til setninger som kan forstås. Slik er det også når vi skal kommunisere med en datamaskin. Akkurat som det finnes ulike menneskespråk, finnes det også ulike programmeringsspråk som kan benyttes for å fortelle datamaskinen hva den skal gjøre. Ulike språk er egnet for ulike formål, og i denne boken har vi valgt å bruke det språket som heter Python.

1.2.1 Python

Python er et programmeringsspråk som dukket opp i 1991, og som siden har blitt oppdatert etter hvert som den teknologiske utviklingen har gått videre. I dag er Python et av de språkene som brukes mest i begynneropplæringen i programmering. Hovedårsaken til dette, er at språket er ganske nært menneskespråk, og at vi kan skrive enkle, korte programmer som godt illustrerer akkurat det konseptet vi ønsker å illustrere, uten at man behøver ytterligere ord eller tegn som leder oppmerksomheten bort.

En annen fordel med Python som språk for opplæring i programmering, er at det er meget generelt. Vi kan benytte det til å løse problemer innen en rekke felt – alt fra tolkning av menneskespråk til programmering av nettsider, til avansert matematikk. I de senere årene er også Python blitt et svært relevant språk for utvikling av kunstig intelligens. Selv om vi i denne boken ikke går inn på avanserte programmeringskonsep-ter, bør du vite at Python brukes mye i arbeidslivet; det er altså ikke kun et begynner-språk. Med Python kan man programmere på ulike

måter, ofte kalt programmeringsparadigmer. Ett slikt paradigme kalles *objektorientert programmering*, og er det vi skal fokusere på i denne boken.