



*Andreas Drolsum Haraldsrud,
Henrik Andersen Sveinsson og Henrik Hillestad Løvold*

PROGRAMMERING I SKOLEN



UNIVERSITETSFORLAGET

Programmering i skolen

Programmering i skolen

Andreas Drolsum Haraldsrud
Henrik Andersen Sveinsson
Henrik Hillestad Løvold

© Universitetsforlaget, Oslo, 2020

ISBN: 9788215041230

1. utgave, 1. versjon, Universitetsforlaget, 2020

E-utgaven baserer seg på 1. trykte utgave, Universitetsforlaget 2020

Elektronisk tilrettelegging: Universitetsforlaget

Omslagsdesign: Endre Barstad

Sats/design: Andreas D. Haraldsrud, Henrik A. Sveinsson og Henrik H. Løvold

Kapittelfoto: 1. Monkey Business Images/Shutterstock.com. 2. sondem / shutterstock.com. 3. Damrongwit Preecha / Shutterstock.com. 4. Dmitriy Rybin / Shutterstock.com. 5. whiteMocca / Shutterstock.com. 6. Olga Medvedeva / Shutterstock.com. 7. PopTika / Shutterstock.com. 8. Photographee.eu / Shutterstock.com. 9. GaudiLab / Shutterstock.com. 10. Trismegist san / Shutterstock.com. 11. frankie's / Shutterstock.com. 12. KieferPix / Shutterstock.com. 13. Wanf LiQiang / Shutterstock.com. 14. Golubovy / Shutterstock.com. 15. Gina Buliga / Shutterstock.com. 16. Yurchanka Siarhei / Shutterstock.com.

Boka er utgitt med støtte fra Kunnskapsdepartementet ved Lærebokutvalget for høyere utdanning.

Forfatterne har mottatt støtte fra Det faglitterære fond.

Materialet i denne publikasjonen er omfattet av åndsverklovens bestemmelser. Uten særskilt avtale med Universitetsforlaget er enhver eksemplarframstilling og tilgjengeliggjøring bare tillatt i den utstrekning det er hjemlet i lov eller tillatt gjennom avtale med Kopinor, interesseorgan for rettighetshavere til åndsverk.

www.universitetsforlaget.no

post@universitetsforlaget.no

Innhold

I Grunnleggende programmering

1	Introduksjon	13
1.1	Valg av programmeringsspråk	13
1.2	Programmering styrker fagene	14
1.3	Programmering endrer fagene	14
1.4	Programmering sikrer fellesskapet	15
1.4.1	Digital dannelse	15
1.4.2	Kunstig intelligens i samfunnet	16
1.5	Hva er et dataprogram?	17
1.5.1	Å programmere hverdagslivet	17
1.6	Installasjon	18
1.7	Hva kan du forvente av denne boka?	19
1.7.1	Strukturen i boka	19
2	Tall og variabler	21
2.1	Kom i gang	21
2.2	Tall	22
2.2.1	Heltall	22
2.2.2	Desimaltall	23
2.2.3	Standardform	23
2.3	Variabler	24
2.4	Output	25
2.5	Tekststrenger	25
2.6	Boolske variabler	26
2.7	Aritmetikk	27
2.7.1	Regnerekkefølge	28
2.8	Kommentarer	28
2.9	Lister	29
2.10	Input fra kommandolinja	31

2.11	Moduler og pakker	32
2.12	Feilsøking	35
2.13	Oppgaver	37
3	Kontrollstrukturer og funksjoner	41
3.1	Vilkår – «if-tester»	42
3.1.1	Flytskjemaer	44
3.1.2	Intervaller	45
3.1.3	Sammensatt eksempel	45
3.2	Iterasjon med løkker	47
3.2.1	While-løkker	47
3.2.2	For-løkker	49
3.3	Funksjoner	52
3.3.1	Kalle på funksjoner	52
3.3.2	Definere en funksjon	53
3.3.3	Syntaks for funksjoner i Python	55
3.3.4	Flere returvariabler	58
3.3.5	Funksjon inni funksjon	58
3.3.6	Utført eksempel: Finn funksjonsuttrykket	60
3.3.7	Ikke-matematiske funksjoner	62
3.3.8	Funksjoner som ikke returnerer	63
3.3.9	Keyword-argumenter	64
3.3.10	Funksjoner og definisjonsområde*	65
3.4	Oppgaver	68
4	Datastrukturer	73
4.1	Array	73
4.1.1	Vektorisering	76
4.1.2	Matrise som array av arrayer*	77
4.2	Tuppel	79
4.3	Dictionary*	80
4.4	Oppgaver	85
5	Blokkprogrammering	87
5.1	Om Scratch	88
5.2	Blokkprogrammering med Scratch	88
5.3	Start-blokken	89
5.4	Variabler	90
5.5	Vilkår og if-setninger	91
5.6	Løkker	92
5.7	Funksjoner	94
5.8	Bevegelse	95

5.9	Geometri i Scratch	96
5.10	Matematisk kunst i Scratch	97
5.10.1	Fyrverkeri med geometriske former	99
5.11	Overgang fra blokk til tekst	102
5.11.1	Syntaks	102
5.11.2	Faglig tilnærming til overgang til tekstprogrammering . . .	103
5.11.3	Gruppearbeid er nøkkelen	103
5.12	Oppgaver	104
6	Plotting og grafikk	105
6.1	Plotting av data	105
6.1.1	Subplott	109
6.1.2	Andre grafiske representasjoner	111
6.2	Plotting av funksjoner	114
6.2.1	3D-plott*	115
6.3	Grafikk	117
6.3.1	Skilpadder som objekter	119
6.3.2	Eksempel: Virrevandring (brownske bevegelser)	121
6.4	Oppgaver	124
7	Data og statistikk	127
7.1	Lese inn data	127
7.1.1	Lese inn tabellbaserte filer	128
7.1.2	Linje for linje	131
7.1.3	Linje for linje fra internett	134
7.1.4	«Hva må jeg huske om å lese filer?»	136
7.2	Lineær regresjon	137
7.3	Gjennomsnitt, median, typetall og standardavvik	139
7.4	Logiske operasjoner på arrayer	142
7.5	Prosjekt: Solflekker	142
7.6	Tilfeldige tall	146
7.7	Statistiske simuleringer	147
7.7.1	Nedarving og sannsynlighet	147
7.7.2	Stigespill	148
7.7.3	Har noen bursdag på samme dag?	150
7.7.4	Virrevandring*	151
7.8	Samle inn data	155
7.9	Oppgaver	156

II Programmering i klasserommet

8	Programmering i klasserommet	161
8.1	Muligheter og utfordringer	161
8.1.1	Hvordan motiverer jeg de svake og de sterke elevene?	163
8.1.2	Hva hvis jeg ikke klarer å holde tritt med elevene?	164
8.1.3	Hva gjør jeg når det tekniske ikke virker?	164
8.1.4	Alle trenger hjelp! Hvordan strekker jeg til?	165
8.2	Vurderinger	166
8.2.1	Kortsvar og flervalg	166
8.2.2	Feil og omskrivninger	167
8.2.3	Skisser og forklaringer	168
9	Metoder	169
9.1	Å snakke om programkode	169
9.1.1	Programmeringspuslespill	170
9.2	Programmeringsmetoder i klasserommet	170
9.2.1	Parprogrammering	171
9.2.2	Programmeringsdiktat	172
9.2.3	Programmeringsalias	172
9.2.4	Smidig utvikling	173
9.3	Jupyter Notebook	175
9.4	Versjonskontroll*	176
9.4.1	Versjonskontroll med Git	177
9.4.2	Samarbeid og deling	179

III Numeriske metoder og modellering

10	Algoritmer og modellering	183
10.1	Algoritmer	183
10.1.1	Beregning av kvadratrøtter	184
10.1.2	Beregning av π : Arkimedes' algoritme	185
10.1.3	Beregning av π : Tilfeldige tall	187
10.2	Algoritmisk tenking	189
10.3	Modellering	190
10.4	Numerisk matematikk	191
11	Ligninger	193
11.1	Utforskende ligningsløsning	195
11.2	Halveringsmetoden	196
11.3	Newtons metode	199
11.4	Renter, sparing og lån	201

11.4.1	Annuitetslån	202
11.4.2	Variierende rente	204
11.4.3	Prosjekt: Simulere personlig økonomi	204
11.5	Oppgaver	209
12	Derivasjon og integrasjon	211
12.1	Derivasjon av funksjoner	211
12.2	Derivasjon av diskrete data	215
12.2.1	Bilder	217
12.3	Bestemt integrasjon av funksjoner	219
12.3.1	Rektangelmetoden	219
12.3.2	Trapesmetoden	221
12.4	Integrasjon av diskrete data	224
12.5	Numeriske biblioteker	226
12.6	Oppgaver	228
13	Dynamiske systemer	229
13.1	Krefter og bevegelse	230
13.1.1	Rettlinjet bevegelse (Fysikk 1)	230
13.1.2	Didaktiske perspektiver	234
13.1.3	Eulers metode	236
13.1.4	Bevegelse i flere dimensjoner (Fysikk 2)*	237
13.2	Populasjonsdynamikk (Biologi)	239
13.2.1	Kontinuerlige modeller	241
13.2.2	Modeller med ressursbegrensninger	242
13.2.3	Modeller med to arter	243
13.3	Reaksjonsfart (Kjemi 2)	246
13.4	Oppgaver	251
 IV Teknologi og avansert programmering		
14	Mikrokontrollere	255
14.1	Programmering av micro:bit	256
14.1.1	Blokk eller tekst?	256
14.2	Blokkprogrammering med micro:bit	257
14.3	Tekstprogrammering med micro:bit	258
14.4	Vilkår og løkker	259
14.5	Knapper	262
14.6	Radio	262
14.7	Analoge og digitale porter	264
14.8	Mikrokontrollere i klasserommet	266
14.8.1	Signaloverføring – temperatur i en solfanger	266

14.8.2	Kalibrering – fuktighetssensor	269
14.8.3	Lyd og toneskalaer	272
14.9	Oppgaver	277
15	Objektorientert programmering*	279
15.1	Klasser	280
15.1.1	Eksempel: En beholder for primtall	282
15.2	Arv	284
15.3	Når er klasser nyttig?	286
15.4	Prosjekt: Flokkdynamikk	287
15.4.1	Fra enkelt dyr til flokk	288
15.4.2	Interaksjoner	289
15.4.3	Periodiske randbetingelser	292
15.4.4	Subklasser for spesifikke dyr	292
15.5	Oppgaver	295
16	Maskinlæring og kunstig intelligens	297
16.1	Kunstige nevrale nettverk	298
16.2	Lineær regresjon med maskinlæring	299
16.3	Mer avanserte modeller	303
16.4	En maskinlæringsmodell for blomster	307
16.5	Overtilpasning*	313

Forord

Takk for at du leser boka vår. Med fagfornyelsen 2020 har programmering og algoritmisk tenking blitt en integrert del av flere fag. Vi ønsket derfor å lage en bok som skulle hjelpe lærere til å gjøre gode valg i programmeringsundervisningen. Dette er altså resultatet.

Boka er ganske omfattende og dekker mye. Det er altså ikke en spesialisert bok i ett bestemt fag eller på ett bestemt trinn. Vi prøver heller å gi deg de generelle verktøyene du trenger for å bruke programmering på en god måte i realfagene. Noe av stoffet vil du ha direkte nytte av, andre deler er mest for din egen del. Det er også litt opp til deg hva du ønsker å sette deg inn i og hva du trenger. Dette avhenger av det nivået du selv er på (som lærer), elevenes nivå, øvrige læremidler, klassetrinn og fag. Derfor må du ikke føle at du trenger å lære alt. Se på denne boka som en konfekteske med mange lekre biter. Noen spiser alt, mens andre lar bananlikøren være.

Programmering er både spennende, kreativt, frustrerende og morsomt. Det prøver vi å trekke frem med utforskende eksempler, faglig relevante oppgaver og undervisningsopplegg som du kan bruke direkte i klasserommet. Noen oppgaver er kanskje mest egnet for deg selv, andre kan du gi direkte til elevene dine. Vi håper at denne boka kan hjelpe deg med å gjøre disse valgene. Vi prøver også å adressere ulike utfordringer og muligheter i didaktiske refleksjoner underveis i boka. Dette er erfaringer som vi har høstet fra undervisning av både elever og lærere. Gjennom undervisning i videregående skole og på Universitetet i Oslo i etterutdanningsprosjektet ProFag har vi vært nær på undervisningen og de lærende. Dessuten er flere av perspektivene og metodene basert på forskning. Takk til Valler videregående skole og Universitetet i Oslo, spesielt CCSE (Center for Computing in Science Education), for mulighetene de har gitt oss.

Du kan få mye ut av å samarbeide med andre når du jobber med denne boka. I programmering får både du og elevene dine mye nytte av samarbeid. Skulle du likevel stå fast, finner du løsningsforslag til oppgavene i boka på bokaas hjemmesider: <https://nettressurser.no/programmeringiskolen>. Her finner du også annet relevant fagstoff.

Utforsk, eksperimenter og diskuter. La deg bli frustrert, fascinert og ha det gøy. Lek deg med fagene dine og utforsk nye perspektiver. Det viktigste er å være nysgjerrig og tørre og prøve. God reise!

Oslo, mars 2020

Henrik, Henrik og Andreas

Del I

Grunnleggende programmering

A photograph showing a female teacher with brown hair tied back, wearing a black and white polka-dot shirt, leaning over a table and pointing at a small robot. Three students are gathered around the table: a boy in a red shirt looking at the robot, a girl with blonde hair looking at a laptop, and another student partially visible on the left. The table is covered with a blue cloth and has several small robots with yellow wheels and various sensors. A laptop is open on the right side of the table. The background is a bright, modern classroom with large windows.

1. Introduksjon

Programmering og algoritmisk tenking er viktige ferdigheter i dagens samfunn. Derfor har det nå også blitt en integrert del av flere fag i skolen. I denne boka skal vi se på hvordan vi lærer og lærer bort programmering i en realfaglig sammenheng. Vi bruker eksempler fra matematikk, naturfag, fysikk, kjemi og biologi for å illustrere at vi kan lære programmering gjennom å arbeide med fag, men at vi også kan lære fag gjennom å programmere.

Boka handler om programmering i en realfaglig sammenheng og er spesielt rettet mot lærere i ungdomsskolen og videregående skole. Lærere i barneskolen som har lyst til å gå litt dypere inn i programmering, kan også ha godt utbytte av boka.

1.1 Valg av programmeringsspråk

Boka tar utgangspunkt i tekstbasert programmering fordi dette gir det beste grunnlaget for å drive med og lære bort programmering. Blokkbasert programmering har senere kommet som et pedagogisk virkemiddel som spesielt kan lette innlæringen av programmering for barn og ungdom, og derfor adresserer vi også dette i et eget kapittel.

Vi har valgt programmeringsspråket *Python* i denne boka. Dette er et valg vi har tatt av både pedagogiske, faglige og didaktiske grunner:

1. Python er et av de enkleste og mest lettleste tekstbaserte språkene.
2. Den rene syntaksen i Python gjør det enklere å oversette matematikk til Python-kode.
3. Python blir brukt over hele verden, både i akademia og i næringslivet.
4. Python har veldig mange gode biblioteker med ferdigskrevet kode som kan brukes til praktiske formål, spesielt i realfaglige sammenhenger.

For mange elever føles det matnyttig og motiverende at Python ikke er et *skolespråk* som de ikke får bruk for utenfor skolen. Det er et verktøy som de kan få direkte bruk for senere i livet, i tillegg til at grunnstrukturene i Python er lett overførbare til de fleste andre populære programmeringsspråk.

I tillegg til det tekstbaserte språket Python gjennomgår vi hovedtrekk i blokkbaserte språk som Scratch og MakeCode. Dette er to populære og pedagogiske språk som både er morsomme og lærerike å jobbe med.

1.2 Programmering styrker fagene

Et sentralt element i fagfornyelsen i skolen i 2020 er *dybdelæring*. Konseptet er ikke nytt, men i de nye læreplanene skal det legges bedre til rette for varig forståelse og dybdekunnskap ved at det blant annet introduseres færre ulike temaer i de ulike fagene. Programmering kan være et ledd i dybdelæringen ved at vi kan utforske kjente konsepter på flere måter, i tillegg til at det lar oss utforske temaer vi før ikke har kunnet utforske. Dessuten gir programmering oss metoder og teknikker som kan brukes på tvers av fag, noe som kan bidra til en dypere forståelse av problemløsningsstrategier og bestemte arbeidsmåter.

1.3 Programmering endrer fagene

Realfagene er utviklet gjennom flere hundre år og har lange tradisjoner. Mange av temaene som dekkes i læreplanene i de naturvitenskapelige fagene, har blitt valgt med utgangspunkt i hvor mye matematikk elevene behersker på det gitte årstrinnet.

Med programmering er det enklere å arbeide ut fra et *modelleringsperspektiv* der elevene kan utforske ulike sammenhenger og modeller uten å ta for mye hensyn til matematiske begrensninger. Dette gjør at vi kan flytte fokuset til drøfting, analyse og en dypere forståelse av de ulike modellene. Ved å bruke programmering gjør vi derfor tilgjengelig en rekke nye faglige problemstillinger fordi vi ikke er begrenset av hvilke ligninger som lar seg løse med penn og papir.

I matematikk kan vi også ha stor nytte av programmering. Mange analytiske metoder er utviklet for å løse *ett* spesielt problem. Med programmering kan vi utforske nye fenomener og generalisere løsningsstrategier i større grad enn tidligere. Disse numeriske metodene skal vi se på i del tre av boka.

Store deler av denne boka handler om hvordan vi kan bruke programmering i de eksisterende fagene. Men siden programmering gir nye muligheter, ønsker vi å utforske disse mulighetene. Derfor vil enkelte deler av denne boka inneholde temaer og metoder som ikke er direkte inne i læreplanene. Disse temaene vil derimot kunne berike perspektivene til både lærere og elever.

1.4 Programmering sikrer fellesskapet

Gjennom 2010-tallet har det digitale blitt vevd inn i alle deler av samfunnet. Da smarttelefonen kom, ble vi koblet kontinuerlig til internett, og derfor har 2010-tallet av enkelte blitt kalt «tiåret da dødtiden forsvant». Men vet vi hvordan denne digitale veven fungerer? Vi stoler på at finanssystemet fungerer, fordi vi er relativt fortrolige med den grunnleggende operasjonen, nemlig transaksjon av penger mot en vare, tjeneste eller eiendel. I den digitale verden har vi mange andre typer transaksjoner. Men den grunnleggende enheten er en linje med kode. Når vi åpner en nettside, kjøres det mange linjer med kode, og datamaskinen vår utveksler informasjon med en server som er koblet til internett. Et kjennetegn ved tjenester på internett, i motsetning til tjenester i «den virkelige verden», er at de er tilsynelatende gratis. Vi betaler riktignok for internetttilkoblingen vår, men mange tjenester vi bruker på nettet, betaler vi ikke for med penger. Men det koster jo penger å drifte disse tjenestene, så et eller annet sted må det komme inntekter. Dersom vi forstår programmering, kan vi bedre forstå hvordan transaksjoner av informasjon kan føre til inntekter for selskaper som driver tjenester på internett.

1.4.1 Digital dannelse

Barn blir ikke automatisk digitalt kompetente av å få en iPad, på samme måte som voksne ikke blir mekanikere av å kjøre en bil. Det er *hva* vi gjør med det digitale mediet, som har noe å si. Når vi snakker om digital dannelse, mener vi det å forstå hva som ligger bak det digitale samfunnet, og forholde seg reflektert til dette samfunnet. Kanskje stikker vi nesa litt sjeldnere ned i mobilen når vi vet at ønsker om å gjøre nettopp det ofte skyldes bevisst manipulasjon fra multinasjonale selskaper.

Programmering er ikke bare for spesielt interesserte. Det er for alle. Det betyr ikke at alle skal bli programmerere, men vi mener at alle trenger en viss forståelse av de grunnleggende byggesteinene i vårt digitale samfunn.

Når samfunnet gjennomdigitaliseres, må vi også vurdere om de digitale verktøyene spiller på lag med verdiene i samfunnet vårt. I Norge er det i stor grad konsensus om fellesskolen, likestilling, at alle skal ha gode muligheter i livet sitt, og at alle skal bli tatt vare på. Digitaliseringen utfordrer slike verdier fordi teknologien i lang tid har blitt utviklet av en smal gruppe dataingeniører. Da Apple lanserte Health-kit i 2014, hadde de glemt at kvinner har en

menstruasjonssyklus. Da Microsoft prøvde å slippe løs samtaleroboten *Tay* på Twitter i 2016, måtte de ta den ned etter under et døgn fordi den raskt ble til en rasistisk kvinnehater. Det var ingen som eksplisitt hadde programmert inn denne oppførselen, det var bare sånn den ble.

Vi har blitt vant til at et uregulert økonomisk system fører til stor økonomisk ulikhet, selv om ingen politikere har laget lover som eksplisitt innfører denne ulikheten. Vi har derfor også en levende debatt om hvor sterke virkemidler vi skal bruke for å motvirke den ulikheten som kapitalismen selv produserer. Vi kommer i de neste tiårene til å bli stilt overfor tilsvarende problemstillinger, der digitalisering fører til systemer som «bare blir sånn» dersom vi ikke passer på. Vi trenger at hele samfunnet settes i stand til å delta i debatten om slike problemstillinger.

Ved å sørge for at alle har tilgang på det grunnleggende innen all digital teknologi, nemlig en linje med kode, bidrar vi til å opprettholde det store fellesskapet som samfunnet vårt er.

1.4.2 Kunstig intelligens i samfunnet

Kunstig intelligens kan høres både komplisert og skremmende ut – og det er det jo også. Men grunnlaget for enhver kunstig intelligens er programkode. Denne koden kan være omfattende, og den kan også være slik laget at datamaskinen kan lære av erfaringer. Ofte er strukturen i dataprogrammer med kunstig intelligens laget for å etterligne hvordan hjernen fungerer, med «nevroner» som er koblet sammen. Disse «nevronene» tar imot og viderebringer signaler. Et signal inn kan for eksempel være et bilde. Dette signalet blir sendt gjennom et nettverk av kunstige nevroner, og signalet ut kan bli et tall som representerer hva som er avbildet på dette bildet, for eksempel en katt.

Vi kan også la input til denne kunstige hjernen være informasjon om et menneske, og output være sannsynligheten for at vedkommende er skyldig i en straffesak. Da kan en dommer få hjelp av en datamaskin til å avsi en dom. Men hva er en rettferdig kunstig intelligens? Er det greit at et menneske får økt sannsynlighet for å bli uskyldig dømt bare fordi vedkommende kan grupperes sammen med andre som er mer kriminelle enn gjennomsnittet?

Kommer den selvkjørende bilen til å stoppe bare for barn og eldre, men ikke rullestolbrukere og syklistene? Det høres kanskje rart ut, men denne oppførselen kan oppstå helt uten at noe menneske engang har tenkt på at dette er viktig å ta hensyn til når vi designer selvkjørende biler. Og hva med fordommene som utviklerne av den selvkjørende bilen har? Har det for eksempel noe å si at holdningene til bilister sammenlignet med fotgjengere varierer dramatisk mellom kulturer? Slike etiske spørsmål trenger vi mennesker for å besvare. Vi kan ikke etterlate maskinene til seg selv, og vi trenger mennesker som forstår hvordan maskinene fungerer og handler.

1.5 Hva er et dataprogram?

Et program er et sett med instruksjoner. Et vaskeprogram er et sett med instruksjoner som utføres av en vaskemaskin når en vaskemaskin kjører programmet. Et dataprogram er et sett med instruksjoner som utføres når en datamaskin kjører programmet.

En av de enkleste instruksjonene vi kan gi datamaskinen, er å skrive ut «Hei, verden!». I Python skriver vi da:

```
print("Hei, verden!")
```

`print` er en funksjon som ber datamaskinen skrive noe på skjermen. Når vi kjører programmet ovenfor, får vi følgende utskrift:

```
Hei, verden!
```

Instruksjonen ovenfor gjør ikke egentlig noe nyttig. Et nyttigere sett med instruksjoner kan for eksempel være:

1. Sjekk temperaturen i Oslo og Bergen.
2. Om temperaturen i Oslo er lavere enn i Bergen, skriv «Det er kaldere i Oslo enn i Bergen». Ellers, skriv «Det er kaldere i Bergen enn i Oslo».

I Python-kode kunne dette blitt noe slikt som

```
from yr.libyr import Yr

weather_oslo = Yr(location_name='Norge/Oslo/Oslo/Oslo')
temperatur_oslo = float(weather_oslo.now()["temperature"]["@value"])

weather_bergen = Yr(location_name='Norge/Hordaland/Bergen/Bergen')
temperatur_bergen =
    float(weather_bergen.now()["temperature"]["@value"])

if temperatur_oslo < temperatur_bergen:
    print("Det er kaldere i Oslo enn i Bergen")
else:
    print("Det er kaldere i Bergen enn i Oslo")
```

Vi skal ikke se på detaljene i dette programmet nå. Det som skjer, er at vi laster ned værddata fra Yr, og sammenligner dataene vi får for Oslo og Bergen.

1.5.1 Å programmere hverdagslivet

Datamaskiner er *veldig* nøye med at vi gir den presise instruksjoner. Dersom vi i eksemplet ovenfor hadde glemt kolon til høyre i linja som starter med «if», ville hele programmet krasjet. Dessuten hentet vi været i Bergen med