Oppgavesett for Python

Petter Mæhlum*

Januar 2019

1 List comprehension

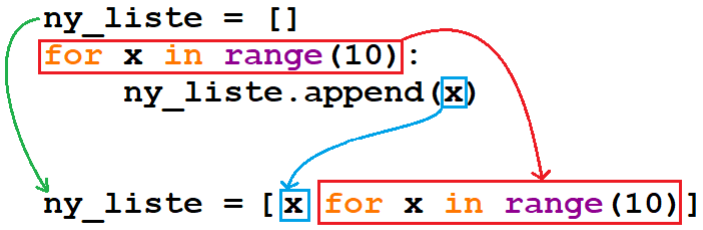
Dette er ment som en rask innføring i list comprehension, med tilhørende oppgaver. List comprehension er en spesiell type syntaks i Python som lar oss lage lister med variert innhold, uten å måtte ty til eksplisitte løkker over ei (tom) liste. I tillegg til å være plassbesparende kan det i enkelte tilfeller også være raskere med list comprehension, og vi kan også definere generatorer og andre nyttige strukturer.

Si at vi vil lage ei liste med alle heltall fra og med 0 til og med 9. Vi kunne skrevet:

```
ny_liste = []
for x in range(10):
    ny_liste.append(x)
```

Med list comprehension kan vi skrive det mer kompakt: [1]

```
ny_liste = [x for x in range(10)]
```



*Ta gjerne kontakt på pettemae@student.matnat.uio.no hvis du oppdager feil eller mangler.

[1] I dette tilfellet kunne vi selvsagt skrevet noe som list(range(10), men poenget her er å ha et enkelt eksempel.

1.1 Enkle lister og transformasjoner

Ofte har vi bruk for å kunne ta hvert element i ei gammel liste og gjøre dem om til ei ny liste, gitt en viss funksjon eller metode. Si vi har ei liste hvor ordene har blandede store og små bokstaver, men vi vil at de bare skal være små.

Vi definerer ei liste med strenger.

```
gammel_liste = ["Eple","bAnan", "kaffekVern","OSTepop"]
```

For å lage ei ny liste hvor hvert ord bare har små bokstaver, kan vi bruke list comprehension med et kall på streng-metode .lower() på x.

```
ny_liste = [x.lower() for x in gammel_liste]
print(ny_liste)
>>>['eple', 'banan', 'kaffekvern', 'ostepop']
```

Dette gjelder alle mulige funksjoner og operatorer. Noen andre korte eksempler:

```
tall = [1,2,3,4,5,6,7]
kvadrater = [x * x for x in tall]
print(kvadrater)
>>>[1, 4, 9, 16, 25, 36, 49]
broeker = [1/x for x in tall]
print(broeker)
>>>[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666,
0.14285714285714285]
```

1.2 Automatisk utpakking av lister

Det kan være veldig nyttig å pakke ut lister, altså å ta strukturer som er nøstede, og gjøre dem om til ei liste som ikke er nøsta. En måte å gjøre dette på er å spesifisere hvilket element man vil ha tak i.

I dette eksempelet vil vi bare ha ordformene, ikke ordklassene fra ei liste med tupler:

```
gammel_liste = [("jeg","PRON"),("elsker","V"),("IN2110","NN")]
ny_liste = [x for x,y in gammel_liste]
```

1.3 If-else med list comprehension

Vi kan bruke en if-test inne i list comprehension for å spesifisere noe. For eksempel, si at vi vil ha alle verdiene fra en liste hvis de er større enn 0.

```
gammel_liste = [1,91,4,-6,2,-9,15,65,1828,-53]
ny_liste = [x for x in gammel_liste if x >= 0]
```

```
print(ny_liste)
>>>[1, 91, 4, 2, 15, 65, 1828]
```

Vi kan også ha med *else*, men da må vi ta *if-else* i begynnelsen av uttrykket. La oss si at vi vil at alle tall som ikke er større enn 0 skal være 1 i den nye lista vår. Når vi har *if* foran, kan vi ikke utelate *else*.

```
gammel_liste = [1,91,4,-6,2,-9,15,65,1828,-53]
ny_liste = [x if x > 0 else 1 for x in gammel_liste]
print(ny_liste)
>>>[1, 91, 4, 1, 2, 1, 15, 65, 1828, 1]
```

En praktisk forskjell mellom å ha *if* bak, og å ha *if-else* foran, er at if-uttrykket bak fungerer som et filter, og vi kan derfor få ei liste som inneholder færre elementer enn den opprinnelige lista. Hvis vi har *if-else* foran derimot, vil den resulterende lista alltid ha like mange elementer som den opprinnelige lista.

1.4 Nøstede løkker

Vi kan også nøste list comprehension, som tilsvarer nøstede for-løkker. Det kan være nyttig å tenke på hvordan en nøsta for-løkke ser ut, og hvordan vi kan bruke dem både til å lage nøstede lister, eller lister som bare krever nøstede for-løkker, men som ikke selve er nøstede.

Det generelle mønsteret for en nøstet list comprehension ser slik ut. Vi kan tenke oss at elementet som legges til kommer først, men så følger løkkene etter hverandre slik de ville gjort i en for-løkke.

```
ny_liste = [subitem for item in gammel_liste for subitem in item]
```

Stemmer overens med:

```
ny_liste = []
for item in gammel_liste:
    for subitem in item:
        ny_liste.append(subitem)
```

For å lage ei nøsta liste må vi ha et list-comprehension-uttrykk inne i en list comprehension:

```
#En 4x4 matrise med bare 0:
ny_liste = [[0 for x in range(4)] for y in range(4)]
```

For et litt mer sammensatt eksempel kan vi lage en identitetsmatrise [2]

Med "vanlige" for-løkker kunne vi for eksempel skrevet fullt ut slik:

[2] En matrise der tallene på diagonalen er 1, og resten er 0. Se https://en.wikipedia.org/wiki/Identity$_matrix$

```
ny_liste = []
for x in range(4):
    midl = []
    for y in range(4):
        if x == y:
            midl.append(1)
        else:
            midl.append(0)
    ny_liste.append(midl)
```

Med list comprehension kan vi skrive:

```
ny_liste = [[1 if x == y else 0 for x in range(4)] for y in range(4)]
```

En annen veldig nyttig ting er å kunne flate ut nøstede strukturer. Hvis vi for eksempel har ei liste med setninger, hvor hver setning er ei liste med ord, så kan vi flate ut denne nøstede lista sånn at vi får ei endimensjonal liste med ord:

```
tekst = [['Hesten', 'rir', 'inn', 'i', 'solnedgangen', '.'],
['Fuglene', 'kvitrer', 'lystig', '.'], ['Vannet', 'sildrer',
'i', 'bekken', '.']]
#Vi kan skrive:
ny_liste = [word for setning in tekst for word in setning]
print(ny_liste)
>>>['Hesten', 'rir', 'inn', 'i', 'solnedgangen', '.', 'Fuglene',
'kvitrer', 'lystig', '.', 'Vannet', 'sildrer', 'i', 'bekken', '.']
```

1.5 Ordbøker

Vi kan også bruke dictionary comprehension på samme måte. Da bruker vi krøllparenteser, og vi må spesifisere både nøkkelverdi og innholdsverdi.

For eksempel, si at vil ha ei ny ordbok der nøkkelverdien er et ord, og innholdsverdien er antall bokstaver i ordet. Vi antar at det kun er unike elementer i lista vi tar utgangspunkt i, og skriver:

```
gammel_liste = ["melkesyre","salat","kaffegrut","kantarell"]
ny_ordbok = {x:len(x) for x in gammel_liste}
```

Vi kan også ta utgangspunkt i en nøstet struktur:

```
gammel_liste = [("bønne","bean"),("kaffe","coffee"),
("sentimentanalyse","sentiment analysis")]
ny_ordbok = {x:y for x,y in gammel_liste}
print(ny_ordbok)
>>>{'sentimentanalyse': 'sentiment analysis', 'kaffe': 'coffee', 'bønne': 'bean'}
```

OBS: Hvis vi bruker krøllparenteser men med bare én verdi, så får vi en mengde.

1.6 Generatorer

Generatorer er en struktur, et itererbart objekt, hvor hver verdi hentes når den trengs, men ikke før. Vi kan lage generatorer for høye verdier der lister kan bli for lange og gi problemer, og slik spare plass. Når vi kaller en for-løkke på en generator, blir hvert element generert etter hvert som det trengs. En ting som er viktig å merke seg, er at når vi først har kalt på et generator-objekt, kan vi ikke kalle på det samme objektet igjen.

Generatorer lages på samme måte som med lister og ordbøker, men med parenteser.

```
(x * 2 for x in range(100))
```

TIPS: Hvis du kommer over et generator-objekt som du helst skulle hatt i et annet format, så kan du gjøre det om til ei liste ved å bruke den innebygde funksjonen list(), slik:

```
generator = (x for x in range(10))
print(generator)
```

1.7 Oppsummering

liste:
[x for x in old]
ordbok:
k:v for k,v in old.items()
generator:
(x for x in old)
Vi kan kombinere flere av tingene, på samme måte som med vanlige for-løkker. Vi kan for eksempel bruke if-else med dictionary comprehension, osv.

1.8 Oppgaver

Oppgaver til del 1.1

Bruk list comprehension til å lage følgende lister:
a) Ei liste med kubene (**3) av alle tall for alle tall opp til og med 10.
b) Ei liste med alle heltall fra og med 0 til og med 15
c)Ei liste av tupler av heltall, der det andre tallet er kvadratet av det første, fra og med 0 til og med 10.
d) Gitt følgende liste:

```
rotete_data = ["kAke","pålegGsMasKIN","toFU","gARDinSTaNG"]
```

Lag ei liste med utgangspunkt i rotete_data hvor alle ordene er med store bokstaver.

Oppgaver til del 1.2

Gitt følgende liste med tupler:

```
gammel_liste = [('Kari', 'PN'), ('jager', 'VT'), ('dyret', 'N'),
('ved', 'P'), ('vannet', 'N'),('.','SYM')]
```

a) Lag ei liste med alle ordformene fra setningen.
b) Lag ei liste med alle ordklassetaggene fra setningen.
c) Lag ei liste av tupler hvor det første elementet er ordformen med små bokstaver, og den andre med store.

Oppgaver til del 1.3

Gitt samme liste som i oppgavene til del 1.2:
a) Lag ei liste av alle substantivene.
b) Lag ei liste av alle ordene som ikke er preposisjoner.
c) Lag ei liste av alle ordene som er lengre enn tre tegn.

Gitt følgende liste.

```
tall = [x for x in range(100)]
```

a) Lag ei liste av alle partallene.
b) Lag ei liste av alle tallene som har et kvadrat mellom 200 og 500.
c) Lag ei liste der alle partall erstattes av 2 og alle oddetall av 3.

Oppgaver til del 1.4

Gitt følgende tekst:

```
tekst = [['Kari', 'maler', 'huset', '.'], ['Det', 'er', 'lettere',
'overskyet,', 'men', 'varmt', '.'], ['Spurvene', 'leker',
'i', 'dammen', '.']]
```

a) Lag ei liste med alle ordene.
b) Lag ei liste med alle ordene som ikke er punktum.

Følgende oppgaver krever ikke bruk av lista fra a) og b):
c) Lag en 6x6-matrise fylt med 9-ere.
d) Lag en 5x5-matrise hvor hvert element er rad-indeksen ganger kolonne-indeksen (begynn på 0), med mindre produktet er et partall, da skal det være 0.

Oppgaver til del 1.5

Gitt følgende liste:

```
gammel_liste = ["vinter","iskrem","bade","grillings","ski"]
```

a) Lag ei ordbok hvor hvert ord er nøkkelverdien, og innholdsverdien er hvor mange ganger ordet inneholder bokstaven ”i”.
b) Lag ei ordbok hvor hvert ord er nøkkelverdien, og innholdsverdien er ei liste av tegnene i ordet (F.eks: ”kake” blir ’kake’:[’k’,’a’,’k’,’e’]).

Gitt følgende nøstede liste:

```
gammel_liste = [('Hvorfor', 'ADV'), ('gidder', 'VT'), ('vi', 'PRON'),
('dette', 'PRON'), ('?', 'SYM')]
```

c) Lag ei ordbok der ordformen er nøkkelverdi og ordklassen er innholdsverdi.
d) Gjør som i c), men utelat alle ord av ordklassen SYM.

Oppgaver til del 1.6

Lag en generator som produserer følgende: a) alle heltall fra og med 0 til og med 16
b) alle oddetall fra og med 1 til og med 27 Gitt følgende liste med tupler:

```
gammel_liste = [('Kari', 'NP'), ('ser', 'VT'), ('dyret', 'N'),
('ved', 'P'), ('vannet', 'N')]
```

c) Lag en generator som gir alle ordklassene.
d) Lag en generator som gir alle ordene som ikke er substantiver(N eller NP).