Lister og løkker

Johanna Haarseth
johannph på mattermost
johannph@uio.no på mail

Spørsmål please!

Skriv ned spørsmål du kommer på iløpet av denne videoen.

Send inn alle spørsmål du har om samlinger og løkker:

https://www.menti.com/19kuqpn7f1
eller gå inn på menti.com og bruk koden: 18 61 33 6

Jeg går gjennom spørsmål i zoom-timen mandag 14.15-16.00 og tirsdag 14.15-16.00 😊

Samlinger

Ordbøker

Kan sammenlignes med en telefonbok:

- key: navn, value: telefonummer {“Geir”:28394829, “Ahmed”:18292918}

Eller en fysisk norsk/engelsk ordbok:

- key: ord på norsk, value: engelsk oversettelse {“hei”:”hi”, “hade”:”bye”}

Eller en fysisk ordbok med ett språk:

- key: ord, value: beskrivelse av hva ordet betyr {“hei”:”tilrop eller hilsen”}

Ordbok: opprette nye ordbok

Vi kan enten opprette en tom ordbok eller opprette en ordbok med verdier:

```
tomOrdbok = {}


ordbok = {"hallo":"hi", "hade":"bye", "du":"you", "meg":"me"}


telefonbok = {"Geir":28374382, "Ahmed":82920304, "Rusmira":29485940}
```

Ordbok: hente ut value

Akkurat som i en telefonbok og en fysisk ordbok kan vi også i en python-ordbok slå opp på key! I en python-ordbok må key være unik.

```
navnPaaOrdbok[key]
```

>>evaluerer til key sin value

```
ordbok = {"hallo":"hi", "hade":"bye", "du":"you",
"meg":"me"}

halloPaaEngelsk = ordbok["hallo"]
```

>> Ordbok[hallo] evaluerer til “hi”

>> Verdien til variabelen halloPaaEngelsk blir da “hi”

Ordbok: legge inn key og value eller endre value

Samme syntax for å:

1. legge til nye nøkkel med verdi
2. endre verdien til en nøkkel vi allerede har i ordboken

```
# Legger til ny nøkkel, og ny verdi
ordbok["bønner"] = "prayers"
# Endrer verdien til en eksisterende
nøkkel
ordbok["bønner"] = "beans"
#Hva er verdien til ordbok["bønner"] nå?
```

```
telefonbok["Chris"] = 27383728
chrisNyttNummer = 23748948
telefonbok["Chris"] = chrisNyttNummer
#Hva er verdien til
telefonbok["Chris"] nå?
```

Mengder/sets

Mengder/sets er også pensum, ikke glem den!

```
tomMengde = set()
ikkeTomMengde = {"en", "en", "to", "tre"}
print(len(ikkeTomMengde))
```

Hva printes?

Liste: opprette en ny liste

Vi kan enten opprette en tom liste eller opprette en liste med elementer:

```
tomListe = []
ikkeTomListe = [1, 2, 3]
frukt = ["banan", "drue", "pære"]
deltakere = ["Frank", "Tabita", "Erika"]
```

Liste: hente ut et element

Hva printes?

```
deltakere = ["Frank", "Tabita", "Erika"]
print(deltakere[0])  >> Frank
print(deltakere[1])  >> Tabita
print(deltakere[-1])  >> Erika
print(deltakere[-2])   >> Tabita
print(deltakere[3])  >> ERROR
```

Test deg selv!

1. Sett videoen på pause 2. Finn penn og papir 3. Løs oppgavene før du fortsetter

En liste kan ha ulike typer elementer: boolean, tall, string, objekter og liste.
Hva printes her?

```
nostedListe = [[1, 2, 3],["en", "to", "tre"],[[1],[2, 3, 4]]]
print(len(nostedListe))          >> 3
print(len(nostedListe[0]))       >> 3
print(nostedListe[0])            >> [1, 2, 3]
print(nostedListe[1])            >> ["en", "to", "tre"]
print(len(nostedListe[2]))       >> 2
print(len(nostedListe[2][0]))    >> 1
print(nostedListe[2][1])         >> [2, 3, 4]
```

Ordbok vs mengde vs liste

Ordbok: ikke sortert, og man trenger unik nøkkel. Bra hvis man ofte har key og trenger value. Da er den mye raskere enn liste.

Mengde: ikke sortert, og ingen duplikater. Bra for å sjekke om noe finnes i mengden, eller for ting som ikke er sortert.

Liste: Sortert, man kan hente ut elementer basert på indeks.

Boolske uttrykk: for å forstå løkker

Hva er et boolsk uttrykk?

Et boolsk uttrykk evaluerer til enten `True` eller `False`

Alle disse evaluerer til en boolean:

```
"a" == "a"   >>True
1 != 2       >>True
1 == "1"     >>False
1 < 1        >>False
1 <= 1       >>True
```

“!=” leses “er ikke lik”

“<” leses “er mindre enn”

“<=” leses “er mindre enn eller lik”

Not

Dersom et uttrykk er usant, vil not()
av uttrykket være sant, og omvendt.

```
not(False) > True

not(True) > False

not(5 == 5) > False

not(5 > 1) > False

not(5 > 20) > True

not(5 == 1) > True
```

Sette sammen boolske uttrykk til et nytt boolsk uttrykk

Vi kan sette sammen flere boolske uttrykk, og vi skal lære to måter å gjøre det på:

1) And
2) Or

And

Begge uttrykkene må være sanne, både a og b!

```
False and False > False

False and True > False

True and True > True

(5 == 1) and (5 > 20)  > False

(5 == 1) and (5 > 1)  > False

(5 == 5) and (5 > 1)  > True
```

Or

Minst ett av uttrykkene må være sanne, enten a, eller b, eller begge!

```
False or False > False

False or True > True

True or True > True

(5 == 1) or (5 > 20)  > False

(5 == 1) or (5 > 1)  > True

(5 == 5) or (5 > 1)  > True
```

Test deg selv!

1. Sett videoen på pause 2. Finn penn og papir 3. Løs oppgavene før du fortsetterc

```
not("a" == "a") >> False
not(1 != 2 and 1 == "1") >> True
not(1 == "1")  >> True
not(1 < 1) >> True
not(1 <= 1) >> False
```

```
"a" == "a" or 5 > 20 >> True
not(1 != 2) and 5 > 20 >> False
not(1 == "1" and 5 > 20)  >> True
1 in [1, 2, 3] >> True
1 in [[1, 2, 3],[1],[2, 3]]  >> False
```

Løkker

While-løkke

Fortsetter så lenge uttrykket evaluerer til True!

Syntax:

```
while <boolsk uttrykk>:
    <repeterende kode>
```

Eksempel:

```
inputBruker = ""
while inputBruker != "q":
    print("Du har valgt å fortsette programmet")
    inputBruker = input("Vil du avbryte programmet så tast q")
```

For-løkker

2 typer for-løkker

1. for-løkke på en liste
   - går igjennom elementer i en liste
2. for-løkke med teller
   - gjør noe et gitt antall ganger med en teller
   - eks skrive ut tall fra 0 - 10

1. for-løkke på en samling

Syntax:

```
for <variabelnavn> in samling:
   <gjør noe>
```

Eksempel:

```
#   skriver ut ett og ett element.
liste = [5, 3, 7, 4, 1, "Hello", -1.3]
for element in liste:
   print(element)
```

2. for-løkke med teller

Syntax:

```
for <variabelnavn> in range(<antall ganger>):
   <gjør noe>
```

Eksempel:

```
# skriver ut tallene fra 0 til 9, så printer ut 10 tall.
for teller in range(10):
   print(teller)
```

While-løkke vs for-løkke

Bruk for-løkke hvis du vet hvor mange ganger noe skal skje! Dvs. at hvis du kan bruke for-løkke så bruk den.

Ellers bruker vi while-løkke. F.eks. hvis antall ganger noe skal skje avhenger av brukerinput.

To måter å iterere gjennom en liste

Med indeks (bruk denne hvis du trenger indeks):

```
liste = [5, 3, 7, 4, 1, "Hello", -1.3]
for indeks in range(len(liste)):
   print(liste[indeks])
```

Iterere gjennom listen:

```
liste = [5, 3, 7, 4, 1, "Hello", -1.3]
for element in liste:
   print(element)
```

Iterere gjennom nøstet liste

Med indeks (bruk denne hvis du trenger indeks):

```
nostedListe = [[1, 2, 3],["en", "to", "tre"],[[1],[2, 3, 4]]]
for indeksNostedListe in range(len(nostedListe)):
   for indeksListe in range(len(nostedListe[indeksNostedListe])):
       print(nostedListe[indeksNostedListe][indeksListe])
```

Iterere gjennom listen:

```
for liste in nostedListe:
   for element in liste:
       print(element)
```

Iterere gjennom ordbok

To måter:

```
telefonbok = {"Geir": 28374382, "Ahmed": 82920304, "Rusmira": 29485940}


for key in ordbok:
    value = ordbok[key]
    print(key, value)


for key, value in ordbok.items():
    print(key, value)
```

Iterer gjennom mengde

Men obs, hvor mange elementer er det i denne mengden?

```
mengde = {1, 2, 3, 3, 4, 4, 4}
for element in mengde:
   print(element)
```

Iterere gjennom string

Hva printes her?

```
minString = "Johanna sin string"

for element in minString:
   print(element)

minStringListe = minString.split()

for element in minStringListe:
   print(element)
```

Oppgaver

Oppgave 9 (15 poeng)

a) (7 poeng)

Skriv en funksjon **trimZeros (a)** som tar inn en liste med heltall og returnerer en liste med heltall hvor alle (eventuelle) nuller i starten og slutten av listen er fjernet. Dersom det er nuller inne i listen (dvs som har andre tall foran og bak seg) skal disse *ikke* fjernes. Gitt en liste [0,0,1,2,0,3,0,0,4,0] som argument, skal funksjonen altså returnere listen [1,2,0,3,0,0,4]. Effektiviteten av løsningen blir ikke tillagt vekt, formålet er kun at koden skal gi ønsket resultat.

Oppgave 5, 2014

Skriv en funksjon som har en liste med tall som parameter og som returnerer en verdi av type boolean. Funksjonen skal sjekke om alle verdiene i listen er i stigende rekkefølge (sortert). Dersom alle verdiene er i sortert rekkefølge skal funksjonen returnere true, ellers skal funksjonen returnere false. Du kan anta at alle verdiene i listen er ulike. Funksjonen trenger altså ikke ta hensyn til eventuelle like verdier.

Oppgave 6, 2014

- a) Skriv en funksjon med liste av tall som parameter og som returnerer en verdi av type int (heltall). Dersom alle verdiene i listen er like, skal metoden returnere denne verdien. Dersom ikke alle verdiene er like, skal den returnere tallet -1. Du kan anta at listen inneholder minst en verdi.
- b) Dersom du kaller funksjonen fra a) med en ikke-tom liste av tall og får -1 tilbake, kan du da være sikker på at ikke alle tallene i listen du sendte inn var like? Begrunn svaret.
-

Oppgave 9, 2014

- Dersom du kaster 3 terninger, er det 6*6*6=216 mulige utfall av antall øyne på de tre terningene (1-1-1, 1-1-2, …, 6-6-6). Bare i 6 av disse 216 utfallene er det samme antall øyne på alle de tre terningene (1-1-1, 2-2-2 osv). Skriv de nødvendige programlinjene for å printe ut alle kombinasjoner av antall øyne på de tre terningene på terminalen. Print til slutt ut hvor mange kombinasjoner som hadde minst 2 like terninger. Merk at 1-1-2 og 1-2-1 i denne sammenhengen er to ulike kombinasjoner, slik at begge skal printes ut og telle med i antall kombinasjoner med minst 2 like terninger.