Dictionaries og teksthåndtering

Ole Christian Lingjærde, Dept of Informatics, UiO

3. oktober 2019

Dagens agenda

- Første time: dictionaries og stringhåndtering
- Andre time: oppvarming til midtveiseksamen

Dictionaries

En dictionary er en samling med regler for å tilordne én verdi til en annen:

```
d = {0:6, 1:3, 2:7} # Regler: 0-->6, 1-->3 og 2-->7
print(d[1])         # Her skriver vi ut 3
```

Eksempel

Anta at vi ønsker to regler $0 \mapsto 10$ og $1 \mapsto 15$.

Implementasjon med funksjon

```
def f(x):
    if x == 0:
        return 10
    elif x==1:
        return 15
print(f(0))          # Resultat: skriver ut 10
```

Implementasjon med dictionary

```
d = {0: 10, 1: 15}
print(d[0])          # Resultat: skriver ut 10
```

Et eksempel til

Anta at vi ønsker disse reglene:

```
"Norway" --> "Oslo"
"Sweden" --> "Stockholm"
"France" --> "Paris"
```

Implementasjon med funksjon

```
def f(x):
    if x == "Norway":
        return "Oslo"
    elif x == "Sweden":
        return "Stockholm"
    elif x == "France":
        return "Paris"
```

Implementasjon med dictionary

```
d = {"Norway": "Oslo", "Sweden": "Stockholm", "France": "Paris"}
```

Legge til flere regler

Anta at vi ønsker å utvide funksjonen med en regel til:

```
"Norway" --> "Oslo"
"Sweden" --> "Stockholm"
"France" --> "Paris"
"Nepal"  --> "Kathmandu"      # NY REGEL
```

Implementasjon med funksjon

Vi må skrive en helt ny funksjon, denne gangen med fire regler.

Implementasjon med dictionary

```
# Vi kan enkelt utvide en dictionary med nye regler:
d["Nepal"] = "Kathmandu"
```

En dictionary gir indeksfrihet!

Indeksene er alltid `0, 1, ...., N-1` i lister (`N` = lengden på listen). Indeksene kan være hva som helst i en dictionary.

Eksempler:

```
# Parene (-2,6), (6,3), (3,7)
d = {-2:6, 6:3, 3:7}

# Parene ("Hamar","Norway"), ("Uppsala","Sweden")
d = {"Hamar":"Norway", "Uppsala":"Sweden"}

# Parene ("pi", 3.14), ("e", 2.718), ("g", 9.81)
d = {"pi":3.14, "e":2.718, "g":9.81}

# Parene ((0,0), "lowerleft") and ((1,1), "upperright")
d = {(0,0):"lowerleft", (1,1):"upperright"}
```

Det er noen restriksjoner på nøklene: de må være entydige og være av en konstant (immutable) data type (int, float, complex, string, tuple, ...).

Nøkler må være konstante objekter

Nøklene i en dictionary må være konstante (*immutable*) objekter, dvs objekter som ikke går an å endre.

Eksempler:

- Et listeobjekt er ikke konstant. Vi kan f.eks. endre *i*te verdien med `a[i] = new_value` uten å lage en helt ny liste.
- Et tuppelobjekt er konstant. Forsøk på å endre *i*te verdien gir feilmelding.

```
d = {1: 34, 2: 67, 3: 0}   # Nøkkelen er int
d = {1:6424, "X":64345}    # Nøkkelen er int eller string
d = {(0,0): 4, (1,-1): 5} # Nøkkelen er et tuppel
d = {[0,0]: 4, [-1,1]: 5} # FEIL: nøkkelen kan ikke være liste
```

Reglene i en dictionary

Nøklene er entydige

Vi kan ikke ha to regler med samme nøkkel i en dictionary. Hvis vi først definerer `d = {"Oslo": 3}` og så skriver `d["Oslo"] = 5` så overskrives den første regelen.

Reglene har ingen bestemt rekkefølge

En dictionary lagrer ikke reglene i en bestemt rekkefølge - tenk heller på en dictionary som en "sekk med regler".

Å lage en dictionary

For å lage en dictionary: initialisér + legg inn nye par etter behov.

Initialiser en ny dictionary

```
# Lag en tom dictionary:
d = {}

# Lag en dictionary med to par:
d = {"Oslo": 13, "London": 15.4}

# Lag en dictionary med to par (alternativ metode):
d = dict(Oslo=13, London=15.4)
```

Utvide en dictionary

```
# Anta at d er en dictionary

# Legg et nytt par til d:
d["Madrid"] = 26.0

# Legg en hel dictionary d2 til d:
d.update(d2)
```

Å fjerne noe fra en dictionary

Vi kan fjerne et par `(a,b)` fra en dictionary med `del d[a]`.

Eksempler:

```
In [1]: d = {"pi":3.14, "e":2.718, "g":9.81}
In [2]: del d["e"]  # Fjern paret ("e", 2.718)
In [3]: del d["pi"] # Fjern paret ("pi", 3.14)
In [4]: d = dict(Yes=1, No=0)
In [5]: del d["Maybe"]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call
<ipython-input-37-759d96d71ff6> in <module>()
----> 1 del d["Maybe"]

KeyError: "Maybe"
```

Er en bestemt nøkkel tilstede i en dictionary?

- Vi kan teste om en nøkkel er tilstede med `key in d`.
- Vi kan hente ut verdien med `d[key]` eller `d.get(key)`

Eksempel:

```
d = {"Berkeley": "US", "Cambridge": "UK"}

key = "Berkeley"

if key in d:
    print("Key: %s   Value: %s" % (key,d[key]))
else:
    print("%s is not found" % key)

value = d.get(key)
if value != None:
    print("Key: %s   Value: %s" % (key,value))
else:
    print("%s is not found" % key)
```

Er en bestemt nøkkel tilstede?

- Vi kan teste om en nøkkel er tilstede med `key in d`.
- Vi kan hente ut verdien med `d[key]` eller `d.get(key)`
- Med `d[key]` fås feilmelding når nøkkel ikke blir funnet
- Med `d.get(key)` fås `None` når nøkkel ikke blir funnet

Eksempel

```
d = {"Berkeley": "US", "Cambridge": "UK"}
key = "Berkeley"

# Alternativ A

if key in d:
    print("Key: %s   Value: %s" % (key,d[key]))
else:
    print("%s is not found" % key)

# Alternativ B

value = d.get(key)
if value != None:
    print("Key: %s   Value: %s" % (key,value))
else:
    print("%s is not found" % key)
```

Kjøring av program

```
Key: Berkeley   Value: US
Key: Berkeley   Value: US
```

Eksempel

```
d = {"Berkeley": "US", "Cambridge": "UK"}
key = "Harvard"

# Alternativ A

if key in d:
    print("Key: %s    Value: %s" % (key,d[key]))
else:
    print("%s is not found" % key)

# Alternativ B

value = d.get(key)
if value != None:
    print("Key: %s    Value: %s" % (key,value))
else:
    print("%s is not found" % key)
```

Kjøring av program

```
Harvard is not found
Harvard is not found
```

Å løpe gjennom elementene i en dictionary

Løpe over elementene i vilkårlig rekkefølge:

```
d = {-2:6, 6:3, 3:7}
for key in d:
    print("Key = %g and value = %g" % (key, d[key]))
```

Løpe over elementene i sortert nøkkelrekkefølge:

```
d = {-2:6, 6:3, 3:7}
for key in sorted(d):
    print("Key = %g and value = %g" % (key, d[key]))
```

Lage en liste over alle nøkler/verdier

Anta at vi har definert tabellen

```
d = {"Paris": 17.5, "London": 15.4, "Madrid": 26.0}
```

```
>>> d.keys()
<listiterator at 0x111b2a4d0>
>>> list(d.keys())
["Paris", "London", "Madrid"]
>>> d.values()
<listiterator at 0x111b2a710>
>>> list(d.values())
[17.5, 15.4, 26.0]
```

Eksempel: lese en fil med to kolonner

Mange datafiler består av kolonner med data, slik som denne hvor første kolonne er en entydig pasient-ID og andre kolonne er svaret på en blodprøve:

```
MB.0000         0.00096
MB.0002         0.24787
MB.0005         0.2779
MB.0006         0.29428
MB.0010         0.61225
...       ...
```

Vi kan lese filen over inn i en dictionary slik:

```
bloodtest = {}
infile = open("blood_test.txt", "r")
for line in infile:
    words = line.split()
    bloodtest[words[0]] = float(words[1])
infile.close()

# For å skrive ut blodprøve-svaret for pasient MB.0005:
print(bloodtest["MB.0005"])    # 0.00096
```

Eksempel: lese en fil med tre kolonner

Datafil:

```
Oslo        21.8    "Norway"
Bergen      17.6    "Norway"
London      18.1    "UK"
Berlin      19      "Germany"
Paris       23      "France"
Rome        26      "Italy"
Helsinki    17.8    "Finland"
```

Program:

```
infile = open("cityinfo.txt", "r")
data = {}
for line in infile:
    words = line.split()
    data[words[0]] = [float(words[1]), words[2]]
infile.close()

# For å skrive ut informasjonen om Paris:
print(data["Paris"])      # [23.0, ""France""]
print(data["Paris"][0])   # 23.0
print(data["Paris"][1])   # "France"
```

Lese en fil med navngitte kolonner

Datafilen `table.dat`:

```
Navn      Alder    Utdanningssted
Anne      10       Eiksmarka
Tom       15       Marienlyst
Vidar     18       Persbråten
Johanne   17       Wang
Silje     16       Eikeli
Per       19       UiO
```

Løsningsskisse

```
infile = open("table.dat", "r")
data = {}
headers = infile.readline().split()
for i in range(len(headers)):
        data[headers[i]] = []
for line in infile:
    words = line.split()
    for i in range(len(headers)):
            data[headers[i]].append(words[i])
infile.close()
```

Quiz 1

Hva skrives ut?

Spørsmål A

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[0])
```

Spørsmål B

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[d[0]])
```

Spørsmål C

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[-2]*d[2])
```

Svar på Quiz 1

Hva skrives ut?

Spørsmål A

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[0])          # 1
```

Spørsmål B

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[d[0]])       # 2
```

Spørsmål C

```
d = {-2:-1, -1:0, 0:1, 1:2, 2:-2}
print(d[-2]*d[2])    # 2
```

Quiz 2

Hva skrives ut?

Spørsmål A

```
table = {"age":[35,20], "name":["Anna","Peter"]}
for key in table:
    print("%s: %s" % (key,table[key]))
```

Spørsmål B

```
table = {"age":[35,20], "name":["Anna","Peter"]}
vals = list(table.values())
print(vals)
print(vals[0])
print(vals[0][0])
```

Spørsmål C

```
table = {"age":[35,20], "name":["Anna","Peter"]}
print(table["name"][1], table["age"][1])
```

Svar på Quiz 2

Spørsmål A

```
table = {"age":[35,20], "name":["Anna","Peter"]}
for key in table:
    print("%s: %s" % (key,table[key]))
# age: [35, 20]
# name: ["Anna", "Peter"]
```

Spørsmål B

```
table = {"age":[35,20], "name":["Anna","Peter"]}
vals = list(table.values())
print(vals)
print(vals[0])
print(vals[0][0])
# [[35, 20], ["Anna", "Peter"]]
# [35, 20]
# 35
```

Spørsmål C

```
table = {"age":[35,20], "name":["Anna","Peter"]}
print(table["name"][1], table["age"][1])
# ("Peter", 20)
```

Quiz 3

Hva blir innholdet i oppslagstabellen d?

Spørsmål A

```
d = {3:5, 6:7}
e = {4:6, 7:8}
d.update(e)
```

Spørsmål B

```
d = {3:5, 6:7}
e = {4:6, 7:8}
d.update(e)
d.update(e)
```

Spørsmål C

```
d = {6:100}
e = {6:6, 7:8}
d.update(e)
```

Svar på Quiz 3

Hva blir innholdet i oppslagstabellen d?

Spørsmål A

```
d = {3:5, 6:7}
e = {4:6, 7:8}
d.update(e)
# {3: 5, 4: 6, 6: 7, 7: 8}
```

Spørsmål B

```
d = {3:5, 6:7}
e = {4:6, 7:8}
d.update(e)
d.update(e)
# {3: 5, 4: 6, 6: 7, 7: 8}
```

Spørsmål C

```
d = {6:100}
e = {6:6, 7:8}
d.update(e)
# {6: 6, 7: 8}
```

Quiz 4

Filen "teledata.txt" inneholder info om mobilkunder:

```
Age     Income     Gender      Monthly calls    ID
45      720k       Female      46               A001
27      440k       Male        3                A002
17      0          Male        52               A006
24      60k        Female      18               A014
...     ...        ...         ...              ...
```

Kom med forslag til hvordan vi kan:

- lagre dataene som fem lister
- lagre dataene som én liste
- lagre dataene som en dictionary

Svar på Quiz 4

```
# Fem lister:
# ["Age", 45, 27, ...]
# ["Income", "720k", "440k", ...]
# ["Gender", "Female", "Male", ...]
# ["Monthly calls", 46, 3, ...]
# ["ID", "A001", "A002", ...]

# En liste:
# [["Age",45,27,...], ["Income","720k", "440k", ...], osv]

# En oppslagstabell:
# {"Age":[45,27,17,...], "Income":[720k,440k,0,...], osv}
```

Tekstbehandling - kjapp repetisjon

```
s = "min fil.txt"

# Splitte i enkeltord:
s.split()  # ['min', 'fil.txt']

# Splitte i enkeltord med selvvalgt skilletegn:
s.split(".")          # ['min fil', 'txt']
s.split(" fil")       # ['min', '.txt']

# Finne plassering av substreng:
s.index(".")          # 7

# Sjekke om substreng er tilstede:
"fil" in s            # True
"FIL" in s            # False

# Plukke ut ett tegn:
s[0]                  # 'm'
s[1]                  # 'i'
s[2]                  # 'n'
s[3]                  # ' '
```

Plukke ut substreng

```
s = "Dette er en tekststreng"

# Alt unntatt første tegn
s[1:]    # "ette er en tekststreng"

# Alt unntatt første og siste tegn
s[1:-1]  # "ette er en tekststren"

# Alt unntatt to første og to siste tegn
s[2:-2]  # "tte er en tekststre"

# Tegnene med indeks 2,3,4
s[2:5]   # "tte"

# Alt fra og med en substreng
s[s.index("tekst"):]  # "tekststreng"

# Fjern blanke foran og bak
s = "   A B C   "
s.strip()   # "A B C"
s.lstrip()  # "A B C   "
s.rstrip()  # "   A B C"
```

Slå sammen tekststrenger

```
a = ["I", "am", "happy"]

# Slå sammen listeelementer
s = "".join(a)          # "Iamhappy"

# Slå sammen listeelementer med blanke i mellom
s = " ".join(a)         # "I am happy"

# Slå sammen listeelementer med "--" i mellom
s = "--".join(a)        # "I--am--happy"
```

Bytt ut substreng

```
s = "Dette er en tekststreng"

# Erstatt alle blanke med "X"
t = s.replace(" ", "X")                  # 'DetteXerXenXtekststreng'

# Erstatt en substreng med en annen
t = s.replace("Dette", "Her")            # 'Her er en tekststreng'

# Erstatt alt foran "tekst" med noe annet
t = s.replace(s[:s.index("tekst")], "Ny ")       # 'Ny tekststreng'

# Erstatt alt fra og med "tekst" med noe annet
t = s.replace(s[s.index("tekst"):], "setning")   # 'Dette er en setnin
```

Ny linje i tekststrenger

Her er det forskjeller mellom operativsystemer!

```
# Unix/Linux/Mac:
s = "\n".join(["Line A", "Line B", "Line C"])
s.split("\n")

# Windows:
s = "\r\n".join(["Line A", "Line B", "Line C"])
s.split("\r\n")

# Alle operativsystemer:
s.splitlines()
```

Noen flere tekstfunksjoner

```
# Test om en string bare består av sifre
s = "314"
s.isdigit()    # True
s = "   314"
s.isdigit()    # False
s = "3.14"
s.isdigit()    # False

# Endre alt til små eller til store bokstaver
s = "ABC def"
s.lower()      # "abc def"
s.upper()      # "ABC DEF"

# Test om en string starter/slutter med en gitt string
s = "Dette er en string"
s.startswith("Dette er")    # True
s.endswith("Dette er")      # False
```

Eksempel

Anta at vi ønsker å lese en fil med følgende format:

```
(1.3,0)     (-1,2)     (3,-1.5)
(0,1)       (1,0)      (1,1)
(0,-0.01)   (10.5,-1)  (2.5,-2.5)
```

Algoritme:

1. Les én linje av gangen
2. For hver linje: splitt opp i ord
3. For hvert ord: fjern parenteser og splitt på komma

Implementasjon

```
infile = open("pairs.dat", "r")
pairs = []    # Tom liste som skal holde dataene vi leser
for line in infile:
    words = line.split()
    for w in words:
        w = w[1:-1]    # Fjern parenteser
        numbers = w.split(",")
        pair = (float(numbers[0]), float(numbers[1]))
        pairs.append(pair)
```

Resultat

Listen pairs

```
[(1.3, 0.0),
 (-1.0, 2.0),
 (3.0, -1.5),
 (0.0, 1.0),
 (1.0, 0.0),
 (1.0, 1.0),
 (0.0, -0.01),
 (10.5, -1.0),
 (2.5, -2.5)]
```

Oppgave 1

Hva skrives ut?

```
s = -2

for k in range(2, 5, 2):
    s += 2

print("s = %d" % s)
```

Svar på oppgave 1

Hva skrives ut?

```
s = -2                          # s: -2

for k in range(2, 5, 2):        # k: 2, 4
    s += 2                      # s: 0, 2

print("s = %d" % s)
```

Vi får skrevet ut:

```
s = 2
```

Oppgave 2

Hva skrives ut?

```
a = [8, 9, 10, 11]
b = a[1:-1]
c = a[-1:1]
d = a[1:3]
print(b, c, d)
```

Svar på oppgave 2

Hva skrives ut?

```
a = [8, 9, 10, 11]
b = a[1:-1]          # b: [9, 10]
c = a[-1:1]          # c: []
d = a[1:3]           # d: [9, 10]
print(b, c, d)
```

Vi får skrevet ut:

```
([9, 10], [], [9, 10])
```

Oppgave 3

Hva skrives ut?

```
for i in range(2, 5):
    print(i, end=' ')
    for j in range(i-1, i+1):
        if i != j:
            print(j, end=' ')
```

Svar på oppgave 3

```
for i in range(2, 5):
    print(i, end=' ')
    for j in range(i-1, i+1):
        if i != j:
            print(j, end=' ')
```

i	j	Utskrift
2	-	2
2	1	1
2	2	
3	-	3
3	2	2
3	3	
4	-	4
4	3	3
4	4	

Svar på oppgave 3

```
for i in range(2, 5):
    print(i, end=' ')
    for j in range(i-1, i+1):
        if i != j:
            print(j, end=' ')
```

i	j	Utskrift
2	-	2
2	1	1
2	2	
3	-	3
3	2	2
3	3	
4	-	4
4	3	3
4	4	

Dvs utskrift blir: 2 1 3 2 4 3

Oppgave 4

Hva skrives ut?

```
import numpy as np

def h(n=1):
    x = np.zeros(n+1, int)
    x[0] = 1
    x[1] = 1
    for i in range(2, n+1):
        x[i] = -x[i-1] + 2*x[i-2]
    return n, x[n]

print(h(n=3))
```

Svar på oppgave 4

```python
import numpy as np

def h(n=1):
    x = np.zeros(n+1, int)
    x[0] = 1
    x[1] = 1
    for i in range(2, n+1):
        x[i] = -x[i-1] + 2*x[i-2]
    return n, x[n]

print(h(n=3))                    # (3, 1)
```

```python
# Kallet h(n=3) utfører følgende kode:
n = 3
x = np.zeros(4, int)             # x: [0, 0, 0, 0]
x[0] = 1
x[1] = 1                         # x: [1, 1, 0, 0]
for i in range(2, 4):            # i: 2, 3
    x[i] = -x[i-1] + 2*x[i-2]    # x[2]: -1+2=1  x[3]: -1+2=1
return 3, x[3]
# Dermed returneres tuplet (3, 1)
```

Oppgave 5

Hva skrives ut?

```
def table(k):
    n = 5
    primelist = [2,3,5,7,11]
    print(primelist)
    d = primelist
    for i in range(1, k):
        for j in range(n-i):
            d[j] = abs(d[j+1]-d[j])
        print(d[:n-i])

table(2)
```

Svar på oppgave 5

Hva skrives ut?

```
def table(k):
    n = 5
    primelist = [2,3,5,7,11]
    print(primelist)
    d = primelist
    for i in range(1, k):
        for j in range(n-i):
            d[j] = abs(d[j+1]-d[j])
        print(d[:n-i])

table(2)
```

```
# Kallet table(2) utfører følgende:
k = 2
n = 5
primelist = [2,3,5,7,11]
print(primelist)  # UTSKRIFT: [2,3,5,7,11]
d = primelist
for i in range(1, 2):              # i: 1
    for j in range(n-i):           # j: 0, 1, 2, 3
        d[j] = abs(d[j+1]-d[j])    # 3-2, 5-3, 7-5, 11-7
    print(d[:n-i])                 # UTSKRIFT: [1,2,2,4]
```