1.25 Dictionaries og strings

Ole Christian Lingjærde, Dept of Informatics, UiO

4 oktober - 10 oktober 2021 (Del 2 av 2)

Denne ukens agenda

- Dictionaries (mandag)
- Oppgave A.14 og 5.14 i Langtangens bok (mandag)
- Tekststrenger (i dag)
- Oppvarming til midtveiseksamen (i dag)

Denne ukens agenda

- Dictionaries (mandag)
- Oppgave A.14 og 5.14 i Langtangens bok (mandag)
- Tekststrenger (i dag)
- Oppvarming til midtveiseksamen (i dag)

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: Ekstrahere tall i en tekstfil

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 parentesene rundt
        numbers = w.split(",")
        pair = (float(numbers[0]), float(numbers[1]))
        pairs.append(pair)
```

Resultat

```
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)]
```

Eksempel: Finne vanligst forekommende ord

Vi ønsker å lage et program som finner de vanligst forekommende ordene i en tekstfil.

Innledende strategi:

- Lage en tom liste `words` for å holde alle ordene
- Lese filen linjevis med `readline()` og finne enkeltordene med `split()`
- Legge ordene fortløpende inn i `words`

Eksempel: Finne vanligst forekommende ord

Problem: Hvordan unngå at spesialtegn og store/små bokstaver skaper problemer? Anta for eksempel at vi har tekstfilen

```
Solen skinner. Det er veldig varmt i solen
i dag. Hvis det ikke hadde vært for solen,
ville det ha vært veldig kaldt.
```

Her vil `split()` fange opp blant annet ordene

```
Solen
solen
solen,
```

og vi ønsker å telle disse som tre forekomster av samme ord.

Løsningsstrategi:

- Trekke ut bokstavene i teksten med `isalpha()`
- Konvertere til små bokstaver med `lower()`.

Eksempel: Finne vanligst forekommende ord

Hvordan teller vi opp antall forekomster av hvert ord?

Løsningsstrategi A:

- Innføre en ny liste `counts` som er like lang som `words` og som inneholder antall forekomster vi har sett til nå av hvert ord

Løsningsstrategi B:

- Erstatte listen `words` med en dictionary `wordcounts`
- Første gang vi ser et ord, legger vi det inn som en ny nøkkel i dictionaryen `wordcounts` med tilhørende verdi satt til 1
- Hvis samme ord dukker opp senere i teksten, søker vi det opp i dictionaryen og øker tilhørende verdi med 1.

Eksempel: Finne vanligst forekommende ord

Komplett program:

```
filename = input("File name? ")
file = open(filename)

# Read words and add to dictionary with count
wordcounts = {}
for line in file:
  words = line.split()
  for word in words:
    w = "".join([e for e in word if e.isalpha()])
    w = w.lower()
    if w in wordcounts:
      wordcounts[w] = wordcounts[w]+1
    else:
      wordcounts[w] = 1

# Put all words in list and sort on counts
words = list(wordcounts.keys())
words.sort(key=lambda x:wordcounts[x], reverse=True)

# Print 30 most frequent words with counts
for i in range(0,min(len(words),30)):
    print(f"{i+1:3d}. {words[i]:10s} {wordcounts[words[i]]}")
```

Oppgave 1

Hva skrives ut?

```
a = []

for k in range(2,5):
    a = a + [k,k+1]

print(a)
```

Oppgave 2

Hva skrives ut?

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

Oppgave 3

Hva skrives ut?

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

Oppgave 4

Hva skrives ut?

```
def f(x,y):
  return x-y

def g(x):
  return abs(x) - x

x = 4
y = 3
print(g(f(y,x)))
```

Oppgave 5

Hva skrives ut?

```
x = [1, 3, 6, 9]
m = list(range(1,len(x)))
y = [x[k]-x[k-1] for k in m]
print(y[1])
```

Oppgave 6

Hva skrives ut?

```
def cubes(n):
  return sum([k**3 for k in range(n+1)])

def test_cubes():
  expected = sum([0, 1, 8, 27])
  computed = cubes(4)
  tol = 1e-12
  success = abs(computed-expected) < tol
  assert success, "Testen feilet"

test_cubes()
```

Oppgave 7

Vi har en liste a med tall og programmet

```
s = 0
for i in range(1,len(a)):
  if a[i] < a[i-1]:
    s += 1
```

Fyll inn det som mangler i disse programmene for at de skal gi samme resultat, dvs gi samme verdi til s:

```
# Program A
s = 0
i = 1
while a[i] < a[i-1]:
  <fyll inn kode her>

# Program B
s = 0
i = 0
while i < len(a)-1:
  <fyll inn kode her>
```