Øving 1

INF621 - Høstsemesteret 2021

Sammendrag

Øvingsoppgavene er ikke obligatoriske, men vi anbefaler likevel at du gjør de og leverer de innen fristen — Den eneste måten å lære å programmere på er ved å programmere. Ved å gjøre oppgavene får du også testet deg selv og sjekket at du forstår begrepene. Du skal levere én zip-fil, `oving1.zip`, som inneholder filene `oppg1.py`–`oppg2.py`. For å komprimere en eller flere filer til en zip-fil høyreklikker du filene (i dette tilfellet `oppg1.py`–`oppg2.py`) i maskinens filnavigasjonsprogram og velger `Komprimer` eller `Send til` → `Komprimert mappe`. Frist: Torsdag 4. november kl 23:59

1 Erstatt (40%)

Svar leveres på fil med navn `oppg1.py`.

I denne oppgaven skal vi erstatte ord i en tekst ved hjelp av en ordbok (oppslagstabell, `dict`).

1.a
(10%) Først lager vi en funksjon `erstatt(tekst, ordbok)` som tar en streng og en ordbok som parameter. Funksjonen skal returnere en ny streng hvor ordene i tekst som er nøkler i ordbok byttes ut med verdien knyttet til nøkkelen. Dersom et ord ikke forekommer som nøkkel lar vi det stå. *Hint*: Bruk strengmetoden `split()`. Eksempel på kjøring fra konsollen:

```
In [1]: tekst = 'Joseph Robinette Biden'
In [2]: ordbok = {'Joseph': 'Joe',
...: 'Bernard': 'Bernie',
...: 'Robinette': 'R.'}
In [3]: erstatt(tekst, ordbok)
Out[3]: 'Joe R. Biden'
```

1.b
(10%) Vi trenger også en funksjon `les_ordbok()` som får inndata fra en bruker via tastaturet og som returnerer en ordbok tilsvarende den vi brukte i oppgaven over. Dersom brukeren prøverå erstatte et ord som er lagt til fra før, kan du gi en advarsel om dette. Eksempel på kjøring fra konsollen:

```
In [4]: les_ordbok()
Gi meg det neste ordet (avslutt med tom streng): Joseph
```

```
Hva skal Joseph erstattes med: Jooe
Vi erstatter Joseph med Jooe
Gi meg det neste ordet (avslutt med tom streng): Robinette
Hva skal Robinette erstattes med: R.
Vi erstatter Robinette med R.
Gi meg det neste ordet (avslutt med tom streng): Joseph
Tidligere ville du erstatte Joseph med Jooe.
Hva skal Joseph erstattes med: Joe
Vi erstatter Joseph med Joe
Gi meg det neste ordet (avslutt med tom streng):
Out[4]: {'Joseph': 'Joe', 'Robinette': 'R.'}
```

1.c

(20%) Til slutt i denne oppgaven lager vi et lite program oversett() som først lager en ordbok basert på input fra brukeren. Programmet skal deretter bruke denne ordboken til å erstatte ordene i forskjellige strenger (også gitt av brukeren). Eksempel på kjøring fra konsollen:

```
In [5]: oversett()
Gi meg det neste ordet (avslutt med tom streng): Joseph
Hva skal Joseph erstattes med: Joe
Vi erstatter Joseph med Joe
Gi meg det neste ordet (avslutt med tom streng): Robinette
Hva skal Robinette erstattes med: R.
Vi erstatter Robinette med R.
Gi meg det neste ordet (avslutt med tom streng):
Skriv en tekst som vi skal oversette (avslutt med tom streng):
Joseph Robinette Biden Jr.
Den nye teksten er: Joe R. Biden Jr.
Skriv en tekst som vi skal oversette (avslutt med tom streng):
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg1.py` legge til

```
if __name__ == '__main__':
    oversett()
```

2 Vareoversikt (60%)

Svar leveres på fil med navn `oppg2.py`.

I denne oppgaven skal vi lage et program som holder oversikt over varen som selges i en liten butikk. Vi antar at alle varene i butikken har unike navn.

2.a

(10%) Først lager vi en funksjon som heter `les_ny_vare()`. Denne funksjonen skal hente informasjon om en vare via tastaturet. Vi trenger informasjon om navn, pris og kategori. Denne informasjonen returneres i en ordbok som har nøkkelordene `'navn'`, `'pris'` og `'kategori'`. Du kan anta at brukeren alltid gir oss et heltall

som pris. Eksempel på kjøring fra konsollen:

```
In [6]: les_ny_vare()
Navn på vare: eple
Pris på vare: 14
Kategori: frukt
Out[6]: {'navn': 'eple', 'pris': 14, 'kategori': 'frukt'}
```

2.b

(10%) Nå skal vi lage en funksjon `les_utvalg()` som henter informasjon fra tastaturet om alle varene vi har på lager i butikken. Funksjonen skal returnere en liste av disse varene (en vare er fortsatt representert som en ordbok). Du kan igjen anta at brukeren alltid gir oss gyldig input. Eksempel på kjøring fra konsollen:

```
In [7]: les_utvalg()
Vil du legge til en ny vare (j/n)? j
Navn på vare: eple
Pris på vare: 100
Kategori: frukt
Vil du legge til en ny vare (j/n)? j
Navn på vare: kål
Pris på vare: 31
Kategori: grønt
Vil du legge til en ny vare (j/n)? j
Navn på vare: drue
Pris på vare: 40
Kategori: frukt
Vil du legge til en ny vare (j/n)? n
Out[7]:
[{'navn': 'eple', 'pris': 100, 'kategori': 'frukt'},
{'navn': 'kål', 'pris': 31, 'kategori': 'grønt'},
{'navn': 'drue', 'pris': 40, 'kategori': 'frukt'}]
```

2.c

(10%) Så langt har vi ikke lagt inn informasjon om hvor mange eksemplarer vi har av hver enkelt vare. Vi må derfor lage en funksjon `tell_varer(vareliste)` som tar inn en liste av varer (fortsatt representert som ordbøker) og oppdaterer denne slik at det nå står hvor mange eksemplarer vi har på lager. Du kan sette hvert vareantall til å være et tilfeldig tall fra og med 0 til og med 3. Eksempel på kjøring fra konsollen:

```
In [8]: vareliste = [
...: {'navn': 'eple', 'pris': 100, 'kategori': 'frukt'},
...: {'navn': 'kål', 'pris': 31, 'kategori': 'grønt'},
...: {'navn': 'drue', 'pris': 40, 'kategori': 'frukt'}]
In [9]: tell_varer(vareliste)
In [10]: vareliste
Out[10]:
[{'navn': 'eple', 'pris': 100, 'kategori': 'frukt', 'antall': 2},
{'navn': 'kål', 'pris': 31, 'kategori': 'grønt', 'antall': 0},
```

```
{'navn': 'drue', 'pris': 40, 'kategori': 'frukt', 'antall': 1}]
```

2.d

(10%) Det er tilbud på noen typer varer denne uken. Vi trenger derfor å lage en ny funksjon `sett_ned_pris(vareliste, kategori, prosent)` som oppdaterer ordbøkene i vareliste slik at prisene på alle varer av kategori kategori er satt ned med `prosent` %. *Hint*: Når en pris `x` settes ned med `p` % er den nye prisen `x - x*p/100`. Eksempel på kjøring fra konsollen:

```
In [11]: sett_ned_pris(vareliste, 'frukt', 10)
In [12]: vareliste
Out[12]:
[{'navn': 'eple', 'pris': 90.0, 'kategori': 'frukt', 'antall': 2},
{'navn': 'kål', 'pris': 31, 'kategori': 'grønt', 'antall': 0},
{'navn': 'drue', 'pris': 36.0, 'kategori': 'frukt', 'antall': 1}]
```

2.e

(10%) Vi trenger også en funksjon `selg_vare(varenavn, vareliste)` som tar et varenavn og en vareliste som parameter, og som returnerer prisen på varen. Om vi ikke har varen inne skal funksjonen returnere -1. Funksjonen skal også oppdatere varelisten slik at den reflekterer hvor mange eksemplar som gjenstår i butikken. Vi antar at vi bare selger ett eksemplar av varen om gangen. Eksempel på kjøring fra konsollen:

```
In [13]: selg_vare('eple',vareliste)
Out[13]: 90.0
In [14]: selg_vare('kål',vareliste)
Out[14]: -1
In [15]: vareliste
Out[15]:
[{'navn': 'eple', 'pris': 90.0, 'kategori': 'frukt', 'antall': 1},
{'navn': 'kål', 'pris': 31, 'kategori': 'grønt', 'antall': 0},
{'navn': 'drue', 'pris': 36.0, 'kategori': 'frukt', 'antall': 1}
```

2.f

(10%) Noen vil kjøpe flere varer i butikken. De sender derfor en bestillingsliste på e-post. Skriv en funksjon `selg_varer(handleliste, vareliste)` som tar inn to parametere: En liste med navn på varer og en vareliste tilsvarende de vi har sett i oppgaven over. Funksjonen skal returnere summen av prisen på alle varene vi har på lager som kunden vil kjøpe. Funksjonen må også oppdatere vareantallet i vareliste. Eksempel på kjøring fra konsollen:

```
In [16]: selg_varer(['eple', 'kål', 'drue', 'eple'], vareliste)
Out[16]: 126.0
In [17]: vareliste
Out[17]:
[{'navn': 'eple', 'pris': 90.0, 'kategori': 'frukt', 'antall': 0},
{'navn': 'kål', 'pris': 31, 'kategori': 'grønt', 'antall': 0},
{'navn': 'drue', 'pris': 36.0, 'kategori': 'frukt', 'antall': 0}]
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg2.py`

legge til

```
if __name__ == '__main__':
    vareliste = les_utvalg()
    tell_varer(vareliste)
    sett_ned_pris(vareliste, 'frukt', 10)
    selg_vare('eple',vareliste)
    selg_vare('kål',vareliste)
    selg_varer(['eple', 'kål', 'drue', 'eple'], vareliste)
    print(vareliste)
```