Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4110 IT Grunnkurs
Høst 2014

Øving 9

Alle teorispørsmål skal besvares og begrunnes. Alle oppgavene skal demonstreres til en studentassistent på sal. I oppgaver der du skriver programkode skal også denne vises fram. Lykke til!

1 Sparing

Hvis en sum A blir investert med en årlig nominell rente på r over k år, vil verdien V etter denne tiden være gitt ved formelen $V = A(1 + \frac{r}{n})^{nk}$, hvor n er antall terminer per år.

a) Lag en funksjon som regner ut V med en startsum på 1000 kroner og rente på 5% over 20 år. Regn med månedlige terminer.

b) Utvid forrige deloppgaven til å ta inn A, r og k som parametere i funksjonen.

c) Utvid den forrige deloppgaven slik at den lagrer den oppsparte summen for hvert år i en liste som returneres.

d) Lag et program som tar inn A, r og k fra brukeren og skriver ut resultatet i tabellform.

e) Lag et program som tar inn en ønsket sluttsum, og skriver ut hvor stor startbeløp du må ha (I hele tusenlapper) for å oppnå minst så mye (med 5% over 20 år). Bruk funksjonen fra deloppgave b) for å finne ut dette.

2 Ωsteviruset

Ostelageret til Tine benytter et smart system som mapper oster med lagerhylleplass. Ostene er indeksert på navn, og hver type ost er mappet til en tuple med hylleplasser hvor nøyaktig én ost av denne typen befinner seg [1].

Kodesnutt 1 viser en liten bit av Tine sin ostedatabase i form av en dictionary.

Indeksering av dictionaries er ganske likt indeksering av lister. Den største forskjellen er at indekser (nøkler) kan i tillegg til tall, også være andre datatyper. I denne oppgaven er nøklene strenger.

Kopier dictionarien i kodesnutt 1, og bruk den til å løse de følgende deloppgavene.

PS: I denne oppgaven trenger du ikke lage noen ekstra funksjoner. Bruk gjerne innebygde funksjoner/metoder som `dict.items` og `str.split` for å løse oppgaven.

[1] **Advarsel**: Ostelageret til tine er i virkeligheten ikke indeksert slik.

Kodesnutt 1: Tines ostelager

```
cheeses = {
  'cheddar':
    ('A235-4', 'A236-1', 'A236-2', 'A236-3', 'A236-5', 'C31-1', 'C31-2'),
  'mozarella':
    ('Q456-9', 'Q456-8', 'A234-5', 'Q457-1', 'Q457-2'),
  'gombost':
    ('ZLAFS55-4', 'ZLAFS55-9', 'GOMBOS-7', 'A236-4'),
  'geitost':
    ('SPAZ-1', 'SPAZ-3', 'EMACS45-0'),
  'port salut':
    ('B15-1', 'B15-2', 'B15-3', 'B15-4', 'B16-1', 'B16-2', 'B16-4'),
  'camembert':
    ('A243-1', 'A234-2', 'A234-3', 'A234-4', 'A235-1', 'A235-2', 'A235-3'),
  'ridder':
    ('GOMBOS-4', 'B16-3'),
}
```

a) Finn og skriv ut alle hylleplasser til osten "port salut".

b) Dessverre så har hyllene A234 til A235, B13 til B15, og C31 blitt infisert av et ostespisende virus! Finn og skriv ut alle typer ost som potensielt er smittet av viruset.

c) Tross osteviruset, ønsker Tine fortsatt å selge ost til det sultne norske folk. Finn alle typer ost der ingen individ er smittet av viruset og skriv ut resultatet på formen `<hylleplass> <ostetype>`.

3 Bursdagsdatabasen

Vemund, som er en smule glemsk når det gjelder bursdager, ønsker å lage en stor database med bursdagene til alle vennene sine. For å skaffe seg et godt utgangspunkt har han etterspurt og fått tilsendt følgende database (i form av en python-dictionary) fra IDIs seksjon for "data and information management", som har det statlige ansvaret for lagring av bursdager.

Kodesnutt 2

```
birthdays = {
  "22 nov": ["Lars", "Mathias"],
  "10 des": "Elle",
  "30 okt": ["Veronica", "Rune"],
  "12 jan": "Silje",
  "31 okt": "Willy",
  "8 jul": ["Brage", "Øystein"],
  "1 mar": "Nina"
}
```

Databasen består av datonøkler som peker til enten et navn eller en liste med navn som har bursdager den datoen.

Dessverre er ikke denne databasen særlig komplett, så Vemund ønsker å legge til nye bursdager etter hvert som han kommer på de. Han har laget følgende funksjon for å gjøre dette:

Kodesnutt 3

```
def add_birthday_to_date(date, name):
  birthdays[date].append(name)
```

Som f.eks kan kalles slik:

Kodesnutt 4

```
add_birthday_to_date("30 okt", "Gunnar")
```

Vemund møter derimot på et problem når han prøver å legge til et bursdagsbarn på datoer som ikke finnes, eller bare har ett bursdagsbarn fra før. Istedet for at funksjonen hans gjør som den skal, grynter den og spytter tilbake en exception (et unntak) på Vemund:

Kodesnutt 5

```
>>> add_birthday_to_date("12 jan", "Sindre")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "unntak.py", line 13, in add_birthday_to_date
    birthdays[date].append(name)
AttributeError: 'str' object has no attribute 'append'

>>> add_birthday_to_date("9 feb", "Lillian")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "unntak.py", line 13, in add_birthday_to_date
    birthdays[date].append(name)
KeyError: '9 feb'
```

Den første feilen kommer av at verdien tilsvarende nøkkelen "12 jan" ikke er en liste, men en enkel streng. Den andre feilen kommer av at nøkkelen "9 feb" ikke en gang finnes i databasen.

Vemund ønsker at funksjonen tar hensyn til disse grensetilfellene (streng isteden for liste, manglende nøkkel) og gjør som den skal likevel, men han er litt sløv i Python, og etterspør din hjelp til å løse problemet[2].

Oppgaven din er å løse dette problemet ved hjelp av Python sin støtte for unntakshåndtering der du benytter setningene `try` og `except`. Mao. må du modifiserere funksjonen slik at den virker med brukseksemplene som nettopp ble vist.

Vemund har også spesifisert at han er allergisk mot if-setninger, så denne oppgaven er du nødt til å løse uten noen form for kondisjonelle setninger bortsett fra `try`/`except`.

[2]Hvis du tilfeldigvis også heter Vemund så må du likevel gjøre denne oppgaven.

4 Tallforekomster i fil

I denne oppgaven skal du lese fra en fil `nummer.txt` som inneholder en liste med tall. Filen finner du på øvingssiden.

a) Lag funksjonen `number_of_lines(filename)` som returnerer antall linjer i filen med navn `filename`.

b) Lag funksjonen `number_frequency(filename)` som returnerer en dictionary der hver nøkkel er et tall, og verdien til en nøkkel er antall forekomster av det tallet i filen med navn `filename`.

c) Kall funksjonen `number_frequency` på `nummer.txt`, og for hvert tall i resultatet skriver du ut en linje med tallet sammen med antall forekomster, separart med et kolon.

Eksempel 1: Input

```
1
3
4
5
5
3
1
2
0
9
8
2
```

Eksempel 2: Dictionary (intern representasjon)

```
{ 0: 1, 1: 2, 2: 2, 3: 2, 4: 1, 5: 2, 8: 1, 9: 1 }
```

Eksempel 3: Output

```
0: 1
1: 2
2: 2
3: 2
4: 1
5: 2
8: 1
9: 1
```

PS: output trenger ikke å være sortert på noen måte.

5 (frivillig) Opptaksgrenser

I denne oppgaven skal vi lese inn en fil med poenggrensene (for 2011) fra Samordna Opptak, filen er på CSV (Comma separated values)-format, noe som betyr at hver linje er en liste med felter separert med komma. Tekstfelter er omsluttet av hermetegn (`"`). Første felt er studiets navn. Andre felt er poenggrensen, som er enten et tall, eller `"Alle"` dersom alle kom inn.

F.eks. `"NTNU 194459 Antikkens kultur","Alle"` sier at alle som søkte kom inn på dragvoll-studiet "Antikkens kultur" ved NTNU.

Hver funksjon i de følgendene deloppgavene tar data fra filen `poenggrenser_2011.csv`[3] som input. Det anbefales at du leser inn denne filen og lagrer innholdet en plass slik at du slipper å lese den på nytt i hver deloppgave.

a) Skriv en funksjon som finner ut hvor mange studier som tok inn alle søkere.

b) Skriv en funksjon som finner gjennomsnittlig opptaksgrense for NTNU. Ikke ta med studier som tok inn alle søkere.

c) Skriv en funksjon som finner studiet med lavest opptaksgrense (som IKKE tok inn alle søkere) og studiet med høyeste opptaksgrense. Funksjonen skal returnere en tuppel med de to verdiene.

[3] Denne filen finner du på øvingssiden.