NTNU

Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:
Python: Repetisjon

Professor Alf Inge Wang

Aktuelle tema i Python

- Todimensjonale lister og generering av lister
- Dictionaries
- Filbehanlding (tekstfiler og binærfiler)
- Unntak/Exception

To-dimensjonale lister

Kapittel 7.8

To-dimensjonale lister

- To-dimensjonale liste: liste som inneholder andre lister som elementer
  - Også kjent som nøstede lister
  - Vanlig å betrakte to-dimensjonale lister som om de har rekker og kolonner
  - Nyttig til å jobbe med flere datasett
- For å prosessere data i to-dimensjonale lister trenger man å bruke indekser
- Typisk brukes nøstede løkker til å prosessere dem

Lage to-dimensjonale lister (liste av lister)

```
students=[   ['Joe','Kim'],
             ['Sam','Sue'],
             ['Kelly','Chris']]
```

	Column 0	Column 1
Row 0	'Joe'	'Kim'
Row 1	'Sam'	'Sue'
Row 2	'Kelly'	'Chris'

NTNU
Det skapende universitet

Referere til verdier i 2d-lister

- Første indeks gir rekke og andre kollonne
- For å referere til rekke 0 skrives:
  `students[0]  *['Joe','Kim']`
- For å referere til rekke 2 skrives:
  `students[2]  *['Kelly','Chris']`
- For å hente ut eller endre 'Chris':
  `students[2][1]`
- For å finne lengde på dimensjonene:
  - Rekker: `len(students)`
  - Kollonner: `len(students[0])`

`students`

	Column 0	Column 1
Row 0	'Joe'	'Kim'
Row 1	'Sam'	'Sue'
Row 2	'Kelly'	'Chris'

Lage flerdimensjonale tabeller av vilkårlig størrelse

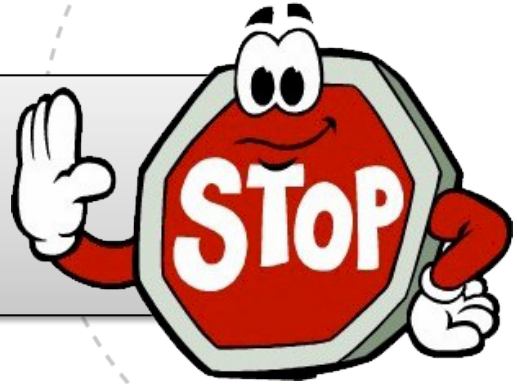
- Man kan også opprette en fler-dimensjonal tabell av en gitt størrelser uten å angi alle elementene:
  - Lage en 2-dimensjonal 10x10 matrise med 0er:

```
tabell_10x10 = [[0 for col in range(10)]
                   for row in range(10)]
```

  - Lage en 3-dimensjonal 3x3x3 matrise med sum av koordinater:

```
tabell_3d = [[[x+y+z for x in range(3)]
          for y in range(3)]
          for z in range(3)]
```

Oppgave: createTable

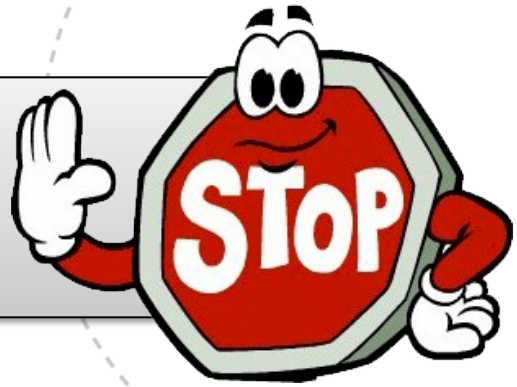


- Skriv funksjonen `createTable`, som har to inn-parametre `number` (heltall) og `size` (heltall).
- Funksjonen skal returnere en to-dimensjonal tabell av størrelse `size` x `size` fylt med tallet `number`.
- Eksempel på kjøring:

```
>>> createTable(0,3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

kode: matriser.py

Oppgave: fillTable



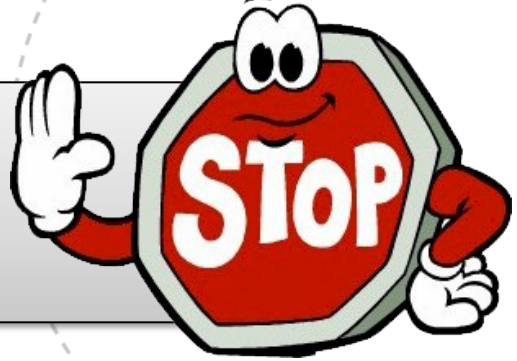
- Skriv funksjonen `fillTable` som har inn-parameteren `Table` som er en 2-dimensjonal tabell av ukjent størrelse.
- Funksjonen skal returnere tabellen fylt med tall fra 2-gangen.

2	4	6
8	10	12
14	16	18

NTNU
Det skapende universitet

kode: matriser.py

Oppgave: sumTable



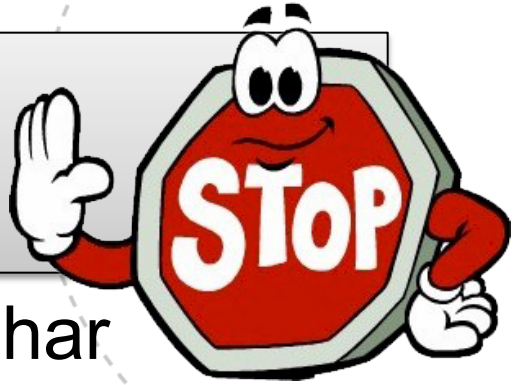
- Skriv funksjoner for å gjøre følgende `sumTable` som ha innparameteren `table` som er en tabell av vilkårlig størrelse.
- Funksjonen skal returnere summen av alle tallene i tabellen.
- Eksempel på kjøring:

```
>>> A
[[2, 4, 6], [8, 10, 12], [14, 16, 18]]
>>> sumTable(A)
90
```

NTNU
Det skapende universitet

kode: matriser.py

Oppgave: printTable



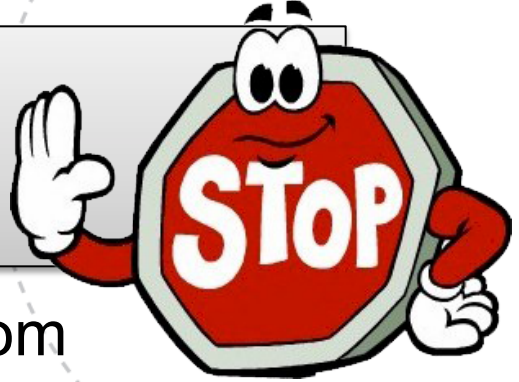
- Skriv funksjoner `printTable` som har innparameteren `Table`.
- Funksjonen skal skrive ut tabellen til skjerm, der hvert tall skal være midtstilt og det er plass til 5 tegn per tall. Eksempel på kjøring:

```
>>> A
[[2, 4, 6], [8, 10, 12], [14, 16, 18]]
>>> printTable(A)
  2    4    6
  8    10   12
  14   16   18
```

NTNU
Det skapende universitet

kode: matriser.py

Oppgave: pickRemoveNumber



- Skriv funksjoner `pickRemoveNumber` som har innparameteren `List`.
- Funksjonen skal returnere verdien til en tilfeldig plass i lista, samt fjerne dette innslaget fra lista.
- Kan bruke: `random.randint(start,stopp)` og `del List[index]`. Eksempel på kjøring:

```
>>> A
[2, 3, 4, 5]
>>> pickRemoveNumber(A)
3
>>> A
[2, 4, 5]
```

NTNU
Det skapende universitet

kode: matriser.py

Bruk av in...range i tabeller

- Hvis man trenger indeks for å hente ut innhold eller endre på innhold i en tabell, må man bruke in range i for-løkkene. Dvs. Hvis man trenger noe ala:
  - Tabell[y][x] må man bruke `in range` for å få riktig y og x!
- Hvis man bare skal hente ut verdier i en tabell, uavhengig av hvor de befinner seg i tabellen trenger man ikke bruke `in range`.

Dictionaries

Kapittel 9

Oppgave 4 – Programmering (40 %)

I denne oppgaven skal du programmere ulike funksjoner som skal brukes til å tilby highscore-funksjonalitet i et dataspill. Highscore-lista skal kunne ta vare på de 10 beste highscorene bestående av poengsummer og navn. Highscore-lista skal representeres som en dictionary der nøkkelen er plasseringen (1 til 10), og verdien er ei liste som består av navn og poengsum. Koden under viser et eksempel på opprettelse av en highscore-liste:

```
highscores = {}
highscores[1] = ['Vernon',100]
highscores[2] = ['Sirius',90]
highscores[3] = ['Severus',80]
…
highscores[10] = ['Albus',10]
```

Bruk funksjoner som defineres i andre deloppgavene hvis mulig. Du kan bruke funksjoner fra andre deloppgaver selv om deloppgaven ikke er løst.

Oppgave 4 a) (5 %)

Skriv funksjon `check_highscore` som tar inn en poengsum (*`points`*) og en highscore-liste (*`scores`*) og returnerer plassen poengsummen får på highscore-lista (fra plass 1 til 10). Merk at poengsummen må være høyere enn et innslag på lista for å kapre plassen. Hvis poengsummen ikke er høyere enn noen av innslagene i lista, skal verdien -1 returneres.

Oppgave 4 b) (5 %)

Skriv funksjonen `print_highscores` som tar inn en highscore-liste (*`scores`*) og skriver ut til skjerm alle highscores med plassering, navn og poengsum som vist under. Merk at plassering skal skrives ut høyrejustert med 2 tegns feltbredde, navn venstrejustert med 20 tegns feltbredde, og poeng høyrejustert med 5 tegn feltbredde.

```
 1 Albus                100
 2 Frank                 90
 3 Fleur                 80
 4 Sirius                70
 5 Vernon                60
 6 Ron                   50
 7 Harry                 40
 8 Minerva               30
 9 Hermine               20
10 Severus               10
```

Oppgave 4 c) (10%)

Skriv funksjonen add_highscore som tar inn en poengsum (`points`) og et navn (`name`) og en highscore-liste (`scores`), og legger til poengsum og navn i highscore-lista hvis poengsummen er høy nok. Merk at det er kun innslaget med laveste poengverdi som skal ut av lista når en ny score blir lagt til. Funksjonen skal returnere highscore-listen som kan være enten uendret eller endret .

Figuren under viser highscore-lista før og etter add_highscore(65,'Luna',highscores) er kjørt:

1	Albus	100
2	Fleur	90
3	Frank	80
4	Harry	70
5	Hermine	60
6	Minerva	50
7	Ron	40
8	Severus	30
9	Sirius	20
10	Vernon	10

```
add_highscore(65,'Luna',highscores)
```

1	Albus	100
2	Fleur	90
3	Frank	80
4	Harry	70
5	Luna	65
6	Hermine	60
7	Minerva	50
8	Ron	40
9	Severus	30
10	Sirius	20

Oppgave 4d) (10%)

Skriv funksjonen **`most_highscores`** som tar inn en highscore-liste **`(scores`**) og returnerer navnet på den person som har flest innslag på lista. Hvis det er flere med like mange innslag, skal funksjonen returnere navnet til spilleren som er lengst oppe på lista. Hvis lista inneholder kun 10 forskjellige navn, skal en tom streng returneres.

Oppgave 4e) (10%)

Skriv funksjonen **`new_highscorelist`** som returnerer en ny highscore-liste (dictionary) med poengsummer fra 100 ned til 10 (100, 90, 80...) der følgende ti navn skal plasseres tilfeldig i highscore-lista (merk at alle navn skal representeres i lista): Albus, Fleur, Frank, Harry, Hermine, Minerva, Ron, Severus, Sirius, and Vernon.

Filbehandling

Prosessen for filoperasjoner i Python

- Filoperasjoner i Python gjøres i tre steg:
  1. Fila åpnes: Etablerer en link mellom filvariabelen og informasjonen lagret på disken.
     - Filreferansen som peker til den fysiske fila på disk blir lagret i en variabel.
     - Alle operasjoner mot fila må bruke denne variabelen som filreferanse.
  2. Verdier leses fra og skrives til fila ved hjelp av filreferansen:
     - Lesing: Data lagret i fil leses inn og lagres i variabler.
     - Skriving: Data lagret i variabler skrives som data lagret i en fil.
  3. Fila stenges.
     - Etter at fila er stengt, kan man ikke lese eller skrive til fila.

Filhåndtering i Python

Filkommandoer	Forklaring
f = open(‘filnavn’)	Åpner ei fil, returnerer filreferanse
f = open(‘filnavn’,’tilgang’)	Åpner ei fil, med spesifisert tilgang. F.eks. ‘w’ åper ei fil for skriving (se neste side)
f.read()	Returnerer hele innholdet av fila
f.read(n)	Returnerer n karakterer av innholdet
f.readline()	Returnerer neste linje (før \n)
f.readlines()	Returnerer hele innholdet av fila som ei liste
f.write(s)	Skriver ***strengen*** s til fil
f.writelines(liste)	Skriver innholdet av liste av ***strenger*** til fil
f.seek(offset,fra_hvor)	Forflytter filpekeren (index) i fila
f.tell()	Returnerer posisjon til filpekeren i fila
f.close()	Stenger fila

f representerer variabelen som tar vare på filpekeren

Søk igjennom stor fil

- Skriv funksjonen *find_rate* som har tre input-parametere, *filname*, *check_rate*, og *acc*.
- Funksjonen skal søke igjennom en fil som inneholder vekslingsrate mellom USD og NOK sammen med dato, og skrive ut til skjerm alle datoer der vekslingsraten i fila matcher med *check_rate*. Fila (USD_NOK.txt) har følgende format:

  26.11.2013 6.1036

  25.11.2013 6.1236

  ...

- Parameteren *acc* brukes til å angi hvor mange siffer det avrundes til under sammenligning av vekslingsrater.

NTNU
Det skapende universitet

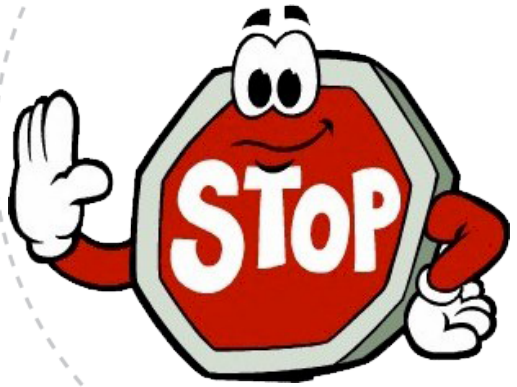
kode: find_rate.py

Unntak/ Exception

Exception / Unntak

- En exception er en feil som oppstår under kjøring som får programmet til å stoppe opp.
- Typiske feil som gir exception er:
  - Prøver å gjøre om tekststrenger til tall med strenger uten tall
  - Divisjon på 0
  - Prøver å åpne filer som ikke eksisterer
- En måte å unngå dette er å sjekke bruker-input.
- I Python kan du også bruke `try/exception` uttrykk for å unngå at programmet stopper opp under slike feil.

Exception...



- Utvid funksjonen *find_rate* slik funksjonen tåler klønete brukere:
  - Hvis man kaller *find_rate* med et filnavn til ei fil som ikke finnes eller som kan leses, så skal følgende skrives ut til skjermen: ”Feil: Finner ikke fila eller har ikke lesetilgang!”
  - Hvis man oppgir feil *check_rate* eller *acc*, f.eks. tekststrenger i stedet for tall, så akl følgende skrives ut til skjermen: ”Feil: Det er noe feil med argumentene!”)

kode: find_rate.py

DICTIONARY

Datastruktur: *Dictionaries*

- Dictionary er et objekt som lagrer en samling av data.
- Minner litt om lister men har klare forskjeller:
  - Defineres ved å bruke krøllparanteser { } (ALT+SHIFT 8/9 på Mac)
  - Kan bruke hva som helst som nøkkel(indeks) (ikke bare tall som i lister):
    - Tekst strenger, Heltall (men trenger ikke å være i rekkefølge), Flyttall, Sannhetsverdier (True eller False), En kombinasjon av de ovenfor

```
A = {}     # Tom dictionary
A['Kari'] = 92925492 # Oppretter et element
tlf={'Jo':73540000,'Per':92542312,'Else':54239212}
print(tlf['Per']) # Skriver ut verdien 92542312
```

Eksempel på bruk av Dictionaries

- Database for bibliotek med følgende funksjonalitet:
  - Legge til bok, vise bøker i database, slette bok, avslutte
- Datastruktur:
  - Nøkkel: ISBN
  - Innhold: Forfatter, Tittel, Forlag, Årstall
- Evt. Binær filbehandling hvis ønskelig.

kode: library.py