INF100 – Presentasjon: Tekstfiler, CSV og Mutasjon

Hva er en tekstfil

- En tekstfil inneholder ren tekst lagret som bytes.
- Lesbar for mennesker og programmer.
- Eksempler: .txt, .csv, .json, .py

Encoding

- Encoding = hvordan tegn oversettes til bytes.
- UTF-8 er standard i moderne systemer.
- Feil enkoding gir rare tegn (Ã¸ istedenfor ø).

```
from pathlib import Path
text = Path("data.txt").read_text(encoding="utf-8")
print(text)
```

Lese og skrive filer

- Lesing: Path('fil.txt').read_text()
- Skriving: Path('fil.txt').write_text('ny teskt')
- Current working directory = mappen programmet kjører fra
  - Absolute path: /home/haron/Documents/inf100/data.txt
  - Relative path:
    - ./cwd = Denne mappen
    - ../cwd = Opp enn mappe
    - ../../cwd = Opp to mapper
    - Osv.

Eksempel fra oppgave

- Kodesporing av asterisk.py

Hva er CSV

- CSV = Comma Separated Values – tabell i tekstform.
- Hver linje er en rad, hver verdi skilles med komma eller semikolon.
- Vanlig format for regneark og datautveksling.

Split og join

- Konverter mellom CSV-streng og 2D-liste.
- split() og join() brukes for å bygge eller lese teksttabeller.

```
text = "name,price\napple,10\nbanana,5"
rows = [line.split(',') for line in text.splitlines()]

name,price
apple,10
banana,5
```

csv-biblioteket

- Python har innebygd csv-modul som håndterer anførselstegn og separatorer.
- Anbefalt for ekte filer.

```
import csv
from pathlib import Path
with Path("varer.csv").open(encoding="utf-8") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

Eksempel fra oppgave

- Kodesporing av sales.py

Hva er mutasjon

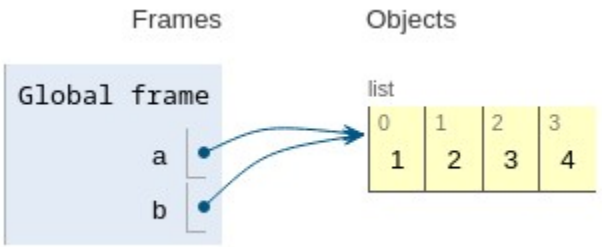
- Mutasjon = å endre et eksisterende objekt i minnet.
- Lister og dicts er muterbare, strenger og tupler er ikke.

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)  # [1, 2, 3, 4]
```

Lister i minnet

- Variabler peker til objekter i minnet.
- To variabler kan peke til samme liste (alias).
- Mutasjon endrer innholdet, ikke referansen.

```
1  a = [1, 2, 3]
2  b = a
3  b.append(4)
```



Destruktiv vs ikke-destruktiv

- Destruktiv funksjon endrer originalen.
- Ikke-destruktiv lager og returnerer en ny liste.

```
def double(lst):
    for i in range(len(lst)):
        lst[i] *= 2
def doubled(lst):
    return [x * 2 for x in lst]
```

Eksempel fra oppgave

- adults_and_children.py fra lab4