optimizing

Mål: Raskere, Mindre minne, Laste ned noe fra internett, Mindre batteribruk,

Optimere: minne

Python2 - range(10) gir en liste Python3 - range(10) git en generator En generator tar lite minne, men må bygges for hver gang den brukes. Generatorer kan ikke indekseres.

Når optimalisere

1. programmet fungerer og gir rett svar
    - Du har skrevet tester og programmet består alle testene
2. programmet er lett å forstå
    - dokumentajson
    - refakturering
    - kjente design patterns
    - naturlige variabelnavn
    - tester
    - opp koden

Er det hvert å optimalisere koden?

Det kan være mye jobb å optimalisere. Vi kan bruke verktøy for å vite om det er nødvendig.

ulemper ved optim

- Det tar tid
- Vanskelig å forstå
- Kan innføre nye bugs

profiling

...

Timing

```
t1 = time.perf_counter()
t2 = time.perf_counter()
```

Brudre kjøre koden flere ganger å finne et gjennomsnitt. Kan bruke:

```
%timeit slow_sum()
```

benchmarking

- Et standard problem som skal løses, og prøves med forskjellige algrotimer eller implementasjoner. For å se hvor god algoritmen eller implementasjonen er.

Profiling

- 90-10 regel 90% av tiden brukes i 10% av koden.
    - inne i doble og triple for loops.
- Vi burde finne disse 10% og fokusere på å optimalisere denne delen
- Det er ikke alltid lett å vite hvor falaskehalsene liger.

Example: The Dunkard's Walk

drunk.py profiler:

```
python -m cProfile -s "calles" drunk.py

-s: sorterer på "<sort_on>"

ncalls: The number of times the function is called
tottime: Time spent inside the function without sub-functions it calls
cumtime: Time spent inside the funtion and inside functions it calls
per call: The time divided by number of calls

python -m cProfile -o drunk.prof drunk.py
```

Kan også bruke; snakeviz for å visualisere.