NTNU

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Et større case

Terje Rydland - IDI/NTNU

Læringsmål og pensum

- Mål
  - Lære å lage større og sammensatte programmer
- Pensum
  - Kapitlene 1-9 og 12.

Programmering av systemer

- Vi skal nå se på hvordan vi kan sette sammen det vi har lært til å lage et lite "system" som består av flere deler.
- Koden til systemet inkluderer (stort sett alt dere har lært):
  - Bruk av flere moduler
  - Bruk av funksjoner
  - Bruk av for- og while-løkker
  - Bruk av if-setninger
  - Bruk av diverse datatyper og datastrukturer (bla. dictionary)
  - Bruk av filhåndtering

CASE: hybelregnskap

- Vi skal lage et system for å kunne føre hybelregnskap.
- Funksjonelle krav til systemet:
  - F1: Brukeren skal kunne legge inn utgifter
  - F2: Brukeren skal kunne legge inn inntekter
  - F3: Brukeren skal kunne se på alle utgifter og innteker og få vist balanse
  - F4: Brukeren skal kunne fjerne en utgift/inntekt
  - F5: Brukeren skal kunne laste inn regnskap fra disk
  - F6: Brukeren skal kunne lagre regnskap til disk
  - F7: Systemet skal tilby et tekstlig brukergrensesnitt

Hvordan analyserer vi problemet?

- Analysen kan svare på følgende spørsmål:
  - Hvilke data trenger vi å ta vare på?
  - Hvordan kan vi ta vare på dataene?
  - Hvilke datatyper eller datastrukturer kan vi bruke?
  - Hvordan skal vi dele opp systemet i moduler?
  - Hvordan er flyten i systemet?

Dataanalyse

- Dataene vi skal håndtere i dette systemet er utgifter og inntekter.
- Vi må da bestemme hva vi trenger å vite om utgifter og inntekter.
- Vi gjør det enkelt og går for følgende:
  - ID: Gjøre det mulig å skille innslaget fra de andre
  - Dato: Datoen for utgiften eller inntekten
  - Beløp: Hvor stor inntekten eller utgiften er i kr
  - Beskrivelse: Beskrivelse av inntekten og utgiften.

Datatype analyse

- Neste skritt er å se på hvilken data type de ulike elementene skal være:
  - ID: int (heltall)
  - Dato: streng: åååå-mm-dd (lettere å sortere)
  - Beløp: float (desimaltall)
  - Beskrivelse: streng (kan evt. begrense lengden på beskrivelsen)

Datastruktur analyse

- Neste skritt
  - se på hvilken datastruktur vi kan gruppere de ulike elementene på.
- Et krav var at vi skulle kunne fjerne utgifter eller inntekter.
  - Den mest fleksible løsningen for dette er dictionary med en liste av elementene vi skal ha med for hver id.
- En mulig datastruktur er da:
  - db= {ID:[dato, beløp, beskrivelse]}
- For å kunne ha unike innslag kan vi bare bruke:
  - ID som er et nummer som vi øker med en for hvert nytt innslag.

Hvilke moduler trenger vi?

- Alle systemer har en hovedmodul som starter programmet. Denne modulen kalles gjerne **`main.py`**
- Hovedmodulen skal ha den overordnede funksjonaliteten til systemet uten å gå i detaljer.
- Hovedmodulen importerer andre moduler som systemet består av og kaller funksjoner i de andre modulene.
- Hovedmodulen viser gjerne den overordnede prosessen (stegene i programmet) for systemet.

Hvilke moduler trenger vi?

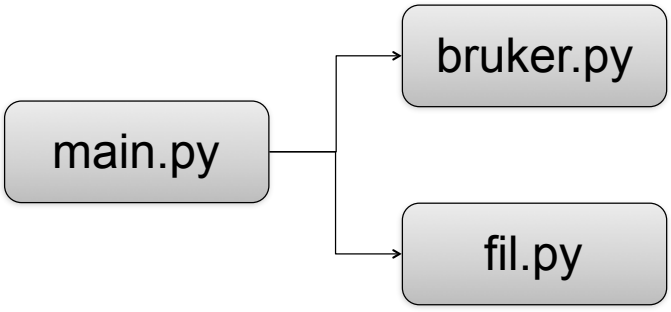
- I systemutvikling anbefales det å skille ut interaksjon med brukeren i en egen modul: **`bruker.py`**
- Denne modulen har som hovedansvar å tilby funksjoner som:
  - Henter input fra brukeren (fra tastaturet)
  - Skrive ut til skjerm
- Ved å skille ut brukergrensesnittet, kan man lett endre på utseende og interaksjon med bruker uten å endre på resten av systemet.

Hvilke moduler trenger vi?

- I systemutvikling anbefales det også å skille ut filhåndtering i en egen modul: `fil.py`
- Denne modulen har som hovedansvar å tilby funksjoner som:
  - Laste inn data fra fil til programmet
  - Lagre data til fil fra programmet
- Ved å skille ut filhåndtering, tar man høyde for eventuelle endringer i måten man kan lagre på eller for eksempel bruk av database i stedet for fil.

Hvilke moduler trenger vi?

- I systemutvikling anbefales det også å skille ut hjelpefunksjoner som ikke passer inn i andre moduler i en hjelpe-modul (util).
- Vi trenger ingen slik i dette systemet
- Systemet består nå altså av følgende moduler:



Hvordan er flyten i systemet?

- Her må vi prøve å tenke oss hvordan programmet skal fungere steg for steg.
- Det kan være en fordel å prøve å beskrive flyten i hovedmodulen først, uten å spesifisere flyten til funksjoner i andre moduler.
  - Dette kaller vi et top-down approach.
- Alternativt kan vi prøve å finne alle detaljer i alle moduler først og så sette det samme til slutt.
  - Dette kaller vi bottom-up approach.

Hvordan er flyten i systemet?

- Flyten i programmet kan skrives med pseudokode eller tegnes med flytdiagram.
- Her er en mulig pseudokode for hybelregnskap:
  - Importer moduler
  - Opprett nødvendige variabler
  - Vis en velkomstmelding til brukeren
  - Så lenge brukeren ikke vil avslutte:
    - La brukeren få et valg mellom ulike funksjoner i systemet
      - Utfør funksjonene i systemet basert på brukerens valg

Tekstbaserte brukergrensesnitt

- Tekstbaserte brukergrensesnitt brukes ikke så ofte lenger, men det finnes en del systemer som bruker det (typisk databasesystemer).
- Systemene må da opplyse brukeren om hvilke valg hun/han har.
- Brukeren skriver så inn en kommando for å gjøre et valg.
- Brukeren kan så bli spurt om å skrive inn mer data relatert til kommandoen han/hun valgte

Moduler

- Vi skal nå se på hvordan vi kan implementere (programmere) systemet ved hjelp av modulene:
  - **main.py**: Hovedmodul for initialisering, hovedflyt
  - **bruker.py**: Interaksjon med bruker (input/print)
  - **fil.py**: Innhenting og lagring av data fra/til disk.

main.py bruker.py fil.py

Oppsummering

- I utvikling av systemer bør man dele opp programmet i flere deler, for eksempel ved hjelp av moduler.
- Før man skriver kode for et system, bør man skrive opp kravene til systemet, samt lage en skisse for inndeling av moduler og flyten igjennom programmet.
- Man kan angripe problemet på to ulike måter:
  - Top-down: Ser på overordnet struktur først, så detaljer
  - Bottom-up: Ser på detaljer først og deretter overordnet struktur
- Skill ut brukergrensesnitt, filbehandling, hjelpefunksjoner og hovedprogram.