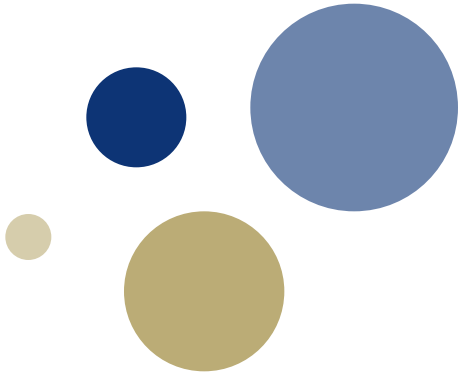
NTNU

Kunnskap for en bedre verden



# Lage større programmer
## (Python, relatert til teoridelen om ”Software Engineering”)

TDT 4110 IT Grunnkurs

Professor Guttorm Sindre

# Læringsmål og pensum

- Mål
  - Lære å lage større og sammensatte programmer
  - Kunne løse problemer hvor man må bruke mange deler av pensum samtidig
- Pensum
  - Pythonboka kap. 1-9, 12
  - Teorikapitlet om ”Software Engineering”

# Programmering av systemer

- Lage et lite “system” som består av flere deler.
- Bruke mye av det vi har lært:
  - Kontrollstrukturer
    - if, while, for
  - Funksjoner og moduler
  - Datatyper og datastrukturer
    - Tall, strenger, lister, tupler, mengder, dictionary
  - Filer
  - Unntaksbehandling

# CASE: bondesjakkspill

- Funksjonelle krav:
  - F1: Brukerne skal kunne spille partier
    - utføre lovlige trekk
    - se brettet oppdatert etter hvert trekk
  - F2: Programmet skal oppdage om noen har vunnet
  - F2: Brukerne skal kunne lagre et parti på fil
  - F3: Brukerne skal kunne hente fram parti fra fil
  - F4: Brukerne skal kunne spille videre på et uferdig parti
  - F5: Brukerne skal kunne se gjennom et parti trekk for trekk
  - F6: Bruker skal kunne få fram statistikk
    - Hvem har spilt, antall partier, seire og tap

# Andre krav

- **Utstyr:** I første omgang kun PC (ikke for eksempel mobil, nettbrett…)
- **Brukergrensesnitt:** I første omgang fornøyd med et ordinært tekstlig (konsoll) brukergrensesnitt, men programmet bør lages slik at det ikke er altfor vanskelig å bytte til et penere brukergrensesnitt senere.
- **Datalagring:** I første omgang fornøyd med å lagre på ordinære filer, men programmet bør struktureres slik at det i fremtiden ikke er altfor vanskelig å skifte til å lagre vha. database.
- **Spillmuligheter:** I første omgang menneske mot menneske, men bør være lett å utvide med at man spille mot maskinen.

# Hvordan analyserer vi problemet?

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

# Dataanalyse

- Dataene vi skal håndtere i dette systemet er
  - Navn på spillere
  - Partiene som spilles
    - Trekkene
    - Hvem som vant og tapte (spillernavn)
  - Brettposisjonen til en hver tid

# Datatyper og datastrukturer

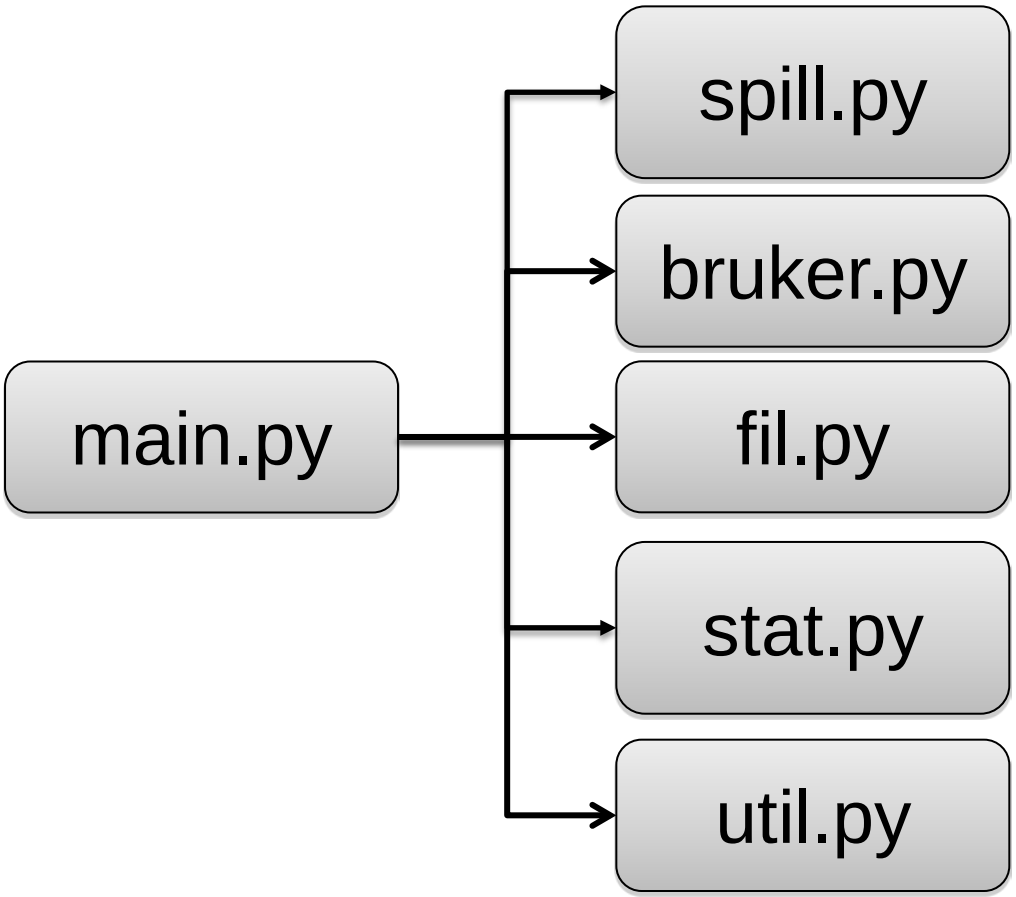
- Hvilke datatyper passer for informasjonen:
  - Spillernavn: streng
  - Parti: liste med to spillernavn og deretter alle trekkene
    - Trekk: tuppel (Rad, Kolonne)
      - Rad, Kolonne: heltall
      - Hvilket trekk som ble gjort (X, O)? (Unødvendig, alltid annet hvert trekk)
  - Brettposisjon: 2D-liste (liste av lister)
    - Hver «rute»: blank, X eller O
    - NB: det vi viser brukeren, trenger ikke være det samme som vi har i datastrukturen. Kan f.eks.
      - Vise på skjermen `'-', 'X', 'O'`
      - Lagre 0, 1, −1
    - Vise: det som er lettest for brukeren å forstå
    - Lagre: det som tar minst lagerplass / er lettest å prosessere

# Hvilke moduler trenger vi?

- Hovedmodul `main.py`
  - Starter programmet
  - Overordnet funksjonaliteten uten å gå i detaljer
  - importerer andre moduler, kaller funksjonene deres
  - viser overordnet prosess for systemet
- Modul for brukergrensesnittet `bruker.py`
  - All lesing fra tastatur og skriving til skjerm
  - Skal kun måtte endre her for å skifte UI
- Modul for filbehandling `fil.py`
  - All lesing fra og skriving til fil
  - Skal kun måtte endre her for å skifte lagringsmåte
- Moduler for å unngå for mye detaljer i `main.py`
  - Modul for spill-logikken `spill.py`
  - Modul for statistikk `stat.py`
  - Modul for hjelpefunksjoner `util.py`

# Hvilke moduler trenger vi?

- Systemet består nå altså av følgende moduler:



# Hvordan er flyten i systemet?

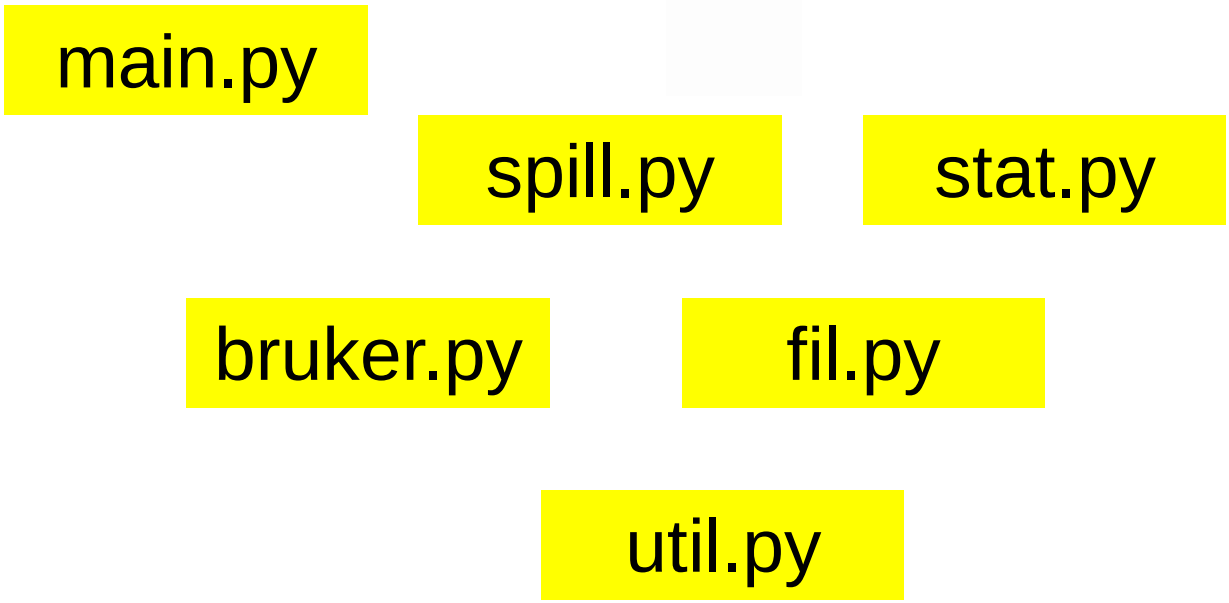
- Hovedmodulen:
  - Gi startinfo
  - Så lenge bruker ikke velger avslutt:
    - Motta menyvalg fra brukeren
    - Hvis valget er gyldig og mulig å utføre:
      - Utfør det
  - Gi sluttinfo

# Hvordan er flyten i systemet?

- Spille parti:
  - Initialiser brettet
    - Uten trekk hvis nytt parti
    - Med tidligere utførte trekk hvis fortsettelse av avbrutt parti
  - Så lenge ingen har vunnet og ingen har avbrutt
    - Les inn lovlig trekk
    - Utfør trekket
  - Hvis vunnet: lagre resultat

# Moduler

- Vi skal nå se på hvordan vi kan implementere (programmere) systemet ved hjelp av modulene



# Oppgave:

- Nytt krav: Hvis **'CPU'** skrives inn som navn på en av spillerne, skal denne spilles av maskinen

1. Finn ut **HVOR** i koden vi må gjøre endringer for å få til dette…

2. Prøv å skriv noen få kodelinjer så maskinen trekker
   - trenger ikke være gode trekk

# 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.