Oppgaver

a) Applikasjonsnivåprotokoll

Format

Å bruke ett json format virker som et bra utgangspunkt, siden da kan man sende meldinger i json, så lagre disse i en json-liste. Dette gjør også at man kan bruke ett python dictionary som "database".

Med oppbygning slik:

```
database = {
    100 : {
    "tittel": "Hva skal jeg ha til middag?",
    "alternativ": {
        "Pølse": 0, "Hamburger": 0, "PIzza": 0
        }
    }
}
```

(Først kommer problemID, så all tilhørende informasjon om selve problemet.)

Meldinger

Post problem: (valg 1. i `client.py`)

Klient oppgir tittel og alternativ, får tilbake problemID som ble generet da tjeneren ligger til problemet i databasen.

- Klient $\rightarrow$ Server:

  ```
  "kommando" : 1,
  "tittel" : "tittel tekst",
  "alternativ" : [
      "Alternativ 1",
      "Alternativ 2",
      "(...)"]
  ```

- Klient $\leftarrow$ Server:

  ```
  "melding" : "problem lagt til i database",
  "problemID" : problemID,
  ```

Spør etter problem: (valg 2. i `client.py`

Klient velger valg nr. 2 i prompt, server returnerer alle problemer som er lagret i databasen.

- Klient $\rightarrow$ Server:

  ```
  "kommando" : 2
  ```

- Klient $\leftarrow$ Server:

  ```
  {
      100 : {
          "tittel": "Hva skal jeg ha til middag?",
          "alternativ": {
              "Pølse": 0, "Hamburger": 0, "PIzza": 0
              }
          },
      99 : { (...) },
      88 : { (...) },
  }
  ```

Vis en problemformulering: (valg 3. i `client.py`

Klient velger valg 3 i prompt, så oppgir problemID, server returnerer samsvarende problem (hvis det finnes).

- Klient $\rightarrow$ Server:

```
"kommando" : 3,
"problemID" : problemID,
```

- Klient $\leftarrow$ Server:

```
"tittel" : "tittel tekst",
"alternativ" : [
    "Alternativ 1",
    "Alternativ 2",
    "(...)"]
```

Hvis ikke så returnerer server:

```
"melding" : "Ønsket problem finnes ikke"
```

Vis alternativer: (valg 4. i `client.py`

Klient gir problemID, så returnerer server alternativene for gitt problem.

- Klient $\rightarrow$ Server:

```
"kommando" : 4,
"problemID" : problemID
```

- Klient $\leftarrow$ Server:

```
"tittel" : "tittel tekst",
"alternativ" : [
    "Alternativ 1",
    "Alternativ 2",
    "(...)"]
```

Stem på alternativ: (valg 5. i `client.py`

Klient oppgir problemID, så oppgir tjener tittel+alternativ (via samme funksjon som i valg 3.), så stemmer klient på ønsket alternativ. Deretter så bekrefter tjener at den har mottatt stemmen.

- Klient $\rightarrow$ Server:

```
"kommando" : 5,
"problemID" : problemID,
```

- Klient $\leftarrow$ Server:

```
"tittel" : "tittel tekst",
"alternativ" : [
    "Alternativ 1",
    "Alternativ 2",
    "(...)"]
```

- Klient $\rightarrow$ Server:

```
"stemme" : STEMME (tall = 1..n)
```

- Klient $\leftarrow$ Server:

```
"melding" : "stemme mottatt",
"tittel" : "tittel tekst",
"alternativ" : [
    "Alternativ 1",
    "Alternativ 2",
    "(...)"]
```

Vis stemmer på et problem: (valg 6. i `client.py`

Klient oppgir valg 6 i prompt, så problemID. Tjener svarer med tittel på problemet og antall stemmer.

- Klient $\rightarrow$ Server:

```
 "kommando" : 6,
"problemID" : problemID
```

- Klient $\leftarrow$ Server:

```
 "tittel" : "tittel tekst",
"alternativ" : [
    "Alternativ 1" : antall_stemmer (1..n),
    "Alternativ 2" antall_stemmer (1..n),
    "(...)"]
```

b) Tilstandsløs vs tilstandsfull og TCP vs UDP

Tilstandsløs vs tilstandsfull

Applikasjonsprotokollen er tilstandsløs, da alle meldinger inneholder all informasjonen som trengs.

TCP vs UDP

TCP er best egnet da vi trenger pålitelig overføring av data, siden det er viktig at stemmer blir registrert riktig

c) Implementering klient-server-applikasjon NVDA i python

Se `server.py` og `client.py`